B.Sc. Computer Science
Computer Science Department

# Demonstrating Decentralised Web Applications Using Solid

Candidate
**Joshua Prout**
**Student ID 700021737**

Supervisor
**Dr. David Wakeling**
**University of Exeter**

Co-supervisor

Academic Year
2022/2023

## Abstract

Since its inception, the World Wide Web has grown increasingly centralised, with masses of Web user's data being stored in huge data stores owned by companies such as Facebook. When data is centralised in this way, users are often locked in to using certain applications, as data is not inter-operable. Furthermore they do not have full control over who has access to their data, leaving them open to data collection and misuse. The Solid platform, led by Sir Tim Berners-Lee, provides a system where user data is stored in "Pods" (Personal Online Datastores), an area of the Web that they have full control over. This project uses Solid to produce two decentralised Web applications, a Pod manager tool and a fitness tracking application, and proposes a new format for decentralised social groups using Solid. While the applications produced were basic in nature, this project found that Solid provides a useful platform to rival centralised applications and other decentralised platforms. The project also discusses some of the challenges Solid faces before bringing the decentralised Web to a wider market, including data storage standardisation, Pod stability and security, as well as promoting greater knowledge and understanding among users and developers.

# Contents

# List of Figures

# List of Acronyms

| | |
|---|---|
| **RDF** | Resource Description Framework |
| **WAC** | Web Access Control |
| **ACL** | Access Control List |
| **OIDC** | Open ID Connect |
| **WebID** | Web Identity and Discovery |

# 1 Motivation, Aims and Specification

## 1.1 Motivation

It is the opinion of many in the Web community, including Sir Tim Berners-Lee, that the Web requires a dramatic shift from its current centralised nature. Far from the original dream of Web pioneers, of a decentralised, user controlled, inter-operable network, the modern Web is dominated by "walled gardens", large companies that store user data only accessible through restrictive APIs. Online social networks have many privacy flaws [1]

Berners-Lee believes in a future of the Web where the majority of data is decentralised, where people are free to control which people and companies can view and use their data [2]. The open-source framework developed to push this idea is called Solid (Social Linked Data).

The Solid framework is based around the concept of user "Pods", an area of the Web where they can store any data they choose. Each Pod has its own URL, and the user is free to choose where their Pod is hosted, either from a range of cloud hosts or hosted on their own machine.

There are three main goals of the Solid platform [3]:

- True data ownership: Users can choose where their data is stored, and who has access
- Modular design: Applications are decoupled from data they produce and use, meaning users aren't locked to one application to deal with the data
- Re-use of data: Data can be stored once and re-used for different applications, for example name, date of birth, friends list etc can be stored in the Pod and requested by applications instead of being stored redundantly across many applications and companies

## 1.2 Aims

The overall aim of this project is to solve some of the problems inherent with centralised applications by building decentralised Web applications using the Solid platform. The outcome from these apps will be used to help evaluate the current viability of the Solid platform for developing decentralised apps, specifically in the following areas:

- User Experience
- Developer Experience
- Adherence to decentralised goals (True data ownership, modular design, re-use of data)

### 1.2.1 Existing Work

The project will aim to improve upon these existing works in the decentralised Web space.

**Diaspora:**

Disapora is currently one of the biggest decentralised social networks, developed before the Solid project was established [4]. Users can host and join various communities and groups, both public and private. While anyone can host a Diaspora group, users are limited to using Diaspora's application to interface with data stored in the group. This project will aim to improve on this by producing decentralised social groups that can be modified and accessed easily by a range of applications.

**Solid Groups:**

An approach to providing Solid social groups exists, however it is arguably over-complicated [5], requiring an entire Pod per group, and many auxiliary files to produce web pages and UIs for the group. This project will aim to define a way to store groups in a simpler, efficient way that is decoupled from specific applications, and allow a user to host and store multiple groups within their main Pod.

**Penny (Pod Manager):** A range of Pod Manager tools have been developed for Solid, one of the best examples being 'Penny' [6]. Penny allows for changing access permissions for files and folders within the Pod, as well as uploading data. Penny works well as a tool for Solid developers, however this project aims to produce a similar concept, aimed at non-technical users with less knowledge about Solid and the decentralised web. This will be achieved with a simpler interface, and restrictions on potentially damaging access control decisions.

**Strava:**

A key research and development field for the decentralised Web is creating decentralised versions of successful centralised applications. This project identified Strava, an exercise tracking and social networking application as a useful concept to decentralise. Strava allows users to upload their exercise sessions and share them with their friends, however a lack of effective data sharing controls can lead to data breaches due to third party tracking. This included a notable breach of the location of sensitive military sites [7]. The decentralised version of this application will aim to eliminate this possibility, by requiring specific consent from the user for people and companies to access their data.

The second drawback the project will aim to solve is the data interoperability restrictions of applications such as Strava. After a user has uploaded their fitness data, it is difficult for them to change the application they use to view and edit the data without downloading all of their data from Strava servers, and re-uploading to a different application. Furthermore, if a friend wishes to view their friend's data, they are forced to use Strava to access the data and cannot use an application of their choice. The project aims to solve this by storing application data in the user's Pod, in a simple, application detached format. This will allow it to be accessed by any application that understands the proposed schema for data storage.

## 1.3 Specification and Requirements

This project will provide a set of applications to introduce the average internet user to the Solid platform. This will include a Pod manager tool to manage access to the contained file-system and to create and join Solid social groups. To demonstrate a useful user application, a fitness application similar in concept to Strava will be developed to allow the user to store and share their fitness data with other users in a decentralised way.

The decision to separate the Pod manager and the fitness application was made so that the Pod manager tool could be used outside of the context of the fitness application, and can be used to manage the data for a range of applications. Furthermore, splitting the applications will adhere to best security principles of least privilege, the fitness application does not have the ability to change access control in the Pod, meaning the user only has to trust one application with their access controls.

### 1.3.1 Pod Manager Tool Requirements

The Pod Manager should provide a user with a tool that is abstracted from the underlying decentralised architecture as much as reasonably possible, such that the transition to Solid is comfortable for the user. The tool should provide the following functionality:

1. Allow the user to view the containers and datasets within their Pod
2. Change permissions for specific containers and datasets.
3. Join a social group
4. Create a new social group
5. Accept or reject new members from the created group
6. Have safeguards to protect the user from causing catastrophic errors in the Pod

### 1.3.2 Fitness Application Requirements

The fitness application should provide a clean, easy to use interface that will be convenient for users to upload their exercise sessions. The requirements for the fitness app are:

1. Upload exercise sessions to their own Pod.
2. View chronological feeds of exercise sessions from friends.
3. Interface with the Solid group system so that links to exercise sessions can be shared to groups
4. Use the Solid groups to choose a group to view the exercise sessions uploaded to that group.
5. Store data in a standardised way such that it can be used by other applications

# 2 Design, Methods and Implementation

## 2.1 Underlying Technologies for Solid

### 2.1.1 Pods and WebIDs

The Pod is an area on the Web, where all of the user's data is stored. A Pod can either be hosted by the user on their own machine [8], or with an established Pod provider [9]. When the user signs up for a Pod, they also receive a WebID (Web Identity and Discovery) [10]. A webID is a URL used to identify each user on the Solid platform, built on top of the OIDC protocol (Open ID Connect) [11]. A typical webID resembles "alicedemo.solidcommunity.net/profile/card#me".

