



**Trinity College Dublin**  
Coláiste na Tríonóide, Baile Átha Cliath  
The University of Dublin

**A Solid-powered decentralised social network for  
academics: An evaluation of key considerations for  
developing practical Solid-powered applications**

**Akashdeep Singh Lamba, B.Tech.**

**A Dissertation**

Presented to the University of Dublin, Trinity College  
in partial fulfilment of the requirements for the degree of

**Master of Science in Computer Science (Intelligent Systems)**

Supervisor: Prof. Declan O'Sullivan

Co-Supervisor: Dr. Jeremy Debattista

August 2019

# Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

---

Akashdeep Singh Lamba

August 12, 2019

## Permission to Lend and/or Copy

I, the undersigned, agree that Trinity College Library may lend or copy this thesis upon request.

---

Akashdeep Singh Lamba

August 12, 2019

This work is dedicated to Professor Seamus Lawless

# Acknowledgments

I'd like to take this opportunity to acknowledge Prof. Declan O'Sullivan and Dr. Jeremy Debattista for continued support and guidance throughout the process. Prof. Siobhán Clarke for valuable insights on Software Engineering and pointing me to ICSE which ultimately led me to some critical resources for my literature review. I also appreciate the efforts of the entire staff of Trinity for making sure the students have a pleasant learning experience.

I'd also like to extend my sincere appreciation to the entire Solid Community, especially Ruben Verborgh, Sir Tim Berners-Lee, Mitzi László, James Martin, Mark Hughes and all other individuals who took the time to answer my questions on various platforms, and for keeping Solid going.

Finally, I'd like to express my gratitude towards my family and friends for their constant encouragement and for helping me edit. I wouldn't have been able to complete this journey if it weren't for all the motivation from everyone.

AKASHDEEP SINGH LAMBA

*University of Dublin, Trinity College  
August 2019*

# **A Solid-powered decentralised social network for academics: An evaluation of key considerations for developing practical Solid-powered applications**

Akashdeep Singh Lamba, Master of Science in Computer Science  
University of Dublin, Trinity College, 2019

Supervisor: Prof. Declan O'Sullivan  
Co-Supervisor: Dr. Jeremy Debattista

Contemporary Social Networking applications are centralised, provide limited access control capabilities which are driven by dense privacy policies subject to change due to business considerations. Additionally, data ownership is impossible, interoperability between applications is in-feasible, and privacy of user data is not guaranteed.

Decentralisation is a potential solution to some of these problems, and Solid is a project that intends to make a fully decentralised read/write Semantic Web a reality. Solid is a set of standards and tools whose core tenets are decentralisation, complete and customisable access control, full data ownership, and reusable code.

With this study, we aimed to learn about Solid and to evaluate the developer experience. The objective was to produce a proof-of-concept Solid application, arrive at a set of guidelines for Solid development, and recommend improvements to Solid. A critical analysis of the developer experience revealed that Solid, though promising and important, is not yet ready for general uptake as the specification is not frozen. Solid development requires considerable prior knowledge of Linked Data and additional concepts such as Web Access Control, and the existing learning resources are inadequate. Thus, the developer experience is not mature enough to be scalable.

# Contents

<b>Acknowledgments</b>	<b>iv</b>
<b>Abstract</b>	<b>v</b>
<b>I Thesis</b>	<b>x</b>
<b>List of Tables</b>	<b>xi</b>
<b>List of Figures</b>	<b>xii</b>
<b>List of Abbreviations</b>	<b>xiv</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 Background . . . . .	2
1.2 Motivation . . . . .	2
1.3 Goals and Contributions . . . . .	3
1.4 Research Question . . . . .	4
1.5 Non-goals . . . . .	5
1.6 Research Areas . . . . .	5
1.7 Organisation of this document . . . . .	7
<b>Chapter 2 Related Work</b>	<b>9</b>
2.1 Literature review strategy . . . . .	9
2.2 State of privacy in centralised social networks . . . . .	11
2.3 Bootstrapping a developer in a new technology . . . . .	17
2.3.1 On-boarding and Developer Experience . . . . .	17

2.3.2	Documentation and Learning resources . . . . .	20
2.4	Decentralisation efforts . . . . .	23
2.5	State of the art of Solid . . . . .	30
2.5.1	Linked Data . . . . .	30
2.5.2	Building blocks of Solid . . . . .	31
2.5.3	Solid applications . . . . .	35
2.6	Summary and Next Chapter . . . . .	36
<b>Chapter 3 Research Methods</b>		<b>37</b>
3.1	Participant profile . . . . .	38
3.2	Study design . . . . .	38
3.3	Summary and Next Chapter . . . . .	39
<b>Chapter 4 Application Design and Implementation</b>		<b>40</b>
4.1	Development Environment . . . . .	40
4.2	Use-case Analysis . . . . .	41
4.3	Functional Requirements . . . . .	41
4.4	Design Decisions . . . . .	43
4.5	System design . . . . .	48
4.6	Data models . . . . .	49
4.7	Technical Implementation Details . . . . .	50
4.8	Summary and Next Chapter . . . . .	51
<b>Chapter 5 Evaluation</b>		<b>53</b>
5.1	Development Experience Evaluation . . . . .	53
5.1.1	DX1: Quality of learning resources . . . . .	53
5.1.2	DX2: Activity in the community . . . . .	55
5.1.3	DX3: Quality and quantity of tooling . . . . .	57
5.1.4	DX4: Stability of the platform . . . . .	60
5.1.5	DX5: Technical capabilities and features of the project . . . . .	60
5.2	Application Evaluation . . . . .	62
5.2.1	Performance . . . . .	62
5.2.2	Privacy by design . . . . .	62
5.2.3	DOSN classification . . . . .	64



5.3 Summary and Next Chapter . . . . .	64
<b>Chapter 6 Discussion and Conclusions</b>	<b>65</b>
6.1 Threats to Validity . . . . .	67
<b>Chapter 7 Future work</b>	<b>68</b>
<b>Bibliography</b>	<b>71</b>
<b>Appendix A Development Journal</b>	<b>95</b>
A.1 Background . . . . .	95
A.1.1 20-12-2018 . . . . .	95
A.2 First Contact . . . . .	96
A.2.1 20-12-2018 . . . . .	96
A.3 Initial hurdles . . . . .	97
A.3.1 23-12-2018 . . . . .	97
A.3.2 27-12-2018 . . . . .	98
A.3.3 18-01-2018 . . . . .	98
A.4 Read data from and Write data to Pod . . . . .	99
A.4.1 2-02-2019 . . . . .	99
A.4.2 2-05-2019 . . . . .	99
A.4.3 11-05-2019 . . . . .	99
A.4.4 12-05-2019 . . . . .	101
A.4.5 12-05-2019 . . . . .	101
A.4.6 13-05-2019 . . . . .	101
A.4.7 16-05-2019 . . . . .	102
A.4.8 17-05-2019 . . . . .	102
A.4.9 17-05-2019 . . . . .	104
A.5 Migration of Pods . . . . .	104
A.5.1 21-05-2019 . . . . .	104
A.6 Implementation: A very long post . . . . .	105
A.6.1 25-05-2019 . . . . .	105
A.6.2 30-05-2019 . . . . .	106
A.6.3 1-06-2019 . . . . .	106

A.6.4	3-06-2019 . . . . .	106
A.6.5	5-06-2019 . . . . .	107
A.6.6	7-06-2019 . . . . .	107
A.6.7	7-06-2019 . . . . .	107
A.6.8	8-06-2019 . . . . .	108
A.6.9	16-06-2019 . . . . .	109
A.6.10	17-06-2019 . . . . .	109
A.6.11	17-06-2019 . . . . .	110
A.6.12	22-06-2019 . . . . .	110
A.6.13	25-06-2019 . . . . .	111
A.6.14	8-07-2019 . . . . .	112
A.6.15	10-07-2019 . . . . .	113
A.6.16	10-07-2019 . . . . .	113
A.6.17	12-07-2019 . . . . .	114
A.7	Final session: almost finishing touches . . . . .	115
A.7.1	14-07-2019 . . . . .	115
A.7.2	17-07-2019 . . . . .	115
A.7.3	18-07-2019 . . . . .	115
<b>Appendix B Application Screenshots</b>		<b>117</b>
<b>II Getting Started with Solid</b>		<b>125</b>

**Part I**

**Thesis**

# List of Tables

2.1	Literature review search terms . . . . .	11
2.2	Framework for Developer Experience (DX) [1] . . . . .	17
2.3	Developer joining model [2] . . . . .	18
2.4	Comparison of decentralised online social networks . . . . .	26
4.1	Functionality implementation mapping with Solid features . . . . .	52
5.1	Evaluation of Solid tutorials . . . . .	54
5.2	Gitter messages on Solid and Deno rooms, as on 06-08-2019 . . . . .	57

# List of Figures

1.1	Research Areas . . . . .	6
2.1	A typical social network [3] . . . . .	15
2.2	A classification of common DOSN privacy models [4] . . . . .	24
2.3	The federation of DOSNs [5] . . . . .	28
2.4a	Example RDF encoded as Turtle . . . . .	31
2.4b	Graphical representation of example RDF . . . . .	31
4.1	Follow request life-cycle . . . . .	46
4.2	High-level overview of Feed aggregation . . . . .	46
4.3	System architecture of Albus . . . . .	48
4.4	User Interface of the homepage of Albus . . . . .	51
B.1	Homepage . . . . .	117
B.2	Homepage with Logout button revealed . . . . .	118
B.3	Homepage with Notifications expanded . . . . .	118
B.4	Login with WebID . . . . .	119
B.5	Pod Provider selection . . . . .	119
B.6	WebID-OIDC authentication: enter credentials . . . . .	120
B.7	Add application to trusted applications . . . . .	120
B.8	Consent to store and publish WebID . . . . .	121
B.9	Write post . . . . .	121
B.10	View and edit your own profile . . . . .	122
B.11	View another user’s profile (Read-only) . . . . .	122
B.12	View your followers and followees . . . . .	123

B.13 List all users on the network who have opted-in to be discovered . . . .	123
B.14 Alert message upon sending a follow request to another user . . . . .	124
B.15 Homepage from the point of view of Jon Snow . . . . .	124

# List of Abbreviations

OSN	Online Social Network
SNS	Social Networking Site
DOSN	Decentralised Online Social Network
OSS	Open-source software
FOSS	Free and Open source software
PII	Personally identifiable information
FIP	Fair Information Practice
UX	User Experience
DX	Developer Experience
TAM	Technology acceptance model
DHT	Distributed Hash Table
P2P	Peer-to-peer
IBBE	IdentityBased Broadcast Encryption
UOT	User Online Table
URI	Uniform Resource Identifier
OWL	Web Ontology Language
REST	Representational State Transfer
WebID	Web Identity and Discovery
OIDC	Open ID Connect
NSS	Node Solid Server
WAC	Web Access Control
LDP	Linked Data Platform
LDN	Linked Data Notifications
PR	Pull Request
TTL	Terse Triple Language
ACL	Access Control Lists
GUI	Graphical User Interface
UI	User Interface
SDK	Software Development Toolkit
API	Application Programming Interface
HOC	Higher-order component
FAQ	Frequently Asked Questions

# Chapter 1

## Introduction

In today's hyper-connected world, Internet has become a necessity and applications are increasingly relying on large-scale data collection, data mining, and machine learning to provide rich, personalised experiences to customers. Internet users, as a result, have become dependent on technologies such as personal assistants for repetitive tasks, and on social networking applications for staying in touch and accessing online content.

A side-effect of this symbiotic relationship is increasing *data privacy and security* concerns [3, 6, 7]. Data privacy, especially on applications relying on advertising for revenue, is an even bigger issue because of the incentive to mine more and more data [8]. Malpractices such as selling data to third-parties without consent notwithstanding, the appetite for mining private user data of contemporary web applications is a cause of concern simply because users themselves end up giving up rights of ownership to data about them without knowing the full extent of the data collection and its implications [9]. Since user data is stored on servers of applications that use it, users have little control over its movement, storage, safety, security, manipulation, and destruction [3, 10].

A major category of data-driven applications that have spurred data privacy concerns are Online Social Networks (OSNs). Most popular online social networks have a centralised architecture, i.e., they store large amounts of user information on their central databases, with users having little to no control over how that data is stored and whom it is shared with. The lack of transparency, coupled with tediously worded privacy statements leads to widespread data leakages and privacy violations leaving



users frustrated and helpless [8,9,11]. The repercussions can range from the relatively harmless such as email or phone spam to truly horrific criminal incidents like stalking.

Attempts have been made to address the centralised aspect of data storage and access control. One such effort is Solid [12], an open-source project born at MIT. Solid is aimed at enabling decentralised applications [13] with the help of Linked Data and Semantic Web [14], using a set of standardised technologies and conventions.

## 1.1 Background

The name *Solid* [12] is an acronym derived from SOcially LInked Data. It aims to transform the way we think of the design and use of Web-based applications. In technical terms, Solid is the specification of set of standards and tools that espouses a decentralised design philosophy for Web applications using Linked Data as the means to represent data and connections between data.

It was conceived in 2015 at MIT's CSAIL [15] as an academic project under the leadership of Sir Tim Berners-Lee [16]. It's widely believed that the current trend of applications that is characterised by a lack of control over data by users [9], and fosters data silos [17] is the wrong direction for the Web [18,19].

Solid can be considered a natural evolution of the read/write Web [20–24] and the read/write Semantic Web [20,23]. The idea that the Web is a readable and writable space where data is machine-readable [23,25,26] and is accompanied by meta-data for source tracing [27] and access control [28,29] is indeed central to realising the vision of Solid, and Solid specifies the infrastructure and the best practices for the implementation of decentralised web applications that also follow the principles of Linked Data and constitute the read/write Semantic Web.

## 1.2 Motivation

The nature of this research is exploratory. There are primarily 3 factors motivating this work.

Firstly, we arrive at *a set of guidelines and key considerations for new software developers being introduced to Solid*. These guidelines are to be based on time-tested

best practices in software development [30–33]. Solid introduces a paradigm shift in how most developers think about web application development. Some questions such as “how do we deal with customer data if they have the responsibility of making it available?” and “how do we build applications that do not store any user data on the server?” are worthy of being answered, and through this work, we will answer them. We also lay out explicitly the challenges inherent in this new approach to designing customer-facing applications and this work enables future work to address them.

The second motivation is to produce *a privacy-preserving decentralised social networking application for academics*. A social network or online social network (OSN) is an example of user-data-driven application. Over the years, social networks have evolved from computer-based education tools to friendship, dating, job-seeking networks with a plethora of private information and interactions attracting viral growth and commoditisation [34]. Through this study, we built a decentralised OSN that complies with the philosophy of Solid, thereby providing data access control and ownership to its users.

Finally, the study enables a more nuanced understanding of the challenges inherent to decentralised application design and deployment, with a focus on Solid’s contribution to addressing these issues.

## 1.3 Goals and Contributions

By the end of this study, we expect to have:

- a. An evaluation of the extent to which Solid can be used to build practical applications from the perspective of an application developer.
- b. A non-exhaustive list of challenges in application development using Solid, and potential approaches to overcome them.
- c. A proof-of-concept decentralised social network for academics to act as an example Solid application for new developers as well as to further Open Science initiatives.
- d. A tutorial-style guide for beginning application development in Solid focusing on real-world use-cases.

The study contributes to the Solid community, the web development community, the free and open source software (FOSS) community, and the Semantic Web/Linked Data communities. Translating our goals to the contributions, we have:

### C1 Solid developer guidelines and best practices

By carrying out a study of Solid development experience, we obtain actionable knowledge that can be applied to attract and train developers to use Solid, and to make the learning process smoother through a set of guidelines for development.

### C2 A Solid Application

The chief benefit of conducting this research is a significant contribution to the Solid community, by way of demonstrating implementation of a practical use-case. The proposed application will use the state of the art development techniques and tools available in the Solid community, and will demonstrate some of the salient capabilities of Solid.

### C3 Improvements to Solid

We discover a list of flaws and bottlenecks in addition to under-explored use-cases of Solid that will facilitate improvements to the documentation and developer resources.

### C4 An Open-science portal

The product of the research itself will be a contribution to the academic community as an Open-science focused decentralised social network.

This study therefore, has three artefacts:

**Artefact 1:** The application, called Albus, which will be available as open-source software (OSS) online.

**Artefact 2:** A Beginner's Guide to Solid, included in this document as Part II.

**Artefact 3:** The development journal, included in this document as Appendix A.

## 1.4 Research Question

The research question this study aims to answer is formulated as follows:

*“What are the key considerations while building practical applications powered by Solid?”*

## 1.5 Non-goals

This section is intended to clarify the non-goals of this study.

This study is not intended to produce a white-paper or a comprehensive reference to Solid. While we look at Solid from several perspectives in order to answer our research question, the study is by no means exhaustive, and an in-depth analysis of Solid is out of its scope. We only study the technical details of Solid to the extent required to build the application.

Finally, while we observe high standards of coding practices, maintainability and performance are not the focus of the application development track. The application developed is to serve as a proof-of-concept system for demonstrating capabilities, rather than a production-ready system.

## 1.6 Research Areas

This project deals with an interaction of four different research areas, as shown in Fig. 1.1.

1. Privacy in Online Social Networks

Solid is focused on building privacy-enhancing web applications through decentralisation, and this reflects in the goals of our study as well. We study the concept of privacy as well as the state of privacy in current OSNs. The understanding of privacy evolved through this study is used to design the application.

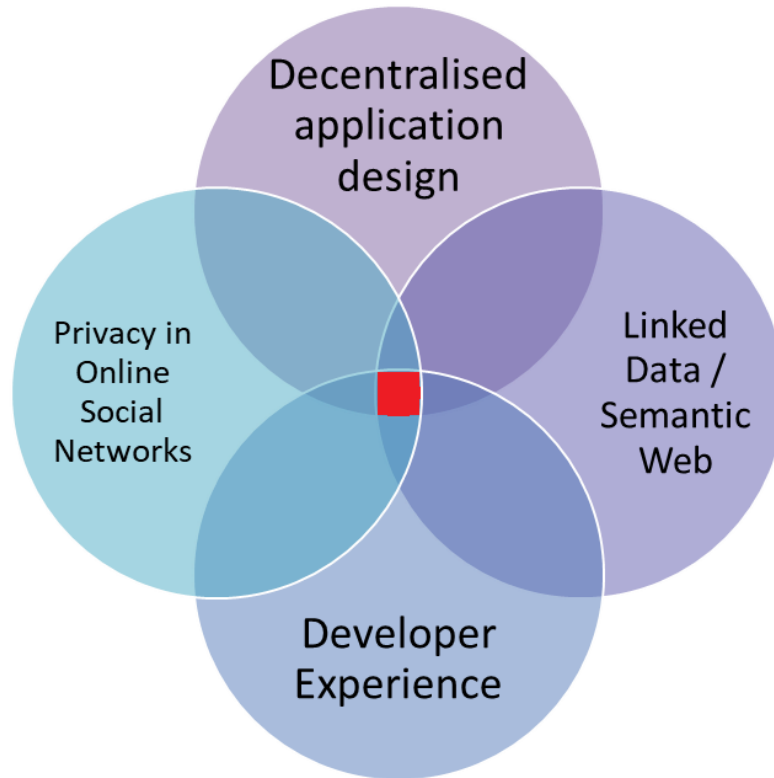
2. Developer experience

This study assesses the developer experience (DX) [35] of the current Solid ecosystem from the point of view of a software developer through exercise and makes observations as to how it can be improved.

The learning curve as a result of available developer on-boarding tools and documentation and its evolution is explored, and is used to build a “Getting Started” guide for building privacy-preserving Web applications using Solid.

3. Decentralised application design

Figure 1.1: Research Areas



Since it's based on Solid, decentralisation is a core characteristic of this project. This has strong implications on both the privacy-preserving aspect and the technical capability aspect of the proposed system, and we will be studying these implications in detail, both in theory and by practice. We evaluate our proposed application against these findings.

#### 4. Linked Data and the Semantic Web

Solid is built on the foundations of Linked Data and Semantic Web and as a result, the technical implementation details involve key Linked Data and Semantic Web concepts as part of the specification [36]. The application we build will use these concepts, and we study how it affects the design and implementation.

Fig. 1.1 is a Venn diagram representation of the logical relationship between these research areas viz-a-viz this project. It's helpful to think of this study partly as an

evaluation of Solid’s development experience. If you consider that Solid is the intersection of privacy in OSNs, decentralised application design, and Linked Data, then this study becomes an intersection of these 3 areas with development experience. Note that this project lies in the centre of the diagram, i.e., the intersection of the four research areas.

### 1.7 Organisation of this document

Throughout this document, the words *this study*, *this research*, *this project*, and *this work* are used interchangeably to refer to the work done towards the completion of this thesis, unless specified otherwise.

The words *the system*, *the application*, and *Albus* are used interchangeably to refer to the decentralised social networking application developed as a part of this study.

This document is organised into two parts. Part I is the primary thesis which represents the research work carried out and the findings. Part I is organised as follows:

Chapter 2 analyses the research areas and establishes the foundation of this work through a study of the state of the art in each research area and links it to this work.

Chapter 3 focuses on the approach used for this study.

Chapter 4 acts as a description of the design and lays out the technical implementation details of the application. A set of features and their corresponding technical approaches will be discussed.

Chapter 5 lists the observations from the development experience portion of this study, and critically analyses it with respect to the findings from related work. This chapter also presents an privacy and performance evaluation of the proposed application.

Chapter 6 presents the conclusions drawn from the findings from Chapter 5, and puts them in context of the goals laid down earlier in this chapter. We conclude with the answer to the research question and ends with the limitations of this research.

Chapter 7 briefly discusses the future work for the improvement of the platform, as well as for making the application more feature-complete.

Appendix A is a verbatim re-formatted reproduction of the development journal, which represents **Artefact 3** of this research and is used to base the assessments presented in Chapter 5 upon.

## 1.7. ORGANISATION OF THIS DOCUMENT

---

Appendix B shows screenshots of the graphical user interface (GUI) of the application that is built as a result of this study (**Artefact 1**).

Part II of this study is the a tutorial-style beginner's guide (**Artefact 2**) for building decentralised web applications using the Solid platform. This artefact is intended to serve as a model for producing more effective learning resources and documentation around Solid.

# Chapter 2

## Related Work

This chapter presents an in-depth review of the literature surrounding the research areas mentioned in Fig. 1.1. The first section sheds light on the method used for this literature review. The second section delves deep into the conceptualisation of privacy grounded in the context of OSNs. The third section explores existing research on software development pedagogy and developer experience with special focus on OSS. The fourth section presents a survey of decentralisation efforts and the challenges involved, especially those concerned with social networking. It also tabulates a comparison of the previously proposed or existing systems against commonly used techniques. The fifth section of this chapter begins with a brief overview of Linked Data, and then explores the state of the art of Solid, concentrating on the fundamental concepts and closing with an outline of existing Solid applications. The sixth section summarises this section.

### 2.1 Literature review strategy

The literature review for this chapter used the strategy of Snowballing [37]. A starting set of articles was obtained by searching a selected list of sources for a list of terms (Table 2.1), and then manually examining the results for relevant references. Neither list is exhaustive and other sources or terms were added on an ad-hoc basis as needed.

Solid is a rather new development, hence, the literature review also involved a considerable number of articles and blogs from the web. The list of initial sources is



## 2.1. LITERATURE REVIEW STRATEGY

---

as follows:

1. IEEE Xplore<sup>1</sup>
2. Stella Search - The Library of Trinity College Dublin: Stella Search<sup>2</sup>
3. ResearchGate<sup>3</sup>
4. Science Direct<sup>4</sup>
5. Google Scholar<sup>5</sup>
6. Solid.mit.edu web site<sup>6</sup>
7. Inrupt Inc. web site<sup>7,8</sup>
8. Ruben Verborgh's Publications<sup>9</sup>
9. Web articles and blogs

The search terms were constructed by examining the key concepts and keywords associated with Solid and the four research areas (Fig. 1.1). One of the challenges we faced while constructing the initial list for search terms was that Solid is an ambiguous search term, being famous for being associated with object-oriented design [38] and for a programming language used for the blockchain Ethereum [39]. We needed to maintain the specificity while not inadvertently rejecting relevant literature that did not include the exact words. This is where Snowballing [37] was very effective as we were able to find relevant literature by association.

---

<sup>1</sup><https://ieeexplore.ieee.org/Xplore/home.jsp> [Accessed: 07-08-2019]

<sup>2</sup><https://stella.catalogue.tcd.ie/iii/encore/?lang=eng> [Requires authentication] [Accessed: 07-08-2019]

<sup>3</sup><https://www.researchgate.net/> [Accessed: 07-08-2019]

<sup>4</sup><https://www.sciencedirect.com/> [Accessed: 07-08-2019]

<sup>5</sup><https://scholar.google.com/> [Accessed: 07-08-2019]

<sup>6</sup><https://solid.mit.edu/> [Accessed: 07-08-2019]

<sup>7</sup><https://solid.inrupt.com/> [Accessed: 07-08-2019]

<sup>8</sup><https://inrupt.com/> [Accessed: 07-08-2019]

<sup>9</sup><https://ruben.verborgh.org/blog/> [Accessed: 07-08-2019]

## 2.2. STATE OF PRIVACY IN CENTRALISED SOCIAL NETWORKS

---

privacy social networks
privacy concerns social networks
privacy violations social networks
solid
solid linked data
solid applications
learning software development
learning new software development frameworks
getting started with new software technology
software development tutorials for new developers
challenges learning new software framework
software developer pedagogy
read write web
solid tim berners-lee
social networking privacy
social networking privacy laws
factors adoption new technology
developer experience

Table 2.1: Literature review search terms

## 2.2 State of privacy in centralised social networks

This section will present the current practices in centralised social networks focusing on the privacy aspect, with the goal of arriving at a better understanding of privacy itself, how current OSNs interpret and implement it, and how a more evolved level of privacy can be delivered to users of OSNs.

Social Networking Sites (SNS) or Online Social Networks (OSN) are web-based applications that allow users to create public profiles [40], share them with other users within the bounds of the application [9, 41], and facilitate communication among the connected users [42]. OSNs may be designed specific to domains such as business-focused, common interests-based, or for purposes such as online dating [9, 43]. Examples of OSNs include LinkedIn<sup>10</sup> for professional networking, Instagram<sup>11</sup> for picture

---

<sup>10</sup><https://linkedin.com/> [Accessed: 04-07-2019]

<sup>11</sup><https://instagram.com/> [Accessed: 04-07-2019]

## 2.2. STATE OF PRIVACY IN CENTRALISED SOCIAL NETWORKS

---

and video sharing, Flickr<sup>12</sup> for Photography, Goodreads<sup>13</sup> and Vivilo<sup>14</sup> for books, Facebook<sup>15</sup> for general purpose networking, Twitter<sup>16</sup> for micro-blogging, and so on. In [6], Heravi et al. state that the main reasons people use OSNs are maintaining relationships, entertainment, building connections and networking, and looking up information. What all of the popular social networks have in common is that they are centralised. In other words, they rely on a central data server [3] that stores the profiles and related data of all users, and all users have to access this central server via network links in order to use the application.

As per Statista<sup>17</sup>, Facebook – the most popular OSN by far – has 2.32 Billion Monthly Active Users worldwide as of April 2019. This steady increase in the number of people using OSNs has been accompanied by an increase in exposure of users to concerns of data security and privacy. Facebook and Google, which are also the largest digital advertising platforms by revenue<sup>18</sup>, have been hit by major data privacy incidents in the past decade and a half, some of which scrutinised by Rubinstein and Good [11].

Last year, it was discovered that loopholes in Facebook’s privacy policies and the Apps tools had led to exposure of personally identifiable information (PII) of over 87 million Facebook users through a company Cambridge Analytica [44], which allegedly used the data for selling micro-targeted advertisements to political parties in order to influence voters in elections [10,45]. The revelation sparked widespread public outrage, led to Facebook’s CEO being summoned by the US Senate for a hearing, and caused material loss to the company’s stock [45]. It is well known that the *free-of-cost* nature

---

<sup>12</sup><https://flickr.com/> [Accessed: 04-07-2019]

<sup>13</sup><https://goodreads.com/> [Accessed: 04-07-2019]

<sup>14</sup><https://web.archive.org/web/20160220134002/https://www.vivilio.com/> [Accessed: 04-07-2019]

<sup>15</sup><https://facebook.com/> [Accessed: 04-07-2019]

<sup>16</sup><https://twitter.com/> [Accessed: 20-07-2019]

<sup>17</sup>See: Most famous social network sites 2019, by active users, <https://web.archive.org/web/20190704162437/https://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/> [Accessed: 04-07-2019]

<sup>18</sup>See: Google and Facebook devour the ad and data pie. Scraps for everyone else, <http://web.archive.org/web/20190804135401/https://digitalcontentnext.org/blog/2016/06/16/google-and-facebook-devour-the-ad-and-data-pie-scraps-for-everyone-else/> [Accessed: 04-07-2019]

of OSNs is fuelled by advertising [46], which increasingly involves large-scale data collection [47].

Privacy has been a complex and subjective [48] concept to characterise formally. It has been defined as “the right to control access to one’s personal information” [42, 49, 50], or “the right to be in control of how the information flows” [42, 51, 52]. Although the need and right to privacy has been undisputed, Fuchs, in his work [49, 53], argued that absolute privacy may not be pragmatic as it may lead to non-transparency and can be exploited by criminals and anti-social elements, urging a discussion of privacy from the point of view of protection against “capitalist exploitation by powerful organisations or individuals” at the expense of others. As per Renaud and Gálvez-Cruz [54], privacy allows individuals to set and maintain boundaries of interaction with the society and prevents intrusion.

Several studies have been conducted concerning various aspects of data privacy in centralised OSNs analysing causes, implications and possible solutions.

[7] discusses the information security and privacy challenges faced by businesses due to the collaboration and interactions with respect to social networking sites. The authors suggest that businesses that attempt to take advantage of the collaborative nature of OSNs by exploiting shared information and the connections between users are unable to exercise control over the sites or platforms as they are mostly 3rd party. The utility of OSNs lies in the presence of other people to connect and interact with owing to what the authors call “transient peer pressure”, so attempts to create internal silos are not successful due to the lack of network effect. Another challenge revealed by [7] is that enforcing corporate security practices becomes more complicated due to the various jurisdictions in which the data is hosted.

Pham et al. in their survey of privacy issues in OSNs [3] characterise a social network as one having 4 primary roles: *User*, *Social Network Provider*, *Analyst*, and *Adversary*. As can be seen in Fig. 2.1, the authors argue that privacy breaches can occur due to the activities of users, at the data storage servers, or even at the data opened up for analysis. The authors propose a classification of the privacy issues in four categories, namely, social graph publishing, OSN activities, sharing of media, cyber-physical systems, mobile systems, and other OSN applications such as dating. The authors review privacy preserving techniques like differential privacy [55, 56], pseudonym exchange [57, 58], negative survey [59–61], dummy-based method [62–64], and cryptog-

raphy [65–68]. However, the authors find that most of these techniques have caveats or assumptions and so do not generalise well in the real world. While the use of cryptography has been widespread and increasingly prevalent, cryptographic techniques have shown to be susceptible to collisions [69, 70], and the complex nature of cryptographic models makes it infeasible for certain applications [71]. The authors find that there exist unresolved challenges as follows:

1. *Side information attack* [3], which refers to an adversary having access to additional information apart from an anonymised graph.
2. *Dynamic social graphs* [3] and an adversary having access to the changes to a dynamic graph over time.
3. *Trade-off between privacy and utility* [3], which refers to the notion that privacy preserving techniques implemented to the extreme run the risk of rendering the OSN application stunted in functionality and thereby deplete its economic viability for the service provider.

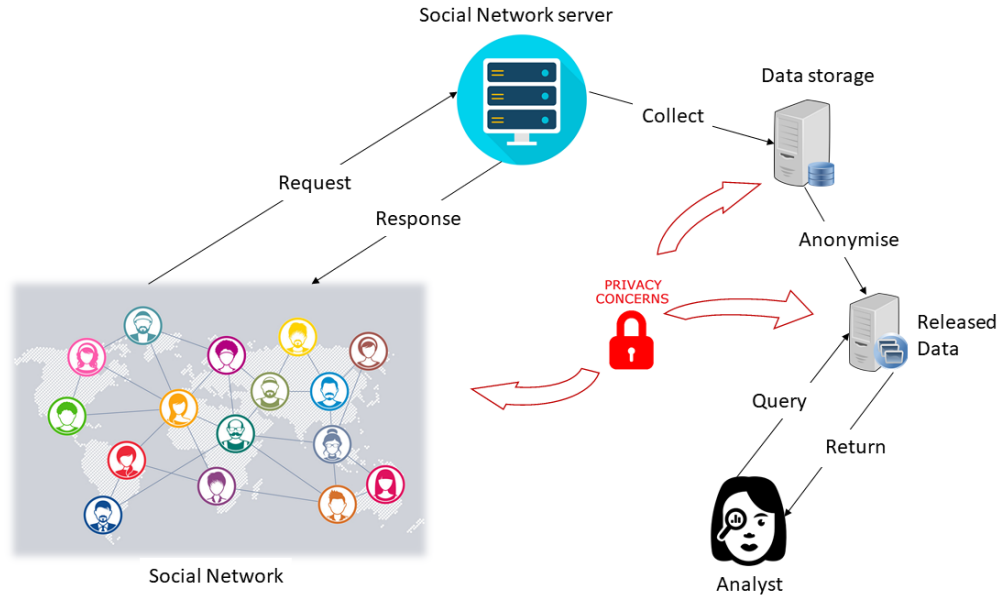
Pham et al. [3] note that a decentralised architecture [13, 72–74] may be a promising solution to the privacy issues in OSN.

Chen and Michael [9] model privacy breaches as “information gathering”, and describe the two methods involved: breaches due to “privacy disclosure”, and breaches due to “attack techniques”. The first refers to willingly provided information such as email addresses and date of birth, and the latter refers to deployment of attacks such as Sybil [75] and malware. In order to protect against privacy disclosure breaches, regulation both market and government driven, apart from self-regulation [76] has been suggested. Attack based breaches are harder to protect against, and architectural changes in OSNs such as peer-to-peer design [77] can help drastically reduce the impact of such techniques.

