CE903 Group 1

Final Report

The Faculty Cooperative – An Interactive Website for “Crowd Sourced” Academic Entrepreneurial Activity

**Team Members:**

ALSHUAIBI, ENAAM ABDULMONEM O

CHEN, XIAOFENG

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

**Supervisor:**

GARDNER, MICHAEL

**Date:** 19 March 2014

Table of Contents

1 Introduction 3

1.1 Background 3

1.2 Server Side Technology 3

1.3 PHP Frameworks 4

2 Software Tools 5

2.1 Laravel 5

2.1.1 Laravel Background 5

2.1.2 Laravel and Composer 5

2.1.3 Laravel Migrations and Seeds 6

2.1.4 Laravel Eloquent ORM 6

2.1.5 Laravel Validation 6

2.1.6 Laravel Blade Templating System 8

2.1.7 Laravel Package Management - Using Imagine 9

2.2 Bootstrap 9

2.3 LAMP/MAMP 9

2.4 GitHub 10

2.5 Trello 10

3 Use Cases 10

3.1 Use Case Description Implementations 10

3.2 Class Diagram 12

3.3 Database Diagram 13

3.4 Downloaded Code 17

4 Testing 17

5 Conclusions 17

5.1 Successes 17

5.1.1 Teamwork 17

5.1.2 Project Management 18

5.1.3 Tools 18

5.2 Shortcomings 18

5.2.1 Teamwork 18

5.2.2 Project Management 18

5.2.3 Tools 18

5.3 Possible Extensions 19

6 Appendices 19

6.1 User Documentation 19

6.2  Minutes of All Group Meetings 20

6.3 C. Full Code Listing 45

7 Glossary 45

8 References 45

# Introduction

## Background

University based entrepreneurs often lack the skills to effectively bring a new idea to market. The *Faculty Cooperative* website aims to provide a means for entrepreneurs to find the people with the skills they need and build a team for their ventures.

The *Faculty Cooperative* website has been produced for the CE903 group project as part of the MSc Advanced Web Engineering program [1].

Requirements for this project have been elicitated in the System Requirements Specification document [2]. In section 3 Use Cases each of the use cases is presented in turn with some details about implementation.

This document also describes the tools used in section 2 Software Tools. Descriptions of written code are provided in section 4 Overview of Code Listings. Test results are presented and discussed in section 5 Testing. Finally successes and shortcomings are reflected upon in section 6 Conclusions.

## Server Side Technology

There are a number of web development server side languages in common usage. The team considered these in relation to the teams skills, available jobs and popularity. Figures for available jobs (Feb 2014, UK) were obtained from [www.itjobswatch.co.uk](http://www.itjobswatch.co.uk). Figures for popularity were obtained by searching Google for the relevant file extension using for example: filetype:php. Whilst this method is rather crude it does support the assertion that PHP is the most popular server language on the Internet.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Team members have skills | Matching job adverts | Google File Results/Millions |
| Ruby | No | 2310 | 14 |
| ASP.NET(MVC) | Little | 4300 |  |
| ASP.NET(Web Forms) | Some | 1020 | 593 |
| PHP | Yes | 6318 | 2620 |
| JSP | Some | 1286 | 88 |
| Python | Little | 2940 | 1.2 |

Table 1 - Popular Server Side Technologies Compared

The team members are currently all following the Advanced Web Engineering MSc program, which includes compulsory modules covering some ASP.NET, and JSP [3]. It was felt that adding some additional PHP experience to this would be beneficial to our education and future career prospects.

## PHP Frameworks

Once the decision has been made to use PHP for the server side of our system, we faced the choice of a number of PHP frameworks. Whilst it was not entirely necessary to use a framework, we felt that it would speed up development. Frameworks can take much of the repetition out of such routine web site tasks such as authentication and validation.

We considered a number of frameworks in terms of licencing, ease of use, testing and documentation. One team member (Bahit) had some experience with CodeIgnitor and another (Sarah) experimented with CakePHP in the initial stages of the project. Based on experiences and research from various web forums we drew up the comparison in Table 2 - Framework Comparison [4][5]

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Laravel | CodeIgniter | Zend | CakePHP | Symfony |
| **Licence** | MIT | MIT | BSD | MIT | MIT |
| **Documentation** | Good | Good | Good | OK | OK |
| **Ease of Use** | OK | OK | Harder | Easy | OK |
| **Testing** | PHPUnit | No Library | PHPUnit | PHPUnit | PHPUnit |
| **Problems** | Installation quite involved | May not continue to be supported | Not Free | Not flexible enough to fulfil requirement |  |
| **Advantages** | Very popular and well rated | Good community support |  | Easy to use and install |  |

Table 2 - Framework Comparison

Table 3 shows recent trends in PHP framework popularity. The raw data was obtained from Google Trends Laravel has shown a steady increase in popularity since its release in June 2011 [6].

Table 3 - Trends in PHP Framework Popularity

Based on the previous findings the most appropriate choice for a framework seemed to be either CodeIgniter or Laravel. As there was some doubt that CodeIgniter would continue to be developed we chose to use Laravel.

# Software Tools

The following sections describes the software tools utilised

## Laravel

### Laravel Background

Laravel is a free open source PHP framework based on the MVC paradigm [7]

When PHP version 5.3 was released in 2009, it introduced several new features that improved the functionality of Object-Orientated applications. Laravel’s creator, Taylor Otwell felt that the PHP frameworks available at the time were not leveraging the new features in PHP. So he created Laravel “*simply to solve the growing pains of using CodeIgniter PHP framework*” [8].

The first version of Laravel was released in June 2011 and it is now up to version 4.1

### Laravel and Composer

Laravel uses Composer for dependency management and installation of packages [9]. It works on a per project level to keep track of which library the project requires, and which libraries those libraries in turn require.

### Laravel Migrations and Seeds

Laravel includes a versioning system for databases, to allow for any structural changes to the database to be shared and tracked. This is accomplished by creating and running a class that extends Laravel Migration. Any changes go into a new class and these are executed in order, or can be rolled back. For example if a developer changed a table name this would go into a new class saved as a file with an auto generated name reflecting creation time. If another developer then changes the table name again, another class is created, and added to the list of classes to be run to update the database. The problem with this method is that the structure of the database will eventually be contained within many different class files, obscuring the structure of the database and making maintenance harder.

As the project already has the version control capabilities provided by GitHub, we decided not to stack Laravel version control on top of GitHub. Instead we kept only a single file of class Migration and used this to reflect all changes to the database. If we needed to roll back any changes we could do this by reverting the single file on GitHub.