The actual webID URL points to a profile file stored on the user's Pod written in RDF (explained in section 2.1.3). This can contain extra information such as an email address, a profile picture etc. that user chooses to make public.

### 2.1.2 Web Access Control

Access to files within the Pod is controlled by Web Access Control (WAC), a decentralised protocol that uses .acl (Access Control List) files to authorise access to files or containers [12]. For each file or container, access levels can be specified to other users identified by a WebID, groups of users or public access. There are four levels of access that can be given:

- Read
- Append (Can add files but not update or delete)
- Write (Can add or delete files)
- Control (Change the access controls for the file or container)

Only applications that have been specifically authorised by the user can be used to modify access control lists in the Pod.

### 2.1.3 Semantic Web and RDF

The Semantic Web is an effort to improve the machine readability of data on the Web. To achieve this, a set of standards and practices have been written for how to describe data on the web using RDF (Resource Description Framework) files [13]. RDF files contain multiple "triples" (three values): a subject, a predicate and an object. The object is an attribute of the subject, and the type of the attribute is described by the predicate. The predicate is a URL linking to a page containing details about the relationship.

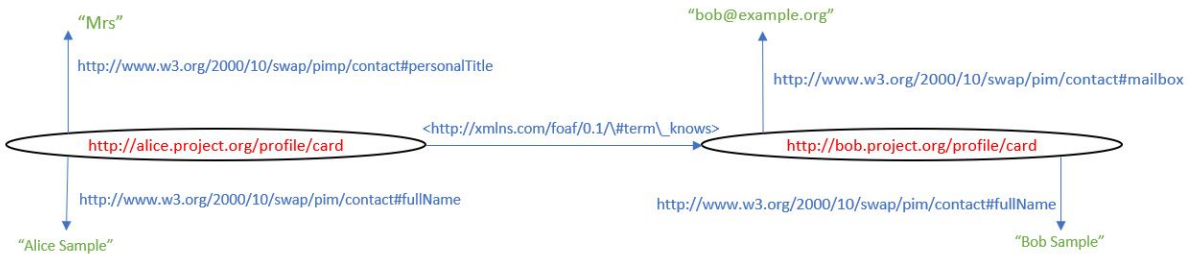
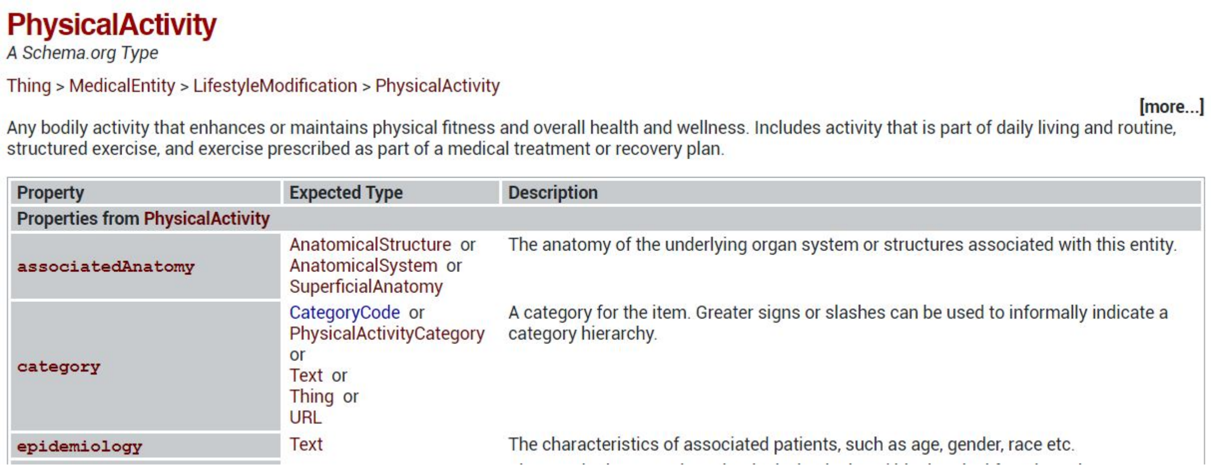

"Mrs"
http://www.w3.org/2000/10/swap/pimp/contact#personalTitle
"bob@example.org"
http://www.w3.org/2000/10/swap/pim/contact#mailbox
<http://xmlns.com/foaf/0.1/\#term_knows>
http://alice.project.org/profile/card
http://bob.project.org/profile/card
http://www.w3.org/2000/10/swap/pim/contact#fullName
http://www.w3.org/2000/10/swap/pim/contact#fullName
"Alice Sample"
"Bob Sample"

Figure 2.1: Example RDF triple connections



PhysicalActivity

*A Schema.org Type*

Thing > MedicalEntity > LifestyleModification > PhysicalActivity

**[more...]**

Any bodily activity that enhances or maintains physical fitness and overall health and wellness. Includes activity that is part of daily living and routine, structured exercise, and exercise prescribed as part of a medical treatment or recovery plan.

| Property | Expected Type | Description |
|---|---|---|
| Properties from PhysicalActivity | | |
| associatedAnatomy | AnatomicalStructure or AnatomicalSystem or SuperficialAnatomy | The anatomy of the underlying organ system or structures associated with this entity. |
| category | CategoryCode or PhysicalActivityCategory or Text or Thing or URL | A category for the item. Greater signs or slashes can be used to informally indicate a category hierarchy. |
| epidemiology | Text | The characteristics of associated patients, such as age, gender, race etc. |

Figure 2.2: Physical Activity Description on Schema.org

Figure 2.1 shows an example of RDF links. The red URLs are webIDs, each representing a user, the blue URLs are predicate links, describing the type of link, and the green strings are regular values. There is also a link between the two users showing that they know each other.

Predicates are listed on semantic web vocabulary sites, such as Schema.org which aim to standardise how concepts are described on the semantic web, and what the predicates should be used for. Figure 2.2 shows the semantic description for the predicate "schema.org/PhysicalActivity".

### 2.1.4 Solid Storage Structure

Data in a Solid Pod is organised in a similar way to a traditional filesystem, with a few key differences to allow for decentralised access.

- The lowest level data store in Solid is the "Thing". Each Thing is an RDF file that represents a single data entity, such as a person, an object, or a data entry such as an exercise session, similar to a single record in a traditional database. A Thing can have any number of attributes within such as name, author, size, distance etc. The nature of each attribute is described by a "predicate", a URL link to a page describing details of the attribute. The value of the attribute could be a string, integer, URL among other common data types.

- "Things" must be stored inside a Solid dataset, they cannot exist independently. A dataset is a file that can hold any number of Things, and maintains an access control list for the Things contained within.
- Solid Pod containers act like traditional folders for organising data in the Pod, they can contain a mixture of sub-containers, Solid datasets or traditional files such as images or audio files.

