MatchMe

Technical Solution Design

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

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.

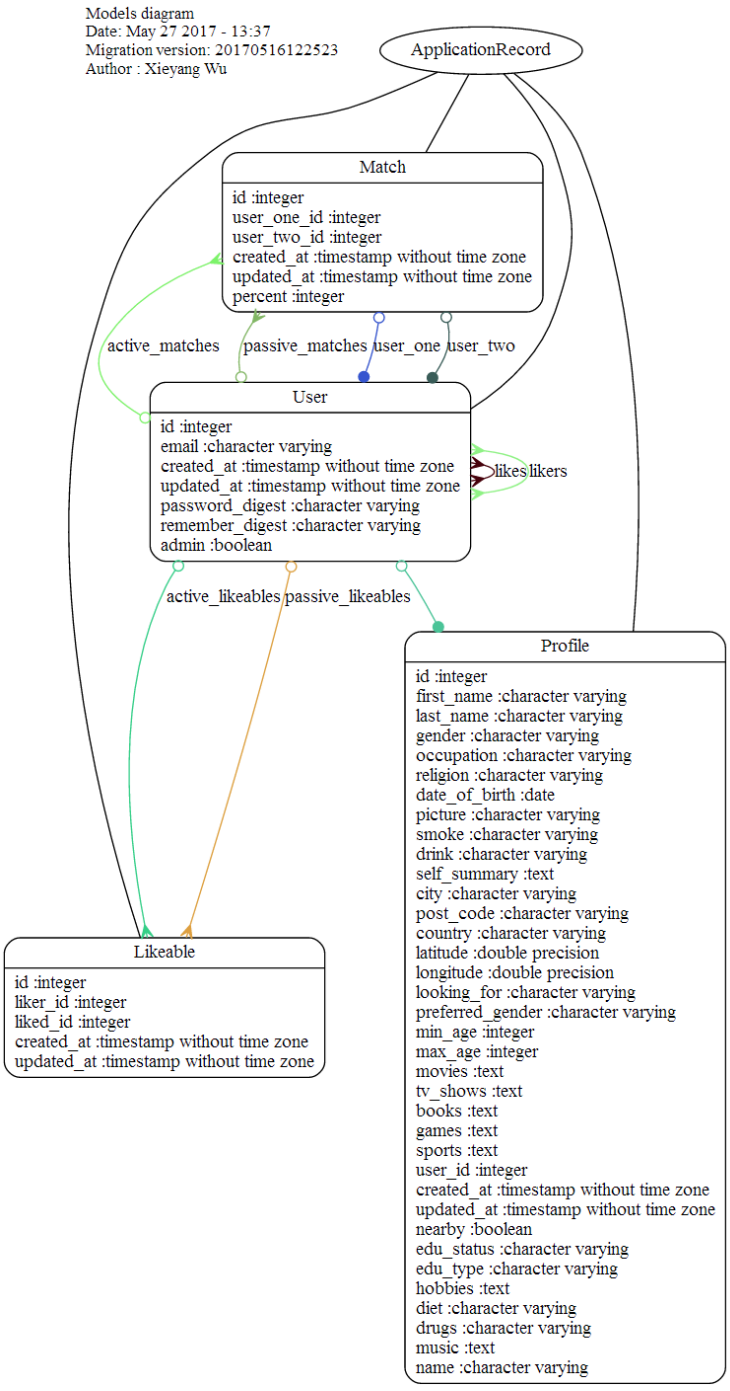
2. Technical Environment

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

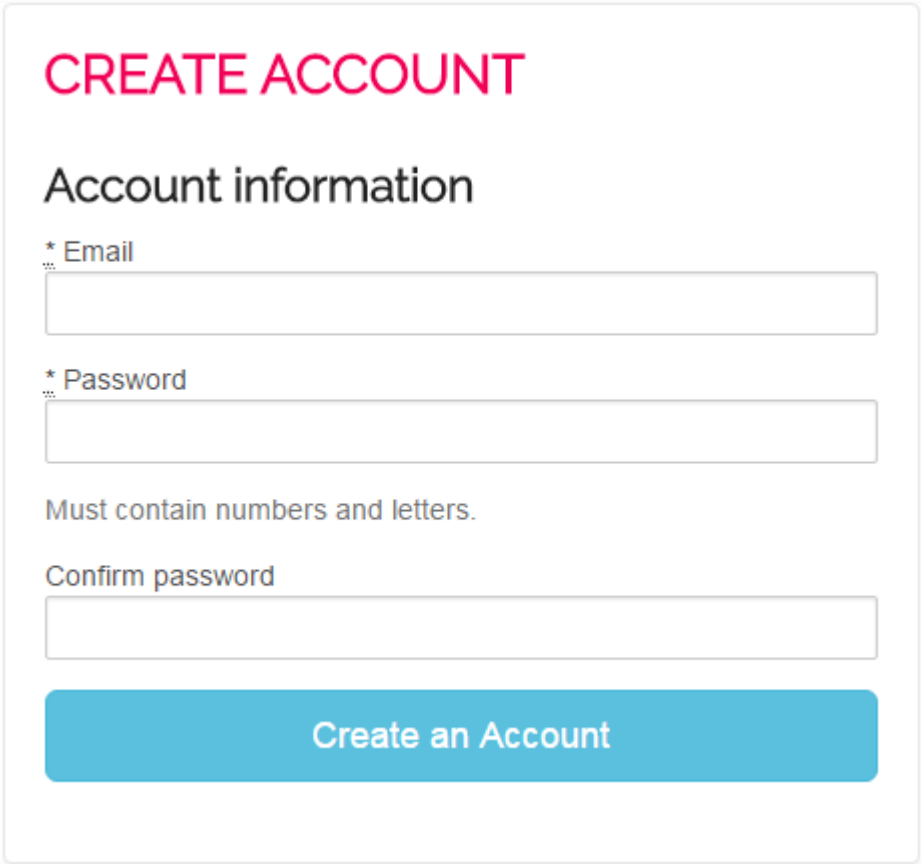


4. System Architecture

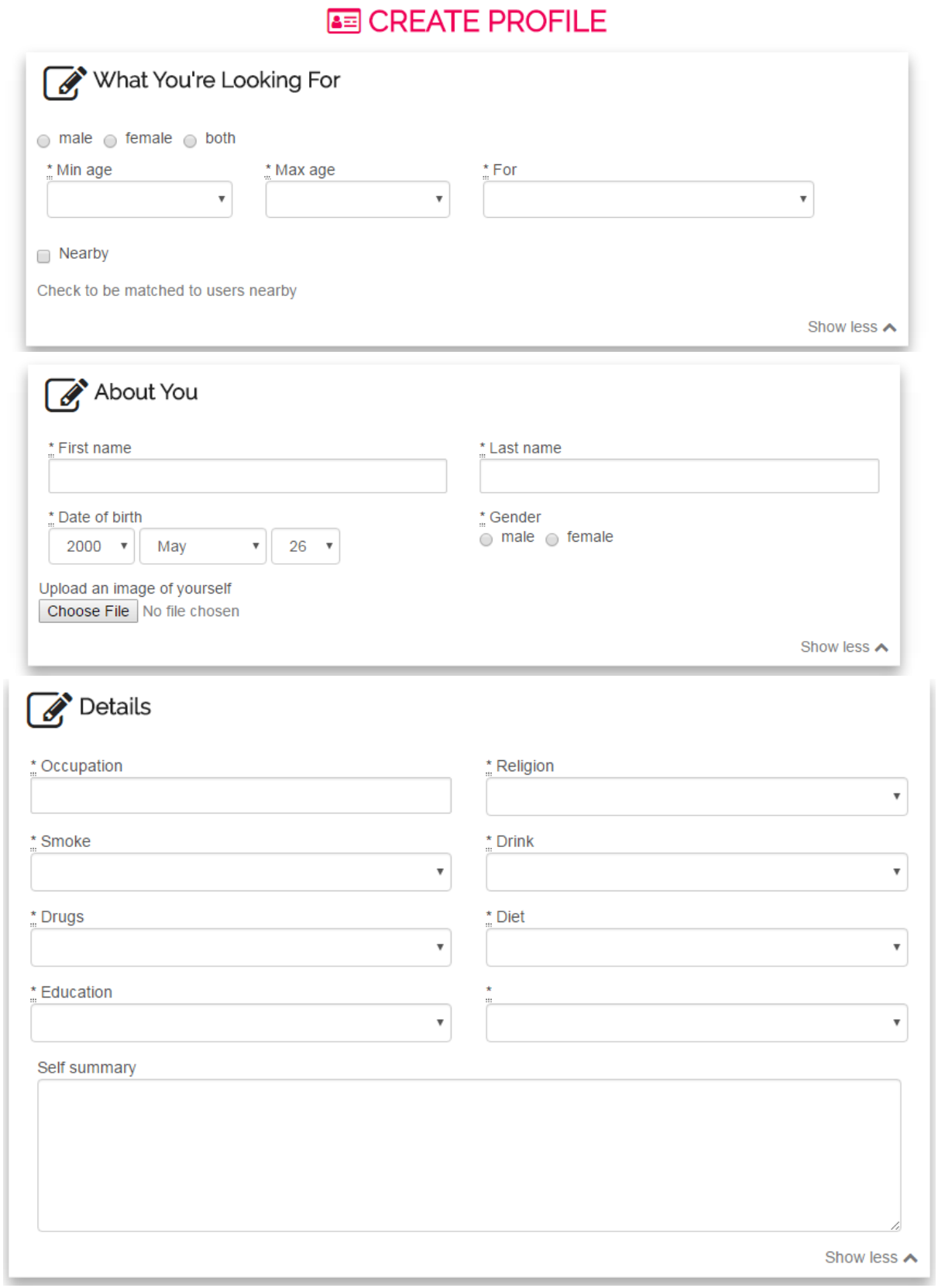
4.1 Functionalities/features

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



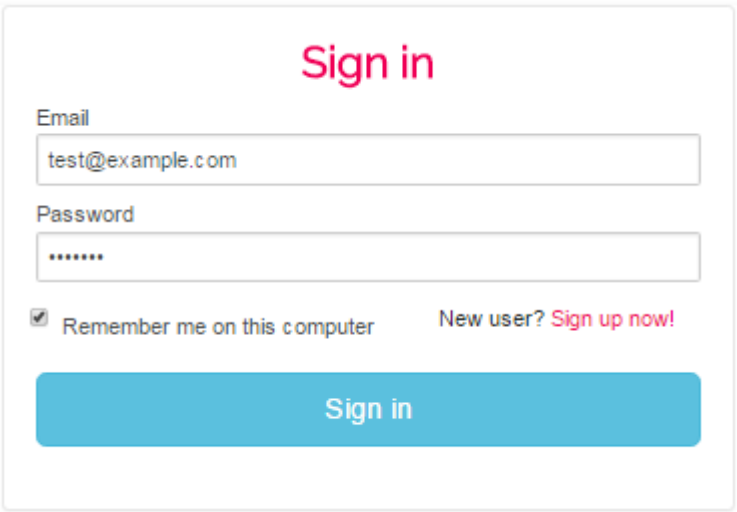
1. User opens MatchMe
2. User clicks “Sign Up” button in header
3. User fills in the email, password, and confirm password fields
4. User clicks “Create an Account” button
    a. If unsuccessful, sign up page re-renders with error message(s)
    b. If successful, user continues to step 5


CREATE PROFILE
What You're Looking For
male
female
both
* Min age
* Max age
* For
Nearby
Check to be matched to users nearby
Show less
About You
* First name
* Last name
* Date of birth
2000
May
26
* Gender
male
female
