MatchMe **Technical Solution Design**

short line

RMIT University  
COSC2408: Programming Project 1

# Document Control

**Distribution**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Issued** | **Recipient** | **Position** |
| V 1.0 | 04/05/2017 | Amir Homayoon Ashrafzadeh | Supervisor |
|  |  |  |  |

**Amendment History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Section** | **Page** | **Version** | **Comment** |
| <Enter Doc. Section No.> | <Enter Page No.> | <Enter Version No.> | <Enter Comments to explain the reason for the document text or other changes,    e.g., Updated text after walkthrough with the stakeholders, or    e.g., Updated section after technical consultation> |
|  |  |  |  |

.

**Staff or Entities Consulted**

|  |  |
| --- | --- |
| **Name** | **Position / Organization** |
| Jasmine Ellis  Hao Li  Haotian Xu  Xieyang Wu | Lead Developer  Developer  Front-end Developer  Tester |

**Related Documents**

|  |  |  |
| --- | --- | --- |
| **Name** | **Author** | **Description** |
| <Enter Document Name> | <Enter Author> | <Enter Document Description> |
|  |  |  |

***Preface***

The purpose of this document is to outline the Technical Solution Design for MatchMe. This document describes the technical environment, overall project architecture, system architecture, application functionalities and features, database architecture, implementation instructions, non-functional specifications, summary of test results, known issues and risks.

# Table of Contents

[**Document Control**](#_39no5nygf1dm)1

[**1. Introduction**](#_d4mr3wbn847)5

[**2. Technical Environment**](#_158op5pdgz9)6

[**3. Overall Architecture**](#_8z79nrnw9m95)8

[Class Diagram](#_adeciogat4z) 8

[**4. System Architecture**](#_5lp1lrld0t7f)9

[4.1 Functionalities/features](#_nm2cgovgyg17) 9

[4.1.1 User Registration](#_gn5ld0nlnwrq) 9

[4.1.2 Login / Logout](#_7096nyyhgbnb) 12

[4.1.3 Edit profile](#_x8hntujgvsmx) 13

[4.1.4 Change account settings](#_jeworo3slhtj) 13

[4.1.5 View Matches](#_f1l9zkwljmdi) 14

[4.1.6 Like / unlike other user](#_ey1hy5i6anyf) 15

[4.1.7 View your likes / who has liked you](#_367bt9o90iq) 15

[4.1.8 Admin: delete user](#_g35ck1jtqnfu) 16

[4.1.9 Admin: search users by name](#_cszyxs2qob9d) 17

[**5. Database Architecture**](#_m2jn5xeaztrf)18

[5.1 Database Diagram](#_97a4oy2gpgsp) 19

[**6. Implementation Instructions**](#_fem98sguak4f)20

[6.2 What you’ll need](#_lfl22i4duss1) 20

[6.1 Getting Started](#_rt54y6gomwcy) 20

[6.3 Detailed Instructions](#_en1i8fcjw95z) 20

[**7. Summary of test results**](#_df29ksjvbpx5)21

[**8. Known Issues & Risks**](#_kohyu337ugn1)21

[8.1 Risks](#_bhidpt5jt47k) 21

[8.2 Issues](#_fpk4324bc3vi) 21

[8.2.1 Known Issues](#_w6acyncmqqsd) 22

[**9. Appendix**](#_jshd44zho2if)23

[9.1 Trello](#_2w7fveeimlqx) 23

[9.1.1 Product backlog](#_86wi2y31bw9i) 23

[9.1.2 Sprint Backlog](#_jvbnj62zvw8) 24

[9.1.3 Weeks 2 - 13](#_v6t1il85bc7k) 25

[9.1.4 Bug report](#_q4ur7368exvx) 26

[9.1.5 Stretch Goals / Ideas](#_hi3gxuhwz4qq) 26

[9.1.6 Resources](#_cimgq6skmd51) 27

[9.1.7 Quality Assurance](#_9s1gml4f4aag) 28

[9.1.8 Colour Legend](#_tqld5bxvse7h) 28

[**10. Coding Standard**](#_cz6lstcpnxy3)29

# 1. Introduction

MatchMe is a simple dating application that matches users up according to their hobbies, interests, and other profile data such as diet and education level. Users who register are able to view users they have been matched with. They may also like and unlike other users. The application also includes some admin functionality. Administrators are able to delete users. A search function was also added to the Users Index to allow for a convenient way to search for and find users by name.

MatchMe is hosted on Heroku. Any image files uploaded by the user are stored in an AWS S3 bucket. The application was built using Ruby on Rails, PostgreSQL, JavaScript, HTML5, and CSS3. It was developed on Cloud9 IDE, which comes equipped with a Unix command line terminal. For version control and to collaborate with the same source code, we used Git and Github. The application also uses several Ruby Gems, including Geocoder and Carrierwave, used for geolocating and image uploading respectively.

The project was not built for commercial use and as such has a relatively low level of complexity by industry standards. Over time if new features are added and current features are enhanced, the project may be said to be of moderate complexity.