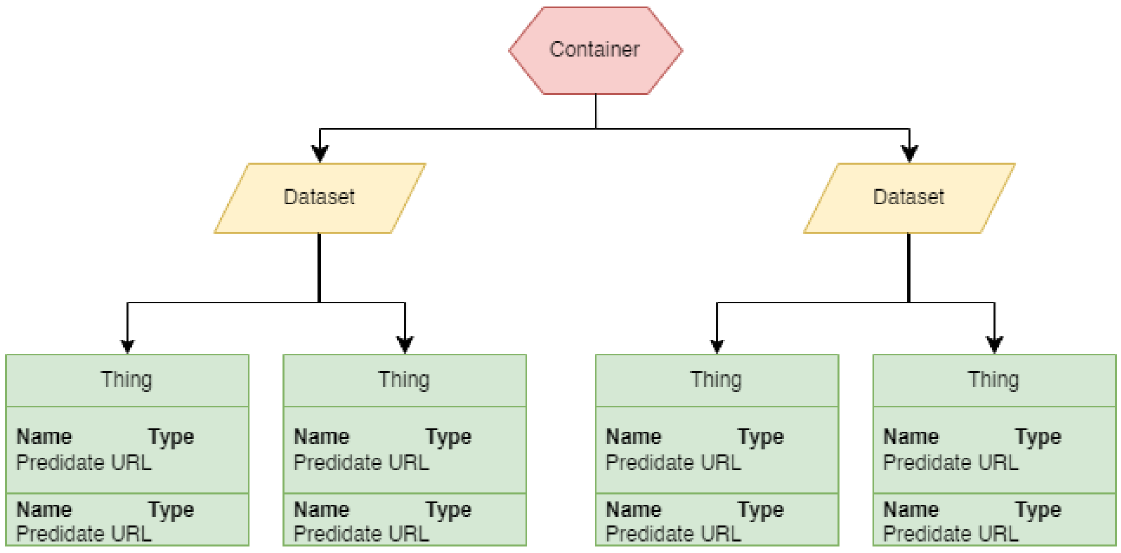


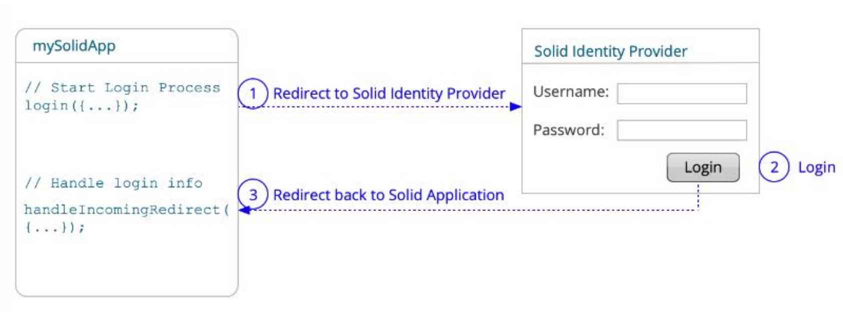
Figure 2.3: Example representation of Solid data: A Container containing two datasets, each containing two Things

Fig. 2.3 shows how these concepts will be represented in this report, and their hierarchy. Each container, dataset and Thing have a URL. The URL for the example data would be: *https://examplepod.solidcommunity.net/Container/Dataset#Thing*. Each Thing is referenced by a # followed by the name of the Thing within their parent dataset.

### 2.1.5 Languages, Libraries and APIs

This project chose to use NodeJS, specifically the ExpressJS framework for developing the tools and applications. ExpressJS was chosen to compliment development style of this project to rapidly produce a minimum viable product. The simple templating function and middleware allowed for focus to be kept on the Solid functionality.

For authenticating to Solid Pods, the Inrupt solid-client-auth-node Javascript library was used [14]. Any request to read or modify private data requires the user to authenticate themselves using the Solid OIDC protocol [11], which is built upon the OAuth 2.0 authorization framework [15].


mySolidApp
Redirect to Solid Identity Provider
Redirect back to Solid Application
Solid Identity Provider
Username:
Password:
Login
Login

```
// Start Login Process
login({...});

// Handle login info
handleIncomingRedirect(
{...});
```

Figure 2.4: Authentication Process

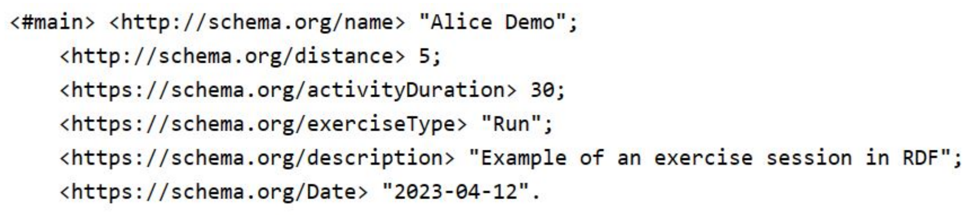
The library allows for the user to select which Solid identity provider they wish to be redirected to for authentication. If successful, they are returned to the application with a session key to be stored in the browser (Fig. 2.4). This key is used to authenticate any data requests made to Pods.

For interaction with a user's Solid Data, the Inrupt solid-client Javascript library was used [16]. The library provides abstraction over RDF files that Solid data are stored in, allowing data to be written and read without knowledge of the RDF syntax.

```

const newRun = buildThing(createThing({name: "main"}))
                        .addStringNoLocale(SCHEMA_INRUPT.name, fullName)
                        .addInteger(SCHEMA_INRUPT.distance, distance)
                        .addInteger("https://schema.org/activityDuration", time)
                        .addStringNoLocale("https://schema.org/exerciseType", "Run")
                        .addStringNoLocale("https://schema.org/description", remarks
    )
                        .addStringNoLocale("https://schema.org/Date", date)
                        .build();
```

Code 2.1: Example JavaScript code to create a new Thing using the solid-client library



```
<#main> <http://schema.org/name> "Alice Demo";
    <http://schema.org/distance> 5;
    <https://schema.org/activityDuration> 30;
    <https://schema.org/exerciseType> "Run";
    <https://schema.org/description> "Example of an exercise session in RDF";
    <https://schema.org/Date> "2023-04-12".
```

Figure 2.5: RDF file produced by Code 2.1

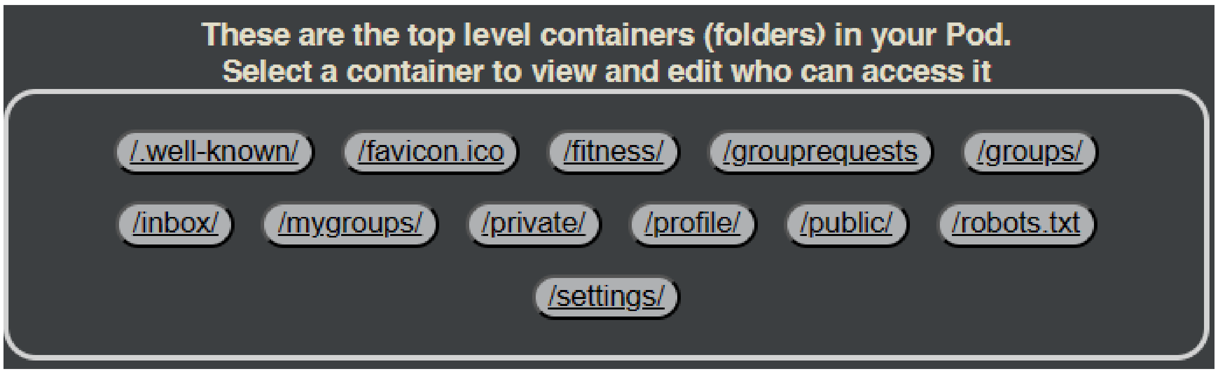
## 2.2 Design of the Pod Manager Tool

The first program developed was the Pod Manager tool. The tool is designed to allow for management of the Pod data and Solid social groups, outside of the context of any specific application.