Upload an image of yourself
Choose File
No file chosen
Show less
Details
* Occupation
* Religion
* Smoke
* Drink
* Drugs
* Diet
* Education
*
Self summary
Show less

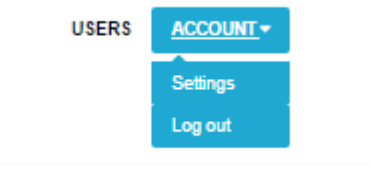

Interests
Hobbies
3D printing, hiking, pottery, writing
Movies
Pulp Fiction, The Godfather, Star Wars
Tv shows
Breaking Bad, The Wire, Rick and Morty, Game of Thrones
Music
The Beatles, Miles Davis, Queen
Books
Rails Tutorial, The Great Gatsby, To Kill a Mockingbird
Games
Legend of Zelda, World of Warcraft
Sports
Football, baseball, cricket
Show less

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

4.1.2 Login / Logout

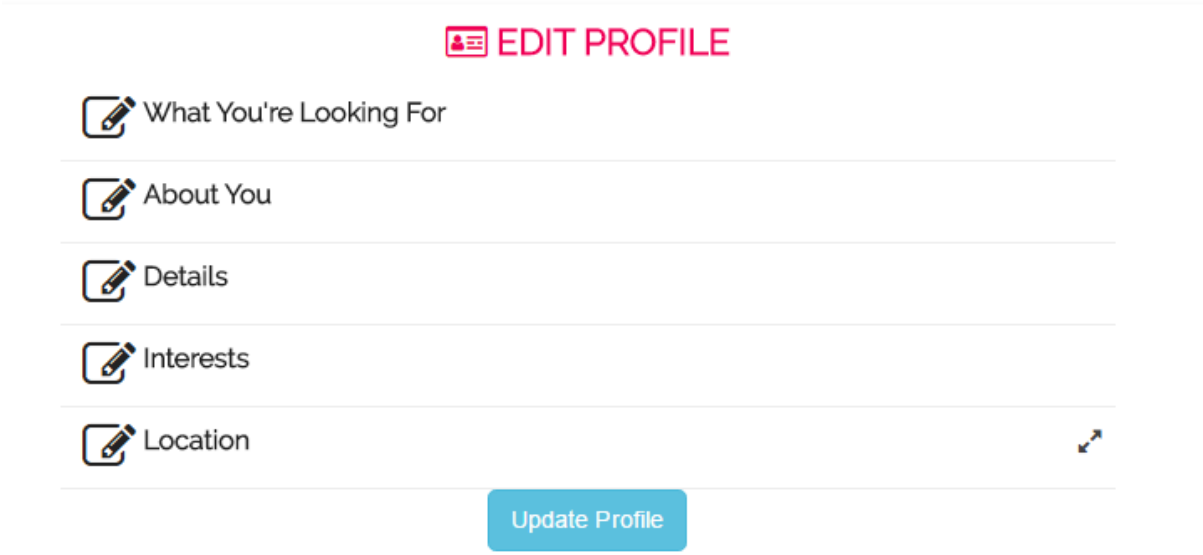


1. User opens MatchMe
2. User clicks “Sign in” link in header
3. User fills in email and password fields
4. If user ticks “Remember Me” checkbox, their session will be remembered, otherwise it will be forgotten once they close their browser
5. User clicks “Sign in”
    a. If successful and they’re a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