Chen et al. [78] also look at the attitude of users towards privacy and security and find that many users compromise their security credentials by reusing publicly available information in their passwords and security questions [79]. Lack of human-centric design [9] and complicated privacy policies lead to under-utilisation of privacy protection tools by users who fall back on their own judgement [80, 81]. The theory of weak ties [82–84] has been used to explain the need felt by OSN users to part with

## 2.2. STATE OF PRIVACY IN CENTRALISED SOCIAL NETWORKS

Figure 2.1: A typical social network [3]



private information due to the perceived benefit of disclosing information being higher than the risk [6,85,86], while also often “systematically underestimating the risks” [87]. Weak ties are said to include people whom a user is not directly close to, such as friends of friends and family of colleagues.

A collaborative OSN has been proposed by Blosser et al. [88] with the intention of preserving privacy of users of various disparate social networks by joining them using a federated client-server architecture. Yang [89] proposed a “sub-graph generalisation” approach to protecting privacy while sharing information across social networks based on general properties of the graph.

In [11], Rubinstein and Good examine five privacy mishaps each at Facebook and Google, and look at those through the lens of “privacy by design”. The authors cite Cavoukian’s “foundational privacy by design principles” [90]:

P1 Proactive not Reactive; Preventative not Remedial

P2 Privacy as the Default Setting

P3 Privacy Embedded into Design

P4 Full Functionality-Positive-Sum, not Zero-Sum

P5 End-to-End Security-Full Lifecycle Protection

P6 Visibility and Transparency – Keep it Open

P7 Respect for User Privacy – Keep it User-Centric

Through an analysis of the past privacy breach incidents of Facebook and Google, it is established that the firms overlooked all of these principles at some point. Gmail showed ads without consent, Google Street View photographed public places without notice, Buzz added Gmail contacts as friends without consent. Facebook News Feed exposed users' posts and activities as broadcasts with no or little control initially with privacy controls latched on later, Beacon showed targeted ads based on online purchases and posted the information in the users' News Feeds, Facebook's Apps gave applications liberal access with little control or explanation to users, automatic tag suggestion in Photos without consent with only opt-out. All of these situations arose because the firms preferred business demands over privacy concerns. In more than one case, transparency was compromised as users were not made aware of sudden changes to how their data was used, how much of it was collected, and controls were grossly inadequate, and at times, absent.

Rubinstein and Good argue that companies should endeavour to proactively include Fair Information Practice (FIPs) principles [11] as privacy concerns into their engineering process from the conceptualisation phase [91]. Microsoft, for instance, published an exhaustive set of guidelines for the purpose of “creating notice and consent experiences, providing sufficient data security, maintaining data integrity, offering customers access (to their data), and supplying [other privacy] controls” [11, 92] in 2006.

OSNs are increasingly influencing the lives of users and even non-users [8]: from being used by law enforcement, intelligence agencies, and detectives [93], to political revolution movements [94], to swaying democratic process [95]. There is no doubt that the use of OSNs will only increase and new avenues of exploiting data of the users of such systems will be devised. There is a need to change the design thinking around how private user data is dealt with, and relying on external regulation [96] alone is not sufficient. It has also been made abundantly clear that current iteration of privacy

tools and policies are immature [11, 78], perhaps because companies that build them prioritise business interests and undermine the usefulness of privacy controls.

Later in Section 5.2.2 on page 62, we will examine to what extent does the proposed application follow the principles of “privacy by design” using Cavoukian’s principles [90, 97].

## 2.3 Bootstrapping a developer in a new technology

This section briefly reviews research that talks of the pedagogical aspect of starting with new technologies for developers and developer experience.

### 2.3.1 On-boarding and Developer Experience

Fagerholm and Münch in [1] have attempted to define developer experience (DX). They characterise DX as a way to understand the feelings and thoughts of developers with respect to their interaction with their working environments. It is similar to user experience [98], customer experience [99] and brand experience [100]. DX includes experiences of a developer with all systems they interact with and all activities they partake during software development [1]. The authors develop a framework for DX consisting of 3 dimensions (Table 2.2):

Thoughts on development infrastructure (cognitive)	Tools, languages, libraries, platforms, frameworks, processes, and methods
Feelings about work (affective)	Respect, attachment, belonging
The value of one’s own contribution (conative)	Goal alignment, plans, intentions, and commitment

Table 2.2: Framework for Developer Experience (DX) [1]

In [101], Fontão, Dias-Neto, and Viana try to capture the “expectations, perceptions and feelings” of developers with respect to on-boarding on a new platform and participation, specifically for mobile application platforms from the point of view of platform maintaining organisations (Apple, Google, Microsoft). They note that developers need incentives for entering the ecosystem as well as staying in it. They find that developers are motivated to participate in a mobile software platform for 1) improving



### 2.3. BOOTSTRAPPING A DEVELOPER IN A NEW TECHNOLOGY

---

technical knowledge, 2) obtaining new knowledge, 3) building successful applications, and 4) project visibility.

[102] posits that adoption of a technology by individuals can be explained by the “Technology acceptance model (TAM)” [30], which establishes a connection between how the ease-of-use and usefulness of a technology is perceived, in addition to social influences. If the technology is supported by an organisation, then it is more likely to be seen as useful as well as credible to an extent. The study finds that, at least in case of enterprises, people are more likely to successfully adopt a technology if the management provides enough supporting resources to reduce the friction while transitioning to a new system. While this finding is not directly applicable to the case of web developers learning new technologies as they are more likely to be accustomed to frequent relearning, it is reasonable to expect supporting developer resources that make learning easier.

A “developer joining model” [2] (Table 2.3) has been proposed to help understand the forces involved in a developer coming in to an OSS project and staying.

Motivation	<b>Internal:</b> Self-marketing, Recognition, Enjoyment, Challenges, Improving programming skills, Identification with a community <b>External:</b> Scholarship, Course assignment, Better jobs, Career Advancement, Building human capital, Personal needs for a software solution [2, 103–105]
Attractiveness	Project age, License type, Intended audience, Type of project, Development status, Number of Hits, Number of Downloads, Number of Members, Number of Open tasks, Time for task completion, Software size, Structural simplicity [2, 106–108]
Hindering Forces	Learning curve, Lack of support from the community, Difficulties finding how to start, Excessive Time to reply to messages, Lack of courtesy in messages, Number of answers, Experience of respondents, Cognitive complexity [2, 109–111]
Retention	Supporting newcomers, i.e., mentorship, Supporting existent contributors to contribute more, i.e., incentives [2]

Table 2.3: Developer joining model [2]

Another detailed study into participation in OSS that looks at sustained participation [112] observes that OSS projects fail due to insufficient participation. This is often because individual developers in OSS are volunteers rather than paid employees.

### 2.3. BOOTSTRAPPING A DEVELOPER IN A NEW TECHNOLOGY

---

The authors also mention the motives that lead developers to initiate participation in OSS, namely, “software use value, status and recognition, learning, personal enjoyment, reciprocity, getting paid, sense of ownership and control, career advancement, free software ideology, and social identity” [112]. The overlap between these findings and those in Table 2.3 indicates an agreement among the academic community as to why developers choose to participate in OSS.

Based on the work by [1, 2, 112], we summarise the factors that affect developer experience and the reasons developers come and stay into OSS projects as follows:

Reasons for coming:

1. Technology shows promise
2. Visibility or popularity
3. Job opportunities
4. Interesting
5. Ideology
6. Recognition and identity

Reasons for staying:

1. Good developer experience
2. Job opportunities
3. Good developer community
4. Growth in platform
5. Incentives

Factors affecting developer experience:

DX1 Quality of learning resources

DX2 Activity in the community

DX3 Quality and Quantity of tooling

DX4 Stability of platform

DX5 Technical capabilities and features of the project

#### 2.3.2 Documentation and Learning resources

Developers learn new technologies constantly as a necessity of the occupation, but the decision to learn a new technology depends on the quality of the available developer resources. Documentation in the form of tutorials and guides, and supporting tools also serve the purpose of selling the development platform to new developers [31]. The quality of technical documentation has been the subject of many studies like [113] and [114] that look into what developers seek in documentation. [113] found that documentation and other learning resources were the “most severe obstacles in learning new APIs” [33]. [33] believes that reading documentation should reduce the time taken to learn with respect to learning without reading documentation. Quality of documentation is inversely proportional to “interruptions” and thereby directly proportional to productivity [115–117].

[32] identified the following 4 core principles for their analysis of tutorials of a set of 30 tutorials of various types (open-ended creative platforms, lecture-style courses such as MOOCs, evidence-based programming games, reference guides with code examples, and social forums):

1. Connecting to learners’ prior knowledge [32, 118, 119]
2. Organising declarative knowledge [32, 120]
3. Practice and feedback [32, 121, 122]
4. Encouraging meta-cognitive learning [32, 123, 124]

Developers regularly forage the web for learning resources: components, libraries [125], code examples [126], advice [127] and answers [128]. Most developers use question and answer sites and forums for getting help, and Stackoverflow<sup>19</sup> is the most popular question and answer site for programmers [129, 130]. Stackoverflow helps developers not only resolve errors they face, but also get advice about the best practices, and

---

<sup>19</sup><https://stackoverflow.com/> [Accessed: 20-07-2019]

undocumented features [125, 131], with answers often used as a substitute for official documentation [130]. In fact, Stackoverflow is so integral to a developer’s workflow that deeper integration of Stackoverflow with Integrated Development Environments has been proposed [125, 132, 133].

Apart from Stackoverflow, GitHub<sup>20</sup> is one of the largest resources for codebase and documentation for developers [125, 134, 135]. The social coding platform is home to millions of open source projects which get contributions in the form of code, issues, and pull requests. Developers also use the documentation in the form of READMEs and issue lists as resources for learning.

[125] associates Stackoverflow activity in the form of asking and answering questions with GitHub commits, while showing that users who tend to answer more questions rather than ask tend to have higher commits. The finding is interesting as it provides insight into engagement in social programming communities, and underlines the importance of Stackoverflow in catalysing development on GitHub.

Stiller and LeBlanc’s study in the effectiveness of “software engineering pedagogy” [31] is helpful in formulating some guidelines for producing developer learning resources that result in more successful transfer of knowledge. Those ideas, along with findings from the study of code examples on Stackoverflow by Nasehi et al. [126] and the 4 core principles [32] can be adapted to a set of *requirements for tutorial writing* as follows:

**T1 *A tutorial should make pre-conditions explicit***

Tutorials should be clear about prerequisites, so they can take into account the prior knowledge of the learners and use that to build new knowledge [32, 118, 119, 136–140].

**T2 *A tutorial should be fun***

The tutorial should not be dry and overly abstract. Learning requires motivation and tutorials should be engaging. One way of improving engagement is to use real life examples as the objective of development. For example, many engineering tutorials use the To-Do application example. It helps developers understand the capabilities of the platform while mapping them to familiar functionality.

---

<sup>20</sup><https://github.com/> [Accessed: 20-07-2019]

**T3 *A tutorial must include relevant code samples***

Coding tutorials must include code examples as mere textual descriptions are insufficient. Programmers must be able to study solved problems [126] and guided how they can be adapted for real-world use-cases. Since it's impossible to predict all issues users might face, tutorial writers should try and anticipate common pitfalls and include them as examples in the documentation.

**T4 *A tutorial should be useful***

Tutorials should lead to actual learning, not merely theoretical or conceptual. A good software technology tutorial has an actual application as an objective. The application may be simple, but it should have non-trivial functionality, or should be connected to higher-order tutorials that produce such an application.

**T5 *A tutorial should be accessible***

A tutorial should be well-scoped, and not excessively prolix. The steps to meet the objectives should be clear, the amount of abstract thinking required limited, and specific notation use minimal or simplified [31].

**T6 *A tutorial should be complete***

While short tutorials meet the criteria T1 and T3, they should nevertheless be complete. An open ended tutorial can leave readers confused. Even if a tutorial is scoped to one concept, it should be linked to the next in a meaningful way, or conclude learning by meeting the objectives. The tutorial should also provide links to additional materials and further reading supporting subsequent learning [32].

**T7 *A tutorial should have a clear, consistent voice***

The tutorial must be well-written and should be focussed on characterising [31] the technology being taught, instead of getting into discussions about software paradigms, or comments on competing technologies unless it is to illustrate functional differences.

## 2.4 Decentralisation efforts

This section presents the existing decentralisation efforts by drawing on the research and implemented technology.

Decentralisation is not a new concept. Owing to the abundant data aggregation opportunities and resulting data privacy concerns [13, 47], several decentralised OSN or DOSN designs have been proposed and implemented. Most notably, Safebook [141], PeerSoN [142], LotusNet [143], SuperNova [144], LifeSocial.KOM [145], Cachet [146], Persona [147], Vegas [148], Prometheus [149], Gemstone [150], Contrail [73], Soup [151], ProofBook [152], DECENT [153], SocialGate [154], Diaspora [155], Vis-à-Vis [156], My3 [157], eXO [158], DiDuSoNet [159], Friendica [160, 161], and RetroShare [162, 163] have been extensively studied in relevant literature [4].

[4] presents a comprehensive survey of DOSNs published in 2018, in which Salve et al. also present a classification system for various DOSN architectures. Most remarkably, they characterise DOSNs by topology as *unstructured* and *structured*. Structured DOSNs treat all peers as similarly capable and typically provide load balancing and dynamic addition and removal of peers out-of-the-box. Usually such solutions rely on some kind of Distributed Hash Table or DHT, and can be described as peer-to-peer or P2P systems [13]. Unstructured DOSNs, on the other hand, expect that some peers are more powerful and provide various services to other peers. This type of topology has been described as federation [13].

Another classification of DOSNs is based on the approach to data storage [4]. Data storage can either be *decentralised*, *semi-decentralised*, or *hybrid*. Decentralised storage imposes no restrictions and data is stored on random nodes; semi-decentralised storage selects a subset of nodes responsible for storage and management of data of all the users of the system; the hybrid approach relies on third-party storage such as public cloud providers.

Crucially, DOSNs differ in the privacy models and the methods used for implementing privacy policies. [4] classifies the privacy models as *relationship-based*, *group-based*, *profile-based*, and *content-based*. Relationship based privacy models exploit the connections formed between users of the DOSN to build the privacy policies; group-based privacy simply lets users create contact groups that they can then assign permissions to; profile-based privacy works by using the profile information of contacts; content-

## 2.4. DECENTRALISATION EFFORTS

based privacy models allow users to arrange their content into logical groupings and then access to these groupings can be defined. These privacy models are shown in Fig. 2.2.

Figure 2.2: A classification of common DOSN privacy models [4]

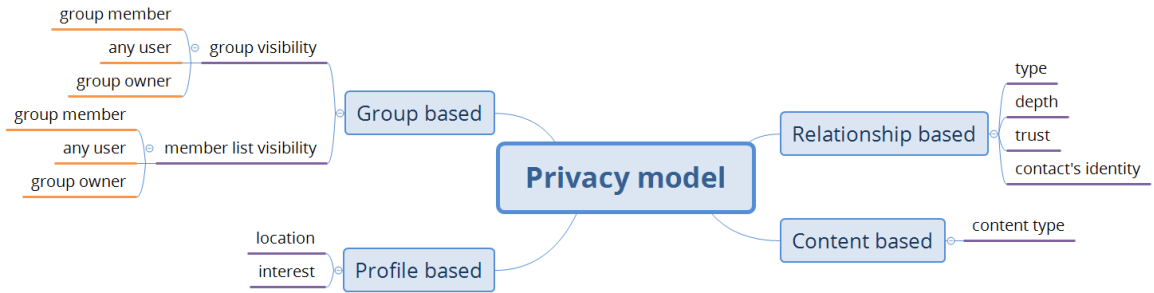


Table 2.4 summarises the classification of DOSNs obtained from the studies [4] and [13].

DOSN	Topology	Data storage approach	Privacy models	Privacy policy enforcement approach
Safebook	Hybrid	Semi Decentralised	Group-based, Relationship-based, Content-based	Cryptography
PeerSoN	Structured	Decentralised	Group-based, Relationship-based	Cryptography
LotusNet	Structured	Decentralised	Relationship-based, Content-based	Cryptography
SuperNova	Unstructured	Semi Decentralised	Relationship-based	Cryptography
LifeSocial.KOM	Structured	Decentralised	Group-based, Relationship-based	Cryptography
Cachet	Structured	Decentralised	Relationship-based	Cryptography

## 2.4. DECENTRALISATION EFFORTS

Persona	Unstructured	Hybrid	Group-based, Relationship-based	Cryptography
Vegas	Unstructured	Hybrid	Relationship-based	Cryptography
Prometheus	Hybrid	Semi Decen- tralised	Relationship- based, Profile- based, Content- based	Cryptography
Gemstone	Hybrid	Semi Decen- tralised	Profile-based, Content-based	Cryptography
Contrail	Structured	Hybrid	Group-based, Profile-based, Content-based	Cryptography
Soup	Hybrid	Semi Decen- tralised	Group-based, Relationship- based, Profile- based	Cryptography
ProofBook	Structured	Semi Decen- tralised	Group-based, Relationship-based	Cryptography
DECENT	Structured	Decentralised	Relationship-based	Cryptography
SocialGate	Structured	Hybrid	Relationship-based	Cryptography
Diaspora	Unstructured	Semi Decen- tralised	Group-based	Trusted peers
Vis-a-Vis	Unstructured	Hybrid	Group-based, Relationship-based	Trusted peers
My3	Hybrid	Semi Decen- tralised	-	Trusted peers
eXO	Structured	Decentralised	Relationship-based	Trusted peers
DiDuSoNet	Hybrid	Semi Decen- tralised	Relationship-based	Trusted peers
Friendica	Unstructured	Semi Decen- tralised	Group-based, Relationship- based, Profile- based	Trusted peers



## 2.4. DECENTRALISATION EFFORTS

RetroShare	Hybrid	Decentralised	Group-based, Relationship-based	Trusted peers
------------	--------	---------------	------------------------------------	---------------

Table 2.4: Comparison of decentralised online social networks

A key finding highlighted in [146] was that structured DOSNs suffer from a high latency while accessing logically linked relational data as it is disseminated across various nodes. For this reason, many DOSNs like Safebook and Prometheus choose a hybrid topology with overlay networks which make retrieval of data faster.

Unstructured DOSNs, while avoiding dynamism due to addition or deletion of nodes, have trouble ensuring availability if they chose semi-decentralised storage like Diaspora and Friendica. For ensuring availability, Contrail and Vegas rely on third party storage.

To overcome the availability hurdle, some systems like Safebook and Prometheus store encrypted data on random nodes, while others allow storing encrypted data on systems of trusted peers. This selection may be manual, or automated using trust models. The use of cryptography for dealing with curious or malicious [13] storage providing nodes creates its own challenges. Access control without encryption can easily be enforced by the servers, but encryption makes it the responsibility of the end host. The use of asymmetric-key cryptography on each outgoing message makes it in-feasible, so researchers have suggested using hybrid encryption [13], attribute-based encryption [147], and “IdentityBased Broadcast Encryption (IBBE)” [164].

As per [13], even encrypted DOSNs can leak private data in the form of social graphs by making metadata [165] openly-readable. For example, PeerSON [142] has to make the IP addresses and the online status of its users publicly available to ensure the DHT can be dynamically rebuilt and the data can be accessed. Persona solves this problem by not making any type of listing possible at all [147]. Another way in which DOSNs leak social graphs is due to access patterns, data size, creation times, and the amount of data [47]. SoNet [166] uses aliasing to remedy this problem, thereby disguising user identities. This approach fails for servers with lopsided distributions of users. For instance, 70% of the total users of Diaspora [13] use the same server. With servers having too few users, aliasing is not enough to hide identities.

## 2.4. DECENTRALISATION EFFORTS

---

Unbalanced servers also present the problem of large-scale aggregation. While the system is theoretically decentralised, having most of the users on a single server is as good as centralised [13].

Safebook [141] builds on the ideas discussed in [77] to propose a system that utilises real-life trust of users by modelling a user’s contacts into concentric circles known as “matryoshkas”. Safebook uses a DHT for routing. As per Schwittmann et al., the design of Safebook may be vulnerable to attacks on the DHT [13], apart from unreliability of data deletion [77] due to the possible unavailability of peers.

PESCA [72] is another DOSN that tackles the fundamental complication of availability by using a replica placement scheme which relies on two peer types: online friends both direct and indirect, and data audiences. It achieves this by constructing a structure called the “user online table (UOT)”, which is then used to determine the schedule and placement of replicas. Since the UOT is manual, it is possible for users to not participate in replication. PESCA also provides a high degree of privacy protection to users by using the broadcast encryption scheme [167] and shielding a user’s contact list from others using aliases. The authors note, though, that it’s still prone to “traffic analysis” [72] to infer social graph.

UNLYNX [168] is a recent DOSN design that decentralises both storage and compute by using the concept of collective authority of servers. The authors demonstrate that the protocols they’ve proposed are capable of fetching data from multiple sources, computing on independent servers, providing proof of correctness, and preserving privacy using a novel approach for obfuscation. The end result is a robust system, albeit slow when maximising security and privacy (a response time of 24m on 400,000 records).

Finally, the most popular operational DOSN is Mastodon [169], a micro-blogging service launched in 2016 with over a million users and over 2500 nodes worldwide. Mastodon uses the ActivityPub [170] protocol, and the architecture is similar to that of Diaspora. Users create accounts on a server which is connected to all other servers in a P2P topology, or host their own server. Like Diaspora, Mastodon is vulnerable to the possibility of leakage of the social graph. In fact, Mastodon and Diaspora are part of a federation<sup>21</sup> of inter-operable servers known as the fediverse [5, 171, 172]. The other most notable services that constitute the fediverse are PeerTube [173] and FrenDica [160]. Both Mastodon and PeerTube are based on the ActivityPub, while Diaspora and

---

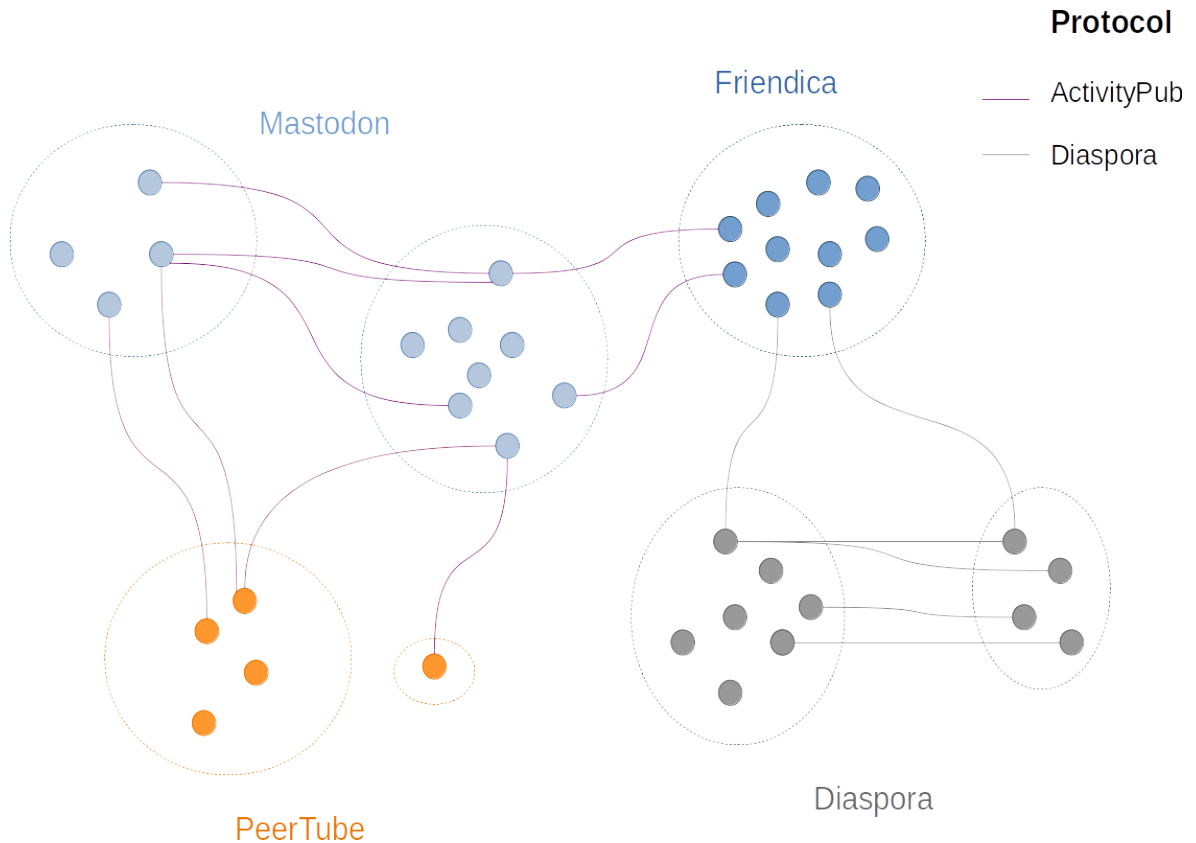
<sup>21</sup><https://the-federation.info/> [Accessed: 20-07-2019]

## 2.4. DECENTRALISATION EFFORTS

---

Friendica are based on the diaspora\* [174] protocol. Friendica also supports ActivityPub, hence Friendica servers are able to communicate with all of these services, as is shown in Fig. 2.3.

Figure 2.3: The federation of DOSNs [5]



Apart from these efforts, some blockchain-based DOSNs are being tested such as Steemit<sup>22</sup> and Blockstack<sup>23</sup>. Zeronet<sup>24</sup> is based on the popular bittorrent protocol. HELIOS<sup>25</sup> is a DOSN project currently underway which is funded by the European Union and aims to “design, implement and validate a state-of-the-art, decentralised P2P social media platform”.

---

<sup>22</sup><https://steemit.com/> [Accessed: 20-07-2019]

<sup>23</sup><https://blockstack.org/> [Accessed: 20-07-2019]

<sup>24</sup><https://zeronet.io/> [Accessed: 20-07-2019]

<sup>25</sup><http://helios-social.eu/blog/index.php/project/about-2/> [Accessed: 20-07-2019]

While a number of papers point to the potential privacy issues that arise due to the possibility of learning patterns of access to disseminated and encrypted data in P2P systems, these concerns are widely overstated. Learning such patterns and inferring any valuable knowledge from this kind of access would require considerably elaborate and expensive mining and automated analysis and can only be carried out on a relatively smaller scale, whereas such mining and analysis on centrally controlled OSNs is much more feasible and comparatively easy [175] since the providers can build support for such techniques into the platform from the beginning and have a much larger corpus of data at their disposal.

In any case, DOSNs still face major usability challenges. In particular, there exist features of OSNs only feasible in central scenarios:

1. ***Searching***

Searching through all the users of the network [175] is considered an important functionality provided by all major and popular centralised OSNs. For some like LinkedIn, it is an important business tool to be able to look for candidates. Such a feature has not been demonstrated so far, especially in an environment where availability of data is not guaranteed.

2. ***Recommendations***

Recommendation or discovery of people with common interests [175] is another area that relies on central indexing and aggregation but is not practical in DOSN scenarios. ARMOR [176] tries to address this, but has not been shown to be scalable.

3. ***Fake profile/content detection***

Classifying fake profiles and content [47] is an application anomaly detection in OSNs [177] with centralised architecture is a well-researched field. However, in a decentralised setting, it becomes much more difficult because data is distributed which makes detecting patterns and anomalies complicated.

## 2.5 State of the art of Solid

### 2.5.1 Linked Data

Linked Data is essential to Solid [12,178]. Linked Data is nothing but a set of best practices prescribed for how data is published. The idea is to develop and use standards for connecting published structured data that newly created data should be able to discover and link to [17]. Discovery of new data sources is a critical aspect of the movement. Not all data is created at the same place, but web standards can help build a Web of Data or the Semantic Web [14,17,19]. Using standards is another vital requirement for the Semantic Web to succeed.

Presently, Linked Data is based on two central pieces: RDF<sup>26</sup>, and HTTP<sup>27</sup>. RDF (Resource Description Framework) [179] is a W3C Recommendation that allows expressing facts using triples. An RDF triple contains a subject, a predicate, and an object. In the example (Fig. 2.4a), RDF is represented using the Terse RDF Triple Language<sup>28</sup>. The first line contains the *subject*, which is a WebID<sup>29</sup> of a Solid user, the third line contains the *object*, which is a WebID for another user, and the second line contains the *predicate* `knows` defined by FOAF Vocabulary Specification<sup>30</sup> which connects the subject to the object. This triple is equivalent to the English sentence “The person with WebID `https://theakashdeepsingh.solid.community/profile/card#me` knows the person with the WebID `https://www.w3.org/People/Berners-Lee/card#i`.” Note that the semantics of the predicate `knows` can be inferred from its definition in the vocabulary, just as the fact that the WebID’s belong to individuals of the class `Person` can be inferred by following the URI<sup>31</sup> and crawling the statements in the resultant document. This is where HTTP comes in. By representing things or concepts using HTTP URIs in the RDF documents, we obtain data that is linked to other useful data that describes it or is linked to it. By using standardised vocabularies [17,83], we

---

<sup>26</sup><https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/> [Accessed: 20-07-2019]

<sup>27</sup><https://tools.ietf.org/html/rfc7230> [Accessed: 20-07-2019]

<sup>28</sup><https://www.w3.org/TR/2014/REC-turtle-20140225/> [Accessed: 20-07-2019]

<sup>29</sup><https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/identity-respec.html> [Accessed: 20-07-2019]

<sup>30</sup>See: Friend of a Friend Vocabulary Specification 0.99, <http://xmlns.com/foaf/spec/20140114.html> [Accessed: 20-07-2019]

<sup>31</sup><https://tools.ietf.org/html/rfc3986> [Accessed: 20-07-2019]

are able to develop and refer to a shared understanding of those links. In this sense, it makes data self-describing, and follows the open world assumption of the Semantic Web.

Other standards in the space of Linked Data include SPARQL<sup>32</sup> – a query language for interacting with RDF triple stores [17], the Web Ontology Language or OWL<sup>33</sup>, and the Linked Data Platform or LDP<sup>34</sup>. LDP defines standards for HTTP based interactions with resources on the web to enable reading and writing Linked Data. These resources are often expressed as RDF using one of the serialisation formats, the most popular formats being Turtle, RDF/XML<sup>35</sup>, and JSON-LD<sup>36,37</sup>.

Figure 2.4a: Example RDF encoded as Turtle

```
<https://theakashdeepsingh.solid.community/profile/card#me>      # subject
<http://xmlns.com/foaf/0.1/knows>                               # predicate
<https://www.w3.org/People/Berners-Lee/card#i> .                # object
```

Figure 2.4b: Graphical representation of example RDF



## 2.5.2 Building blocks of Solid

Solid is composed of six major concepts that address a different part of the Solid Platform development stack. These are 1) Pods and Pod Providers, 2) WebID, 3) RDF, 4) HTTPS RESTful API, 5) Web Access Control, 6) Linked Data Notifications. The Solid specification [36] is currently described as a *draft*, and is “expected to change often”.

<sup>32</sup><https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/> [Accessed: 20-07-2019]

<sup>33</sup><https://www.w3.org/OWL/> [Accessed: 20-07-2019]

<sup>34</sup><https://www.w3.org/TR/2015/REC-ldp-20150226/> [Accessed: 20-07-2019]

<sup>35</sup><https://www.w3.org/TR/2014/REC-rdf-syntax-grammar-20140225/> [Accessed: 20-07-2019]

<sup>36</sup><https://www.w3.org/TR/2014/REC-json-ld-20140116/> [Accessed: 20-07-2019]

<sup>37</sup><https://json-ld.org/> [Accessed: 20-07-2019]

### Pods and Pod Providers

A Solid Pod or POD is a user’s “personal data storage” space [180]. A Pod represents a user on the Solid network, where the user can store data and grant applications access to parts of the Pod. In order to get a Pod, a user has two options:

1. Host their own Pod Server
2. Sign up with a Pod Provider<sup>38</sup>

Hosting one’s own Pod Server currently entails downloading and installing `node-solid-server`<sup>39</sup>, which is presently the only active implementation of the Solid Specification<sup>40</sup>. The users who can not host their own Pods can sign up with a Pod Provider. A Pod Provider is a multi-tenant installation of the `node-solid-server`, which allows users to create an account and issues a WebID for their new Pod. The concept of Pod Providers enables a federated topology in the Solid ecosystem. However, users on the same Pod Provider are logically isolated. Every Pod has the same capabilities as described by the Solid Specification (depending on the version of `node-solid-server`). Note that this is in contrast with Mastodon and Diaspora in which users on the same server can search for and discover other users on the same server. Hence, while a Pod Provider is similar in the sense that it allows multiple user accounts on the same physical system, it is not a central hub for those users.

### WebID

Solid depends on the WebID (Web Identity and Discovery) specification to describe Identity and Authentication. WebID’s are HTTP URI’s that are used to identify users. Every Solid Pod has a WebID that is protected by two schemes of authentication:

---

<sup>38</sup><https://web.archive.org/web/20190729213840/https://solid.inrupt.com/get-a-solid-pod> [Accessed: 29-07-2019]

<sup>39</sup><https://web.archive.org/web/20190729213729/https://solid.inrupt.com/docs/installing-running-nss> [Accessed: 29-07-2019]

<sup>40</sup><https://github.com/solid/solid-spec/tree/103b1e027356bd525e4cad0138e8288f4881df39> [Accessed: 29-07-2019]

WebID-OIDC<sup>41</sup> and WebID-TLS<sup>42</sup>.

The WebID-OIDC protocol is a decentralised authentication mechanism that extends the “Open ID Connect (OIDC)”<sup>43</sup> protocol. Essentially, a username and a password are used to authenticate a user and the Identity Provider issues an ID token, which is used for authenticated access to the Pod.

The alternative to WebID-OIDC is WebID-TLS which uses a TLS certificate to establish authenticity.

Usage of WebID entails that each Pod must have an RDF representation of WebID Profile accessible by dereferencing the WebID URI. This Profile document may contain other information about the user such as name, avatar, e-mail address, and is a public document. If the user does not want to share their personal information publicly, they need not include that data in the WebID Profile document, but a `foaf:name` predicate is strictly required.