Migrations are made to the database with the following steps:

1. Empty the migrations table in the database. This will allow us to run a migration that has been run previously
2. Use Laravel’s command line interface: php artisan migrate

Laravel also has a method for seeding the database. A test set of data is included within a class that extends Seeder and run with the following command line: php artisan db:seed. This allowed the team to maintain a consistent set of test data which could be used to populate the database when required.

### Laravel Eloquent ORM

*Eloquent* is *Laravel’s* (Object Relational Model) ORM implementation [7]. Each table in the database has a corresponding model that can interact with it. *Laravel* uses certain naming conventions to facilitate the mapping. A class name becomes a singular version of the table name, in Pascal case. For example the table name skills will map to the class Skill. The following code shows the Skill class

<? php class Skill extends Eloquent { }

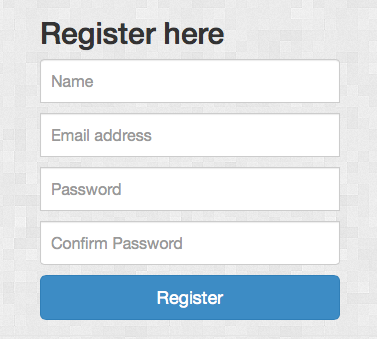
This is all that is required to map a row from the database to an object in the application.

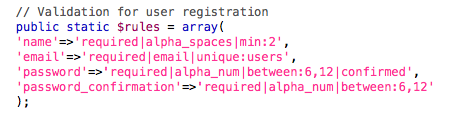
### Laravel Validation

Since forms validation is an essential part of any web applications to ensure that only valid data is saved in the database and in a certain format or length, Laravel has a validation library to make this part simple and easy to implement [10].

*FacultyCooperative* website has many forms to validate their input fields before completing the action process of each form.

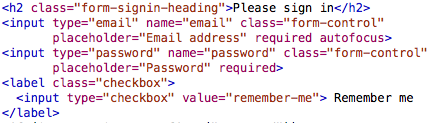
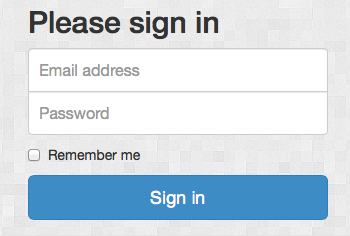
Validation of *Register from*:

* All fields are required for this form.
* Email field must be unique.

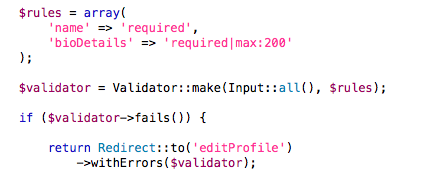


Validation of *login from*:

All fields are required in this form.

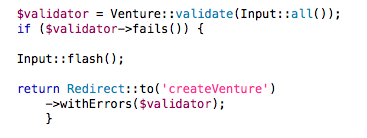


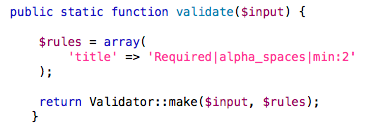
Validation of editProfile form:

name and bioDetails fields are required.

Validation of createVenture form:

Title of the venture is required.



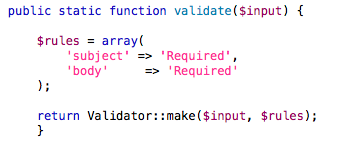
In this part, the *validate* method called from *model/venture.php*

Validation of sendMessage form:

subject and message fields are required.



In this part, the validate method called from model/Message.php



Finally, all forms have their validation part now and all of them are tested and work properly as expected.

### Laravel Blade Templating System

Blade is a templating engine that provided by Laravel [7]. It allows pages to inherit from predefined templates. It has been used on the Faculty Cooperative site to template the header, footer and navigation sections. It has various control structures that can be used to work with data that has been returned to a view. For example it can be used to loop an array of objects, printing out properties of each. In the file showProfile.blade.php it is used to loop through a list of teams that a user is involved with.

@foreach($teamInvolvement as $team)

<li class="list-group-item">

<ahref='/viewVenture/{{$team->venture\_id}}'>{{$team->title}}</a>

</li>

@endforeach

@endif

### Laravel Package Management - Using Imagine

Imagine is a PHP image manipulation library. We added this library to Laravel as a package via Composer*.* Laravel uses facades as a static interface to classes within the inversion of control (IoC) container. The IoC container is used to manage class dependencies.

We used the Imagine library to resize uploaded images ‘on-the-fly’. As users can upload their own profile images, and these may be large files, the website needs to be able to resize these images before display thus making efficient use of bandwidth.

## Bootstrap

Bootstrap is a framework designed to facilitate the front-end development of websites and user interfaces. It is a combination of HTML, CSS and JavaScript, and it supports HTML5 and CSS3.

We have decided to use Bootstrap for two main reasons.

First of all, it saved us a lot of time in the design of the website. By using Bootstrap, we could simply implement the features that we wanted without laboriously working on the CSS code first; for instance, the progress bar on the Venture pages was easily implemented. Also, it saves time in deciding the overall layout of the website by easily replacing features.

Second, Bootstrap is very customizable. Having the features that we wanted for the website, we modified them to suit our needs. A good example for this is the carousel on the home page, or the navigation bar in the header area, both of which we customized. Among other changes, the carousel in particular was modified so that it will not appear when the website is viewed on screens with a size smaller than 767 px.

Other features that we have used from Bootstrap are the buttons on the search page, the search bars themselves, and *glyphicons* for the profile and venture pages. Glyphicons are icons and symbols that are easy to use, and that have been made available for Bootstrap free of charge. Therefore, we have found this feature to be extremely useful and straightforward, as the users already know for example that the “envelope” glyphicon suggests sending or reading a message.

## LAMP/MAMP

In order to develop Laravel to build FacultyCooperative, a development environment is required to make Laravel work and this requires setting up all the software that would be running on a web server.

Typically Apache, MySQL and PHP is used to make Laravel work and setting this up on a development workstation requires some initial setting up and it would take some time if it is done manually. When having multiple operating system such as Windows and OSX, this furthers complicates the setup as it have different configuration settings and dependency.

As a general rule, the team required to have at least PHP version 5.3 and above as a baseline since this is the minimum requirement of Laravel [7].

To make this process easier, a third party application is used called \*AMP (where the first letter designates the first letter of the OS, i.e. L for Linux, M for Mac and W for Windows), A for Apache, M for MySQL and P for PHP. This application bundles all the software to run a development environment to build website or web applications using PHP as the main language.