### 2.2.1 Managing Dataset Permissions And Requests

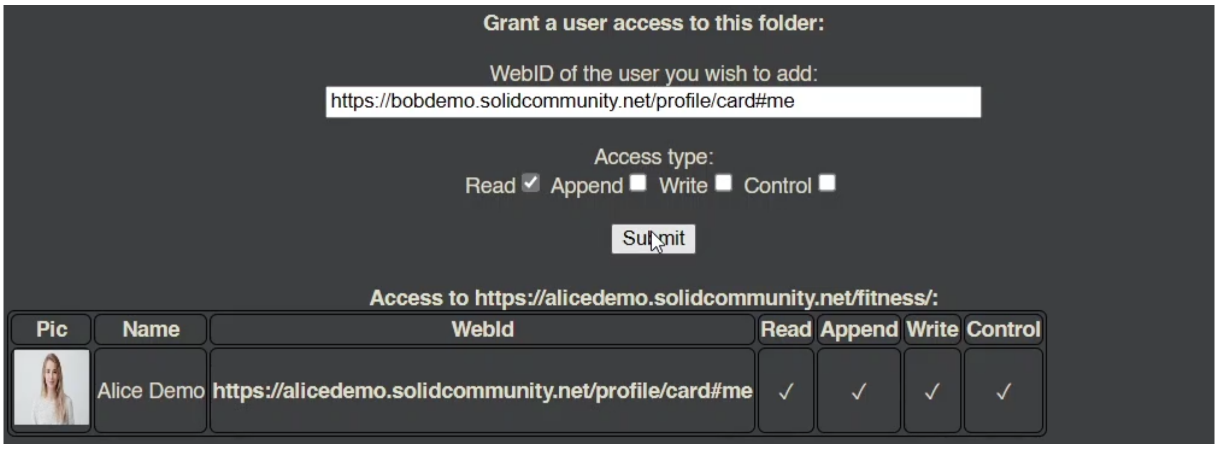
A key feature of the Pod Manager is managing specific permissions for individual containers and datasets. From the homepage of the tool, there is a list of the top level containers and datasets. Once the user selects a container or dataset, they have the option to view all of the users that have access and their associated access rights.

The user can modify the access control list by inputting a user's WebID, and selecting which of the access types they wish for them to have (Fig. 2.7). This updates the .acl file for the resource.


These are the top level containers (folders) in your Pod.
Select a container to view and edit who can access it

/.well-known/ /favicon.ico /fitness/ /grouprequests /groups/
/inbox/ /mygroups/ /private/ /profile/ /public/ /robots.txt
/settings/

Figure 2.6: Container selection


Grant a user access to this folder:

WebID of the user you wish to add:
https://bobdemo.solidcommunity.net/profile/card#me

Access type:
Read ✓ Append ☐ Write ☐ Control ☐

Submit

Access to https://alicedemo.solidcommunity.net/fitness/:

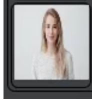
| Pic | Name | WebId | Read | Append | Write | Control |
|---|---|---|---|---|---|---|
|  | Alice Demo | https://alicedemo.solidcommunity.net/profile/card#me | ✓ | ✓ | ✓ | ✓ |

Figure 2.7: Viewing and modifying the access control list

### 2.2.2 Creating a Group

This project has chosen to prioritise a simple, standardised storage and access format for Solid social groups, so that a user can use any program they wish to access the group, as long as it understands the standard format. The group is stored within the group owner's own Solid Pod.

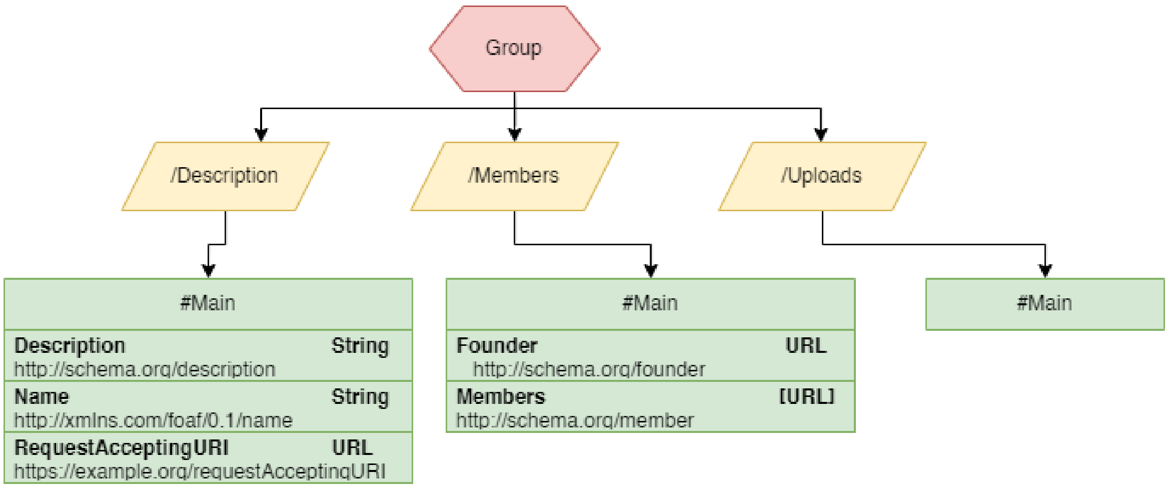
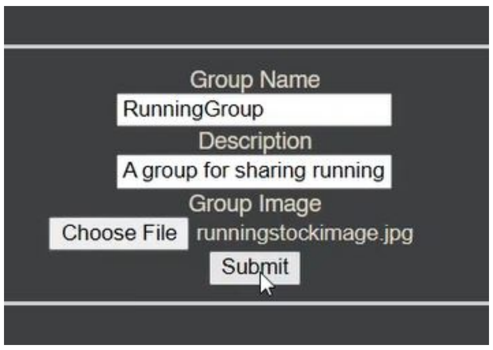


Figure 2.8: Project proposed structure for a Solid Group



| | |
|---|---|
| Group Name | RunningGroup |
| Description | A group for sharing running |
| Group Image | runningstockimage.jpg |

Figure 2.9: Form for creating a group

When the user creates a new group, they choose a name for the group, a description of what the group is for, and an optional image. Inside their Pod, a new container is created for the group, containing three Solid datasets (Fig. 2.8).