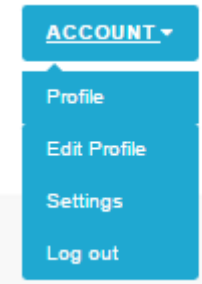


6. To logout, user clicks “Account” dropdown in header
7. User clicks “Logout”
8. User is logged out and redirected to homepage

4.1.3 Edit profile

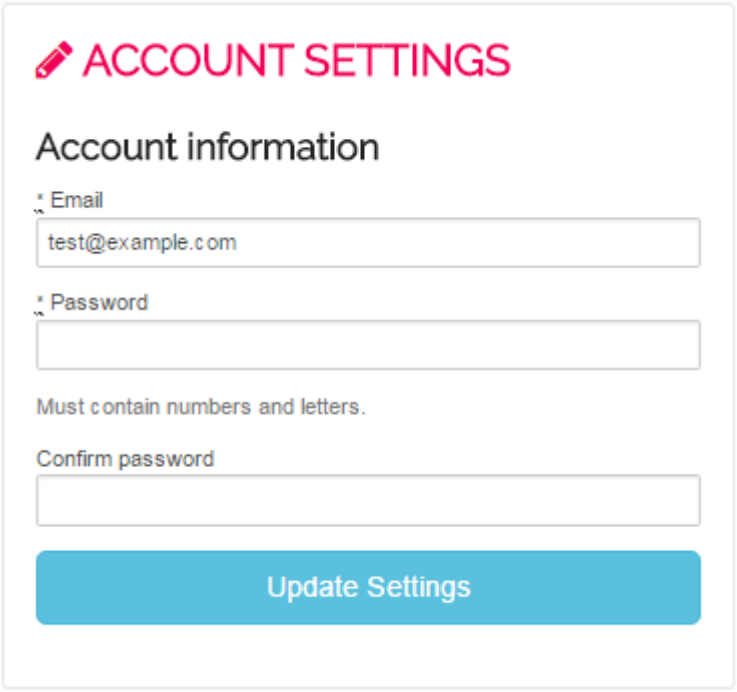


1. User logs in
2. User clicks “Account” drop down and selects “Edit profile”



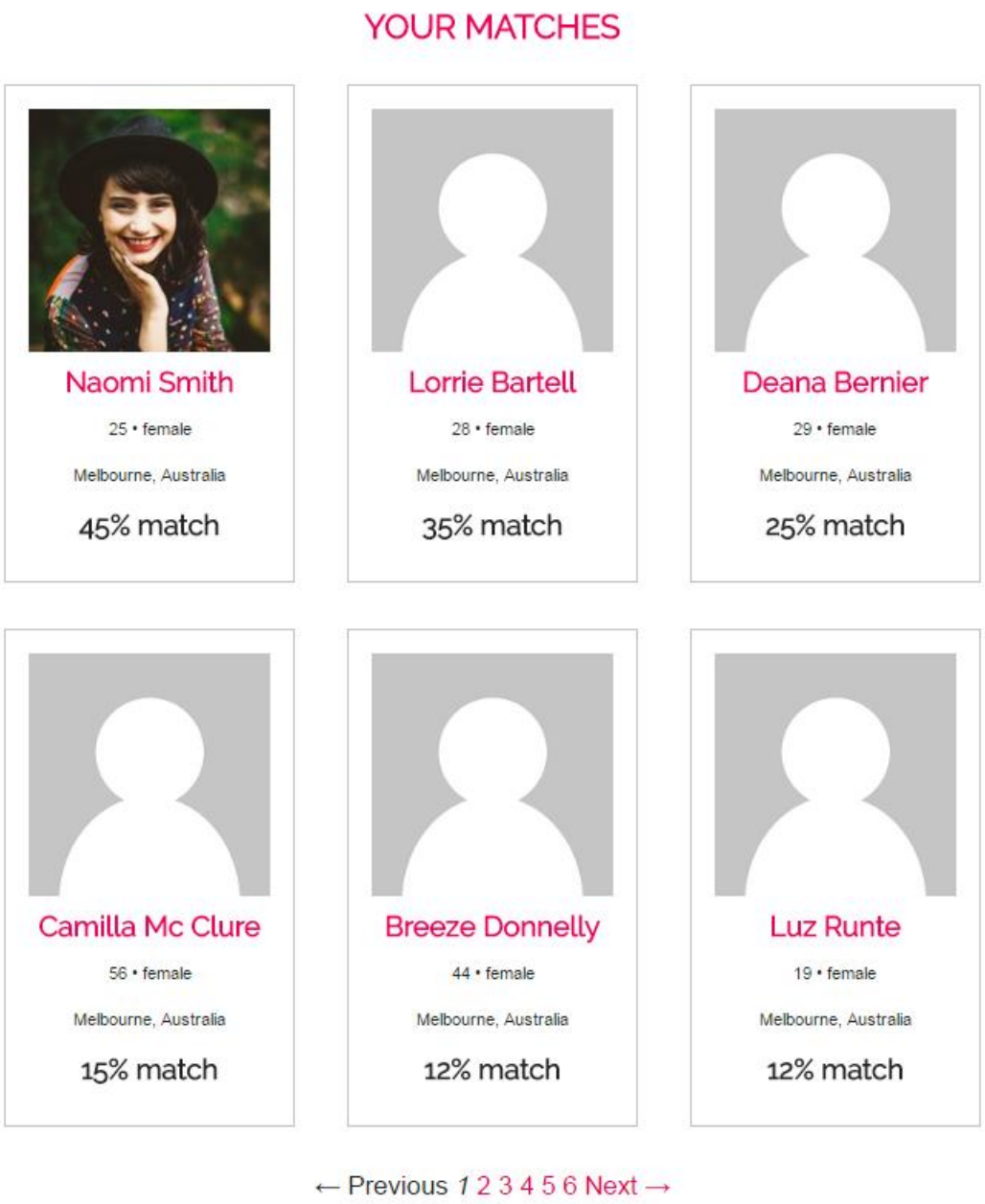
3. User modifies their profile
4. User clicks “Update Profile”
    a. If successful, they are forwarded to their profile and a confirmation message appears
    b. If unsuccessful, page re-renders with relevant error message(s)

