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

School of Computer Science and Statistics

Making personal data Solid

An evaluation of the capabilities of Solid in enterprise-level application development

Dylan Storey

A Dissertation presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

MAI Computer Engineering

Supervisor: Prof. Declan O'Sullivan

April 2022

Declaration

☐ I agree that this thesis was completed in line with the plagiarism provisions in the General Regulations of the University Calendar for the current year, found at http://www.tcd.ie/calendar.

☐ I have completed the Online Tutorial on avoiding plagiarism 'Ready Steady Write', located at http://tcd-ie.libguides.com/plagiarism/ready-steady-write.

☐ I agree that this thesis will not be publicly available but will be available to TCD staff and students in the University's open access institutional repository on the Trinity domain only.

Signed:

Date:

Acknowledgements

I would like to acknowledge Prof. Declan O'Sullivan for continued support and advice throughout this project. I would also like to thank with Prof. Lucy Hederman, Prof. Gaye Stephens, researchers at Adapt and in Trinity College for their insights into professional practice in the healthcare sector and data engineering.

Further acknowledgements must be extended to the Solid community for their continued support and encouragement to continue this project, particularly Emelia Smith and Justin Bingham for their assistance and insights into working professionally with Solid.

Finally, I would like to acknowledge my absolute gratitude to my family and friends who have shown constant support throughout this year to complete this project.

Abstract

The healthcare industry is under increasing threat from ransomware attackers for its large volumes of valuable and sensitive data stored on fragmented systems, managed by ill-equipped system administrators. Trends have shown that this pattern has grown over the course of the COVID-19 pandemic via new exploits and worldwide collection of health data related to vaccinations. This project has attempted to address a solution for this problem by developing a concept application to store and access health data in decentralised data storage using the Solid Protocol.

Several applications have already been released using the Solid Protocol, proving that development in the space is possible. However, this project differed slightly from other approaches in that it attempted to build an enterprise application for the healthcare industry, dealing with extremely sensitive data and aimed for use by individuals across several healthcare processes.

By way of this development, the capabilities of Solid with regard to building complex organisational applications were evaluated. It was found that Solid technologies are capable of most fundamental development tasks, but a few critical shortcomings with the current implementations would deter most organisations from investing the additional time and resources required to develop a commercial product with Solid.

Contents

Chapter One 1
1. Introduction 1
2. Solid Concepts 3
2.1 Solid Applications 3
2.2 Solid Resources 4
3. Background 5
3.1 The Semantic Web 5
3.2 Linked Data 6
3.3 Decentralisation 7
4. Motivation 8
5. Research Question 10
6. Overall Goal and Objectives 11
Chapter Two 13
7. Literature Review 13
Section Overview 13
7.1 Decentralisation 13
7.2 State of the Art of Solid 16
7.3 Security in Healthcare IT Systems 21
Chapter Three 24
8. Design 24
Section Overview 24
8.1 Phase 1 – Exploration 24
8.2 Phase 2 – Application Planning 27
9. Implementation 36
Section Overview 36
9.1 Setup 36
9.2 Login and Health Data Configuration 37
9.3 Uploading Records 40
9.4 Accessing Records 43
9.5 Managing Record Access 44
9.6 Deployment 46
Implementation Summary 48
Chapter Four 49
10. Evaluation 49

Section Overview 49
10.1 Application Evaluation 49
10.2 Evaluation of Contribution to Solid 55
10.3 Solid Evaluation 57
Evaluation Summary 61
Chapter Five 62
11. Conclusion 62
12. Future Work 64
12.1 Future Work in Solid 64
12.2 Future Development Work 64
13. Final Remark 66
References 67
Appendices 79
A1 – Original Application Requirements 79
A2 – Application Gantt Chart 82
A3 – Development Diary 83
A4 – Application Demonstration Presentation 99

List of Figures

Figure 1 - Sharing data with Solid [7] 3
Figure 2 - Solid applications accessing Pod data [78] 4
Figure 3 - RDF knowledge graph example 5
Figure 4 - Linked datasets example [105] 6
Figure 5 - Number of incidents per sector (April 2020 - July 2021) [33] 8
Figure 6 – Decentralisation of organisations per region [114] 15
Figure 7 - PodPro tool in use 20
Figure 8 - Number & value of ransomware attacks observed by ENISA (April 2020 - July 2021) [33] 22
Figure 9 - Use case diagram 30
Figure 10 - Application data inside Pod 33
Figure 11 – Application authentication flow 34
Figure 12 - Login process flow 37
Figure 13 - Institution metadata example in Turtle 38
Figure 14 - Register process flow 39
Figure 15 - Application home page 40
Figure 16 - Upload appointment process flow 41
Figure 17 - Upload diagnosis, prescription or general record process flow 42
Figure 18 - Access medical records process flow 43
Figure 19 - Pod diagram 44
Figure 20 - Manage record access flow 45
Figure 21 - Insurance request flow 46
Figure 22 – Test run specification 50
Figure 23 - Mean & standard deviation of feedback form results 53

List of Tables

Table 1 - Solidcommunity.net authorisation levels 27
Table 2 - Developed basic operations 27
Table 3 - List of user types for application 28
Table 4 - Draft of application use cases 29
Table 5 - Application test user credentials & WebIDs 48
Table 6 - Distribution of code in application implementation 48
Table 7 - Application technical critiques 51
Table 8 - Demonstration feedback form questions 52
Table 9 - Application procedural critiques 54
Table 10 - Inrupt Solid library critiques 59
Table 11 - Solid specification critiques 60
Table 12 - Gitter chat activity 61
Table 13 - Stack Overflow Solid interactions 61

Document Organisation

This dissertation is broken up into five chapters; each containing a varying amount of sections and sub-sections. Where a section is notably large, it is prefixed with an outline of the contained sub-sections. There are also four appendices related to the project, included at the end of the dissertation.

Chapter One details an introduction to the project, including descriptions of common terminology and the goals that the project hopes to achieve.

Chapter Two describes the state-of-the-art in research areas that the project has an impact on, including strengths and shortcomings of other projects in related areas.

Chapter Three describes the process of designing and implementing the primary work item of this project and describes the capabilities of the developed product.

Chapter Four describes the methods and results of evaluating the project.

Chapter Five summarises the results of the project evaluation and provides an outline of work remaining in the project.

Appendix 1 lists the initial requirements that were designed for the developed application.

Appendix 2 shows the project Gantt chart that was used as a schedule for development work.

Appendix 3 is the diary of notes maintained throughout developing the application.

Appendix 4 shows slides from the presentation that was used in demonstration sessions during the project evaluation.

Chapter One

1. Introduction

In the modern day, where huge amounts of personal data is shared with and between applications with or without our knowledge or consent [13, 14], self-regulated data distribution is an area that should be of huge importance to developers. This has been expressed by analysts of today's web for a number of years, but concerns grow louder and more widespread as big tech companies gather an increasing amount of personal data about the individual to enable model-driven product targeting.

Solid [7] is an emerging technology that offers a potential solution to this problem by keeping the individual as the owner and controller of the data; heralding a degree of personal data ownership and portability that has not been seen since the conception of the internet. This is made possible by a set of standards and tools that access data stored in a single, decentralised web store, accessible from anywhere on the web.

Solid applications are front-end interfaces which allow users to access and update data held in these data stores. While most of the currently available Solid applications [40] have been designed to show that the technology is suitable for use in recreational and social settings, none have been designed for use by a complex organisation. This project will attempt to evaluate the maturity of the Solid specification from the perspective of a software developer by designing and implementing a proof-of-concept application for the healthcare industry which could benefit from the advantages that Solid offers.

This industry has been selected for two reasons. Firstly, healthcare operates with the most sensitive and private data, tightly regulated by data protection legislation and procedures used in professional practice. An aim of this project is to investigate how the technologies of Solid are capable with handling sensitive data. Secondly, the healthcare industry has been a growing target for ransomware attacks due to the large volumes of valuable patient data held in hospital systems. An attack on the Irish healthcare service in 2021 caused widespread disruptions to healthcare services and estimated millions of euro in damages. This project will attempt to address this problem by offering the sector an alternative to traditional data storage techniques, by storing data using the Solid approach. The concept application will then be validated by professionals in the healthcare industry to evaluate its potential usability in the sector.

By developing this application, an evaluation of the entire Solid ecosystem will be gathered to identify potential strengths and weaknesses from the perspective of a software developer. This ecosystem will be assisted in any way possible throughout the project and beyond, to foster engagement from future developers and advance the realisation of the many advantages that it offers to society.

2. Solid Concepts

Solid (SOcially LInked Data) is a set of standards and tools created in 2016 as an academic project by a team at MIT [8], led by the creator of the W3C and World Wide Web itself, Sir Tim Berners-Lee. Solid describes itself as a 'mid-course correction' for the Web as it functions today [9], where Berners-Lee believes it could be a realisation of his original vision of allowing users to exchange public and private data securely. It is designed for use in the Semantic Web [10], the next generation of the internet still very much in early adoption. As such, Solid relies heavily on principles of Linked Data [11] and best practices recommended by the W3C [12].

Solid offers the ability to securely store any type of personal data in Pods, currently available from a small set of trusted Pod Providers [52] or by hosting an individual Pod server, which must adhere to the Solid Protocol [58]. Pod Providers currently provision Pod storage at various geographical locations around the world, free of charge.

Pods are personal decentralised web stores where access to any part of the contained data can be granted or revoked by the Pod owner. In this sense, one of the key advantages of Solid storage is that data only exists in a single location, where the owner can tightly control the access that other individuals or applications have. An illustration of sharing sections Solid Pod data is shown below in Figure 1.

Figure 1 - Sharing data with Solid [7]

2.1 Solid Applications

Applications which implement the Solid specification to interact with data stored in Pods are called Solid applications. These applications facilitate user authentication with a varying amount of Pod Providers, depending on the specific application implementation. This means that in the context of Solid applications, Pod Providers assume the role of identity providers.

Therefore, unless an application implements a custom Solid server to provide Pods to users, there is no need for Solid applications to be involved with user credentials.

Pods essentially become an interrelated collection of application data for Solid applications, where applications access data stored in Pods to serve some purpose to users. This access to Pod data is subject to what has been granted by the owner of any given Pod. An example of how three Solid applications interact with sections of an individual's data stored in their Pod is shown below in Figure 2.

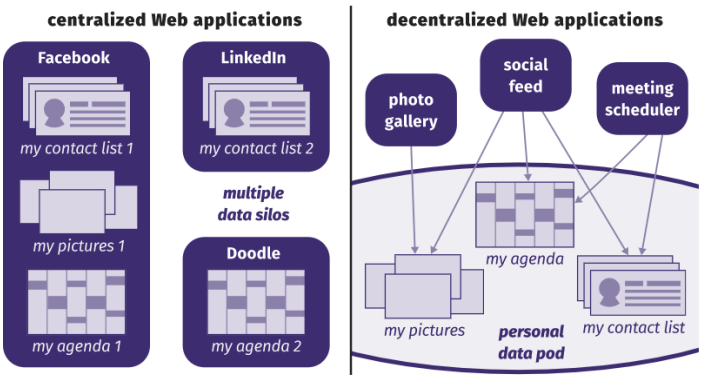


Figure 2 - Solid applications accessing Pod data [78]

2.2 Solid Resources

Resources in Solid are defined as a space of URIs [6] in which data can be accessed by a HTTP [107] request. Resources could be an individual file or object, or a collection of objects contained within another Resource. Four Resource types will be referenced throughout this project, with descriptions gathered from official sources [6, 91, 45]:

Container – This is a Resource that can contain other Resources or Containers, analogous to a folder on a traditional file system. Containers have their own access rules for the contained Resources. The URI for a Container ends with a slash ('/'). A Solid Container behaves according to the Linked Data Platform (LDP) Basic Container type found in the LDP Vocabulary [127].

Thing – A set of data or properties about a particular entity, e.g. a novel. Things are saved as part of a SolidDataset.

SolidDataset – A Resource that contains a set of Things, e.g. novels written by an author.

Auxiliary Resource – A Resource to provide supplementary information about another Resource such as metadata, authorisation conditions or shape constraints.

3. Background

Aside from Solid, this project relies heavily on some background knowledge of three core concepts:

1. The Semantic Web
2. Linked Data
3. Decentralisation

These concepts will be briefly explained in this section.

3.1 The Semantic Web

The Semantic Web is the next logical step for the Web where data becomes readable both by humans and machines. This will be achieved by transitioning from the document-based approach to data representation that is used today, to a graph-based approach where data and relationships between data are defined in a machine-readable format known as Resource Description Framework (RDF) [37, 62]. RDF syntax is based around triples of information, where a data entity (subject) is associated by some relationship (predicate) to some value (object); which can be either another data entity or a literal value. Individual data entities and relationships in RDF are located by unique HTTP IRIs [107].

Groups of RDF triples form knowledge graphs of information that define one or more resource. An example of a simple knowledge graph describing an individual is shown below in Figure 3. Here the subject of each triple is 'Person#2913', which has a unique IRI. These triples reference either a literal value or another IRI, of which other statements could be made in a separate knowledge graph.

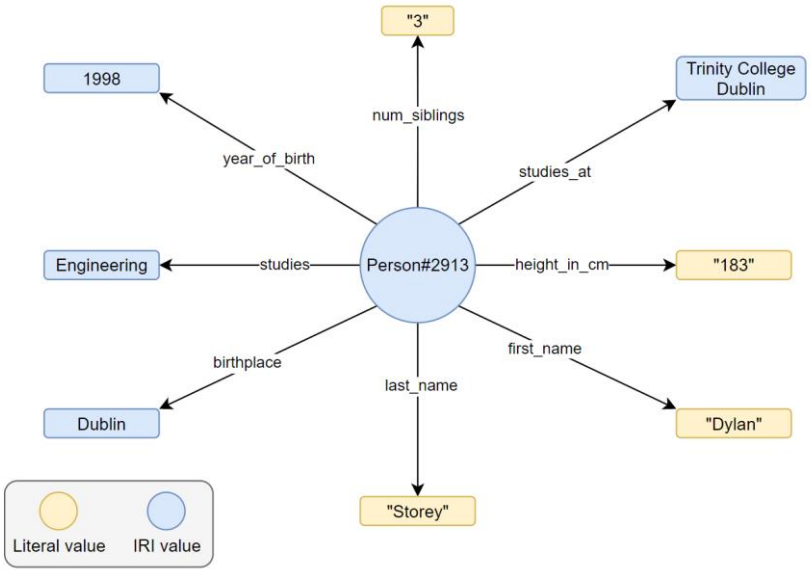


Figure 3 - RDF knowledge graph example

In the Semantic Web, collections of resources in RDF are known as vocabularies or ontologies [104] and they provide a number of definitions for entities in a particular domain. These ontologies provide a class-like structure to entities, including data and relationship constraints.

Ontologies create a Web of Data similar to data that would be found in conventional databases. By defining data in a format that machines can process, it will enable computers to do most of the work in finding related data and will allow systems to be developed to exchange information easier across the Web [103].

3.2 Linked Data

To make this Web of Data a reality, a huge amount of data needs to be available in this format that can be reached from any part of the Web, where data from individual datasets can interact with each other for querying and enriching the collective knowledge of real-world entities. This aggregation of related datasets is known as Linked Data [11, 35].

Linked Data must be reachable by tools used in The Semantic Web, for example, the SPARQL query language [106]. This discovery of other datasets, for creating inferences based on shared knowledge, is a pivotal aspect of Linked Data. A visual representation of this collection of datasets is available at The Linked Open Data Cloud [105].

An example of the Linked Open Data Cloud as it was in September 2010 is shown below in Figure 4. This shows how datasets across industries are connected with each other to varying degrees. The number of linked datasets has risen from 203 to 1301, as of May 2021.

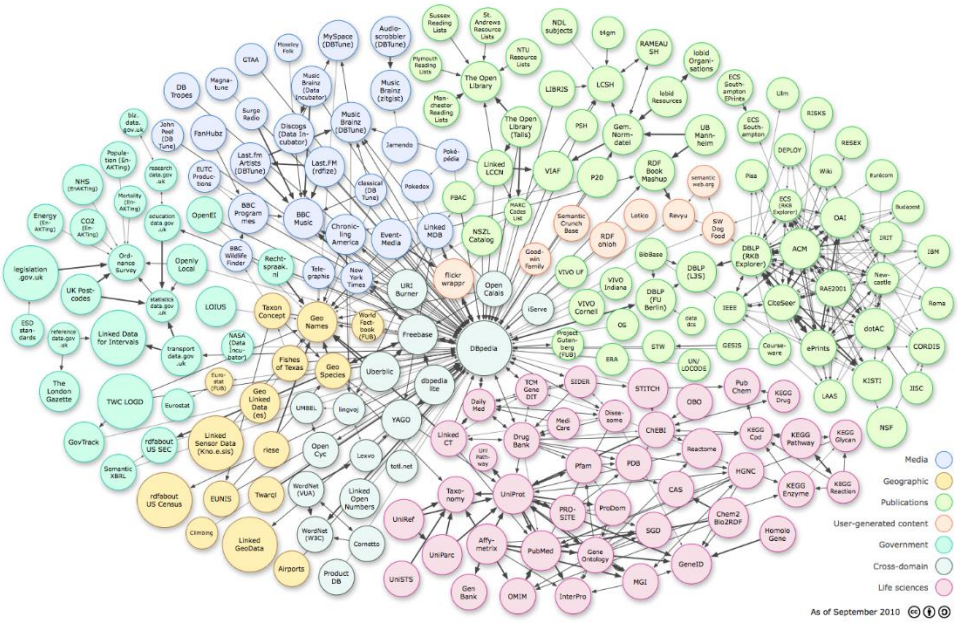


Figure 4 - Linked datasets example [105]

3.3 Decentralisation

The concept of decentralisation in this project refers to the natural progression of the internet if The Semantic Web becomes widespread, using Linked Data stored in Solid Pods or other decentralised data stores. Data will no longer be replicated and isolated across individual organisations' databases; and will return to the control of the individual, where it can be shared out to parts of the web as required.

Solid Pods can be hosted virtually anywhere, meaning that data would be accessible from anywhere on the web. This means that Solid can be thought of as decentralised data storage and traditional database storage can be thought of a centralised data storage.

Service providers on a decentralised web would access data across this array of individual stores (instead of owning it), to provide functionality that would benefit individuals more than the current model which compels organisations to aggregate independent data silos about their users.

Decentralising the web will not be an easy task. It requires a massive amount of data migration from existing databases, architecture shifts for web services and a complete redefinition of data protection legislation. This will need to be a gradual and considerate process, addressing areas of particular need first.

4. Motivation

The Solid Project is appealing to any individual with an interest in privacy and data control. True data ownership and transparency of access to data are ideals that seem distant in today's perspective; where dominant tech corporations fight to gather the maximum amount of data about the individual. With these collections of personal data, they are often the target for hacker groups [130, 131] or choose to sell it to third parties [13]. Solid offers a fix for these problems, but widespread adoption will only come if a service is made available to the public that offers functionality unsupported by anything other than Solid.

Data protection and security in the healthcare sector has become a topic of major concern in recent years [1, 2], particularly after a data breach of the state-owned healthcare system in Ireland (HSE) suffered a cyberattack that saw hundreds of individuals' personal data leaked online [3]. This attack was made possible and attractive to perpetrators by storing volumes of sensitive data on fragmented and vulnerable IT systems used by the HSE [4]. It caused huge disruptions to healthcare across the country [5] as well as millions of euro in estimated damages [3].

This trend has been seen across Europe, as the European Union Agency for Cybersecurity (ENISA) [51] stated in the report published in 2021 that there has been a "surge in healthcare sector related data breaches" due to the collection of scientific information related to the COVID-19 vaccine, and will continue to be heavily targeted by ransomware groups as long as the pandemic lasts [33]. Figure 5 below shows the distribution of ransomware incidents as reported by ENISA, where healthcare is the fourth-highest on the list.

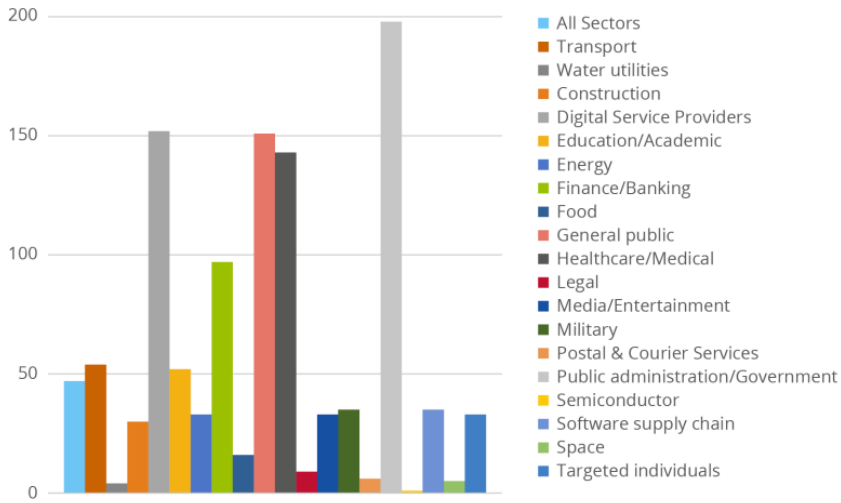


Figure 5 - Number of incidents per sector (April 2020 - July 2021) [33]

An application developed using principles from the Solid Protocol [6] offers a potential solution to this problem by allowing users to securely store decentralised data in online personal web servers and grant or revoke access to other users as required [7].

By developing an application that could store patient data in Solid pods, it would eliminate the need for a single collection of data held in medical institution systems, and make the healthcare sector less of an appealing target to attackers.

5. Research Question

The research question that this project will attempt to answer is as follows:

“To what extent is it feasible to develop an enterprise application with Solid?”

For this dissertation, the healthcare domain has been chosen as a challenging environment for which to try to develop a prototype enterprise application using Solid standards.

6. Overall Goal and Objectives

The overall goal of this work is to evaluate the maturity of the Solid specification as it stands today from the perspective of a software developer. This will be primarily evaluated by the technologies currently supported by the Solid specification, quality of the available documentation and the activity and support of the developer community.

A number of objectives to achieve this goal are listed below with targets that will allow a reasonable evaluation of the Solid specification when achieved. These targets will be referenced later in the evaluation of the project.

O1. Develop a functional application which will serve the purpose of storing and accessing medical health records in a Solid data Pod.

Target ID	Description
T1.1	Allow explicitly authorised users to view medical records belonging to an individual.
T1.2	Allow explicitly authorised users to grant permissions to other users to view or update medical records.
T1.3	Allow implicitly unauthorised users basic access to certain information in the event of an emergency.

O2. Evaluate the practical usability of the developed application with test users experienced with the healthcare sector and/or data engineering.

Target ID	Description
T2.1	Establish live testing environment for the developed application to be hosted on.
T2.2	Validate application functionality with individuals currently working in healthcare informatics.
T2.3	Develop a PSSUQ for application test users and present results.
T2.4	Compare PSSUQ results with those from a similar application developed using traditional technologies.

O3. Obtain ethical approval for the desired functionality of the proposed application.

Target ID	Description
T3.1	Consider all aspects of the ethical implications, both short-term and long-term, that the application may have on the end user and society.
T3.2	Complete Ethics Committee project checklist and a subsequent ethical application form, if required.

O4. Maintain a detailed diary of any obstacles faced while developing with Solid and solutions or mitigations employed.

Target ID	Description
T4.1	Log lessons learned/tasks completed for each day of development.
T4.2	Discuss effectiveness of the Solid documentation and/or Solid community in developing features or overcoming obstacles.

O5. Help to grow the Solid developer community by engaging with development topics as much as possible.

Target ID	Description
T5.1	Engage with other Solid developers to discuss solutions for common issues and share experiences.
T5.2	Earn two trust member levels in the Solid community forum [15].
T5.3	Make application code repository public and list achievements to the community, enabling Solid developers to use the application source code in their own development.

Chapter Two

7. Literature Review

Section Overview

This project touches on a wide range of topics and therefore, research had to be placed into a large domain to gain an adequate understanding of how the developed application would fit into the current landscape. This chapter describes the approach to researching this landscape. Each subsequent section describes the approach to finding literature for that area, as well as a description of the state of the art and efforts that are being made to advance the field. The three domains that are most relevant to this project are listed as:

7.1 Decentralisation.

7.2 State of the art of Solid.

7.3 Security in healthcare IT systems.

The societal implications of widespread decentralisation are considered in section 7.1, along with the current status of enterprise decentralisation. Next, the current state of Solid is described in section 7.2, evaluated in terms of the documentation supplied to developers, technical literature on Solid and similar projects discovered through engagement with the Solid community. Finally, some security cases in healthcare IT systems are examined in section 7.3 to identify the current threats that the industry faces and determine if there are gaps or critical areas that this project could help to address.

7.1 Decentralisation

Decentralisation has been a topic of interest for web analysts for many years, many of whom have fears for the state of data privacy as it stands today. However, the term ‘decentralisation’ has multiple meanings across literature that make finding work relevant for this project difficult. For example, decentralisation can mean the distributing of power and decision-making away from a central government to regional authorities [128]. Another definition of the term as described by the co-founder of the popular cryptocurrency, Ethereum, is for a system with individual nodes co-operating towards a collective purpose [129].

In the context of this project, decentralisation refers to the process of transferring vast amounts of data from centralised databases, either online or privately accessed, to decentralised data stores. This is no easy feat, but it would bring a level of transparency with

data control and portability that is desirable in the current climate; where data is replicated and stored across an unknown number of service providers' private storage.

7.1.1 Societal Implications

The need to move from the today's web, known as Web 2.0 [53], towards a better solution has been expressed by many analysts in recent years. The inventor of the web itself [54] has written multiple articles expressing concerns for the web as it has evolved from its creation [55,25,26], where his calls to action have become more urgent in recent years.

One resource that has been particularly helpful in understanding the status of decentralisation are the publications by Ruben Verborgh [21], a Professor of decentralised Web technology in Ghent and an Ecosystem Architect at Inrupt [19]. One interesting publication [22] discusses the need for decentralising the web by highlighting ongoing problems such as the echo chambers in modern social media companies each using a centralised approach [23], or the "filter bubble" [24] effect that is happening on modern search engines where personalised user models effectively hide content in an attempt to maximise engagement. These claims are reinforced by letters written by Berners-Lee, who states that we need increased transparency with where our data goes [25] and expresses concerns for the future of the web if we continue on our current path [26].