## GitHub

Github is a web hosting service which offers support for software development projects that are hosted on an online repository. Github has both commercial and free versions respectively. The paid version allows for privately hosting a repository, while the free version allows for open source projects to be hosted. The coding languages used to develop GitHub are Ruby on Rails and Erlang. Github has a variety of features, primary of which is online hosting of source codes for software development. Another important feature of GitHub is integrated issue tracking. This feature filters by open and closed issues, assignees, labels, and milestones. The issues are sorted by issue age, number of comments, and update time. This helps filter out bugs that may exist in the software code [11].

In our team, we used Github to share the files in our project. It was especially useful because it updated easily and we all had the latest version of the project. Another good point of Github is in our team, some members are using Windows 7 operating system and others are using Macs, it took a while to get set with it, but in the end even people used Windows or Macs, It didn’t matter. In Github, we had a clear track of who did what and what changes were made in Github history part, it is quiet good function because master can track the members’ working process, and on the other side, all the members in our group can know the process of our project very clearly. And if there were problems with the code, we could revert to a previous version.

In conclusion, Github is a very user friendly software code hosting repository that members can use to socially. Interact with other members of the group and keep track of each member contribution or work on the software that is being coded. GitHub is also very secure and provides encrypted layers of system security.

## Trello

Trello is a free web-based project management application. It is a perfect tool for team management or individual work [12]. Trello is composed of three elements: *board*, *list*, and *card*. Board is your project, for example, our project is called CE903-1, so CE903-1 is our board. Lots of lists make up a board. The function of list is to show all parts of our project, in our project, there are eight lists: to do, doing, done, meeting, final report, specification documents, discussion, Github master commit. It is easy to understand, to do, doing, and done list. They show something we need to do, something we are doing now and something we have done. Usually, we have a meeting to discuss what to do, and add the results in the to do list. If we have done some work in doing list we can move the card to the done list. For meeting list, we had a meeting every week, to which we can submit minutes of the meeting to meeting list. It is very useful, because we can check every decision made in meeting. Final report and the other three lists also allow us to submit documents for each list. The card is the base element of Trello, we add cards to list, look at the to do list, for example, this week, we need to design UI for homepage, master will add “UI design for homepage”, in the card, we can add more details about the topic, such as text or documents, also, we can add members in this card who will do the work in this card.

Trello is a flexible project management application mainly in that it can create many list for the users’ requirements, and each list corresponding to a process. Another feature of Trello is easy to use. For every list and card, there is only one basic manipulation, dragging. If you want to adjust the process of teamwork, the only way you need to do is drag the list and card. For example, you want to move a card from doing list to to do list, just drag it. Compare to other project management application, the reason we choose Trello as our Project management is not only its flexible and easy to use, the other reasons are: (1)Trello design a very useful function for team work, we can set a member in a card, which means the member need to complete the work. So the division if labour cut, each one being charged with specific responsibilities. (2)Trello will record any members do any changes in the Trello. This is useful, especially for the leader in the team, he can check team work or individual work and make some changes for all parts of work.

In conclusion, Trello is a very good project management for our team, flexible, easy to use, multifunction. Trello makes our project teamwork easier.

# Use Cases

This section summarises how each of the use cases set out in the SRS document have been achieved [2].

## Use Case Description Implementations

**New User Registration**

This allows a new user to register with their email and password. To keep the registration process as short as possible minimal details are required at this stage. The email will be used to login. If the email already exists in the system an error will be sent to the user. The password must be between six and twelve characters. The password must be entered twice and these entries must match.

**User login**

A registered user can login with their email and password. The email and password combination must match the database.

**Search Public profile/ Search Ventures**

The search page allows any user to search public profiles by name and skills offered and to search for ventures by their titles or the skills offered.

**View Public profile**

As a result of a search or by seeing a users name in the team of a venture any user can view another users public profile. A URL can also be provided or shared to allow direct access to a public profile.

**View ventures details**

As a result of a search any user can view a venture page. A URL can also be provided or shared to allow direct access to a venture page.

**Edit private profile**

A registered user can edit their profile

**Contact another user**

Clicking the contact link in a public profile allows the user to send a message

**Create a new venture**

A registered user can create a new venture by entering a title and then proceeding to the edit venture page to full in the remaining details. In the original use case only users from the university ecosystem could create new ventures. However in the actual implementation any user can create a new venture. This could easily be altered in the future if required.

**Build team**

A logged in user who is a team leader can edit the team for that venture. They can add new members, delete members and change the status of members. Member status can be: Leader, Mentor or Member. One venture can have more than one team leader and each team leader has access to edit the venture and the team.

**Enter private team area**

Not implemented. We did not have time o complete this but it could be added to the existing system fairly easily.