# 

# 

# 

# 

# 

# 2. Technical Environment

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Ruby on Rails** | Ruby on Rails is a server-side web application framework written in Ruby. Rails is a model–view–controller (MVC) framework, providing default structures for a database, a web service, and web pages. |
| **CSS3, Sass** | Used to style our website, e.g. change backgrounds and fonts. |
| **Bootstrap** | A front-end framework that includes an extensive CSS and JS library. Used as a convenient way to style our site and make certain elements responsive. Was also used for the profile image pop up modal. |
| **JavaScript (JS)** | An interpreted, runtime language. Used for various client-side functions. |
| **jQuery** | JS library used to simplify client-side scripting. |
| **Git** | A version control system for tracking changes and coordinating work amongst multiple people. This is what we used for version control. |
| **GitHub** | A web-based Git version control system. Our method of source control. |
| **Cloud9** | A web-based development environment. We chose to use it because we could access our work environment from different machines. C9 also came with Ruby and PostgreSQL pre-installed. |
| **Postgresql** | An open source database engine used for storing all of the user’s data. Pre-installed on both C9 and Heroku. |
| **Heroku** | A PaaS built specifically for deploying web applications. |
| **AWS S3** | A web service for storing files. This was used for image uploading in production. |
| **Lean Testing** | Online bug tracker and test case manager. We used this application to create, manage, and run test cases. |
| **Trello** | Online kanban board used for the management and allocation of tasks. |
| **Slack** | Written communication tool. We used both the desktop and mobile version to communicate with each other throughout the duration of this project via group chat and direct message. Useful for sharing code, files, and external links. |
| **WeChat** | Written communication tool that is exclusive to mobile. Used mainly when a more immediate response from a team member was required. |
| **Google Drive, Docs, Sheets** | A file storage and synchronisation service. We used this to share files and collaborate on project documents. |
| **Namecheap** | Used to manage the domain name and DNS management. |
| **ImageMagick** | Open-source image manipulation software used to resize images uploaded by the user. |

# 

# 

# 3. Overall Architecture

## **Class Diagram**

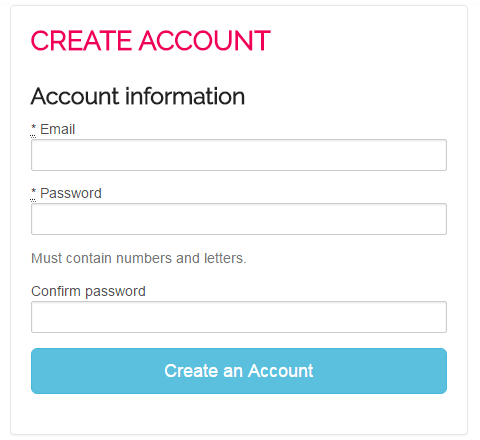
# 

# 4. System Architecture

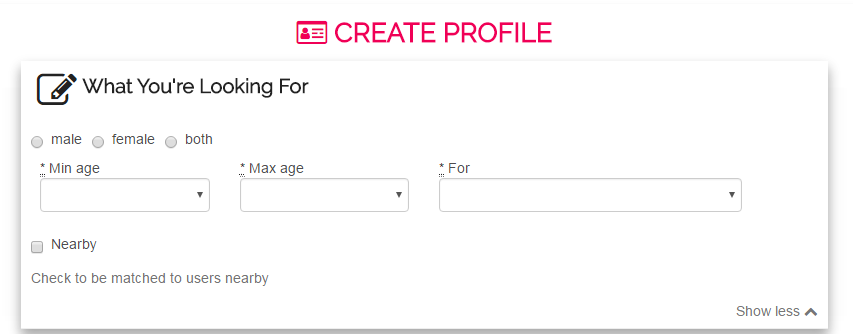
## 4.1 Functionalities/features

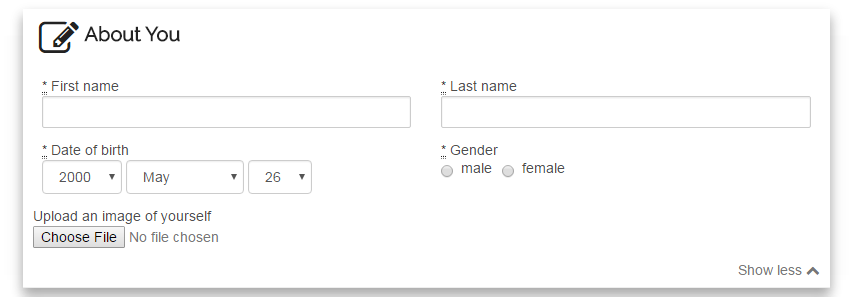
1. User Registration
2. Login (with or without remembering) / Logout
3. Edit profile
4. Changing account settings
5. View Matches
6. Like / unliking other users
7. View your likes / who has liked you
8. Admin: delete users
9. Admin: search users by name