4.1.4 Change account settings



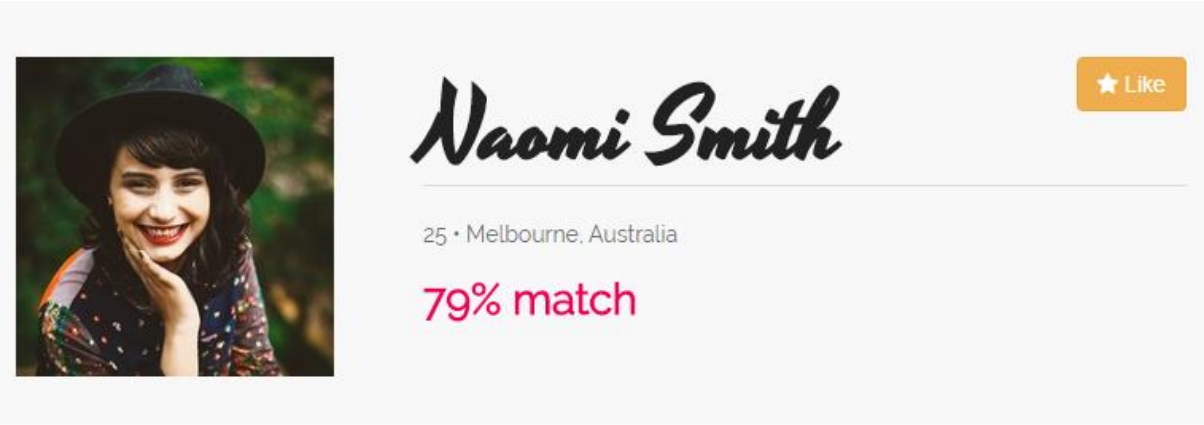
1. User logs in
2. User clicks “Account” drop down and selects “Settings”
3. User modifies their email and/or their password
4. User clicks “Update Settings”
    a. If successful, they are forwarded to their profile and a confirmation message appears
    b. If unsuccessful, page re-renders with relevant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