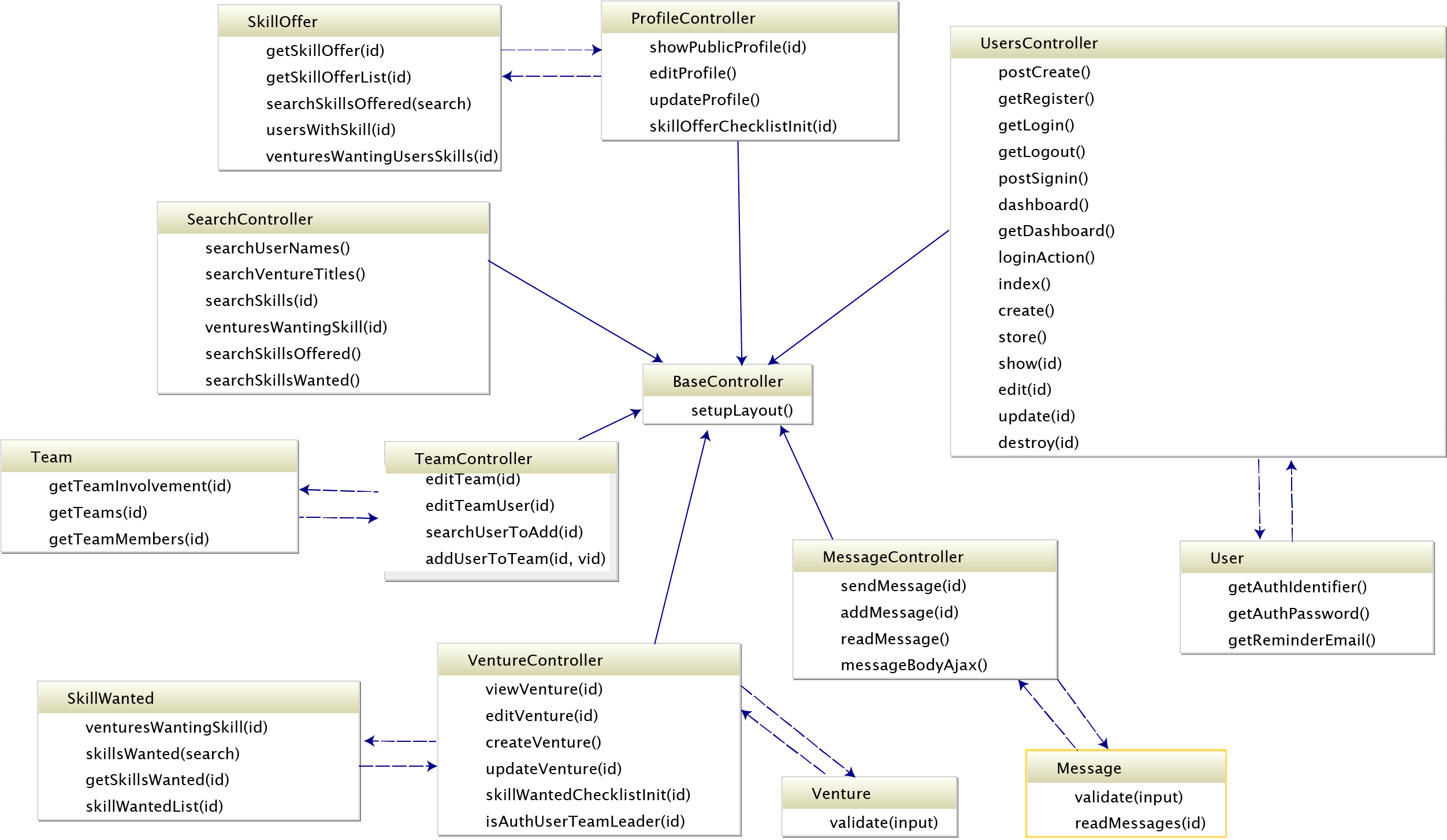
**Site administrator**

Not implemented. This would take the form of a control panel for a non-technical site administrator. Again we run too short of time to implement this but it could be done fairly easily.

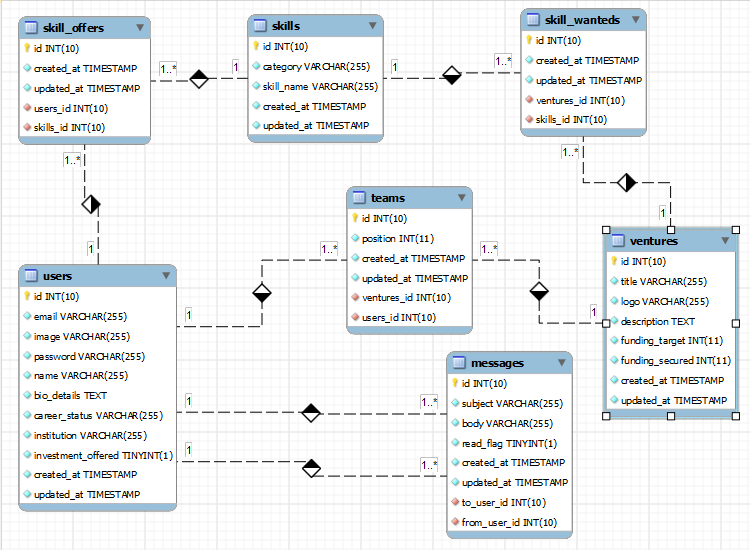
# Implementation

The following sections describe implementation details for the *Faculty Cooperative* website

## Class Diagram

****

## Database Diagram



## Database Implementation

We used a MySQL database. Connections to the database were configured in app/config/database.php. It would be straightforward to reconfigure the application to use a different database at any stage. Laravel currently has support for MySQL, Postgres, SQLite and SQL Server.

To use Laravel ORM, Eloquent, certain conventions apply. Laravel assumes every table to have an auto-incrementing integer primary key named *id*. It expects every table to have TIMESTAMP columns named *created-at* and *updated-at*. It expects foreign keys to be named after the related table and appended with *–id*. Thus the foreign key in the *skill\_offers* table from the *users* table is named *users\_id.*

Modelling skills in the database was one of the hardest problems faced by the team. Different approaches were discussed. We could have allowed uses to enter skills in free form text. Whilst this may have been more flexible, there seemed to be several problems. Users may describe their skills using different terms, which would make skill matching difficult. The algorithm needed to match the free form text from a venture to the free form text from a user to match skills would be rather involved. A natural language solution may be viable, but probably not achievable within the time frame of this project. Instead we modelled the skills in a fixed hierarchy. The *skills* table modelled this hierarchy by recording a category for each skill. This could be expanded at any time, and the website pages would automatically reflect these changes. Skills wanted by users could now be modelled in the *skills\_offered* table by recording the *user\_id* and *skill\_id*. Now a simple algorithm could be used to match skills wanted by ventures, *skills\_wanted,* with the user that have that skill. This skills matching process is core to our application.

## The MVC (model-view-controller) Pattern

The Laravel architecture supports the MVC architectural pattern. The code is arranges into model, view and controller folders [10].

The **model** is based on a real world item and typically maps to a table in a database. In Laravel validation can be placed inside the model class along with raw SQL database queries.

The **view** generates the user interface and is based on HTML. We have made use of the Laravel Blade templating system in the views, which can loop through datasets and display object properties using an abbreviated syntax. PHP can also be used in the views to navigate the data.

The **controller** receives requests and input from the view and coordinating the response. The controller may retrieve data from the appropriate model and return this data to the view.

Figure 1 - MVC for the Show Venture View

Figure 1 shows how Laravel uses the MVC pattern in our application. The view requests data for a venture from the controller, the controller obtains this from the model and returns this to the view. The view and the model need have no knowledge of each other in this case.

## Code Listings

The following code was written by members of the team:

#### app/routes.php

This file contains all the routing instructions for the application. Most of the routes direct requests at a named controller to separate application logic from routes. For example the URL: /publicProfile/1 follows this route:

Route::get('publicProfile/{id}',

'ProfileController@showPublicProfile');

This routes a request with the parameter *id=1* to the *showPublicProfile* function of the *ProfileController*.

Routes can go straight to views instead of controllers, for example the route to the search page view:

Route::get('search', function()

{

return View::make('search');

});

#### app/models/

This folder contains classes that correspond to tables in the database. These classes extend the Laravel Eloquent class and form the foundations for Laravel’s object relational mapping (ORM). By convention each class is named as a singular of the plural table name, with a capitalised first letter. Thus the class *Message* maps to the table *messages*. Column names in the database correspond to properties in the class, so the *body* column of a row in the corresponding *messages* table can be set by:

//create a new object of Message Class

$message = new Message;

//body property now available and maps to database column

$message->body = ‘This is some text in the message body’

Validation can also be added to models.