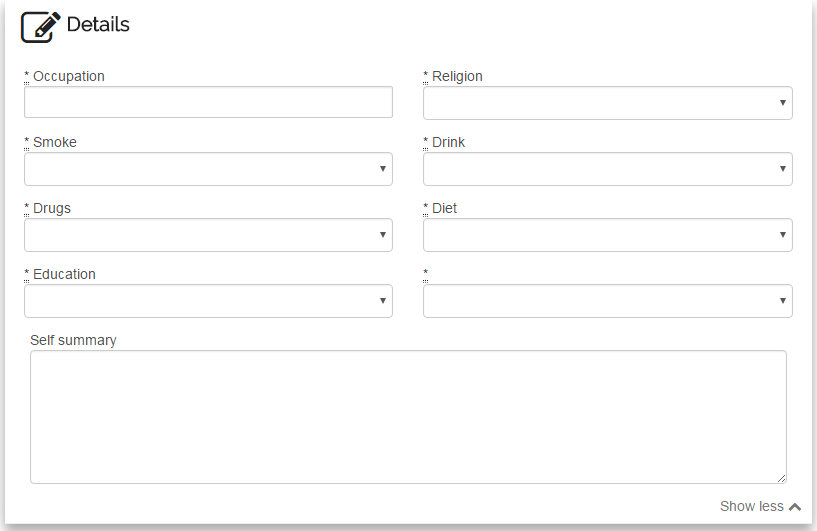
### 4.1.1 User Registration

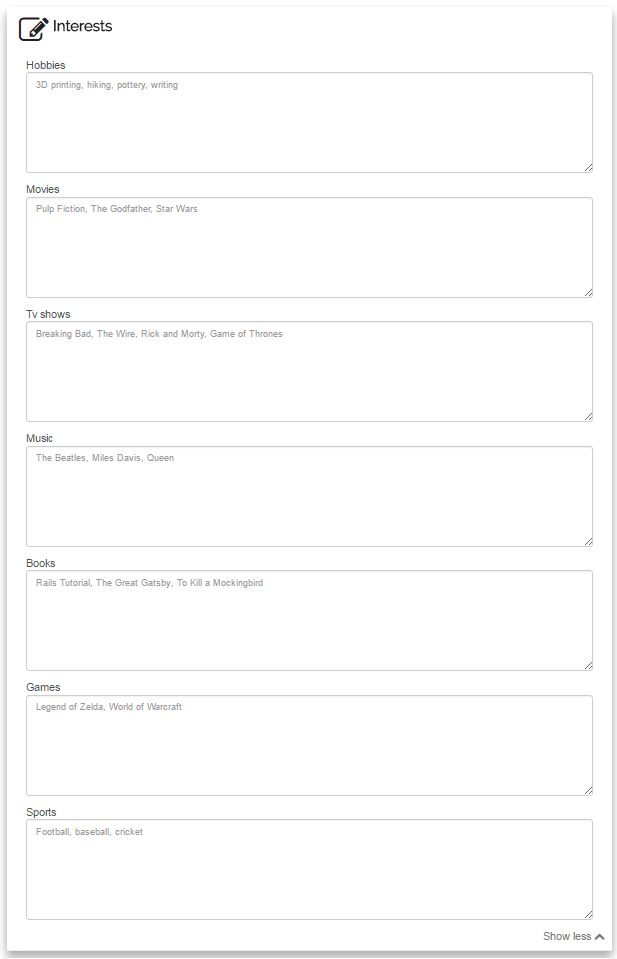


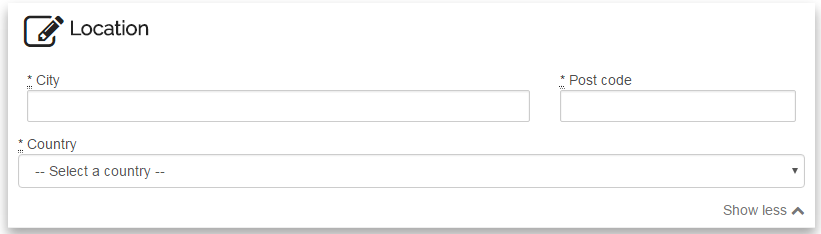
1. User opens MatchMe
2. User clicks “Sign Up” button in header
3. User fills in the email, password, and confirm password fields
4. User clicks “Create an Account” button
   1. If unsuccessful, sign up page re-renders with error message(s)
   2. If successful, user continues to step 5





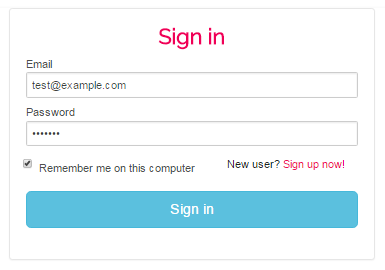


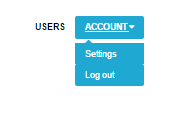




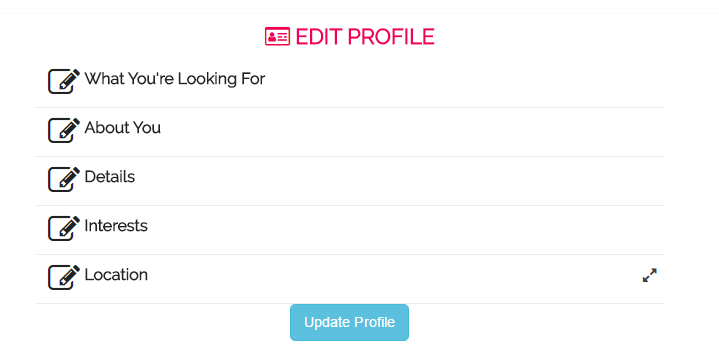
1. User forwarded to “Create Profile” page
2. User the form in with their details
3. User clicks “Create Profile” button
   1. If unsuccessful, page re-renders with relevant error message(s)
   2. If successful, user forwarded to their new profile

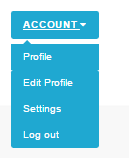
### 4.1.2 Login / Logout



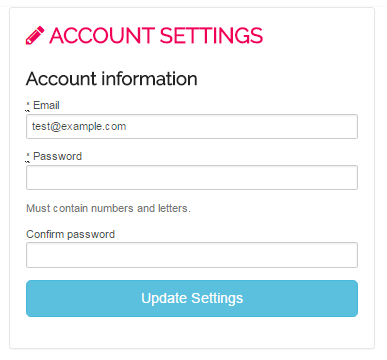
1. User opens MatchMe
2. User clicks “Sign in” link in header
3. User fills in email and password fields
4. If user ticks “Remember Me” checkbox, their session will be remembered, otherwise it will be forgotten once they close their browser
5. User clicks “Sign in”
   1. If successful and they’re a non-admin, they are forwarded to their profile
   2. If successful and they’re an admin, they are forwarded to the Users Index page
   3. If unsuccessful, login page re-renders with error message
6. To logout, user clicks “Account” dropdown in header
7. User clicks “Logout”
8. User is logged out and redirected to homepage