## RDF

As mentioned before, RDF is the representation format for Linked Data. Within Solid, RDF is the ubiquitous content representation format. All data is expected to be modelled in the form of RDF Triples, and stored as one of the many serialisation formats such as Turtle, JSON-LD, N3<sup>44</sup>, RDF/XML, RDFa in HTML<sup>45</sup> and others. The other form of content on a Solid Pod while can not be parsed as valid RDF<sup>46</sup> is Binary files such as photos and other non-structured files [181].

LDP, which Solid is based on top of, describes a logical hierarchy for data. The Linked Data stored in a Pod describes Resources, which are grouped into Containers<sup>47</sup>. A helpful analogy (which is used in practise by `node-solid-server`) is to think of Resources as files, and Containers as directories. It is possible to create Containers

---

<sup>41</sup><https://github.com/solid/webid-oidc-spec/tree/2b2c5a3625be7e0286066db9f29a41e6c3d80b6f> [Accessed: 29-07-2019]

<sup>42</sup><https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/tls-respec.html> [Accessed: 29-07-2019]

<sup>43</sup>[https://openid.net/specs/openid-connect-core-1\\_0.html](https://openid.net/specs/openid-connect-core-1_0.html) [Accessed: 29-07-2019]

<sup>44</sup><https://www.w3.org/TeamSubmission/2011/SUBM-n3-20110328/> [Accessed: 29-07-2019]

<sup>45</sup><https://www.w3.org/TR/2015/REC-html-rdfa-20150317/> [Accessed: 29-07-2019]

<sup>46</sup><https://www.w3.org/RDF/Validator/> [Accessed: 29-07-2019]

<sup>47</sup><https://www.w3.org/TR/2015/REC-ldp-20150226/#ldpbc> [Accessed: 29-07-2019]



inside Containers, just like directories can be created inside directories.

### HTTPS RESTful API

While LDP prescribes a framework for HTTP-based interaction with Linked Data Resources<sup>48</sup>, Solid specifies a RESTful API<sup>49</sup>, extending the guidelines of LDP. Any Solid application that interacts with the Pod can use this API to perform CRUD operations on Resources and Containers, depending on access permissions. Apart from this REST API, a WebSockets API<sup>50</sup> is also available for publish/subscribe applications.

### Web Access Control

The “Web Access Control or WAC”<sup>51</sup> protocol is essential to Solid’s core principles. It describes a decentralised authorisation mechanism using .acl files. Access information is represented as Linked Data and uses the ACL ontology<sup>52</sup> for its terms [182]. Using WAC, it is possible to give a particular user identified by a WebID, a group of users, or everyone the access to a Resource or a Container [182]. It’s not mandatory to have an explicit ACL document for each Resource as Resources inherit the access level of their parent containers. The access control for the root container of the Pod is defined by default. The access modes available are Read, Write, Append (which means the ability to create new resources but not to update or delete existing resources), and Control (which means the ability to modify the access level) [182]. If an agent requests a Resource it does not have access permissions to, an HTTP Status 403 error is returned.

---

<sup>48</sup><https://www.w3.org/TR/2015/REC-ldp-20150226/#specs-http> [Accessed: 29-07-2019]

<sup>49</sup><https://github.com/solid/solid-spec/blob/103b1e027356bd525e4cad0138e8288f4881df39/api-rest.md#solid-https-rest-api-spec> [Accessed: 29-07-2019]

<sup>50</sup><https://github.com/solid/solid-spec/blob/103b1e027356bd525e4cad0138e8288f4881df39/api-websockets.md#solid-websockets-api-spec> [Accessed: 29-07-2019]

<sup>51</sup><https://github.com/solid/web-access-control-spec/tree/a71580b46a3ff124fa72d765a90432e488e96260> [Accessed: 29-07-2019]

<sup>52</sup><http://www.w3.org/ns/auth/acl> [Accessed: 29-07-2019]

### Linked Data Notifications

“Linked Data Notifications or LDN”<sup>53</sup> is a protocol for push message communication at the most basic level. It is also a W3C Recommendation. This is the only specified way for Pods to communicate, thereby making it the only way users can interact over Solid. The sender and receiver agree on a shared space on the receiver’s Pod where the sender can only create resources (messages), and the receiver can act on those messages at a later time. It’s like a simple postbox, where the sender only has Append access, while the receiver – the owner of the Pod – has complete access. As per the LDN Specification [183], the receiver’s endpoint is called an *inbox*.

### 2.5.3 Solid applications

Since 2016, a number of applications have been built on Solid that demonstrate its capabilities. As per [184], there are 73 applications out of which 14 are considered historical due to inactivity. Most of these applications are simple tools that implement a particular feature of Solid.

The oldest Solid application, dokiele [185], actually precedes the term Solid and originally targeted the read/write web<sup>54</sup>. dokiele is a “decentralised authoring tool” which implements the Web Annotations<sup>55</sup> specification, and LDN.

Another noteworthy Solid application is solid-chess<sup>56</sup> that uses the modern tool query-ldflex<sup>57</sup> apart from LDN and SQARQL to enable users to play a game of chess without a central server. Communication happens directly between users’ browsers and Pods.

---

<sup>53</sup><https://www.w3.org/TR/2017/REC-ldn-20170502/> [Accessed: 29-07-2019]

<sup>54</sup><http://web.archive.org/web/20190730015911/https://csarven.ca/dokieli> [Accessed: 30-07-2019]

<sup>55</sup><https://www.w3.org/TR/2017/REC-annotation-model-20170223/> [Accessed: 29-07-2019]

<sup>56</sup><https://github.com/pheyvaer/solid-chess/tree/dd45668da6ab59cf24e0580546bf39882be3f99d> [Accessed: 29-07-2019]

<sup>57</sup><https://github.com/solid/query-ldflex>

## 2.6 Summary and Next Chapter

This chapter establishes the key elements we will be using in the evaluation of Solid’s developer experience and the proposed application. We will examine whether the proposed application adheres to the principles of “privacy by design” (Section 2.2) [90]. We will use the “requirements for tutorial writing” (Section 2.3) to determine whether Solid’s existing learning resources meet those criterion. Lastly, we will show how the proposed application fits in the classification laid out in Table 2.4 (Section 2.4). These analyses will help us better understand the major themes involved in designing and deploying a decentralised OSN. They will also constitute much needed evidence-backed contributions to the Solid community.

# Chapter 3

## Research Methods

This chapter will focus on elucidating the methods used for this study.

This is an *applied* research, in the sense that the purpose is to find a practical solution to the problem of privacy invasion and data leakage in the large domain of web applications [186]. The research philosophy is *phenomenology* [187, 188] which is considered a variant of interpretivist [189] philosophy. This follows from the fact that we will be using a data collection method that is *qualitative* [190, 191] instead of purely quantitative. Specifically, the study involves maintaining the record of development activity, and obtaining insights from an analysis of the development journey of the participant.

The research is designed to be *exploratory* as we focus on developing a more nuanced understanding of the process [192] of building a privacy-preserving web application by studying the current solutions and building one. The research is not intended to be conclusive, rather it enables further research [193] into decentralised web application design and semantic web application development.

Our approach towards research is *inductive* since we formulate our research question at the outset, and plan to answer it by interpreting our observations [194, 195].

The method chosen for the qualitative analysis is a variant of *participant observation* [196, 197], whereby a single participant (the lead researcher) will observe their own software development process while programming an application and will record the programming sessions in a journal (See Appendix A). The journal will then act as the source-of-truth for the analysis (See Section 5.1) of the experience, which will in turn

lead to uncovering challenges and proposing improvements.

The following section details the attributes of the participant, and Section 3.2 explains the process followed by the participant.

## 3.1 Participant profile

The participant is an English-speaking Computer Science graduate student with about 36 months of professional software development experience. The skills relevant to this study that the participant possesses are: Node.js<sup>1</sup> programming, front-end web development<sup>2</sup>, Linked Data [25], and reading/writing technical documentation. Apart from this, the participant has built a social-networking application professionally. The participant is also an active contributor to OSS projects.

## 3.2 Study design

The study is executed in two phases: 1. Application Development and 2. Experience Analysis. In the first phase, we will develop an application using the Solid platform and record the development process through a journal. The development life-cycle is not formal, and uses the “build-and-fix model” [198] paired with “continuous integration” [199] since a single developer is involved in the lifecycle and the focus is not on code maintainability. The second phase involves an evaluation of the chosen platform, i.e., Solid, and the application built, i.e., Albus. This evaluation involves an assessment of Albus against the principles of “privacy by design” [11] as discussed in Section 2.2 on page 15, an assessment of the Solid developer on-boarding documentation against the “requirements for tutorial writing” as established in Section 2.3 on page 21, and looks at Albus through the classification of DOSNs as summarised in Table 2.4. In addition, we also look at some raw metrics to capture the size of the Solid community and compare them with those of projects with comparable age to determine the direction in which Solid is growing.

---

<sup>1</sup><https://nodejs.org/en/>

<sup>2</sup>[https://web.archive.org/web/20190804182140/https://en.wikipedia.org/wiki/Front-end\\_web\\_development](https://web.archive.org/web/20190804182140/https://en.wikipedia.org/wiki/Front-end_web_development) [Accessed: 04-08-2019]

### **3.3 Summary and Next Chapter**

This chapter covered the research methods employed for this study. In the next chapter, we detail the design and implementation phase of the application development.

# Chapter 4

## Application Design and Implementation

This chapter will act as a description of the design process and implementation details of the application. We start with an overview of the development environment (Section 4.1). Then, a set of planned features (Sections 4.2 and 4.3) and their corresponding design philosophy (Sections 4.4 and 4.5) will be discussed. Finally, we present the data models (Section 4.6) and technical details of the implementation (Section 4.7), including a brief discussion of the available state-of-the-art development tools.

### 4.1 Development Environment

1. **Operating System:** Ubuntu 18.04 LTS on WSL 1 (Windows 10)
2. **IDE:** Visual Studio Code<sup>1</sup>
3. **Node.Js version:** v10.15.1
4. **Web Browser:** Firefox Quantum 68.0.1 (64-bit)
5. **Node Solid Server version:** v5.1.1<sup>2,3</sup>

---

<sup>1</sup><https://code.visualstudio.com/> [Accessed: 05-08-2019]

<sup>2</sup><https://github.com/solid/node-solid-server/tree/v5.1.1> [Accessed: 05-08-2019]

<sup>3</sup><https://github.com/solid/node-solid-server/commit/bdc5acba326f215c3d32eb2b234ec73d0f5cc9ce> [Accessed: 05-08-2019]

## 4.2 Use-case Analysis

We propose to build an OSN since social networking is one of the most popular use-cases for the web, and involves a cross-section of data-driven functions like social interactions and user-generated content. The proposed system is to be specialised for academics and researchers because this is one of the use cases where a federated Pod hosting model can be practically implemented. As Solid depends on users having their own Pods, the users have to choose between hosting their own Pod which involves hosting a web-server and ensuring availability [77], and selecting a public Pod provider which involves establishment of trust with the Pod provider. Hence, we propose an inter-network of academics who have Pods at their university's servers. We believe establishing trust between users and the university or institution they are affiliated to is much easier. This also theoretically ensures a more uniform distribution of users [13] at a global level, given the system is adopted at scale.

## 4.3 Functional Requirements

### FR1 Log In with a Pod Provider

The application should allow a user to select a Pod Provider and log-in using WebID authentication. At the minimum, the application should support WebID-OIDC, but also attempt to provide WebID-TLS mode authentication. The application should make no assumptions about the user's Pod Provider. The application should also provide a list of well-known public Pod Providers.

### FR2 Creating and Editing one's Profile

The user should be able to create their profile by adding optional information such as email, phone, country, and company. None of this information should be mandatory, and the user should be allowed to edit this information through the application's Web-based graphical interface.

### FR3 Setting a Profile Picture

The user should have the option of uploading a profile picture, which acts as their avatar.



### 4.3. FUNCTIONAL REQUIREMENTS

---

#### FR4 Discover/find Users

The user should be able to discover users on the network or search for another user. It is critical for a social network to support interaction between users, and being able to find the other users is a basic requirement for interaction.

#### FR5 Following a User

It should be possible for a user to create connections with other users. In this case, we prescribe *following* as a one-way connection, whereby if a user Bob that follows another user Alice, then Bob becomes the *follower* and Alice becomes the *followee*.

#### FR6 Viewing Profile of a User

A user should be able to view the public profile information of another user, to be able to identify them.

#### FR7 Feed: Getting List of Posts of Followees

The follower-followee relationship should enable the follower to list the public posts of the followees. The follower should also be able to list posts they have been given explicit access to, through visibility levels defined below.

#### FR8 Creating Posts

An important part of most popular OSNs is the user-driven creation of content. The proposed application should allow users to create posts with text, and hyperlinks. Advanced features such as organisation tools like tags and keywords, file attachments, and hyperlinks should also be supported.

#### FR9 Assigning Visibility Permissions to Posts

We are of the view that the users should have full control over who is able to view their posts as being able to decide the access level of one's data is central to privacy protection. The application should support access control from the ground up, and the privacy tools should be flexible yet easy to use. At the minimum, the following visibility levels should be available for configuration:

- **Private:** Only the user sees their posts.

- **Public:** Everyone can see the posts theoretically if they visit the user's profile, but only followers (both approved and unapproved) see the user's posts in their feed.
- **ApprovedFollowers:** Only those followers, whose follow requests have been approved by the followee can see the followee's posts. If the followee explicitly disapproves the follow request of a user, then that user can not see the posts marked for ApprovedFollowers.
- **Specific users:** Allows specification of user(s) by WebID.
- **Custom groups:** Advanced; allows users to create custom named friend groups, and select the groups while creating a post.

### FR10 Pod migration

The application should allow users to migrate across Pod Providers, without having to rebuild their profiles and without losing any data.

### FR11 Comments and Likes on Posts

The ability to interact with posts is another important social element seen in most modern OSNs. A similar functionality should be provided by the proposed system.

### FR12 Collaboration

The application should allow users to give *editing* permissions to other users, thereby allowing multiple users to author content collaboratively.

## 4.4 Design Decisions

In this section, we discuss the design decisions and their rationale especially those arising from Solid's peculiarities.

### 1. React SDK<sup>4</sup>

---

<sup>4</sup><http://web.archive.org/web/20190805202814/https://inrupt.com/sdk> [Accessed: 05-08-2019]

## 4.4. DESIGN DECISIONS

---

We will be using the tools provided by the React SDK to build the application. For features that the React SDK does not fully support at the time of development, we will fall back on the lower-level tools such as `query-ldflex`<sup>5</sup> and `rdflib.js`<sup>6</sup>. The reason for using the React SDK is the latest attempt at creating a modern and “fun developer experience (DX)” [200]. The SDK consists of two parts: the scaffolding generator<sup>7</sup>, and `react components for Solid`<sup>8</sup>.

### 2. UI template

We will be using the UI template tools provided by “Shards Dashboard Lite React”<sup>9</sup> to develop the look-and-feel of the application in the interest of speeding up development.

### 3. Application data

Since Solid does not prescribe any hierarchy for storing data within the Pod, it is up to the application to manage the data. Our application will take inspiration from Noel De Martin’s blog<sup>10</sup> in which he recorded his thoughts while developing a “Todo list” application using Solid. The application data is stored in a container created directly in the root (`/`) of the Pod, and this container will be named `albus`. Inside the container, the application will create a resource named `user` which contains user’s data specific to Albus such as the followees and the group of *approved* followers, and a container named `posts` which stores posts created by the user. Any other metadata required by the application will also be stored inside the `albus` container. This container hierarchy will be created on the user’s

---

<sup>5</sup><http://web.archive.org/web/20190805203631/https://solid.github.io/query-ldflex/> [Accessed: 05-08-2019]

<sup>6</sup><http://web.archive.org/web/20190805204816/http://linkeddata.github.io/rdflib.js/doc/> [Accessed: 05-08-2019]

<sup>7</sup><https://github.com/inrupt/generator-solid-react/tree/6f8dec65ba04a9c28549e4e00a11f4a0ab864809> [Accessed: 05-08-2019]

<sup>8</sup><https://github.com/inrupt/solid-react-components/tree/61fc46e92eb398f7c5c759c8b1a482fff9bb314d>

<sup>9</sup><https://designrevision.com/downloads/shards-dashboard-lite-react/> [Accessed: 05-08-2019]

<sup>10</sup><http://web.archive.org/web/20190805214150/https://noeldemartin.com/tasks/implementing-a-task-manager-using-solid>

first log-in to the application; at every subsequent log-in, the application just checks if the container `albus` exists and proceeds to the home page if it does.

### 4. Follow request life-cycle

Fig. 4.1 depicts the life-cycle of a typical Follow Request. This process makes use of the Linked Data Notifications protocol (Section 2.5.2 on page 35). Bob clicks on the Follow button corresponding to Alice’s profile and that initiates the follow request. The request is sent to Alice’s `inbox` as a message using terms from the ActivityStreams<sup>11</sup> vocabulary. When Alice logs on to the application next time, she pulls the notifications from her `inbox` and is shown options to either *approve* or *ignore* the request. The approval action adds Bob to Alice’s `ApprovedFollowers` group, while the ignore action adds Bob to Alice’s `UnapprovedFollowers`. In both cases, the notification is removed from the `inbox`. Note that even if the follow request is ignored, Bob is able to read *public* posts by Alice – the *ignore* action is not equivalent to *block*. In order to maintain a list of users who are following Alice, but are not `ApprovedFollowers`, we maintain the *UnapprovedFollowers*. Another advantage is that Alice can move Bob to `ApprovedFollowers` in the future if she so wishes.

### 5. Feed aggregation

Aggregation is a centralised task. In a decentralised set-up, there are essentially two ways of building a feed of followees:

#### (a) Pull

While pulling, we have to build the feed – on the fly – when a user logs in. This entails crawling through all of the followees, and their posts, and aggregating them. This is a time-consuming task. Moreover, sorting becomes harder. Marking posts as seen so as to not repeat them across sessions is also harder as you’d have to incur storage cost for storing such data. Discovering all your followees’ posts is time consuming enough, and when done on the client-side it makes the presentation harder and brittle. If you want to sort

---

<sup>11</sup><https://www.w3.org/TR/2017/REC-activitystreams-vocabulary-20170523/> [Accessed: 05-08-2019]

#### 4.4. DESIGN DECISIONS

Figure 4.1: Follow request life-cycle

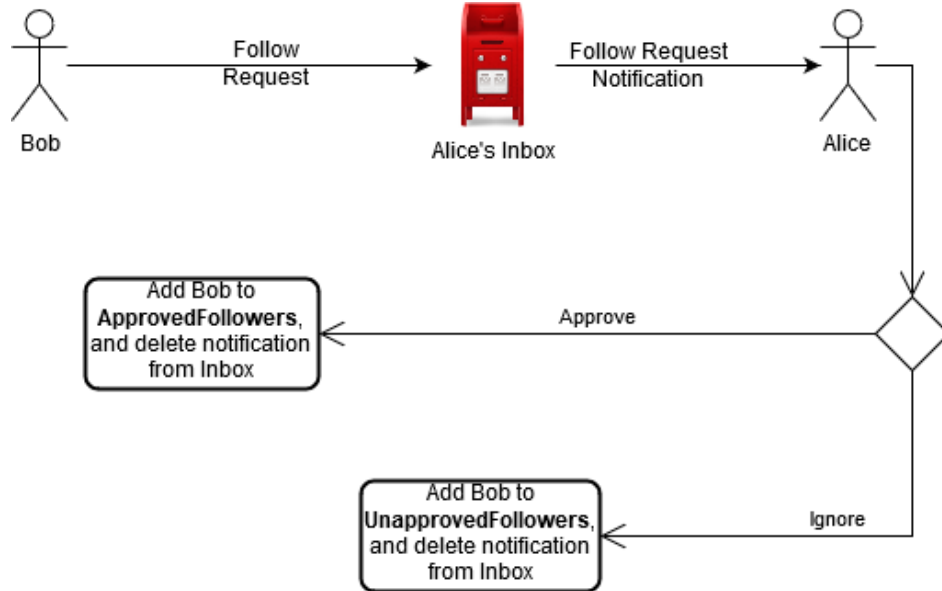
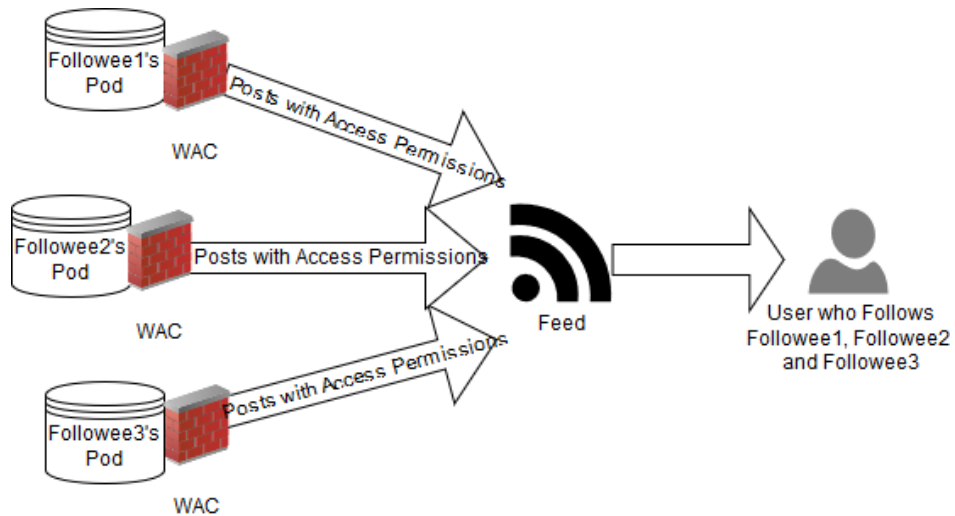


Figure 4.2: High-level overview of Feed aggregation



before presenting, as you should, that's increased latency. Over time, as the the number of posts increases, we'd also want to limit the posts fetched based on recency, and whether the post has been seen. Since we lack global knowledge beforehand, this is exceptionally hard.

(b) Push

In this approach, we expect that every time a user publishes a new post, it gets sent to the followers' inboxes as a notification. The rest of the process remains the same. The feed is still created on the fly, but the network cost of getting posts from followees' Pods can be avoided. The problem is that it requires followers' Pods and inboxes to be perpetually available. Another issue with this approach is that it requires a multi-cast to all of the targets of a post. If a post has a large number of target followers, this will get time consuming as the post creator's browser will have to make network requests to all the followers' inboxes.

To keep the implementation simple, we will use the first approach, as listing from multiple sources is easier than writing to multiple destinations.

#### 6. Discovering users

As established in Section 2.4 on page 29, searching for other users or discovering other users on the network is an unsolved problem for DOSNs. Such functionality depends on a central server maintaining a directory of all users. We consider two solutions to the discovery problem:

##### (a) Centralised

This approach involved using a central registry that all users are registered to. While this solution relies on additional mechanism, outside of Solid, it is possible to make the feature optional such that it's only enabled upon user's explicit consent, thereby maintaining the privacy-preserving nature of the application.

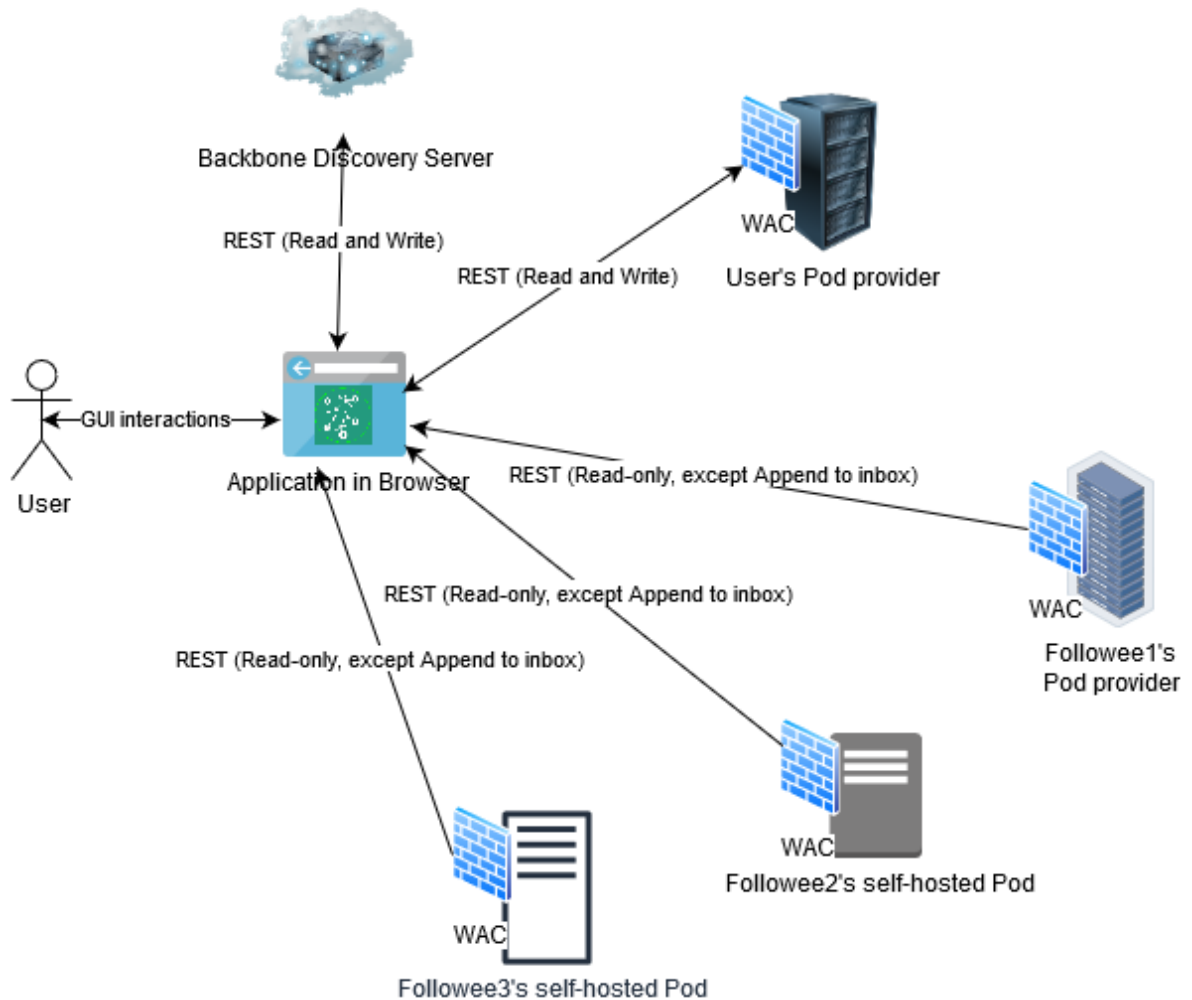
##### (b) Decentralised

This approach involves crawling friends, and friends of friends, and eventually building a graph. This method is not feasible as it is time-consuming and requires a mesh of network calls. Moreover, there exist cold start problems: i) a user having no friends will not be able to find anyone; and ii) no one will be able to find a user with no friends.

We will be using the centralised discovery mechanism for this application.

## 4.5 System design

Figure 4.3: System architecture of Albus



Albus, the system built as a result of this work, is a front-end web application written in the Javascript<sup>12</sup> programming language using the React.Js<sup>13</sup> framework.

To support discovery of users, a central backbone exists, which exposes two RESTful

<sup>12</sup><http://web.archive.org/web/20190805213715/https://developer.mozilla.org/en-US/docs/Web/JavaScript> [Accessed: 05-08-2019]

<sup>13</sup><https://reactjs.org/> [Accessed: 05-08-2019]

APIs: GET /users, and POST /users. Fig. 4.3 shows a high-level overview of the architecture and interaction protocols of various physical components of the system.

## 4.6 Data models

There are primarily 3 data models used by Albus. For the user resource, we use vcard<sup>14</sup> and sioc<sup>15</sup> vocabularies for describing its predicates. We also define a hasORCID predicate<sup>16</sup> to model a user's ORCID<sup>17</sup>. An example of the user model in RDF/TTL is given below:

```
@prefix : <#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix n: <http://www.w3.org/2006/vcard/ns#>.
@prefix ns0: <http://rdfs.org/sioc/ns#>.
@prefix cont: <http://www.w3.org/2000/10/swap/pim/contact#>.
@prefix ldp: <http://www.w3.org/ns/ldp#>.
@prefix inbox: <inbox/>.
@prefix n0: <https://akashdeep-singh.github.io/
    ontologies/albus/albus.ttl#>.

<#ApprovedFollowers/>
    n:hasMember "https://localhost:8553/profile/card#me".

:user
    a owl:NamedIndividual;
    ns0:follows "https://localhost:8553/profile/card#me";
    ns0:follows "https://localhost:8663/profile/card#me";
    ns0:follows "https://localhost:8773/profile/card#me";
    cont:preferredURI "https://localhost:8443/profile/card#me";
    ldp:inbox inbox:;
    n0:hasORCID "1234-5678-91011-121212".
```

---

<sup>14</sup><https://www.w3.org/TR/2014/NOTE-vcard-rdf-20140522/> [Accessed: 05-08-2019]

<sup>15</sup><http://rdfs.org/sioc/spec/> [Accessed: 05-08-2019]

<sup>16</sup><https://akashdeep-singh.github.io/ontologies/albus/albus.ttl> [Accessed: 05-08-2019]

<sup>17</sup><https://orcid.org/> [Accessed: 05-08-2019]



## 4.7. TECHNICAL IMPLEMENTATION DETAILS

---

The post resource is modelled primarily using the “DCMI Metadata Terms”<sup>18</sup> and sioc vocabularies. A sample post in RDF/TTL is shown below:

```
@prefix : <#>.
@prefix posts: <./>.
@prefix terms: <http://purl.org/dc/terms/>.
@prefix XML: <http://www.w3.org/2001/XMLSchema#>.
@prefix n: <http://rdfs.org/sioc/ns#>.

posts:884f9d0cd892ca6f70d42663099efe122663da33_1563538137312_3iss1m
  terms:created "2019-07-19T12:08:57Z"^^XML:dateTime;
  terms:title "Friday post";
  n:content "Lorem ipsum doler sum. ".
```

The third data model Albus uses is the LDN message for follow requests. This is modelled using the ActivityStreams vocabulary and is represented using JSON-LD as mandated by the LDN protocol specification [183]. A sample is reproduced below:

```
{
  "@context": "https://www.w3.org/ns/activitystreams#",
  "@id": "",
  "type": "Follow",
  "actor": "https://localhost:8553/profile/card#me",
  "object": "https://localhost:8443/profile/card#me",
  "updated": "7/5/2019, 6:29:17 PM"
}
```

## 4.7 Technical Implementation Details

Apart from the RESTful API, the Solid community has built two higher-level libraries for interacting with Pods, which allow reading and writing data: `rdflib.js` and `query-ldflex`. These libraries take different approaches to data access, and are currently at different levels of maturity. `rdflib.js` covers more features of the Solid specification, like creation of containers and resources, which `query-ldflex` does not. However, `query-ldflex` can be used for retrieval and update operations on resources due to its simplicity.

---

<sup>18</sup><https://www.dublincore.org/specifications/dublin-core/dcmi-terms/2012-06-14/> [Accessed: 05-08-2019]

## 4.8. SUMMARY AND NEXT CHAPTER

Figure 4.4: User Interface of the homepage of Albus

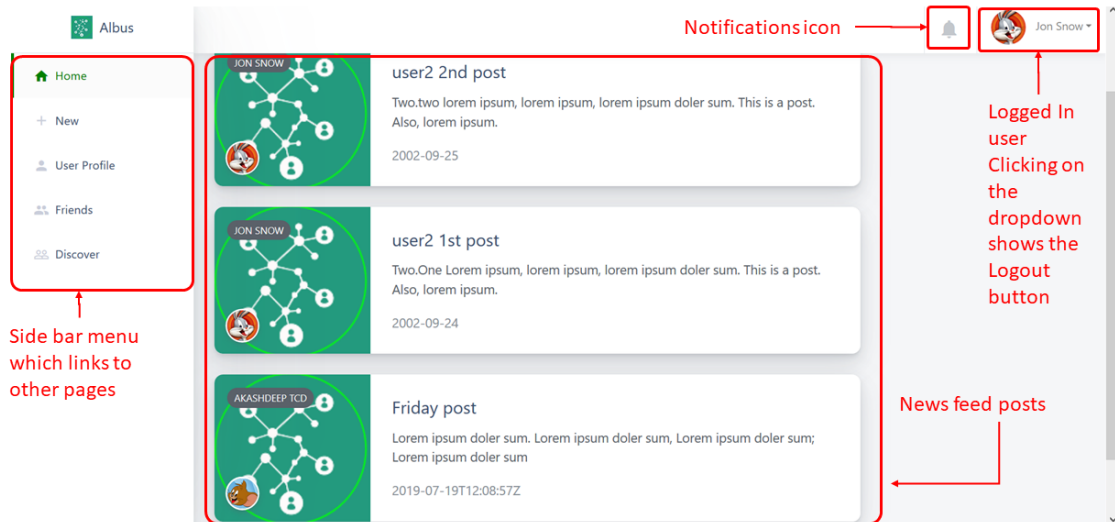


Fig. 4.4 shows a screenshot of the homepage of Albus with the key UI elements labelled. More screenshots are included in Appendix B.

Table 4.1 summarises the implementation of the functional requirements with respect to the aspect of Solid used, the data access mechanism used, and whether the React SDK was used for that particular feature or not. Due to time constraints, FR11 and FR12 were not implemented. However, implementing FR11 is straightforward as it would involve similar concepts as the implementation of FR8 (post creation) along with LDN. Similarly, FR12 may be implemented using the *Write* mode permissions in WAC. FR10 was attempted but tests were unsuccessful for reasons detailed later in Section 5.1.5 on page 61. The detailed implementation process can be followed in the development journal included in Appendix A.