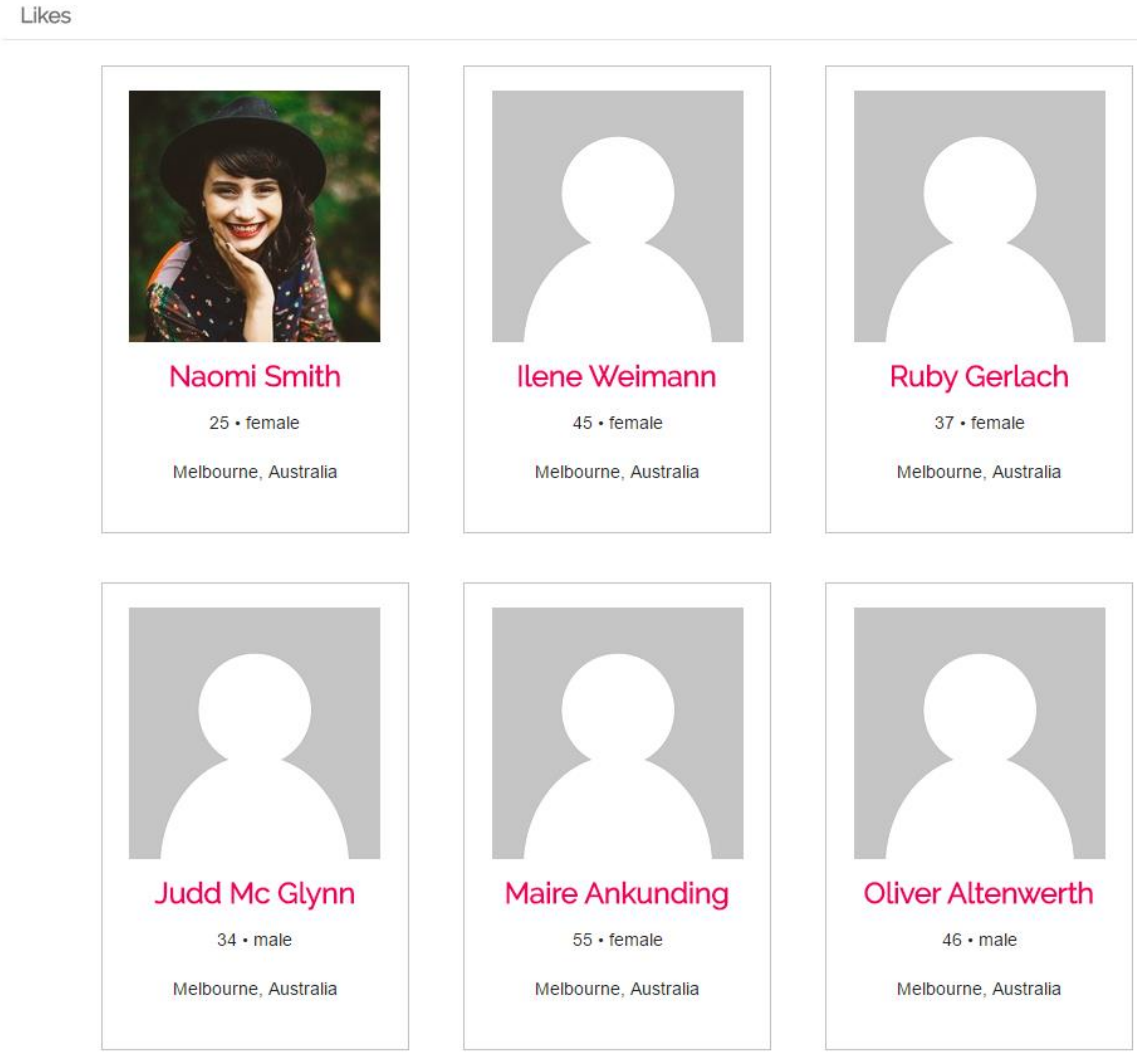
4.1.6 Like / unlike other user



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

4.1.7 View your likes / who has liked you

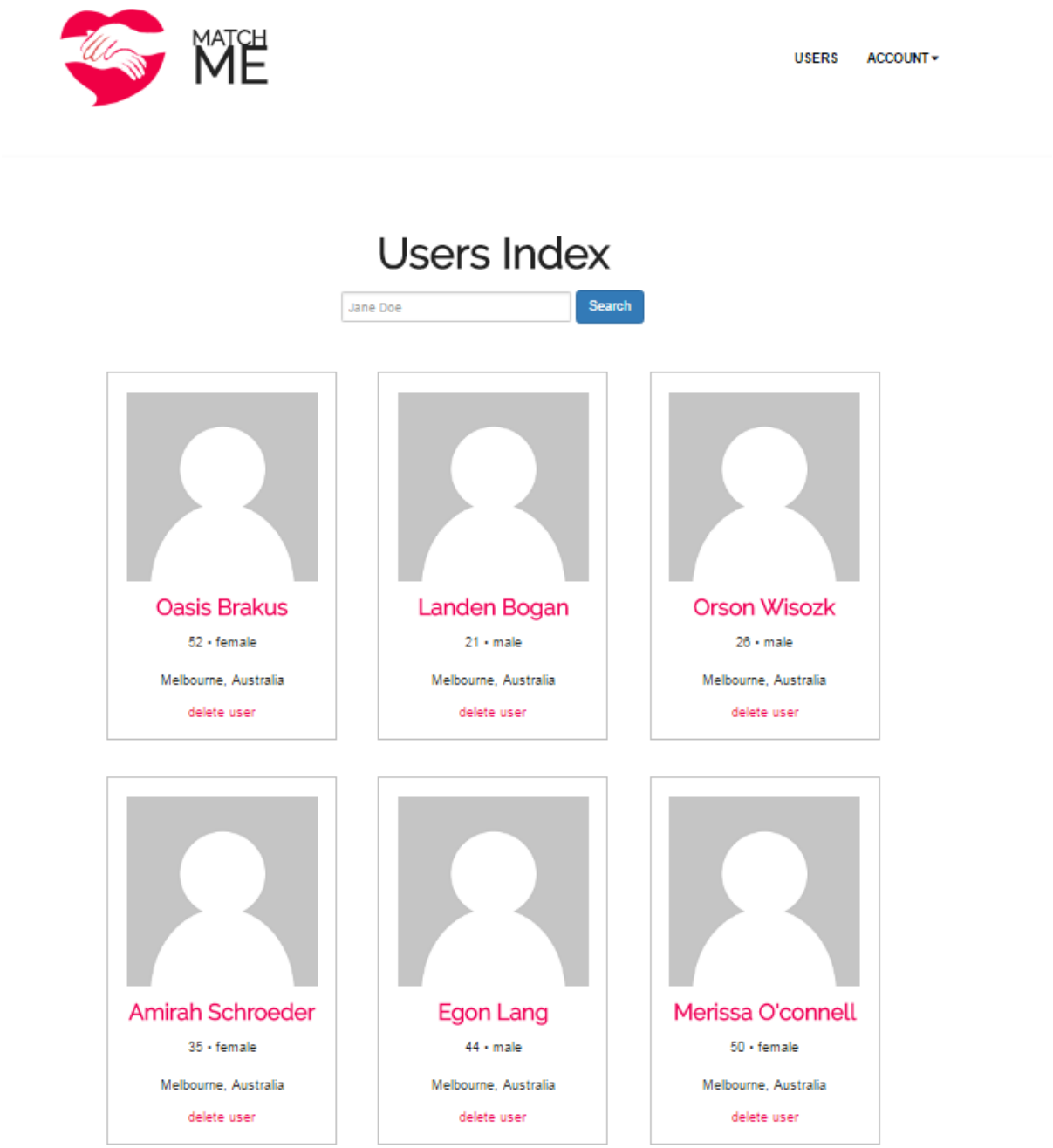
1. User logs in
2. User opens their profile page
3. User clicks "Likes" button
4. Page with users the user has liked opens



5. Click “Like by” button
6. 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

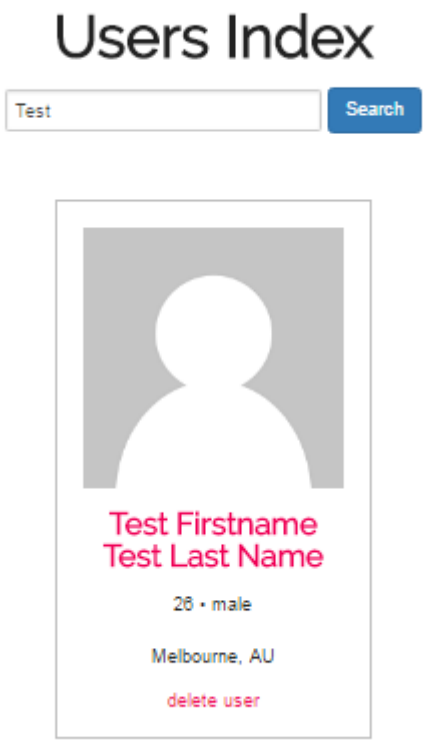


3. Admin clicks “delete user” link on any given user
4. Confirmation message appears
5. Admin clicks “Confirm”
6. Users index reloads, the user is removed from list as it no longer exists in the database

4.1.9 Admin: search users by name

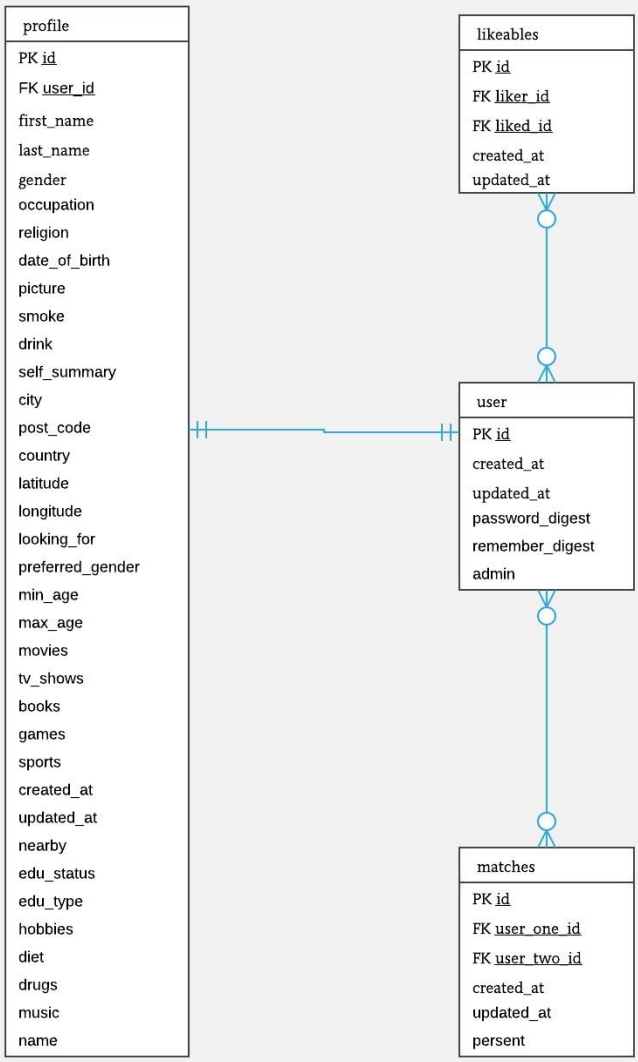
1. Admin logs in
2. Admin is forwarded to the Users Index page
3. Admin enters all or part of a name in the search field
4. Admin clicks “Search” button

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 into the application, which acts as an interface between the Rails and Postgres.


profile
PK id
FK user_id
first_name
last_name
gender
occupation
religion
date_of_birth
picture
smoke
drink
self_summary
city
post_code
country
latitude
longitude
looking_for
preferred_gender
min_age
max_age
movies
tv_shows
books
games
sports
created_at
updated_at
nearby
edu_status
edu_type
hobbies
diet
drugs
music
name
likeables
PK id
FK liker_id
FK liked_id
created_at
updated_at
user
PK id
created_at
updated_at
password_digest
remember_digest
admin
matches
PK id
FK user_one_id
FK user_two_id
created_at
updated_at
persent

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 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 document.