### 4.1.3 Edit profile



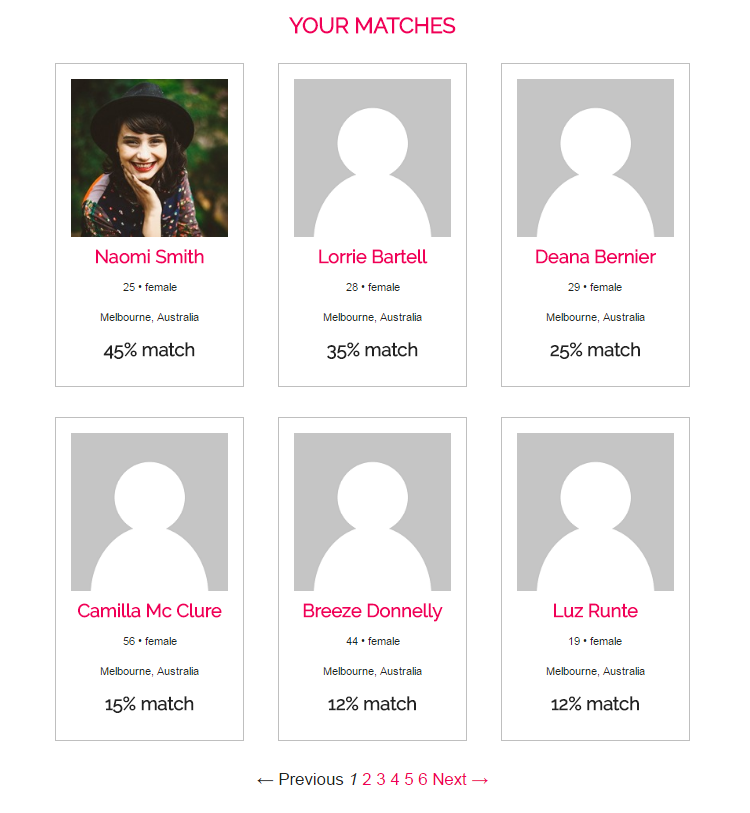
1. User logs in
2. User clicks “Account” drop down and selects “Edit profile”
3. User modifies their profile
4. User clicks “Update Profile”
   1. If successful, they are forwarded to their profile and a confirmation message appears
   2. If unsuccessful, page re-renders with relevant error message(s)

### 4.1.4 Change account settings



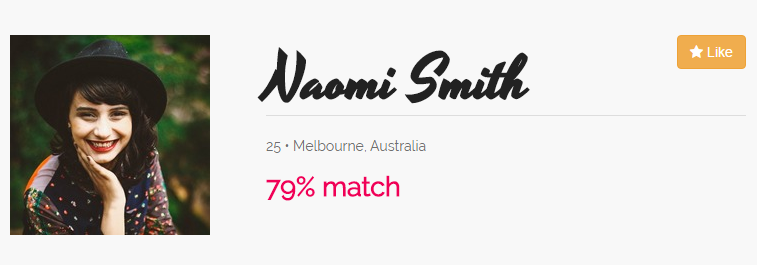
1. User logs in
2. User clicks “Account” drop down and selects “Settings”
3. User modifies their email and/or their password
4. User clicks “Update Settings”
   1. If successful, they are forwarded to their profile and a confirmation message appears
   2. If unsuccessful, page re-renders with relevant error message(s)

### 4.1.5 View Matches



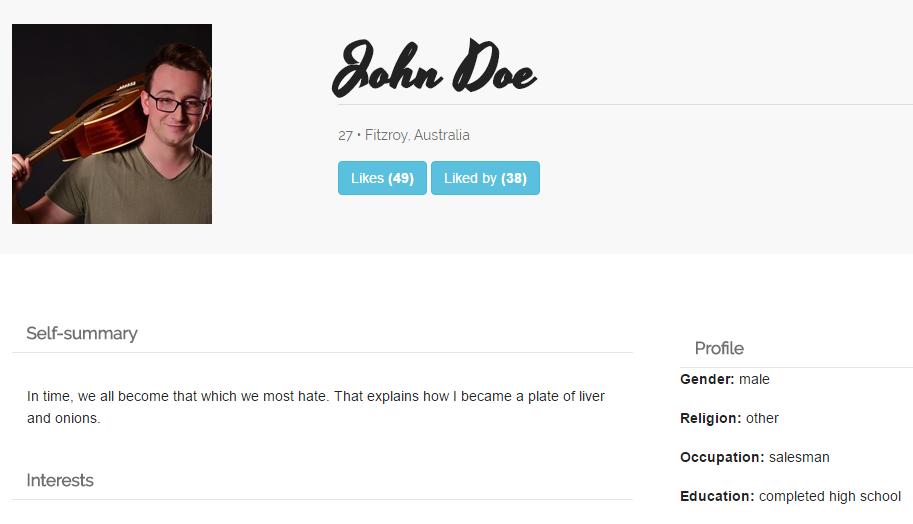
1. User logs in
2. User clicks “Matches” link in header
3. Page renders with all matches ordered by match percent