## 4.8 Summary and Next Chapter

This chapter dealt with the design process and internal workings of Albus. We discussed the planned functionality, implemented functionality, and mapped the implemented

#### 4.8. SUMMARY AND NEXT CHAPTER

Functional Requirement	Screenshot	Solid feature used	Data access interface used	Implemented using React SDK Components?
FR1	Fig. B.4, B.5, B.6, and B.7	WebID-OIDC	solid-auth-client	Yes
FR2	Fig. B.10	RESTful API	query-ldflex	Yes
FR3	Fig. B.10	RESTful API	RESTful API	Yes
FR4	Fig. B.13 and B.8	(not provided by Solid)	–	No
FR5	Fig. B.14 and B.12	RESTful API, Linked Data Notifications	rdflib.js	No
FR6	Fig. B.11	RESTful API	query-ldflex	No
FR7	Fig. B.15	RESTful API	query-ldflex	No
FR8	Fig. B.9	RESTful API	rdflib.js	No
FR9	Fig. B.9	RESTful API, Web Access Control Lists	rdflib.js	No
FR10	–	(test unsuccessful)	–	–
FR11	–	(not implemented)	–	–
FR12	–	(not implemented)	–	–

Table 4.1: Functionality implementation mapping with Solid features

features with Solid concepts and the tools used to build them. In the next chapter, we will evaluate the developer experience and Albus.

# Chapter 5

## Evaluation

This chapter will describe the findings from the study. Section 5.1 deals with the evaluation of the developer experience driven by the journal (Appendix A) and classified using the “factors affecting developer experience” established earlier in Section 2.3.1 on page 19. Section 5.2 evaluates the application built, i.e., Albus, for its performance, examines it against the principles of “privacy by design”, and determines the key attributes of DOSNs it employs.

### 5.1 Development Experience Evaluation

#### 5.1.1 DX1: Quality of learning resources

The present set of learning resources were found to be outdated as well as ineffective. This is primarily because documentation is sparse and disconnected. The only official source of documentation is the website<sup>1</sup>. This website is divided into mostly unconnected pages, the first of which is a “Getting Started” page which links to other pages for “Learning the Basics”. There are three tutorial pages, none of them connected to each other, and none of them pointing to the *basics* as prerequisites. The lunch break tutorial<sup>2</sup> is the simplest, yet confusing and inadequate. The other tutorials simply

---

<sup>1</sup><http://web.archive.org/web/20190807004544/https://solid.inrupt.com/docs/getting-started> [Accessed: 07-08-2019]

<sup>2</sup><http://web.archive.org/web/20190807004517/https://solid.inrupt.com/docs/app-on-your-lunch-break> [Accessed: 07-08-2019]

## 5.1. DEVELOPMENT EXPERIENCE EVALUATION

---

show how to install a scaffolding generator for generating a sample application with Angular<sup>3,4</sup> and React.Js<sup>5</sup>. These tutorials do not attempt to connect the knowledge to the *basics* of Solid, nor do they indicate further reading for building real applications. Table 5.1 tabulates the evaluation based on tutorial requirements framed in Section 2.3.2 on page 21.

	<b>lunch break tutorial</b>	<b>angular tutorial</b>	<b>react tutorial</b>
T1	No	No	No
T2	Yes	No	No
T3	Yes	No	No
T4	No	No	No
T5	Yes	Yes	Yes
T6	No	No	No
T7	Yes	Yes	Yes

Table 5.1: Evaluation of Solid tutorials

All other documentation regarding Solid is scattered across various Github repositories, the “official community forum”<sup>6</sup>. In Part II of this study(*Getting Started with Solid*), we have provided a model for creating a more effective tutorial for beginning Solid development. This beginner’s guide is inspired from the VueJs guide<sup>7</sup> and focused on filling the gaps we found in the official documentation. Specifically, we begin with FAQ-style questions and answers to address what we feel are important concerns, and then gently introduce Solid and some functionality. While this guide is by no means exhaustive, we link to other resources so that readers can continue learning. These links are given context so that readers don’t feel lost.

---

<sup>3</sup><http://web.archive.org/web/20190807004510/https://solid.inrupt.com/docs/writing-solid-apps-with-angular> [Accessed: 07-08-2019]

<sup>4</sup><https://angular.io/> [Accessed: 07-08-2019]

<sup>5</sup><http://web.archive.org/web/20190807004502/https://solid.inrupt.com/docs/writing-solid-apps-with-react> [Accessed: 07-08-2019]

<sup>6</sup><https://forum.solidproject.org/> [Accessed: 07-08-2019]

<sup>7</sup><https://vuejs.org/v2/guide/> [Accessed: 07-08-2019]

### 5.1.2 DX2: Activity in the community

#### Analysis of Solid community

- **Stackoverflow**

On Stackoverflow (as of 22-07-2019), 50% out of the  $10^8$  questions tagged `solid` asked went unanswered, while 30% had only one answer, and the rest had 2 answers. This is much worse than the average for Stackoverflow: an estimated 14.19% questions were unanswered, while over 50% questions had more than one answers [130]. For comparison, consider the project `deno`<sup>9</sup> which was launched in June 2018, still is in heavy development, is not recommended for production, and has 13 tagged questions<sup>10</sup>. While there is no real baseline for comparing Solid with, we will take `deno` as an indicator of what an average project in the developer community with can look like. `Deno` is not exactly comparable as it is simply a Typescript-based<sup>11</sup> back-end web development platform much more comparable to Node.js, but the interest in it is clearly higher as is the community engagement, despite being younger, more of a niche, and having started as an experiment. Given the age of these projects, Solid is clearly lacking in generation of interest among users of Stackoverflow. As discussed in Section 2.3.2 on page 20, Stackoverflow is an important community site for developers, and can play a crucial role in providing help to developers. We believe that the lack of Solid-related activity on Stackoverflow needs to be addressed.

- **Github**

An analysis of the GitHub repositories (as of 22-08-2019) confirms the suspicion that Solid might be suffering from a lack of interest from the general web development community. `Deno` has a single core repository<sup>12</sup>, while Solid has several.

---

<sup>8</sup><https://web.archive.org/web/20190722191907/https://stackoverflow.com/questions/tagged/solid>

<sup>9</sup><https://deno.land/manual.html#introduction> [Accessed: 22-07-2019]

<sup>10</sup><https://web.archive.org/web/20190722192305/https://stackoverflow.com/questions/tagged/deno> [22-07-2019]

<sup>11</sup><https://www.typescriptlang.org/> [Accessed: 07-08-2019]

<sup>12</sup><https://web.archive.org/web/20190722193152/https://github.com/denoland/deno> [Accessed: 22-07-2019]

However, the NSS repository<sup>13</sup> is the most popular and most contributed to as well as the single official reference implementation of the Solid specification.

Deno (first commit: June 2018) has 1,747 commits in the master branch, while NSS (first commit: October 2014) being almost 4 years older has 2,489. The number of open issues is higher in deno at 211 to NSS's 152, as is the number of open PRs - deno has 29 and NSS has 9. Deno also has more stars, which are considered an indicator for attention [201] on Github, with 37,078 users having starred deno's repository compared to NSS's much lower 1,308. The number of watchers can also be considered as a proxy for interest in the community; deno has 1446, whereas NSS has 88. In terms of indicators of contributions, deno has 1712 forks and 144 contributors so far, while NSS lags again with 222 forks and a much smaller group of 57 contributors. While it is out of the scope of this study to look into the reasons of this seemingly low interest relative to an experimental project like deno, we believe it is worth finding out so that steps can be taken to improve adoption by the larger web development community.

- **Gitter**

A look at the number of messages on the Gitter rooms (Table 5.2) shows a slightly different picture. Indeed, Solid being about 3-4 years older than deno causes it to have a much larger number of messages if we just compare the solid/chat room to deno's only denolife/Lobby room. The number of users who have participated in solid/chat is also much higher.

### **Interactions with the Solid community**

Our interactions with the Solid community happened through 4 channels: Github, Gitter, the forum, and twitter. All of these channels serve different purposes.

The interactions over Github were limited to raising issues and pull requests (PRs) or commenting on them. Notably, one PR created by us has since been approved<sup>14</sup> and is pending merge into the master branch of the rdflib.js library. The turnaround time for Github activity was between 2 to 7 days.

---

<sup>13</sup><http://web.archive.org/web/20190722193200/https://github.com/solid/node-solid-server> [Accessed: 22-07-2019]

<sup>14</sup><https://github.com/linkedata/rdflib.js/pull/325>

## 5.1. DEVELOPMENT EXPERIENCE EVALUATION

---

room	message count	first message	5000th message	user count
solid/app-development	5461	05-09-2018	30-06-2019	130
solid/chatapp	135	18-02-2019		
<b>solid/chat</b>	<b>46726</b>	<b>15-01-2016</b>	<b>10-03-2016</b>	<b>795</b>
solid/data-browser	272	30-05-2019		
solid/node-solid-server	26121	07-02-2015	16-09-2016	171
solid/solid-spec	1918	10-04-2015		
<b>denolife/Lobby</b>	<b>9882</b>	<b>28-06-2018</b>	<b>04-03-2019</b>	<b>310</b>

Table 5.2: Gitter messages on Solid and Deno rooms, as on 06-08-2019

On Gitter, the interaction was in the form of messages to chat-rooms as well as private messages to members of the community. On chat-rooms, we found that questions were answered within minutes by helpful members who happened to be online, while private messages were responded to within a day or two. This was the fastest way of getting help. Similar interactions took place over twitter via tweets, but these were few.

The forum was the richest source of information in the form of questions and answers. We created topics as well as follow-up posts to existing topics to ask questions and to ask for clarifications. These queries were usually addressed within 24 hours.

The interactions with the community showed that getting help from the members is fairly easy on Gitter and the forum, and the community is welcoming.

### 5.1.3 DX3: Quality and quantity of tooling

#### React SDK

The React SDK consists of 2 parts: the components and the generator.

We found that the components did not support much of Solid's built-in features when we began development (February 2019), especially creation of resources. The components lacked in documentation, so it was unclear how to use them, especially in a more generalised manner apart from the example use-cases shown in the documen-



tation. However, the `withWebId` and `useWebId` “higher-order components (HOCs)”<sup>15</sup> were indeed useful in making the WebID available as well as in checking for authentication status.

The generator has taken the shape of a sample application instead of just scaffolding. It’s necessary to point out that generated scaffolding would not actually have any features and would not be a functioning Solid app at all. Rather it would be a generated react app with Solid dependencies included. The next logical step would be to have a tutorial that builds our sample app piece by piece using the components and lets developers learn the component SDK. We believe that the SDK should focus more on the components rather than the generator.

### Data access interfaces

Solid has 3 popular ways of interacting with data programmatically:

1. RESTful API

It would seem that this is the basic and the canonical approach also laid out in the specification. All other interfaces seem to be built on top of the REST interface, which is documented well enough and is quite comprehensive. It requires knowledge of HTTP, LDP, and RDF.

2. `rdfib.js`

This is perhaps the second most mature interface (apart from the Data Browser). The problem is it assumes a knowledge of RDF and LDP. That would be fine if the documentation was better, but it’s fragmented, incomplete, and inconsistent.

3. `query-ldflex`

This is the newest, and by far the easiest to get started with. Unfortunately, this too suffers from the incompleteness of documentation apart from the fact that it does not have all the features to be able to provide all of Solid’s capabilities. While it’s being actively worked on, it will take some time for it to be mature.

The bottom line is that no one solution fits all of the needs for a practical application like an OSN. At the very least, a document covering all the common CRUD scenarios

---

<sup>15</sup><https://reactjs.org/docs/higher-order-components.html> [Accessed: 05-08-2019]

for containers, resources, and triples should exist for the 3 interfaces from a view of helping new developers started. This is not the case yet. None of the documents that exist contain this information.

Some use cases such as filtered listing do not even seem to be possible yet, due to the way data is stored and serialised on Pods.

Given that the primary data access interface for Solid apps is RESTful, this creates a gap for meaningful querying. Most applications are backed by a database, RDBMS or NoSQL, which provides a sophisticated query interface that lets developers perform selection and projection operations at the very least, apart from more complex aggregation. Solid stores data mainly as Turtle representation of RDF. While this serialisation may not be the most efficient, it is expressive enough for designing versatile data models. However, querying Turtle has traditionally been done using SPARQL, which is highly expressive, albeit complex.

While the Solid specification prescribes it<sup>16</sup>, NSS lacks a SPARQL endpoint<sup>17</sup> for querying over the data in the Pod, making it necessary to perform operations such as selection, projection, and aggregation over Pod data inside the client application. This makes modelling data difficult especially when designing with privacy in mind. Since permissions using .acl files within the Pod are assigned at the level of resources (files on disk in case of NSS), different access levels need to be isolated at the resource/container level rather than predicate/property level.

This means that if a user Alice wants to create a post visible only to a specific follower Bob, then a separate resource needs to be created specifically for Bob with the .acl file associated with that resource having the WebID of Bob. Since the listing of resources in a container is done at the container level without any filters, a listing by every follower of Alice will show that resource but a request to get the content of that resource will fail with a 403.

To fill this gap, libraries such as `rdflib.js` and `sparql-fiddle`<sup>18</sup> have implemented workarounds that allow executing SPARQL over loaded RDF documents but the ex-

---

<sup>16</sup><https://github.com/solid/solid-spec/blob/b941ff795acdedb7d7a24d40dabdfcce7efa9283/api-rest.md#alternative-using-sparql> [Accessed: 05-08-2019]

<sup>17</sup><https://github.com/solid/node-solid-server/issues/962#issue-383649959> [Accessed: 05-08-2019]

<sup>18</sup><https://github.com/jeff-zucker/sparql-fiddle/tree/6dc2a464d0b637466a23f3027c7e3ad308746791> [Accessed: 05-08-2019]

perience is not smooth.

### 5.1.4 DX4: Stability of the platform

The platform currently comprises of only NSS. The user interface of the default data browser<sup>19</sup> that ships with NSS was found to be lacking in user-friendliness. In particular, the sharing feature was found unusable and made it necessary to edit `.acl` files manually.

Another problem was the frequency of bugs in NSS and other libraries. This was not entirely unexpected as both the Solid specification and NSS are evolving rapidly. There are open issues<sup>20,21</sup> and threads<sup>22</sup> that deal with the compliance differences between NSS and the specification. As a result of these, developing on top of NSS is hard because even minor version updates introduce breaking changes and regressions. Most libraries don't have adequate or any tests which in turn leads to bugs slipping into releases.

### 5.1.5 DX5: Technical capabilities and features of the project

#### Usability issues

Apart from the challenges in developing functionality mentioned in Section 2.4 on page 29, we noticed a number of other issues that arose as a result of Solid's approach to decentralisation. In particular we found that the Sign-up process was longer now, as it requires at least an extra step for selecting Pod provider. In addition, loading the feed on the client application is slow, as it requires multiple steps and several network calls. Since we modelled each `post` as a resource, getting the post details requires one HTTP GET for each `post`. Even before getting the individual posts, it is necessary to discover their URI's by listing the posts of each followee. As discussed in Section 4.4

---

<sup>19</sup><https://github.com/solid/userguide/tree/c9f7e62e184fafb0c66e5a15a16562b7d90197f5> [Accessed: 05-08-2019]

<sup>20</sup><https://github.com/solid/node-solid-server/issues/1190> [Accessed: 05-08-2019]

<sup>21</sup><https://github.com/solid/solid-spec/issues/174> [Accessed: 05-08-2019]

<sup>22</sup><https://lists.w3.org/Archives/Public/public-solid/2019May/0015.html> [Accessed: 05-08-2019]

on page 45, this is likely to perform poorly in a real world scenario and is not scalable beyond a handful of followees.

### Pod migration

Pod migration is not officially supported, and hard to accomplish because the WebID is essential to the Pod, and the WebID URI is strongly coupled with the Pod data. The implication of this dependence on URIs is that moving to a different Pod provider involves getting a new WebID URI. Not only do all external references to one's WebID URI need to change, but also the ones inside the Pod.

Our tests involved executing NSS inside a docker container<sup>23</sup> and mounting the `data` directory on an external volume. In order to simulate a migration, we started NSS in another container. Now, we had two containers with different domain names (`localhost:8443` and `localhost:8553`). Detaching the volume from one container and re-attaching it to the other did not work. It was observed that the OIDC provider was configured using the server's domain name, so the user account was tied to the domain name as well.

The considered workaround for these problems was automatically recreating the user's account<sup>24</sup> on the new server using `solid-cli`<sup>25</sup>, reconstructing all of the data by copying, and replacing the old WebID URI with the new one. This method is not only unreliable, but also invasive as it requires white-box access to the Pod.

These issues exist due to the inherently centralised nature of DNS. These issues have been discussed extensively in the Solid community<sup>26,27,28</sup>, but a robust solution is yet to be implemented.

---

<sup>23</sup><https://www.docker.com/> [Accessed: 05-08-2019]

<sup>24</sup><https://github.com/solid/solid-spec/blob/bea2322d2ada2a9402599176e4b04fc50970348b/recommendations-client.md#creating-new-accounts> [Accessed: 05-08-2019]

<sup>25</sup><https://github.com/solid/solid-cli/tree/26bb238f511487a50ee878b8313caaa7ea109d>  
28 [Accessed: 05-08-2019]

<sup>26</sup><https://forum.solidproject.org/t/transferring-a-solid-pod/1902> [Accessed: 05-08-2019]

<sup>27</sup><https://forum.solidproject.org/t/move-it-at-any-time-without-interruption-of-service/565> [Accessed: 05-08-2019]

<sup>28</sup><https://forum.solidproject.org/t/will-tying-web-ids-to-hosters-create-lock-in/756> [Accessed: 05-08-2019]

### Access Control

Listing contents (using blobbing or `ldp:contains`) of a container can lead to a breach of privacy, because a user may not have access rights to all of the contents. The user will then be able to learn which of those they do not have access rights to by making GET requests to each of the resources as the requests will either succeed or fail with a 403 error. Thus, they will be able to learn which resources have been withheld from them.

## 5.2 Application Evaluation

### 5.2.1 Performance

To evaluate the performance, we consider the News Feed functionality as it is the most commonly used feature in OSNs. As already mentioned, decentralised feed aggregation can be slow. A comparison with Facebook’s News Feed performance showed that loading the first 7 posts by visiting the homepage takes 48 ms on an average. On the other hand, Albus takes 1873 ms to fetch the list of one followee’s posts, and then about 1639 ms for each post containing just three predicates (title, created, and content) with the content being 90 characters long. Clearly, Albus loads the News Feed considerably slower than a centralised OSN, even when the data is being served from a Pod hosted on the same machine.

### 5.2.2 Privacy by design

We refer to the “privacy by design” principles we discussed in Section 2.2 of page 15 and hold Albus against these principles using the following two cases:

Case 1: Albus does not collect any user data without consent. The only user data Albus stores on a central server is the WebID. The feature is designed in such a way that users explicitly opt-in. The buttons have been given neutral colours so that users are not misguided into giving consent. The application still works correctly for a user if they do not opt-in to register for the discovery listing.

## 5.2. APPLICATION EVALUATION

---

Case 2: Albus provides easy-to-use and flexible privacy tools to configure the visibility of posts. The default visibility level is private so that users do not accidentally expose data. Even if a user keeps all their posts private they can continue to use all features of the application.

P1 Albus is proactive in its approach to privacy. Both Case 1 and Case 2 illustrate this.

P2 As mentioned in Case 2, the default setting for post visibility is private. Care has been taken to make sure that users have to give explicit permissions before exposing any data.

P3 Every design decision has been taken keeping the privacy of user data in mind. Moreover, Albus is built using Solid, which puts privacy at the centre of its core values.

P4 As mentioned in Case 1 as well as Case 2, the application works fully and correctly for all users regardless of their privacy preferences. The only exception is their name, which can be edited, but it's public and anyone having access to their WebID can see it.

P5 Albus does nothing to encrypt connections or user data. That is the responsibility of the underlying infrastructure. While Solid mandates the use of HTTPS, encrypting Pod data is not yet possible. The implication of this is that a malicious Pod provider can gain unauthorised access to a user's private Pod data.

P6 Albus is built on Solid which itself is composed of open standards. Additionally, the code for Albus will be released under the Open-source MIT license<sup>29</sup>.

P7 Albus is built with the user's needs at the centre. Albus makes no decisions for the benefit of application developers, service providers, Pod providers or any other parties other than its end users. The user's privacy is considered sacrosanct, and features that compromise it are not developed.

---

<sup>29</sup><https://opensource.org/licenses/MIT> [Accessed: 07-08-2019]

In summary, as per our evaluation, Albus fully complies with 6 out of the 7 “privacy by design” principles. We find that P5 is outside the scope of Albus and needs to be addressed by the platform, i.e., Solid, in this case.

### 5.2.3 DOSN classification

The classification of Albus based on the comparison features laid down in Table 2.4 on page 26 is given below:

1. **Topology:** Unstructured
2. **Data storage approach:** Semi Decentralised
3. **Privacy models:** Group based, Relationship based
4. **Privacy policy enforcement approach:** Trusted peers

## 5.3 Summary and Next Chapter

This chapter detailed the evaluation of Solid and Albus through previously established concepts. We found the weaknesses of the application, Albus, as well as those of the platform, Solid, and discussed the correlations between their weaknesses. The next chapter discusses these findings against the motivations of goals discussed in Chapter 1, and concludes this study.

# Chapter 6

## Discussion and Conclusions

This work looked into the privacy challenges posed by modern centralised OSNs and potential decentralised solutions. We also built a decentralised OSN using the Solid platform, and evaluated the developer experience as well as the application.

A review of literature indicated that not enough people care about privacy or most people do not care about privacy enough [6, 85–87]. This is primarily because a trade-off exists between usability-enhancing and privacy-preserving designs. Additionally, privacy and security are seen as non-functional software requirements [202] and hence, treated as an afterthought.

Moving from old style app development to Solid also involves completely reforming how we model and interact with data from protocols like SQL to RDF and SPARQL. The bigger challenge here is to reorient developers of traditional db-backed systems to this way of working with data.

There are also infrastructure-related and usability concerns which remain largely unsolved. Hosting Pods is in-feasible for regular people, while trusting providers is similar to trusting central servers of application providers. We explored a federated model in the academic domain, but it remains to be seen if such a model is scalable generally.

Apart from the availability challenge of Pod-hosting, Solid needs to overcome certain barriers in order to evolve a mature developer experience. The instability of the specification, buggy and inadequate tooling, and the lack of proper documentation are the top 3 concerns that deserve attention.



---

Finally, we would like to point out that there is some confusion in the web development community about the taxonomy to use for Solid, and a discussion<sup>1</sup> on the forum best represents the current status of the dialogue surrounding this issue. The question is, “what is Solid?” Or rather, where does it fit in our current – perhaps limited – vocabulary of classifying concepts in the software development and engineering field. First let’s just get the obvious out of the way: Solid is not a programming language. The other classifications that we consider are *framework* [203], *paradigm* [204], and *platform* [205].

Solid, is described by the official website<sup>2</sup> as “a proposed set of conventions and tools” [206]. As per the definitions on Wikipedia, we argue that Solid is a paradigm in the sense that it invokes “distinct concepts and thought patterns” [204] concerning web development, and also attempts to induce a “paradigm shift” [207] in how web applications design is approached. We also believe that while Solid itself does not constitute a platform, it specifies one, and NSS as an implementation of that specification is a platform, that provides services such as data management, access control and authentication through APIs. Finally, we submit that Solid in its present form does not constitute a framework, nor does a framework exist for enabling Solid application development. However, the React SDK for Solid represents an attempt at creating one out of lower level libraries<sup>3</sup>.

The above discussion is a summary of our answer to our research question we put forth in Section 1.4 on page 4. We conclude by summarising our contributions. By means of this work, we have produced a Solid application that is a social networking application for academics (C2) and also serves as an Open Science portal (C4). We have written a beginner’s guide to Solid for new developers (C1; See Part II). We have also discussed in detail the weaknesses to Solid, contributed to its development and stability by interacting with the community, raising issues, and fixing bugs (C3). In the next chapter, we also briefly consider the future developments in the Solid community. In summary, we posit that the Solid platform, which is the result of over a decade of

---

<sup>1</sup><https://forum.solidproject.org/t/newbie-perceptions-on-solid-app-framework-vs-web-standards-effort/844/4> [Accessed: 07-08-2019]

<sup>2</sup><https://solid.mit.edu/> [Accessed: 07-08-2019]

<sup>3</sup><http://web.archive.org/web/20190807202549/https://pragprog.com/magazines/2010-04/tangled-up-in-tools> [Accessed: 07-08-2019]

progress in the field of Semantic Web development, is a promising technology paradigm for decentralised web development. We have shed light on various aspects of Solid and highlighted the weaknesses that need to be worked upon in order to create a successful web development movement.

### **6.1 Threats to Validity**

The qualitative evaluation portion of this study is, by necessity, subjective, and is contingent on the ability of one person (the lead researcher) to gather sufficient evidence and information accurately, and the reasoning applied. While every care was taken to perform the evaluation with sound logic and without bias, it was not possible under prevailing restrictions to construct a more robust evaluation framework to obtain more empirical results. However, as a report of experience-based evidence, this study addresses a large number of aspects of Solid and development with it.

# Chapter 7

## Future work

A number of efforts towards the improvement of Solid and the development experience are underway. Some of these began after we completed the development and evaluation of this study.

The endeavour to use Solid in conjunction with efforts such as SAFE<sup>1</sup> is worthwhile and promising. This attempts to address the availability issue by providing an alternative network topology and data storage.

While some might argue that DOSNs are opposed to all forms of data mining, we believe that personalisation/adaptation without invading privacy or exposing user data can be achieved using the power of Linked data combined with decentralised machine learning [208–212]. As noted in Section 4.4 on page 44, Solid does not prescribe any hierarchy for storing containers and resources. This poses a challenge for application developers as application design heavily depends on where and how data is stored. Since application developers are left to define their own hierarchies, this can lead to data silos and redundancies within the Pod. There are currently no well-defined best practices for ensuring interoperability between applications. Verborgh’s recent blog [213] attempts to lay down the way forward for realising the promise of “the separation of data and apps” [213]. In the blog, Verborgh states that applications should not assume anything about the layout of data in the Pods. Current apps either make such assumptions or create their own layout in workspaces/namespaces corresponding to directories (called folders on Windows systems) on disk. Shape-aware applications would instead use

---

<sup>1</sup><https://safenetforum.org/t/solid-on-safe-updates/29318> [Accessed: 05-08-2019]

---

declarative shapes to build data-driven interfaces that would rely on technologies such as ShEx<sup>2</sup> and SHACL<sup>3</sup> [213].

It also came to our attention that another open-source implementation<sup>4</sup> of the Solid specification under the ambit of Inrupt<sup>5</sup> is being built. This is an important development, as having multiple implementations can facilitate better compliance and better innovation.

There's an imminent need to produce more learning resources. But these resources must acknowledge the gap in the capabilities of developers of traditional applications who do not know Linked Data. It's also important to understand the different classes of application developers the learning resources are targeted at. Some developers will have no knowledge of Linked Data, while some will have basic understanding of Linked Data while yet other may know the fundamentals of Solid. Using the same set of resources to teach such a diverse group of developers is not effective. Instead, we suggest production of learning resources for Solid application development at least 3 levels: Beginner, Intermediate, and Advanced. The first level should guide a developer through creation of a basic CRUD application using Solid using the very basic resources while not mentioning the more complicated tools such as Shapes at all. The guide should start with an introduction to the basic of Linked Data, and NSS. The Intermediate level should on-board the developer with some of the higher level tooling available such as `ldflex` and `rdflib.js` and should apply the newly introduced tools and practices to the application developed in Level 1. The concept of ACL should also be introduced and applied at this level. Finally, at the Advanced level, the developer is expected to understand Solid and Linked Data and the commonly used tools. At this point, the guide should introduce the best practices related to Shapes and any other advanced concepts deemed unnecessary at earlier levels. We recommend modelling tutorials in a similar manner to the C# tutorials on Microsoft's .NET website<sup>6</sup>, which are interactive and connected. The VueJs guide<sup>7</sup> is also a good example for introducing new concepts

---

<sup>2</sup><http://shex.io/shex-semantic/> [Accessed: 07-08-2019]

<sup>3</sup><https://www.w3.org/TR/2017/REC-shacl-20170720/> [Accessed: 07-08-2019]

<sup>4</sup><https://github.com/inrupt/pod-server> [Accessed: 07-08-2019]

<sup>5</sup><http://web.archive.org/web/20190807204241/https://inrupt.com/vision> [Accessed: 07-08-2019]

<sup>6</sup><https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/> [Accessed: 07-08-2019]

<sup>7</sup><https://vuejs.org/v2/guide/> [Accessed: 07-08-2019]

---

explicitly and gently before diving into code examples.

Security has been cited as an important requirement in DOSNs as we discussed in Section 2.4. We also noted that Solid currently, does not support encryption of Pods, and so complete end-to-end encryption is not possible. While early discussions on this topic<sup>8</sup> have been held in the community, there is no proposal in the specification to address this concern; NSS doesn't provide Pod encryption tools either. In the future, we would like to see the aspect of data security addressed in a more meaningful way.

Finally, future work on the application we built, Albus, should focus on improving usability by adding social features such as comments, likes, and collaboration tools. An optimisation of common case functions should also be carried out to improve data loading performance. We also recommend A/B testing [214] to test and improve the usability of the application.

---

<sup>8</sup><https://forum.solidproject.org/t/encrypted-pod-is-solid-designed-with-this-in-mind-if-not-would-it-be-possible-to-add/399/16> [Accessed: 07-08-2019]

# Bibliography

- [1] F. Fagerholm and J. Münch, “Developer experience: Concept and definition,” in *2012 International Conference on Software and System Process (ICSSP)*, pp. 73–77, June 2012. Accessed: 27-06-2019.
- [2] I. Steinmacher, M. A. Gerosa, and D. Redmiles, “Attracting, onboarding, and retaining newcomer developers in open source software projects,” 2 2014. Accessed: 18-07-2019.
- [3] V. V. H. Pham, S. Yu, K. Sood, and L. Cui, “Privacy issues in social networks and analysis: a comprehensive survey,” *IET networks*, vol. 7, no. 2, pp. 74–84, 2017. Accessed: 27-06-2019.
- [4] A. D. Salve, P. Mori, and L. Ricci, “A survey on privacy in decentralized online social networks,” *Computer Science Review*, vol. 27, pp. 154–176, 2018. Accessed: 23-07-2019.
- [5] Wikipedia, “Fediverse.” <https://en.wikipedia.org/wiki/Fediverse>. Accessed: 27-07-2019.
- [6] A. Heravi, S. Mubarak, and K.-K. R. Choo, “Information privacy in online social networks: Uses and gratification perspective,” *Computers in Human Behavior*, vol. 84, pp. 441–459, 2018. Accessed: 7-07-2019.
- [7] “Social networks - problems of security and data privacy background paper.” [http://www.cepis.org/files/cepis/20090901104125\\_CEPISsocialnetworkBackgroun.pdf](http://www.cepis.org/files/cepis/20090901104125_CEPISsocialnetworkBackgroun.pdf), Mar. 2008. Accessed: 27-06-2019.

## BIBLIOGRAPHY

---

- [8] I. Kayes and A. Iamnitchi, “Privacy and security in online social networks: A survey,” *Online Social Networks and Media*, vol. 3-4, pp. 1–21, 2017. Accessed: 4-07-2019.
- [9] X. Chen and K. Michael, “Privacy issues and solutions in social network sites,” *IEEE Technology and Society Magazine*, vol. 31, no. 4, pp. 43–53, 2012. Accessed: 4-07-2019.
- [10] J. Isaak and M. J. Hanna, “User data privacy: Facebook, cambridge analytica, and privacy protection,” *Computer*, vol. 51, no. 8, pp. 56–59, 2018. Accessed: 4-07-2019.
- [11] I. S. Rubinstein and N. Good, “Privacy by design: A counterfactual analysis of google and facebook privacy incidents,” *Berkeley Tech. LJ*, vol. 28, p. 1333, 2013. Accessed: 4-07-2019.
- [12] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Abounaga, and T. Berners-Lee, “A demonstration of the solid platform for social web applications,” in *Proceedings of the 25th International Conference Companion on World Wide Web*, pp. 223–226, International World Wide Web Conferences Steering Committee, 2016. Accessed: 4-07-2019.
- [13] L. Schwittmann, M. Wander, C. Boelmann, and T. Weis, “Privacy preservation in decentralized online social networks,” *IEEE Internet Computing*, vol. 18, pp. 16–23, Mar. 2014. Accessed: 4-07-2019.
- [14] O. L. Tim Berners-Lee, James Hendler, “The semantic web,” *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001. Accessed: 26-07-2019.
- [15] M. C. S. . A. I. Lab, “Web inventor tim berners-lee’s next project: a platform that gives users control of their data.” <https://web.archive.org/web/20190625202235/https://www.csail.mit.edu/news/web-inventor-tim-berners-lees-next-project-platform-gives-users-control-their-data>, 2015. Accessed: 2019-06-25.
- [16] Solid, “A definition of the culture around how decisions are made about solid and a record of how this has changed over time.” <https://github.com/solid/c>

## BIBLIOGRAPHY

---

- ulture/tree/00b1d130209775e4aff4e0da51715b1103e6c03e, 2019. Accessed: 2019-06-25.
- [17] C. Bizer, T. Heath, and T. Berners-Lee, “Linked data - the story so far,” *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, 2009. Accessed: 4-07-2019.
- [18] T. Berners-Lee, “The world wide web: A very short personal history.” <https://web.archive.org/web/20190625202157/https://www.w3.org/People/Berners-Lee/ShortHistory.html>, 1998. Accessed: 2019-06-25.
- [19] N. Shadbolt, T. Berners-Lee, and W. Hall, “The semantic web revisited,” *IEEE Intelligent Systems*, vol. 21, pp. 96–101, Jan. 2006. Accessed: 26-07-2019.
- [20] S. Aghaei, M. A. Nematbakhsh, and H. K. Farsani, “Evolution of the world wide web: From web 1.0 to web 4.0,” *International Journal of Web & Semantic Technology*, vol. 3, no. 1, p. 1, 2012. Accessed: 4-07-2019.
- [21] S. Murugesan, “Understanding web 2.0,” *IT Professional Magazine*, vol. 9, no. 4, p. 34, 2007. Accessed: 4-07-2019.
- [22] P. Miller, “Web 2.0: building the new library,” *Ariadne*, no. 45, 2005. Accessed: 4-07-2019.
- [23] S. Coppens, R. Verborgh, M. Vander Sande, D. Van Deursen, E. Mannens, and R. Van de Walle, “A truly read-write web for machines as the next-generation web?,” in *Proceedings of the SW2012 workshop: What will the Semantic Web look like*, vol. 10, 2012. Accessed: 4-07-2019.
- [24] R. Macmanus, “The read/write web.” [https://web.archive.org/web/20190627010533/https://readwrite.com/2003/04/19/the\\_readwrite\\_w/](https://web.archive.org/web/20190627010533/https://readwrite.com/2003/04/19/the_readwrite_w/), 2003. Accessed: 27-06-2019.
- [25] T. Berners-Lee, “Linked data.” <https://www.w3.org/DesignIssues/LinkedData.html>, 2006. Accessed: 27-06-2019.
- [26] T. Berners-Lee, “Read-write linked data.” <https://www.w3.org/DesignIssues/ReadWriteLinkedData.html>, 2009. Accessed: 27-06-2019.