This publication references another blog post by David Rosenthal [27], an established researcher in data storage, who claims that "it isn't about the technology", and the obstacle in decentralising web is not in the technology to do so, namely Solid pods, but in the reluctance of dominant tech companies that do not want to release the vast amounts of personal data that entirely drives their advertising business models. Another publication by Verborgh [28] drives a similar point; that to elicit widespread Solid pod use, there will need to be more cooperation with technical research from dominant companies. This is very interesting because it seems that the reluctance of the big tech companies is a hindrance to the adoption of Solid and decentralisation as a whole.

However, in another post from Verborgh [108], he hints that this will ultimately be a good thing. If big tech companies can't attract new users by the possession of information about current users, it will spur competition and force them to become innovative with other appealing aspects of a service, such as added functionality, a user-friendly interface or proper data security.

Some other publications in Verborgh's work were very helpful in understanding Linked Data concepts and its role in the next generation of the web [30]. He is truly at the front of advancing the adoption of Linked Data and Solid, for both the developer community [29] and the average user [78], by providing advice for developers and explaining what the transition

to Solid would mean for end users, including granular control of data and true data portability.

7.1.2 Enterprise Decentralisation

In a report on the state of data in 2022 [114] that surveyed 400 companies, published by Enterprise Managements Associates, it was shown that organisations in the EMEA region (Europe, Middle East, Africa) are the leaders in adopting a decentralised architecture. However, other regions have signalled that this is the direction that they would like to move to in the coming year, shown below in Figure 6.

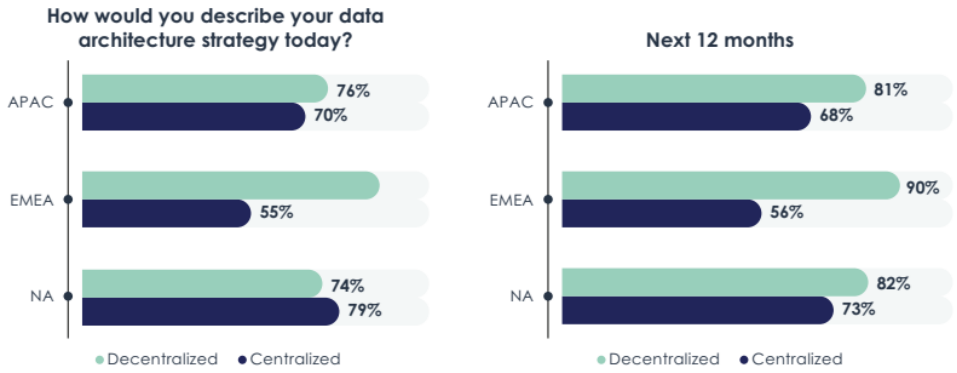


Figure 6 – Decentralisation of organisations per region [114]

This report continued to state that 31% of respondents claimed that data constantly being moved and changed makes finding data difficult. Decentralisation of data on the Semantic Web would definitely address both of the concerns listed above, since data will exist in a single location and will only enriched by the addition of new data.

Furthermore, multi-use data storage techniques such as object storage [115] is the most influential aspect of buying cloud data storage for 30% of respondents. Object storage shares some of the same principles as decentralised storage in Solid, in that data held in object storage is self-contained with a unique identifier accessed by the HTTP protocol. Therefore, it seems from this report that many companies would be willing to embrace decentralised data storage with Solid.

An example of a company accelerating decentralisation, and a good source of articles about decentralisation for organisations, are at the website of Janeiro Digital [70]. This is a company attempting to help businesses or entire systems transition from traditional data storage to decentralised storage. They partner with Solid when it comes to storage but perform all the data collection, integration and presentation using their proprietary XForm software [71]. Their work is particularly relevant to this project as the advantages that decentralising the healthcare industry are listed [79] as:

1. No burdens of data privacy for healthcare providers.
2. Patient ownership and control of their health data.
3. Possibilities for insights across vast amounts of patient data via “decentralised intelligence”.

The last point here is particularly attractive, as it suggests that a shared pool of anonymous patient health data could assist the medical community in accelerating research projects. The applications of shared decentralised data are not limited to the healthcare industry; as one can easily imagine the appeal of a globally shared representation of real-world data to agriculture or engineering. However, this of course relies on equal participation from nations around the world, many of which would not have the technical infrastructure to create Linked Data en masse.

7.2 State of the Art of Solid

As Solid is still evolving and in early adoption, finding related work proved to be very difficult. Searches on any of the major technical paper databases such as IEEE or ACM do not return any publications on Solid and results are adulterated due to overlapping applications of the term ‘solid’. However, various technical specifications of the Solid protocol, a previous dissertation on this topic by Akashdeep Lamba [16], articles written by the Solid community and the projects of other Solid developers have been the most valuable resources for researching the area.

7.2.1 Documentation

The most useful technical resources for the entire Solid specification are found in the Solid Technical Reports repository [58]. This contains individual reports for all aspects of using Solid, from the authorization flow (built on top of Open ID Connect 1.0 specification [59]), to the notifications protocol for messaging [60], or the Application Interoperability Primer [61] which describes some requirements and recommendations for developers of Solid applications. These primers were very useful in defining the terminology referenced in other publications and did allowed for a granular understanding of the background processes that occur while using Solid.

However, almost all of the examples of the specifications in use are written in RDF [62], Turtle [63] or JSON-LD [64], meaning that a prior knowledge of these concepts is required to be able to read and understand the Solid flow. Furthermore, all of the specifications are still in some editable status, meaning that they are works in progress of a W3C working group and are subject to updates by the working group at any stage. Nevertheless, these working groups are quite active, with feedback channels and mostly open meetings for anyone with interest to attend.

All Solid applications adhere to these specifications, regardless of technologies or frameworks used by the application. Therefore, there are a number of available Solid client libraries that can be used by developers to work with Solid, listed in the solidproject.org website [39]. The benefit of these libraries is that they bind application logic to the Solid specifications, so that Solid developers need only ensure that their developed code works for the desired library. At the time of writing, the programming languages with libraries endorsed by solidproject.org are: JavaScript, Perl, Python, Rust, Kotlin and Swift; each with varying amounts of documentation. JavaScript is supported the most, with substantially more available libraries and supporting documentation.

7.2.2 Literature

As mentioned by Lamba, finding literature on Solid is especially difficult due to several implementations of the word 'solid' within scientific literature – it is used as an acronym for object-oriented design principles [17], as a programming language for the Ethereum blockchain [18], and features in the titles of many works across all branches science and engineering. Furthermore, frequently updates to the Solid Protocol likely acts as a deterrent for some researchers who do not want to design a full proof-of-concept application in Solid in its current state, in fear that it may become obsolete in a future version of the specification.

However, the aims of this project are very similar to those of Lamba's work in 2019. His primary goal was to evaluate "the extent to which Solid can be used to build practical applications from the perspective of an application developer", which is almost identical to the overall aim of this project. To reach his goal, Lamba was attempting to develop an online social network, where the primary design considerations were 'feed aggregation' from followed users and 'user discovery' of other users of the application. This project will focus more so on implementing procedures used in professional medical settings in a Solid application to serve both patients and healthcare workers. Furthermore, there were no separate user roles with corresponding use cases in his application, where this is a crucial aspect of this application.

Some of the aspects of Solid that Lamba considered were;

1. The developer experience using Solid.
2. The feasibility of a decentralised social network developed with Solid technologies.
3. Improvements that could be made to documentation and developer resources.

These same aspects will be covered from a different practical application approach, but the key difference is that 3 years have passed between the undertaking of these two projects. This means that comparing the results at the end of this project with Lamba's from 2019 will give an estimation of the extent that Solid has changed in this time.

Lamba took a sound approach to both researching related areas and investigating the feasibility of development in Solid, in that his research was placed in the appropriate domains. However, a difference between the two research domains appears in that Lamba was developing a social network application; so an aspect of his research was placed in "Privacy in Online Social Networks", whereas this project will be considering the security of applications used in the healthcare domain.

Another dissertation written in 2020 by a student at MIT [77] was found that investigated Solid's capabilities for a mobile application that uploads biometrics from smart wearables to Solid Pods, where the biometrics are modelled using the FHIR RDF framework [109]. This publication was a useful resource for the list of references and some other descriptions of concepts related to Solid. However, very little evaluation was given into the state of Solid at the time and the technical contributions at the end of the project were not beneficial in this project due to different technologies in use, meaning that Lamba's project from the previous year is a much more valuable resource.

No other technical literature was found on the state of Solid alone, as most of the blog posts and articles featured and discussed among the community were focused more on decentralisation and Solid's role in a decentralised web. As will be discussed in the next section, some researchers were found during the project that are working on technical papers at the current time of writing, which were unavailable for this project's research.

7.2.3 Community

The most valuable resource for literature related to Solid was through the various community channels. Although this mostly consisted of informal discussions, blog posts or recorded meetings, it was material straight from those most active in Solid and with the most knowledge.

Most of the active developer community involved in Solid seem to be working to make Solid more widely used by businesses at Inrupt [19]. This company was co-founded by Berners-Lee in 2017 and is undoubtedly the largest collection of professionals working to make Solid widely accessible. Other members of the community are testing the capabilities of the technology by developing Solid applications using data stored in Solid Pods [20], or trying to expand the list of technologies and frameworks currently supported by Solid [56, 57].

Developers can engage on Solid topics at the following recommended media:

- Solid Community Forum [15] – This is the home for most blog posts, application concepts, discussions, job opportunities, event listings etc. The forum operates with a 'trust level' system, where users earn their way up to higher trust levels by 1)

engaging with posts on the forum and earning likes on their own posts, which grants them additional privileges to maintain the forum and 2) earning badges, which are small achievements earned for various activities on the forum.

- Gitter channels [32] – This is a more informal platform for engaging with the Solid community, across 30 channels related to Solid. This medium has thousands of users across the various channels and allows for short-form conversation with some of the developers at Inrupt or editors of the Solid specification, including Berners-Lee himself.
- Solid World Events [65] – These are monthly events held by Inrupt since 2020 which highlight some of the key news from the past month, including some Solid developers as guest speakers and a Q&A session. Minutes and recordings are taken for each meeting which allow the community to work through past meetings and view the progress of some work items related to Solid.

It was through engagement on the Community Forum that connections were made with other Solid developers. One of which was a final year student attempting to integrate Community Solid Server (CSS) [66] into CERN infrastructure, including slight modifications to the CSS code and integration to work with CERN's single sign-on. This individual is still working on their thesis at the time of writing, so exact details on the implementation could not be acquired.

Another developer had just released a Solid Pod browser called PodPro [67], using their separately developed Solid client which interacts with Solid Pods using RDF data models written in the Elixir programming language [68], using the RDF.ex library [69]. Documentation of the project is currently underway and will be released in the second half of 2022. Their Solid client is currently private so it could not be used in this project, but PodPro remained a valuable resource throughout. An example of the tool in use is shown below in Figure 7. It has the advantage over the standard solidcommunity.net Pod browser [38] in that it shows a clearer overall structure of Pod data with the access that individuals have, shows the full HTTP response of calls to retrieve or insert data and allows for much faster swapping between Pods, but comes with a requirement of a reasonable understanding of Turtle syntax.

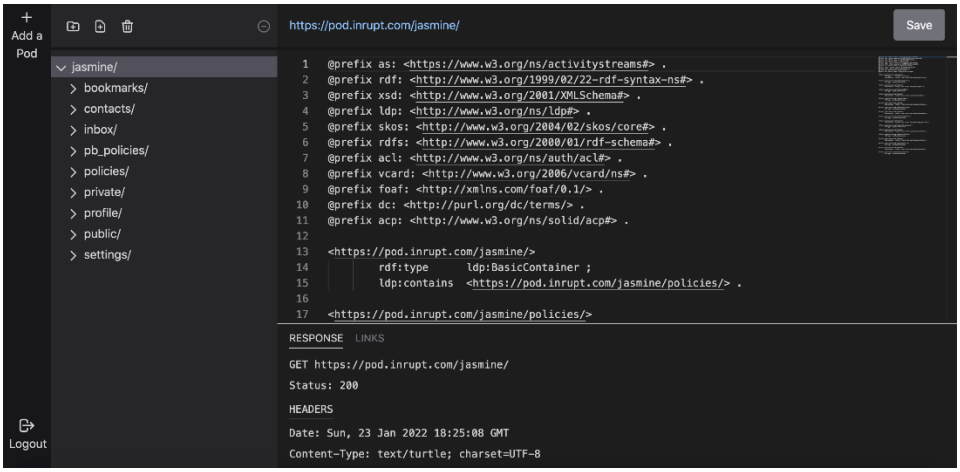


Figure 7 - PodPro tool in use

Through discourse with this developer, another project was discovered which is related to this project. Janeiro Digital [70] is a company founded in 2009 with similar aims to the Solid Project itself, returning control and security of data to the individuals and organisations which own it. They released a software platform in 2017 known as XForm [71] which acts as a service for companies to store centralised data in Solid Pods, which performs all the necessary data collection, integration and transformation.

An interesting application of XForm arose in 2021 when Janeiro Digital worked with the NHS [72] to restore ownership of patient data to the patient themselves, by retrieving the data from the many healthcare systems in which it already existed and storing it in individual Solid Pods. They faced two main challenges;

1. Integrate the data from a wide spectrum of existing systems.
2. Ensure the managing and sharing of patient data is intuitive to the average user.

While the first challenge is not as important for this project, which is to develop a proof-of-concept application using empty Pods, the second challenge that Janeiro faced in their project is one that was surely at the forefront of this project. As discussed in a presentation of the project [73] made by the CTO of Janeiro, Justin Bingham, it was important not to complicate users with too much information about the physical structure of their data or the permissions related to a piece or section of data.

Where Janeiro's implementation excelled far beyond this project was in their approach to a hierarchical ontology of healthcare concepts, known as Shape Trees, which helped with intuitive data authorisation to other individuals or applications. They also enabled integration with wearable fitness trackers to store activity data in the patient's Solid Pod, which was outside of the scope of this project. However, their application did not attempt to enable the

creation of new healthcare records to be stored in Solid Pods, which is something that the application for this project will attempt to achieve.

Another application named “Solid Health” was discovered during the research into this area, featuring a mobile application for storing and accessing health data in Solid Pods [74]. The application was built using React Native [75] and only supports Android devices, with data collected from fitness trackers using Google Fit APIs [76]. There was some initial concern that the overlap between the two applications would be large but fortunately the similarities ended at the name of the applications, as it became clear that the technologies and application scopes were quite different.

7.3 Security in Healthcare IT Systems

Lastly, some research has been placed in the area that the proposed application will be designed for, the healthcare sector. This included an examination of the security threats that the sector faces, to determine if decentralising the sector could be beneficial.

Logically, no resources were found describing the exact system architecture or application stack used by healthcare in Ireland, as it would give unnecessary information to attackers, which meant that generalised use cases of IT systems in hospitals were examined instead. Some examples of decentralised healthcare applications are then evaluated to determine if they present some shortcomings that this project could address.

7.3.1 Current Landscape

Resources have been found describing the landscape of cyber security in Europe [33] which reveal that the healthcare sector has been particularly targeted recently due to the collection of scientific information related to the COVID-19 vaccine. This report also states that attackers have taken advantage of the pandemic to impersonate medical bodies and carry out an increased number of phishing [80] attacks by creating new lures tailored for individuals working from home or related to vaccinations. Phishing was listed as the method for gaining access to the HSE’s system in [4], with attackers using the Conti ransomware [81] to disrupt efforts of recovery made by the HSE during the attack. This lines up with the landscape of attacks across Europe, as ransomware incidents using Conti were most popular as of 2021, as shown below in Figure 8.

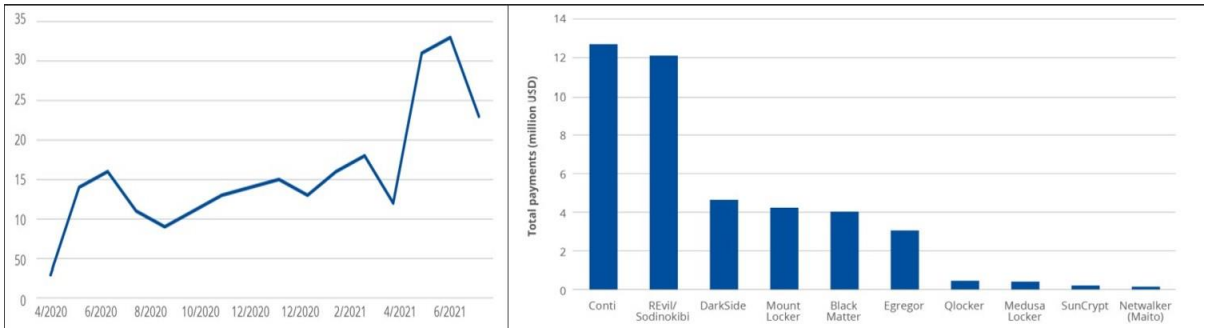


Figure 8 - Number & value of ransomware attacks observed by ENISA (April 2020 - July 2021) [33]

Some case studies of security breaches in healthcare systems were examined to determine the extent to which hospitals and other institutions must allocate resources for security and compliance around ownership of patient health data. These papers concluded that huge efforts needs to be placed in providing adequate training and awareness [82, 84] across the organisation and an ever-expanding array of technologies make it difficult for management to ensure that the countermeasures that they have in place are adequate [83]. A common theme across these case studies is that adequate data security is a huge IT overhead for hospitals which is sometimes overlooked in favour of allocating resources towards healthcare assets or personnel.

Although it would take a great deal of effort to address data protection regulation following the proposed move of patient health data to decentralised data storage, it would mean that healthcare institutions may not remain as highly valued for ransomware attacks and could be better equipped to effectively respond to attacks as they occur.

7.3.2 Security of Decentralised Healthcare Applications

It was difficult to find any evidence of applications using Solid pods in use in the healthcare sector but two examples were examined of decentralised healthcare applications using blockchain technologies. A comprehensive list of similar applications was found in a paper that discusses the advantages of integrating blockchain into modern healthcare [113], proving there is no shortage of projects in the area. These projects express the same need for moving to decentralised systems: increased threats to centralised health records databases and the desire for increased portability of patient data. However, their implementation differs from this project in that they have employed blockchain technologies to create a decentralised system while this project intends to use a network of interconnected Solid Pods in the Semantic Web.

Medicalchain [111] is a blockchain based application that has gained a lot of interest since its creation in 2016. As specified in the product's white paper document [112], access to health records in Medicalchain is controlled using a Hyperledger Fabric and records are

encrypted using symmetric key cryptography. However, in the instance where the access of an individual is to be revoked, the record must be re-encrypted with a new key. In his thesis, Paulos [77] states that this means the record might always be available to the individual with revoked access, which is a huge security risk. This is not a concern for an application which uses Solid as opposed to a blockchain network, as access is tightly controlled in real-time with the Solid protocol.

MedRec [110] is another blockchain application built on Ethereum, where a network is established comprised of patients and medical providers, elected by the network of patients. This creates a system where patients can request access to their medical records, which are still ultimately stored in individual medical institutions but are accessible from anywhere in the network. The problem with using the Ethereum approach is that the metadata of transactions between patients and institutions are public by nature, meaning anyone who could link a patient's Ethereum address to their real-world identity could determine the healthcare providers that a patient has.

They took steps to address this problem by anonymising the metadata of transactions, via creating intermediate accounts between patients and institutions; however, they acknowledged that the correlation between associated accounts in the network could be tracked over time and thus the system is not entirely secure. Being able to associate users to the institutions from which they receive care would make the application unusable in the eyes of both patients and healthcare workers.

Chapter Three

8. Design

Section Overview

This chapter describes how the application was designed. This design fell into two phases, described in section 8.1 and 8.2, respectively:

8.1. Exploration

8.1.1) Design purpose of application based on research of what Solid could offer to the healthcare sector.

8.1.2) A more thorough inspection of technologies supported by Solid.

8.1.3) Development of basic read/write operations and data access management.

8.2. Application Planning

8.2.1) Design of use cases for application, including definition of user types and their roles within the application. User and Functional Requirement specification. Project Gantt chart/schedule.

8.2.2) Application data specification, including structure and location.

8.2.3) Interface design and approach to implementation.

The first phase was started in October of 2021 and helped with generating reasoning for design decisions, along with insights gained through research in related areas. This phase was necessary for deciding on the application framework and predicting what is possible to be developed within the timeframe. The second phase was started in January of 2022 and lasted for a week or so; to gather conclusions from the first phase and use them to develop a full application specification that the application will be validated against.

8.1 Phase 1 – Exploration

8.1.1 Application Purpose

The first design decision to make was the purpose that a Solid application could serve the healthcare sector. The original concept that was appealing was an application to use in the event of a medical emergency; where information about particular medications needs to be made available to paramedic, such as blood thinners. Traditionally this has been in the form of an information card that the patient needs to carry at all times, but this is susceptible to misplacement.

Therefore, a concept application was designed where patients could grant paramedics access to view a list of current medications or contact the caregiver facility of patients to request information about the patient, while concurrent research was placed into context-aware computing and user modelling. Research revealed the context of an entity (individual or application) is defined as the combination of location, identity, activity and time [92], and developing an application to collect and process this data from the user was deemed outside of the scope of investigating Solid's maturity. Furthermore, the nature of Solid requires access to be explicitly granted before users can view data in another Pod. Where the patient may be unresponsive or the delay between communications to caregiver facilities may be too large, this concept was deemed unsuitable for Solid in its current state.

Therefore, an application to view and modify healthcare records was considered instead, where data could be stored in Solid pods. This would ideally help with the growing problem mentioned in section 7.3 above, where the healthcare sector has been an attractive target for ransomware attacks due to large volumes of valuable data stored across fragmented and sometimes outdated systems. A lot of factors had to be taken into account for an application of this type, such as the location and shape of the data, the processes to create and manage data, the types of users that would be interacting with data and the interface of the application.

8.1.2 Application Framework

The second design decision to make was the framework to use for the application. To help with making this decision, applications listed on the Solid Project website [40] and on the community forum [85] were examined first. While a lot of these did not have available source code or documentation, it was helpful to see the technologies that developers had already used in developing a working Solid application. Next, the list of libraries readily available for use in Solid development was considered [39] along with some application concepts from the community [20], which revealed that JavaScript libraries were dominant in the space for development, but a lot of application concepts remained with little or no engagement from the community over a number of years.

Then the developer tutorial listed on the Solid Project website [43] was followed, which uses the Inrupt solid-client JavaScript package [44]. This required a Solid Pod and corresponding WebID, which is a unique URI identifier for each individual with a Pod, so these were acquired from solidcommunity.net [38], as the Pods are listed as being hosted in the closest location to the application developer. While this developer tutorial was basic, it did provide some code for authentication that could be used later. At the GitHub repository for the package, a link was available to a more in-depth tutorial at the Inrupt website [86], which has

some additional functionality and uses webpack [87] to run the application as opposed Parcel [88], which is used in the Solid Project tutorial.

Therefore, due to the amount of documentation available with the Inrupt JavaScript library, and concerns about being unable to find any support when reaching a development issue using a less-used technology; it was decided to develop a JavaScript web application, using some third-party npm packages available in the Node.js environment.

8.1.3 Development of Basic Operations

An important aspect of this initial exploration with the chosen technology was to learn how it could be used for granting and revoking access to Resources in Solid. The Inrupt Solid library does this with an 'access' object where the individual levels of access are specified with a Boolean flag. The access modes are described below as per the documentation [97]:

- **read:** The ability to view the contents of a Resource.
- **append:** The ability to add new data to a Resource.
- **write:** The ability to add new data to a Resource, and to change or remove existing data.
- **controlRead:** The ability to view the access to a Resource.
- **controlWrite:** The ability to change the access to a Resource.

There are two access control mechanisms for authorising individuals to Resources in Solid, Web Access Control [98] (WAC) and Access Control Policy [99] (ACP), where the main difference is that WAC uses clauses within the Access Control List (ACL), which is an Auxiliary Resource to a Resource which requires an authorisation control system. ACP uses a different type of Auxiliary Resource, the Access Control Resource (ACR), to enforce ACP Access Rules and ACP Access Modes.

Solidcommunity.net only supports WAC at the time of writing, which means that WAC using ACLs was the only way to specify the access that individuals may have to a particular Resource. Furthermore, for WAC 'controlRead' and 'controlWrite' must be in sync, meaning that a user cannot have the ability to view who has access to a Resource without being able to change the access to that Resource.

Solidcommunity.net provides a few definitions for combinations of access modes, that will be referred to throughout the rest of the paper. These are listed below in Table 1.

	read	append	write	control
Viewers	✓			
Submitters		✓		
Posters	✓	✓		
Editors	✓	✓	✓	
Owners	✓	✓	✓	✓

Table 1 - Solidcommunity.net authorisation levels

Expanding on the code for the tutorial applications using further documentation available from Inrupt [89, 90], a number of basic operations were developed that were deemed crucial in order to build an application specification that could be fulfilled with the library. These operations are listed in Table 2 below. Definitions of these terms can be found at [91].

Operation ID	Operation Title
O1	Authentication with Pod provider.
O2	Create a new SolidDataset in a Solid pod at a specified URI.
O3	Create an ACL Resource for a SolidDataset with configurable access levels.
O4	Read the access that a user has to a SolidDataset.
O5	Read the contents of a SolidDataset at a specified URI.
O6	Grant configurable access to a specified user for a SolidDataset.
O7	Read back a Thing within a SolidDataset using its URI.
O8	Create and upload a new Thing to a SolidDataset.
O9	Delete a Thing from a SolidDataset using its URI.
O10	Delete a SolidDataset at a specified URI and all contents.

Table 2 - Developed basic operations

With these operations capable using the selected Solid library and the Node.js development framework, it was certainly feasible to develop an application to read and write files to an individual's Pod. Some consideration had to then be placed in ensuring that the application an enterprise-level application concept could be developed that would help the healthcare sector and maximising the usability for both patients and medical professionals.

8.2 Phase 2 – Application Planning

8.2.1 Functionality

To begin the design process for the use cases that the developed application should be able to offer to users, a number of roles had to be created for the application. These are the user types that would be using the application and therefore the use cases were designed around

offering maximum usability to this group. These user types are shown below in Table 3, alongside a description of the intended functionality for each user type.

