**UCL Students: Heidi Tan, Bill Ma**

# Alternative Financing Portal

# Contents

**COMP205P Alternative Financing Portal Documentation**
**UCL Students: Heidi Tan, Bill Ma**

# 1 Introduction

This project is carried out as part of the COMP204P and COMP205P Systems Engineering modules for UCL Computer Science Year 2 students. The team had been assigned to a client, ATOS, to work with. The aim of the project is to build an alternative financing portal to match small and medium enterprises (SMEs) with suitable credit providers.

The project documentation website can be viewed by accessing this link:
http://students.cs.ucl.ac.uk/2015/group5/

The application was deployed on Heroku to allow other people to access it for the user acceptance tests. It can be viewed by accessing this link:
https://alternativefinancingportal.herokuapp.com/

# 2 Background and Context

## 2.1 Aim of Application and Context

In recent years, the UK alternative finance market has grown rapidly, leaping from £666 million in 2013 to £1.74 billion in 2014. SMEs, entrepreneurs, charities and other organisations that previously faced difficulties obtaining financing from conventional credit providers like banks are now obtaining funding easier as they look to other sources of finance such as P2P business lending, invoice trading, equity-based crowdfunding and so on. This is beneficial as the growth of the alternative finance market contributes to UK's economy.

However, even with such evident progress, more can be done to connect SMEs to potential investors faster and easier. We hope to come up with a financial portal to bridge the gap between these SMEs and people with the financial ability to fund them. We plan to focus on SMEs that have been established less than 5 years ago and have a turnover less than £10 million a year first. Our portal aims to help an SME that needs financing to work out where to get it. SMEs will be able to sign up for an account on the portal and use their webpage in the portal to pitch their product or service virtually. The portal will serve as a platform to connect with potential investors where they can exchange further details if interested to kickstart funding talks.

In addition, the portal will match SMEs to different alternative finance sources based on a range of information parameters such as the object of borrowing, the terms of borrowing, the nature of the relationship sought with the lender, the risk level of the business activity and so on. All these will help to streamline the search process and match SMEs up with investors most suitable to meet their needs and vice versa.

## 2.2 Initial Problem Statement

To help advance the growth of UK's economy, the alternative finance market should be further developed to enable SMEs to easily obtain funding. However, there are currently very few platforms for SMEs to find sources of alternative finance easily and effectively, without the representatives of these SMEs having to spend time at multiple networking events to make contacts. Hence, building an online financial portal that matches SMEs with suitable credit providers will save time as well as collate the SMEs requiring financial funding and credit providers in a single database that is easily accessible for both parties.

## 2.3 Our Client

Atos is a global digital services corporation that provides Consulting & Systems Integration services, Managed Services & BPO, Cloud operations, Big Data & Cyber-security solutions, as well as transactional services through Worldline, the European leader in the payments and transactional services industry. Using their expertise in the technology sector, the company has clients from all over the world across various business sectors such as defense, financial services, health,

manufacturing, media, utilities, public sector, retail, telecommunications and transportation.

## 2.4 Challenges to be Met

1. Learning new languages (eg. Ruby on Rails) that are necessary to build the financial portal
2. Building a search engine that will filter results according to the criteria the user chooses
3. Having a database to store the users' account details
4. Building an interface and framework for users to create their company profile on their own webpage
5. Integrating a communication platform with the portal

## 2.5 Criteria for Success

- Clear communication between the client and the team
- Sufficient technical knowledge in order to build the financial portal and all its features successfully
- Detailed documentation of the progress of the project
- A well-thought out project timeline that will be strictly followed to ensure that the project is progressing on time

# 3 Features

## 3.1 Search Tab

The search engine is the main functionality of the alternative financing portal. The search page displays all users. There is a search bar on top for general queries. It is followed by a "sorted-by" option which enables users to be sorted by their email addresses (in alphabetical order), registration date (by newest/ by oldest), name (in alphabetical order) or organisation type (SME or credit provider). To narrow down searches, the user can select to show only SMEs or credit providers. Following that, they can filter those users based on a set of questions specific to the organisation type. The text in the "Name of Organisation" column displayed on the table is a link to that user's profile page as well.