8. Known Issues & Risks

8.1 Risks

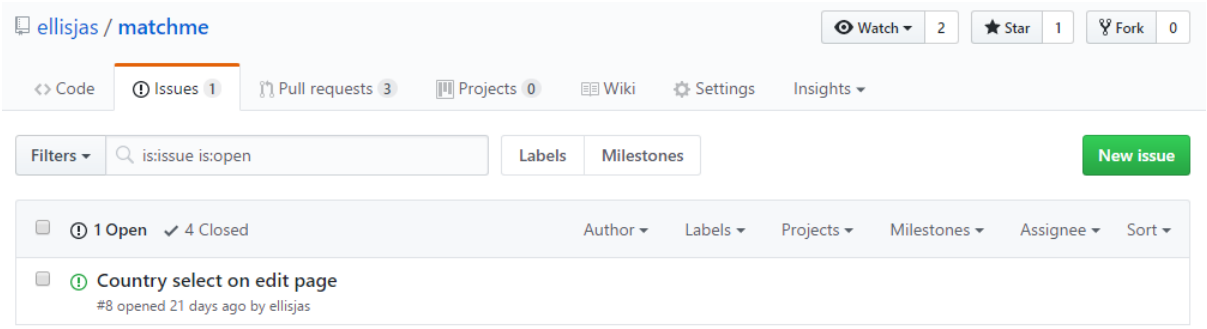
No	Description	Impact	L'hood	Owner	Mitigation Strategies	Contingency Plan
1	Members lack knowledge of Git, Ruby on Rails, MVC, testing, and other web development concepts.	H	H	Hao Li Haotian Xu Xieyang Wu Jasmine Ellis	Risk cannot be prevented from happening. Instead we discussed a research plan so that some members could catch up and begin contributing as soon as possible.	More knowledgeable team members must be available for guidance. Pair programming sessions scheduled as needed. Relevant tutorials, books, and other forms of learning material are forwarded to members lacking knowledge. Members lacking knowledge given an easier task load to allow for time to research. A "Useful Resources" list is made on Trello.
2	Main source code becomes corrupted	M	L	Jasmine Ellis Hao Li Haotian Xu Xieyang Wu	Multiple versions of source code is backed up on both Cloud9 and Git. Source code is committed and push to Git anytime changes are made. When making changes, group members will use branches to avoid problems with merge conflicts.	Using git, revert back to a previous, uncorrupted commit. Ensure everyone does this so that no one re-introduces the problem at a later point.
3	Scope creep causes delay in completion of priority functions	H	L	Jasmine Ellis Hao Li	Clearly define what is in and out of scope. Make a list of things that are out of scope on Trello and Project Charter. Ensure everyone understands what is and isn't a priority. If anyone wants to add anything that is not in scope, they must discuss it with the group first.	Tasks undertaken that fall out of scope are carefully assessed by the group to ensure they will not cause problems with time constraint. If they can completed on time without delaying priority tasks, they can be added to the scope of the project. Otherwise, they must be set aside. Group members who undertake out of scope tasks are to be reminded of what is and isn't within scope.
4	Departure of a group member	H	L	Jasmine Ellis Hao Li Haotian Xu Xieyang Wu	Risk cannot be prevented. Members are advised to give as much notice as possible before leaving. Group members must keep up with how the application works and understand the tasks at hand so that the departure of a group member does not lead to the impossibility of completing the application.	Workload will be redistributed among remaining group members. If possible, departing group member must bring other group members up to speed on any required information about tasks they have completed, as well as those they have been assigned to but are unable to complete. Required tutorials and reading material, if needed to complete tasks, must be given from departing group member to those that are taking up to their task. If departing group member is unable to do this, it is up to the group members to look over meeting minutes, trello, and other documents to understand what needs to be done, and assess who should take up which tasks.

Impact = Impact on the Business or project in the event the risk is realized
L'hood = Likelihood of risk being realized

Potential and realised risks were documented in the Risks Register.

8.2 Issues

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

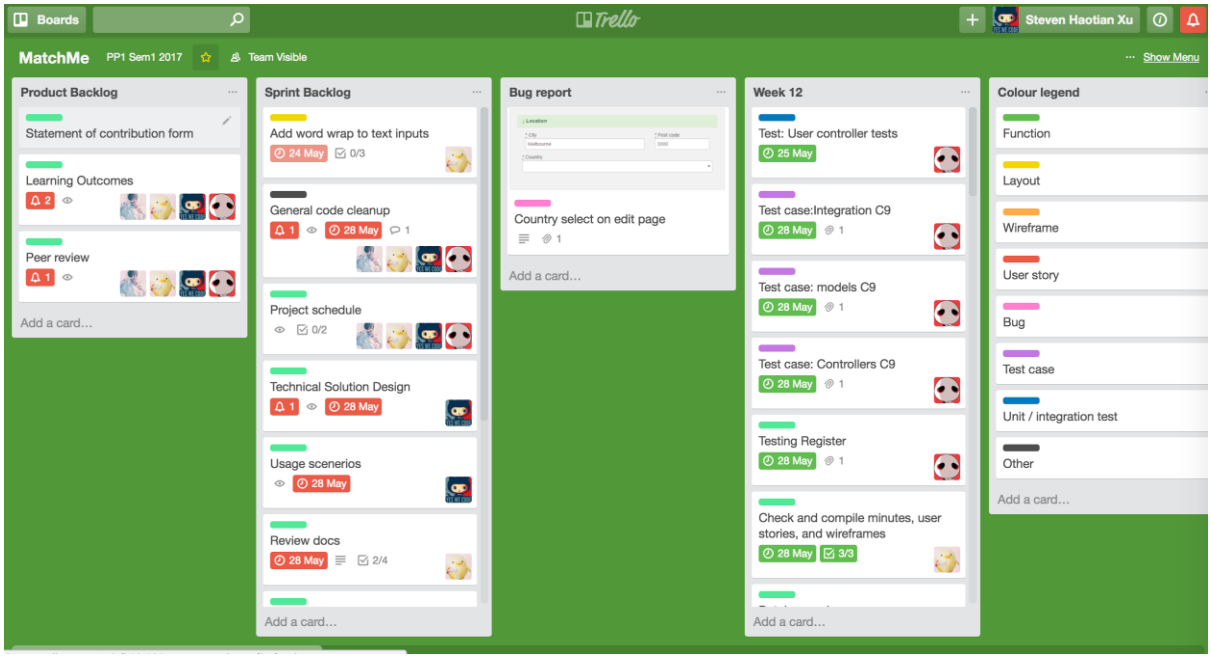


8.2.1 Known Issues

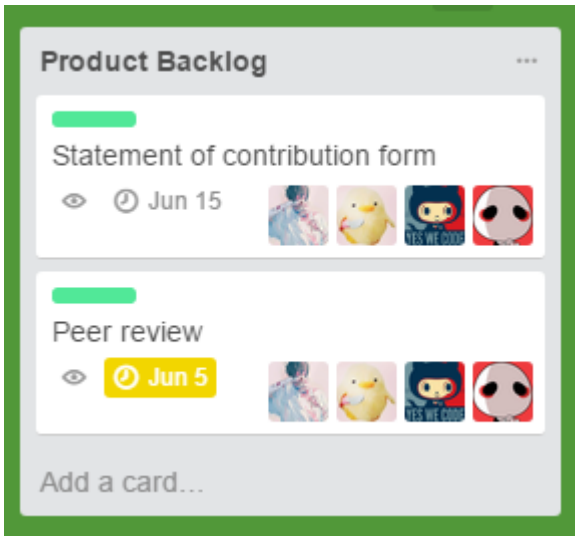
- The first time a user edits their profile, the country select field is blank even though they have previously selected a country while registering.
- In the mobile version, the account menu is not accessible.
- If a user registers an account but does not complete their profile, the admin cannot search 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