Use Type	Description
Patient	Receives medical care from a Doctor.
Institution Administrator	Uploads appointment details of upcoming appointments between Patients and Doctors.
Doctor	Conducts a medical appointment with a Patient.
Pharmacist	Works at a pharmacy to dispense medication to a Patient.
Emergency Worker	Ambulance worker, Paramedic or other who needs knowledge of any current medication that a Patient may be taking.
Health Worker	Either a Hospital Administrator, Doctor, Pharmacist, or Emergency Worker.
Insurer	Sells life insurance to a Patient and therefore needs knowledge of ongoing health issues.

Table 3 - List of user types for application

With these user types in consideration, a list of use cases involving each was drafted. One of the most important aspects to developing these use cases was finding a balance between usability for each user type and practicality to match with what happens in professional healthcare. Therefore it was important to not have a notably larger workload on a particular user to complete a given process, as the application would become unappealing for users in that role. At the same time, it was important to consider the procedures used in professional practice and minimise the disruptions that moving to this system would incur. Finding the balance between these two design considerations proved to be a difficult task throughout the design and implementation of the application. The list of initial use cases is shown below in Table 4.

Use Case ID	Description
UC1	Patient can register care under a new medical institution, allowing the Institution Administrator of the institution to operate on their behalf to log new appointments.
UC2	Patient or authorised Health Worker can access a Patient's medical records.
UC3	Pharmacist can view a list of prescriptions belonging to a Patient if they have been granted explicit access.
UC4	Emergency Worker can view the current medication that a Patient is taking by default.
UC5	Insurer can view a list of diagnoses for pertinent healthcare departments, dated from the past 5 years, if they have been granted explicit access.
UC6	Doctor can upload a new healthcare record to a Patient's pod, if they have been granted access to interact with the Patient's health data for a given healthcare department.
UC7	Patient can view who has permission to any piece of their health data and add or revoke permission from any user.

Table 4 - Draft of application use cases

Following the development of these use cases, a set of initial User Requirements and downstream Functional Requirements were drafted to break up development tasks and try to identify important tasks. The aim was that if each Functional Requirement was met then each of the use cases would be possible, with some additional functionality. These requirements are listed in Appendix 1. There was also a Gantt chart created to identify task dependencies and provide a rough schedule of the development work, which is provided in Appendix 2.

Note that in Table 4 above, there are three terms which have broad definitions in the scope of this application. The first is 'medical institution', which is any location where an individual may receive medical care, for example a hospital. The second is 'healthcare record', which includes four possible types:

1. **Appointment** – Details an appointment that a Patient has in a specified healthcare department (e.g. Cardiology, Ophthalmology), with a specified Doctor in a given medical institution.

2. **Diagnosis** – Information about a diagnosis that may have resulted from an appointment between a Patient and a Doctor.
3. **Prescription** – Details about a prescription for a Patient with an option to be shared with a specified Pharmacist.
4. **General Record** – Any type of record that falls outside of the three listed above, for example a blood test result or a record of drug administration.

The third is 'Doctor', which in this application is any individual that would be creating medical records on behalf of a patient. In reality, this could be a nurse, physiotherapist, radiologist or any other medical professional that would create healthcare records for a patient.

A UML Use Case Diagram [125] is shown below in Figure 9, to show the various user types and the actions they can perform within the application.

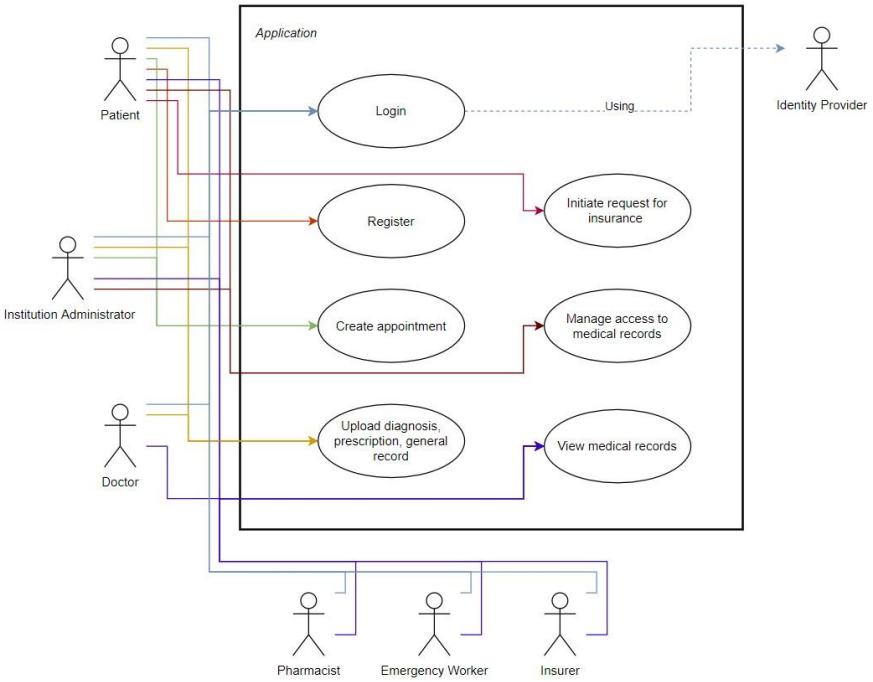


Figure 9 - Use case diagram

These use cases were sent to a group of individuals working professionally in healthcare informatics, who were able to provide some useful feedback that went towards refining the approach to the application design. Although this feedback didn't change any of the use cases listed above, it did influence the implementation in many ways. Some of the comments that went back towards design refinements were:

- There needed to be greater transparency of the role of different user types within different use cases while the application is being used.
- There should be an option for registering different types of medical institutions.

- The access that all individuals have to any piece of health data should be transparent while using the application.

From this feedback, an idea was conceived to allow users to have multiple different types of health data in operation at one time. This is to reflect the fact that Patients may receive care from a public hospital and a general practitioner simultaneously, and while records may be shared across these two medical institutions, they are still independent sets of data for use in their own domain. This decision was also based on the fact that there are different data sharing policies or integrated care duration within institutions of different types. Although patients could receive care across a large range of healthcare facilities, the possible options in the application were limited to public, private or general practitioner.

8.2.2 Application Data Structure

An important distinction to make is the difference between the WebID of Solid Pod users and the web address of their Pod. For example, if a particular user signed up for a Pod with solidcommunity.net under the username 'UserName1', then the two important addresses for use with Solid development would be:

1. **Pod 'base' URI – "https://username1.solidcommunity.net/":** This is the address of the root directory of the Pod.
2. **Pod owner WebID - "https://username1.solidcommunity.net/profile/card#me":** This is the unique identifier for the Pod owner, used for authentication and authorisation.

The next design decision to make before development of application features could begin was the location and structure of application data. This was an important aspect of the application's development, as one of the most common roadblocks for developers in Solid is 'where to store data' and 'how to access it', since the specification relies so heavily on the exact URI's of Resources. Three key design considerations impacted this process:

1. Adherence to the Solid ethos with regard to personal data ownership and control.
2. Isolation from all other data stored in Pods.
3. Usability for both patient and health worker by modelling existing file systems in healthcare.

The first design consideration was the reason for storing patient health data in their respective Pods, as opposed to the Pods of healthcare workers or of some medical institution. This would make patients become the owner of their own data and allow them to control the access to it as they desired. This decision was also impacted by a desired purpose that the application could serve to the healthcare sector; a reduction of health data

stored in one centralised location, that could make the sector less of an appealing target to attackers. Therefore, patients can be considered synonymous with Pod owners in the context of this application.

The second design consideration was the reason for storing application data within a Solid Container at the root of the patient's Pod, separate from all other data that they may have in their Pod. This is the most suitable type of Resource for this application's data, as Containers are the only Resource that can contain further Containers or other Resources. Following from the decision to allow multiple types of health data to exist concurrently; each respective health data type would have its own Container stored at the root and be independent from each other.

The third design consideration was the reason for creating a structure to the data where records from one healthcare department could exist without interfering records in another department, each with their own access policies. Records could then be shared across departments as required. Within each department that a patient may receive within a given type of health data, there needs to be space for each of the permitted healthcare record types that the application enables the creation of; Appointments, Diagnoses, Prescriptions and General Records.

With these decisions regarding application data in place, a diagram of the shape of application data with regard to other Pod data is shown below in Figure 10.

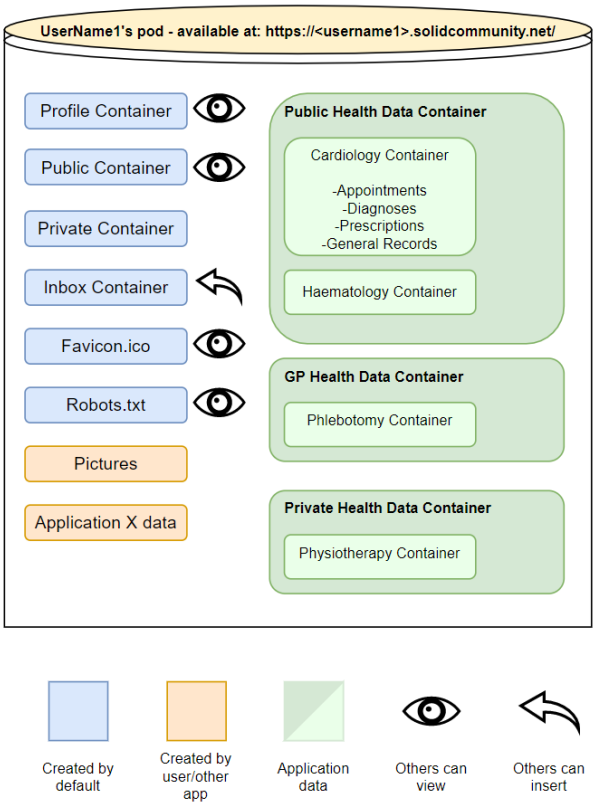


Figure 10 - Application data inside Pod

Here the Resources that are created by default along with a Pod are shown in blue. They include the Profile and Public Containers, for information that the user would like to be publicly available to the web, along with the Favicon.ico and Robots.txt files. The Private Container is where the user can store sensitive data, but storing application data here might put other files at risk. In this instance for illustration, the user has uploaded some personal pictures to a separate 'Pictures' Container and has some data from some other application stored in 'Application X data'.

All of the data created by this application is shown in green, and is entirely separate from other data in the pod. The access to all of this data is private by default but the implementation of features and execution of use cases will cause access to be granted appropriately to other users.

It can be seen in this instance that the patient receives some care from a public institution in the Cardiology and Haematology departments, they have some records with their GP possibly for blood tests, and they receive some physiotherapy from a private institution. Note that the structure for all departments across all health data types is identical with the same 4 contained record types as with Cardiology in the public container.

8.2.3 Interface Design

With the desired functionality in the form of use cases and requirements, an idea of the application data shape and different user types with tasks for a given application process; the next big design decision was the appearance of the application and the steps required to complete application processes.

The initial design featured a separate interface for each user type, with each interface being a single-page application [93] where possible operations within the app were clear and matched with the use cases for the user that was signed in. However, it became clear early in development that there is currently no way to verify the role that a user has within a given institution.

Users all have a WebID provided by solidcommunity.net when their Pod is created which only includes their username, and Pod contents by default are identical. After authentication with the Pod provider, the application is sent a JSON Web Token [94] (JWT) enriched with a claim for their WebID which matches with the user account that was created with the Pod provider. The flow of the authentication process is shown below in Figure 11.

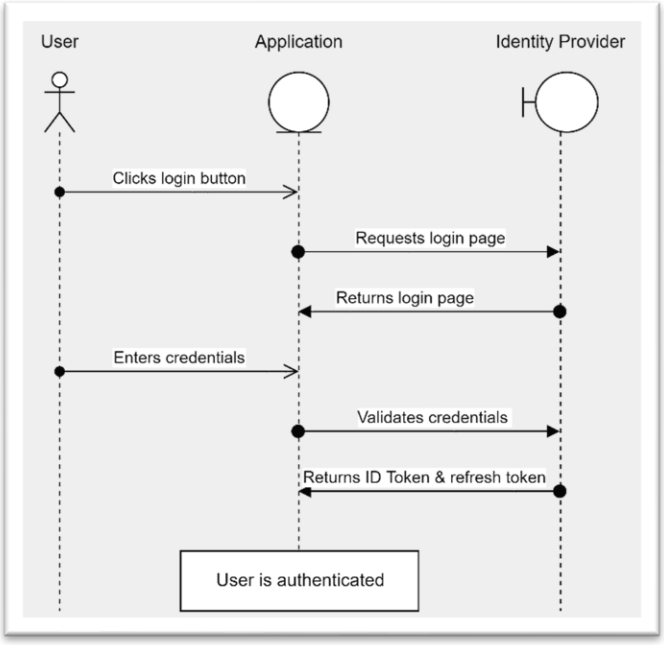


Figure 11 – Application authentication flow

There was no readily available way to add additional claims to the JWT to describe the role that a given user has within an organisation and while it could have been possible to store a value at some location within each user's Pod which contained their role, this value could have been changed by the Pod owner at any stage. This means that any user could potentially have any role within the application. Therefore, it was decided to keep the

interface identical, regardless of the user that was signed in. Certain operations would be visibly unavailable if the current user is neither the Pod owner nor the Institution Administrator.

The only way to distinguish the role that a user has for a given Patient scenario is the access that has been granted to them by the Patient via previously completed use cases. This brings additional functionality to the application in that any user could be viewed as both a Healthcare Worker for a Patient, while being a Patient themselves to some other user.

With this uniform display approach, it was necessary to enable an option for both:

- Patients to access the data in their Pod.
- Healthcare workers to access a Patient's Pod and make changes to data.

This was for the application to know the location of where to look for data, either in the Pod of the authenticated user or in the Pod of a specified user. Therefore, before any application actions could be performed, it was necessary to obtain this intention from the user. Once this decision has been made within the application session and the user has selected the type of health data to access, the user will be brought to some 'home page' where they can begin to perform operations within the app.

Regarding the design of other application screens, it was decided that they should be easily found using buttons on the home page. This will update the HTML DOM accordingly, using HTML element manipulation techniques gathered from an online course by Ray Villalobos [116]. A pleasant user experience with the app would be desired but pushing the limits of the technical capabilities of Solid development is the goal, so priority will be placed in this area instead.

9. Implementation

Section Overview

This chapter describes how the features of the application were implemented. The chapter is broken up into six sections:

9.1. Setup – Describes the process of setting up the full development environment that was used throughout.

9.2. Login and Health Data Configuration – Describes the implementation of features for registering with the application and configuring the data used to register.

9.3. Uploading Records – Describes the implementation of features that upload healthcare records to a Patient's pod.

9.4. Accessing Records – Describes the implementation of features that access healthcare records stored in a Patient's pod.

9.5. Managing Record Access – Describes the implementation of features that change the access to healthcare records stored in a Patient's pod.

9.6. Deployment – Describes the process for deploying the application on a live environment.

Sections 9.2 – 9.5 describe the performance of the application to complete each action possible within the application, including breakdown of application processes and justifications for decisions, where appropriate. UML Activity Diagrams [117] are provided for explanatory purposes of each core piece of functionality.

9.1 Setup

The first step to set up the development environment was creating a Git repository for the application [124]. Next, a diary of notes during development was created, to share some insights on Solid development with future developers. This is provided in Appendix 3. Frequent updates to the Git repository and development diary were essential to share insights into the entire development process and not just the final point reached in the implementation.

The Windows 10 operating system was used, with code developed in Visual Studio Code and run on a Chrome browser.

During development, the project Gantt chart was consulted to determine the pace of work that had to be completed along with gauging an estimation of the time that can be allocated for each development task.

9.2 Login and Health Data Configuration

The first feature to implement as part of the login process was to gather whether the user would like to access their own Pod or the Pod of some other user. Following this decision, a check has to be made to verify if the Pod to be accessed has already created some form of health data. It was important that only the Pod owner be able to initialise health data in their Pod, as the access for Resources should be set on creation. Therefore, if the Pod owner chooses to access their own Pod and no data exists, they will be shown a form to register health data with a medical institution. The flow of the login process is shown below in Figure 12.

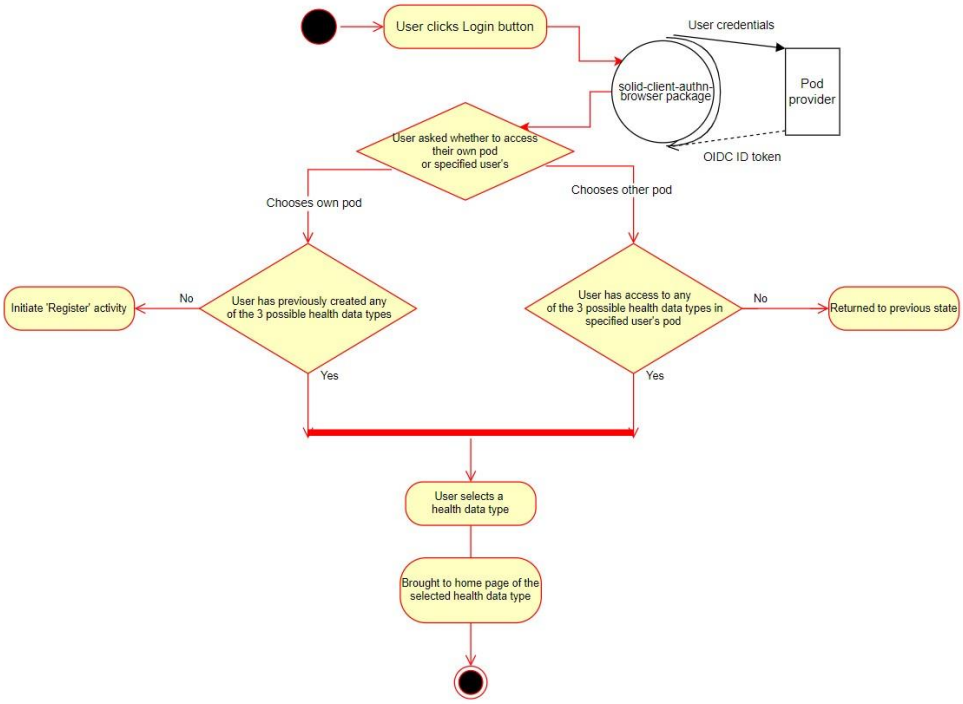


Figure 12 - Login process flow

Logging out of the application is currently unsupported by the Inrupt package, and is a known problem that the development team is aware of [118].

The next feature to implement after login and selecting the Pod / type of health data they would like to access, was to allow users to register care under a medical institution. Basic metadata about the institution is gathered from the user via a standard form HTML element, including:

- Type of institution – public, private or general practitioner
- Name
- Address
- WebID of the Institution Administrator

This information about the institution is stored as a Thing within a SolidDataset called ‘Info’, created inside the accessed health data container.

After these values are inputted by the user, appropriate properties had to be added before this data could be stored as a Thing in the Info SolidDataset, because data is ultimately stored in RDF format. The main vocabulary for finding properties and classes was from Schema.org [95] due to the large amount of terms across an array of concepts, which would be needed for data used in this application.

A healthcare specific vocabulary would be ideal to give more accurate properties that are consistent with the terminology used in practice. While it was clear that a lot of work had been put towards these ontologies, a suitable one with a comprehensive list of medical terms [96] was not found until late in the project’s development. Quality RDF representation of data is outside of the scope of this project, so the decision was made to focus instead on developing application features that would test the capabilities of Solid.

The Turtle representation of the Thing to represent the institution metadata is shown below in Figure 13, where the name has been set as ‘medicalInstitutionDetails’.

```
@prefix schema: <http://schema.org/> .
@prefix ns0: <https://schema.org/> .

<https://patientone.solidcommunity.net/privateHealthData/Info#medicalInstitutionDetails>
  schema:name "Mater Misericordiae University Hospital" ;
  schema:address "Eccles St, Dublin 7" ;
  ns0:dateCreated "Wed, 07 Jan 2022 16:15:33 GMT" ;
  a ns0:MedicalOrganization ;
  ns0:member <https://administratorone.solidcommunity.net/profile/card#me> .
```

Figure 13 - Institution metadata example in Turtle

An important aspect of the register process is the specification of the Institution Administrator, found at the ‘member’ property in the diagram above. This is the person that is able to upload new appointments on behalf of the patient, which reduces the workload that the patient would have using the application and mirrors real-world practice. If the appointment is in a department where the patient has no existing records, this will create a new Container within the health data Container; meaning that the administrator needs to have access to both view and create new records in the health data Container. They also need the ability to grant access to others in the created department Container, meaning that they need owner access in the health data container. Furthermore, they need the ability to check if an individual has access to the ‘Info’ SolidDataset; so that they could view the home page of the Patient’s Pod, meaning that they also need owner access to this dataset.

The flow of the register process is shown below in Figure 14. This diagram continues from the login process after it is discovered that the user has no existing health data in their Pod.

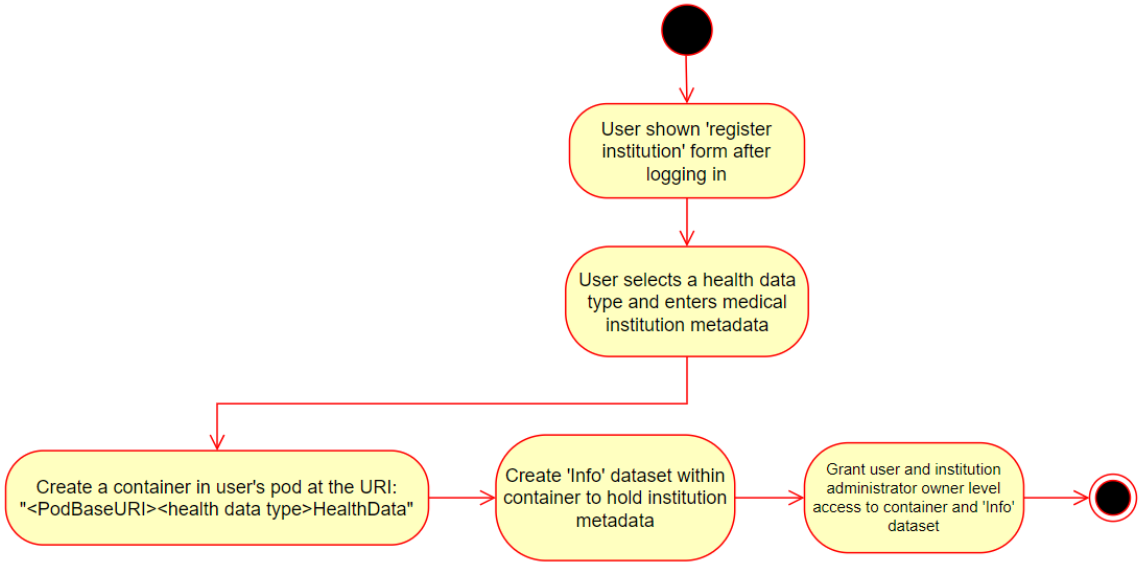


Figure 14 - Register process flow

After the register process has been completed, the user is brought to the home page of the newly created health data, where they can begin to complete other use cases. The information entered in the registration process is shown alongside buttons to initiate other processes.

To test the ability to update individual values within a Thing, edit buttons were added to the medical institution details fields, which update the fields when the user finishes editing and presses the enter key. The name and address of the institution can be edited by the Pod owner or the Institution Administrator, while the administrator field can only be edited by the Pod owner as this will remove all owner access from the existing administrator and grant it to the newly appointed one. The interface for this home page is shown below in Figure 15.

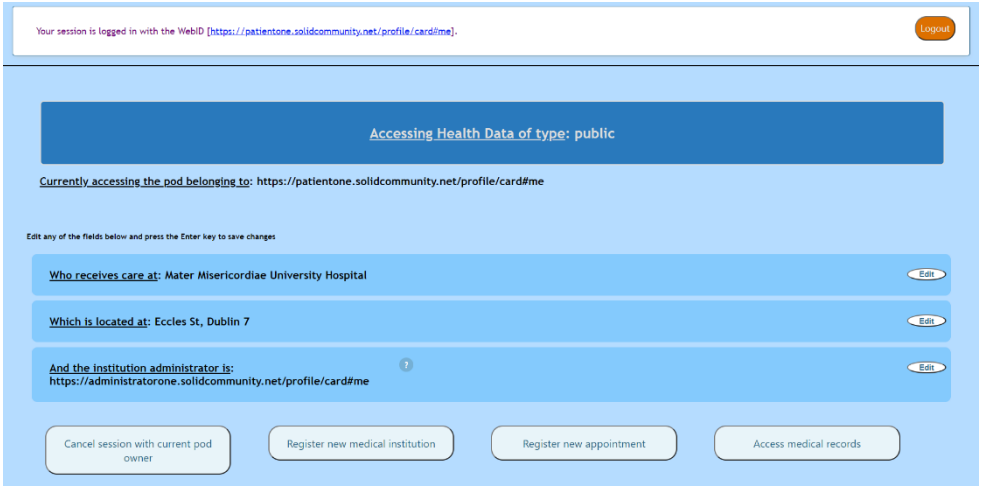


Figure 15 - Application home page

The final configuration feature to be implemented from this screen is one which initialises a new health data Container; started by clicking 'Register new medical institution'. This essentially repeats the register activity shown in Figure 14, except that it deletes all health data of the selected type if one already exists, i.e., if a Pod owner already has records with a public medical institution and wants to create records with a different public institution, it will delete all records with the existing institution and create a new Container. As this feature could potentially delete a large amount of data, it is restricted to only the Pod owner and a warning message is shown to the user to explain the significance of the action.

9.3 Uploading Records

From the home page, users can upload healthcare records in two separate ways:

1. Uploading a new appointment by clicking 'Register new appointment'.
2. Uploading a diagnosis, prescription, or general record by clicking 'Upload medical records'.

The first upload feature to be implemented is to register new appointments. This feature could potentially require access to be granted to a number of users and the creation of several Resources within the accessed health data Container, so it is restricted to the Patient or Institution Administrator alone. As part of this process, the individual uploading the appointment must specify the WebID of the healthcare worker that will conduct the appointment with the Patient, which will grant that person editor access for that department in the Patient's pod.

The flow of the process is shown below in Figure 16.