- The description dataset has public read access, so that users can view the name, description and other public information about the group.
- The members dataset holds the list of members in the group. Once a member's WebID is added to the members list, they should be given read access to the dataset so they can view which other users are in the group.
- The uploads dataset holds links to files in user's own pods, such as links to images or social media posts, but the actual data is not stored in the group. Once a user is accepted into the group, they should be given read and append access to the uploads dataset, so that they can upload links to data they wish to share.

The decision was made to split the group into three different datasets for ease of access control. There was consideration to having a single dataset for the Group which would contain multiple Things, however Things cannot have their own access control lists, therefore different datasets for different access levels were required.

The decision to store data in user's own Pods, and only store links in the group helps the system to adhere to the data ownership goal of Solid. If a user decides to revoke group access to some of their data, they do not have to rely on the group owner to remove the data, they can simply revoke the access permission to the data from their own Pod, and access is instantly denied.

### 2.2.3 Joining a group

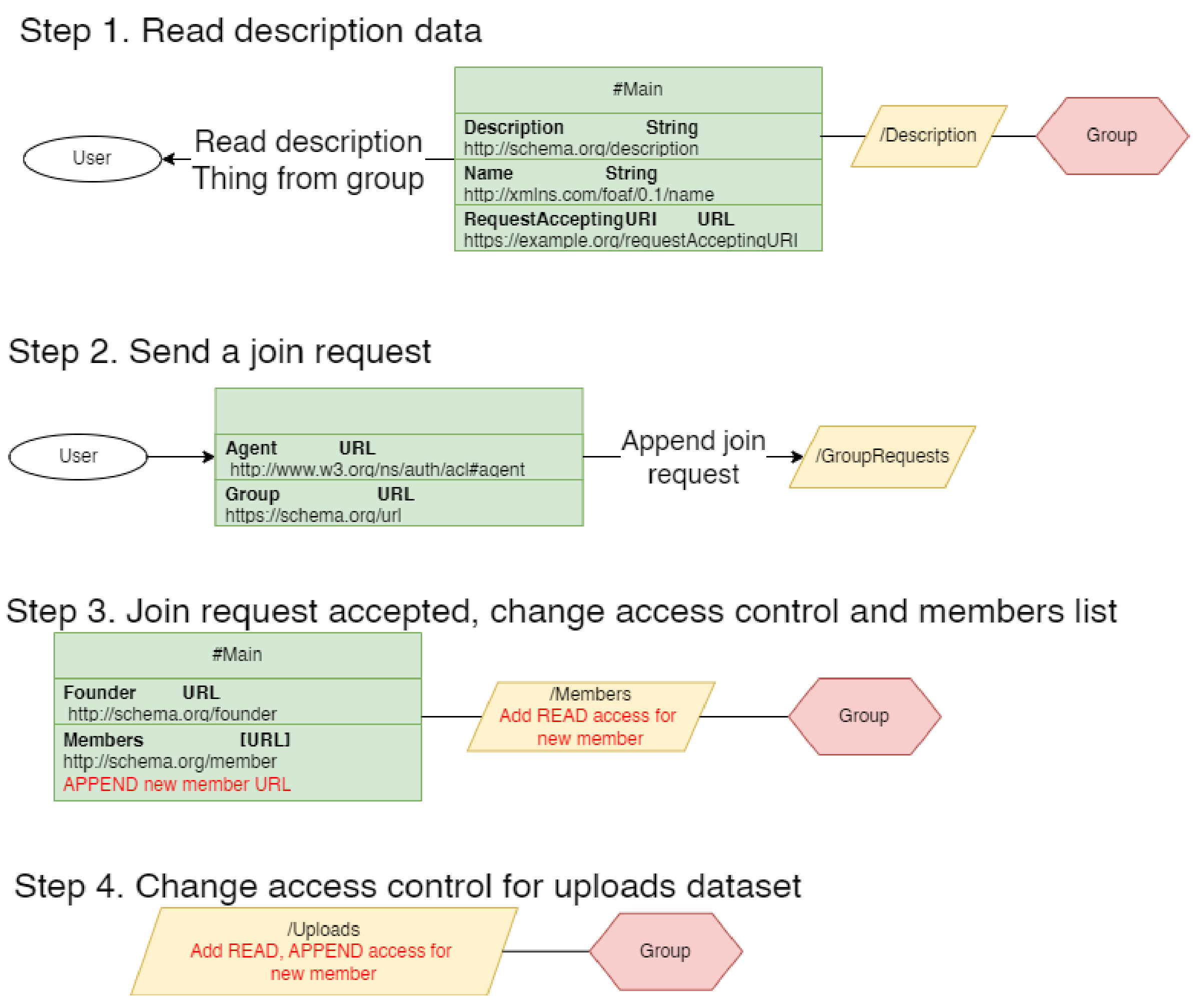

Step 1. Read description data
User
Read description Thing from group
#Main
Description String
http://schema.org/description
Name String
http://xmlns.com/foaf/0.1/name
RequestAcceptingURI URL
https://example.org/requestAcceptingURI
/Description
Group
Step 2. Send a join request
User
Agent URL
http://www.w3.org/ns/auth/acl#agent
Group URL
https://schema.org/url
Append join request
/GroupRequests
Step 3. Join request accepted, change access control and members list
#Main
Founder URL
http://schema.org/founder
Members [URL]
http://schema.org/member
APPEND new member URL
/Members
Add READ access for new member
Group
Step 4. Change access control for uploads dataset
/Uploads
Add READ, APPEND access for new member
Group

Figure 2.10: Process of joining a group. Only Things and datasets that are relevent to each stage are shown

The stages of a user joining a new group are described below and in Figure 2.10.

- Step 1. When a user finds a group they would like to join, they retrieve the description Thing from the public "description" dataset. The Thing contains a URL field "requestingAcceptingDataset", this links to a dataset in the group owner's Pod where they receive all of their group join requests. The decision to allow the group owner to specify the exact dataset to receive group requests to gives them greater flexibility, they can choose to have a requests dataset for each group, or have a combined dataset for all of the groups they own in one place.
- Step 2. The user then appends a group request Thing to the request accepting dataset.

The Thing contains two fields, the requesting user's WebID and the URL of the group they wish to join.

- Step 3. The group owner can choose to accept or reject the request. If the request is accepted, then the requesting user's WebID is added to the members list in the members dataset. The user's webID is also given read access in the members dataset ACL.
- Step 4. The requesting user is given read and append access in the uploads dataset.

The user can now upload data to the group, and a link to the group is added in the user's Pod in a dataset called "subscribedTo". The decision to make the group simple and standardised helps adhear to the modular design principle of Solid, any Solid application has the potential to use this group system easily, not just the Pod Manager for this project.

## 2.3 Design of the Fitness Application

The fitness application allows the user to upload data relating to their health, as well as viewing health data from other people that have given them access.