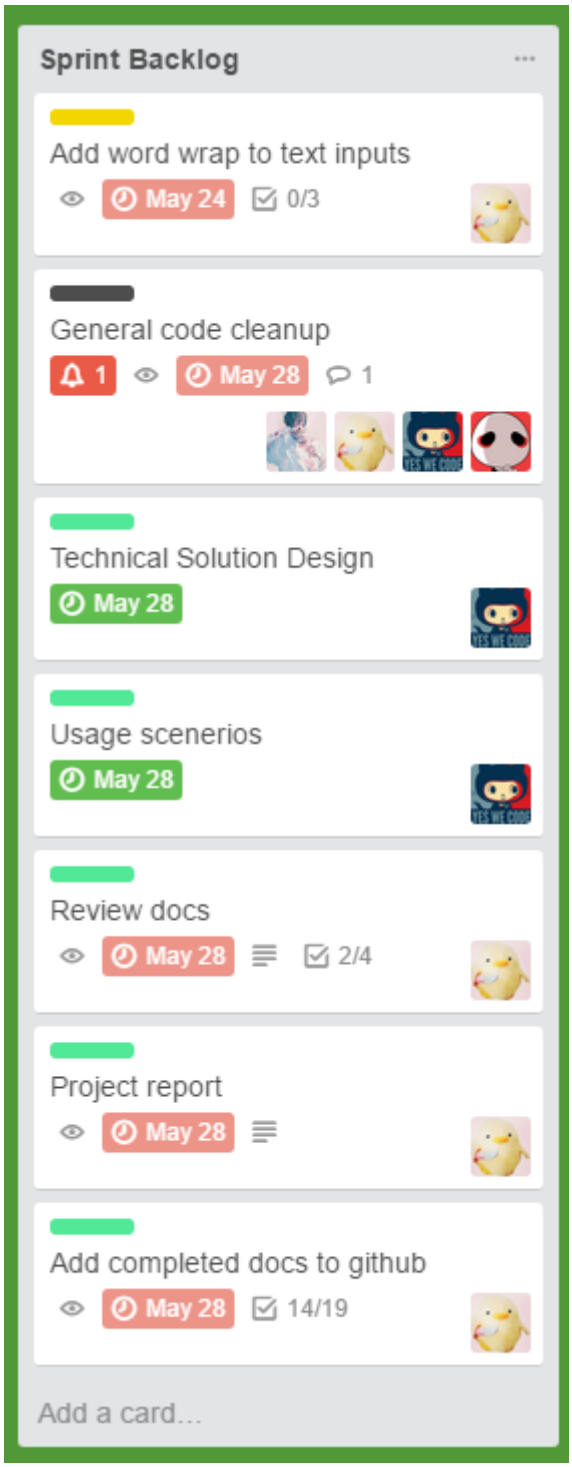
We used Trello to manage our project. Each task was given it's own card and placed in the relevant column. Tasks were assigned to group members and given due dates. Cards were also used for other miscellaneous things such as resources and the colour legend.



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.

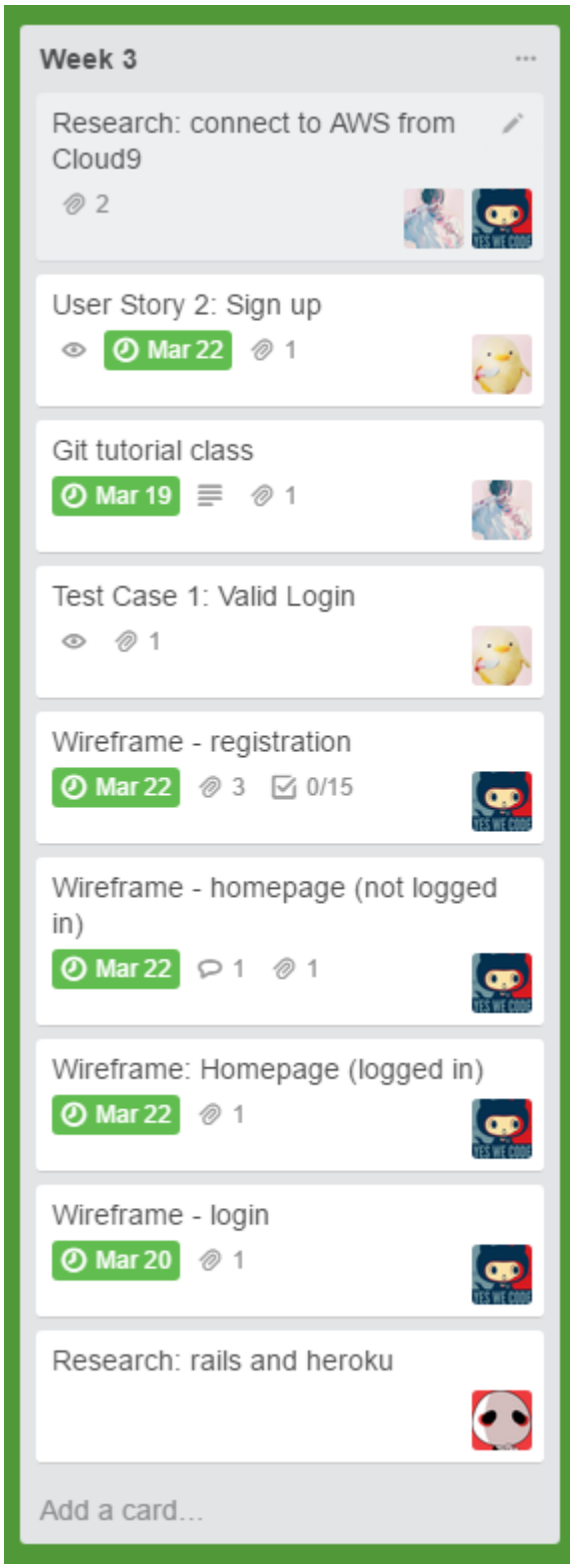
9.1.2 Sprint Backlog



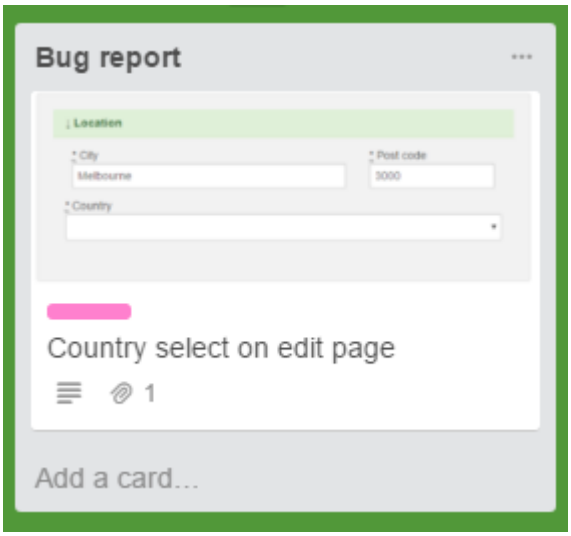
When we were ready to commence a task, we moved the card from the Product Backlog to the Sprint Backlog. All the items in the Sprint Backlog represented items to be completed until the commencement of the next Sprint. Each Sprint went for one week. When an item was added to the Sprint Backlog, it was 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.