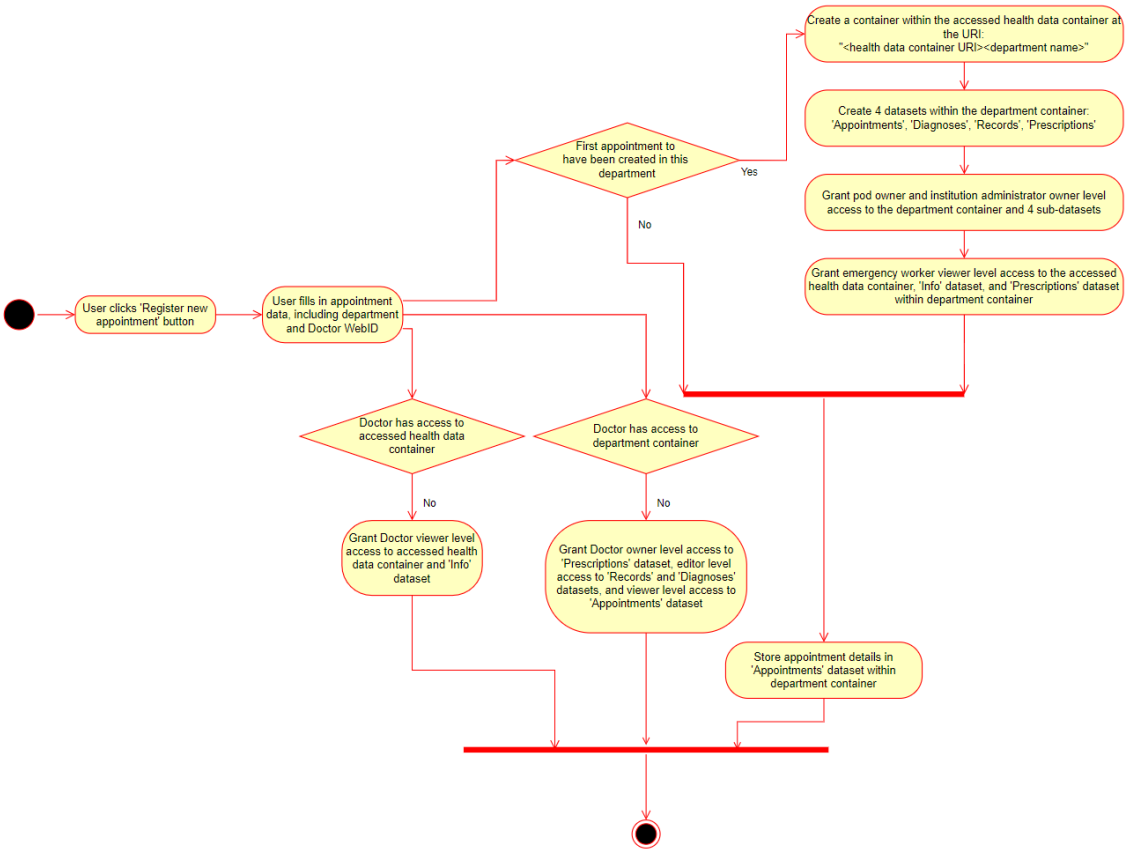


Figure 16 - Upload appointment process flow

As can be seen, this feature performs a lot of access checks and creates a lot of new Resources if the appointment is the first in the selected department:

- A new Container for the department must be created within the current health data Container, and 4 SolidDatasets must be created within this Container for each of the possible record types.
- Owner access to the department Container is granted to both the Patient and the Institution Administrator.
- Viewer access for the Emergency Worker user is also granted, meaning that they will always have the access to view prescriptions for a Patient across all departments, unless the Patient explicitly revokes their access.

Checks are also made on the access of the Doctor conducting the appointment, to ensure that they have the correct access to upload records to the Patient's pod following the appointment. It was important to only grant access that was necessary, which is why they are only given viewer access to the department appointments. They are granted editor access to the Records and Diagnoses SolidDatasets because they should only be able to read and upload/update files. When uploading to the Prescriptions SolidDataset there is an

option for provisioning access to a specified Pharmacist to view the prescription, which means that they must previously have been granted owner level access to this SolidDataset.

As this feature could create a new Container with its own URI that must be found by the application later; it was important to let users choose with a dropdown as opposed to an open text box, to avoid similar Containers created with small typos. A modified dropdown component from W3Schools [100] was used for this and loaded with a list of healthcare departments from Beaumont Hospital [101].

The second upload feature to be implemented is to upload other record types to one of the SolidDatasets in a department Container. The user selects the type of record type to upload, and a form with the relevant properties for that record type is shown. One of the fields common across all record types is a dropdown that retrieves the list of departments Containers in the Patient's pod that the user has some type of access to. The flow of the process for uploading Diagnoses, Prescriptions or General Records to a department Container is shown below in Figure 17.

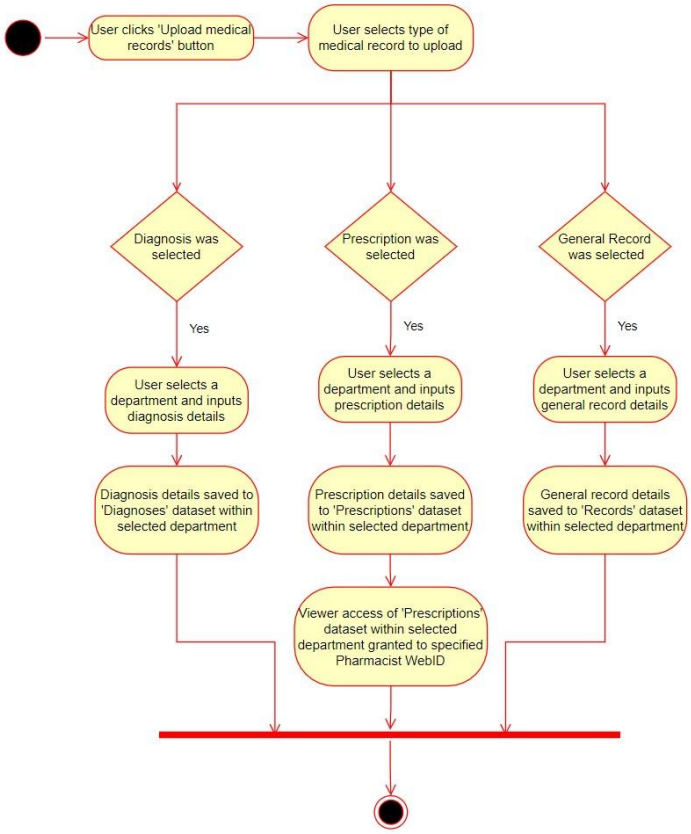


Figure 17 - Upload diagnosis, prescription or general record process flow

Although this process could be performed by the Patient or the Institution Administrator, it is intended to be performed by the Doctor or other Health Worker that conducted the appointment with the Patient. However, it was necessary for the Patient or Institution

Administrator to have the required access for this process so that they may upload Appointments at a later stage.

9.4 Accessing Records

From the home page, users can access healthcare records in two separate ways:

1. Accessing the individual records within any dataset by clicking ‘Access medical records’ and then ‘View records in selected dataset’.
2. Viewing an overview of the entire record landscape by clicking ‘View pod diagram’,

For accessing records within a Patient’s pod, it was important to develop a system that could process any number of Things within a SolidDataset, each with their own different number of properties of varying data types. As Things are returned to the application as JavaScript objects, this was done by a key-value approach where both Things and properties are iterated through, processed and transformed for UI, then added to the DOM.

The function used above to retrieve the list of departments was also required here, so it was modified slightly to work for both features. The URI of each individual file is also displayed to the user if they wish to reference it later or view it in another Pod browser. The process flow for displaying all record types is shown below in Figure 18.

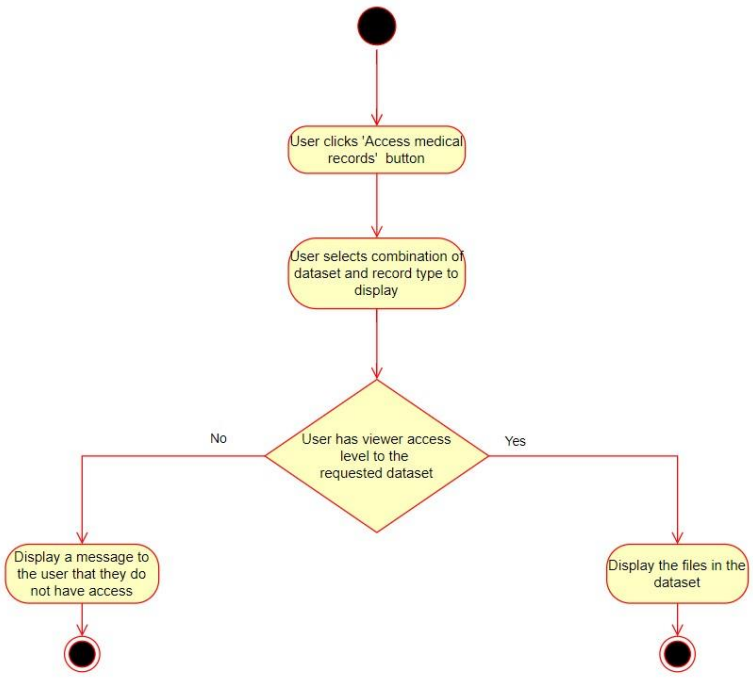


Figure 18 - Access medical records process flow

Midway through development of the application, it became clear that a lot of the concepts and actions done by the user in the application could be very confusing to the average

person, especially one with no technical knowledge or knowledge about Solid. This was a shared concern of Janeiro Digital as mentioned in their NHS project presentation [73]; transparency of data structure or location should be important for users of Solid applications. Therefore, an additional feature was added to the project that could act as a help page for such users. This was called ‘Pod diagram’ and it was intended for 3 purposes:

1. Help users with their understanding of the landscape of their healthcare records, i.e., the total number of files, departments with high/low number of records.
2. Give users some background information on Solid and Solid Resources.
3. Explain to users exactly what data the application stores in their Pods.

Fortunately, this feature could use a lot of code that had previously been written, for example retrieving the number of departments and the Things within a SolidDataset. Therefore, it only took a small amount of application logic to systematically iterate through URIs and retrieve the count of objects. It was also at this time that a tooltip was added to the application, found at Tutorialzine [102] that could display a help message when the cursor is hovering over it. An example of the Pod diagram is shown below in Figure 19, where the cursor is hovering over the explanation for a Solid Container.

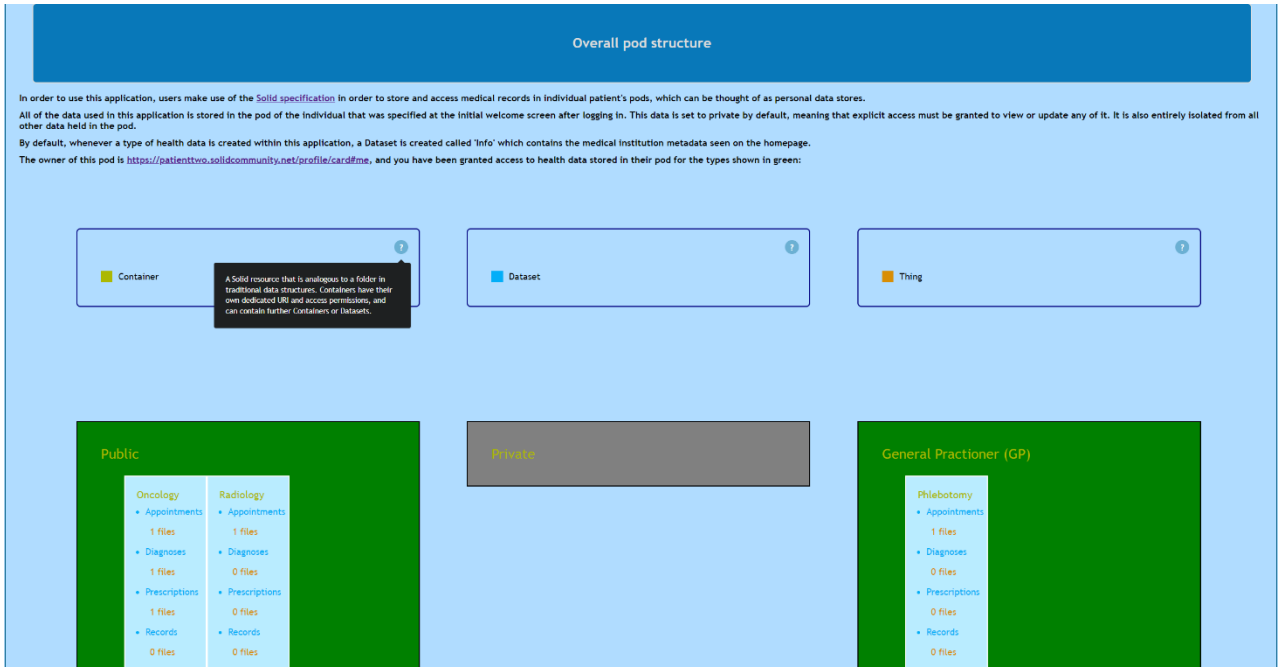


Figure 19 - Pod diagram

9.5 Managing Record Access

From the home page, users can change the access to records in two separate ways:

1. Manually by clicking ‘Access medical records’ and then ‘Manage access to selected dataset’.

2. Sharing information with an insurer by clicking ‘Make insurance request’.

The logic for managing record access was similar to how records are retrieved and displayed, so a lot of application code could be reused. Checkboxes are displayed beside each individual with access to a particular dataset, and the individual is removed from the list if an update is made that removes all of their access. A button is displayed at the bottom of the list to grant access to a new individual. Lastly, it was important to ensure that no individual is able to change the owner access that the Pod owner has on their own records, so the button to change access is disabled in the case of the Pod owner. The flow of this process is shown below in Figure 20.

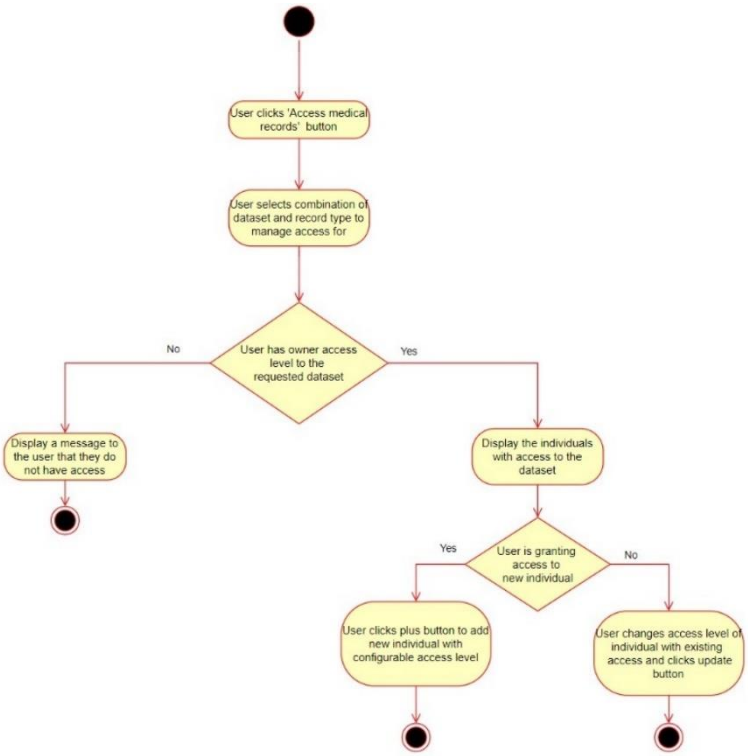


Figure 20 - Manage record access flow

The next feature that shares access to records could also be considered as uploading records but since no new records are being created, it is best described in this section. The ‘insurance request’ feature was added to the application to see how Solid could handle conditional sharing of individual Things based on the values of certain attributes. The practical application of this was that if a Patient is going to get insurance from an Insurer, the Insurer must be aware of some health conditions in order to give an estimation on how much of a risk the Patient will be to cover with insurance. The conditions that were set for this sharing of records was set as:

- Diagnoses that are less than 5 years old, from the following departments: Cardiology, Haematology, Oncology, Psychiatry, Rheumatology,

The aim was that access could be shared to the WebID of a specified Insurer, and they would be able to only view the individual Things held within the Diagnoses SolidDatasets across all department Containers that matched the criteria. However, it was discovered that granting access to any Thing within a SolidDataset grants the same access to all other Things held within the SolidDataset.

Therefore, a workaround was developed where the individual Things that matched the criteria were added to a new, separate SolidDataset and the Insurer was granted viewer access to this dataset. This meant that the records to be shared still existed at their original URIs but could be viewed by accessing the new insurance dataset, while the Insurer was still unable to view the other records within SolidDatasets that did not fit the criteria.

This feature also checks to see if the insurance SolidDataset had been previously created, and grants viewer access to the newly specified Insurer if this is the case, rather than creating a duplicate dataset. The flow of this process is shown below in Figure 21.

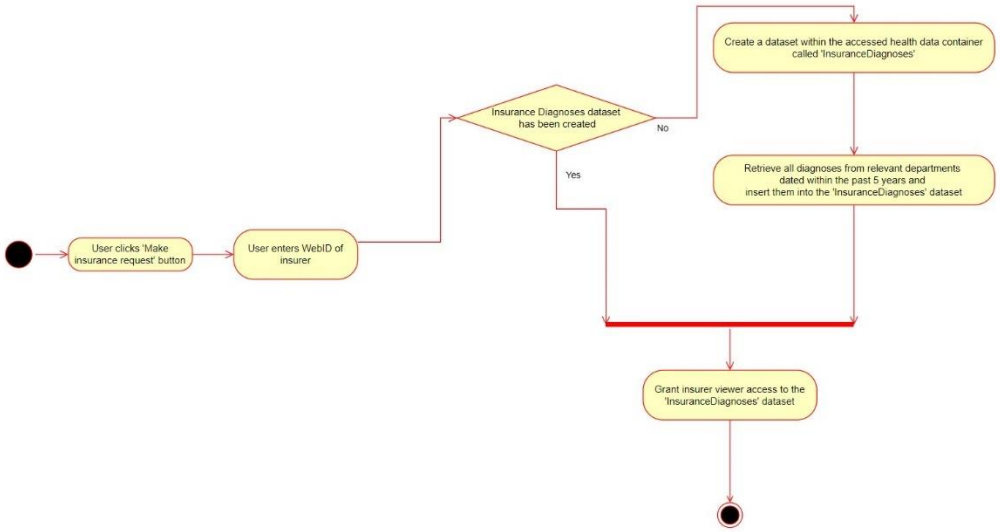


Figure 21 - Insurance request flow

Viewing the diagnoses that were shared with the Insurer is restricted to the Patient and Insurer only and is available by clicking ‘Access medical records’ from the home page and then the ‘View diagnoses for insurance’ button which is enabled after the process is complete.

9.6 Deployment

Several approaches to deployment were attempted before the final working implementation was found. The first attempt was to host the application on GitHub Pages [119] but errors

were thrown stating that the Inrupt node module could not be found on the server, due to a misconfiguration of the application that would not work with the Jekyll runtime agent used by GitHub pages [126].

Next, an Amazon Web Services (AWS) EC2 instance was created and configured for public access. The application repository was cloned onto the remote machine and ran following a tutorial on Medium [120], but a similar error appeared due a missing dependency for the Inrupt package.

After posing a question to the developer Gitter channel [121], discourse began with an individual working on the Inrupt Solid Library development team. A call was scheduled where this developer shared information about hosting the application using Vercel [122]. After confirming that hosting using AWS was not mandatory, a new approach was attempted using Vercel.

This service was very straightforward and allowed for easy integration with the project repository on GitLab. Vercel was confirmed to be the method for hosting the Solid server used by Inrupt [123]. This did require changing the application code bundler from parcel [88] to webpack [87] and small updates to the webpack configuration file including commands to run the application on the Vercel server.

With the help of this developer, the application is now hosted using Vercel at https://solid-health-app.vercel.app/. A list of credentials and corresponding WebIDs for each application user type can be found below in Table 5 below:

User Type	WebID	User Name	Password
Patient	https://patientone.solidcommunity.net/profile/card#me	PatientOne	tLADUu!86wy!LSy
Institution Administrator	https://administratorone.solidcommunity.net/profile/card#me	Administrator One	!EtWBHP55JVq8XA
Doctor	https://doctorone.solidcommunity.net/profile/card#me	DoctorOne	hApECN6QW!jfTi6
Emergency Worker	https://patientone.solidcommunity.net/profile/card#me	EmergencyWorker	jHV3C6mUVy8ZYC!
Pharmacist	https://pharmacistone.solidcommunity.net/profile/card#me	PharmacistOne	hTwc3q65LLdZf8!
Insurer	https://insurerone.solidcommunity.net/profile/card#me	InsurerOne	g!NJtiH6nLMTYp9

Table 5 - Application test user credentials & WebIDs

Implementation Summary

With the implementation complete, all of the use cases specified in Table 4 are possible within the application, along with some additional functionality. A summary of the code distribution is shown below in Table 6.

Programming Language	Lines of code	% of total lines of code
JavaScript	1789	60.9
HTML	657	22.4
CSS	490	16.7

Table 6 - Distribution of code in application implementation

Although the code could have been optimised further to reduce some lines of code, this gives an estimation of the output required to build a simple, enterprise-level application using the Inrupt Solid client package.

Chapter Four

10. Evaluation

Section Overview

Regarding the overall goal of this project set out in section 6, “to evaluate the maturity of the Solid specification as it stands today, from the perspective of a software developer”, a number of objectives were set forward which will reach this goal. This chapter describes the method under which each objective and associated targets were evaluated and the results of this evaluation. These targets spanned across different areas; therefore, they were evaluated using different methods. Each target will be mentioned in this chapter under the relevant method of evaluation.

Section 10.1 describes how the developed application as part of this project was evaluated, both from technical and conceptual perspectives. Section 10.2 describes the contributions that were made to advance Solid’s growth and assist the community. Section 10.3 describes the evaluation of Solid that has been generated over the course of undertaking this project.

10.1 Application Evaluation

There are two aspects of the application under which it can be evaluated:

10.1.1. Technically – The extent to which the application satisfies the specification that was set forward in the design phase of development, and the quality of the developed software.

10.1.2. Conceptually – The level of use that the application would serve to its intended domain, i.e., the usability that professional health workers have when using the application to complete real-world tasks.

10.1.1 Technical Evaluation of Application

To verify that each of the use cases are possible within the application, a set of Use Case Tests was designed with instructions on how to complete a particular feature within the application. These tests do not correspond to the ID of individual Use Cases as specified in Table 4 but the set of all Use Case Tests satisfies the set of Use Cases.

Additionally, Functional Tests were developed to test background processes or additional functionality that the application performs as part of completing a main feature, e.g. data type validation on user input, updates to access of Patient pod when the Institution Administrator is updated.

These tests can be performed by a number of users depending on the required access of the application feature in question. Some of these tests have a dependency of the successful completion of a prior test, so a test run specification suite was designed to identify the necessary order of test runs and record the result of each test result. This is shown below in Figure 22.

Order	Test Document ID	Test Name	Test Dependencies	Testing Date	Test Result
1	UC-01	Register for application	N/A	07/03/2022 14:42	Pass
2	F-02	Update medical institution details	UC-01	07/03/2022 14:46	Pass
3	UC-02	Register new medical institution	UC-01	07/03/2022 14:50	Pass
4	UC-03	Register new appointment	UC-02	07/03/2022 14:54	Pass
5	F-04	Verification of form field validation	UC-01	07/03/2022 14:58	Pass
6	F-01	Verification of insitution administrator privileges	UC-01, UC-03	07/03/2022 15:02	Fail
7	UC-04	Upload general record	UC-01, UC-03	N/A	N/A
8	UC-05	Upload diagnosis	UC-01, UC-04	N/A	N/A
9	UC-06	Upload prescription	UC-01, UC-05	N/A	N/A
10	F-03	Verification of emergency worker privileg	UC-01, UC-03, UC-06	N/A	N/A
11	UC-09	View pod diagram	UC-01, UC-02, UC-03, UC-04, UC-05, UC-06	N/A	N/A
12	UC-07	Add/revoke access to data	UC-01, UC-03	N/A	N/A
13	UC-08	Make insurance request	UC-01, UC-03, UC-05	N/A	N/A
1	UC-01	Register for application	N/A	07/03/2022 17:02	Pass
2	F-02	Update medical institution details	UC-01	07/03/2022 17:06	Pass
3	UC-02	Register new medical institution	UC-01	07/03/2022 17:09	Pass
4	UC-03	Register new appointment	UC-02	07/03/2022 17:13	Pass
5	F-04	Verification of form field validation	UC-01	07/03/2022 17:19	Pass
6	F-01	Verification of insitution administrator privileges	UC-01, UC-03	07/03/2022 17:26	Pass
7	UC-04	Upload general record	UC-01, UC-03	07/03/2022 17:42	Pass
8	UC-05	Upload diagnosis	UC-01, UC-04	07/03/2022 17:46	Pass
9	UC-06	Upload prescription	UC-01, UC-05	07/03/2022 17:52	Pass
10	F-03	Verification of emergency worker privileg	UC-01, UC-03, UC-06	07/03/2022 17:56	Pass
11	UC-09	View pod diagram	UC-01, UC-02, UC-03, UC-04, UC-05, UC-06	07/03/2022 18:01	Pass
12	UC-07	Add/revoke access to data	UC-01, UC-03	07/03/2022 18:13	Pass
13	UC-08	Make insurance request	UC-01, UC-03, UC-05	07/03/2022 18:20	Pass

Figure 22 – Test run specification

The initial plan for these test runs was to gather a group of test users and provide a Use Case test with corresponding login credentials to each user, ensuring that both test documents and login credentials are distributed anonymously when a user begins the testing process, so that no sensitive data is collected from participants. They would then fill out a feedback form, specifying the ID of the test that they had completed, which would give an idea on perceived quality of the application from the perspective of the different user types with their corresponding intended use cases.

However, due to initial difficulties deploying the application on a live environment, the style of user evaluation plan had to be changed towards an active demonstration/feedback approach with professionals in the industry, as discussed in section 10.1.2 below. This meant that the researcher conducting this project undertook the test process individually and took video recordings of test executions to verify results. This required changing the evaluation participant information sheet and consent form appropriately to reflect the changes to data collection, found at participant consent.