#### views/createVenture.blade.php

This view allows a logged in user to create a new venture and give it a title. This creates a new *ventures* record in the database and assigns the logged in user as a team leader. The user is then presented with a link that will take them to the *editVenture* view with the *id* corresponding to the newly created venture.

#### views/editVenture.blade.php

This view edits the venture in the database. The logged in user must be a team leader to gain access.

If details already exist in the database these will appear in the editable form areas for the user to adjust as they see fit.

In the skills the venture seeks area the checkboxes reflect the state of the *skill\_wanteds* table in the database. Any changes here are reflected back to the database.

A loop has been used in this view to avoid repeating the skill category for every skill in the list, but just print out the skill category as a header. The *Laravel Blade* template does not allow for variable assignment, as it is not recommended to assign variables in views. However, as this is more a view issue than a control issue, we felt using a little embedded PHP here to be an acceptable solution.

<?php $category=$skill["category"]?>

This just keeps track of the current category. Interestingly, the Blade mark up below does work for variable assignment, but as it also performs a PHP echo back to the page it could not be used in the case

{{$category=$skill["category"]}}

Due to the nature of HTML checkboxes, to maintain state of the checkboxes an array of arrays has been returned to the view from the controller instead of an array of objects.

#### views/showProfile.blade.php

This page displays the fields from the *users* table that are available for any other user to view. Offered skills are obtained fro the *skill\_offers* table.

It also provides a link to send a message to this user.

#### views/editProfile.blade.php

This page allows a logged in user to edit the information that is displayed in the public profile with *showProfile.blade.php.* The user can upload a new photo and edit their biography text as well as change the list of skills they offer.

#### views/readMessage.blade.php

This allows a user to view their internal messages. This page uses jQuery to allow the message body to slide down when the message subject button is clicked. We felt that a simple sliding animation here helped users to navigate and understand the page. An AJAX request sent via jQuery marks messages that have been read with a request via the *messageController* . This updates the *read\_flag* in the *messages* table. Clicking on the message subject also reveals a link to respond to the sender of the message.

#### views/sendMessage.blade.php

This allows for a message to be sent to a user using the internal messaging system. The message is sent from the logged in user

#### views/search.blade.php

This page allows for various searches to be performed. This has been split into several different searches, as the logic required for each search is very different. For example it would not be meaningful to search the entire site/database for ‘php’. This phrase only occurs once in the database in the *skills* table. It is however, more meaningful to search for *all users who offer php as a skill* or *all ventures seeking php skills*.

The searches are all based on SQL LIKE statements so empty search fields list all possible results. We only have a small dataset so this does not present a problem but in a real system some sort of pagination or limit on the number of results returned would be required.

The *search for user by name* selects all user names that contain the search string. The results are presented as links to that users profile page. The *search venture title* works in a similar fashion presenting links to ventures as results.

The *search for skills offered* does not include all skills in the system as possible results, but only searches the subset of skills actually being offered by users. So for example, as no users are offering the skill of Garden Design, a search for ‘garden’ yields no results. A search for ‘java’ will display all offered skills containing the string ‘java’ as links. Clicking on one of these links will display all users offering Java as a skill. Clicking on one of these users will take you to their profile page, and allow you to contact them with the messaging system,

The *search for skills wanted by ventures* works in a similar fashion to the previous search. This time it searches for skills that ventures require, and links to the relevant venture page.

#### views/layout/footer.blade.php

Using the Laravel Blade templating system allow a footer template to be defined once. The simple footer just holds a copyright statement but could be extended to include useful links.

#### views/layout/header.blade.php

The header also utilises the Blade templating system. It contains the main navigation for the site. The request path is used to maintain a highlighted state for the current page. The Laravel authorisation system is used to change the menu according to the logged in state of the user. A logged in user has their icon displayed in the menu bar.

#### views/layout/marketing.blade.php

This view provides the content for the marketing section of the home page.

#### views/layout/carousel.blade.php

This view provides the content for the carousel section of the home page.

#### views/users/dashboard.blade.php

This view allows registered users to access their profile, messages and ventures. It is a landing page having logged into the system.

#### views/home.blade.php

This view selects the template content to be displayed on the home page.

#### controllers/MessageController.php

This controller provides the database interactions for the *sendMessage* and *readMessage* views.

#### controllers/ProfileController.php

This controller provides the database interactions for the *showProfile* and *editProfile* views.

#### controllers/SearchController.php

This controller provides the database interactions for the *search* view.

#### controllers/TeamController.php

This controller provides the database interactions for the *editTeam* view.

#### controllers/VenureController.php

This controller provides the database interactions for the *viewVenture*, *editVenture* and *createVenture* views.

#### controllers/UsersController.php

Bahit?

#### css/main.css

This is where any changes to the bootstrap.css settings are made.

## Downloaded Code

Code not listed above has been downloaded as part of the Laravel installation or an installed library. Some configuration files have minor modifications.

## Security

The Laravel Hash class has been used to hash the passwords. The logged in user can be accessed via the Auth class. Laravel has some built in security features to protect the session id of the user, such as automatically generating session id.

Laravel has various more advanced security features such as the use of tokens to protect against cross-site request forgeries. These have not been implemented at this stage.

### XSS

We tested the following inputs in the name field of our ‘Edit Your Profile’ page.

<script>alert("XSS")</script>

"><b>Bold</b>

'><u>Underlined</u>

<img src="javascript:alert('hello everybody')"></img>

Unfortunately these tests did reveal some vulnerabilities in our code. The input was not being correctly sanitised. This was solved by installing the following package:

<https://github.com/FrozenNode/XssInput>

FrozenNode successfully provided a way to filter all the input.

### SQL Injection

We attempted the following SQL injection attacks on the search for user by name field of the search page

something or 1=1’

This returned no results so the SQL injection failed in this attempt.

# Testing

- strategy - unit level (black box / white box) - integration level - acceptance

- examples of important test results

(A complete set of tests may not be necessary, as long as it can be shown that your strategy would result in a fully tested system.)

# Conclusions

## Successes

### Teamwork

The teamwork aspect of the project went very well. Meetings and the initial lectures were well attended. Communication was generally good using a combination of Trello, WhatsApp and email. Everyone on the team was willing to contribute and on the whole commitments were met on time.

The team worked well together brainstorming ideas and sharing skills and experiences. Differences of opinion were all sorted out very amicably in the group meetings with a little discussion.

### Project Management

There was no overall team leader. Chairperson and secretarial duties were shared on a rotating basis. This worked well for our team.