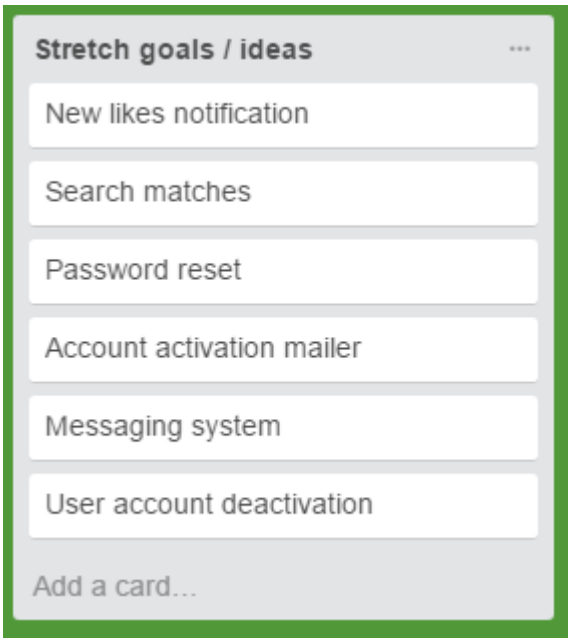


9.1.4 Bug report



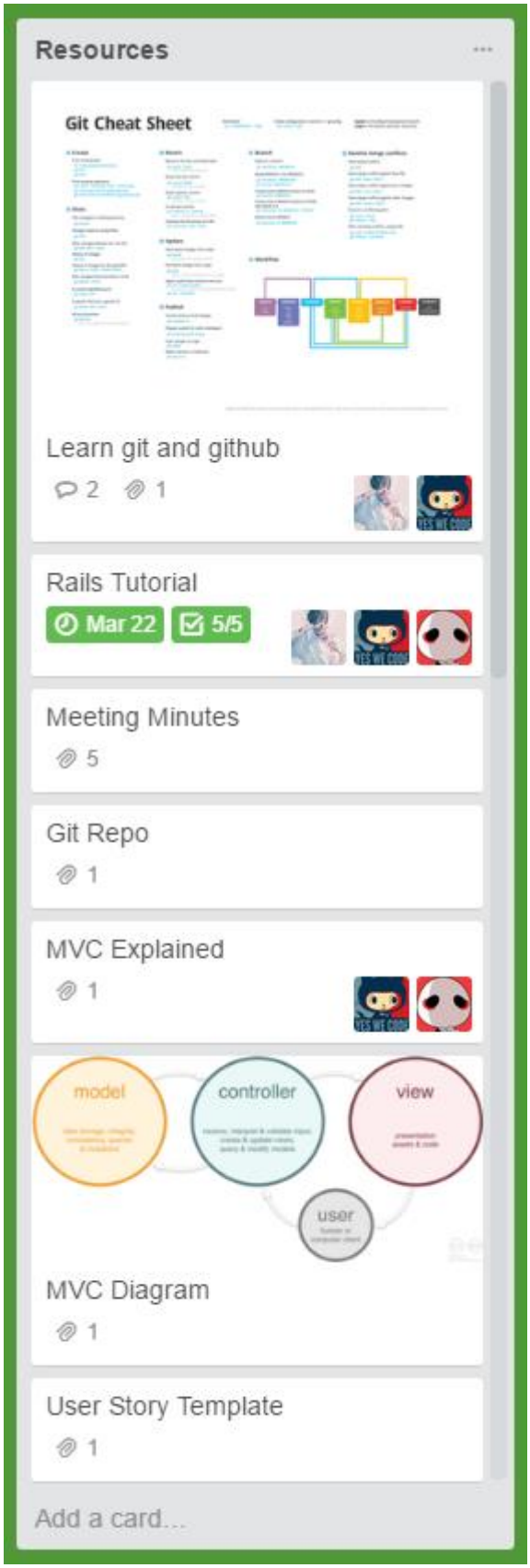
As the name suggests, this column was used to report bugs.

9.1.5 Stretch Goals / Ideas



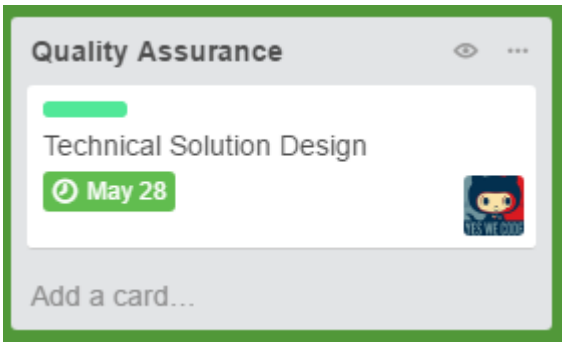
All out of scope items were placed here. These would be undertaken if all items in Product Backlog were completed ahead of schedule.

9.1.6 Resources



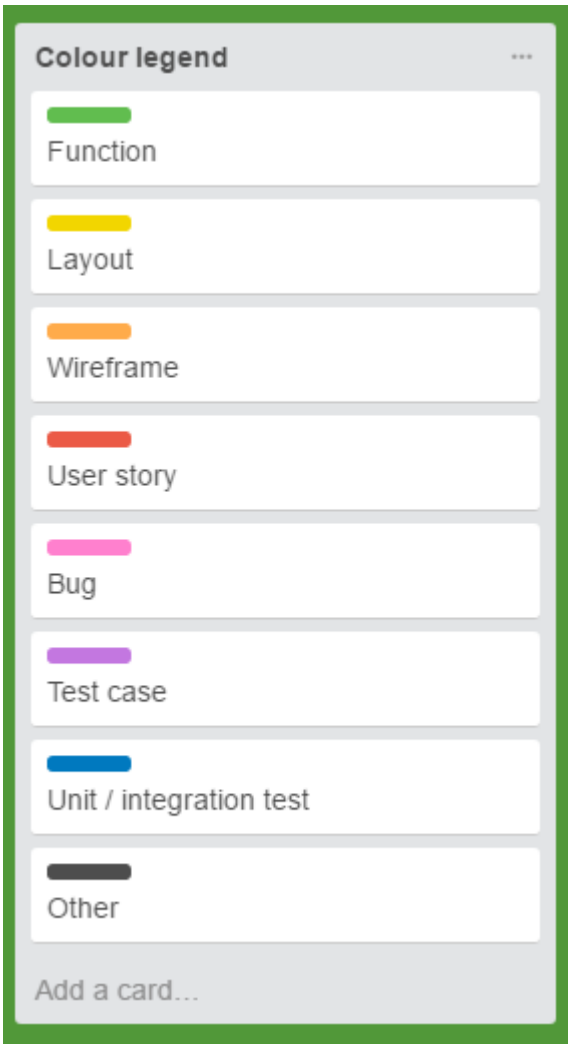
Any useful links or documents were placed here.

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