The test result videos, test run specification and individual test scripts can be found in the following repository: Testing Repository. This repository also includes an instruction document on running tests and their purpose.

As can be seen from Figure 22 above, the first test run failed on F-01 due to an error updating Institution Administrator access. After implementing the fix and attempting a second run, the tests passed. With this result, Objective **O1** has been met. Target **T1.1** is satisfied by Use Case Tests [UC-04, UC-06], **T1.2** is satisfied by Use Case Test UC-07, and **T1.3** is satisfied by Functional Test F-03.

In terms of a critical evaluation of the developed code, a few critiques are listed below in Table 7.

Application Technical Critiques
All of the application UI components use native HTML and CSS, but a more appealing interface could have been achieved using React [75], or some other component library.
Application often updates HTML elements in the DOM to be 'hidden', which is not an issue when the application is the size that it is currently, but this would be an issue for scaling up the application to include more interfaces. Proper routing between HTML components would be desired instead.
Form input field validation is performed using HTML RegEx pattern matching [132], but validation could be made stricter to reduce downstream errors. e.g., by validating that URIs entered are the WebID of a Solid profile, as opposed to just using the HTTPS scheme as is done currently.
Asynchronous code execution was important in the creation of new resources and subsequent access provision, but it means that the required time to complete some operations can be between 5 to 10 seconds.
Application styling was tailored for a laptop-sized screen, meaning that some components look misplaced when the application is opened on a large monitor.
A lot of code updates the DOM with elements based on Resources where the size of the Resource is not known until a GET request is complete. This means that the Resources are processed and converted to HTML elements in the JavaScript logic file, where it takes many more lines of code as it would in a HTML file. This could be made better with more placeholder HTML elements that describe styling and other attributes, while the JavaScript file strictly updates element content.

Table 7 - Application technical critiques

10.1.2 Conceptual Evaluation of Application

As mentioned in the section above, the main external evaluation of the application took place in the form of demonstration sessions with individuals working in two separate areas:

1. Healthcare / Healthcare informatics
2. Solid / Data engineering

It was important to get the perspective of people from these two cohorts, as the application touches on both areas. Only a handful of individuals working in these areas were available but their feedback is more valuable to this project than a larger number of individuals with limited knowledge of either topic.

The goal of these demonstration sessions was to get some context on how the processes that have been created for the application compare with what is done in practice. The sessions lasted between 30 minutes to an hour, depending on the length of the discourse about the project. An initial presentation of the project and the purpose of the application was given, which is supplied in Appendix 4, followed by a walkthrough of some use cases using the application. Participants were then asked to fill out an anonymised feedback form, modelled from a standard PSSUQ form and altered slightly to include questions aimed at evaluating the sentiment towards a Solid application in the healthcare domain. The questions are rated from a scale of 1 – 10, and are listed below in Table 8.

Question ID	Question Description
Q1	The system was easy to use.
Q2	The steps taken to achieve a task using the system were similar to what I would have done intuitively.
Q3	At all times using the system, it was clear what was being performed and why.
Q4	At any point where I felt unsure about the purpose of what was being performed, the use of explanatory tooltips and other on-screen information helped in my understanding.
Q5	The user interface of the application was appealing.
Q6	I think this application would be useful in the healthcare sector.
Q7	I think the transition from current practice in healthcare to use this application would be easy.
Q8	I would feel confident storing my medical records in a Solid pod.
Q9	If my records were stored in a Solid pod, I think this application would offer no greater of a risk to their security.

Table 8 - Demonstration feedback form questions

Due to the length of time required to conduct these sessions, a total of 12 sessions were held; 9 of which generated responses on the feedback form. A plot of the mean result of each of the above questions is shown with blue markers below in Figure 23, along with the corresponding standard deviation of responses above and below the mean, shown with a red line.

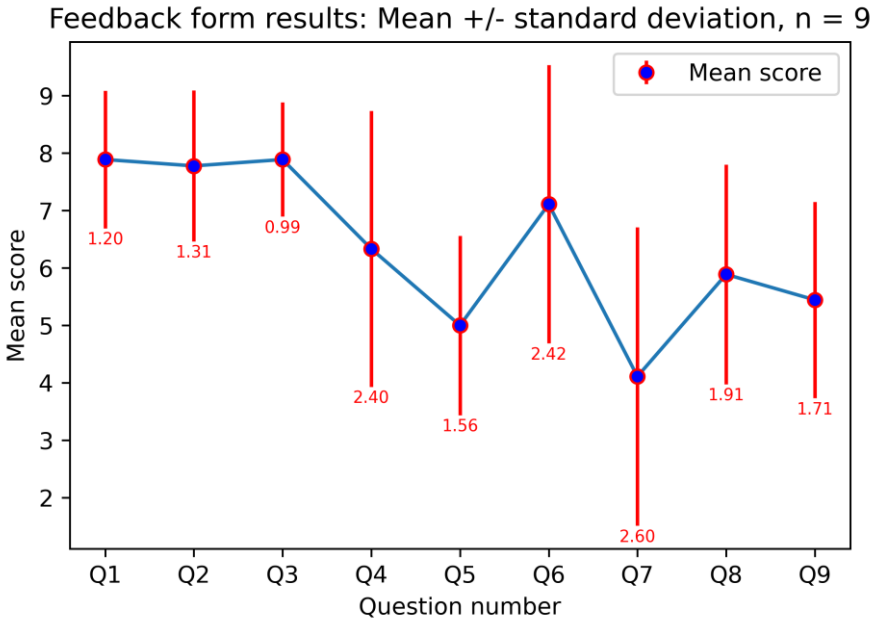


Figure 23 - Mean & standard deviation of feedback form results

From these results, a few conclusions could be drawn:

- **The application is relatively straightforward to use.** This is due to high scoring results on Q1 – Q4, with a slight drop in scores and bigger spread of results in Q4. This is possibly due to the rushed nature of the demonstrations, where participants may have had more time to properly interact with tooltips if they were using the application by themselves.
- **The application would be helpful to the healthcare sector, but the transition to Solid would not be easy.** This is due to a high average score in Q6 and a low average score in Q7. This was an expected result, as the widespread use of this application proposes substantial changes, both architecturally and procedurally, to how healthcare would be delivered. However, high variance in results for both questions means that a more accurate mean score could be acquired with a larger sample size.
- **Individuals introduced to Solid are cautious.** This is due to mediocre scores on Q8 and Q9, which signals that the general population would be hesitant to fully trust Solid technology to protect their personal data.

Tailoring the feedback form to include more personalised questions relating to the project enabled these conclusions to be drawn, but it meant that the standard PSSUQ metrics (SYSUSE, INFOQUAL, INTERQUAL) could not be gathered accurately. Therefore, **T2.3** was disregarded in an attempt to gather more project-specific feedback, meaning **T2.4** became unachievable in any meaningful way.

However, the most valuable feedback from these sessions came from outside of the feedback form, in the form of conversations about professional practice. As there was no proper model of the healthcare domain before designing the application use cases, this was the area with the most valuable insights from professionals. A list of notable comments about application processes with regard to professional healthcare practice is shown below in Table 9.

Application Procedural Critiques
In practice, data shared with insurance companies is the required minimum. This would usually come in the form of a questionnaire that healthcare workers would fill out on behalf of the patient and would not include a list of diagnoses, as is in the application.
Some health data is sensitive to the patient. In most cases, they certainly should not be able to change records and in some cases, they should not be able to even view specific records. Records that a health worker, e.g., doctors, nurses, could view are different to what a patient could view.
Pharmacists would need to see more than a single prescription in order to be sure that the patient is not taking any other medication that might interfere with the prescription that they have to fill. Emergency workers would be in a similar position, they would need to be aware of other diagnoses, allergies, etc.
It is common for applications which work with electronic health records to have some sort of 'break the glass' functionality, where an individual without access to a record can gain access when necessary. This would need to be possible within the application before a medical institution would think about using the application.
Emergency Workers may not know the WebID of a patient that they have just come in contact with. Without a uniform WebID structure based on an individual's known identifier, this feature would have to resort to an identification card containing a patient's WebID, which is subject to displacement, the initial motivation for developing this feature.

Table 9 - Application procedural critiques

Many other concerns about data security, sharing policies and bio-ethics were discussed in these demonstration sessions, which confirmed the complexity of operating with extremely

sensitive patient data. This was suspected to be the case, as the aim of the project was to test the capabilities of Solid in building a serious application.

However, with these feedback sessions completed, objective **O2** has been met. Target **T2.1** and **T2.2** have been explicitly met as part of this, while **T2.3** and **T2.4** were discarded to get more specific feedback on sentiments towards Solid usage in the area. Furthermore, the demonstration sessions did not require collecting any participant data, so a full ethical application form was not required, but the appropriate consent form is found at the participant consent directory. This means that objective **O3**, including **T3.1** and **T3.2** have also been met.

10.2 Evaluation of Contribution to Solid

An important aspect of completing this project was not only to investigate the maturity of Solid; but to do all that is possible to assist those that are already involved in Solid and help introduce Solid to others. Therefore, efforts were made to engage with the Solid ecosystem to share insights and feedback. This was done by:

10.2.1. Committing to maintain reusable application code, a diary of notes and lessons learned during development that could help other developers.

10.2.2. Contribution to Solid specifications, events, software libraries and documentation that can help maintainers.

10.2.1 Contributions for Solid Developers

From as early as the exploration phase of the design process, a development diary was kept to record daily progress and experiences using Solid, shown in Appendix 3. A lot of these entries were made irrelevant by discoveries or decisions later in development, but it shows the process and adaptive approach required that was required for development with Solid This also means that objective **O4**, with corresponding targets **T4.1** and **T4.2** have been met.

Another important aspect of the implementation of the application design was to create code related to Solid which could be reused in another practical application. This was important because it was noticed that only a very small amount of code could be found online that could be adapted for use in this application, so an effort was made to reduce this problem for others. In the application repository [124], most application-specific code was kept to one main source file; while code to read and write to Solid Pods was kept in two fully-commented files. These two files purely interacted with Solid Pods generically and will be a valuable resource for the future Solid developers.

A post was made to the community forum at the end of development [133], sharing the achievements of the application. These were pieces of functionality that the application could

perform which could be used in a Solid application for any industry or practical setting, and were not found anywhere in the documentation or resources for use in this project. These achievements included:

- Dynamic building of a Thing based on the presence or absence of data provided by user input.
- Generic way of reading and displaying properties and values of Things stored in a Dataset, regardless of property names of the Thing.
- Generic function to insert new Things into a Dataset, given an object containing property-value pairs and the Dataset URI.
- Deleting a solid Container by cycling through contained Containers and Datasets and deleting each before deleting the overall Container.
- Retrieve all Things within a Dataset and provide access for an individual to a subset of the Things based on search condition, by inserting the subset to a new Dataset.
- A diagram to show the application data stored in the user's pod, including the overall number of Containers, Datasets and Things related to the application.

This post will hopefully be available for Solid developers in the future to reuse and save time, meaning that target **T5.3** of objective **O5** has been met. Through active engagement with individuals on the forum throughout the project, a number of valuable discussions were had, including sharing solutions to a common problem [134]. Two trust member levels were gained through this engagement, meaning that both **T5.1** and **T5.2** have been met too, and therefore, **O5** has been met also.

10.2.2 Contributions for Solid Maintainers

Through communication on the Solid community forum, several meetings were set up with people professionally involved in Solid. The first of these was with a developer of the Inrupt Solid library, where they helped with deploying the application. Following the assistance they gave, a long form conversation was had about experiences using Solid. This included feedback on inconsistencies using the Inrupt library, which was relayed to the rest of the development team and will be discussed more in section 10.3 below. Feedback was also shared about the solidcommunity.net Pod browser, which was found to be buggy and temperamental throughout development. Furthermore, attempts were made to create a Pod with Enterprise Solid Server [123], but a 405 HTTP error prevented login, which was also shared with the rest of the development team.

Another meeting was held with Justin Bingham of Janeiro Digital where a discussion was had about the NHS project [73] and its similarities with this project. This started with sharing experiences of developing a full application specification in Solid, including concerns about

the usability of Solid applications for the average user, the amount of work required to overcome basic roadblocks with Solid that seem to affect most development projects, and the extent to which Solid developers need to be able to adapt the specification to what is supported by the technology.

Next, a long conversation was had about the Solid Application Interoperability specification [61], of which Justin is an editor. Justin's company are currently working on a number of libraries in the Java, JavaScript and TypeScript programming languages which implement this specification and will ultimately facilitate better experiences developing in Solid. This is the most appealing Solid specification, as it encourages good development practices and a well-needed structure to the process of development across a range of frameworks. Therefore, commitments have been made to get involved with the interoperability panel [135], which meet weekly and contribute towards writing the specification.

Separate arrangements have been made to speak at a Solid World event [65], which are held every month. This was suggested by the developer of the PodPro application [67] in a separate meeting, who spoke at the March 2022 event. This will hopefully help share insights and lessons learned to a wider audience and help to foster more engagement in the Solid community.

10.3 Solid Evaluation

Lastly, an evaluation of Solid at the current time of writing is given. This evaluation includes the aspects of Solid that were engaged with during the development of this project:

10.3.1. Documentation

10.3.2. Technology

10.3.3. Community

10.3.1 Solid Documentation Evaluation

The official technical specifications [98] are thorough and cover all aspects of Solid usage. However, as mentioned in section 7.2.1, understanding some of these specifications requires a reasonable understanding of related concepts (RDF, Turtle, JSON-LD), which makes them slightly intimidating to newcomers. This is to be expected with technical documents, but there is nowhere else that describes Solid at a slightly higher abstraction level, meaning that these documents are almost the single point of reference for developing in the space, across all frameworks.

Looking at the documentation of libraries endorsed by the Solid Project website, it's clear that the Inrupt JavaScript package is the most supported. This can be said because it is the package used in the official developer guide [43] and is the most documented library out of

the collection. Packages in other languages either do not have a corresponding document reference at all, or are not adequately documented.

Then looking at the documentation for this library, it does provide a list of terms used in Solid [91] and a full API reference [90], which were extremely valuable throughout development. However, the API references lists some key functions as 'subject to change, even in a non-major release', which is not reliable for professional development. Furthermore, on one occasion during development, a major section of the API reference disappeared for a number of hours.

The documentation for this package is missing a key aspect of data access management in Solid, creating groups. There is a function for setting/getting the access that a group has, but nowhere in the API reference or Inrupt tutorial [89] mentions how to create a group or manage group members. This would have been valuable in this application, instead of assigning access to individuals. This request has been shared with the Inrupt team and will be included in a future release of the documentation.

Another aspect of the documentation that is missing, this time in the Inrupt developer tutorial [97], is for creating an Auxiliary Resource for access permissions to a primary Solid Resource. This is deceiving because the tutorial skips ahead to getting the access or updating it, but never mentions instantiating the access Resource to begin with. Therefore, following the Inrupt tutorial for 'Manage Access to Data' as it stands currently, will only cause errors. It was difficult in situations like this, when it is unknown whether the implementation, technology or instructions are to blame and can consume a lot of time when very few resources exist on the web.

10.3.2 Solid Technology Evaluation

Technically, Solid can support any type of application technology. Solid servers must support standard HTTP requests for accessing and updating Resources, although the request body needs to be in Turtle or SPARQL format, which requires an amount of data transformation. Although this can be helped by packages that help with the conversion from one programming language to another [69], it still requires a considerable amount of work that should be taken into account in any application design. Therefore, an increased number of Solid libraries in various languages would be helpful and could attract developers specialised in those languages to get involved with Solid development.

One of the largest drawbacks to developing with Solid was that there was a huge amount of time spent on developing the basic interactions with Solid Pods that made appending additional application logic difficult. This was either down to functions not behaving as they should, inconsistencies in the documentation, or functionality which is simply not supported

at the current time of writing. In communication with other Solid developers, this was a recurring roadblock for development that seems to affect most projects. A useful feature that could be implemented, as mentioned by an individual during a demonstration session, would be an API interface between a client application and Solid Pods, that could handle all of the data interactions and identity & access management that is required for an enterprise application, and free developers to focus on the practicality of their application logic.

In terms of the Inrupt technology used in this project to implement the Solid specification, a number of critiques are listed below in Table 10 which were noticed throughout development.

Inrupt Solid Library Critiques
No way to provide access to individual Things within a SolidDataset without providing access to the entire SolidDataset.
No data validation on updates to an ACL Resource. This can cause a lock out of the Resource, where it becomes impossible to view or delete the Resource. Furthermore, provisioning access to arbitrary URL's is allowed, e.g., **www.google.com**
Inconsistent ways to grant access to a Resource. When granting access using WAC for a newly created Resource, only 'control' needs to be specified (which sets both 'controlRead' and 'controlWrite' to true). However, when updating a Resource's ACL later, both 'controlRead' and 'controlWrite' need to be specified.
Proper log out is not supported. An authentication cookie remains in the browser storage, meaning that the previous session is continued when the user attempts to log in again.

Table 10 - Inrupt Solid library critiques

Furthermore, a number of limitations of the Solid specifications are listed below in Table 11 which restricted the ideal functionality of the application.

Solid Specification Critiques
No way to limit the maximum amount of access that a user can grant to other users. This would be ideal for granting Doctors ‘control’ access to a Prescriptions SolidDataset, where they would only need to grant viewer access to downstream Pharmacists.
Cannot grant editor or viewer access to a Resource without possessing that access. This led to a lot of conceptual weaknesses, with Patients required to be able to change their own records to grant this permission to Health Workers.
If requesting a Resource that does not exist in another individual’s Pod, given a 403 Unauthorized response instead of a 404 Not Found. This made it impossible to know if records exist or not, unless the Pod owner is signed in.
Unable to view the list of individuals with access to a Resource without being able to change the access to the Resource. This means that certain individuals had to have access above what is required from them from a procedural perspective.

Table 11 - Solid specification critiques

10.3.3 Solid Community Evaluation

From the experience during this project, the community’s eagerness to help has been the greatest asset for Solid development. Of the channels mentioned in section 7.2.3, the community forum was the most valuable. With over 17,000 posts across 1,500 topics, the community forum has seen plenty of activity since its conception in October of 2018. There are over 1,500 users on this platform, but only 92 have been active in the last 90 days at the time of writing, and only 773 have earned the most basic trust level which is earned after any sort of interaction. This suggests that the forum has a high turnover rate.

Responses on the platform were mostly timely and informative. The majority of these are from community employees at Inrupt or maintainers of the community forum (of which there is a large overlap), but some come from independent developers who have experienced a similar issue with development in Solid. However, it appears as though there is not enough of a workforce to address all topics, as a lot of posts remain unanswered.

A lot of posts from the community are focused on implementing the Solid specification in other technologies and frameworks such as ASP.NET [137], which has been a work-in-progress for over 3 years. As discovered through meetings with the developer of PodPro, a dedicated library has been developed for the Elixir programming language; with plans to be released to the public in the future. Based on this, it appears that the community are the greatest advancers of implementing Solid on new technologies, while Inrupt continue to focus on their JavaScript libraries.

Gitter channels were also a useful resource for getting quick answers but are more unstructured and difficult to search through. A comparison of the activity on some of the main Gitter channels is shown below in Table 12, compared with the status in 2019 as reported by Lamba [16]. This shows us that the general chat has received a 149% increase in users, while the development chat received a 208% increase.

Year	Room	User Count
2019	solid/chat	795
2019	solid/app-development	130
2019	solid/solid-spec	310
2022	solid/chat	1982
2022	solid/app-development	401
2022	solid/solid-spec	243

Table 12 - Gitter chat activity

Outside of these dedicated channels for Solid communication, activity remains low. For example, Table 13 below shows the state of Solid engagement on Stack Overflow in comparison with its state in 2019. While it can be seen that there is more engagement on average Solid topics than in 2019, the overall number of questions being asked remains low.

	2019	2022
# Questions:	10	41
# Questions: No answer	5	12
# Questions: 1 answer	3	21
# Questions: >1 answer	2	8

Table 13 - Stack Overflow Solid interactions

Evaluation Summary

To summarise the evaluation of the work that has been completed, all of the objectives and related targets have been achieved except for **T2.3** and **T2.4** of objective **O2**, as they were discarded in favour of gathering the consensus on questions more closely related to Solid and its potential adoption in the healthcare sector.

In terms of the requirements specifications as listed in Appendix 1, all of the User Requirements were also met apart from **UR014** – 'Allow Doctors to edit and delete existing medical records stored in a Patient's pod'. However, this functionality has been shown in other parts of the application; for example, editing fields within a Thing is possible when updating the medical institution details, and deleting individual Things is possible when instantiating a new type of health data. This requirement was omitted from the implementation in favour of the pod diagram feature.

Chapter Five

11. Conclusion

Solid as a concept is extremely appealing. Widespread adoption of the specifications paint a utopian depiction of data control that is desperately needed in today's society. This belief is shared by mostly everyone who becomes aware of the specification, which is echoed in the community's eagerness to advance the growth of Solid. It has already been shown that Solid is fully capable of building simple applications to support basic read/write operations, as can be seen from the list of applications available at the Solid Project website [40]. Therefore, this project attempted to push the capabilities of the recommended Solid technology to build a fully functional enterprise application that deals with the most sensitive type of data, medical records.

It is clear that there was an improper model of the healthcare domain while designing the application, particularly regarding the tightly controlled data protection regulations surrounding medical data. There was also a very broad scope of the application that attempted to encompass many aspects of handling medical records and has been shown to be too great of a transition for health workers.

Furthermore, the attempt to embody the Solid ideals with regard to complete data ownership was not implemented in a procedurally appropriate way for medical records stored in the Pod of the patient, due to limitations of the Solid specification that requires pod owners to be able to perform any type of action on the data stored in their Pod. However, avoiding this problem would require storing records in a central location, leading back to the concerns of ransomware attacks that motivated the application area.

While the bugs reported with the Inrupt package could have been avoided by building a separate Solid client package, it would take a huge amount of additional effort and require an in-depth knowledge of the protocol. Furthermore, applications would still be constrained by limitations in the underlying specification that the client must adhere to. For example, the lack of a proper identity & access management system is something that would seriously deter organisations from attempting to build an application in Solid. Of course this could be avoided by building a dedicated IAM system upon the WebID returned by Solid servers, but again this would take a huge amount of additional effort and would require additional user data to be stored outside of user's pods.

Other limitations with the Solid protocol from an application design perspective may be avoidable by a regulated system of data stored across multiple Pods and accessed by corresponding proxy user accounts authenticated in parallel application sessions, but this would require a lot of additional resources along with a monitored register of proxy user data and data locations, since Solid heavily relies on the exact URI of Resources.

Therefore, in relation to the research question in section 5 that this paper has attempted to answer; unless an organisation desperately requires using Solid in their design, it's not entirely feasible in its current state. If an organisation wanted to avoid the issues mentioned with the currently available technologies that adhere the Solid specification, it is possible that an enterprise-level application could be built from first principles if they possessed a thorough knowledge of RDF and the Solid specification. This would require tailored libraries and servers to interact with Solid Pods, which many companies would simply view as having too much overhead in the form of development resources and related research.

Solid will ultimately become more widespread with the release of many interesting libraries and technologies to be released in the coming years, which will attract a wave of developers and investors to the space, generating products for the general public. However, until such a time is truly realised, Solid will remain as a passion project at the brink of breaking into the mainstream.

12. Future Work

The remaining work to be done as part of this project is divided between furthering the advance of Solid for developers and continuing to improve the application that has been developed for the healthcare sector.

12.1 Future Work in Solid

In terms of working with Solid, efforts have begun to become associated with the Interoperability Panel. This has been seen to be the most important workforce for Solid developers and they are thought of as the regulators for proper data handling across applications using separate frameworks, which will certainly be necessary when Solid becomes widespread across these frameworks.

Outside of this, further reports of the experience using the Inrupt JavaScript library will be sent to Inrupt. Inrupt are working hard to make Solid appealing to all types of individuals with their products and documentation and they could benefit from the feedback that has been generated through this development. One such product in development is a top-level library to simplify the specification down to very primitive operations, effectively allowing developers to enjoy the benefits of Solid without the burden of developing around the specification. Beta access to this package will be acquired when possible.

Finally, communications between the community will be monitored for the foreseeable, to give aid to anyone who is experiencing similar problems or would like to gain some insights on the experience of the project.

12.2 Future Development Work

Although a proposal has been received to collaborate in development for a Solid healthcare application, efforts are better spent in developing additional features for the prototype application that would make it usable in the healthcare sector. This is not in an attempt to commercialise the product, but to complete a commemorable application that could be featured on the official Solid Project website list of applications.

The first feature to implement from a data control perspective would be to build a log of changes to data or data access within a Pod. This would not be difficult to implement from an application logic perspective and does seem required in the healthcare sector, as per the feedback received from professionals. However, due consideration will have to be placed in the location and access rules to these logs, as it could easily lead back to the initial problem of collections of sensitive data.

With a deeper dive into the Solid Notifications Protocol, a full messaging system could be implemented in the application. This would be ideal for requesting access to a Resource, or for notifying data owners when access to their data has changed.

Another feature to implement would be to retrieve the current list of medications or the next appointment that a Patient may have. This could have been implemented in this project, but will only require additional application logic that is not related to testing Solid. Now that the capabilities of Solid are known, this feature would certainly make the interface more appealing and usable for Patients.

13. Final Remark

Although this project was frustrating at times, I have genuinely enjoyed it. There was a time about mid-way through the project when implementing features with Solid seemed seamless and I actually enjoyed the process of thinking dynamically to devise workarounds when certain functionality didn't work as intended. The project has also drastically improved my skills in JavaScript development, of which I started with very little. I've also become much more conscious of online security and data privacy by way of developing the application and researching related fields, which I plan on continuing throughout my future career.

It was an ambitious application specification and after conversations with professionals in healthcare, I see that there could have been a feasible application of Solid in a more narrowed field, e.g., simply sharing record access between healthcare professionals. However, as mentioned above, I came into this project with no knowledge of healthcare procedures other than what I had personally experienced; so I don't have too many regrets about the conceptual shortcomings of the application. I'm also quite proud of the project's accomplishments, with a fully validated application, and much more engagement with both the Solid community and healthcare professionals than I could have expected.