### 2.3.1 Creating an exercise session

When the user chooses to upload a new exercise session to their Pod, they are given a form to input details such as the type of exercise, the distance they covered, the time it took and an optional image of the map of the route they took.

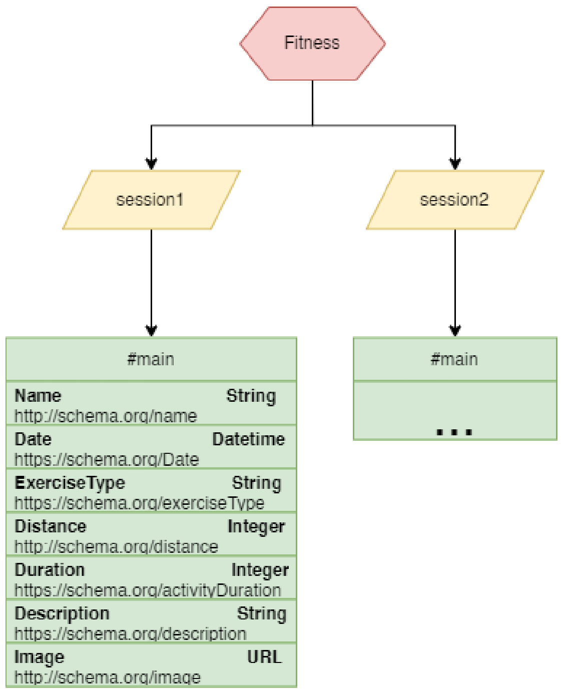


Figure 2.11: Schema for the exercise sessions container

Figure 2.11 shows the schema of an uploaded fitness data in the user's Pod. Conventionally there would be a single dataset with multiple Things for each exercise session, however this project chose to create a container with multiple datasets. Each dataset is for one exercise session, and contains one Thing with the data for the session. This is to allow the user to change the access permissions for each individual exercise session, as they may wish to give another user access only to certain sessions. Things cannot have their own access control lists, only datasets, therefore there is one dataset per session.

The user can also choose to share their exercise session with a group that they are a member of, either to a group stored on their own Pod that they manage, or a group managed and stored on someone else's Pod. If this option is chosen the following steps are executed:

1. The exercise session is uploaded to the user's pod as usual.
2. Using the URL of the chosen group the list of group members is retrieved, this is a list of their webIDs.
3. The dataset for the exercise session is edited, to give read access to every webID in the members list.
4. The URL for the exercise session is appended to the uploads dataset in the group.

The other members of the group can now see the URL of the exercise session, and have permission to access the data at that URL.

### 2.3.2 Viewing exercise sessions

The user can choose to view a social media scrolling style feed from their friends or members of a group. If they choose friends, the program retrieves all of the links in their profile with the predicate "foaf.org/knows", the standard friend list in Solid, helping to adhere to the re-use of data principle by using the existing friend list in the Pod. If they choose groups, they select a group from the list of groups they are subscribed to, and the program retrieves the list of members. For either option, the program now has a list of user webIDs to retrieve from. It is important that the location of the container in the Pod holding the fitness sessions is in the correct relative place, which is in a top level container named "Fitness". The application will iterate through the webIDs and try to retrieve the datasets contained in "https://[webID]/Fitness".

This process relies on the other users having our webID listed as having permission in their ACL. If the request is denied, then the other user's pod will return a HTTP 403 unauthenticated response, in this case the application gracefully handles the error and moves on to the next user to retrieve from.

# 3 Project results, Evaluation and Testing

## 3.1 User Experience Testing

To test the viability of the applications developed, a user experience testing plan was created. This project assumes that the user already has a basic Solid Pod set up with a provider, as creating a Pod is outside of the scope of this project. The user will be provided with a newly created Pod to use for the test. The feedback sessions will use standard software focus group methods [17], however common practice suggests that more users should be recruited [18], unfortunately this was not practical for this project.

### 3.1.1 Initial Survey

The aim of the user testing will be to first discover the user's knowledge and attitude towards decentralised applications, as well as their general attitude towards privacy matters. The user will read the following context setting paragraph:

*Most of the websites and web applications you have used before such as Facebook, Twitter, Strava are "centralised". Your data is stored on the company's computers, and they have control over who can access it and why, and you have to trust them to respect your data wishes. The apps you are testing today are "decentralised", and use a system called the Solid platform. All of your data is stored in a space on the web called your Pod, here you can store any data you wish in one place, and the same data can be used by multiple apps. You have full control over who can access specific parts of the data in your Pod. Any data that the application loads from other people, is loaded from the other person's Pod.*

They will then answer a short survey before completing the tasks.

### 3.1.2 Testing

The tasks each user are asked to complete are as follows:

1. Open the Fitness application which will initialise the relevant datasets automatically.
2. Using the Pod Manager tool, and given another user's WebID address, give this user read access to the Fitness container.
3. Using a given group URL, send a join request to a group.
4. Create their own group, and accept a join request from a new member.
5. Open the Fitness application, and upload an exercise session, giving access to the group they joined earlier.

6. View the exercise sessions feed for the logged in user.
7. View the exercise sessions from the group.

After they have attempted all of the tasks, the test user will be asked a short questionnaire about their experience using the app and whether their opinions of decentralised applications have changed.

## 3.2 User Testing Results

### 3.2.1 User 1

User 1 is a 22 year old University student.

**Initial questionnaire:**

| | |
|---|---|
| Which social media apps do you currently use? | "Instagram, Facebook, Reddit mainly" |
| Do you use any fitness tracking applications? | "Apple Fitness" |
| Have you used any decentralised applications before | "No" |
| What concerns, if any do you have around your online privacy? | "I worry that if my GPS location was being leaked by an app, somebody could use it to stalk or rob me" |

**Application testing:**

User 1 found the app intuitive and fairly simple to use, and required no intervention or help during the test to complete all of the tasks.

**Final questionnaire:**

| | |
|---|---|
| Are there any features of the apps that you liked or found easy to use? | "It was easy to navigate around the site and find what I needed to do" |
| Are there any parts that were particularly difficult or confusing? | There was no notification when the user requested to join my group, so unless you keep checking the requests bit you'd have no way of knowing someone had requested to join" |
| Did the decentralised apps feel different to use compared to centralised apps you have used before? | "Uploading the exercise session and seeing other people's in the feed felt the same as a normal social media, however having to manually give someone access to my folder was different." |
| Now that you have used these decentralised apps, would you consider using decentralised apps in future | "In principle yes, it's nice to be able to choose who can see my data, however it would depend on whether my friends also wanted to use it" |

### 3.2.2 User 2

User 2 is a 20 year old university student.

**Initial questionnaire:**

| | |
|---|---|
| Which social media apps do you currently use? | "Tiktok, Instagram, Snapchat" |
| Do you use any fitness tracking applications? | "Strava" |
| Have you used any decentralised applications before | "No" |
| What concerns, if any do you have around your online privacy? | "When you say something or search something and it comes up in adverts later on that day" |

**Application testing:**

User 2 found the apps fairly easy to use, but was slightly confused about whether they should be using the Pod manager tool or the fitness application for each of the tasks.