### 4.1.6 Like / unlike other user

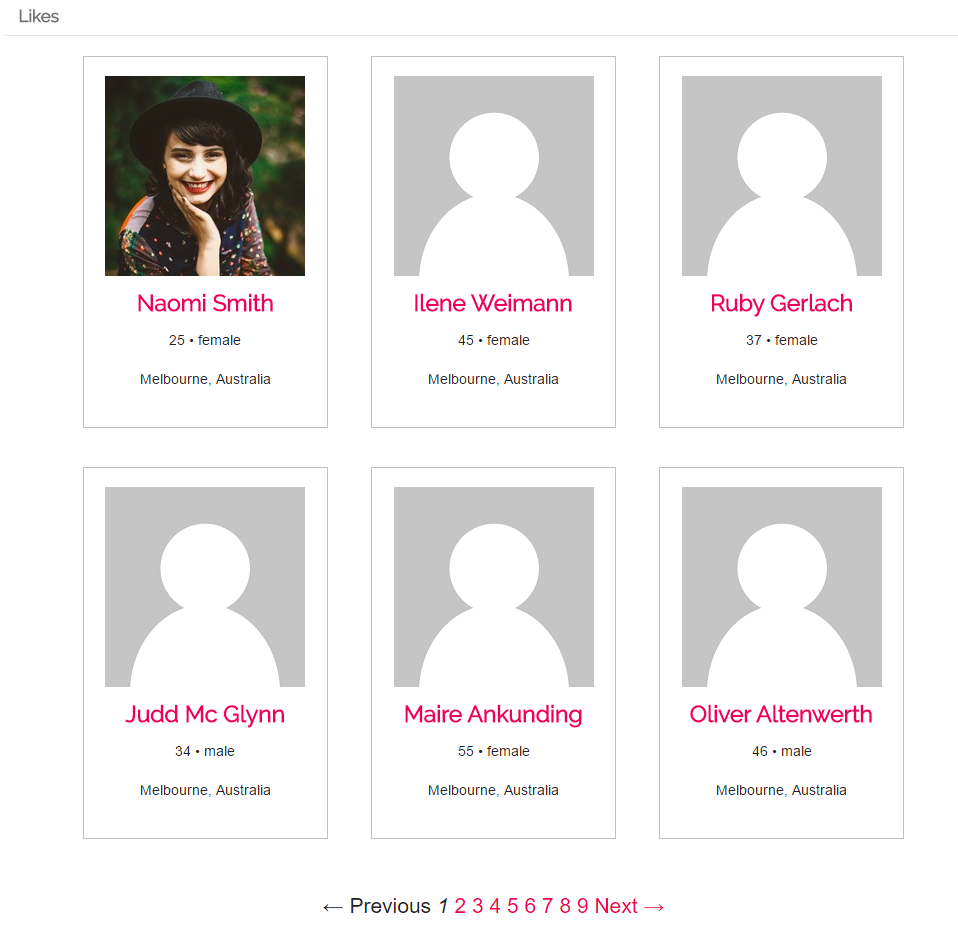


1. User logs in
2. User navigates to another user’s profile
3. User clicks “Like” button
   1. Like button then changes to a “Unlike” button
4. User clicks “Unlike” button
   1. Button then changes back to “Like” button

### 4.1.7 View your likes / who has liked you



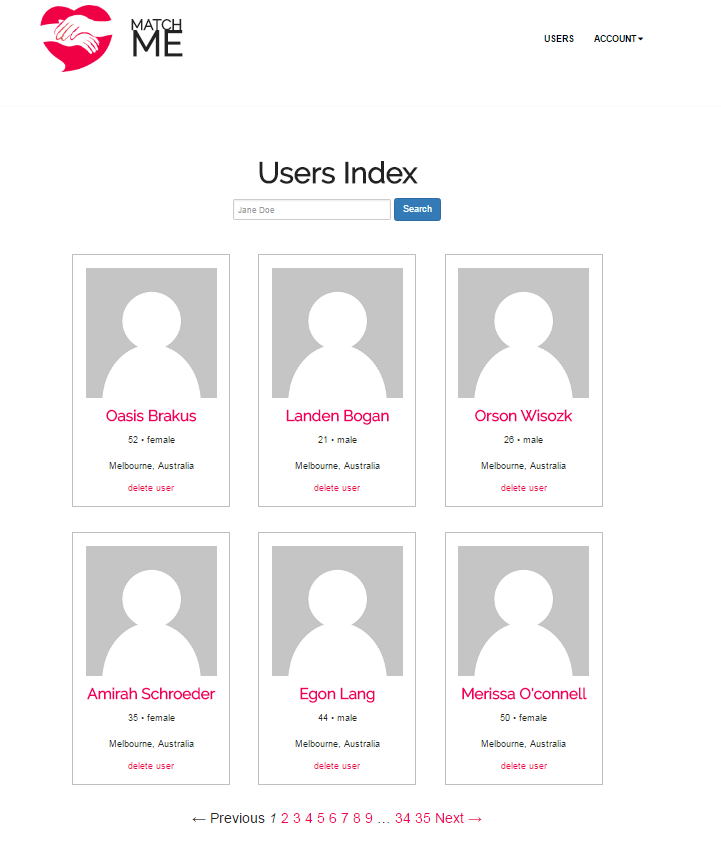
1. User logs in
2. User opens their profile page
3. User clicks “Likes” button
4. Page with users the user has liked opens



1. Click “Like by” button
2. Page with users who have liked the user opens

### 4.1.8 Admin: delete user

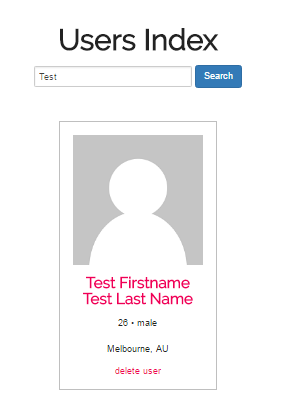
1. Admin logs in
2. Admin is forwarded to the Users Index page



1. Admin clicks “delete user” link on any given user
2. Confirmation message appears
3. Admin clicks “Confirm”
4. Users index reloads, the user is removed from list as it no longer exists in the database

### 4.1.9 Admin: search users by name

1. Admin logs in
2. Admin is forwarded to the Users Index page
3. Admin enters all or part of a name in the search field
4. Admin clicks “Search” button
5. The page now only displays users with those characters in their name



# 5. Database Architecture

We decided to use PostgreSQL because it can handle larger amounts of data than SQLite, and because Heroku uses Postgres in production by default. Postgres is also pre-installed onto Cloud9 workspaces. Because of this, choosing to use Postgres saved us a lot of time by reducing the amount of configuration required for both our development and production environments. To make the database work with Rails, we installed the [pg gem](https://deveiate.org/code/pg/) into the application, which acts as an interface between the Rails and Postgres.