I have no doubt in my mind that Solid, and what it stands for, is a good thing. We're at a time where personal data has been monopolised and commercialised by a small group of corporations, and I think a lot of people just don't see that this is a problem, or more importantly that there is a solution in sight. I believe Solid is this solution, but it will only become a part of daily life when there is a product that draws in the masses. For society's sake, I hope that this product will be created and we can regain control of our online identity. Until then, I'll remain as one of the many developers who have been introduced to Solid and have been interested by the concept of it.

References

[1] Hovhannisyan, K., Bogacki, P., Colabuono, C., Lofu, D., Marabello, M. and Eugene Maxwell, B., 2021. Towards a Healthcare Cybersecurity Certification Scheme. 2021 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA),.

[2] Zalozhnev, A., Andros, D., Ginz, V. and Loktionov, A., 2019. Information Systems and Network Technologies for Personal Data Cyber Security in Public Health. 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC),.

[3] The Irish Times. 2022. HSE confirms data of 520 patients published online. [online] Available at: <https://www.irishtimes.com/news/crime-and-law/hse-confirms-data-of-520-patients-published-online-1.4578136> [Accessed 8 January 2022].

[4] McNally, S., 2022. HSE hackers were in health service's computer system for eight weeks before cyber attack. [online] TheJournal.ie. Available at: <https://www.thejournal.ie/hse-hack-report-5626054-Dec2021/> [Accessed 8 January 2022].

[5] BBC News. 2022. Cyber attack 'most significant on Irish state'. [online] Available at: <https://www.bbc.com/news/world-europe-57111615> [Accessed 8 January 2022].

[6] Solidproject.org. 2022. Solid Protocol. [online] Available at: <https://solidproject.org/TR/protocol> [Accessed 8 January 2022].

[7] Solidproject.org. 2022. About Solid · Solid. [online] Available at: <https://solidproject.org/about> [Accessed 8 January 2022].

[8] Solid.mit.edu. 2022. Solid. [online] Available at: <https://solid.mit.edu/#about> [Accessed 8 January 2022].

[9] Solidproject.org. 2022. Origin · Solid. [online] Available at: <https://solidproject.org/origin> [Accessed 8 January 2022].

[10] Lassila.org. 2022. [online] Available at: <https://www.lassila.org/publications/2001/SciAm.pdf> [Accessed 8 January 2022].

[11] W3.org. 2022. Linked Data - Design Issues. [online] Available at: <https://www.w3.org/DesignIssues/LinkedData> [Accessed 8 January 2022].

[12] W3.org. 2022. World Wide Web Consortium (W3C). [online] Available at: <https://www.w3.org/> [Accessed 8 January 2022].

[13] Knowledge@Wharton. 2022. Your Data Is Shared and Sold...What's Being Done About It? - Knowledge@Wharton. [online] Available at: <https://knowledge.wharton.upenn.edu/article/data-shared-sold-whats-done/> [Accessed 8 January 2022].

[14] Dragan, I. and Zota, R., 2017. Collecting Facebook data for big data research. 2017 16th RoEduNet Conference: Networking in Education and Research (RoEduNet), pp. 1-3, doi: 10.1109/ROEDUNET.2017.8123757.

[15] Solid Community Forum. [online] Available at: <https://forum.solidproject.org/> [Accessed 10 January 2022].

[16] 2018. A Solid-powered decentralised social network for academics: An evaluation of key considerations for developing practical Solid-powered applications. Akashdeep Lamba MSc, Postgraduate. Trinity College Dublin. [online] Available at: < https://www.scss.tcd.ie/publications/theses/diss/2019/TCD-SCSS-DISSERTATION-2019-046.pdf>

[17] Sites.google.com. 2022. Getting a SOLID start. - Clean Coder. [online] Available at: <https://sites.google.com/site/unclebobconsultingllc/getting-a-solid-start> [Accessed 10 January 2022].

[18] Team, S., 2022. Solidity Programming Language. [online] Solidity Programming Language. Available at: <https://soliditylang.org/> [Accessed 11 January 2022].

[19] Inrupt.com. 2022. [online] Available at: <https://inrupt.com/> [Accessed 11 January 2022].

[20] Ideas for Solid Apps. [online] Solid Community Forum. Available at: <https://forum.solidproject.org/c/build-a-solid-app/app-communities/10> [Accessed 11 January 2022].

[21] Verborgh, R., 2022. List of Publications. [online] Ruben.verborgh.org. Available at: <https://ruben.verborgh.org/publications/> [Accessed 12 January 2022].

[22] Verborgh, R., 2022. Re-decentralizing the Web, for good this time. [online] Ruben.verborgh.org. Available at: <https://ruben.verborgh.org/articles/redecentralizing-the-web/> [Accessed 12 January 2022].

[23] The Economist. 2022. Break down these walls. [online] Available at: <https://www.economist.com/leaders/2008/03/19/break-down-these-walls> [Accessed 13 January 2022].

[24] Pariser, E., 2014. The filter bubble. New York: Penguin Books.

[25] World Wide Web Foundation. 2022. Three challenges for the web, according to its inventor. [online] Available at: <https://webfoundation.org/2017/03/web-turns-28-letter/> [Accessed 14 January 2022].

[26] World Wide Web Foundation. 2022. The web is under threat. Join us and fight for it.. [online] Available at: <https://webfoundation.org/2018/03/web-birthday-29/> [Accessed 14 January 2022].

[27] Blog.dshr.org. 2022. It Isn't About The Technology. [online] Available at: <https://blog.dshr.org/2018/01/it-isnt-about-technology.html> [Accessed 14 January 2022].

[28] Verbrugge, S., Vannieuwenborg, F., Van der Wee, M., Colle, D., Taelman, R. and Verborgh, R., 2021. Towards a personal data vault society: an interplay between technological and business perspectives. 2021 60th FITCE Communication Days Congress for ICT Professionals: Industrial Data – Cloud, Low Latency and Privacy (FITCE),.

[29] Verborgh, R. and Taelman, R., 2022. LDflex: a Read/Write Linked Data Abstraction for Front-End Web Developers. [online] Drive.verborgh.org. Available at: <https://drive.verborgh.org/publications/iswc2020-ldflex.pdf> [Accessed 14 January 2022]

[30] Haesendonck, G., Dimou, A., De Meester, B. and Verborgh, R., 2022. Linked Data in the Web of Things. [online] W3.org. Available at: <https://www.w3.org/WoT/ws-2019/Papers/31%20-%20Haesendonck%20et%20al,%20Ghent%20University%20-%20Linked%20Data%20in%20the%20Web%20of%20Things.pdf> [Accessed 14 January 2022].

[31] App, B. and →, n., 2022. Build a Solid App. [online] Solid Community Forum. Available at: <https://forum.solidproject.org/c/build-a-solid-app/24> [Accessed 15 January 2022].

[32] Gitter.im. 2022. solid/chat. [online] Available at: <https://gitter.im/solid/chat> [Accessed 15 January 2022].

[33] Enisa.europa.eu. 2022. ENISA Threat Landscape 2021. [online] Available at: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021> [Accessed 16 January 2022].

[34] Ncsc.gov.ie. 2022. National Cyber Security Strategy. [online] Available at: <https://www.ncsc.gov.ie/pdfs/National_Cyber_Security_Strategy.pdf> [Accessed 16 January 2022].

[35] W3.org. 2022. Linked Data. [online] Available at: <https://www.w3.org/standards/semanticweb/data> [Accessed 17 January 2022].

[37] W3.org. 2022. RDF 1.1 Concepts and Abstract Syntax. [online] Available at: <https://www.w3.org/TR/rdf11-concepts/> [Accessed 17 January 2022].

[38] Solidcommunity.net. 2022. [online] Available at: <https://solidcommunity.net/> [Accessed 17 January 2022].

[39] Solidproject.org. 2022. Tools and libraries overview · Solid. [online] Available at: <https://solidproject.org/developers/tools/> [Accessed 17 January 2022].

[40] Solidproject.org. 2022. Solid Applications · Solid. [online] Available at: <https://solidproject.org/apps> [Accessed 17 January 2022].

[41] GitHub. 2022. GitHub - NoelDeMartin/solid-focus: Solid Task Manager. [online] Available at: <https://github.com/NoelDeMartin/solid-focus> [Accessed 17 January 2022].

[42] GitHub. 2022. GitHub - NoelDeMartin/media-kraken: Track your movies with Media Kraken.. [online] Available at: <https://github.com/NoelDeMartin/media-kraken> [Accessed 17 January 2022].

[43] Solidproject.org. 2022. Getting Started · Solid. [online] Available at: <https://solidproject.org/developers/tutorials/first-app> [Accessed 17 January 2022].

[44] GitHub. 2022. GitHub - inrupt/solid-client-js: Library for accessing data and managing permissions on data stored in a Solid Pod. [online] Available at: <https://github.com/inrupt/solid-client-js> [Accessed 17 January 2022].

[45] Docs.inrupt.com. 2022. Inrupt JavaScript Client Libraries — Inrupt JavaScript Client Libraries. [online] Available at: <https://docs.inrupt.com/developer-tools/javascript/client-libraries/> [Accessed 17 January 2022].

[46] Docs.inrupt.com. 2022. Module: access/universal — Inrupt solid-client API Documentation. [online] Available at: <https://docs.inrupt.com/developer-tools/api/javascript/solid-client/modules/access_universal.html#getagentaccessall> [Accessed 17 January 2022].

[47] Stack Overflow. 2022. Stack Overflow - Where Developers Learn, Share, & Build Careers. [online] Available at: <https://stackoverflow.com/> [Accessed 18 January 2022].

[48] Codeproject.com. 2022. CodeProject - For those who code. [online] Available at: <https://www.codeproject.com/> [Accessed 18 January 2022].

[49] Teamgantt.com. 2022. Online Gantt Chart Maker - Try for Free! | TeamGantt. [online] Available at: <https://www.teamgantt.com/> [Accessed 19 January 2022].

[50] Ippon | IT Consulting from Discovery to Delivery | Advanced AWS Partner. 2022. Beginning Decentralized Identity Applications with Solid. [online] Available at: <https://blog.ippon.tech/beginning-decentralized-identity-applications-with-solid/> [Accessed 19 January 2022].

[51] Enisa.europa.eu. 2022. [online] Available at: <https://www.enisa.europa.eu/> [Accessed 7 March 2022].

[52] Solidproject.org. 2022. Get a Pod · Solid. [online] Available at: <https://solidproject.org/users/get-a-pod> [Accessed 6 March 2022].

[53] DiNucci, D., 1999. Fragmented Future. [online] Darcyd.com. Available at: <http://darcyd.com/fragmented_future.pdf> [Accessed 6 March 2022].

[54] W3.org. n.d. Tim Berners-Lee. [online] Available at: <https://www.w3.org/People/Berners-Lee/> [Accessed 6 March 2022].

[55] Berners-Lee, T., 2007. Hearing on the "Digital Future of the United States: Part I -- The Future of the World Wide Web". [online] Dig.csail.mit.edu. Available at: <http://dig.csail.mit.edu/2007/03/01-ushouse-future-of-the-web.html> [Accessed 7 March 2022].

[56] Dynamo, R., 2022. Reproducing the Solid Project First App demo. [online] Dynamorando.com. Available at: <https://dynamorando.com/blog/solidprojectfirstapp/> [Accessed 8 March 2022].

[57] GitHub. 2022. GitHub - phochste/Web-Solid-Auth: A Perl Solid authentication tool (OIDC). [online] Available at: <https://github.com/phochste/Web-Solid-Auth> [Accessed 8 March 2022].

[58] Solid.github.io. 2022. Solid Technical Reports. [online] Available at: <https://solid.github.io/specification/> [Accessed 8 March 2022].

[59] Openid.net. 2022. OpenID Connect Core 1.0. [online] Available at: <https://openid.net/specs/openid-connect-core-1_0.html> [Accessed 8 March 2022].

[60] Solid.github.io. 2022. Solid Notifications Protocol. [online] Available at: <https://solid.github.io/notifications/protocol#sotd> [Accessed 10 March 2022].

[61] Solid.github.io. 2022. Solid Application Interoperability. [online] Available at: <https://solid.github.io/data-interoperability-panel/specification/> [Accessed 10 March 2022].

[62] W3.org. 2014. RDF 1.1 Primer. [online] Available at: <https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/> [Accessed 10 March 2022].

[63] W3.org. 2011. Turtle - Terse RDF Triple Language. [online] Available at: <https://www.w3.org/TeamSubmission/turtle/> [Accessed 10 March 2022].

[64] W3.org. 2020. JSON-LD 1.1. [online] Available at: <https://www.w3.org/TR/json-ld11/> [Accessed 12 March 2022].

[65] Eventbrite. 2021. Solid Project. [online] Available at: <https://www.eventbrite.co.uk/o/solid-project-30026804546> [Accessed 12 March 2022].

[66] GitHub. 2020. Community Solid Server: an open and modular implementation of the Solid specifications. [online] Available at: <https://github.com/CommunitySolidServer/CommunitySolidServer> [Accessed 12 March 2022].

[67] Podpro.dev. 2022. PodPro. [online] Available at: <https://podpro.dev/> [Accessed 12 March 2022].

[68] Elixir. 2012. elixir-lang.github.com. [online] Available at: <https://elixir-lang.org/> [Accessed 14 March 2022].

[69] Rdf-elixir.dev. 2021. RDF on Elixir. [online] Available at: <https://rdf-elixir.dev/> [Accessed 14 March 2022].

[70] Janeirodigital.com. 2009. Software to Connect The Decentralized Web | Janeiro Digital. [online] Available at: <https://www.janeirodigital.com/> [Accessed 14 March 2022].

[71] Janeirodigital.com. 2017. XFORM | Janeiro Digital. [online] Available at: <https://www.janeirodigital.com/xform/> [Accessed 14 March 2022].

[72] The NHS. [online] Available at: <https://www.nhs.uk/> [Accessed 14 March 2022].

[73] Vimeo. 2020. NHS Personal Health Stores by Janeiro Digital. [online] Available at: <https://vimeo.com/471703581> [Accessed 14 March 2022].

[74] GitHub. 2020. Solid-Health: A proof-of-concept mobile app for decentralized health record management. [online] Available at: <https://github.com/jasonpaulos/solid-health> [Accessed 15 March 2022].

[75] React Native · Learn once, write anywhere. [online] Available at: <https://reactnative.dev/> [Accessed 15 March 2022].

[76] Google Developers. 2021. Google Developers. [online] Available at: <https://developers.google.com/fit/android> [Accessed 15 March 2022].

[77] Paulos, J., 2020. Investigating Decentralized Management of Health and Fitness Data. MSc. MASSACHUSETTS INSTITUTE OF TECHNOLOGY. [online] Available at: < https://dspace.mit.edu/bitstream/handle/1721.1/127459/1192966772-MIT.pdf?sequence=1&isAllowed=y>

[78] Verborgh, R., 2017. Paradigm shifts for the decentralized Web. [online] Ruben.verborgh.org. Available at: <https://ruben.verborgh.org/blog/2017/12/20/paradigm-shifts-for-the-decentralized-web/> [Accessed 18 March 2022].

[79] Bingham, J., 2019. How The Decentralized Web Will Drive Innovation In The Healthcare Industry | Janeiro Digital. [online] Janeirodigital.com. Available at: <https://www.janeirodigital.com/blog/decentralized-web-drives-innovation-in-healthcare/> [Accessed 18 March 2022].

[80] CCPC Consumers. 2019. Phishing - CCPC Consumers. [online] Available at: <https://www.ccpc.ie/consumers/money/scams/phishing/?gclid=1ea0ebcaa9151e81967e89f984dfc61b&gclsrc=3p.ds&&utm_source=bing&utm_medium=cpc&utm_campaign=Money%20%7C%20Scams%7C%20Levy&utm_term=%2Bphishing&utm_content=Phishing> [Accessed 18 March 2022].

[81] Cisa.gov. 2021. Conti Ransomware | CISA. [online] Available at: <https://www.cisa.gov/uscert/ncas/alerts/aa21-265a> [Accessed 18 March 2022].

[82] Bava, M., Cacciari, D., Sossa, E., Zotti, D. and Zangrando, R., 2009. Information Security Risk Assessment in Healthcare: The Experience of an Italian Paediatric Hospital. 2009 First International Conference on Computational Intelligence, Communication Systems and Networks,.

[83] Y. He and C. W. Johnson, "Generic security cases for information system security in healthcare systems," 7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012, 2012, pp. 1-6.

[84] J. Kwon and M. E. Johnson, "Healthcare Security Strategies for Regulatory Compliance and Data Security," 2013 46th Hawaii International Conference on System Sciences, 2013, pp. 3972-3981.

[85] Solid Community Forum. n.d. Applications. [online] Available at: <https://forum.solidproject.org/c/applications/41> [Accessed 22 March 2022].

[86] Docs.inrupt.com. n.d. Inrupt JavaScript Client Libraries — Inrupt JavaScript Client Libraries. [online] Available at: <https://docs.inrupt.com/developer-tools/javascript/client-libraries/#getting-started> [Accessed 22 March 2022].

[87] webpack. n.d. webpack. [online] Available at: <https://webpack.js.org/> [Accessed 22 March 2022].

[88] Parceljs.org. n.d. Parcel – The zero configuration build tool for the web.. [online] Available at: <https://parceljs.org/> [Accessed 22 March 2022].

[89] Docs.inrupt.com. n.d. Using the Libraries — Inrupt JavaScript Client Libraries. [online] Available at: <https://docs.inrupt.com/developer-tools/javascript/client-libraries/using-libraries/> [Accessed 22 March 2022].

[90] Docs.inrupt.com. n.d. solid-client API — Inrupt solid-client API Documentation. [online] Available at: <https://docs.inrupt.com/developer-tools/api/javascript/solid-client/> [Accessed 22 March 2022].

[91] Docs.inrupt.com. n.d. Glossary — Inrupt JavaScript Client Libraries. [online] Available at: <https://docs.inrupt.com/developer-tools/javascript/client-libraries/reference/glossary/> [Accessed 23 March 2022].

[92] Abowd, G. Towards a Better Understanding of Context and Context-Awareness. In Internation Symposium on Handheld and Ubiquitous Computing, Proceedings of the First International Symposium, HUC'99 Karlsruhe, Germany, 27–29 September 1999; Gellersen, H.-W., Ed.; Springer: Berlin/Heidelberg, Germany, 1999; ISBN 978-3-540-48157-7

[93] Developer.mozilla.org. n.d. SPA (Single-page application) - MDN Web Docs Glossary: Definitions of Web-related terms | MDN. [online] Available at: <https://developer.mozilla.org/en-US/docs/Glossary/SPA> [Accessed 24 March 2022].

[94] M. Jones; J. Bradley; N. Sakimura. JSON Web Token (JWT). May 2015. Proposed Standard. URL: https://www.rfc-editor.org/rfc/rfc7519

[95] Schema.org. n.d. Schema.org - Schema.org. [online] Available at: <https://schema.org/> [Accessed 24 March 2022].

[96] Bioportal.bioontology.org. n.d. NCBO BioPortal. [online] Available at: <https://bioportal.bioontology.org/> [Accessed 23 March 2022].

[97] Docs.inrupt.com. n.d. Manage Access to Data — Inrupt JavaScript Client Libraries. [online] Available at: <https://docs.inrupt.com/developer-tools/javascript/client-libraries/tutorial/manage-access/> [Accessed 25 March 2022].

[98] Solidproject.org. n.d. Web Access Control. [online] Available at: <https://solidproject.org/TR/wac> [Accessed 26 March 2022].

[99] Solid.github.io. n.d. Access Control Policy (ACP). [online] Available at: <https://solid.github.io/authorization-panel/acp-specification/> [Accessed 28 March 2022].

[100] W3schools.com. n.d. How To Create a Dropdown Menu With CSS and JavaScript. [online] Available at: <https://www.w3schools.com/howto/howto_js_dropdown.asp> [Accessed 28 March 2022].

[101] Beaumont.ie. n.d. Beaumont Hospital - Department Listing. [online] Available at: <http://www.beaumont.ie/departments> [Accessed 29 March 2022].

[102] Tutorialzine. n.d. Create inline help tips for your site with a bit of CSS. [online] Available at: <https://tutorialzine.com/2014/07/css-inline-help-tips> [Accessed 29 March 2022].

[103] W3.org. n.d. Semantic Web Standards. [online] Available at: <https://www.w3.org/2001/sw/wiki/Main_Page> [Accessed 29 March 2022].

[104] W3.org. n.d. Ontologies - W3C. [online] Available at: <https://www.w3.org/standards/semanticweb/ontology> [Accessed 29 March 2022].

[105] McCrae, J., n.d. The Linked Open Data Cloud. [online] Lod-cloud.net. Available at: <https://lod-cloud.net/> [Accessed 29 March 2022].

[106] W3.org. n.d. SPARQL Query Language for RDF. [online] Available at: <https://www.w3.org/TR/rdf-sparql-query/> [Accessed 29 March 2022].

[107] Datatracker.ietf.org. n.d. RFC 7230 - Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. [online] Available at: <https://datatracker.ietf.org/doc/html/rfc7230#section-2.7.1> [Accessed 29 March 2022].

[108] Verborgh, R., 2019. How we will regain control of our personal data. [online] Medium. Available at: <https://towardsdatascience.com/ruben-verborgh-on-data-privacy-accf91d280c9> [Accessed 29 March 2022].

[109] Hl7.org. n.d. Rdf - FHIR v4.0.1. [online] Available at: <https://www.hl7.org/fhir/rdf.html> [Accessed 29 March 2022].

[110] N. Nchinda, A. Cameron, K. Retzepi and A. Lippman, "MedRec: A Network for Personal Information Distribution," 2019 International Conference on Computing, Networking and Communications (ICNC), 2019, pp. 637-641, doi: 10.1109/ICCNC.2019.8685631.

[111] Medicalchain. n.d. Home. [online] Available at: <https://medicalchain.com/en/> [Accessed 29 March 2022].

[112] Medicalchain.com. n.d. Whitepaper 2.1 [online] Available at: <https://medicalchain.com/Medicalchain-Whitepaper-EN.pdf> [Accessed 30 March 2022].

[113] A. Kumar, R. Krishnamurthi, A. Nayyar, K. Sharma, V. Grover and E. Hossain, "A Novel Smart Healthcare Design, Simulation, and Implementation Using Healthcare 4.0 Processes," in IEEE Access, vol. 8, pp. 118433-118471, 2020, DOI: 10.1109/ACCESS.2020.3004790.

[114] Starburst. 2022. The 2022 State of Data & What's Next | Starburst. [online] Available at: <https://www.starburst.io/info/the-2022-state-of-data-whats-next/> [Accessed 30 March 2022].

[115] Education, IBM., n.d. object-storage. [online] Ibm.com. Available at: <https://www.ibm.com/cloud/learn/object-storage> [Accessed 30 March 2022].

[116] LinkedIn Learning. n.d. JavaScript: Enhancing the DOM Online Class | LinkedIn Learning, formerly Lynda.com. [online] Available at: <https://www.linkedin.com/learning/javascript-enhancing-the-dom> [Accessed 31 March 2022].

[117] Tutorialspoint.com. n.d. UML - Activity Diagrams. [online] Available at: <https://www.tutorialspoint.com/uml/uml_activity_diagram.htm> [Accessed 31 March 2022].

[118] Solid Community Forum. n.d. Local First / Offline and Solid Login. [online] Available at: <https://forum.solidproject.org/t/local-first-offline-and-solid-login/4896/19> [Accessed 31 March 2022].

[119] GitHub Pages. n.d. GitHub Pages. [online] Available at: <https://pages.github.com/> [Accessed 31 March 2022].

[120] Medium. n.d. How to deploy Node.js app on AWS with Github. [online] Available at: <https://sumantmishra.medium.com/how-to-deploy-node-js-app-on-aws-with-github-db99758294f1> [Accessed 31 March 2022].

[121] Gitter.im. n.d. solid/app-development. [online] Available at: <https://gitter.im/solid/app-development> [Accessed 31 March 2022].

[122] Vercel n.d. Vercel. [online] Available at: <https://vercel.com/> [Accessed 02 April 2022].

[123] Enterprise Solid Server, Inrupt.com. n.d. [online] Available at: <https://inrupt.com/products/enterprise-solid-server/> [Accessed 02 April 2022].

[124] App, S., n.d. Dylan Storey / Solid Health App. [online] GitLab. Available at: <https://gitlab.com/storeydy/solid-health-app> [Accessed 02 April 2022].

[125] Tutorialspoint.com. n.d. UML - Use Case Diagrams. [online] Available at: <https://www.tutorialspoint.com/uml/uml_use_case_diagram.htm> [Accessed 02 April 2022].

[126] Jekyll • Simple, blog-aware, static sites. n.d. GitHub Pages. [online] Available at: <https://jekyllrb.com/docs/github-pages/> [Accessed 02 April 2022].

[127] W3.org. n.d. The W3C Linked Data Platform (LDP) Vocabulary. [online] Available at: <https://www.w3.org/ns/ldp#> [Accessed 02 April 2022].

[128] Oecd.org. 2019. Making Decentralisation Work: A Handbook For Policy-Makers. [online] Available at: <https://www.oecd.org/cfe/Policy%20highlights_decentralisation-Final.pdf> [Accessed 02 April 2022].

[129] Buterin, V., 2017. The Meaning of Decentralization. [online] Medium. Available at: <https://medium.com/@VitalikButerin/the-meaning-of-decentralization-a0c92b76a274> [Accessed 02 April 2022].

[130] O'Flaherty, K., 2018. Google+ Security Bug -- What Happened, Who Was Impacted And How To Delete Your Account. [online] Forbes. Available at: <https://www.forbes.com/sites/kateoflahertyuk/2018/10/09/google-plus-breach-what-happened-who-was-impacted-and-how-to-delete-your-account/> [Accessed 02 April 2022].

[131] Cadwalladr, C. and Graham-Harrison, E., 2018. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. The guardian, 17, p.22

[132] Developer.mozilla.org. n.d. Regular expressions - JavaScript | MDN. [online] Available at: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions> [Accessed 02 April 2022].

[133] Construction, U., 2022. Solid Health Application. [online] Solid Community Forum. Available at: <https://forum.solidproject.org/t/solid-health-application/5145> [Accessed 02 April 2022].

[134] Solid Community Forum. 2022. Acl file is not there when creating new resource. [online] Available at: <https://forum.solidproject.org/t/acl-file-is-not-there-when-creating-new-resource/5031/3> [Accessed 02 April 2022].