## BIBLIOGRAPHY

---

- [27] I. Herman, “Provenance interchange working group charter.” <https://www.w3.org/2011/01/prov-wg-charter>, 2011. Accessed: 27-06-2019.
- [28] J. Hollenbach, J. Presbrey, and T. Berners-Lee, “Using rdf metadata to enable access control on the social semantic web,” in *Proceedings of the Workshop on Collaborative Construction, Management and Linking of Structured Knowledge (CK2009)*, vol. 514, p. 167, 2009. Accessed: 4-07-2019.
- [29] W3C, “Webaccesscontrol.” <https://www.w3.org/wiki/index.php?title=WebAccessControl&oldid=109272>, 2009. Accessed: 27-06-2019.
- [30] F. D. Davis, R. P. Bagozzi, and P. R. Warshaw, “User acceptance of computer technology: A comparison of two theoretical models,” *Management Science*, vol. 35, no. 8, pp. 982–1003, 1989.
- [31] E. Stiller and C. LeBlanc, “Effective software engineering pedagogy,” *J. Comput. Sci. Coll.*, vol. 17, pp. 124–134, May 2002. Accessed: 18-07-2019.
- [32] A. S. Kim and A. J. Ko, “A pedagogical analysis of online coding tutorials,” in *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, SIGCSE ’17, (New York, NY, USA), pp. 321–326, ACM, 2017. Accessed: 18-07-2019.
- [33] R. Watson, “Developing best practices for api reference documentation: Creating a platform to study how programmers learn new apis,” in *2012 IEEE International Professional Communication Conference*, pp. 1–9, Oct. 2012. Accessed: 22-07-2019.
- [34] A. Azimi and A. A. Ghomi, “Social networks privacy issues that affect young societies,” *Planetary Scientific Research Center Proceeding*, pp. 35–39, 2011. Accessed: 27-06-2019.
- [35] F. Fagerholm and J. Münch, “Developer experience: Concept and definition,” in *Proceedings of the International Conference on Software and System Process*, ICSSP ’12, (Piscataway, NJ, USA), pp. 73–77, IEEE Press, 2012. Accessed: 7-07-2019.

## BIBLIOGRAPHY

---

- [36] Solid, “The solid spec and architecture.” <https://github.com/solid/solid-spec/tree/103b1e027356bd525e4cad0138e8288f4881df39>, 2015. Accessed: 29-06-2019.
- [37] C. Wohlin, “Guidelines for snowballing in systematic literature studies and a replication in software engineering,” in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, p. 38, Citeseer, 2014. Accessed: 18-07-2019.
- [38] R. C. Martin, “Getting a solid start,” *Robert C. Martin-objectmentor.com*, 2013. Accessed: 7-08-2019.
- [39] C. Dannen, *Introducing Ethereum and Solidity*. Springer. Accessed: 7-08-2019.
- [40] C. Pérez-Sola, “Towards understanding privacy risks in online social networks,” 2016. Accessed: 7-07-2019.
- [41] D. M. Boyd and N. B. Ellison, “Social network sites: Definition, history, and scholarship,” *Journal of computer-mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007. Accessed: 27-06-2019.
- [42] M. Haumann, “Social media & privacy: A facebook case study,” 10 2015. Accessed: 4-07-2019.
- [43] R. Gross and A. Acquisti, “Information revelation and privacy in online social networks,” in *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*, pp. 71–80, ACM, 2005. Accessed: 27-06-2019.
- [44] H. Davies, “Ted cruz using firm that harvested data on millions of unwitting facebook users.” <http://web.archive.org/web/20190704174014/https://www.theguardian.com/us-news/2015/dec/11/senator-ted-cruz-president-campaign-facebook-user-data>, 2015. Accessed: 4-07-2019.
- [45] E. D. Craig Timberg, Tony Romm, “Zuckerberg apologizes, promises reform as senators grill him over facebook’s failings.” [https://www.washingtonpost.com/business/technology/2018/04/10/b72c09e8-3d03-11e8-974f-aacd97698cef\\_story.html](https://www.washingtonpost.com/business/technology/2018/04/10/b72c09e8-3d03-11e8-974f-aacd97698cef_story.html), 2018. Accessed: 4-07-2019.

## BIBLIOGRAPHY

---

- [46] D. M. Scott, *The New Rules of Marketing and PR.: How to Use Social Media, Online Video, Mobile Applications, Blogs, News Releases, and Viral Marketing to Reach Buyers Directly*. John Wiley & Sons, 2015. Accessed: 23-07-2019.
- [47] L. Bahri, B. Carminati, and E. Ferrari, “Decentralized privacy preserving services for online social networks,” *Online Social Networks and Media*, vol. 6, pp. 18–25, 2018. Accessed: 23-07-2019.
- [48] S. Yu, “Big privacy: Challenges and opportunities of privacy study in the age of big data,” *IEEE Access*, vol. 4, pp. 2751–2763, 2016. Accessed: 4-07-2019.
- [49] C. Fuchs, “An alternative view of privacy on facebook,” *Information*, vol. 2, no. 1, pp. 140–165, 2011. Accessed: 4-07-2019.
- [50] Y. Y. Guo, “The privacy issue on social network sites: Facebook,” *Journal of Digital Research & Publishing*, vol. 2, pp. 83–90, 2010. Accessed: 4-07-2019.
- [51] H. Nissenbaum, *Privacy in context: Technology, policy, and the integrity of social life*. Stanford University Press, 2009. Accessed: 4-07-2019.
- [52] A. F. Westin, “Privacy and freedom,” 1970. Accessed: 7-07-2019.
- [53] C. Fuchs, “The political economy of privacy on facebook,” *Television & New Media*, vol. 13, no. 2, pp. 139–159, 2012. Accessed: 4-07-2019.
- [54] K. Renaud and D. Gálvez-Cruz, “Privacy: Aspects, definitions and a multi-faceted privacy preservation approach,” in *2010 Information Security for South Africa*, pp. 1–8, Aug. 2010. Accessed: 4-07-2019.
- [55] C. Task and C. Clifton, “A guide to differential privacy theory in social network analysis,” in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 411–417, Aug. 2012. Accessed: 4-07-2019.
- [56] T. Zhu, G. Li, W. Zhou, and P. S. Yu, “Differentially private data publishing and analysis: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, pp. 1619–1638, Aug. 2017. Accessed: 4-07-2019.

## BIBLIOGRAPHY

---

- [57] R. Yu, J. Kang, X. Huang, S. Xie, Y. Zhang, and S. Gjessing, “Mixgroup: Accumulative pseudonym exchanging for location privacy enhancement in vehicular social networks,” *IEEE Transactions on Dependable and Secure Computing*, vol. 13, pp. 93–105, Jan. 2016. Accessed: 4-07-2019.
- [58] K. Mano, K. Minami, and H. Maruyama, “Pseudonym exchange for privacy-preserving publishing of trajectory data set,” in *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*, pp. 691–695, Oct. 2014. Accessed: 4-07-2019.
- [59] Yihui Lu, Wenjian Luo, and Dongdong Zhao, “Fast searching optimal negative surveys,” in *ICINS 2014 - 2014 International Conference on Information and Network Security*, pp. 82–90, Nov. 2014. Accessed: 4-07-2019.
- [60] R. Liu and S. Tang, “Multiple-negative survey method for enhancing the accuracy of negative survey-based cloud data privacy,” in *2015 International Workshop on Artificial Immune Systems (AIS)*, pp. 1–6, July 2015. Accessed: 4-07-2019.
- [61] W. Luo, Y. Lu, D. Zhao, and H. Jiang, “On location and trace privacy of the moving object using the negative survey,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, pp. 125–134, Apr. 2017. Accessed: 4-07-2019.
- [62] A. Diyanat, A. Khonsari, and S. P. Shariatpanahi, “A dummy-based approach for preserving source rate privacy,” *IEEE Transactions on Information Forensics and Security*, vol. 11, pp. 1321–1332, June 2016. Accessed: 4-07-2019.
- [63] Y. H. Gustav, X. Wu, Y. Ren, Y. Wang, and F. Zhang, “Dummy based privacy preservation in continuous querying road network services,” in *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 94–101, Oct. 2014. Accessed: 4-07-2019.
- [64] X. Wu and G. Sun, “A novel dummy-based mechanism to protect privacy on trajectories,” in *2014 IEEE International Conference on Data Mining Workshop*, pp. 1120–1125, Dec. 2014. Accessed: 4-07-2019.
- [65] A. Saxena, K. Saxena, and K. Chouhan, “Visual cryptography and concurrent slice-tracking with privacy preserving in data mining,” in *2016 International*

## BIBLIOGRAPHY

---

- Conference on ICT in Business Industry Government (ICTBIG)*, pp. 1–6, Nov. 2016. Accessed: 4-07-2019.
- [66] J. Han, Y. Liu, X. Sun, and L. Song, “Enhancing data and privacy security in mobile cloud computing through quantum cryptography,” in *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pp. 398–401, Aug. 2016. Accessed: 4-07-2019.
- [67] M. N. Sakib and C. Huang, “Privacy preserving proximity testing using elliptic curves,” in *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, pp. 121–126, Dec. 2016. Accessed: 4-07-2019.
- [68] A. K. Agrawal and S. Mehrotra, “Application of elliptic curve cryptography in pretty good privacy (pgp),” in *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 924–929, Apr. 2016. Accessed: 4-07-2019.
- [69] X. Wang, D. Feng, X. Lai, and H. Yu, “Collisions for hash functions md4, md5, haval-128 and ripemd.,” *IACR Cryptology ePrint Archive*, vol. 2004, p. 199, 2004. Accessed: 4-07-2019.
- [70] A. Sotirov, M. Stevens, J. Appelbaum, A. K. Lenstra, D. Molnar, D. A. Osvik, and B. de Weger, “Md5 considered harmful today, creating a rogue ca certificate,” in *25th Annual Chaos Communication Congress*, no. CONF, 2008. Accessed: 4-07-2019.
- [71] W. Liu, R. Luo, and H. Yang, “Cryptography overhead evaluation and analysis for wireless sensor networks,” in *2009 WRI International Conference on Communications and Mobile Computing*, vol. 3, pp. 496–501, IEEE, 2009. Accessed: 4-07-2019.
- [72] F. Raji, M. Davarpanah Jazi, and A. Miri, “Pesca: a peer-to-peer social network architecture with privacy-enabled social communication and data availability,” *IET Information Security*, vol. 9, no. 1, pp. 73–80, 2015. Accessed: 4-07-2019.
- [73] P. Stuedi, I. Mohomed, M. Balakrishnan, Z. M. Mao, V. Ramasubramanian, D. Terry, and T. Wobber, “Contrail: Decentralized and privacy-preserving social

## BIBLIOGRAPHY

---

- networks on smartphones,” *IEEE Internet Computing*, vol. 18, pp. 44–51, Sept. 2014. Accessed: 4-07-2019.
- [74] Yongquan Fu and Yijie Wang, “Bce: A privacy-preserving common-friend estimation method for distributed online social networks without cryptography,” in *7th International Conference on Communications and Networking in China*, pp. 212–217, Aug. 2012. Accessed: 4-07-2019.
- [75] J. R. Douceur, “The sybil attack,” in *Peer-to-Peer Systems* (P. Druschel, F. Kaashoek, and A. Rowstron, eds.), (Berlin, Heidelberg), pp. 251–260, Springer Berlin Heidelberg, 2002. Accessed: 4-07-2019.
- [76] P. P. Swire, R. E. Litan, and R. E. Litan, *None of your business: world data flows, electronic commerce, and the European privacy directive*. Brookings Institution Press, 1998. Accessed: 4-07-2019.
- [77] L. A. Cutillo, R. Molva, and T. Strufe, “Privacy preserving social networking through decentralization,” in *2009 Sixth International Conference on Wireless On-Demand Network Systems and Services*, pp. 145–152, Feb. 2009. Accessed: 27-06-2019.
- [78] L. Cheng, F. Liu, and D. D. Yao, “Enterprise data breach: causes, challenges, prevention, and future directions,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 5, 2017. Accessed: 4-07-2019.
- [79] S. Mansfield-Devine, “Anti-social networking: exploiting the trusting environment of web 2.0,” *Network Security*, vol. 2008, no. 11, pp. 4–7, 2008. Accessed: 4-07-2019.
- [80] H. Ko, S. Kim, and S. Jin, “Usability enhanced privacy protection system based on users’ responses,” in *2007 IEEE International Symposium on Consumer Electronics*, pp. 1–6, June 2007. Accessed: 4-07-2019.
- [81] J. T. Lehtikoinen, T. Olsson, and H. Toivola, “Privacy regulation in online social interaction,” *Proceedings of IADIS*, pp. 25–32, 2008. Accessed: 4-07-2019.
- [82] M. S. Granovetter, “The strength of weak ties,” in *Social networks*, pp. 347–367, Elsevier, 1977. Accessed: 4-07-2019.

## BIBLIOGRAPHY

---

- [83] M. Granovetter, “The strength of weak ties: A network theory revisited,” 1983. Accessed: 4-07-2019.
- [84] R. S. Burt, *Structural holes: The social structure of competition*. Harvard university press, 2009. Accessed: 4-07-2019.
- [85] G. Homans, “Social behaviour as exchange-the american journal of sociology,” 1958. Accessed: 4-07-2019.
- [86] J. Johanson and L.-G. Mattsson, “Interorganizational relations in industrial systems: a network approach compared with the transaction-cost approach,” *International Studies of Management & Organization*, vol. 17, no. 1, pp. 34–48, 1987. Accessed: 4-07-2019.
- [87] J. Grimmelmann, “Saving facebook,” *Iowa L. Rev.*, vol. 94, p. 1137, 2008. Accessed: 7-07-2019.
- [88] G. Blosser and J. Zhan, “Privacy preserving collaborative social network,” in *2008 International Conference on Information Security and Assurance (isa 2008)*, pp. 543–548, Apr. 2008. Accessed: 4-07-2019.
- [89] C. C. Yang, “Information sharing and privacy protection of terrorist or criminal social networks,” in *2008 IEEE International Conference on Intelligence and Security Informatics*, pp. 40–45, June 2008. Accessed: 4-07-2019.
- [90] A. Cavoukian, “Privacy by design-the 7 foundational principles (2011).” <https://www.ipc.on.ca/wp-content/uploads/Resources/7foundationalprinciples.pdf>, 2011. Accessed: 7-07-2019.
- [91] M. Keeling, “Reflections on software engineering,” 2010. Accessed: 7-07-2019.
- [92] “Privacy guidelines for developing software products and services,” 2008. Accessed: 7-07-2019.
- [93] H. Kelly, “Police embrace social media as crime-fighting tool,” *CNN. com*, vol. 30, 2012. Accessed: 7-07-2019.

## BIBLIOGRAPHY

---

- [94] G. Lotan, E. Graeff, M. Ananny, D. Gaffney, I. Pearce, and danah boyd, “The arab spring - the revolutions were tweeted: Information flows during the 2011 tunisian and egyptian revolutions,” *International Journal of Communication*, vol. 5, no. 0, p. 31, 2011. Accessed: 7-07-2019.
- [95] P. Jha, “Facebook users could swing the results in 160 lok sabha constituencies,” *The Hindu*, 2013. Accessed: 7-07-2019.
- [96] G. D. P. Regulation, “Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46,” *Official Journal of the European Union (OJ)*, vol. 59, no. 1-88, p. 294, 2016. Accessed: 7-07-2019.
- [97] A. Romanou, “The necessity of the implementation of privacy by design in sectors where data protection concerns arise,” *Computer Law & Security Review*, vol. 34, no. 1, pp. 99 – 110, 2018. Accessed: 7-07-2019.
- [98] I. O. for Standardization, *Ergonomics of human-system interaction: Part 210: Human-centred design for interactive systems*. ISO, 2010. Accessed: 23-07-2019.
- [99] A. Palmer, “Customer experience management: a critical review of an emerging idea,” *Journal of Services marketing*, vol. 24, no. 3, pp. 196–208, 2010. Accessed: 23-07-2019.
- [100] “American marketing association dictionary.” [http://www.marketingpower.com/\\_layouts/Dictionary.aspx?dLetter=B](http://www.marketingpower.com/_layouts/Dictionary.aspx?dLetter=B). Accessed: 23-07-2019.
- [101] A. Fontão, A. Dias-Neto, and D. Viana, “Investigating factors that influence developers’ experience in mobile software ecosystems,” in *2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)*, pp. 55–58, May 2017. Accessed: 23-07-2019.
- [102] C. Suebsin and N. Gerd Sri, “Key factors driving the success of technology adoption: Case examples of erp adoption,” in *PICMET '09 - 2009 Portland Inter-*



## BIBLIOGRAPHY

---

- national Conference on Management of Engineering Technology*, pp. 2638–2643, Aug. 2009. Accessed: 27-06-2019.
- [103] K. Lakhani and R. G. Wolf, “Why hackers do what they do: Understanding motivation and effort in free/open source software projects,” *Perspectives on Free and Open Source Software*, 9 2003. Accessed: 23-07-2019.
- [104] A. Hars and S. Ou, “Working for free? motivations for participating in open-source projects,” *International Journal of Electronic Commerce*, vol. 6, pp. 25–39, 3 2002. Accessed: 23-07-2019.
- [105] S. Shah, “Motivation, governance, and the viability of hybrid forms in open source software development,” *Management Science*, vol. 52, pp. 1000–1014, 7 2006. Accessed: 23-07-2019.
- [106] C. Santos Jr, g. kuk, F. Kon, and J. Pearson, “The attraction of contributors in free and open source software,” *The Journal of Strategic Information Systems*, vol. 22, p. 26–45, 3 2013. Accessed: 23-07-2019.
- [107] P. Meirelles, C. Santos Jr, J. Miranda, F. Kon, A. Terceiro, and C. Chavez, “A study of the relationships between source code metrics and attractiveness in free software projects,” pp. 11–20, 11 2010. Accessed: 23-07-2019.
- [108] I. Chengalur-Smith, A. Sidorova, and S. Daniel, “Sustainability of free/libre open source projects: A longitudinal study,” *Journal of the Association for Information Systems*, vol. 11, 11 2010. Accessed: 23-07-2019.
- [109] C. Jensen, S. King, and V. Kuechler, “Joining free/open source software communities: An analysis of newbies’ first interactions on project mailing lists,” pp. 1–10, 2 2011. Accessed: 23-07-2019.
- [110] I. Steinmacher, I. Wiese, A. P. Chaves Steinmacher, and M. A. Gerosa, “Why do newcomers abandon open source software projects?,” 5 2013. Accessed: 23-07-2019.
- [111] V. Midha, P. Palvia, R. Singh, and N. Kshetri, “Improving open source software maintenance,” *Journal of Computer Information Systems*, vol. 50, pp. 81–90, 3 2010. Accessed: 23-07-2019.

## BIBLIOGRAPHY

---

- [112] Y. Fang and D. Neufeld, “Understanding sustained participation in open source software projects,” *Journal of Management Information Systems*, vol. 25, no. 4, pp. 9–50, 2009. Accessed: 18-07-2019.
- [113] M. P. Robillard and R. DeLine, “A field study of api learning obstacles,” *Empirical Software Engineering*, vol. 16, pp. 703–732, Dec. 2011. Accessed: 22-07-2019.
- [114] J. Nykaza, R. Messinger, F. Boehme, C. L. Norman, M. Mace, and M. Gordon, “What programmers really want: Results of a needs assessment for sdk documentation,” in *Proceedings of the 20th Annual International Conference on Computer Documentation*, SIGDOC ’02, (New York, NY, USA), pp. 133–141, ACM, 2002. Accessed: 22-07-2019.
- [115] T. R. Lister and T. DeMarco, *Peopeware: Productive projects and teams*. Dorset House New York, 1987. Accessed: 22-07-2019.
- [116] J. Spolsky, *Joel on Software: And on Diverse and Occasionally Related Matters That Will Prove of Interest to Software Developers, Designers, and Managers, and to Those Who, Whether by Good Fortune or Ill Luck, Work with Them in Some Capacity*. Apress, Berkeley, CA, 2004. Accessed: 22-07-2019.
- [117] T. DeMarco and T. Lister, “Programmer performance and the effects of the workplace,” in *Proceedings of the 8th International Conference on Software Engineering*, ICSE ’85, (Los Alamitos, CA, USA), pp. 268–272, IEEE Computer Society Press, 1985. Accessed: 22-07-2019.
- [118] J. Piaget, *Success and understanding*. 1 1978. Accessed: 23-07-2019.
- [119] J. Piaget, *The Origins of Intelligence In Children*. 1 1952. Accessed: 23-07-2019.
- [120] J. D. Bransford, A. Brown, and R. R. Cocking, *How People Learn: Mind, Brain, Experience and School*. 1 1999. Accessed: 23-07-2019.
- [121] S. A. Ambrose, M. W. Bridges, M. DiPietro, M. C. Lovett, and M. K. Norman, *How learning works: Seven research-based principles for smart teaching*. John Wiley & Sons, 2010. Accessed: 23-07-2019.

## BIBLIOGRAPHY

---

- [122] K. A. Ericsson, R. T. Krampe, and C. Tesch-Römer, “The role of deliberate practice in the acquisition of expert performance.,” *Psychological review*, vol. 100, no. 3, p. 363, 1993. Accessed: 23-07-2019.
- [123] J. Flavell, “Metacognitive aspects of problem solving,” *Nat. Intell*, vol. 12, pp. 231–235, 1 1976. Accessed: 23-07-2019.
- [124] A. S. Palincsar and A. L. Brown, “Reciprocal teaching of comprehension-monitoring activities,” *Center for the Study of Reading Technical Report; no. 269*, 1983. Accessed: 23-07-2019.
- [125] B. Vasilescu, V. Filkov, and A. Serebrenik, “Stackoverflow and github: Associations between software development and crowdsourced knowledge,” in *2013 International Conference on Social Computing*, pp. 188–195, Sept. 2013. Accessed: 18-07-2019.
- [126] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, “What makes a good code example?: A study of programming q a in stackoverflow,” in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pp. 25–34, Sept. 2012. Accessed: 18-07-2019.
- [127] C. B. Seaman, “The information gathering strategies of software maintainers,” in *International Conference on Software Maintenance, 2002. Proceedings.*, pp. 141–149, Oct. 2002. Accessed: 22-07-2019.
- [128] J. Brandt, P. J. Guo, J. Lewenstein, M. Dontcheva, and S. R. Klemmer, “Two studies of opportunistic programming: interleaving web foraging, learning, and writing code,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1589–1598, ACM, 2009. Accessed: 22-07-2019.
- [129] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, “Design lessons from the fastest q&a site in the west,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 2857–2866, ACM, 2011. Accessed: 22-07-2019.

## BIBLIOGRAPHY

---

- [130] C. Treude, O. Barzilay, and M. Storey, “How do programmers ask and answer questions on the web?: Nier track,” in *2011 33rd International Conference on Software Engineering (ICSE)*, pp. 804–807, May 2011. Accessed: 18-07-2019.
- [131] C. Parnin, C. Treude, L. Grammel, and M.-A. Storey, “Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow,” *Georgia Institute of Technology, Tech. Rep.*, vol. 11, 2012. Accessed: 18-07-2019.
- [132] A. Bacchelli, L. Ponzanelli, and M. Lanza, “Harnessing stack overflow for the ide,” in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering, RSSE ’12*, (Piscataway, NJ, USA), pp. 26–30, IEEE Press, 2012. Accessed: 22-07-2019.
- [133] J. Cordeiro, B. Antunes, and P. Gomes, “Context-based search to overcome learning barriers in software development,” in *Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering, RAISE ’12*, (Piscataway, NJ, USA), pp. 47–51, IEEE Press, 2012. Accessed: 22-07-2019.
- [134] F. Thung, T. F. Bissyandé, D. Lo, and L. Jiang, “Network structure of social coding in github,” in *2013 17th European Conference on Software Maintenance and Reengineering*, pp. 323–326, Mar. 2013. Accessed: 22-07-2019.
- [135] A. Lima, L. Rossi, and M. Musolesi, “Coding together at scale: Github as a collaborative social network,” in *Eighth International AAAI Conference on Weblogs and Social Media*, 2014. Accessed: 22-07-2019.
- [136] P. Cobb, “Theories of mathematical learning and constructivism: A personal view,” in *Symposium on Trends and Perspectives in Mathematics Education, Institute for Mathematics, University of Klagenfurt, Austria*, 1994. Accessed: 23-07-2019.
- [137] P. Cobb, E. Yackel, and T. Wood, “A constructivist alternative to the representational view of mind in mathematics education,” *Journal for Research in Mathematics Education*, vol. 23, 1 1992. Accessed: 23-07-2019.
- [138] L. S. Vygotsky, *Mind in society: The development of higher psychological processes*. Harvard university press, 1980. Accessed: 23-07-2019.

## BIBLIOGRAPHY

---

- [139] L. S. Vygotsky, “Thought and language,” *Bulletin of the Orton Society*, vol. 14, pp. 97–98, Dec. 1964. Accessed: 23-07-2019.
- [140] C. Johnson, M. McGill, D. Bouchard, M. K. Bradshaw, V. A. Bucheli, L. D. Merkle, M. J. Scott, Z. Sweedyk, J. A. Velázquez-Iturbide, Z. Xiao, and M. Zhang, “Game development for computer science education,” in *Proceedings of the 2016 ITiCSE Working Group Reports*, ITiCSE ’16, (New York, NY, USA), pp. 23–44, ACM, 2016. Accessed: 7-07-2019.
- [141] L. A. Cutillo, R. Molva, and T. Strufe, “Safebook: A privacy-preserving online social network leveraging on real-life trust,” *IEEE Communications Magazine*, vol. 47, pp. 94–101, Dec. 2009. Accessed: 23-07-2019.
- [142] S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta, “Peerson: P2p social networking: Early experiences and insights,” in *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS ’09, (New York, NY, USA), pp. 46–52, ACM, 2009. Accessed: 26-07-2019.
- [143] L. M. Aiello and G. Ruffo, “Lotusnet: Tunable privacy for distributed online social network services,” *Computer Communications*, vol. 35, no. 1, pp. 75–88, 2012. Accessed: 26-07-2019.
- [144] R. Sharma and A. Datta, “Supernova: Super-peers based architecture for decentralized online social networks,” in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, pp. 1–10, Jan. 2012. Accessed: 26-07-2019.
- [145] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz, “Lifesocial.kom: A secure and p2p-based solution for online social networks,” in *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pp. 554–558, Jan. 2011. Accessed: 26-07-2019.
- [146] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia, “Cachet: A decentralized architecture for privacy preserving social networking with caching,” in *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT ’12, (New York, NY, USA), pp. 337–348, ACM, 2012. Accessed: 26-07-2019.

## BIBLIOGRAPHY

---

- [147] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, “Persona: An online social network with user-defined privacy,” in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM ’09, (New York, NY, USA), pp. 135–146, ACM, 2009. Accessed: 26-07-2019.
- [148] M. Dürr, M. Maier, and F. Dorfmeister, “Vegas – a secure and privacy-preserving peer-to-peer online social network,” in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*, pp. 868–874, Sept. 2012. Accessed: 26-07-2019.
- [149] N. Kourtellis, J. Finnis, P. Anderson, J. Blackburn, C. Borcea, and A. Iamnitchi, “Prometheus: User-controlled p2p social data management for socially-aware applications,” in *Middleware 2010* (I. Gupta and C. Mascolo, eds.), (Berlin, Heidelberg), pp. 212–231, Springer Berlin Heidelberg, 2010. Accessed: 26-07-2019.
- [150] F. Tegeler, D. Koll, and X. Fu, “Gemstone: Empowering decentralized social networking with high data availability,” in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, pp. 1–6, Dec. 2011. Accessed: 26-07-2019.
- [151] D. Koll, J. Li, and X. Fu, “Soup: An online social network by the people, for the people,” in *Proceedings of the 15th International Middleware Conference*, Middleware ’14, (New York, NY, USA), pp. 193–204, ACM, 2014. Accessed: 26-07-2019.
- [152] S. Biedermann, N. P. Karvelas, S. Katzenbeisser, T. Strufe, and A. Peter, “Proofbook: An online social network based on proof-of-work and friend-propagation,” in *SOFSEM 2014: Theory and Practice of Computer Science* (V. Geffert, B. Preneel, B. Rován, J. Štuller, and A. M. Tjoa, eds.), (Cham), pp. 114–125, Springer International Publishing, 2014. Accessed: 26-07-2019.
- [153] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, “Decent: A decentralized architecture for enforcing privacy in online social networks,” in *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 326–332, Mar. 2012. Accessed: 26-07-2019.

## BIBLIOGRAPHY

---

- [154] D. Koll, D. Lechler, and X. Fu, “Socialgate: Managing large-scale social data on home gateways,” in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pp. 1–6, Oct. 2017. Accessed: 26-07-2019.
- [155] I. Z. R.S.D. GRippi, M. Salzberg, “Diaspora\*.” <https://diasporafoundation.org/>, 2012. Accessed: 26-07-2019.
- [156] A. Shakimov, H. Lim, R. Cáceres, L. P. Cox, K. Li, D. Liu, and A. Varshavsky, “Vis-à-vis: Privacy-preserving online social networking via virtual individual servers,” in *2011 Third International Conference on Communication Systems and Networks (COMSNETS 2011)*, pp. 1–10, Jan. 2011. Accessed: 26-07-2019.
- [157] R. Narendula, T. G. Papaioannou, and K. Aberer, “My3: A highly-available p2p-based online social network,” in *2011 IEEE International Conference on Peer-to-Peer Computing*, pp. 166–167, Aug. 2011. Accessed: 26-07-2019.
- [158] A. Loupasakis, N. Ntarmos, and P. Triantafillou, “exo: Decentralized autonomous scalable social networking.” pp. 85–95, 1 2011. Accessed: 26-07-2019.
- [159] B. Guidi, T. Amft, A. De Salve, K. Graffi, and L. Ricci, “Didusonet: A p2p architecture for distributed dunbar-based social networks,” *Peer-to-Peer Networking and Applications*, vol. 9, pp. 1177–1194, Nov. 2016. Accessed: 26-07-2019.
- [160] MrPetovan, “Happy 9th birthday friendica!” <https://friendi.ca/2019/07/01/happy-9th-birthday-friendica/>, 2019. Accessed: 26-07-2019.
- [161] E. Revah, “Mistpark - distributed social networking (that works).” <https://manurevah.com/blah/en/blog/Mistpark-Distributed-Social-Networkin>, 2010. Accessed: 26-07-2019.
- [162] “Retrosahre.” <http://retrosahre.cc/index.html>. Accessed: 26-07-2019.
- [163] “Retrosahre aims to be a private f2f social network.” <https://sourceforge.net/blog/retrosahre-aims-to-be-a-private-f2f-social-network>, 2010. Accessed: 26-07-2019.
- [164] F. Raji, A. Miri, M. Davarpanah Jazi, and B. Malek, “Online social network with flexible and dynamic privacy policies,” in *2011 CSI International Symposium on*

## BIBLIOGRAPHY

---

- Computer Science and Software Engineering (CSSE)*, pp. 135–142, June 2011. Accessed: 26-07-2019.
- [165] B. Greschbach, G. Kreitz, and S. Buchegger, “The devil is in the metadata — new privacy challenges in decentralised online social networks,” in *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 333–339, Mar. 2012. Accessed: 26-07-2019.
- [166] L. Schwittmann, C. Boelmann, M. Wander, and T. Weis, “Sonet – privacy and replication in federated online social networks,” in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pp. 51–57, July 2013. Accessed: 26-07-2019.
- [167] B. Malek and A. Miri, “Adaptively secure broadcast encryption with short ciphertexts.” Accessed: 26-07-2019.
- [168] D. Froelicher, P. Egger, J. S. Sousa, J. L. Raisaro, Z. Huang, C. Mouchet, B. Ford, and J.-P. Hubaux, “Unlynx: a decentralized system for privacy-conscious data sharing,” *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 4, pp. 232–250, 2017. Accessed: 23-07-2019.
- [169] M. Zignani, S. Gaito, and G. P. Rossi, “Follow the “mastodon”: Structure and evolution of a decentralized online social network,” in *Twelfth International AAAI Conference on Web and Social Media*, 2018. Accessed: 26-07-2019.
- [170] E. S. A. G. E. P. Christopher Lemmer Webber, Jessica Tallon, “Activitypub.” <https://www.w3.org/TR/2018/REC-activitypub-20180123/>, 2018. Accessed: 27-07-2019.
- [171] J. Holloway, “What on earth is the fediverse and why does it matter?.” <http://web.archive.org/web/20190727012123/https://newatlas.com/what-is-the-fediverse/56385/>, 2018. Accessed: 27-07-2019.
- [172] S. Göndör and A. Küpper, “The current state of interoperability in decentralized online social networking services,” in *2017 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 852–857, Dec. 2017. Accessed: 26-07-2019.