## 5.1 Database Diagram

# 

# 6. Implementation Instructions

## 6.2 What you’ll need

* Ruby
* Ruby on Rails
* PostgreSQL

## 6.1 Getting Started

Clone the repo and then install the needed gems:

$ bundle install --without production

Start Postgresql:

$ sudo service postgresql start

Install ImageMagick:

sudo apt-get update  
sudo apt-get install imagemagick --fix-missing

Migrate and seed the database:

$ rails db:migrate  
$ rails db:seed

Run the test suite:

$ rails test

Run the app in a local server:

$ rails server

## 6.3 Detailed Instructions

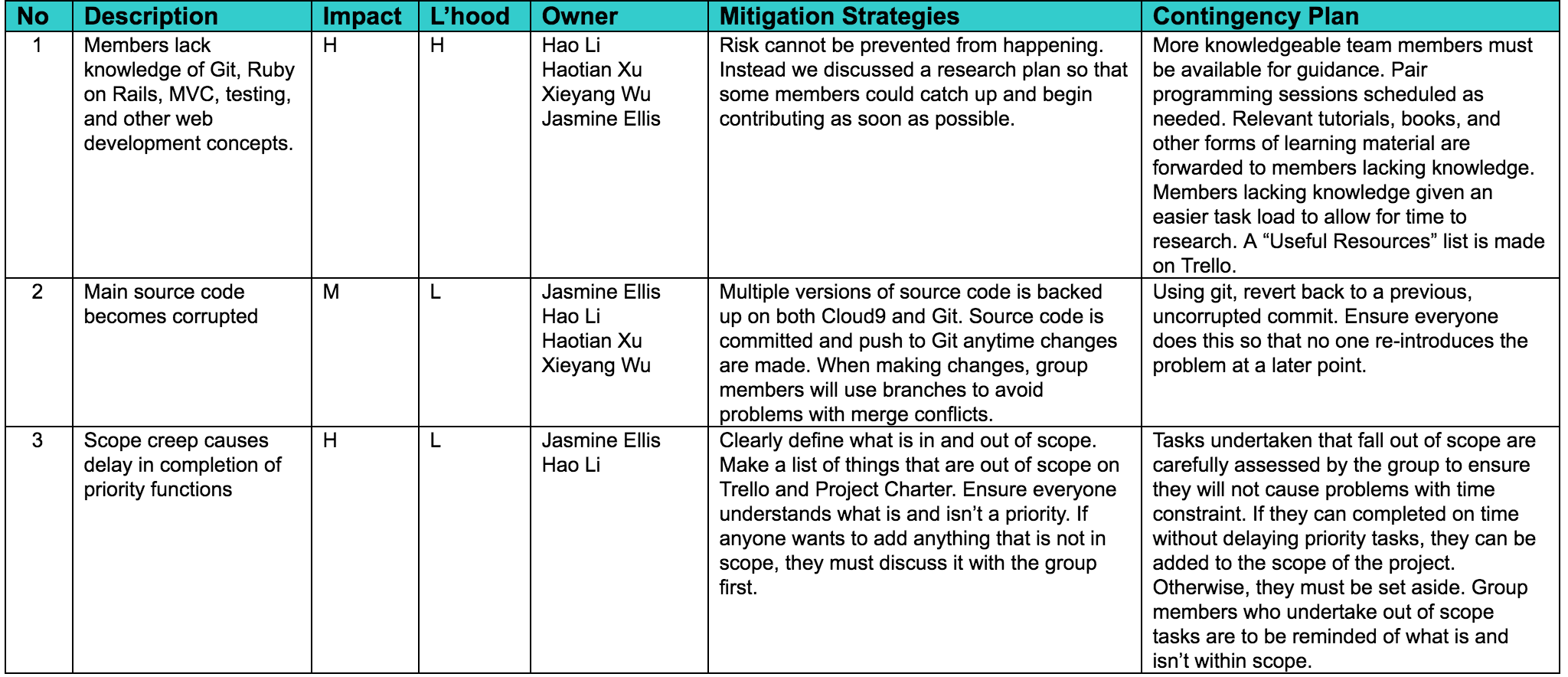
See [Tools Setup Guide](https://github.com/ellisjas/matchme/blob/master/documentation/Tools%20Setup%20Guide.pdf) for a more detailed implementation instructions.