**Final questionnaire:**

| | |
|---|---|
| Are there any features of the apps that you liked or found easy to use? | "Not needing to log in each time for the different apps was nice" |
| Are there any parts that were particularly difficult or confusing? | "The app felt a bit clunky and looked quite dated" |
| Did the decentralised apps feel different to use compared to centralised apps you have used before? | "It felt pretty similar, except it took a while for things to load and process." |
| Now that you have used these decentralised apps, would you consider using decentralised apps in future? | "If it was a bit easier to use yes, if it made it harder for the advertisers to get my data that would be good" |

### 3.2.3 User 3

User 3 is a 45 year old healthcare assistant.

**Initial questionnaire:**

| | |
|---|---|
| Which social media apps do you currently use? | "Facebook and Instagram" |
| Do you use any fitness tracking applications? | "MyFitnessPal" |
| Have you used any decentralised applications before | "No" |
| What concerns, if any do you have around your online privacy? | "I have private accounts on my social media which makes me feel pretty secure. " |

**Application testing:**

User 3 initially found the concept of Solid and the Pods confusing, asking whether you needed a separate Pod for different apps and if each Pod had a separate login.

**Final questionnaire:**

| | |
|---|---|
| Are there any features of the apps that you liked or found easy to use? | "The apps were very clean, not too many options cluttering up the screen" |
| Are there any parts that were particularly difficult or confusing? | "I didn't really understand how the Pods worked or how it was different to a normal app" |
| Did the decentralised apps feel different to use compared to centralised apps you have used before? | "Didn't feel that different but I wasn't really sure what was going on with where the data was being stored" |
| Now that you have used these decentralised apps, would you consider using decentralised apps in future? | "The apps I use have pretty good privacy protections and are easy to use, so I would probably stick with them" |

## 3.3 Functional Requirements Evaluation

### 3.3.1 Pod Manager Tool:

*Allow the user view the containers and datasets within their Pod:* This requirement was met, the user can view the containers and their contained dataset from the index page of the tool.

*Change permissions for specific containers and datasets:* The user can add another user to the access control list for a container or dataset.

*Join a social group:* If the user has the exact URL for a group, they can input it into the tool and view the group, and send a join request. Needing the exact URL is a limitation, as there is currently no way to search for a group without a centralised directory.

*Create a new social group:* This feature works well, the group datasets are created automatically without the user having to deal with the underlying Solid technology.

*Accept or reject new members from the created group:* When a request to join a group arrives, the user can accept or reject the request. An improvement would be to disallow multiple requests as if a user requests multiple times, it shows multiple times in the requests feed.

*Have safeguards to protect the user from causing catastrophic errors in the Pod:* A flaw with Solid is that the user can lock themselves out of a file by removing their control access permission, which cannot be reinstated without control access leading to a permanent deadlock. The Pod Manager Tool does not allow this access to be removed, however it is still possible for a bug to cause an unexpected overwrite of the .acl file and cause this lockout

### 3.3.2 Fitness Application:

*Upload exercise sessions to their own Pod:* This requirement was met, the user can use a simple form to input data about their session, and upload it to their Pod. The exercise session is then saved as an RDF file with appropriate predicate links.

*View chronological feeds of exercise sessions from friends:* This requirement was met, the user can view a feed of friend's sessions so long as the user has the appropriate permissions in the friend's Pod.

*Interface with the Solid group system so that links to exercise sessions can be shared to groups:* The application works properly with the groups, when uploading an exercise session there is a drop down menu to choose which group the user wants to upload the session to.

*Use the Solid groups to choose a group to view the exercise sessions uploaded to that group:* The user can select a group from the list of all groups they are subscribed to. The application doesn't make a distinction between fitness groups and other kinds of groups that the user may be subscribed to. However, the application will only show fitness sessions, as they are linked with the predicate "https://schema.org/PhysicalActivity"

*Store data in a standardised way such that it can be used by other applications:* When exercise sessions are uploaded using this application, the RDF predicates used are standardised and listed on vocabulary sites such as schema.org, so that other applications can understand the data.

## 3.4 Development Process Evaluation

Due to the limited time frame in which to develop this project, a strict development process was needed to keep to the deadline. Before programming began, the applications were decomposed into a list of main tasks and features that would need to be produced. Each task was estimated to take roughly the same amount of time to complete.

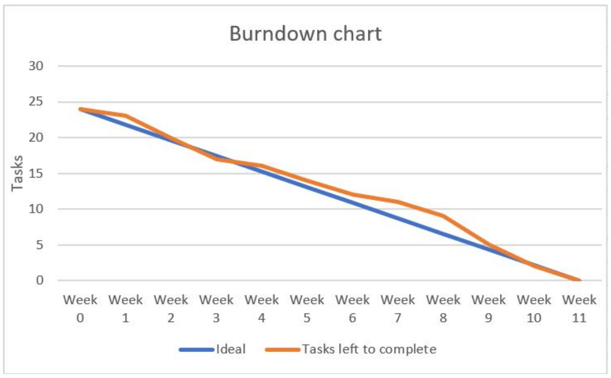


Figure 3.1: Burndown chart for the programming phase of the project

The project used a one week sprint system, whereby the goal was to complete two tasks per week, and present a progress update for supervision at the end of the week. The ideal rate of two tasks per week is plotted against the actual number of tasks completed per week in Figure 3.1. This produces a burndown chart, a progress monitoring tool shown to be practical for undergraduate programming projects [19].

# 4 Discussion And Conclusion

## 4.1 Viability of Solid

This project produced two applications using the Solid platform, meeting the requirements defined in Chapter 1, and receiving generally positive feedback from user testing. The results from this project and the lessons learnt from developing the applications will now be used to evaluate the current viability of the Solid platform for decentralised applications.

### 4.1.1 User Experience

The user interfaces for the project applications were very basic but functional, as focus was on developing the complex interactions with the Solid pods. Solid however does not limit applications to basic UI, especially with the React SDK for Solid [20]. It is possible to create far more modern and engaging user interfaces that reflect centralised applications, an important feature for attracting users towards decentralised applications.

A user experience area identified as lacking during this project is difficulty in searching and discovering other users and groups. To look up a user or group, we require the exact webID or URL, there is no way to search by name or title. Searching and recommendation solutions are being developed [21], however solutions either involve a centralised system or a crawler, which is inefficient and privacy compromising.

### 4.1.2 Development Experience

One of the biggest challenges for new Solid developers is using the Semantic Web and RDF storage instead of traditional data storage methods such as an SQL database. Solid has mitigated this problem using the Solid Client API [16] which was incredibly useful during this project for providing an abstracted way to write and read from RDF files.

Solid is still in the early stages, and as such it is frequently noted in the documentation that the design and code are not finalised. This produces challenges for developers building and operating applications over the long term.

Solid does not currently provide data locking facilities, in a multi-user system where two or more users are trying to modify the same file there is a danger of consistency errors. Pods also do not currently provide rollback features for data.