[135] GitHub. n.d. GitHub - solid/data-interoperability-panel: Repository for the Solid Data Interoperability Panel. [online] Available at: <https://github.com/solid/data-interoperability-panel/#solid-application-interoperability> [Accessed 03 April 2022].

[136] Forum.solidproject.org. n.d. Solid Community Forum. [online] Available at: <https://forum.solidproject.org/badges> [Accessed 03 April 2022].

[137] Solid Community Forum. 2019. .NET, C#, DOTNETRDF for Solid?. [online] Available at: <https://forum.solidproject.org/t/net-c-dotnetrdf-for-solid/1687/7> [Accessed 03 April 2022].

Appendices

A1 – Original Application Requirements

User Requirements

User Requirement ID	Description
UR001	Allow users to log in with their pod provider credentials
UR002	Allow Patients to register care with a new medical institution
UR003	Allow Patients to view the details of the medical institution under which they receive care
UR004	Allow Patients to see a list of all the healthcare departments under which they have an upcoming/past appointment
UR005	Allow Patients and Doctors to see a list of all personal medical records stored in their pod
UR006	Allow Patients and Doctors to view the list of users that have access to any part of their medical records, and the authorization level that these users possess
UR007	Allow Patients to add/revoke access from a user to their medical records
UR008	Allow Hospital Administrators to record details of an upcoming appointment between a Doctor and a Patient
UR009	Allow Pharmacists to see a list of prescriptions relevant for a particular Patient
UR010	Allow Emergency Workers to see the current medication that a particular Patient is taking.
UR011	Allow Patients to grant access to an Insurer to view pertinent medical diagnoses dated within the past 5 years
UR012	Allow Insurers to view pertinent medical diagnoses involving a Patient dated within the past 5 years
UR013	Allow Doctors to add medical records to a Patient's pod
UR014	Allow Doctors to edit and delete existing medical records stored in a Patient's pod

Functional Requirements

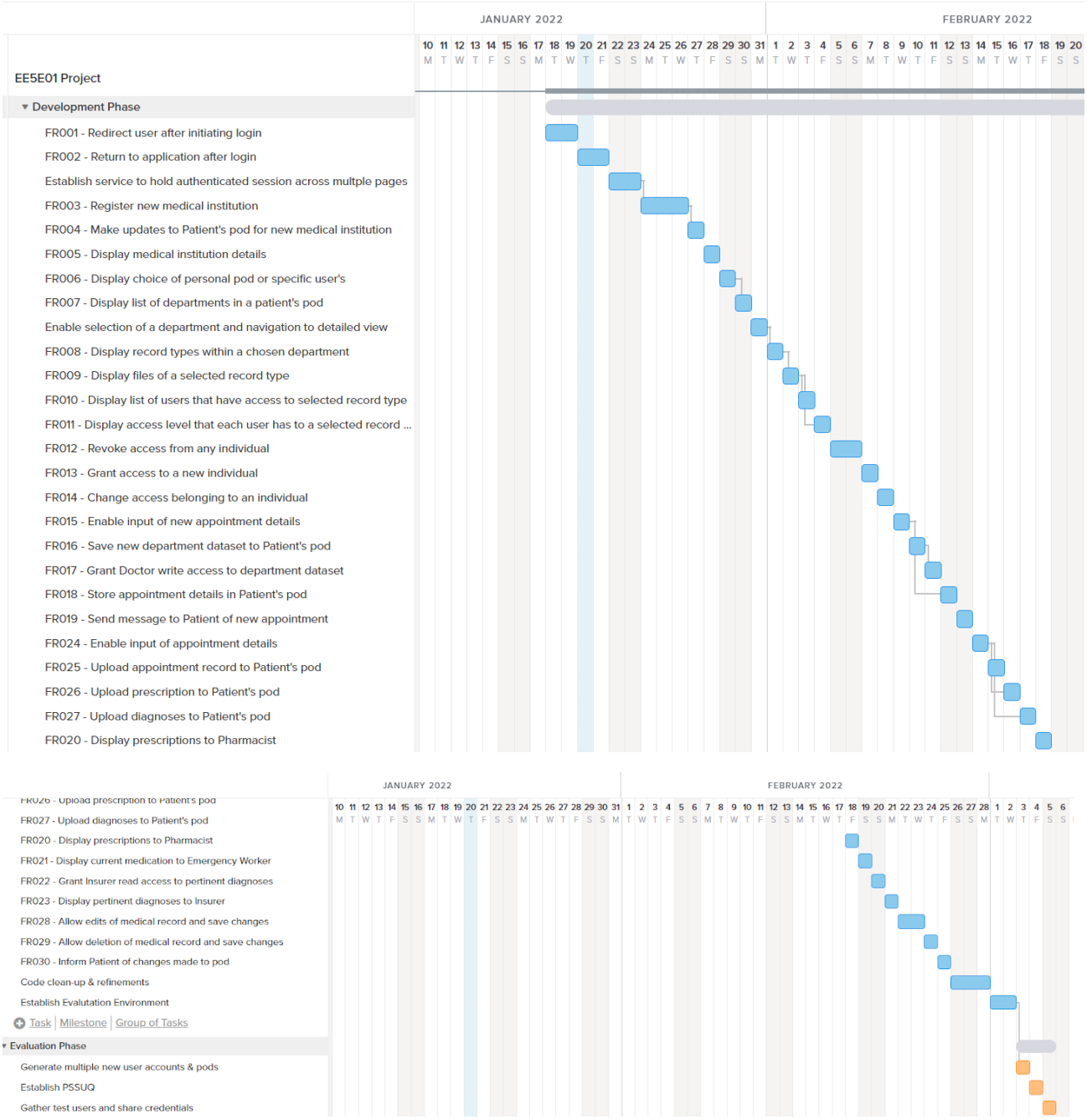
Functional Requirement ID	User Requirement ID	Description
FR001	UR001	Redirect user to pod provider after initiating log in process
FR002	UR001	Return to application with authenticated session after user logs in with pod provider
FR003	UR002	If the user is a Patient, display an option to register a new medical institution, entering the following information: • Name • Address • WebID of Hospital Administrator
FR004	UR002	If the user is Patient and saves details of a new medical institution, create a dataset in the Patient's pod if it does

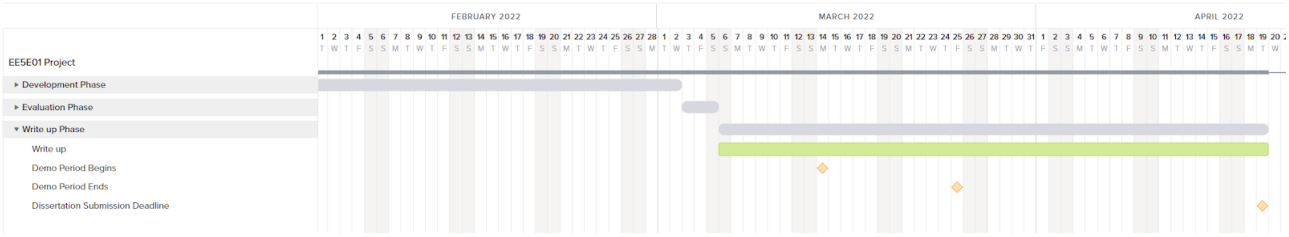
		not already exist. Grant ‘write’ and ‘append’ access to this dataset to the WebID of the Hospital Administrator.
FR005	UR003	If the user is a Patient, display the details of the medical institution under which they currently receive care
FR006	UR004	Display an option to view the contents of the user’s pod or an input box for a WebID to view the contents of a Patient’s pod
FR007	UR004	While viewing an individual’s pod, display a list of the departments that are held within the ‘Health Data’ dataset
FR008	UR005	After selecting a specific department from the list of departments in a Patient’s pod, display the four medical record types held within that department: • Records • Prescriptions • Diagnoses • Appointments
FR009	UR005	After selecting a record type within the selected department in a Patient’s pod, display the list of files in that container
FR010	UR006	Display the list of users that have access to the specific record type dataset within the selected department in a Patient’s pod
FR011	UR006	Display the access that each user has to a specific record type dataset within the selected department in a Patient’s pod
FR012	UR007	If the user is the owner of a particular dataset, allow them to revoke access from any user in the list of authorized users
FR013	UR007	If the user is the owner of a particular dataset, allow them to grant access to a new user by entering the WebID of the new user and having configurable checkboxes for the following access levels: • Read • Append • Write • Control
FR014	UR007	If the user is the owner of a particular dataset, allow them to change the access level that an authorized user has by using the configurable checkboxes
FR015	UR008	If the user is a Hospital Administrator viewing a Patient’s pod, allow them to add details of an upcoming appointment including the details: • Department • Time • WebID of Doctor conducting appointment
FR016	UR008	If a Hospital Administrator has recorded details of an upcoming appointment in a department that the Patient has not had before, create a new dataset in the ‘Health Data’ dataset of the Patient’s pod containing four sub-datasets: • Records

		• Prescriptions • Diagnoses • Appointments Grant ‘read’ access to the Patient’s Pharmacist Web ID to the ‘Prescriptions’ sub-dataset
FR017	UR008	If a Hospital Administrator has recorded details of an upcoming appointment with a Doctor that the Patient has not had before, grant the doctor ‘write’ access to the department dataset
FR018	UR008	If a Hospital Administrator has recorded details of an upcoming appointment, store the details in a new file in the ‘Appointments’ dataset within the department dataset
FR019	UR008	If a Hospital Administrator has recorded details of an upcoming appointment, send a message to the Patient’s pod to inform them of the upcoming appointment
FR020	UR009	If a Pharmacist is viewing the contents of a Patient’s pod, display all files in the ‘Prescriptions’ sub-dataset from all departments, ordered by date
FR021	UR010	If an Emergency Worker is viewing the contents of a Patient’s pod, display the most recent file in the ‘Prescriptions’ sub-dataset from all departments, ordered by date
FR022	UR011	If a Patient is viewing their pod, allow them to grant ‘read’ access to an entered Insurer’s WebID to all files in the ‘Diagnoses’ dataset within all department datasets that are pertinent to acquiring Life Insurance and dated within the past 5 years
FR023	UR012	If an Insurer is viewing a Patient’s pod, display the list of relevant diagnoses that they have been granted access to
FR024	UR013	If a Doctor is viewing a Patient’s pod, allow them to upload a new file to a department dataset including the following details: • Title • Description of appointment • Subsequent prescriptions from appointment, including WebID of Pharmacist to fill prescription • Subsequent diagnoses from appointment
FR025	UR013	If a Doctor has recorded details of an appointment, upload the Title and Description as a new file to the ‘Records’ dataset within the selected department
FR026	UR013	If a Doctor has recorded details of an appointment including a prescription, upload the prescription as a new file to the ‘Prescriptions’ dataset within the selected department
FR027	UR013	If a Doctor has recorded details of an appointment including diagnoses, upload the diagnoses as a new file(s) to the ‘Diagnoses’ dataset within the selected department
FR028	UR014	If a Doctor is viewing the medical records in a patient’s pod, allow them to make edits to a medical record

FR029	UR014	If a Doctor is viewing the medical records in a patient's pod, allow them to delete a medical record
FR030	UR014	If a Doctor has saved changes made to a particular dataset in a Patient's pod, send a message to the Patient's pod to inform them of details of the change

A2 – Application Gantt Chart





A3 – Development Diary

Development Diary – Solid Health App

26th October

- Set up GitLab repository
- Generated SSH key pair and stored public key on GitLab account
- Created Inrupt account and a new data pod, browsing contents with podbrowser application
    - Seems as though ‘Contacts’ can be added to a user account in podbrowser, where they can share files.
    - Can also have multiple pods under the same user

2nd November

- Installing node.js, Angular js -> Node.js downloaded from web, Angular JS installed from command line using ‘npm install -g @angular/cli@8.3.4’
- Styling done with SCSS over CSS as it seems to only have advantages
- Tried to go from the solid project tutorial of creating a first application with solid, was able to log in to pod provider, read and write data from the name of the pod owner, but the solidcomminity.net website seems to be very slow and timed out later in the day.

3rd November

- Tried to adapt the Inrupt JavaScript client library guide into an Angular application, but was only given a directory listing of raw JavaScript files when it ran
- Moved forward with the ‘first app’ tutorial on the solidproject.org website, successfully read a file stored in my data pod and printed the turtle triples to the console using a FileReader

14th December

- Able to create a new dataset in the Solid pod, which we would need to store private files, however the Inrupt guide for creating a dataset does not include anything to do with creating an ACL for a new dataset
    - Inrupt documents mention that in order to modify access to a resource in a solid pod, you must call ‘getSolidDatasetWithAcl’, which requires the dataset to have an existing ACL.
    - Tried to use the ‘createAcl’ function but it fails when trying to read the ACL url. Will try another approach

4th January

- Tried to update code for creating and reading an ACL for a dataset but still getting bugs – it says that the resource has no ACL. Also when viewing the dataset after having created an ACL for it, I cannot expand the object to view it on the solidcommunity.net portal.
    - Tried a few different approaches including functions found in the 'solid-client/access/universal' package, which is only available in version 1.5 of the solid-client package. Not sure how to get to this package number as it is not on node.
- Posed a question to the solid community forum for creating an ACL for a resource using the package, will give some time for replies and come back to it tomorrow.

5th January

- Able to set the access for a resource using the 'setAgentResourceAccess' and the 'setAgentDefaultAccess' functions. This proves the general premise of the application is achievable. Going to try to grant access to specific users through the application and check that only authorized users can view the resource.

6th January

- Able to grant access to a resource to an external user
- External user not able to get the resource with its ACL, but checking with the 'getAgentAccess' and the specific web ID of the user that has been granted permission shows that the user is authorized to perform the actions specified earlier
- Need to tidy the code up a bit and keep only what will be required to design the application around – most of the core functionality is possible, so the rest will likely just be designing around it.

7th January

- Able to upload a new text file to the dataset, of 'TextDigitalDocument' type from the Schema.org vocabulary. Would be ideal to allow a pdf upload here, but not exactly sure how that will work with adding "Things" to Solid Pod.
- Trying to allow users to delete file in a dataset but for some reason when specifying the URI for a file it deletes the file and the container that the file is saved in. Followed the guide exactly so unsure why this is happening.

8th January

- Got around the bug for deleting a file by retrieving the contents of the dataset first and then using the 'removeThing' function to take out the specified file URI from the dataset.
- Set up checkboxes for specifying the specific access permissions that should be granted to a designated user. Need to think carefully about the implications of granting permissions to a user in the event of a security breach.

9th January

- Verified that it's possible to write data to another users pod after they have been given explicit access
- Can read the contents of each file in a dataset
- Added the author to each file that is uploaded to a dataset. This is done by setting the full name of each user in the solidcommunity.net pod browser and reading this string in the application.

- Having an issue with adding the date of creation to an uploaded file.

10th January

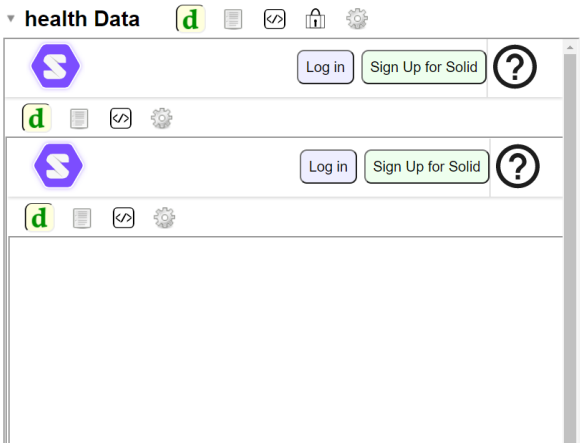
- Got around the issue of adding dates to new files by saving the date as a string as opposed to a date.
- Now that I know most of the desired functionality is possible with the technology, I developed a detailed set of user & functional requirements, alongside some flow diagrams for the tasks that a finished application in this area could offer.

17th January

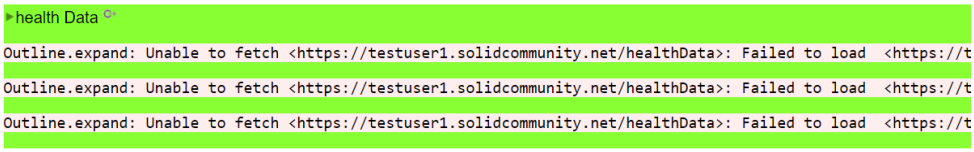
- No notable development updates this past week, I've been working on a set of use cases and requirements for the application as well as the progress interim report which is due on the 19th of January. I have also developed a Gantt chart for the remaining development tasks as I see them, to act as an indication of the work left to complete and the order in which tasks will be completed.

24th January

- Trying to enable multiple pages and routing between them using window.location.replace(<pageName>) but having some trouble after redirecting as the html is not updating – tried a few different approaches but couldn't get the content on screen to change. I decided to just keep it as a SPA and update the html content of each page accordingly.
- Got code working for multiple steps of registering a new medical institution:
    - Checking to see if the health data dataset exists in the individuals pod. Displays an option for adding details of a new one if it doesn't exist
    - When clicking button to add a new one, 3 input boxes are displayed for the name, address and WebID of the medical institution administrator
    - Creates a new dataset and a file in the health data dataset to store these details, and granting access to the institution administrator
    - Got some error logging and different paths covered, but there's still a lot more to do:
        - Cancel changes button
        - Edit details of the currently registered medical institution
        - More messages to the error to inform them of status of save/update operations
- Need to be really careful when updating the access for the dataset, as I've locked myself out from some datasets while developing -> in that I get a 403 unauthorized error when trying to either view the dataset in the solidcommunity.net portal, or delete it from the application side.
    - This might cause a large problem when using 'production' type user accounts, and I will need to make sure there's minimal chance of this happening from any combinations of user inputs.



This is the view from the solidcommunity.net portal of the dataset that I locked myself out of. Clicking 'Log in' or 'Sign Up for Solid' brings back to the same page.



- Tomorrow going to try develop from the perspective of the institution administrator by allowing the access of an individual's health data records, and uploading the details of a new appointment to their pod.

25th January

- Some minor styling changes
- Set up the option to either view the currently signed in user's pod or a specified user
    - Need to finish this with full user alerts of status – who has health records, who has access etc.
    - Need to display the currently accessed pod somewhere in the header, and allow an option for 'resetting', i.e., changing the currently accessed pod to another without having to restart the application

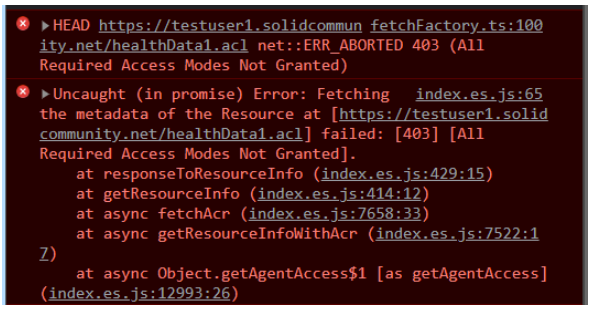
25th January (later)

- More changes to display – now no actions involving an individual's pod are visible until the user verifies that they have access to the pod.
- Added the option for resetting the pod session, to return to the point just after logging in
    - Will need to add in some global logout functionality too
- Currently changing screen content by setting the 'display' property of various divs to 'block' or 'none', as I suspect that there will only be a handful of divs in the finished application – although this may still be bad practice it's a temporary workaround until I can get proper routing working in JavaScript.
- Added alert messages to the user when either: they are unauthenticated to view a pod, there is no pod created, or they have not entered a valid URI/WebID.
- One thing I have noticed so far is that the naming convention of all files and datasets must be uniform in order to pass it as a URI for 'GET' requests. Currently setting it so that the health record dataset is called '<WebID>/healthData1' (had been called just 'healthData' but as mentioned yesterday the permissions got set incorrectly and I cannot either retrieve or delete the dataset at that URI.

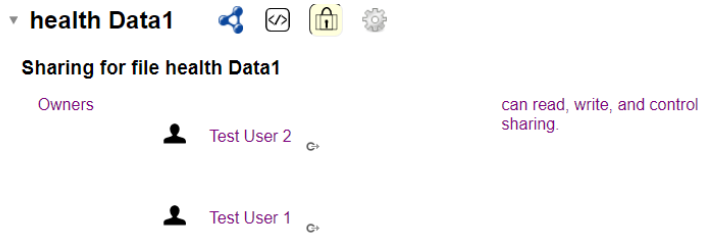
- As long as the application code determines all the URIs and retrieves them using the same system it shouldn't be a problem, but need to be careful that any URIs generated from user input are formatted uniformly.

26th January

- The 'getAgentAccess' function in the 'access' module is not working as it should. When calling the function to determine the access that the signed in user has, it gives the following error:

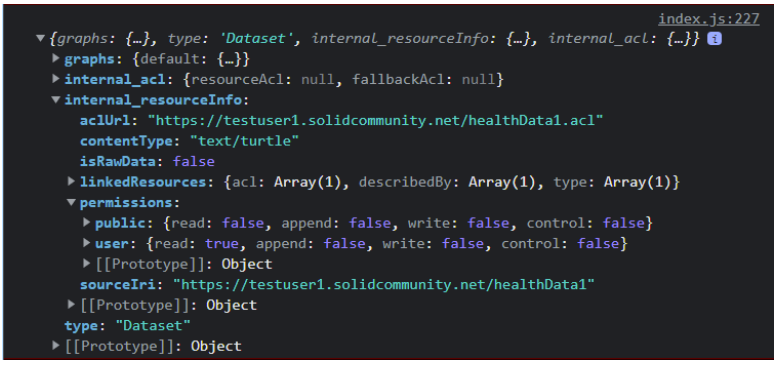


This doesn't exactly make sense because I set that user (Test User 2) to be an owner of the dataset (same access level as the pod owner, Test User 1)



Even when logging in as Test User 1 and making the same function call, it retrieves the access that Test User 2 has:

However, after changing the access that Test User 2 has from an 'Owner' to a 'Viewer' through the solidcommunity.net portal (leaving them only with 'read' access to the dataset), we can read the access that the user has, by calling the 'getSolidDatasetWithAcl'. Logging the result of the result shows us the access that the current user has:



This means that we can view the access that the current user has by checking the value of <datasetObject>.internal_resourceInfo.permissions.user

- Noticed another problem with granting access to individuals via the solidcommunity.net portal – When you add a user as an 'owner' of a resource, meaning they should have 'read', 'write', 'append' and 'control' access, they are actually missing out on 'control' as I have noticed from reading permissions using the above method.
    - That means that the only way to grant 'control' permission someone is explicitly in code via the 'setAgentAccess' and including that user in the function arguments.
    - This is currently done on creation of the health data dataset, but it is inconvenient to go through that whole process again when someone needs to be added as an owner to the dataset, i.e. a hospital administrator.
- Tried to get a dropdown select working for adding a new appointment and selecting the department of the appointment using react, but had some trouble finding a component that works.
    - Tried this guide: https://blog.campvanilla.com/reactjs-dropdown-menus-b6e06ae3a8fe, while adding React to the project using the official documentation: https://reactjs.org/docs/add-react-to-a-website.html#add-react-in-one-minute . Also tried another guide https://mui.com/components/autocomplete/ but this didn't work either – might just try a very basic dropdown using pure JS/HTML.

28th January

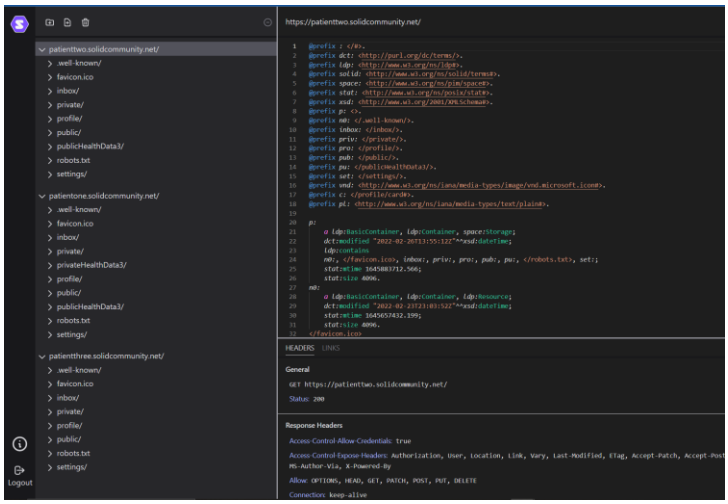
- Got dropdown working with basic JavaScript and CSS, using modified code from the example: https://www.w3schools.com/howto/howto_js_dropdown.asp
- Started gathering SVGs that the application will use from: https://www.flaticon.com
- Got basic row/column classes working using the tutorial at: https://www.w3schools.com/howto/howto_css_three_columns.asp

30th January

- Working on saving new appointment details in a solid dataset, but have to change the way health data has been saved up until now. I thought that the 'saveSolidDatasetInContainer' (https://docs.inrupt.com/developer-tools/api/javascript/solid-client/modules/resource_solidDataset.html#savesoliddatasetincontainer) would allow the application to store datasets within other datasets, similar to a folder structure of a common file explorer, although when trying to use it I see that the dataset must be stored in an actual 'Container' as opposed to a 'Dataset' – although in the

documentation it describes the two almost as identical types. There's also no aspect of the tutorial that covers the basic set up of a container.