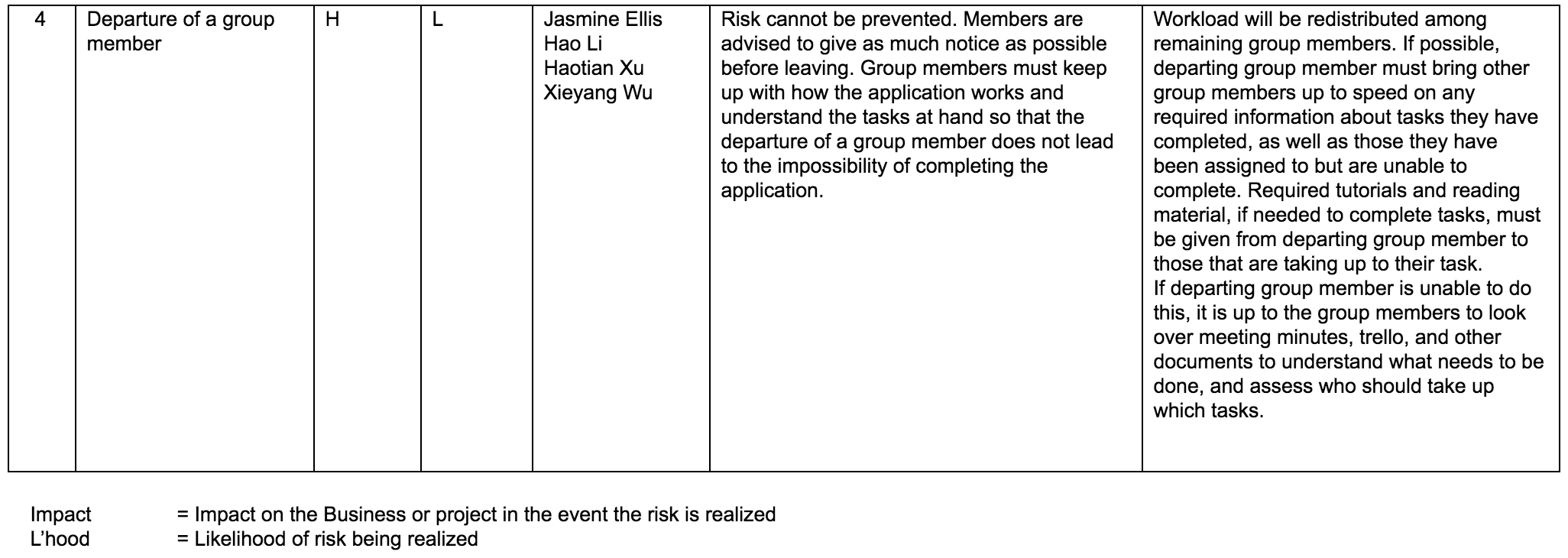
# 7. Summary of test results

We used Lean Testing testing suite to manage and run our test cases. The result of our most recent test run can be found in the the [Testing Register](https://github.com/ellisjas/matchme/blob/master/documentation/Testing%20Register.pdf) document.

# 8. Known Issues & Risks

## 8.1 Risks

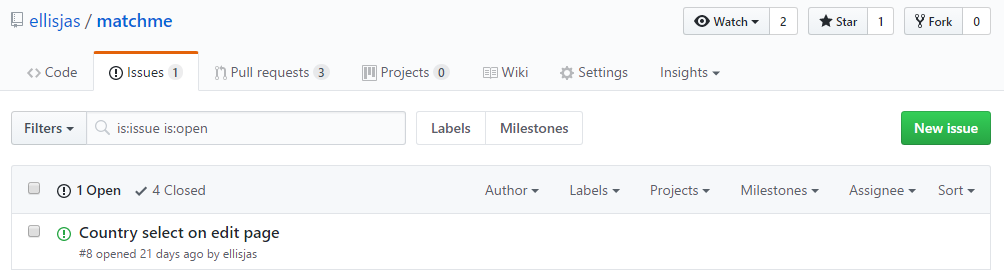




Potential and realised risks were documented in the [Risks Register](https://github.com/ellisjas/matchme/blob/master/documentation/Risk%20Register.pdf).

## 8.2 Issues

We documented issues and bugs both on Trello (via “Bug report” column) and Github (on “Issues” page of repository).