There is currently no standard for where data should be stored within the Pod, meaning developers have to make assumptions. For example in this project, the fitness application assumes that fitness sessions are stored in a top level container "/fitness", and the sessions fit a certain structure, which may differ from other applications, limiting inter-operability. Standards are beginning to be defined but have still not been accepted [22]

### 4.1.3 Adherence To Decentralised Goals

There are three main goals for the Solid platform as a whole: data ownership, modular design and reusable data [3].

Data ownership is currently achievable, this project has succeeded in giving the user full choice over who can access their data, and stops third parties from mining their data without consent. However, Pods are currently not encrypted and are therefore vulnerable to data breaches.

Modular design is also mainly achievable, however for easier data-interoperability between applications it would be useful to have better standards for how data is organised. There are also overlapping semantic data definitions, for example multiple available predicate links for images: "http://xmlns.com/foaf/0.1/Image" "http://schema.org/image".

Resuable data has proven to be a useful concept, with this project reusing the friends list stored in the profile card for example, however for data to be truly resuable it must again be defined using agreed definitions.

## 4.2 Conclusion

At the current stage of its development, Solid provides a very useful initial platform for introduction into decentralised Web applications. A developer with moderate Web development experience can easily pick up Solid development using the JS libraries, and produce functional and user friendly applications, shown by the fulfilled requirements in this project. The applications were also able to make a strong attempt at following the Solid decentralised principles of data ownership, modular design, and reuse of data.

The challenge for Solid moving forward will likely be recruiting users to move to decentralised apps. User testing and surveying in this project has shown that while they were generally happy to use the decentralised applications produced, they were not motivated enough to switch to decentralised for their daily Web usage.

## 4.3 Future Work

One proposal from the literature review advocated for decentralised consensual data gathering for research purposes using Solid [23]. The data generated by this project could work well in research surveys, for example gathering data on average amounts of exercise a population does. Requests for an individual user's fitness data can be sent to the Pod, and attributes such as distance and time can be read easily as they are defined using RDF. This would give the user the choice of which data gathering operations they trust and wish to support.

In order to more easily draw users, decentralised applications should provide tools to transfer data from centralised applications into the Pod. In the case of the fitness application for this project, a tool to transfer existing data from Strava or MyFitnessPal could be a useful step, as well as learning resources about how to use Solid.

# References

[1] V. V. Pham, S. Yu, K. Sood, and L. Cui, “Privacy issues in social networks and analysis: A comprehensive survey,” *IET Networks*, vol. 7, no. 2, p. 74–84, 2018.

[2] L. Clark, “Tim berners-lee: We need to re-decentralise the web,” Feb 2014, [Accessed 21/03/2023]. [Online]. Available: https://www.wired.co.uk/article/tim-berners-lee-reclaim-the-web

[3] “Solid - mit,” [Accessed 15/04/2023]. [Online]. Available: https://solid.mit.edu/

[4] A. Bielenberg, L. Helm, A. Gentilucci, D. Stefanescu, and H. Zhang, “The growth of diaspora - a decentralized online social network in the wild,” in *2012 Proceedings IEEE INFOCOM Workshops*, 2012, pp. 13–18.

[5] A. Veltens, “Solid-groups · gitlab,” [Accessed 20/03/2023]. [Online]. Available: https://gitlab.com/angelo-v/solid-groups

[6] V. Tunru, “Penny,” [Accessed 21/04/2023]. [Online]. Available: https://gitlab.com/vincenttunru/penny

[7] R. Pérez-peña and M. Rosenberg, “Strava fitness app can reveal military sites, analysts say,” Jan 2018, [Accessed 07/11/2022]. [Online]. Available: https://www.nytimes.com/2018/01/29/world/middleeast/strava-heat-map.html

[8] “Running your own solid server,” [Accessed 20/03/2023]. [Online]. Available: https://solidproject.org/self-hosting/css

[9] “Get a pod,” [Accessed 16/11/2022]. [Online]. Available: https://solidproject.org/users/get-a-pod

[10] A. Sambra, H. Story, and T. Berners-Lee, “Webid 1.0,” Mar 2014, [Accessed 06/11/2022]. [Online]. Available: https://www.w3.org/2005/Incubator/webid/spec/identity/

[11] N. Sakimura, J. Bradley, and M. Jones, “Openid connect core 1.0,” Nov 2014, [Accessed 21/04/2023]. [Online]. Available: https://openid.net/specs/openid-connect-core-1_0.html

[12] Solid, “Solid web access control spec,” Jun 2019, [Accessed 06/11/2022]. [Online]. Available: https://github.com/solid/web-access-control-spec

[13] F. Manola and E. Miller, “Rdf primer,” Feb 2014, [Accessed 06/11/2022]. [Online]. Available: https://www.w3.org/TR/rdf-primer/

[14] “Solid-client-authn-node api,” [Accessed 21/04/2023]. [Online]. Available: https://docs.inrupt.com/developer-tools/api/javascript/solid-client-authn-node/

[15] D. Hardt, “The OAuth 2.0 Authorization Framework,” Oct. 2012, [Accessed 21/04/2023]. [Online]. Available: https://www.rfc-editor.org/info/rfc6749

[16] “Solid-client api documentation,” [Accessed 21/04/2023]. [Online]. Available: https://docs.inrupt.com/developer-tools/api/javascript/solid-client/

[17] J. Kontio, L. Lehtola, and J. Bragge, “Using the focus group method in software engineering: obtaining practitioner and user experiences,” in *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE ’04.*, 2004, pp. 271–280.

[18] J. C. Bastien, “Usability testing: a review of some methodological and technical aspects of the method,” *International Journal of Medical Informatics*, vol. 79, no. 4, pp. e18–e23, 2010, human Factors Engineering for Healthcare Applications Special Issue. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1386505608002098

[19] C. J. Woodward, A. Cain, S. Pace, A. Jones, and J. F. Kupper, “Helping students track learning progress using burn down charts,” in *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, 2013, pp. 104–109.

[20] “Solid react sdk,” [Accessed 22/04/2023]. [Online]. Available: https://docs.inrupt.com/developer-tools/javascript/react-sdk/

[21] X. Ma, J. Ma, H. Li, Q. Jiang, and S. Gao, “Armor: A trust-based privacy-preserving framework for decentralized friend recommendation in online social networks,” *Future Generation Computer Systems*, vol. 79, pp. 82–94, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17300547

[22] L. G. J. Emilio, E. Prud’hommeaux, I. Boneva, and D. Kontokostas, *Validating RDF data*. Morgan amp; Claypool Publishers, 2018. [Online]. Available: https://doi.org/10.1007/978-3-031-79478-0

[23] C. Sun, M. G. Ocaña, J. van Soest, and M. Dumontier, “citizen-centric data platform (tidal): Sharing distributed personal data in a privacy-preserving manner for health research,” *Semantic Web Journal*, 2022. [Online]. Available: https://www.semantic-web-journal.net/content/citizen-centric-data-platform-tidal-sharing-distributed-personal-data-privacy-preserving

# Acknowledgments

I would like to thank the Computer Science faculty, especially Dr David Wakeling for his excellent support and invaluable advice throughout this project, as well as extra wisdom about the wider world of Computer Science.