## BIBLIOGRAPHY

---

- [173] PeerTube, “Take back control of your videos.” <https://joinpeertube.org/en/>. Accessed: 27-07-2019.
- [174] diaspora, “diaspora\* federation library.” [https://github.com/diaspora/diaspora\\_federation/tree/59a983b5eb7b07f3bbd8ea5685d842fa29274611](https://github.com/diaspora/diaspora_federation/tree/59a983b5eb7b07f3bbd8ea5685d842fa29274611). Accessed: 27-07-2019.
- [175] S. Buchegger and A. Datta, “A case for p2p infrastructure for social networks - opportunities challenges,” in *2009 Sixth International Conference on Wireless On-Demand Network Systems and Services*, pp. 161–168, Feb. 2009. Accessed: 23-07-2019.
- [176] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, “Armor: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks,” *Future Generation Computer Systems*, vol. 79, pp. 82–94, 2018. Accessed: 23-07-2019.
- [177] K. Anand, J. Kumar, and K. Anand, “Anomaly detection in online social network: A survey,” in *2017 International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 456–459, Mar. 2017. Accessed: 26-07-2019.
- [178] A. Sambra, A. Guy, S. Capadisli, and N. Greco, “Building decentralized applications for the social web,” in *Proceedings of the 25th International Conference Companion on World Wide Web, WWW '16 Companion*, (Republic and Canton of Geneva, Switzerland), pp. 1033–1034, International World Wide Web Conferences Steering Committee, 2016. Accessed: 29-07-2019.
- [179] G. Klyne, J. J. Carroll, and B. McBride, “Resource description framework (rdf): Concepts and abstract syntax. w3c recommendation, feb. 2004,” 2004. Accessed: 29-07-2019.
- [180] Inrupt, “How it works.” <http://web.archive.org/web/20190729212753/http://solid.inrupt.com/how-it-works>. Accessed: 29-07-2019.

## BIBLIOGRAPHY

---

- [181] Solid, “Content representation.” <https://github.com/solid/solid-spec/tree/103b1e027356bd525e4cad0138e8288f4881df39#content-representation>. Accessed: 30-07-2019.
- [182] Solid, “Web access control (wac).” <https://github.com/solid/web-access-control-spec/tree/a71580b46a3ff124fa72d765a90432e488e96260>. Accessed: 30-07-2019.
- [183] W3C, “Linked data notifications.” <https://www.w3.org/TR/2017/REC-ldn-20170502/>, 2017. Accessed: 30-07-2019.
- [184] Solid, “solid-apps.” <https://github.com/solid/solid-apps/blob/f1f568ad1b77ad9f38da8a1b1c0e9255c3390d78/README.md>, 2019. Accessed: 30-07-2019.
- [185] S. Capadisli, A. Guy, R. Verborgh, C. Lange, S. Auer, and T. Berners-Lee, “Decentralised authoring, annotations and notifications for a read-write web with dokieli,” in *Web Engineering* (J. Cabot, R. De Virgilio, and R. Torlone, eds.), (Cham), pp. 469–481, Springer International Publishing, 2017. Accessed: 30-07-2019.
- [186] C. R. Kothari, *Research methodology: Methods and techniques*. New Age International, 2004. Accessed: 18-07-2019.
- [187] D. Remenyi, B. Williams, A. Money, and E. Swartz, *Doing research in business and management: an introduction to process and method*. Sage, 1998. Accessed: 18-07-2019.
- [188] S. W. Littlejohn and K. A. Foss, *Encyclopedia of communication theory*, vol. 1. Sage, 2009. Accessed: 30-07-2019.
- [189] M. D. Myers, *Qualitative research in business and management*. Sage, 2013. Accessed: 30-07-2019.
- [190] D. R. Monette, T. J. Sullivan, and C. R. DeJong, *Applied social research: A tool for the human services*. Nelson Education, 2013. Accessed: 18-07-2019.

## BIBLIOGRAPHY

---

- [191] M. J. Polonsky and D. S. Waller, *Designing and managing a research project: A business student's guide*. Sage publications, 2018. Accessed: 18-07-2019.
- [192] M. N. Saunders, *Research methods for business students, 5/e*. Pearson Education India, 2011. Accessed: 18-07-2019.
- [193] K. Singh, *Quantitative social research methods*. Sage, 2007. Accessed: 18-07-2019.
- [194] W. Goddard and S. Melville, *Research methodology: An introduction*. Juta and Company Ltd, 2004. Accessed: 18-07-2019.
- [195] H. R. Bernard, *Research methods in anthropology: Qualitative and quantitative approaches*. Rowman & Littlefield, 2017. Accessed: 18-07-2019.
- [196] D. L. Jorgensen, *Participant Observation*, pp. 1–15. American Cancer Society, 2015. Accessed: 18-07-2019.
- [197] B. Kawulich, “Participant observation as a data collection method,” *Forum Qualitative Sozialforschung / Forum: Qualitative Social Research*, vol. 6, no. 2, 2005. Accessed: 18-07-2019.
- [198] J. O. Gilliam, “Improving the open source software model with uml case tools,” *Linux Gazette*, vol. 67, 2001. Accessed: 30-07-2019.
- [199] M. Fowler and M. Foemmel, “Continuous integration,” *Thought-Works*) [http://www.thoughtworks.com/Continuous Integration.pdf](http://www.thoughtworks.com/Continuous%20Integration.pdf), vol. 122, p. 14, 2006. Accessed: 30-07-2019.
- [200] R. Verborgh, “Designing a linked data developer experience.” <http://web.archive.org/web/20190805204059/https://ruben.verborgh.org/blog/2018/12/28/designing-a-linked-data-developer-experience/>, 2018. Accessed: 5-08-2019.
- [201] J. Tsay, L. Dabbish, and J. Herbsleb, “Influence of social and technical factors for evaluating contribution in github,” in *Proceedings of the 36th International Conference on Software Engineering, ICSE 2014*, (New York, NY, USA), pp. 356–366, ACM, 2014. Accessed: 6-08-2019.

## BIBLIOGRAPHY

---

- [202] L. Liu, E. Yu, and J. Mylopoulos, “Security and privacy requirements analysis within a social setting,” in *Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, pp. 151–161, Sept. 2003. Accessed: 7-07-2019.
- [203] Wikipedia, “Software framework.” [https://en.wikipedia.org/w/index.php?title=Software\\_framework&oldid=901960229](https://en.wikipedia.org/w/index.php?title=Software_framework&oldid=901960229). Accessed: 7-08-2019.
- [204] Wikipedia, “Paradigm.” <https://en.wikipedia.org/w/index.php?title=Paradigm&oldid=908508732>. Accessed: 7-08-2019.
- [205] Wikipedia, “Computing platform.” [https://en.wikipedia.org/w/index.php?title=Computing\\_platform&oldid=909644295](https://en.wikipedia.org/w/index.php?title=Computing_platform&oldid=909644295). Accessed: 7-08-2019.
- [206] Solid, “Solid.” <http://web.archive.org/web/20190807201013/https://solid.mit.edu/>. Accessed: 7-08-2019.
- [207] T. S. Kuhn, “The structure of scientific revolutions,” *Chicago and London*, 1962. Accessed: 7-08-2019.
- [208] A. Segal, A. Marcedone, B. Kreuter, D. Ramage, H. B. McMahan, K. Seth, K. Bonawitz, S. Patel, and V. Ivanov, “Practical secure aggregation for privacy-preserving machine learning,” in *CCS*, 2017. Accessed: 26-07-2019.
- [209] B. McMahan and R. S. Daniel Ramage, “Federated learning: Collaborative machine learning without centralized training data.” <http://web.archive.org/web/20190727012435/https://ai.googleblog.com/2017/04/federated-learning-collaborative.html>, 2017. Accessed: 27-07-2019.
- [210] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Konečný, S. Mazzocchi, B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander, “Towards federated learning at scale: System design,” in *SysML 2019*, 2019. Accessed: 26-07-2019.
- [211] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017. Accessed: 26-07-2019.

## BIBLIOGRAPHY

---

- [212] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. Accessed: 26-07-2019.
- [213] R. Verborgh, “Shaping linked data apps.” <https://web.archive.org/web/20190703174036/https://ruben.verborgh.org/blog/2019/06/17/shaping-linked-data-apps/>, 2019. Accessed: 3-07-2019.
- [214] R. Kohavi and R. Longbotham, *Online Controlled Experiments and A/B Testing*, pp. 922–929. Boston, MA: Springer US, 2017. Accessed: 7-08-2019.

# Appendix A

## Development Journal

The development journal I maintained is reproduced below. Note that it does not adhere to a formal structure or writing style, nor should it be taken to contain accurate information. The journal represents my leanings and thoughts as I worked on this project, and is included in the interest of completeness and verifiability.

### A.1 Background

#### A.1.1 20-12-2018

I've started working with Solid only recently as a part of my MSc Dissertation project which is aimed at developing a decentralised social network for academics powered by Solid. By building a real-world application based on Solid, I intend to test out its readiness for practical web app development as well as to improve Solid and assess the developer experience of building a Solid application with a view to evolve best practices especially for new developers interested in building Solid applications.

The most obvious thing to do as a beginner is to access the official documentation and complete the *hello world* tutorial. Having done this kind of thing many times over through my career, I'm quite well-versed with the process of learning a new technology. This time, however, I will be taking a continuous documentation and reflection approach with a view to evaluate how easy or hard it was for me to bootstrap as a newcomer to Solid. So, to start, I identified the following initial steps:

1. Run through the official documentation <https://solid.inrupt.com/docs/getting-started>.
2. Sign-up for a pod to get a feel of the interface: <https://solid.inrupt.com/get-a-solid-pod>
3. Run the “hello world” example: <https://solid.inrupt.com/docs/writing-solid-apps-with-angular>
4. Set-up a local Solid server: <https://github.com/solid/node-solid-server>

I dived into these steps after first reading a little about Solid to get an idea about what it really is. This article (<https://www.inrupt.com/blog/one-small-step-for-the-web>) by TBL helped understand the motivation, but Solid was still an abstract concept. What stood out was decentralisation, data privacy, data ownership, permission control, and linked data/semantic web. It still wasn't clear how data ownership and decentralisation is achieved. Reading the docs provided some insight and the concept of a Pod was introduced.

The docs provided some clarity but introduced more doubts. Of course, this is the nature of learning. Over the next few months I intend to understand the full power and limits of Solid.

## A.2 First Contact

### A.2.1 20-12-2018

After reading the docs and articles about Solid, I decided to dive into the code. First, I read the tutorial (<https://solid.inrupt.com/docs/writing-solid-apps-with-angular>) to understand pre-requisites. What was immediately apparent was:

1. The tutorial presumes a lot of knowledge:
  - web development
  - Node.js
  - Angular
  - Linked Data
2. The intro to Linked data may not be enough for a new developer and basically is a wall

### 3. The intro is overtly simple

Having read the document, it occurred to me that my perspective is unique in many ways. The tutorial seems uniquely targeted towards someone with my qualifications. I have worked with Node.Js for nearly 3 years, I know TypeScript, I have used front-end frameworks such as AngularJs 1.5.x, Vue.Js etc that give me an advantage for learning Angular, and I just took Prof. Declan O’Sullivan’s Knowledge and Data Engineering class which taught be the basics of Linked Data and RDF. As a result, the way I look at Solid even being new to it is very different from how, say - a PHP developer or a Java developer would look at Solid.

So, I believe, the Solid community should answer the following questions:

1. Is Solid targeted towards developers who have a few years of experience with a variety of technologies?
2. If the answer to the first question is No, then to what extent can we abstract away the exposure to more complicated technology?

I think that more work needs to be done to design the tutorial in a way that it does not presume knowledge, and making Solid lucrative for developers will require simplifying the learning curve perhaps by abstracting some of the inner working.

## A.3 Initial hurdles

### A.3.1 23-12-2018

Ran `node-solid-server@23d5dad7042546231747dc8e6f23f9f6e9702ffb`. Running server on localhost with self-signed keys. After signing up with the local server and obtaining a local webId, I proceeded to click on Profile, Inbox etc. None of the pages showed any information. Instead it seemed like something was broken.

Inspecting network logs showed that I was getting 500 errors on most GET requests due to *self-signed certificate*. Some searching for the error threw up slightly unrelated solutions for npm.

Looking at the README, I found <https://github.com/solid/node-solid-server/tree/23d5dad7042546231747dc8e6f23f9f6e9702ffb#running-in-development-environments>.



### A.3. INITIAL HURDLES

---

I couldn't immediately find `solid-test` but <https://github.com/solid/node-solid-server/issues/816> showed me the way.

Thoughts:

`solid-test` binary seems unnecessary. This should be a flag or environment variable in the `solid` binary.

#### A.3.2 27-12-2018

Problem described by <https://github.com/solid/node-solid-server/issues/1029>

My `solid` server version: `v5.0.0-beta.4`

In my specific case, I had set-up my `solid-server` a few days ago and I was able to sign-up. I had set it for single user. Then today I tried logging in and kept getting the incorrect password error. (Since there's no email server set-up, I couldn't use forgot password)

So, I decided to re-register. Before that, I cleared the server config and data and re-initialised `solid-server`. This time, I explicitly set it up for single user. Register does not work for single-user set-up.

#### A.3.3 18-01-2018

Pulled `node-solid-server v5.0.0-beta.5` and was finally able to execute single-user registration. Following that, I tried to run the tutorial app (<https://solid.inrupt.com/docs/writing-solid-apps-with-angular>). While there were no errors, the app was not pulling any data to the form. Since I knew there was actually no data except name in my `solid` pod, I expected at least name to be fetched. After a little debugging, I found that the form expected data stored in the `VCARD` namespace, and there was no such data. This indicated that the data needed to be created, so I simply filled the form and submitted. I got 403 errors.

The error messages were rather crude and opaque. A lot of searching around the documentation finally led me to <https://github.com/solid/node-solid-server/issues/877>. After editing the `data\profile\.acl` file in the `solid` server directory, I was finally able to create and edit some data.

Thoughts:

1. Tutorial is seriously inadequate.

2. Several steps relating to permissions have been skipped in the tutorial and it simply assumes the reader is familiar with all configuration steps.
3. The tutorial doesn't even properly explain what it does and how to test it.

## A.4 Read data from and Write data to Pod

### A.4.1 2-02-2019

Since Inrupt's announcement that they'd be focusing on building an SDK based on React and Ldflex, I shifted to trying out the new experience. I ran the React generator and it worked but didn't do much useful more than the Angular one did.

(A long break followed, during which no significant progress was made on learning the SDK. The team behind Inrupt did some releases which added the support to write to the Pod, but not all features laid down in the planned timeline have been released as the developers took a detour to add support for Shapes.)

### A.4.2 2-05-2019

ShardsUI was used to build a UI template and I started retrofitting the components from generator to get data into the UI, and I was able to get the name which was the only thing in the Pod profile at that point.

### A.4.3 11-05-2019

In the two weeks since starting with the front-end, a conversation with James Martin revealed that some documentation of the SDK was out-of-date and that the SDK's version corresponded to the version of the generator and not the components, although the components package is a dependency of the generator. He also told me that some releases of the generator had in fact added new features although not all of the previously planned features.

Session on 11-05-2019:

1. New version of generator to test new features being used to generate new scaffolding.
2. Solid server started

### 3. Albus started parallely

Issues:

1. New generated app doesn't work due to a 401, which theoretically shouldn't happen because the .acl file gives permission for the card.
2. New generated app doesn't accept WebID for login when started on different port. Only existing cookie login is working. "WebID not valid" error message being displayed. This shouldn't happen either because WebID is perfectly valid!

Solution:

The first problem might have been resolved, because now that I was running the new generated app on a fresh port where the cookie did not exist, I went through login again and a fresh cookie was created. It seems like due to solid-auth-client version change, old cookie might not work in the new one. Still worth further investigation.

- Upon further investigation, the problem is still not completely clear but might have something to do with the fact the old and new generator use slightly different versions of solid-auth-client (2.2.14 and 2.3.0).
- There also seem to be major changes in the versions of react-components (<https://github.com/solid/react-components> 1.3.0 vs 1.6.0) which is a dependency of <https://github.com/inrupt/solid-react-components> (which in turn has been bumped from 0.1.0 to 0.3.2). Seems like I should just update all of my code in Albus.

The second problem is tracked here. A workaround allowed me to circumvent the issue for now, by using <https://akashdeepsingh.localhost.in:8443/profile/card#me> as the provider URL. <https://github.com/inrupt/solid-react-components/issues/56> Somehow, this also resolved the first problem, but I have no idea how.

Outcomes:

1. Albus is now using updated dependencies, but doesn't fetch the image.
2. It's clear that the solid ecosystem and developer tools are not stable enough for production consumption as there are major regressions in minor version bumps. A lack of tests in most libraries is also evident.

### A.4.4 12-05-2019

At this point in the process of learning solid app development, there is a need to find a reliable way of bootstrapping the data model.

The current line of thought is to just create a file in the private directory and determine the proper permissions for it. Manually creating a file is possible, but doing it in a production environment might mean using the upload function. There should probably be a cleaner way around it. Giving permissions to a specific site should also be reliable and safe. Need to evaluate how this works.

### A.4.5 12-05-2019

<https://github.com/inrupt/solid-react-components/tree/660a12adecc726b8b1b0f9b8daf08ce19a862ef5> The repo is seriously lacking in documentation and seems ill-designed for consumption by developers. It seems that the focus is on making the generator usable but not truly extensible because what the generator uses is not well-documented. At this point, even though I have the generator, I can't really do much with it when it comes to changing it to build an actual app. While this is obviously not the intention given that the whole SDK is a work in progress, the lack of attention to components does harm the ability of a new developer to pick up easily. Additionally, a concept like ShEx has been introduced in the README with no preamble. On the whole, the developer experience is broken, and documentation (if any) is so fragmented that a developer might give up.

### A.4.6 13-05-2019

I have deviated from the goal of finding a way for application bootstrapping to bring Albus up to speed with the capabilities of the generator. I managed to fit the profile component into the UserAccountDetails component. Profile updates are now possible. Next step is to plug my ontology and actually create my own data. And navbar name, img display are also done.

### A.4.7 16-05-2019

3-4 hours have been spent in trying to understand how to model, store and access application data in solid. The best resource thus found has been <https://noeldemartin.github.io/solid-focus/>, which is an application written in Vue. Having looked through code, and having made it somewhat functional, I have found that it's not yet intuitive and simple.

The two issues I faced were:

1. 403 when I tried to create a workspace. Turns out the app was trying to create a basic container, and didn't have the permissions to do so. I just edited the root container's acl and gave everyone write permissions. This is not ideal and I need to figure out how to give an app permissions for creating a container for itself but and restricting it to that.
  - The important thing is that once an app is authenticated using my webid, shouldn't it be able to use the cookie to get the permission to do the creates and other edits? This needs to be investigated.
  - So I found this: <https://github.com/solid/node-solid-server/issues/1171#issuecomment-481984361> and it makes sense. This also means I'll have to build a new local solid-server to test out the fix.
2. When I reloaded the app, the workspace was lost, i.e., the app was no longer able to discover the workspace. A workspace essentially created a new directory (described by an RDF basic container).

The Solid spec (<https://github.com/solid/solid-spec>) is probably the best and most authoritative resource yet, but it's very dense and requires a lot of context about the Linked Data platform. A tutorial for creating a simple app with the second important step: "how to manage app data" doesn't seem to exist.

<https://noeldemartin.com/tasks/implementing-a-task-manager-using-solid> I've been reading this to understand the app development process. It's similar to my blog, only more focused on building a much simpler application.

### A.4.8 17-05-2019

After much rummaging, I found how to make applications /emphtrusted, by following instructions mentioned in the README of <https://otto-aa.github.io/solid-fi>

lemanager/. Apparently this is a recent change and earlier all apps weren't untrusted by default.

This clears one of my earlier doubts as to how do people give apps access to pods. Earlier, I was only able to give access by editing the acl of resources manually. The new way I have discovered is better, but hidden in obscurity and seems inadequate for two reasons:

1. Apps still don't seem to have the RW permissions I have, so I still can't create TODOs in the TODO (solid-focus) app.
2. Apps are given access by url, or rather domain name. So, multiple apps deployed at different routes on the same domain all get bundled access. No way yet to specify it more finely.
3. The wording on the trusted apps page seems to suggest that apps get the same access as the user, which may not be desirable. A broad RW access on the entire pod is not what I was looking for, but may be required.
4. The big problem is, apps should get their own sandboxes and should have free run in those sandboxes rather than having full admin type access on the whole pod.
5. I'm not sure what trusted app does. The way I found to actually give permissions to an app was to edit the acl file, this time with the app's specific name(domain name). Turns out this has recently been documented here.

An issue on improving permissions UX exists: <https://github.com/solid/node-solid-server/issues/1142>

So, the question is: What does trusting an app do in effect? From what I gather, I still have to give permissions to apps through acl for the folders/files that need to be edited. Is trusting a firewall?

Answer: If done properly (no trailing /), it gives blanket permissions!!! This seems dangerous. Also, if a container exists and there is no acl inside it, I need to create an acl with the owner having all permissions.

Big problems: documentation is highly fragmented and disjointed. Connecting the dots and reverse engineering seem to be the only way.

Current sources of documentation:

1. Github READMEs, Issues, PRs, Commit messages of many repos of the organizations <https://github.com/solid> and <https://github.com/inrupt>

2. <https://forum.solidproject.org>
3. Gitter channel of solid-spec
4. A handful of blogs

### A.4.9 17-05-2019

It seems at this point that the workflow for an app for first login (bootstrapping) would be:

1. Login
2. Ask for permissions (this could be either becoming a trusted app, or getting explicit granular permissions to read append to a particular container to be able to create a subcontainer sandbox)
3. Create a subcontainer within in public or private container that acts as the apps data directory
4. Create an acl in the directory giving explicit full permissions to the owner and read, write permissions to the app
5. Maybe a disconnect button that would edit the file and remove the app's permissions and then direct the user to remove the app from the trusted apps

Actions:

1. Need to understand permissions and how they are resolved for containers and subcontainers. Can a subcontainer have more permissive rules than the parent container for an app?
2. Need to understand what role an ontology would play. How to check constraints, if at all? (where do shapes come in?)

Big UX issue: permissions, both giving (especially granular) and revoking

## A.5 Migration of Pods

### A.5.1 21-05-2019

I wanted to test Solid's promise of portability of data by creating a mechanism for automating Pod migration (migrating a Pod from one provider/server to another). I had an idea that webIds would be a difficulty and that's what I found.

Steps:

1. VM with nodejs and docker
2. Clone server on VM and run with docker
3. Replace default config with custom config
4. Plug imported Pod data via volume and config refers to .db and data directories from this

Issues:

1. .db won't work unless the URI of the new server is same as that of imported Pod data because OIDC is configured using that.
2. User account cannot be imported directly as the webId is tied to the old Pod
3. User data also has the same problem as above.
4. If external webId is used, both the above issues can be resolved to an extent if the external webId is static
  - An external WebId is additional overhead and introduces complexities better avoided Refer: <https://forum.solidproject.org/t/move-it-at-any-time-without-interruption-of-service/565/6> and <https://forum.solidproject.org/t/will-tying-web-ids-to-hosters-create-lock-in/756/8>

Update: <https://forum.solidproject.org/t/transferring-a-solid-pod/1902/>  
2 Someone is doing something at <https://mypod.id/> and it seems to utilize <https://holo.host/> under the hood (blockchain!). Let's see where this goes.

On a related note, some thoughts on automated account creation: <https://github.com/solid/solid-spec/blob/master/recommendations-client.md#creating-new-accounts>. This could in theory be used for migration. Essentially, you don't move a Pod physically, but create a new one from scratch as a replica of the old one with the new webId. I didn't test this out as I've given up on this for my project.

## A.6 Implementation: A very long post

### A.6.1 25-05-2019

Login and logout functionality is done. This involved taking code from the generated app and fitting it into Albus.



### A.6.2 30-05-2019

Generator version 0.5.0 came out a week ago. Profile form is now generated using ShEx. They have chosen to completely change the profile view. The problem I see here is that there doesn't exist the simple way of doing this in the generator now. ShEx has been imposed as the canonical way while the components repo says that it is still an experimental preview.

Other news: Solid-cli exists for non-web interaction with the solid server, that could be useful in the migration part. <https://github.com/solid/solid-cli/>

I chanced upon a thread that revealed that NSS is not strictly compliant with the spec and the spec isn't a spec yet. <https://lists.w3.org/Archives/Public/public-solid/2019May/0015.html>

LDFlex It works quite well. Apart from the fact that there is no documentation, it is a good API. add, delete, set and get have been tested. set is an upsert.

### A.6.3 1-06-2019

I used rdflib.js to list and parse data from my pod. I created two post documents manually. LDFlex doesn't seem to have the option of working on containers, so needed rdflib for that.

Solid-focus' code was useful in this regard. It was also useful in understanding how it creates app data. Now, using <https://solid.github.io/ldflex-playground>, I was able to find out that it is possible to list using LDFlex. The key is using the ldp:contains property on the container. Pretty neat.

### A.6.4 3-06-2019

Getting Posts from Pod

1. Get user
2. Get storage URI
3. Construct app data directory URI
4. List Posts by using ldp:contains

Need to figure out how to check for conditions. Seems like we can't. Should separate public and private stuff by containers

Result: Able to read posts from Pod and display them.

### A.6.5 5-06-2019

Likes, unlikes, comments, crossposts are done using notifications. Maybe feed can also be discovered using notifications? This needs to be figured out.

### A.6.6 7-06-2019

<https://forum.solidproject.org/t/newbie-perceptions-on-solid-app-framework-vs-web-standards-effort/844/4>

This discussion and this post are the best representation of the confusion surrounding Solid. The question is, what is Solid? Or rather, where does it fit in our current - perhaps limited - vocabulary of classifying concepts in the software development and engineering field. First let's just get the obvious out of the way: Solid is not a programming language. The other classifications that we consider are:

- Framework
- Paradigm
- Pattern
- Standard (or set of standards)

So, Solid, is definitely a set of standards. That's clear and well-documented. As for the others, it needs some dissection. We might actually find that Solid doesn't fit any of these moulds perfectly, which isn't necessarily a bad thing, but it leaves the question of 'what is Solid?' largely unanswered especially from the perspective of someone not familiar with Linked Data.

### A.6.7 7-06-2019

We're finally getting social. So, I decided to create another user. At first, I thought I'll need to go to a different server for it, but it turned out that I could just create a new directory with its own config and start a new instance of solid with a new port number. One hiccup: Probably at the /authorize call, solid updates the card, and it seems to be writing hardcoded 8443 into that. That's wrong. I had to manually edit the whole thing. I need to test this with a different app (solid-focus) to isolate the issue.

Good thing: Somehow my solid server's consent page has been updated and is asking for adding to the trusted apps list. This flow is much better, and resolved one of my earlier concerns.

Created User individual for both users using protege. I'm not sure this is correct but let's see.

I need to find a way to store or calculate followers of a user for displaying that information. So far, I store followees for a user. Followers should also be displayed. This will work using inbox.

### **Long session**

Achieved:

1. Followee list
2. Friend profile view readonly
3. Feed with followee's posts

### **A.6.8 8-06-2019**

No coding, just thoughts here.

Feed aggregation is a necessarily centralised task. In a decentralised set-up, there are essentially two ways of building a feed of followees:

1. Pull
2. Push

In 1, we have to build the feed - on the fly - when a user logs in. This entails crawling through all of the followees, and their posts, and aggregating them. This is a time-consuming task. Moreover, sorting becomes harder. Marking posts as seen so as to not repeat them across sessions is also harder as you'd have to incur storage cost for storing such data. Discovering all your followees' posts is time consuming enough, and done at the client, it makes the presentation harder and brittle. If you want to sort before presenting, as you should, that's increased latency.

Over time, as the posts increase, you'd also want to limit the posts you get based on recency, and whether the post has been seen. Since we lack global knowledge beforehand, this is exceptionally hard.

In 2, we expect that every time a followee publishes a new post, it gets sent to the user's inbox as a notification. The rest is pretty much the same. The feed is still only created on the fly, but the network cost of getting posts from followees' Pods can be avoided. The problem is that it requires one's Pod and inbox to be perpetually available. This expectation has to be clear for Solid users.

### A.6.9 16-06-2019

Tried various ways to create a container based on whether it already existed.

LDFlex does support get, but if the container doesn't exist, it returns 404 which isn't being caught (promise not being rejected, <https://github.com/solid/query-ldflex/issues/23>). This means checking for existence may not be possible without rdflib.js either is possible using a ldp:contains listing on the top level container of the Pod and iterating through all contents. Seems robust/reliable enough.

### A.6.10 17-06-2019

Creating a resource entails similar operations to creating containers. LDFlex doesn't support this either yet, but rdflib does. It was a little tricky to figure out how to do it from docs because they are too fragmented, incomplete, and inconsistent. I first tried to use the fetcher's createIfNotExists function, and it threw a weird error which I couldn't work around. Didn't find any solution to it online. It's not clear what it does. I searched on the forum and it led me to <https://forum.solidproject.org/t/my-first-app-adding-resources/275/9> and I found putBack does what I need except it just replaces in case the resource already exists. So the problem of checking before creating comes up. The solutions:

1. Do a get or a listing of the container before creating. If get fails or if listing doesn't contain the resource, create, otherwise error.
2. Or just create. This could work because I will be creating using a unique filename every time. A combination of a hash of webId and timestamp. This decision is justified as an optimization for the common case of no conflicts. We can be reasonably confident that there won't be a conflict, and the check adds overhead that will make experience bad. Since the project is more of a POC/MVP, this

is fine. The safety of the create operation can be enhanced at a later time using better tools.

### A.6.11 17-06-2019

Solid has 3 ways of interacting with data programmatically:

1. **REST**: It would seem that this is the basic and the canonical approach also laid out in the spec. All other interfaces seem to be built on top of the REST interface, which is documented well enough and is quite comprehensive. It requires knowledge of HTTP headers, LDP, and RDF.
2. **RDFLIB.js**: RDFLIB.js is perhaps the second most mature interface (apart from the Data browser). The problem is it assumes a knowledge of RDF and LDP. That would be fine if the documentation was better, but it's fragmented, incomplete, and inconsistent.
3. **LDFlex**: LDFlex is the newest, and by far the easiest to get started with. Unfortunately, this too suffers from the incompleteness of documentation apart from the fact that it doesn't have all the features to be able to provide all of Solid's capabilities. While it's been actively worked on, it will take some time for it to be mature.

The bottom line: no one solution fits all of the needs for a practical application like an OSN. At the very least, a document covering all the common CRUD scenarios for containers, resources, and triples should exist for the 3 interfaces from a view of helping new developers get started. This is not the case yet. None of the documents that exist contain this information.

Some use cases such as filtered listing don't even seem to be possible yet, due to the way data is stored and serialized on Pods.

The bigger challenge here is to reorient developers of traditional db-backed systems to this way of working with data.

### A.6.12 22-06-2019

Given that the primary data access interface for Solid apps is RESTful, this creates a gap for meaningful querying. Most applications are backed by a database, RDBMS or NoSQL, which provides a sophisticated query interface that lets developers select

and project at the very least, apart from more complex aggregation. Solid stores data mainly as Turtle representation of RDF. While this serialization may not be the most efficient, it is expressive enough for designing versatile data models. However, querying on Turtle has traditionally been done using SPARQL, which is highly expressive, albeit complex. While the Solid specification prescribes it (<https://github.com/solid/solid-spec/blob/b941ff795acdedb7d7a24d40dabdfcce7efa9283/api-rest.md#alternative-using-sparql>), NSS lacks a SPARQL endpoint (<https://github.com/solid/node-solid-server/issues/962#issue-383649959>) for querying over the data in the Pod, making it necessary to perform operations such as selection, projection, and aggregation over Pod data inside the client application. This makes modelling data difficult especially when designing with privacy in mind. Since permissions using ACLs within the Pod are assigned at the level of resources (files on disk in case of NSS), different access levels need to be isolated at the resource/container level rather than predicate/property level. This means that if a user Alice wants to create a post visible only to a specific follower Bob, then a separate resource needs to be created specifically for Bob with the ACL file associated with that resource having the webId of Bob. Since the listing of resources in a container is done at the container level without any filters, a listing by every follower of Alice will show that resource but a request to get the content of that resource will fail with a 403.

To fill this gap, libraries such as RDFLib and sparql-fiddle have implemented workarounds that allow executing SPARQL over loaded RDF documents but the experience is not smooth.

Forum topics dealing with this: <https://forum.solidproject.org/t/using-sparql-on-solid-data/1335> <https://forum.solidproject.org/t/fun-fact-using-sparql-to-query-the-type-registry/776> <https://forum.solidproject.org/t/executing-sparql-query-over-json-ld-files-in-solid/1885>

### A.6.13 25-06-2019

There are two solutions to the discovery problem:

1. **Centralized:** Using a central registry that all users are registered to (maybe opt-in; not hard to make it opt-in)
2. **Decentralized:** Crawling friends, friends of friends. Not feasible. Cold start

problems:

- A user having no friends will not be able to find anyone
  - No one will be able to find a user with no friends
3. A 3rd way is a mixture. Instead of having a central registry. A user/bot Albus can be made followee on all accounts, and essentially discovery becomes a matter of crawling friends of Albus.

Potential issues:

1. quasi invasive
  - Solution: opt-in
2. users can unfollow
  - Could be a good thing as an opt-out
  - No real benefit except entire app can be made within the scope of Solid, but still a server will need to be maintained (Pod of Albus).
  - Someone who unfollows is essentially explicitly disconnecting themselves, which may be a good thing.

### A.6.14 8-07-2019

I wired create post and it's working. I figured I don't actually need to use my ontology, really. Not for the posts yet at least.

Group authorization is possible: <https://github.com/solid/web-access-control-spec#groups-of-agents> How I plan to implement this for now is just a single group called ApprovedFollowers.

There are currently 4 types of Visibility I plan to support (not yet editable):

- Private: Only the user sees their posts
- Public: Everyone can see the posts theoretically, but they appear in the feed of UnapprovedFollowers (does not exist in followee's pod)
- Followers: Only ApprovedFollowers see these (exists by default in followee's pod)
- Specific: user(s) by webId

- Custom groups: Advanced

Editability might be good to PoC, at least the visibility if not the rest of the post.