### Tools

Sharing code on GitHub generally worked well. Trello was a very valuable organisational tool.

Laravel worked well as a framework once initial configuration difficulties were overcome. The Eloquent ORM worked very well, reducing the effort required to interact with the database. Laravel’s migrations worked well for sharing the database structure. The login and authorisation system was easy to use on various pages once implemented. The Laravel documentation was very good and it was not too difficult to learn.

## Shortcomings

### Teamwork

No issues. I think everyone enjoyed being on the team.

### Project Management

There were some issues with time management, we did not keep up with the Gantt Chart [2]. It was difficult to complete everything for this project within the time frame. The project scope was quite broad so we think our achievements are still good.

We initially intended to have each team member construct a few pages of the site each. However, as time progressed it seemed more efficient to break down tasks so people took responsibilities for different jobs such as testing, styling, validation etc.

### Tools

Laravel was quite difficult to install and required some changes to the server environment. It was difficult to provide a consistent environment across all the development machines, which led to some problems with code working on some machines but not others. We managed to mitigate most of these problems by helping each other at group meetings and sharing settings and information on Trello.

## Possible Extensions

Extensions could include implementing a team area and administrator control panel. This corresponds to the two user cases we did not complete ie. *Site administrator* and *Enter Private Team Area*. It would not be difficult to implement these and they could be added to the current system in a modular fashion as new views and controllers.

The integration with social websites could be improved.

The search page could be improved by implementing a search on any text in a venture or profile instead of just user name and profile title.















































# Appendices

## User Documentation

The documentation for the site is in the form of a FAQ page. We intend that the FAQ page will help the users to navigate and understand the site.

The contents of the FAQs page are reproduced here

#### Who can view my profile?

Any visitor to the site can view your public profile page

#### How do I create my profile?

You must be registered to create a profile. When you register an empty public profile will be created for you. You can edit this by clicking on 'update your profile' from the dashboard page

#### How do I register?

You can register by clicking on the [register link](http://fc/register). Your email will be used to login. You must pick a password

#### How do I create a new venture?

You must be registered. From your dashboard click the link to create a new venture. You just need to name your venture to begin. You can then add details and build your team

#### How do I add members to my team

You must be logged in. From your dashboard or search page click to view venture details. If you are the team leader there will be an extra link at the bottom of the page allowing you to go to the team management page. From here you can edit your team

#### How do I add a mentor to my team

You must be logged in. From your dashboard or search page click to view venture details. If you are the team leader there will be an extra link at the bottom of the page allowing you to go to the team management page. From here you can edit your team. To add a mentor first add the user as a team member, then promote them to mentor

#### How can I send a user a message

You must be logged in. Find a link to the user from the search page or venture page. Click the link to see the users public profile. There is a link to send that user a message

#### How can I read my messages

You must be logged in. Go to the dashboard and follow the links to read your messages. Click on a message header to expand the message body

## Minutes of All Group Meetings

**CE903-1 Minutes of Group Meeting 1**

Date & time of meeting: 14 Jan 2014

Place of meeting: CSEE Lab 1

Group members present:

ALSHUAIBI, ENAAM ABDULMONEM O

CHEN, XIAOFENG

HAMID, BAHIT BIN

HERASCU, IRINA (Remote)

THOMAS, SARAH ANNE

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting:

HAMID, BAHIT BIN

Name of secretary for this meeting:

THOMAS, SARAH ANNE

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

NOT APPLICABLE

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

NOT APPLICABLE

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

1. General Introductions
2. Identified team skills and summarized to [docs.google.com](https://docs.google.com/spreadsheet/ccc?key=0AguZnvHEQMlTdEFtbmRSOGZkWkZSRnNYOENsRlJsTnc&usp=sharing#gid=0)
3. BAHIT set up WhatsApp group CE903-1 for communication
4. Supervisor, Dr Gardner, contacted to arrange meeting
5. PHP identified as a suitable development language

Issues/problems to be reported to project supervisor:

**CE903-1 Minutes of Group Meeting 2**

Date & time of meeting: 20 Jan 2014 15:00

Place of meeting: Orangery

Group members present:

ALSHUAIBI, ENAAM ABDULMONEM O

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

CHEN, XIAOFENG – Headache

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting:

THOMAS, SARAH ANNE

Name of secretary for this meeting:

ALSHUAIBI, ENAAM

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

Agreed

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

1. Decisions on software platforms to be deferred until after project specifications have been drafted. However a card has been added to Trello for group to add findings and comments Research on Google Trends shows CodeIgniter remains popular but may not meet CM requirements.
2. Trello to be used as a projet tool
3. All members displayed a good understanding of task requirements and have read relevant documents

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

1. The aim/goal of the website.
2. Understanding the website structure.
3. Brainstorming of the website
4. Create a use case draft diagram for the website in ArgoUML
5. BAHIT set up a mind map account for the group.
6. BAHIT set up a Trello account for the group.

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

ALL TO BE COMPLETED BEFORE NEXT MEETING

1. IRINA and XIAOFENG to prepare suggestions to functional/non-functional requirements, input/ output of the website. Possibly add suggestions to Trello. ALL TEAM to consider /comment/ add to these
2. BAHIT- to prepare Gantt chart using MS project.
3. SARAH- to prepare Use Case Diagram using ArgoUML
4. ENAAM to write up minutes and prepare agenda for next meeting

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Thursday 23 Jan, 15:00, Orangery

Chair: ENAAM

Secretary: IRINA

**CE903-1 Minutes of Group Meeting 3**

Date & time of meeting: 23 Jan 2014 15:00

Place of meeting: Orangery

Group members present:

CHEN, XIAOFENG

ALSHUAIBI, ENAAM ABDULMONEM O

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting:

ALSHUAIBI, ENAAM

Name of secretary for this meeting:

HERASCU, IRINA

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

Agreed

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

1. Team members have started using Trello and now share documents and ideas through this project tool.
2. SARAH completed the Use Case Diagram
3. BAHIT completed Gantt chart
4. IRINA and XIAOFENG prepared suggestions on the input/ output of the website

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

1. Brainstorming of the website
2. All members discussed what are the functional/non-functional requirements and the input/ output of the website
3. Members have agreed upon using CodeIgniter as PHP Framework.
4. Team members chose Git as Source Code Management System.