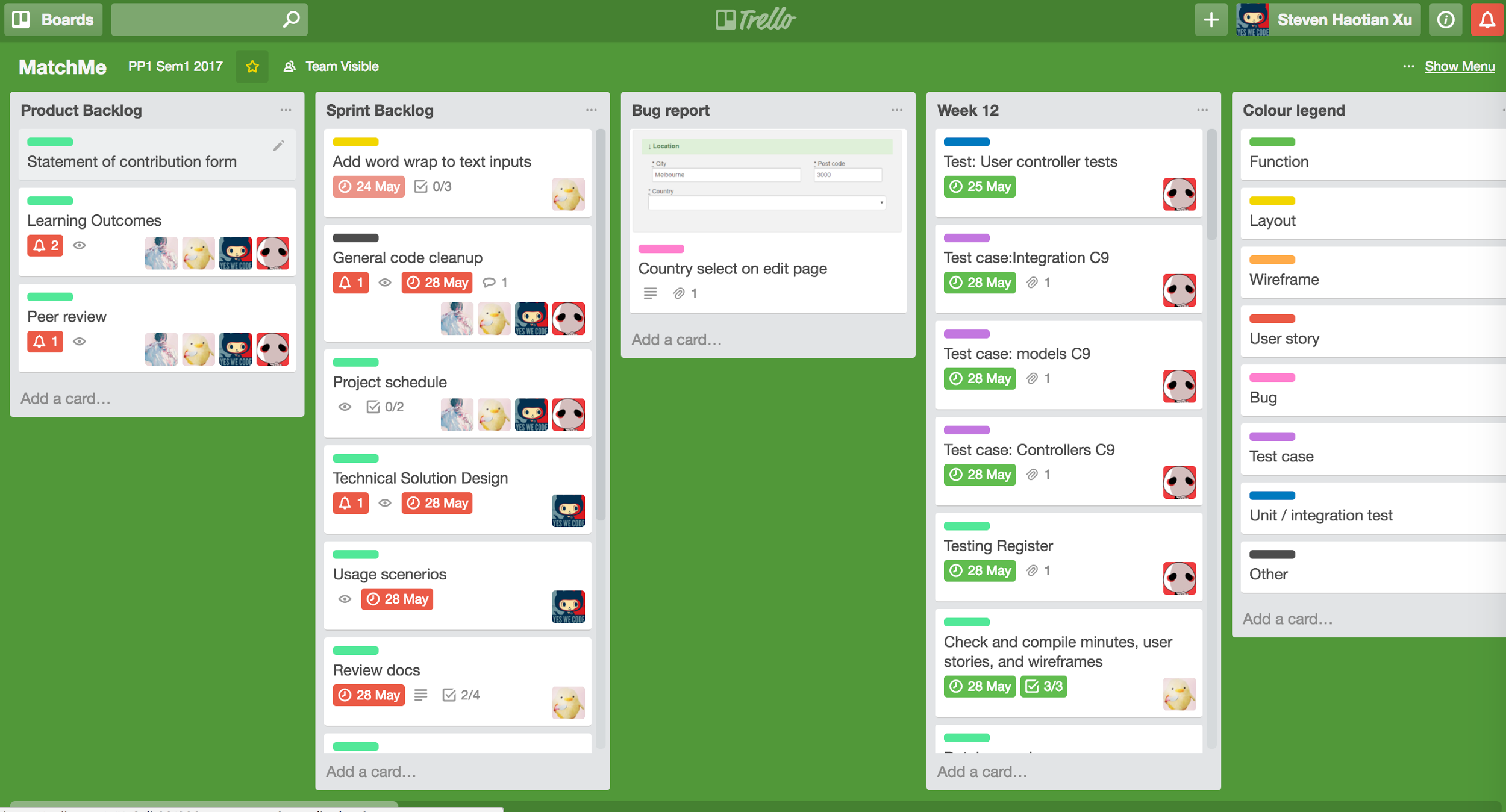
### 8.2.1 Known Issues

* The first time a user edits their profile, the country select field is blank even though they have previously selected a country while registering.
* In the mobile version, the account menu is not accessible.
* If a user registers an account but does not complete their profile, the admin cannot search for them by name.
* If a user enters interests in the wrong format, the will not be properly matched by interests.
* If a user enters long text with no breaks or spaces, the test will overflow from the layout.

# 

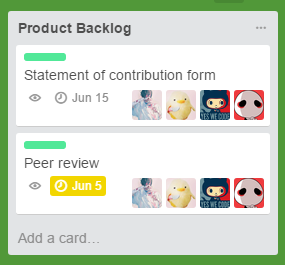
# 9. Appendix

## 9.1 Trello



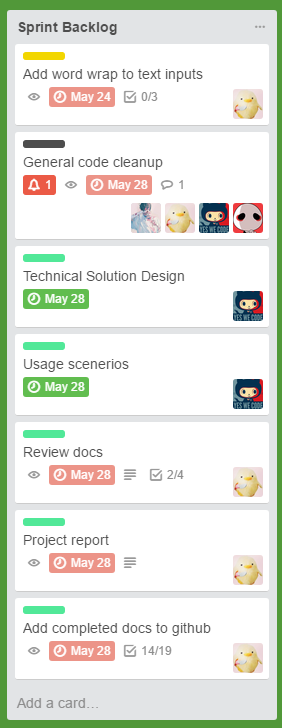
We used Trello to manage our project. Each task was given it’s own card and placed in the relevant column. Tasks were assigned to group members and given due dates. Cards were also used for other miscellaneous things such as resources and the colour legend.

### 9.1.1 Product backlog



At the start of the project, we created a large list of all anticipated tasks for the project and added them to the Product Backlog. These represented all tasks that were within scope. Often items in this section were broken down into smaller tasks.

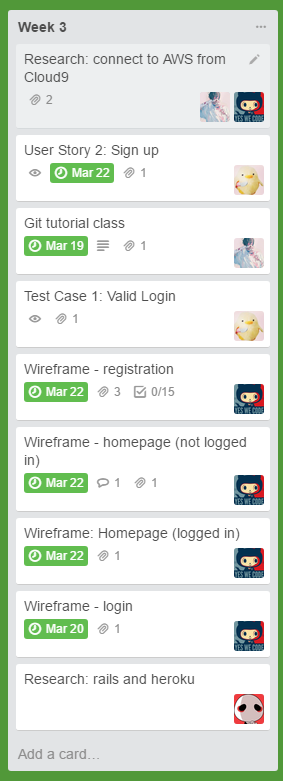
### 9.1.2 Sprint Backlog



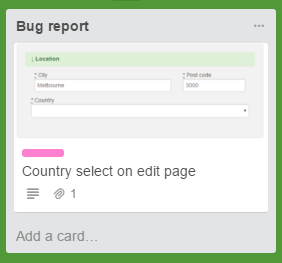
When we were ready to commence a task, we moved the card from the Product Backlog to the Sprint Backlog. All the items in the Sprint Backlog represented items to be completed until the commencement of the next Sprint. Each Sprint went for one week. When an item was added to the Sprint Backlog, it was assigned to a group member(s) and given a due date.

### 9.1.3 Weeks 2 - 13

As we finished tasks, we moved them into the week they were completed. Note: our group was formed in week 2, and as such week 1 is not included.



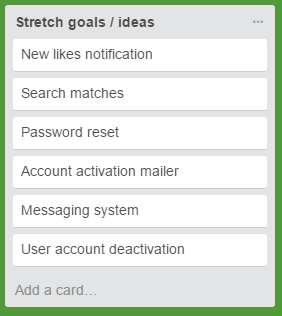
### 9.1.4 Bug report



As the name suggests, this column was used to report bugs.

### 

### 9.1.5 Stretch Goals / Ideas



All out of scope items were placed here. These would be undertaken if all items in Product Backlog were completed ahead of schedule.

### 9.1.6 Resources

Any useful links or documents were placed here.

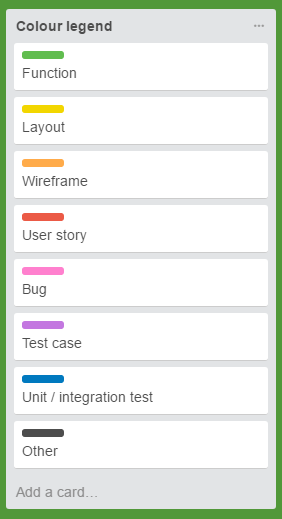
### 

### 9.1.7 Quality Assurance

### 

If feedback is required, items were placed in the Quality Assurance column to alert other group members that the item needs review.

### 9.1.8 Colour Legend



To clarify the meaning behind each label. This made it easier to organise and understand cards.

# 10. Coding Standard

* Classes named using CamelCase, functions and variables named using snake\_case
* SCREAMING\_SNAKE\_CASE used for constants
* 2 space indentation, hard tabs
* Comments added to methods to explain what they do
* Most variable, class, and method names are easy to understand / relevant to their usage
* All model and controller classes thoroughly tested
* Proper indentation
* Use of string interpolation over concatenation
* Use of source control