This UserStory has some ideas but doesn't seem to have been implemented yet: <https://github.com/solid/solid-spec/blob/master/UserStories/PrivateSharing.md>

### A.6.15 10-07-2019

A blocking error in query-ldflex (<https://github.com/solid/query-ldflex/issues/23>) is causing frustration. I can list authorised and unauthorised items, but not knowing which is which. Then, when I get each of them, the ones that do not allow access via acl throw 403 and EXIT. I can't catch and handle those errors, and so the entire thing falls apart just because I don't have access to some items in the list.

Also found a problem with the authenticated users group access policy: for single tenant solid servers, it won't work for users other than the one user that has a pod on it. So, essentially I can't have an access level that allows solid logged in users instead of the whole public access.

Working Solution:

1. Clone and compile this <https://github.com/RubenVerborgh/LDflex-Comunica/commit/73e7ee939ba37edee6ed6b31434aefb1e3c20acf>
2. Replace `node_modules\ldflex-comunica\lib` with the lib compiled in Step 1
3. Remove blocking calls (use `then` instead of `await`)

Unresolved Issues:

1. Still can't catch errors. Detecting using non-empty title only
2. Have to replace `ldflex-comunica` manually outside of the automated build

### A.6.16 10-07-2019

Post creation with basic visibility levels works. Creating acl took some time. The doc (<https://github.com/solid/web-access-control-spec>) is pretty good for forming the acl file itself, but storing it to pod was a headache. I found (<https://otto-aa.github.io/solid-acl-parser/#/>) and ([https://github.com/nmalcev/pod-explorer/blob/master/static/scripts/models/acl\\_manager.js](https://github.com/nmalcev/pod-explorer/blob/master/static/scripts/models/acl_manager.js)) useful but the first



one didn't seem to have documented support for storing and it was using N3 anyway. The second showed me how to parse the acl file blob into rdflib and then it just took some playing with rdflib to get it to write.

Finally, I scratched my head on a stupid mistake. Singular vs Group agents. I was using `agentClass` for all agents, even for self. That was wrong. Singular agents use `agent`, and group agents use `agentClass`.

### A.6.17 12-07-2019

Tldr: it worked, eventually. But it wasn't easy.

LDN: is a protocol at the most basic level. It is simply a way for two parties to communicate with push. The sender and receiver agree on a shared space on the receiver's Pod where the sender can create resources (messages). And the receiver can act on those messages at a later time. It's like a simple mailbox. So, there's a global inbox in the Pod. I didn't want to use that. So, I made an inbox for Albus. For some reason, the LDN spec mentions JSON-LD (<https://www.w3.org/TR/json-ld/>) as the way to do it. There are some useful payload examples (<https://www.w3.org/TR/ldn/#payload-examples>). So, I had to create my messages in JSON-LD using the activitystreams vocabulary (<https://www.w3.org/TR/activitystreams-vocabulary>). JSON-LD isn't too hard (<http://fsteeeg.com/notes/from-rdf-to-json-with-json-ld>). Saving JSON-LD using `rdflib.js` is possible as described here (<https://forum.solidproject.org/t/is-there-a-converter-between-json-ld-and-turtle-n3/1817/3>). The `serialize` method was fine but to save it to the Pod, I was using `Fetcher.putback` and that didn't work due to (<https://github.com/linkedata/rdflib.js/issues/324>), so I had to fix it (<https://github.com/linkedata/rdflib.js/pull/325>) and host a fixed version myself (<https://akashdeep-singh.github.io/thefreeelf/ss/rdflib.min.js>). After that, I was able to get the listing of notifications. Now I need to add Approve/Reject buttons and add the approved followers to a group.

The other part is to create notifications. I created the `notif` file manually. I have to automate it which will require replacing `rdflib.min.js` with my version.

UPDATE 26-07-2019: The Pull request (<https://github.com/linkedata/rdflib.js/pull/325>) has been approved, but is pending merge.

## A.7 Final session: almost finishing touches

### A.7.1 14-07-2019

Wired discover page with listing of public profiles. Next steps: follow button wiring, add to directory while signing up.

I need an onboarding/register page before Feed after first login. The main check can be the existence of the albus container. The check will have to happen every time. Essentially, if the directory exists, go ahead, if it doesn't then go through the entire process of creating it, and the basic containers inside it, and show the option for opting in to the discovery layer. Then, move to Feed.

### A.7.2 17-07-2019

Completed the following functions: \* Sending notification

Using mostly rdflib

One point that stuck out briefly was creating notification with `acl:Append` and `fetcher.putBack`. Turned out PUT doesn't work with Append, so I figured out sending POST request using `webOperation`. Asked a question on gitter and got an answer within minutes.

- add followee after sending notification

Used LDFlex to add the `sioc:follows` predicate.

- approve/ignore follower and delete notification

Approving by adding a member to `ApprovedFollowers` or ignoring by `UnapprovedFollowers` using LDFlex, and then clear that notification from inbox and UI.

- list `ApprovedFollowers` on Friends page

Listing `ApprovedFollowers` using LDFlex

### A.7.3 18-07-2019

Several hours poured into setting up boilerplate programmatically. Some major issues were faced relating to creation of acl files that snowballed into a chain of problems.

This led to me finding out that I had added additional permissions in the root acl file. I removed those, and then wasn't able to create the albus folder either. This made

no sense as I clearly had permissions. So, I decided to update my NSS to the latest(<https://github.com/solid/node-solid-server/commit/5729fe534665a022f4cc41384af2c47a4f1eaf57>).

That broke as well, at a step before even creation of albus folder was attempted due to an open bug (<https://github.com/solid/node-solid-server/issues/1120>). So, I reverted to my old NSS version (not considerably older, <https://github.com/solid/node-solid-server/commit/bdc5acba326f215c3d32eb2b234ec73d0f5cc9ce>), which brought me back to not being able to create the albus folder itself. Some reading around got me to an old issue (<https://github.com/solid/acl-check/issues/24>) which pointed to the fix (<https://github.com/solid/acl-check/issues/24#issuecomment-490144797>). It's a workaround that requires commenting a line in the root acl file. While this is workable, it precludes production launch of the app as the root acl file needs manual editing. As long as these issues are not fixed, Solid is not ready.

Now, bootstrapping also works, but not without manually editing the root .acl file.

Consent for registering to the central directory also works now.

For now, the project is complete, but the application isn't. I've learnt what I needed to learn. I will come back to the application later and finish it.

# Appendix B

## Application Screenshots

Figure B.1: Homepage

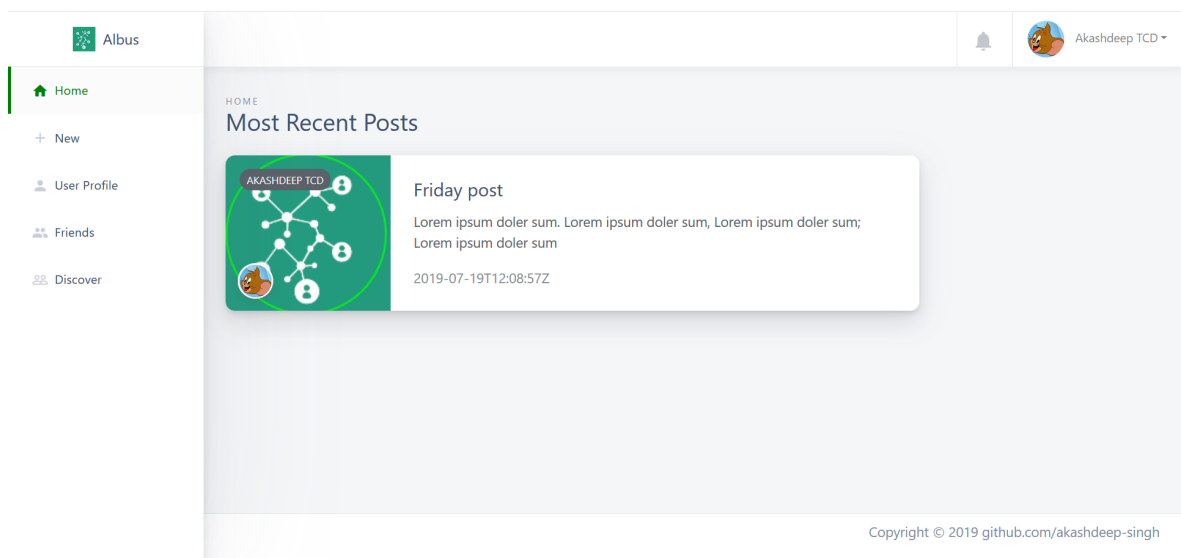


Figure B.2: Homepage with Logout button revealed

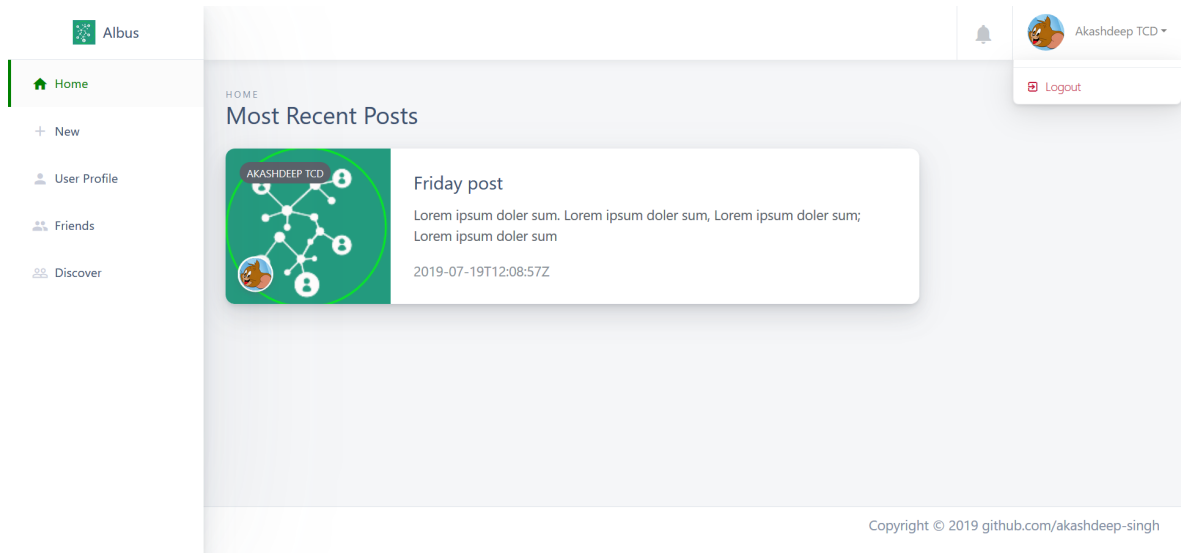
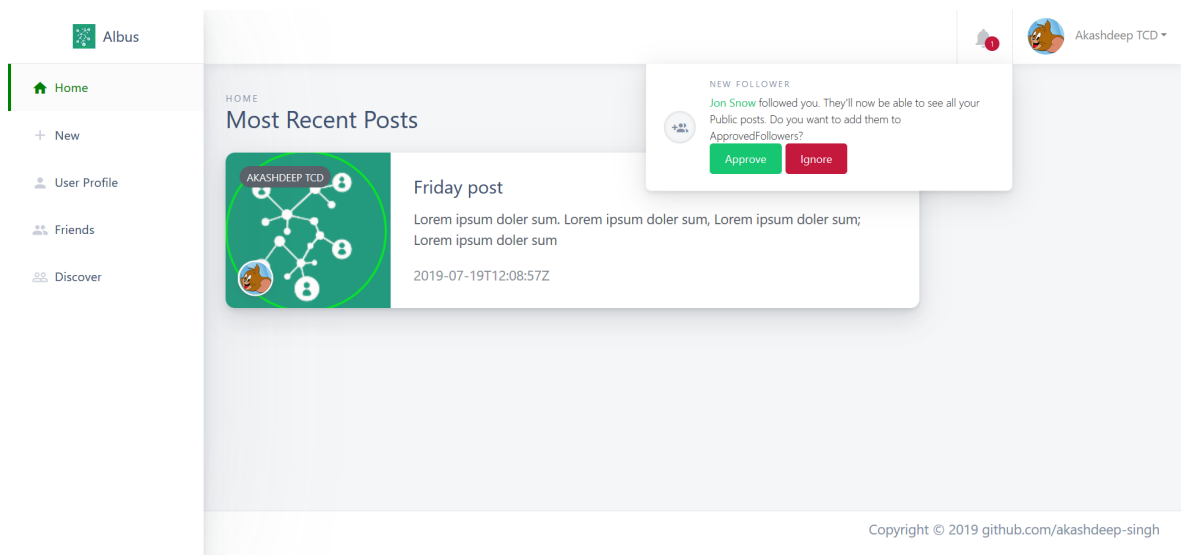


Figure B.3: Homepage with Notifications expanded



---

Figure B.4: Login with WebID

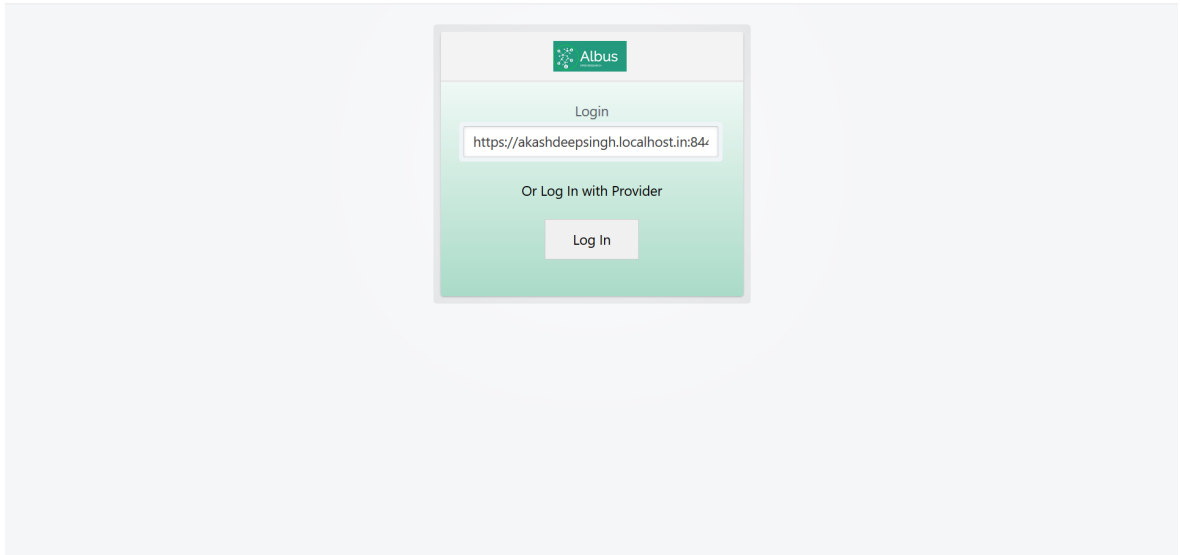


Figure B.5: Pod Provider selection

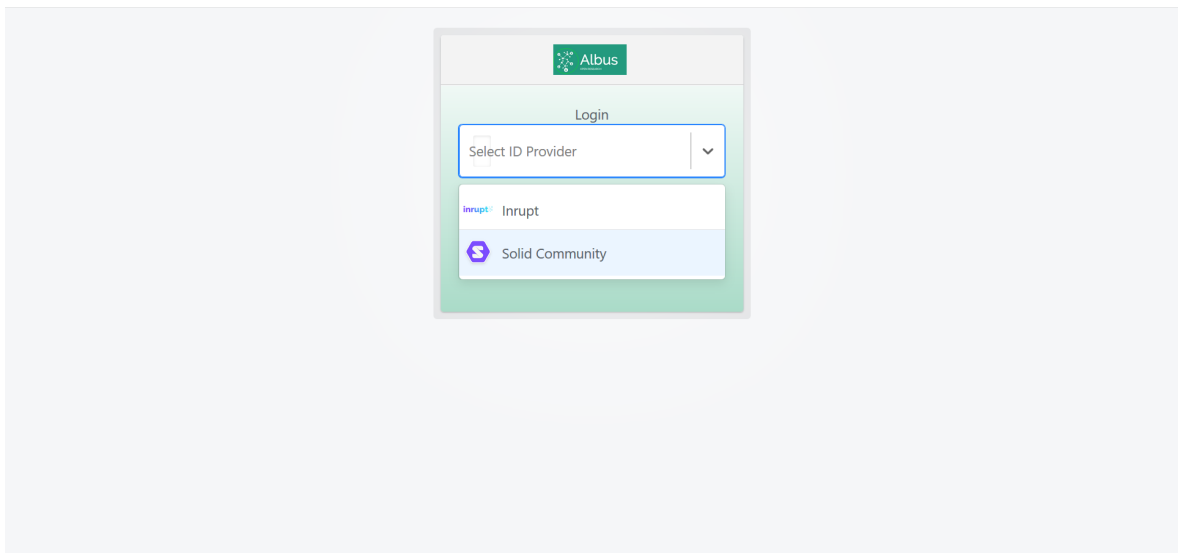
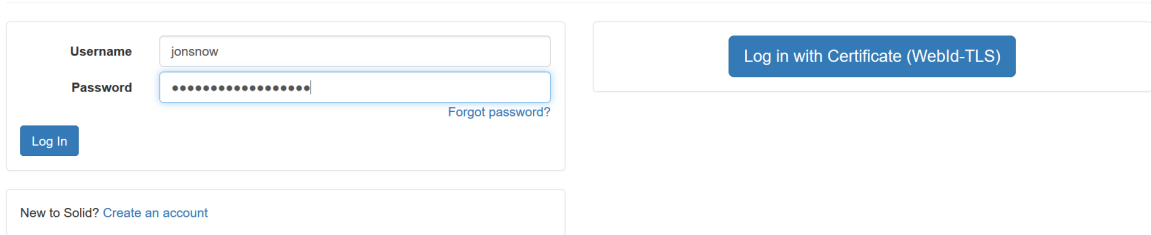


Figure B.6: WebID-OIDC authentication: enter credentials

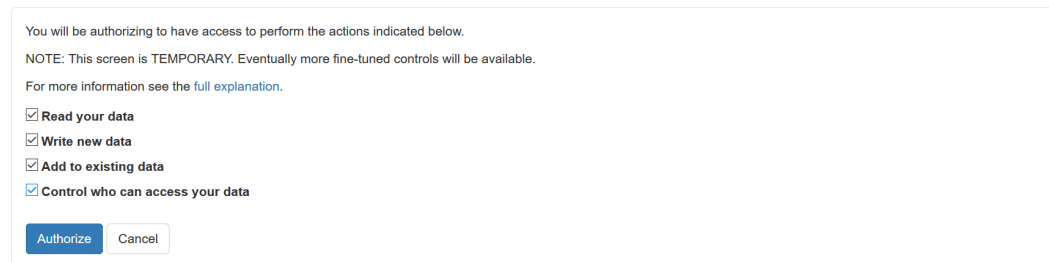
## Login



The login form consists of two main sections. On the left, there is a form with two input fields: 'Username' containing 'jonsnow' and 'Password' containing a masked password. Below the password field is a 'Forgot password?' link. A 'Log In' button is positioned at the bottom left of this form. To the right of the main form is a separate box containing a 'Log in with Certificate (WebId-TLS)' button. Below the main form is a link for 'New to Solid? Create an account'.

Figure B.7: Add application to trusted applications

## Authorize this app to use your data?



The authorization dialog contains the following text and controls:

- You will be authorizing to have access to perform the actions indicated below.
- NOTE: This screen is TEMPORARY. Eventually more fine-tuned controls will be available.
- For more information see the [full explanation](#).
- Read your data
- Write new data
- Add to existing data
- Control who can access your data
- Buttons: Authorize, Cancel

Figure B.8: Consent to store and publish WebID

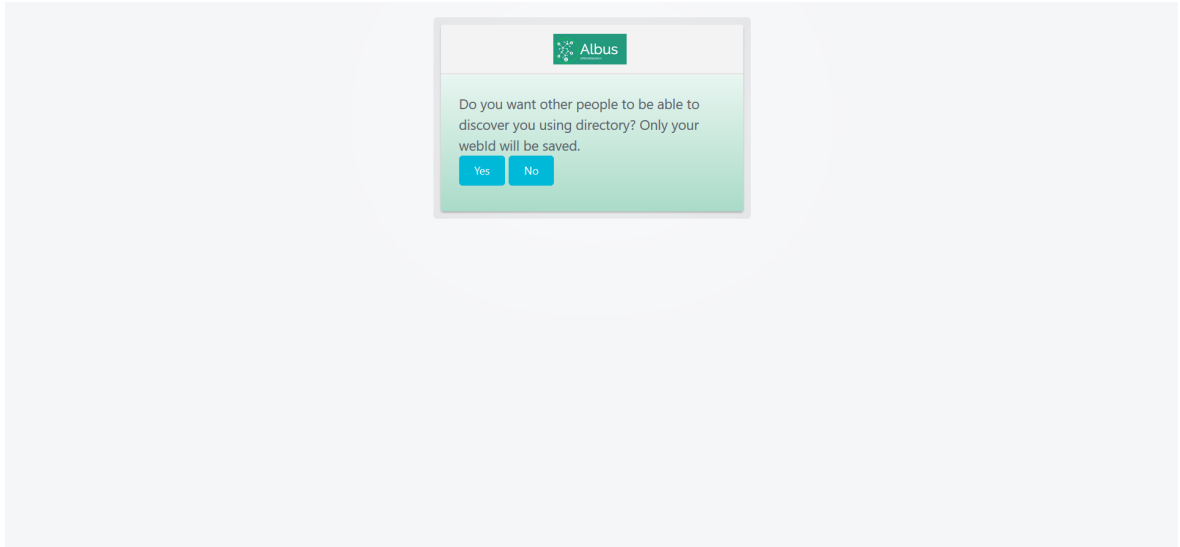


Figure B.9: Write post

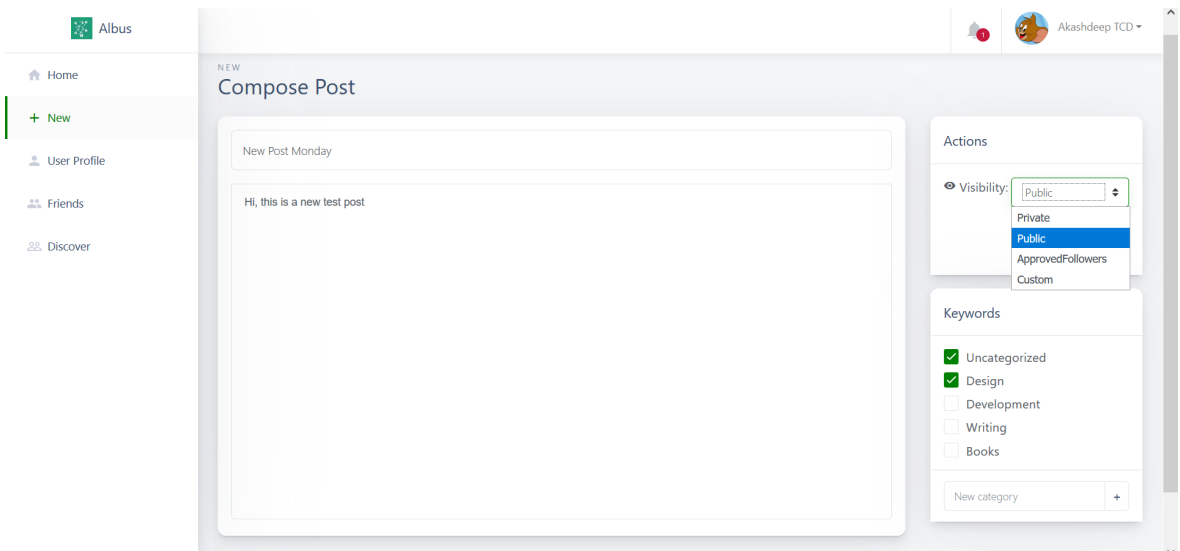




Figure B.10: View and edit your own profile

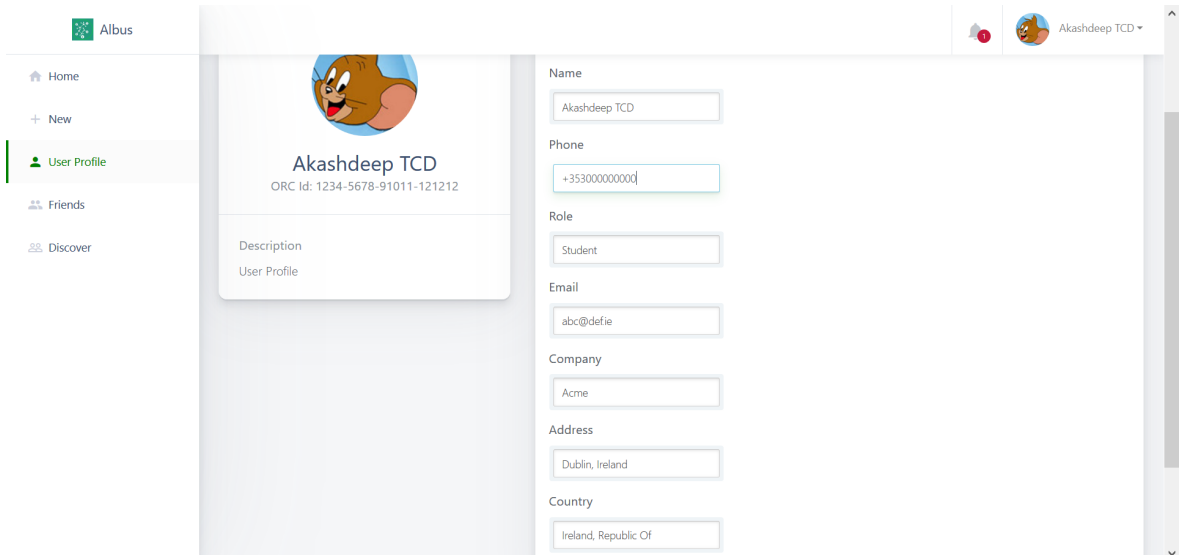


Figure B.11: View another user's profile (Read-only)

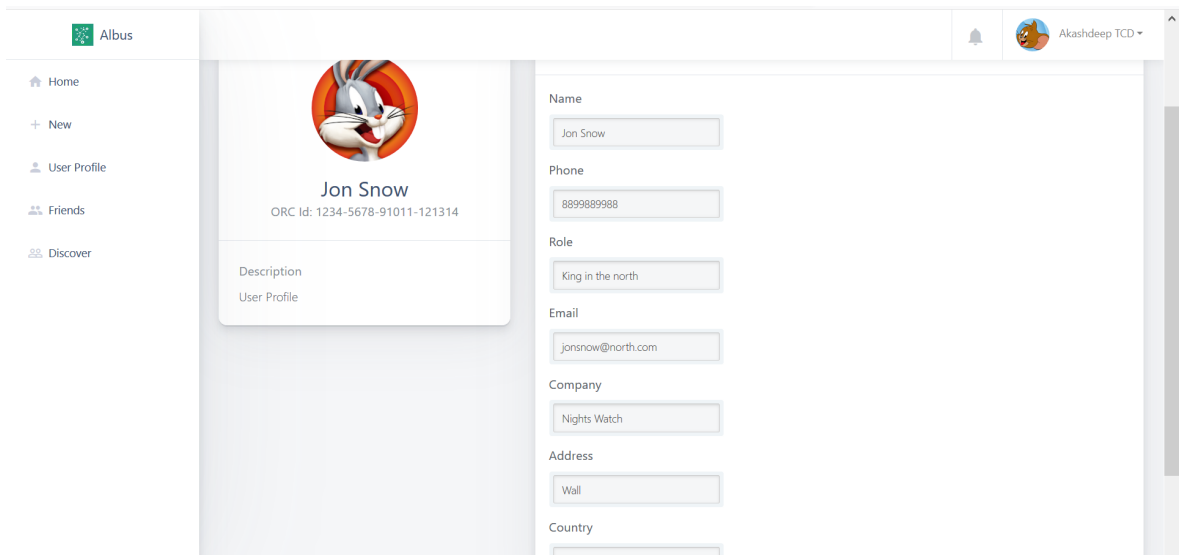


Figure B.12: View your followers and followees

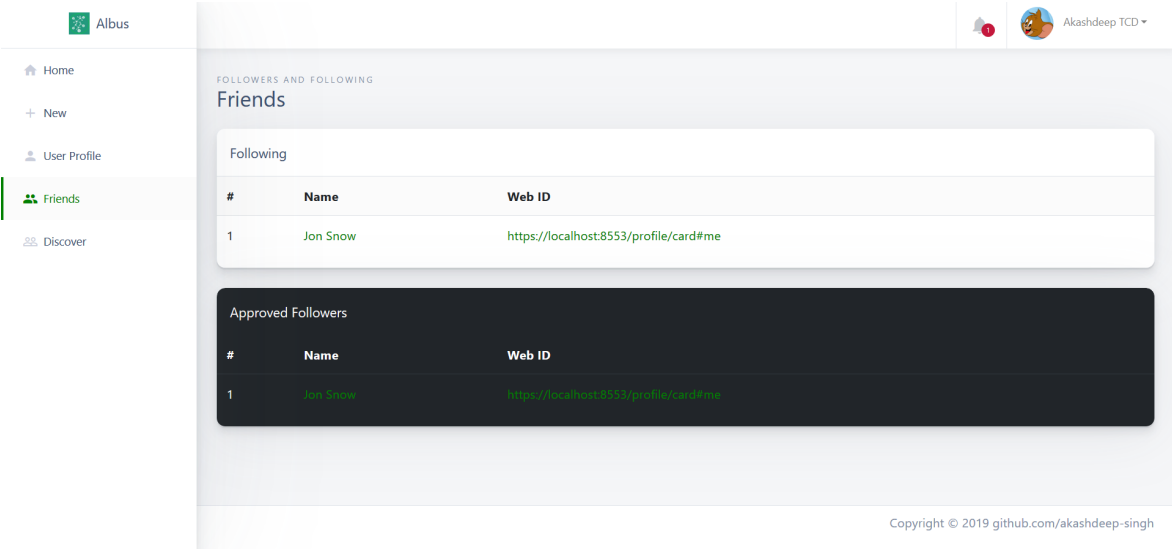


Figure B.13: List all users on the network who have opted-in to be discovered

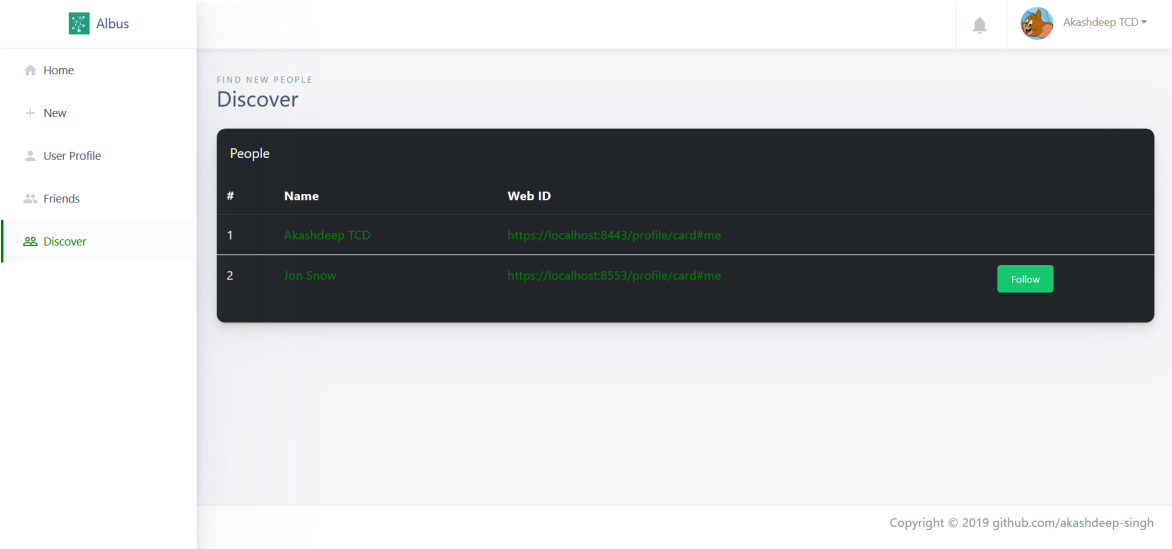


Figure B.14: Alert message upon sending a follow request to another user

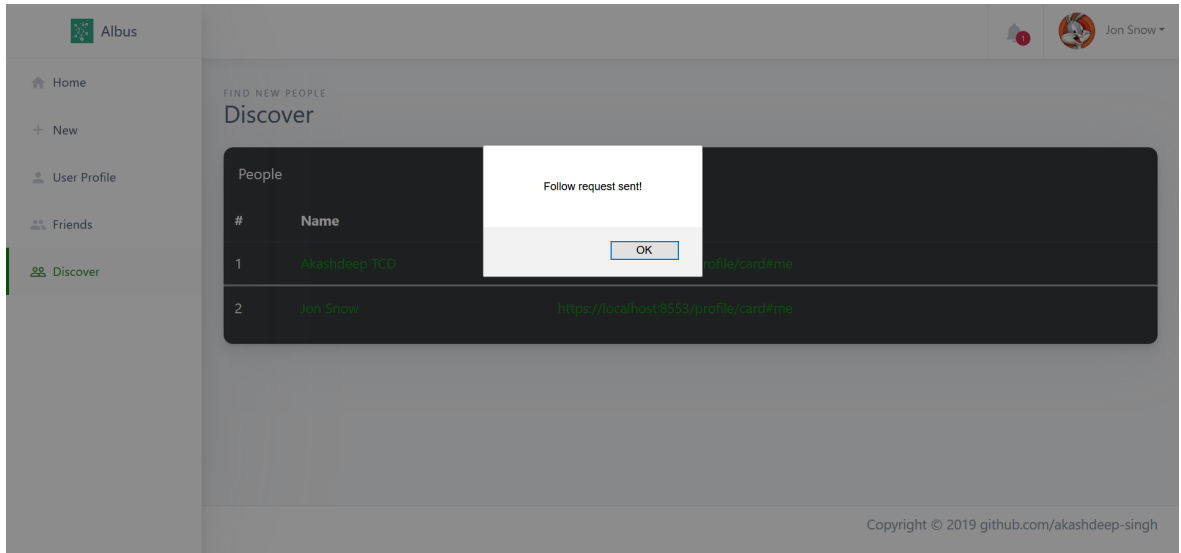
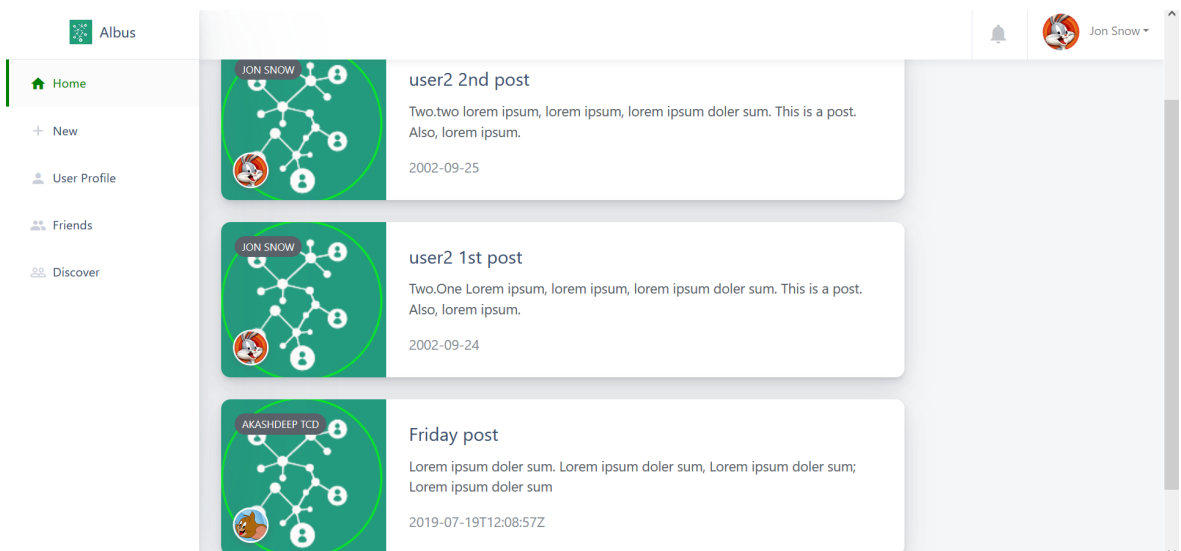


Figure B.15: Homepage from the point of view of Jon Snow



## Part II

# Getting Started with Solid

# A Beginner's Guide to Solid

## What is Solid?

Solid (SOcially LInked Data) is an evolution of the Web. It is a specification of standards, conventions and tools to enable privacy-preserving web applications.