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

ALL TO BE COMPLETED BEFORE NEXT MEETING

1. ENAAM, XIAOFENG, IRINA and SARAH to complete tutorials on using Git and to install it
2. IRINA to prepare design for user registration forms
3. ALL MEMBERS to come up with suggestions on making the Dataflow Model

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Monday 27 Jan, 15:00, Orangery

Chair: IRINA

Secretary: XIAOFENG

**CE903-1 Minutes of Group Meeting**

Date & time of meeting: 27 Jan 2014 15:00

Place of meeting: Orangery

Group members present:

CHEN, XIAOFENG

ALSHUAIBI, ENAAM ABDULMONEM O

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting:

HERASCU, IRINA

Name of secretary for this meeting:

CHEN, XIAOFENG

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

Agreed

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

1. ENAAM, XIAOFENG, IRINA and SARAH completed tutorials on using Git and installed it
2. IRINA created the design for user registration forms

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

1. All members discussed what are the functional/non-functional requirements
2. ALL MEMBERS discussed the Dataflow Model.
3. Team members chose Laravel as Framework.
4. SARAH started a draft for the System Requirements Specification Document
5. SARAH made a draft of the UML Class Diagram

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

ALL TO BE COMPLETED BEFORE NEXT MEETING

1. XIAOFENG to prepare a list of skills for each field of study

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Thursday 30 Jan, 15:00, Orangery

Chair: XIAOFENG

Secretary: BAHIT

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

ALL TO BE COMPLETED BEFORE NEXT MEETING

1. All team members to install GitHub and familiarise themselves with the basics
2. All members to consider project tools such as Trello/Asana along with anything presented in CE903 lecture 2 on Thursday
3. All members to read and understand task requirements and prepare any questions for supervisor meeting
4. Each member to look at the suitability of a framework and report back at next meeting so we can compare and make an informed choice. Look at learning curves, support communities, documentation, and security. Suggest this is shared as followed, but if anyone has time to look at more than one or all please do. Any other suggestions also welcome

ENAAM - Symphony

XIAOFENG – Zend (Only if students get this free? – please check)

BAHIT – Code Ignitor

IRINA - Laraval

SARAH – Cake PHP

1. Consider front end frameworks such as Foundation/Bootstrap <http://www.sitepoint.com/top-10-front-end-development-frameworks/>
2. Any items for agenda to SARAH

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Monday 20 Jan, 15:00, CSEE Lab 1

Chair: SARAH

Secretary: ENAAM

**CE903-1 Minutes of Group Meetings**

Date & time of meeting: 30th January 2014, 15:00

Place of meeting: Orangery

Group members present:

ALSHUAIBI, ENAAM ABDULMONEM O

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

1) CHEN, XIAOFENG – Stomach ache, Irina agreed to replace him as the Chairperson.

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting: HERASCU, IRINI

Name of secretary for this meeting: HAMID, BAHIT BIN

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

1. Security as a non-functional requirements.

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

1. Irina to redo the Login mock into Sign-in mock for appendix of the SRS Documents.
2. Enaam to do Case Description Table for SRS
3. Bahit to Specify Non-functional Requirement, Testing and testing documentation part of SRS
4. Sarah will take everyone’s contribution to the SRS document on Monday, 3 February 2014 evening to be compiled into final draft on Tuesday.

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

1. XIAOFENG to prepare a list of skills for each field of study

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: TBA (announced and will be discussed on WhatsApp mobile messaging platform)  
(Reason doing draft and final version of SRS from Monday – Wednesday)

Chair: BAHIT

Secretary: Enaam

**CE903-1 Minutes of Group Meeting**

Date & time of meeting: 13 Feb 2014

Place of meeting: Orangery

Group members present:

ALSHUAIBI, ENAAM ABDULMONEM O

CHEN, XIAOFENG

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting:

HAMID, BAHIT BIN

Name of secretary for this meeting:

THOMAS, SARAH ANNE

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

YES

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

ALL ACTIONS HAVE BEEN FULFILLED

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

1. Moving forward with **System & Software Design 12 Days**
2. **Setting up environment**
   * **Any problem? One solution is to use Vagrant**

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

ALL TO BE COMPLETED BEFORE NEXT MEETING

1. **Setup Github Repository for FacultyCooperative (1 Day)**
2. Dividing the Functional Requirements above among team members:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Item | Time Scale | Assigned To |
| **1** | **User Authentication and Registration** | **2 Days Total** | **Bahit / Irina** |
| 1.1 | New User Registration | 2 Days | Bahit |
| 1.2 | User Login | 2 Days | Bahit / Irina |
| **2** | **Public Profiles** | **2 Days Total** | **Sarah / Enaam** |
| 2.1 | Search Public Profiles | 2 Days | Enaam |
| 2.2 | View Public Profiles | 2 Days | Sarah |
| 2.3 | Edit Public Profile | 2 Days | Enaam |
| 2.4 | Edit Private Profile | 2 Days | Sarah |
| **3** | **Messaging System** | **2 Days Total** | **Feng / Sarah** |
| 3.1 | Contact Another User | 2 Days | Feng / Sarah |
| **4** | **Venture Administration** | **2 Days Total** | **Irina / Feng** |
| 4.1 | Create New Venture | 2 Days | Irina |
| 4.2 | Edit Venture Details | 2 Days | Feng |
| 4.3 | Build Team | 2 Days | Feng |
| 4.4 | Enter Private Team Area | 2 Days | Irina |
| **5** | **Site Administration** | **3 Days Total** | **Bahit / Enaam** |

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Thurs 20 Feb, 15:00, Orangery

Chair: SARAH

Secretary: ENAAM

**CE903-1 Minutes of Group Meetings**

Date & time of meeting: Thursday 20 Feb 2014, 15:00

Place of meeting: PC lab 4

Group members present:

ALSHUAIBI, ENAAM ABDULMONEM O

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

CHEN, XIAOFENG

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting: THOMAS, SARAH ANNE

Name of secretary for this meeting: ALSHUAIBI, ENAAM ABDULMONEM O

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

New matters discussed:

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

1. Learn Laravel and DB implementation.
2. Each members start to implement interface of pages that assign to them from the previous meeting by PHP

Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Tuesday, 25 Feb 2014 in Dr. Gardner’s office at 4 pm

Chair: Enaam

Secretary: Irina

**CE903-1 Minutes of Group Meetings**

Date & time of meeting: Thursday 11 March 2014, 13:00

Place of meeting: PC lab 4

Group members present:

ALSHUAIBI, ENAAM ABDULMONEM O

HAMID, BAHIT BIN

HERASCU, IRINA

THOMAS, SARAH ANNE

CHEN, XIAOFENG

Group members absent but who explained their absence before the meeting (list given reason(s) for absence against each name):

All other group members (that is, those not listed in the previous two boxes):

Name of chairperson for this meeting: ALSHUAIBI, ENAAM ABDULMONEM O

Name of secretary for this meeting: THOMAS, SARAH ANNE

Are the minutes of the previous meeting agreed as a correct record (after correction of matters of fact)?