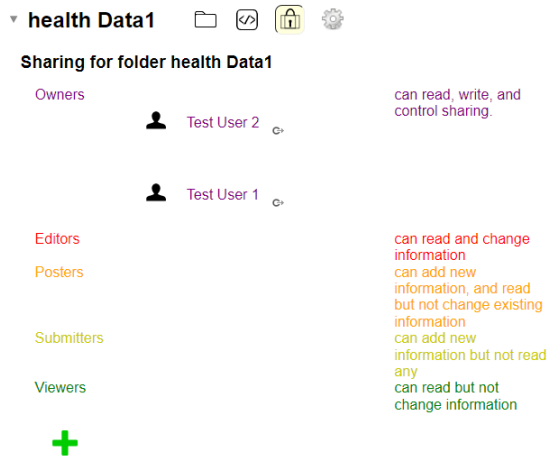
- Up until now I was planning on storing all medical records in a dataset called 'Health Data' stored in a user's pod (to keep access to medical records separate from any other data in the user's pod), but this will now have to change from a Dataset to a Container. Will need to re-factor registering a new medical institution accordingly.
- Noticed another small bug with the solidcommunity.net pod browser; you have to delete a container twice, even after refreshing the page.
    - Found another pod browser application on a post in the developer community (https://podpro.fly.dev/) that may be better for interacting with the pod, although everything is displayed in turtle syntax which makes it a bit more difficult to interpret



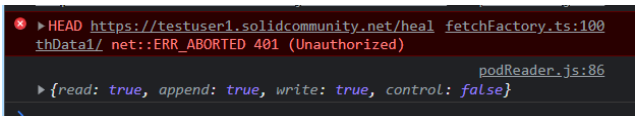
- Seems to also be another bug with setting the access for a Container as opposed to a Dataset. Setting the same permissions to a resource as I was doing for the dataset, the only difference is that the 'getResourceInfoWithAcl' has to be called instead of 'getSolidDatasetWithAcl', as the resource has changed from a dataset to a container. Setting the same access (all levels) to the pod owner.

```
const myDatasetWithAcl = await getResourceInfoWithAcl(healthDataDatasetUrl, { fetch: session.fetch })
const myDatasetsAcl = createAcl(myDatasetWithAcl)

console.log(myDatasetsAcl)
let updatedAcl = setAgentResourceAccess(
    myDatasetsAcl,
    session.info.webId,
    { read: true, append: true, write: true, control: true }
)
updatedAcl = setAgentResourceAccess(
    updatedAcl,
    institutionDetails.administrator,
    { read: true, append: true, write: true, control: true }
)
updatedAcl = setAgentDefaultAccess(
    updatedAcl,
    session.info.webId,
    { read: true, append: true, write: true, control: true }
)

try {
    await saveAclFor(myDatasetWithAcl, updatedAcl, { fetch: session.fetch })
}
catch (err) {
    console.log(err)
}
```

And looking at the container in the solidcommunity.net portal, it seems that both users have all access levels:



However, when calling the 'getResourceInfoWithAcl' function to read the access that the owner has, we can see that the owner doesn't have the control permission for some reason:



31st January

- Noticed a limitation to provisioning access to an individual; If an individual is granted 'control' access (meaning they can view who can access the resource and grant access to other users), there's no way to limit the maximum access that they can grant to other users.
    - For example in this use case, a Doctor will have to be given 'control' access to a dataset held within a Patient's pod, because they will have to be able to grant 'view' access to a designated Pharmacist who will need to be able to see any prescriptions that resulted from the appointment with the Doctor and the Patient. However, if the Doctor is given 'control' access they could theoretically give the Pharmacist any set of access permissions, not only 'view'. This can be prevented from the application layer, but it remains a slight flaw in provisioning access to a resource. It would be better if the pod owner could specify the maximum permission level that a user with 'control' access can grant to other users.

1st February:

- Not much development work done today, working on an ethics application form for the project, including participant consent forms and information sheets.
- Spent some time trying to refine the RDF class for an appointment file. Looked around for a health record ontology and found some research papers for concepts but none actually published on the web. Decided to use the schema.org 'Event' class as its type, with sub properties. It's not a perfect fit for a real world appointment but I can come back later to refine it.

2nd February:

- Got the appointment details now saved to the 'Appointments' sub-Dataset within the overall department dataset.

- Now returning the list of departments under which a patient receives care – next step is to select a sub-dataset of each department (Appointments, Diagnoses, Prescriptions or Records) and display a list of files within.
- Retrieving the list of files within a given dataset, e.g. <Department>/Appointments and extracting the key-value of properties from them.
- Need to just iterate through the list tomorrow and display the properties on screen.

3rd February:

- Generically displaying medical records based on key value pairs.
- Added a few 'return' buttons to return to previous state
- Added a user prompt to display when no records found of the chosen type in the chosen department.

6th February:

- Added a few more classes for reusable buttons and other small styling improvements
- Will need to pull the users departments in another place now, so made the function which assembles it generic for the 2 use cases. The only main difference is the HTML element to insert the department dropdown in, so it should be ok to reuse it multiple times with only small changes to the function

7th February

- Set up inserting a new medical record to a patient's pod, working with a general record first (as in notes from an appointment with a doctor, blood test result etc.)
- Trying to pick what fields will be important here but just went with the basic ones which I suspect will be included regardless (title, description, date etc.), other auto-generated fields will be included too such as author, creation date etc.
- Looking through the ontologies at schema.org, I found a new type which might be useful for all of these medical records (https://schema.org/MedicalWebPage). It's more aimed towards informative medical websites but so far it is the closest match to a medical record as it is used in the application.
- Spent some time refining other class definitions based on the different types and properties on schema.org

8th February

- Got some really good feedback from professionals in the health informatics space about the application proposal, mostly some additional use cases for the application and some related to the actual operation of the app including pod structure changes:
    - Apparently appointment data is usually kept isolated from other medical records, such as the ones that will be created in the application. I had planned on storing appointments as one of the 4 'datasets' within a healthcare department 'container', but this might have to be changed to having appointments stored in another location entirely. At the very least, I think I will have to not grant access for the other medical record types to the hospital administrator as I have been doing, and instead keep it only to those who will be writing or reading other medical record types. This will require a reasonable refactor to the code, but going to finish uploading medical records first.
    - Also someone mentioned it would be good to allow for more than one medical institution at a time (perhaps a private hospital and/or a GP practice). This

shouldn't be too hard and will just require an additional select to be made after accessing an individual's pod.

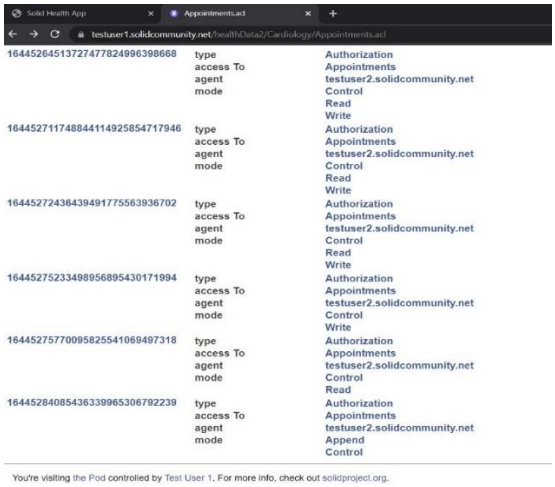
- Got general records, diagnoses and prescriptions saving to the pod, and able to view them within the 'access medical records' pane.
    - Including the option for specifying a pharmacist's WebID to be able to view a new prescription record.
- Developed a generic function to iterate through the key-value pairs of whatever it passed to it as an argument and insert it into the designated dataset. This means that one function can be used to write a new file dynamically to any dataset, and only smaller functions had to be made to construct the object to send to this function, and update the DOM based on the selected medical record type.
- Got logout functionality working from an application perspective

9th February

- Realised that logout isn't actually working as I thought it was. The authentication cookie is still stored in the browser memory, so even though the user is logged out and cannot perform any actions that require authentication, as soon as they press the login button the same user is signed in as before without having to enter credentials.
    - I tried to both forcibly clear the application memory and wipe the cookies, but am unable to do so without actually going into the chrome developer console and remove it that way, which is not really feasible from an application development perspective.
    - I found a topic on the solid community forum discussing this issue https://forum.solidproject.org/t/local-first-offline-and-solid-login/4896/24 and it seems that no solution has been reached, but I reached out to see if any progress has been made.
- Displaying the access that any user has for a particular dataset in checkboxes. Need to find a way to update the access based on user click events.
- Worked on the DOM of the displayed access to a particular dataset, such that changing the value of the checkboxes displays a button beside the relevant user with access, and passes the index of that element to a function.
    - Need to just extract the current access that the user at that index has and make updates to the users access based on the current value of the checkboxes at the time the button is clicked.

10th February

- Working on function to update access that a user has to a dataset based on current value of checkboxes, imported lodash for object comparison to display a message to the user if no change from previous access is detected
- Having trouble actually updating an agent's access, it doesn't seem to update an existing agent's access but instead it inserts a new agent below with that access level.



I posed a question to the community forum so hopefully somebody might know what's going on.

11th February:

- Set up the form to add access to a new user
- Various formatting and general bug fixes
- Got a response from the community on the issue with updating access, will try to implement a fix tomorrow

12th February

- Got both adding and updating access working, with help of the fix provided by the community.
- Some styling changes and new CSS classes that can be used throughout the app.

13th February

- Working on the initiate insurance request feature, got the display part working just need to filter relevant files and grant the insurer access to them
- Added some more departments to the dropdown, based on the list of departments in Beaumont Hospital

14th February

- Able to grant access based on criteria (dated within 5 years), although you cannot grant access to a particular 'Thing' (analogous to a file) inside a dataset, but instead you can only grant access to the dataset as a whole.
    - This means that if I continue with the current structure, whenever there is a diagnosis within a pertinent department that's less than 5 years old, the insurer would be able to see all diagnoses in that department, instead of just the ones that match the search criteria
    - Workaround for this is to add relevant diagnoses to a separate dataset and then grant the insurer access to this whole dataset.
- The 'access/universal' section of the solid-client package disappeared off the documentation webpage today for several hours, presumably to make updates to it. Quite frustrating as a lot of function usage is now hidden.
- Got the functionality working for initiating an insurance process and provisioning access to the insurer, by creating a separate dataset and inserting the files into it that match the search criteria.

- Need to just display the relevant files in the ‘access medical records’ section.

15th February

- Got insurance diagnoses displaying now, had a bit of trouble having two buttons in the one form but fixed it by changing one button to an input with button type.
- Some further formatting of buttons and error handling
- Had a good meeting with developer of the podpro application (https://podpro.fly.dev/), giving feedback on the tool and talking about what it’s like working with Solid.
- Set up provisioning of multiple types of medical institutions; Public, Private or GP. Saving these at separate containers in the user’s pod. Will require a bit of code refactoring to search for different URIs based on the type of medical institution they are currently accessing.
- Changed the WebID of the institution administrator to be optional. Access is not provisioned to anyone else if the field is left blank.

16th February

- Allowed the selection for accessing a different type of health data within a patient’s pod. Doing a slight code factor of expected URIs based on the selection that was made.
- Changing a lot of event handlers from ‘submit’ of a form containing a singular button to a click event handler on the button itself.
- Found another bug when granting access to a dataset. If you try to grant access to a dataset URL that has an extra ‘/’, that dataset becomes locked out and you cannot read from it or delete it.
    - For example when adding a new appointment, you need to make sure that the appointment doctor has permission to view the ‘Info’ dataset of the selected health data container. Although the dataset was already existing and the pod owner was able to view/update it, I tried to grant ‘view’ access to the doctor to this dataset but accidently specified the url as ‘<healthDataContainer> // Info’ with two ‘/’s instead of just one. Instead of throwing an error it locked the entire dataset and subsequently rendered the whole health data container to be useless, as I cannot delete the container or read from one of the crucial datasets.
- Got the registering appointments / creating new departments / viewing records / managing access working again after refactor. Need to do the upload records piece tomorrow but that shouldn’t take long.

17th February

- Got the code refactor done for uploading all new record types, code is now back fully working after allowing the selection of multiple types of different health data
- Got a good tooltip from a tutorial at (https://tutorialzine.com/2014/07/css-inline-help-tips), will display information to the user at several stages of each use case process
- Going to set up the form for registering a new medical institution when one already exists. Need to make sure the user is aware of what ones they currently have, as it will completely wipe existing data if they have one of that type
- Found another slight bug when trying to delete the health data container to replace it with another; it does not delete all contained resources. This might be because the URI of the container hasn’t changed, but I would have expected that deleting a container should delete all of the contained resources within it.

    - Not too much of a problem as I had already built the functionality to retrieve the list of departments within a container so just have to iterate through this list and delete each one explicitly.

17th February (later):

- Swapping between different health data types now working, as well as creating a new medical institution, which deletes all data currently held in the pod for that type.
- Displaying the hospital administrator on the home page now, need to be able to update this as they are the only individual (other than the owner) that is able to register details of new appointments and therefore create new containers for departments.

18th February:

- Allowing the edit of existing institution information without creating a new container (i.e. updating the name, address of administrator of the institution). This means revoking access from the existing administrator and granting it to the new one
- Found another slight bug with the 'setAgentResourceAccess' function when doing so;
    - When revoking access it works by specifying the 'controlRead' and 'controlWrite' properties, but when granting access and specifying these it doesn't work – need to just specify a value for 'control'.

21st February:

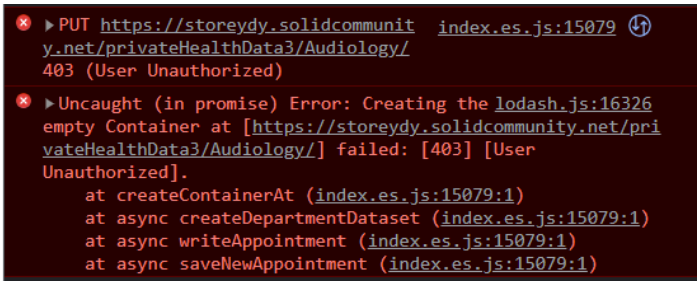
- Earlier bug with retrieving the agent access for a resource is starting to cause issues. When the app is loaded and the user specifies the pod that they wish to access, I need to disable certain actions from being completed based on the user's permissions.
    - For example, creating an appointment in a new department should be reserved for the pod owner or institution administrator, as this will create a new container in the currently accessed health data container along with 4 sub-datasets.
    - Initiating an insurance request and registering details of a new medical institution is reserved for only the pod owner
    - Editing details of the currently accessed health data institution is reserved for the pod owner or institution administrator
- Got this functionality working but since getAgentAccess isn't working as it should I'm checking the value of the hospital administrator that is stored in the 'Info' dataset within each health data container as opposed to reading from the acl. This is not ideal as this value could hypothetically be changed and cause someone to gain admin access
- Got functionality for a pod diagram working, including a list of all departments and datasets within them, along with the number of files saved in each dataset, if this is available to the signed in user.

22nd February:

- Found another issue with trying to create an appointment in a new department as the institution administrator (which creates a new container for the department and 4 datasets within for 'Appointments', 'Diagnoses', 'Prescriptions', 'Records'). Even when the administrator and the pod owner have the same access level on the overall health data container (owner status), the administrator cannot create an ACL for the resource, and this is reserved for only the pod owner.

- Means that only the pod owner can create an appointment in a new department

- Error log:
  - This message is shown when trying to initialise the new department:



  - Even though the user that tried to complete the operation (Test User 2) had all of the possible access on the parent container.

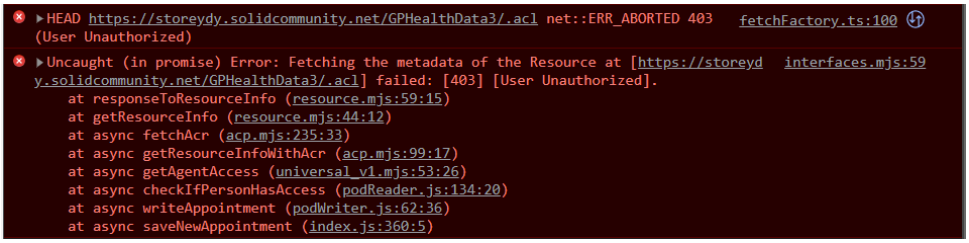
- Decided to remove some of the access from the specified doctor of an appointment:
  - Instead of having all of the access to the 4 inner datasets, they will only have the ‘control’ access for the Prescriptions dataset – as this is the only type of data that they will upload to the Patient’s pod where they have the option of granting ‘read’ access to a Pharmacist. They will have ‘read’ and ‘write’ access only for the Records and Diagnoses datasets, as these are the datasets that they will upload files to after their appointment with the Patient. They will only have ‘read’ access to the Appointments dataset, as they will not be recording details of a new appointment and this is down to the administrator or the patient themselves.
  - I thought of granting them the ‘control’ access to all datasets, in the event that they wanted to share a Patient’s health records with another doctor, but decided it would be better from a security perspective if the Patient themself is the only person that can grant access to records to other individuals.
- Worked on some changes to the process, trying to find a balance between ensuring the security of the data (i.e. not granting unnecessary access to people who don’t need it) and optimising the usefulness of the application (i.e. creating application processes that are similar to real-life and would serve use to all users – while trying to minimise the amount of work required by the patient themselves). Worked out a

flow of each application operation and the required permissions for each user type. Will implement it in code tomorrow.

23rd February:

- Made these changes to the process flow of creating a new appointment and examining the permissions in the solidcommunity.net portal, it seemed that everyone had the correct access rights;
    - Pod owner remained the owner of all created datasets
    - Institution Administrator is granted read, write access to the Appointments dataset so they can read and upload files of upcoming appointments to that department container.
      They are also granted control access to the Diagnoses, Prescriptions and Records datasets so that they can grant read, write access to the specified doctor when uploading an appointment later
      They had previously also been granted the exact same permissions as the pod owner when creating both the initial health data container and the first appointment within a department.
    - The specified doctor when entering the appointment details is granted read access to the Appointments dataset, read/write access to the Diagnoses and Records datasets, and read/write/control access to the Prescriptions dataset so they can upload a new prescription and specify the WebID to grant them read access to the prescription.

  This worked fine when uploading the initial appointment from the pod owner's account, however when logging in as the Institution Administrator and trying to upload a new appointment, the process fails when checking if the specified doctor has previously been granted access to the overall health data container, as it seems this check can only be made by the pod owner – despite both users being granted the exact same access.

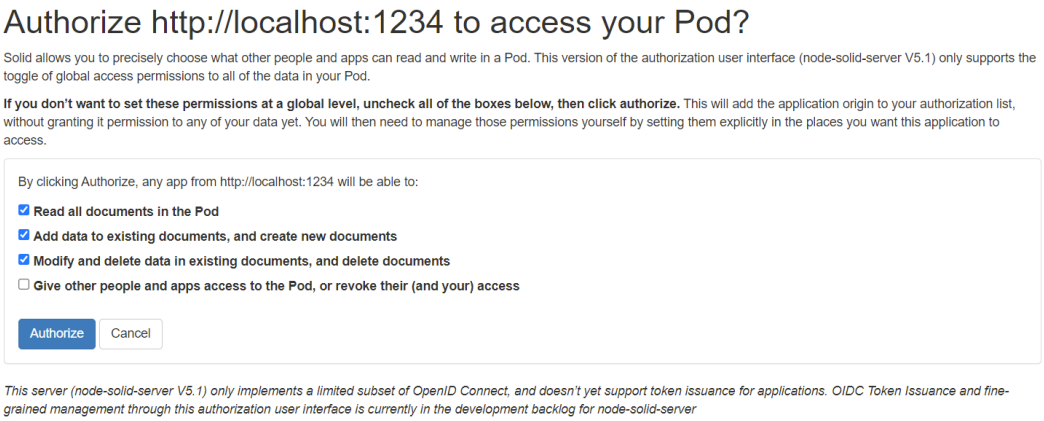


  This essentially means that only the pod owner (Patient) can upload details of a new appointment and the institution administrator role is somewhat redundant in the application.
  Can modify the upload function to add a 'prescription' option, to upload the raw file without granting child permissions to other users.

- Having the same error in the 'Manage access to selected dataset' functionality, it only works for the pod owner
- And in the upload prescription functionality, cannot grant permission to the pharmacist unless they are the pod owner

24th February:

- Found the fix for not being able to grant access. It was the initial dialog that is displayed when a user signs in to the application, which is by default set to not allow access to be changed to resources within the user's pod.



I had forgotten about this dialog as I had been working with a small number of debugging user pods, but it was only when I tried to restart the process with one of the new user pods created yesterday that I realised that must be the source of the unauthorized errors I was getting.

  - With this now working, I unmade the changes that I had done yesterday. This explains the comments made about 'getAgentAccess' and setting agent access when not signed in as the pod owner. The application should now work for all functionality as planned.

- Went through the main application code and removed log messages and added explanatory comments where deemed necessary.

25th February:

- Tried to host the application on Github Pages but was getting errors with node modules. Looked at a few possible solutions online relating to the Jekyll generator that GitHub Pages uses, but could not a working one. Also tried the 'gh-pages' npm package as an alternative solution but did not work either. Deciding to try finish implementation fully tomorrow and retry after the application is finished.

26th February:

- Some more bug fixes and UI improvements
- Have full process working now and successfully tested except for the requesting insurance feature – will tidy this up tomorrow and make final updates to pod diagram feature also.

28th February:

- Further test of all application features – everything seems to be working fully now as intended
- Cleaned up all comments and old unused code
- Added some definitions of application terms found at (https://docs.inrupt.com/developer-tools/javascript/client-libraries/reference/glossary/)
- Set up an ec2 instance on aws and running the application online but having trouble with the inrupt packages, using two tutorials (https://sumantmishra.medium.com/securely-ssh-into-aws-ec2-linux-instance-42ad8a322ac5, https://sumantmishra.medium.com/how-to-deploy-node-js-app-on-aws-with-github-db99758294f1). Already had to make some changes to get parcel running on the ec2 instance and importing the packages, but may have to make further changes to get the package to work.

1st March

- Tried with a new ec2 instance and running the application using webpack (as specified on the inrupt tutorial) as opposed to parcel (as specified on the solid project tutorial), but seemed to make no difference. I was able to get the application displayed and initiate the login process, but after the user authenticates the browser does not strip the authentication code from the URL, and the code fails at the callback section.
- Looked at some possible solutions involving the ec2 instance configuration but I'm not sure whether the fault is at the solid inrupt package or with the ec2 instance settings.
- Posed a question to the Solid developer chat early in the day but haven't got any responses.
- Saw a link to another method of authentication that uses a popup (https://solid.github.io/solid-auth-client/) , will try this tomorrow and see if it makes a difference.

A4 – Application Demonstration Presentation

Solid

A specification to allow people to store their data securely in decentralized data stores called Pods. (**P**ersonal **O**nline **D**ata **S**tores)

- Fully controlled by the owner
- Data and access rules completely separate from other pods

Pod owners can share access of their data with other people/apps, or revoke access as needed.

Motivation

Short-term:

- Proof-of-concept app for storing & interacting with medical records in Solid pods
- More control & transparency in who can access patient health data
- Immediate access to records for Patients

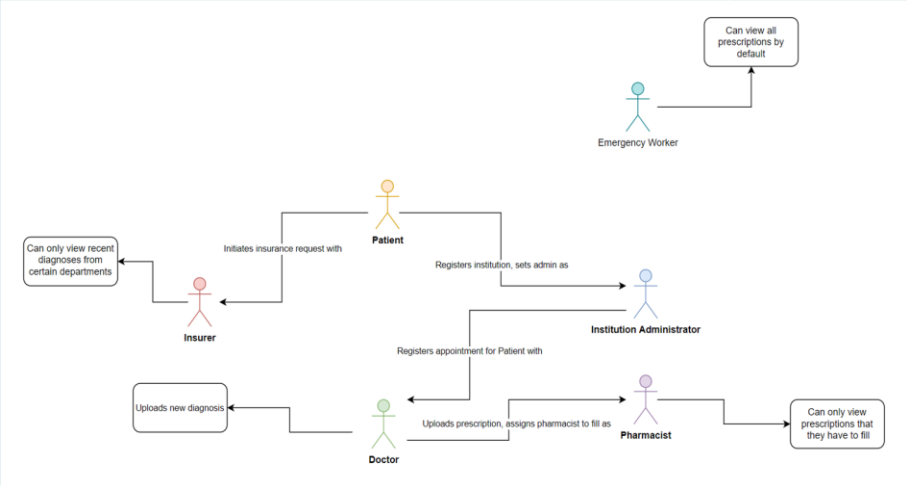
Long-term:

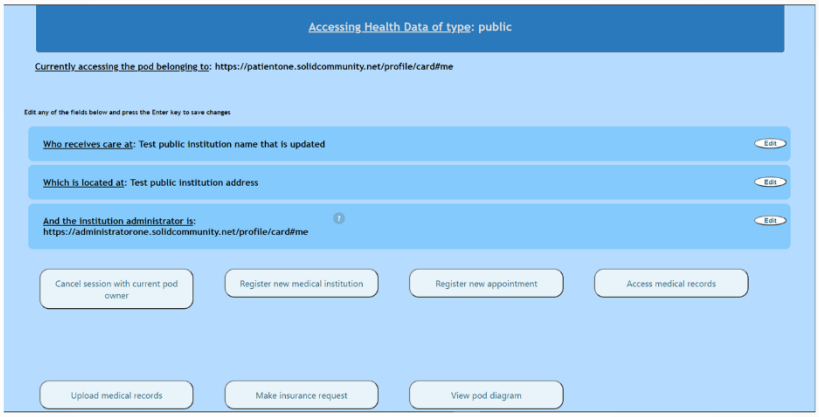
- Reduction of centralised health data
- Replace existing paper-based medical record approach

Users & Use Cases

	Login	Register Institution	Create Appointment	View Records	Upload Records	Manage Record Access	Request Insurance
Patient	✓	✓	✓	✓	✓	✓	✓
Administrator	✓		✓	✓	✓	✓	
Doctor	✓			✓	✓		
Pharmacist	✓			✓			
Emergency Worker	✓			✓			
Insurer	✓			✓			

Process flow





Application Home Page

Accomplishments

Solid

- Helped community with dev issues & literature
- Published new Solid app on the web
- Reusable code
  - Generic reading/writing
  - Grant access to subset of dataset based on conditions
  - App data diagram w.r.t. pod structure

Application

- Share access to health records in new way
- Allow patients to be involved in healthcare process
- Immediate access to health records from anywhere
- (close to) Entire healthcare process possible from system

Limitations

Solid

- Can't grant access higher than you possess. Can't limit max access to be granted to others.
- No logout functionality
- No IAM
- Missing documentation (temporarily/entirely)
- Can't provide access to individual Things
- No validation of updates to access
  - Causes lockouts
  - Possible to grant access to arbitrary URLs

Application

- Data validation
- Should have used pre-built components
- No logs of changes
- Insurance feature is not practical
- Patient shouldn't see everything / be able to change data
- Health workers need more than explicit access
  - Pharmacists would need more than 1 prescription
  - GP / Hospital sharing is common
- Location of data may be unsuitable

Conclusion

- Very helpful and active community
- Difficult for solo development – lack of resources
- Requires a lot of background knowledge
- Technology is ready (a bit buggy) for most basic development
- Will become the norm in the future
- Would not recommend for enterprise applications