## How is it different from Javascript?

Javascript/ECMAScript is a programming language. Solid is more of a platform or a paradigm. It let's you use Javascript/ECMAScript to build your application.

## How is it different from React/Vue/Angular?

React/Vue/Angular are frameworks/libraries that help you write scalable front-end Web applications with features that facilitate state management, modular code, data binding, and templates. These frameworks are associated with the popular MVVM pattern<sup>1</sup>. Solid applications can use any of these frameworks. Solid doesn't replace these frameworks.

Figure i: The traditional three layer application architecture

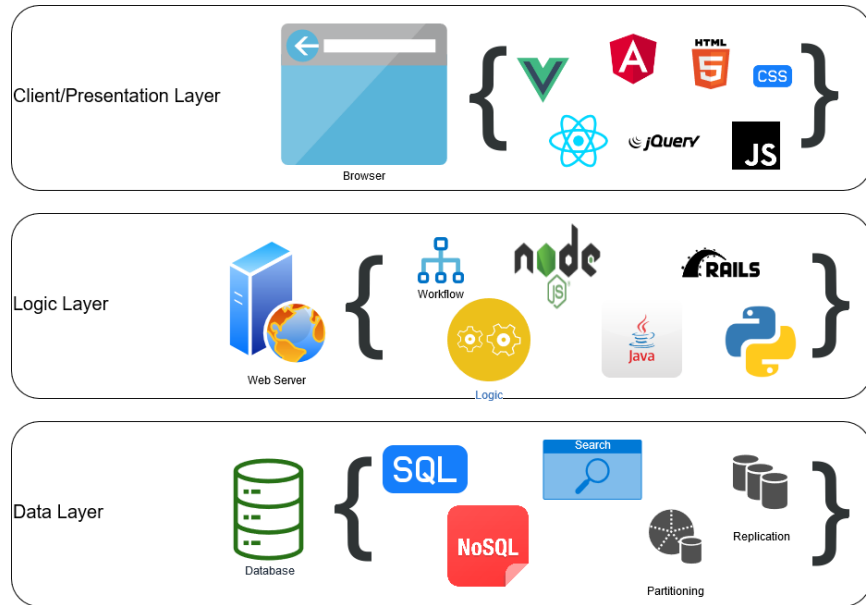
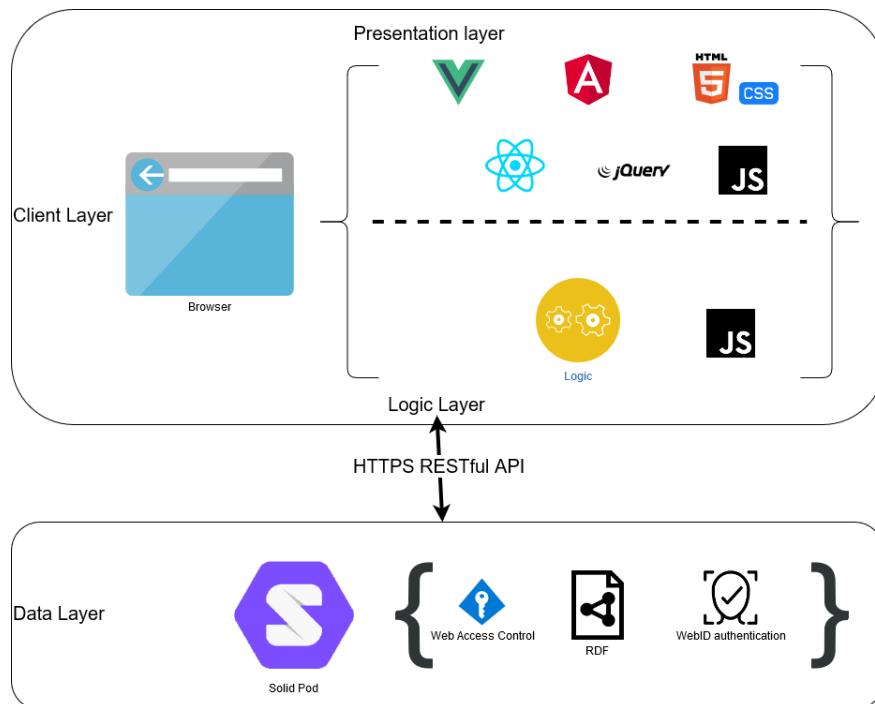


Figure ii: Solid application architecture



---

## How is it different from Django/Spring Boot/Express.js/Rails?

These technologies are Web frameworks<sup>2</sup> used for writing HTTP-based servers for the backend of the applications. With Solid, you don't need these technologies, because you won't be writing backend servers. Solid applications are front-end applications, and your Solid Pod acts as the backend for your application.

Consider the traditional three-layer architecture<sup>3</sup> (Fig. i) and compare with the architecture of a Solid application (Fig. ii). The Solid application's logic layer is written in the front-end with the Pod acting as the data layer.

## Basics of Solid

Solid has 5 main components:

1. Pod
  - A Pod is a *personal storage space* owned and managed by each user
  - Users can give access to parts of the Pod to apps
  - Users can host their own Pods or create a Pod with a Pod provider of their choice
  - Read more on the Solid community site<sup>4</sup>
2. WebID-OIDC, WebID-TLS Authentication
  - You don't log-in to individual Solid applications
  - You log-in to your Pod

---

<sup>1</sup><https://en.wikipedia.org/w/index.php?title=Model%E2%80%93view%E2%80%93viewmodel&oldid=909212899> [Accessed: 07-08-2019]

<sup>2</sup>[https://en.wikipedia.org/w/index.php?title=Web\\_framework&oldid=907817661](https://en.wikipedia.org/w/index.php?title=Web_framework&oldid=907817661) [Accessed: 08-08-2019]

<sup>3</sup>[https://en.wikipedia.org/w/index.php?title=Multitier\\_architecture&oldid=904835974#Three-tier\\_architecture](https://en.wikipedia.org/w/index.php?title=Multitier_architecture&oldid=904835974#Three-tier_architecture) [Accessed: 07-08-2019]

<sup>4</sup><http://web.archive.org/web/20190810190711/https://solid.inrupt.com/how-it-works> [Accessed: 10-08-2019]

- 
- Each Pod has an associated WebID URI, which is the web address of your profile page in your Pod
  - There are two ways of authenticating to your Pod
    - (a) WebID-OIDC, which is a protocol based on OpenID Connect. You log in with a username and a password, and an ID token is generated for authenticating subsequent requests to your Pod. Read more about the protocol on the official Github repository<sup>5</sup>
    - (b) WebID-TLS, which is a protocol that utilises TLS certificates for authentication. Read more about the protocol on the editor's draft of its specification<sup>6</sup>

### 3. Access Control through Web Access Control (WAC)

- Solid allows you to give specific users or groups of users access to parts of your Pod through the WAC protocol
- A `.acl` file for each container<sup>7</sup> and resource<sup>8</sup> in your Pod defines which users get what kind of access to the associated resource
- More detailed information can be found in the official Github repository<sup>9</sup> of the specification

### 4. RDF

- In Solid, data is either in the form of Linked Data or binary files<sup>10</sup>
- Linked Data is modelled using RDF<sup>11</sup>

---

<sup>5</sup><https://github.com/solid/webid-oidc-spec/tree/2b2c5a3625be7e0286066db9f29a41e6c3d80b6f> [Accessed: 10-08-2019]

<sup>6</sup><https://dvcs.w3.org/hg/WebID/raw-file/tip/spec/tls-respec.html> [Accessed: 29-07-2019]

<sup>7</sup><https://www.w3.org/TR/2015/REC-ldp-20150226/#ldpc> [Accessed: 10-08-2019]

<sup>8</sup><https://www.w3.org/TR/2015/REC-ldp-20150226/#ldpr-resource> [Accessed: 10-08-2019]

<sup>9</sup><https://github.com/solid/web-access-control-spec/tree/a71580b46a3ff124fa72d765a90432e488e96260> [Accessed: 10-08-2019]

<sup>10</sup><https://github.com/solid/solid-spec/tree/103b1e027356bd525e4cad0138e8288f4881df39#content-representation> [Accessed: 10-08-2019]

<sup>11</sup><https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/> [Accessed: 10-08-2019]



- 
- RDF represents data structured as graphs which are both labelled and multi-directed<sup>12</sup>
  - RDF can be serialised using various file formats
  - In Solid, you are most likely to encounter either Turtle<sup>13</sup> or JSON-LD<sup>14,15</sup>
  - Read more about content representation in the Solid specification<sup>16</sup>

## 5. HTTPS REST API

- All interaction between an application and your Pod is through an HTTPS RESTful API<sup>17</sup>
- This API is specified in the Solid where you can read about it in more detail<sup>18</sup>
- A WebSockets API<sup>19</sup> is also available for implementing the publish/subscribe pattern

## 6. Linked Data Notifications

- A protocol for push message communication between users
- Sender has permissions to send messages to Receiver's *inbox*
- Receiver can list the messages later and act on them later
- Read more about the protocol in its specification<sup>20</sup>

---

<sup>12</sup>[https://en.wikipedia.org/w/index.php?title=Resource\\_Description\\_Framework&oldid=904648151](https://en.wikipedia.org/w/index.php?title=Resource_Description_Framework&oldid=904648151) [Accessed: 10-08-2019]

<sup>13</sup><https://www.w3.org/TeamSubmission/2011/SUBM-turtle-20110328/> [Accessed: 10-08-2019]

<sup>14</sup><https://www.w3.org/TR/2014/REC-json-ld-20140116/> [Accessed: 10-08-2019]

<sup>15</sup><https://json-ld.org/> [Accessed: 10-08-2019]

<sup>16</sup><https://github.com/solid/solid-spec/blob/103b1e027356bd525e4cad0138e8288f4881df39/content-representation.md> [Accessed: 10-08-2019]

<sup>17</sup><https://github.com/solid/solid-spec/tree/103b1e027356bd525e4cad0138e8288f4881df39#https-rest-api> [Accessed: 10-08-2019]

<sup>18</sup><https://github.com/solid/solid-spec/blob/103b1e027356bd525e4cad0138e8288f4881df39/api-rest.md> [Accessed: 10-08-2019]

<sup>19</sup><https://github.com/solid/solid-spec/blob/103b1e027356bd525e4cad0138e8288f4881df39/api-websockets.md> [Accessed: 10-08-2019]

<sup>20</sup><https://www.w3.org/TR/2017/REC-ldn-20170502/> [Accessed: 10-08-2019]

---

## Prerequisites

Programming with Solid requires working knowledge of the following areas and this tutorial presumes the reader is familiar with these:

- Web development using HTML, JS, CSS
- Data modelling
- Client-server architecture
- Linked Data basics (RDF, SPAQRL, OWL)<sup>21</sup>

## Tutorial objectives

- Hosting your Pod
- Logging in using Solid
- CRUD (Creating, Retrieving, Updating, and Deleting) operations on data in a Solid Pod
- Learning about data access and data modelling

## Working with Solid

This tutorial will demonstrate the use of the following tools and libraries:

- `node-solid-server`<sup>22</sup> for hosting your own Pod for development
- `rdflib.js`<sup>23</sup> for creating and deleting containers and resources

---

<sup>21</sup><http://web.archive.org/web/20190807004544/https://solid.inrupt.com/docs/intro-to-linked-data> [Accessed: 10-08-2019]

<sup>22</sup><https://github.com/solid/node-solid-server/tree/5729fe534665a022f4cc41384af2c47a4f1eaf57> [Accessed: 10-08-2019]

<sup>23</sup><http://web.archive.org/web/20190805204816/http://linkeddata.github.io/rdflib.js/doc/> [Accessed: 05-08-2019]

- 
- [query-ldflex](#)<sup>24</sup> for retrieving and updating resources
  - [solid-auth-client](#)<sup>25</sup> for authentication

## How to get help

If you get stuck somewhere, you can look for help in the following places:

- Official forum of the Solid project<sup>26</sup>
- Gitter room [solid/chat](#)<sup>27</sup> for general queries or [solid/node-solid-server](#)<sup>28</sup>
- Stackoverflow<sup>29</sup>
- Github issues of the following projects. If you find a bug, you may also raise an issue
  - [node-solid-server](#)<sup>30</sup>
  - [rdflib.js](#)<sup>31</sup>
  - [query-ldflex](#)<sup>32</sup>
  - [solid-spec](#)<sup>33</sup>

---

<sup>24</sup><http://web.archive.org/web/20190805203631/https://solid.github.io/query-ldflex/> [Accessed: 05-08-2019]

<sup>25</sup><http://web.archive.org/web/20190810230225/https://solid.github.io/solid-auth-client/> [Accessed: 10-08-2019]

<sup>26</sup><https://forum.solidproject.org/> [Accessed: 10-08-2019]

<sup>27</sup><https://gitter.im/solid/chat> [Accessed: 10-08-2019]

<sup>28</sup><https://gitter.im/solid/node-solid-server> [Accessed: 10-08-2019]

<sup>29</sup><https://stackoverflow.com/tags/solid> [Accessed: 10-08-2019]

<sup>30</sup><https://github.com/solid/node-solid-server/issues> [Accessed: 10-08-2019]

<sup>31</sup><https://github.com/linkedata/rdflib.js/issues> [Accessed: 10-08-2019]

<sup>32</sup><https://github.com/solid/query-ldflex/issues> [Accessed: 10-08-2019]

<sup>33</sup><https://github.com/solid/solid-spec/issues> [Accessed: 10-08-2019]

---

## Getting a Pod

For this tutorial, you will require a Pod to develop against. There are two ways of getting a Pod<sup>34</sup>:

- Signing up with a Pod provider

You may get a Pod by signing up with one of the public Pod providers<sup>35</sup>.

- Hosting your own

You can also host your own Pod by simply installing `node-solid-server`. This is the recommended approach for this tutorial as you will want to explore the internals of the Pod yourself, which is not possible with public Pod providers.

## Installation of `node-solid-server`

### Installing Node.Js

In order to install the `node-solid-server`, you need Node.Js installed. We recommend installed Node.Js using `nvm`<sup>36,37</sup>.

Once you have Node.Js, run the following command:

### Installing `node-solid-server` from npm

```
npm install -g solid-server
```

---

<sup>34</sup><http://web.archive.org/web/20190810191205/https://solid.inrupt.com/get-a-solid-pod> [Accessed: 10-08-2019]

<sup>35</sup><http://web.archive.org/web/20190810191205/https://solid.inrupt.com/get-a-solid-pod> [Accessed: 10-08-2019]

<sup>36</sup>For Linux based systems, see `nvm`: <https://github.com/nvm-sh/nvm/tree/07b20d5008a480f7e579fd34e6d39919909206f4#node-version-manager---> [Accessed: 10-08-2019]

<sup>37</sup>For Windows, see `nvm-windows`: <https://github.com/coreybutler/nvm-windows/tree/0c58b2eed8fd515113157fd90a787f2348d7d331#node-version-manager-nvm-for-windows> [Accessed: 10-08-2019]

---

## Generating self-signed certificate for SSL

Solid mandates the use of TLS/SSL encryption with HTTP. For development and testing, you need to generate a self-signed certificate<sup>38</sup> as follows. For Linux-based systems, you can use the following command in the terminal:

```
openssl req -outform PEM -keyform PEM -new -x509 -sha256 -newkey rsa
:2048 -nodes -keyout ../privkey.pem -days 365 -out ../fullchain.
pem
```

## Initialising your Pod server

Now, you need to configure your Pod server. The easiest way is to run the wizard using the `solid init` command. For this tutorial, we recommend configuring the server with the default options.

## Running your Pod server

We will be using the `solid-test` command to start Solid in testing mode instead of the standard `solid` command because the `solid` command doesn't allow the server to execute requests with self-signed certificates. The `solid-test` command can be found in the `bin` directory of the `solid-server` global module directory. To add the command to your `PATH`, run the following command in the terminal:

```
export PATH = $PATH:$NVM_DIR/versions/node/'nvm current'/lib/
node_modules/solid-server/bin/
```

Then, to start the solid server (on a Linux-based system) in testing mode, run the following command in the terminal:

```
solid-test start
```

This will start the server at `https://localhost:8443`. If you configured some other port and hostname, replace them in the URL to access your server.

---

<sup>38</sup><http://www.selfsignedcertificate.com/> [Accessed: 10-08-2019]

---

## Registering for an account

Once your server is running, you need to create an account by navigating to the local Pod server's URL, `https://localhost:8443/register`. This will also generate a WebID which will look similar to `https://localhost:8443/profile/card#me`.

## Setting up test application

You will be creating a front-end Solid application using just HTML and Javascript. You need a text editor (like Visual Studio Code<sup>39</sup>, Atom<sup>40</sup>, Sublime Text<sup>41</sup>, or vim<sup>42</sup>) and a simple web server (like Node.Js' local-web-server<sup>43</sup> or Python's http.server<sup>44</sup>).

Create two files `index.html` and `app.js` in the same directory and open these files using your text editor.

Now, paste the following code to `index.html`:

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/solid-auth-client@2.3.0/
      dist-lib/solid-auth-client.bundle.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/rdflib@0.20.1/dist/
      rdflib.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@solid/query-ldflex@2
      .5.1/dist/solid-query-ldflex.bundle.js"></script>
    <title>Hello, Solid!</title>
  </head>
  <body>
    <h2>Hello, Solid!</h2>
    <p id="login">
      You are not logged in.
      <button id='loginBtn '>Log in</button>
    </p>
```

---

<sup>39</sup><https://code.visualstudio.com/> [Accessed: 10-08-2019]

<sup>40</sup><https://atom.io/> [Accessed: 10-08-2019]

<sup>41</sup><https://www.sublimetext.com/> [Accessed: 10-08-2019]

<sup>42</sup><https://www.vim.org/> [Accessed: 10-08-2019]

<sup>43</sup><https://www.npmjs.com/package/local-web-server> [Accessed: 10-08-2019]

<sup>44</sup><https://docs.python.org/3/library/http.server.html> [Accessed: 10-08-2019]

```
<p id="logout">
  You are logged in as <span id="user"></span>.
  <button id='logoutBtn '>Log out</button>
</p>
<p id="loggedin">
  <label for="profile">Profile: </label>
  <input id="profile">
  <button id="getNameBtn">getName</button>
  <label for="fullName">Full name: </label>
  <span id="fullName"></span>

  <br><br>
  <label for="postTitle">Enter Post title: </label>
  <input id="postTitle">
  <button id="createPostBtn">createPost</button>
  <label for="newPostTitle">Post created with URI: </label>
  <span id="newPostUri"></span>

  <br><br>
  <label for="creatorName">Creator Name: </label>
  <span id="creatorName"></span>

  <br><br>
  <button id="addCreator">addCreator</button>

  <br><br>
  <label for="newCreator">Enter new creator: </label>
  <input id="newCreator">
  <button id="changeCreator">changeCreator</button>

  <br><br>
  <button id="deletePost">deletePost</button>

  <br><br>
  <button id="listPublicContents">listPublicContents</button><br>
  <span id="listPublicContentsResult"></span>
</p>
```

---

```
    <script src="/app.js"></script>
  </body>
</html>
```

The HTML file lays out the User interface of the tutorial application, and adds the libraries we need – rdfib.js, query-ldflex, and solid-auth-client.

In app.js, place the following code:

```
// Utility function that toggles display of elements
function _toggleDisplay(el, flag) {
  if (flag) {
    el.style.display = "block";
  } else {
    el.style.display = "none";
  }
}

document.addEventListener('DOMContentLoaded', async () => {
  // Step 1

  // Step 2

  // Step 3

  // Step 4

  // Step 5

  // Step 6

  // Step 7

});

// getName()

// createPost()

// addCreatorAttribute()
```



```
// changeCreatorAttribute()

// deletePost()

// listPublicContents()
```

This Javascript code simply adds a utility function we'll need and attaches an event listener to the document, which runs a callback when the document is ready. We will add functionality to this application.

Once you have created these files, run your web server and navigate to `/index.html`. Every time you complete a Step, you may reload the page and test the functionality.

## Step 1: Logging In to your Pod

We add a listener to the login button, and one listener to the logout button.

The login listener simply opens a popup which allows for selection of Pod provider and log-in. The code for the popup is contained in a file called `popup.html` which is available in the accompanying Github repository of this tutorial<sup>45</sup> and has been adapted from Solid's `profile-viewer-tutorial`<sup>46</sup>.

The logout listener calls the `logout` method of the `solid-auth-client`.

We also call the `trackSession` method of `solid-auth-client`. This method keeps track of the session, i.e., the log-in status, and we pass a callback to it to update the elements based on the log-in status.

Add the following code under the comment `// Step 1` in `tapp.js`:

```
solid.auth.trackSession(session => {
  const loggedIn = !!session;
  _toggleDisplay(document.getElementById('login'), !loggedIn);
  _toggleDisplay(document.getElementById('logout'), loggedIn);
  _toggleDisplay(document.getElementById('loggedin'), loggedIn);
});
```

<sup>45</sup><https://github.com/akashdeep-singh/solid-tutorial/blob/6339397608f1b91dd4c170fdcdace44f70dc4ae4/popup.html> [Accessed: 11-08-2019]

<sup>46</sup><https://github.com/solid/profile-viewer-tutorial/blob/tutorials/lunch-break/steps/09/popup.html> [Accessed: 10-08-2019]

```

document.getElementById('user').textContent = session && session.
  webId;

if (session) {
  document.getElementById('user').textContent = session.webId;
  if (!document.getElementById('profile').value) {
    document.getElementById('profile').value = session.webId;
  }
}
});

document.getElementById('loginBtn').addEventListener('click', () =>
  solid.auth.popupLogin({ popupUri: 'popup.html' }));

document.getElementById('logoutBtn').addEventListener('click', () =>
  solid.auth.logout());

```

## Step 2: Getting data

Now that we have authenticated with the Pod, we are interested in getting data from the Pod. For this, we use the `query-ldflex` library. This library makes it easy to read data from the Pod, without having to deal with the complexities of RDF graphs. Here, we will get the full name of the Pod's authenticated user from the user's profile. Recall that the WebID URI points to the user's profile. The user's profile has the full name of the user modelled using the `fn` predicate of the `vcard` ontology<sup>47</sup>. We can be certain that this predicate exists because it is created during registration.

Using `query-ldflex`, the following statement can be used to get a user's full name if the WebID is `https://localhost:8443/profile/card#me`:

```

// Example

const user = solid.data[https://localhost:8443/profile/card\#me]['
  vcard:fn'];

```

<sup>47</sup><https://www.w3.org/TR/2014/NOTE-vcard-rdf-20140522/> [Accessed: 11-08-2019]

---

We don't need to give profile any additional context to query-ldflex to understand what vcard means, because it includes a set of vocabularies in its context by default<sup>48</sup>.

To complete the code for this step, add the following code under the comment **Step 2** in app.js:

```
document.getElementById('getNameBtn').addEventListener('click',
  async () => {
    document.getElementById('fullName').textContent = await getName();
  });
```

and add the following code under the comment `// getName()`:

```
async function getName () {
  const session = await solid.auth.currentSession();
  const user = solid.data[session.webId];

  // gets the vcard:fn predicate from user's profile
  return (await user['vcard:fn']);
}
```

### Step 3: Creating a resource

Now, we will create a simple post in our Pod. The post will have only one predicate: `title` defined by the `dcterms`<sup>49</sup> vocabulary. To create a resource, we can't use query-ldflex because it doesn't have support for creation of resources and containers yet. We will use `rdflib.js`. `Rdflib.js` requires some additional ceremony compared to query-ldflex, but is much more versatile. `Rdflib.js` is meant for graph manipulation. We also need to declare the namespaces of the predicates we use before operating on them.

---

<sup>48</sup><https://github.com/solid/query-ldflex/blob/ffb9927b3cecc0a698f6e5867322b3bb618ce1bb/src/context.json> [Accessed: 10-08-2019]

<sup>49</sup><https://www.dublincore.org/specifications/dublin-core/dcmi-terms/2012-06-14/> [Accessed: 10-08-2019]

---

In order to create a post, we need to initialise a **graph**, and add the triple to it. Then, we PUT the graph onto an instance of **Fetcher** to persist the post on the user's Pod. We will create the post in the `/public` container of the user's Pod.

To complete the code for this step, add the following lines of code under the comment `// Step 3` in `app.js`:

```
document.getElementById('createPostBtn').addEventListener('click',
  async () => {
    const postTitle = document.getElementById('postTitle').value;
    if (postTitle){
      const session = await solid.auth.currentSession();
      const user = solid.data[session.webId];
      const storageUri = (await user['pim:storage']).value;
      const newPostUri = await createPost(storageUri, postTitle);
      document.getElementById('newPostUri').textContent = newPostUri;
    } else {
      alert('Post title cannot be empty!');
    }
  });
```

and the following lines of code under the comment `// createPost()`:

```
async function createPost (storageUri, title) {
  // declares the namespace for the dcterms vocabulary
  const DCTERMS = $rdf.Namespace('http://purl.org/dc/terms/');

  const store = $rdf.graph(); // initialise graph
  const timeout = 5000 // set timeout to 5000 ms
  const fetcher = new $rdf.Fetcher(store, timeout); // initialise
  // fetcher object
  const postUri = storageUri + '/public/solidTutorialTestPost-' + Date
    .now(); // create the post's URI
  const entry = store.sym(postUri); // convert the
  // post URI to the symbol object
  store.add(entry, DCTERMS('title'), title, entry); // add a triple
  // to the graph
  await fetcher.putBack(entry); // putBack
  // creates the resource at the URI using PUT
```

```
    return postUri;
}
```

## Step 4: Adding predicates to the resource

Adding additional predicates to a resource is supported by query-ldflex. It is as easy as calling the `add` method on the resource, which creates a new predicate with the passed value. We will create a predicate `creator` defined by the `dterms` vocabulary.

Add the following code under the comment `// Step 4` in `app.js`:

```
document.getElementById('addCreator').addEventListener('click',
  async () => {
    const postUri = document.getElementById('newPostUri').textContent;
    if (postUri) {
      document.getElementById('creatorName').textContent = await
        addCreatorAttribute(postUri);
    } else {
      alert('Post has not been created yet!');
    }
  });
```

and the following code under the comment `// addCreatorAttribute()`:

```
async function addCreatorAttribute (postUri) {
  // adds the dct:creator predicate to the post
  await solid.data[postUri]['dct:creator'].add((await getName()).value
  );
  return await solid.data[postUri]['dct:creator'];
}
```

## Step 5: Modifying a predicate of the resource

Modifying an existing predicate is similar to adding one. To modify, we can use the `set` method of query-ldflex, which replaces the existing value of the resource with the one passed to it. In fact, `set` is not just useful for updating an existing predicate, it

---

can also be used to create one, i.e., upsert operation. We will update the value of the creator to the newly input value.

Add the following code under the comment `// Step 5` in `app.js`:

```
document.getElementById('changeCreator').addEventListener('click',
  async () => {
    const postUri = document.getElementById('newPostUri').textContent;
    const newCreator = document.getElementById('newCreator').value;
    if (postUri) {
      if (newCreator) {
        document.getElementById('creatorName').textContent = await
          changeCreatorAttribute(postUri, newCreator);
      } else {
        alert('New creator name cannot be empty!');
      }
    } else {
      alert('Post has not been created yet!');
    }
  });
```

and the following code under the comment `// changeCreatorAttribute()`:

```
async function changeCreatorAttribute (postUri, newCreator) {
  // updates the dct:creator predicate of the post
  await solid.data[postUri]['dct:creator'].set(newCreator);
  return await solid.data[postUri]['dct:creator'];
}
```

## Step 6: Deleting the resource

To delete a resource, we will again use `rdflib.js`. Deleting is supported using the `delete` method of `Fetcher` by passing the URI of the resource to be deleted. For deletion, we use an empty graph, as we don't deal with any data.

Add the following code under the comment `// Step 6` in `app.js`:

```
document.getElementById('deletePost').addEventListener('click',
  async () => {
```

```

const postUri = document.getElementById('newPostUri').textContent;
if (postUri) {
  await deletePost(postUri);
  document.getElementById('newPostUri').textContent = '';
} else {
  alert('Post has not been created yet!');
}
});

```

and the following code under the comment `// deletePost()`:

```

async function deletePost (postUri) {
  let store = $rdf.graph();
  let timeout = 5000 // 5000 ms timeout
  let fetcher = new $rdf.Fetcher(store, timeout);
  return await fetcher.delete(postUri);          // deletes the
  resource at the URI
}

```

## Step 7: Listing contents of a container

Sometimes, we need to list the contents of a container. For example, if we wanted to list the posts we created. For this, we can use `query-ldflex` to get the `contains` predicate of the container. This predicate is defined in the LDP<sup>50</sup> vocabulary. We will list the

Add the following code under the comment `// Step 7` in `app.js`:

```

document.getElementById('listPublicContents').addEventListener('
  click', async () => {
  document.getElementById('listPublicContentsResult').innerHTML =
    '';
  const session = await solid.auth.currentSession();
  const user = solid.data[session.webId];
  const storageUri = (await user['pim:storage']).value;
  document.getElementById('listPublicContentsResult').innerHTML = (
    await listPublicContents(storageUri)).join('<br>');
}

```

<sup>50</sup><https://www.w3.org/ns/ldp> [Accessed: 10-08-2019]

---

```
});
```

and the following code under the comment `// listPublicContents()`:

```
async function listPublicContents (storageUri) {
  const publiContents = solid.data[storageUri+'/public/'];
  const list = [];

  // gets the value of the ldp:contains predicate of the public
  // container of the Pod
  // the value is a list, so we have to loop through it to get
  // individual contents of the Pod
  for await (const item of publiContents['ldp:contains']) {
    list.push(item.toString());
  }
  return list;
}
```

## Next Steps: Further learning

We have successfully completed a simple Solid application with CRUD operations on our Pod using 3 important Linked Data libraries. The next logical step would be to learn more advanced concepts:

- Permissions using WAC
- Communication using LDN

Both of these can be implemented using `rdflib.js` as they involve creating resources.

## Recommended Reading

- Make a Solid app on your lunch break<sup>51</sup>, which also serves as the inspiration for this tutorial.

---

<sup>51</sup><http://web.archive.org/web/20190807004517/https://solid.inrupt.com/docs/app-on-your-lunch-break> [Accessed: 10-08-2019]



- 
- The official *Introduction to the Solid Specification*<sup>52</sup>
  - Inrupt Inc's work on Solid<sup>53</sup>
  - The open letter by Sir Tim Berners-Lee<sup>54</sup>
  - Designing a Linked Data developer experience<sup>55</sup>
  - Shaping Linked Data apps<sup>56</sup>

---

<sup>52</sup><http://web.archive.org/web/20190810191233/https://solid.inrupt.com/docs/intro-to-solid-spec> [Accessed: 10-08-2019]

<sup>53</sup><http://web.archive.org/web/20190810211330/https://inrupt.com/solid> [Accessed: 10-08-2019]

<sup>54</sup><http://web.archive.org/web/20190810211649/https://inrupt.com/blog/one-small-step-for-the-web> [Accessed: 10-08-2019]

<sup>55</sup><http://web.archive.org/web/20190805204059/https://ruben.verborgh.org/blog/2018/12/28/designing-a-linked-data-developer-experience/> [Accessed: 05-08-2019]

<sup>56</sup><http://web.archive.org/web/20190806150750/https://ruben.verborgh.org/blog/2019/06/17/shaping-linked-data-apps/> [Accessed: 06-08-2019]