List any corrections of matters of fact here:

Review of all actions agreed at previous meeting and how far individual responsibilities have been fulfilled:

System has been partially build. Authorisation and login complete. Database migrations complete. Reponsibilites met as far as they were loosly laid down at last meeting.

Where individual responsibilities have not yet been fulfilled within the agreed timescale, list (for each responsibility) the reasons given:

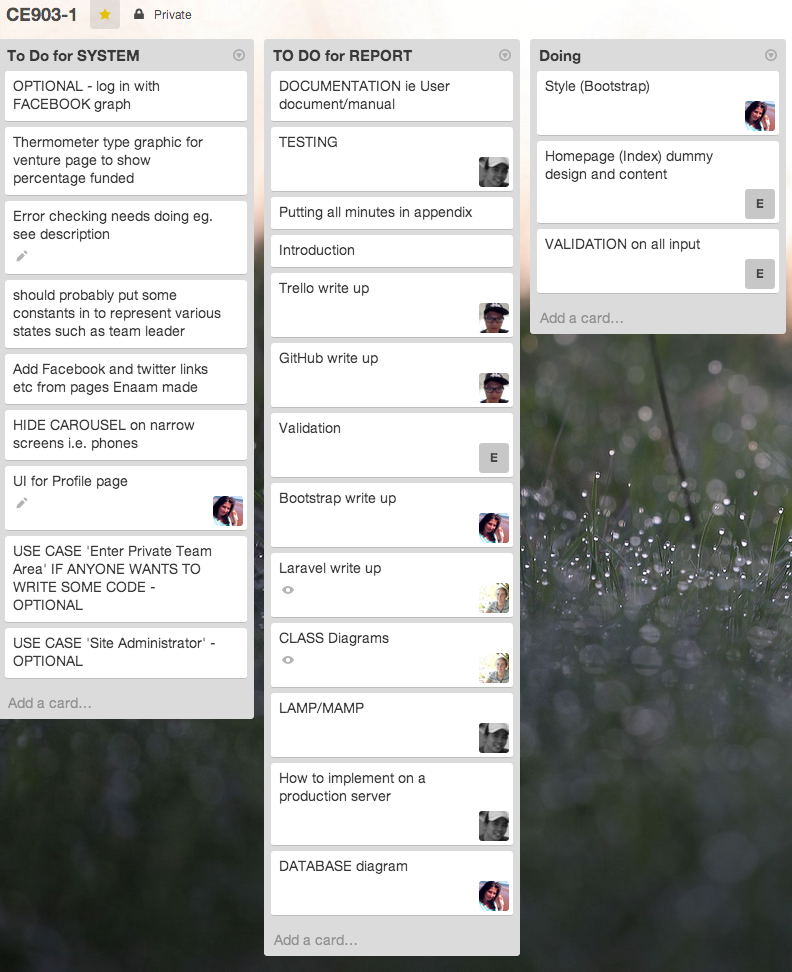
New matters discussed:

Task allocation for report, presentation and finalising system

Issues/problems to be reported to project supervisor:

List all actions requiring attention (i.e., those uncompleted actions from the previous meeting and all actions agreed at the present meeting), the agreed timescale for completion of each one and who is responsible for each one (including setting an agenda for the next meeting and writing the minutes of the present meeting). *Uncompleted actions from the previous meeting should be the first on this list*:

All jobs that need to be completed for report allocated using Trello. Please see screenshot below.



Record here the agreed chairperson, secretary, date, time and place of next formal meeting:

Next Meeting: Monday, 17 Mar 2014 in Lab 4 12 pm

Chair: Bahit

Secretary: Irina

## C. Full Code Listing

(Hard copy is unnecessary. Please save your code in well structured folders in the CD)



# Glossary

**External eco-system** includes professionals and advisors from non-academic organizations and businesses

**University eco-system** includes students, and staff based in university

**Venture** a business idea or entrepreneurial activity requiring a team leader and team members

**HTTP** (hypertext transport protocol) is a communication protocol used by the Internet to communicate from one node to another.

**HTTPS** (hypertext transport protocol secure) is a communication protocol for secure communication over a computer network. Layering HTTP on top of SSL/TLS, adding it to standard HTTP communication.

**SSL** (Secure Socket Layer) a cryptographic protocol designed to provide secure communication over the Internet.

**UX** (User Experience) mainly deals with how users interaction and how they perceive the website design and layout.

**UI** (User Interface) mainly deals with the design of the layout of the website.

# References

[1] M. Gardner, “The Faculty Cooperative – An Interactive Website for ‘Crowd Sourced’ Academic Entrepreneurial Activity,” *List CE903 MSc Gr. Proj. – 2014 MSc Adv. Web Eng.*, pp. 1–7, 2014.

[2] E. A. O. Alshuaibi, X. Chen, B. BIN Hamid, I. Herascu, and S. A. Thomas, “CE903 Group 1 -System Requirements Specification Document.”

[3] “University of Essex :: Module Directory :: Module details.” [Online]. Available: https://www.essex.ac.uk/modules/default.aspx?coursecode=CE881&level=7&period=SP. [Accessed: 07-Feb-2014].

[4] “Choose the right PHP framework | Web design | Creative Bloq.” [Online]. Available: http://www.creativebloq.com/design/choose-right-php-framework-12122774. [Accessed: 12-Mar-2014].

[5] Phpframework, “PHP Frameworks.” [Online]. Available: http://www.phpframeworks.com/. [Accessed: 12-Mar-2014].

[6] “Google Trends.” [Online]. Available: http://www.google.co.uk/trends/. [Accessed: 28-Jan-2014].

[7] “Laravel - The PHP framework for web artisans.” [Online]. Available: http://laravel.com/. [Accessed: 12-Mar-2014].

[8] M. Surguy, “History of Laravel PHP framework, Eloquence emerging - Maks Surguy’s blog on PHP and Laravel.” [Online]. Available: http://maxoffsky.com/code-blog/history-of-laravel-php-framework-eloquence-emerging. [Accessed: 12-Mar-2014].

[9] N. Adermann and J. Boggiano, “Composer.” [Online]. Available: https://getcomposer.org/. [Accessed: 12-Mar-2014].

[10] “Input Validation with Laravel - Laravel Book.” [Online]. Available: http://laravelbook.com/laravel-input-validation/. [Accessed: 13-Mar-2014].

[11] “Github.” [Online]. Available: https://github.com/features. [Accessed: 17-Mar-2014].

[12] “Trello.” .