To filter the list of SMEs, the search criteria is as follows:

- type of business activity (product/ service/ others)
- nature of funding sought (loans/ equity/ options/ others)
- other support sought (financial management/ strategy expertise/ financing advice/ marketing advice/ operations advice/ HR advice/ others)
- date founded

To filter the list of credit providers, the search criteria is as follows:

- type of credit provider (individual/ company)
- nature of funding offered (loans/ equity/ options/ others)
- other support offered (financial management/ strategy expertise/ financing advice/ marketing advice/ operations advice/ HR advice/ others)

## 3.2 Profile Page Tab

The profile page will automatically display the information that the user entered when he signed up for an account. For example, for a user who is acting as a credit provider, his profile page will display the type of credit provider he is (individual or company), the name of the funder and the options he has chosen for the nature of financing offered and other forms of support offered. His contact details will be shown there as well.

The user will be able to edit all these information just by simply clicking the "Edit Profile" button found at the bottom of his profile page. Also, there is a HTML5 editor found on that page where the user can upload images, choose his own fonts and colours to enter further information about his company. This information will be displayed on his profile page as well. SMEs can use this HTML editor to virtually pitch their product or service to credit providers, while credit providers can use it to give more background information about themselves.

### 3.3 Messages Tab

As this alternative financing portal aims to match SMEs to suitable credit providers and vice versa, we felt that it should also act as a platform for these users to communicate with each other. Hence, we decided to implement an instant messaging feature so that users can contact each other easily through the portal. The messaging page displays all users and the current conversations of the signed-in user. This feature makes it convenient for users to contact each other quickly and easily.

### 3.4 Settings Tab

The settings tab allow users to change essential details such as their email, address and contact number. They would need to input their password to verify such changes.

# 4 Project Requirements

Requirements are described using 'MoSCoW' Approach.
M:Must Have
S:Should Have
C:Could Have
W:Won't Have

## 4.1 Functional Requirements

| **User Registration** | | Achieved |
|---|---|---|
| The system shall allow users to register an account using username and email address. | M | ✓ |
| The system shall allow users to input their personal information into their account (contact details, occupation, salary etc). | M | ✓ |
| The system shall send an email to users' email address to activate the account. | S | ✗ |

| **Display Information** | | Achieved |
|---|---|---|
| The system shall display user details (name, contact details etc). | M | ✓ |
| The system shall display new messages for each user. | S | ✓ |
| The system shall display the criteria that is used for searching. | M | ✓ |
| The system shall display the matching candidates after the search. | M | ✓ |

| **Edit Information** | | Achieved |
|---|---|---|
| The system shall allow users to edit their profile details. | M | ✓ |
| The system shall allow users to edit the criteria for searching. | S | ✓ |
| The system shall allow users to edit their own page. | C | ✓ |

| **User Interface** | | Achieved |
|---|---|---|
| The system shall store the user information in a database. | S | ✓ |
| The system shall allow user to search for matching candidates using criteria. | M | ✓ |
| The system shall provide a platform for users to communicate with each other directly. | C | ✓ |

## 4.2 Non-Functional Requirements

| **Performance** | | Achieved |
|---|---|---|
| The system shall direct users to different pages within 0.5 second. | M | ✓ |
| The system shall be up and running 24 hours | S | ✓ |
| The system shall process searching request within 5 seconds. | S | ✓ |
| The system shall allow maximum 100 users to use at the same time. | C | ✓ |

| **Platform Constraints** | | Achieved |
|---|---|---|
| The system shall be accessible using all major web browsers and mobile platforms. | S | ✓ |

| **Accuracy** | | Achieved |
|---|---|---|
| The system shall provide accurate result after searching. | M | ✓ |

| **Reliability** | | Achieved |
|---|---|---|
| The system shall be functional 100% of the time; failure in this context is referring to getting wrong result after searching. | S | ✓ |
| The system shall never crash. | S | ✓ |

| **Security** | | Achieved |
|---|---|---|
| The system shall only be accessed using correct login details. | M | ✓ |
| The system shall only allow user to edit own personal information. | M | ✓ |

| **Cost** | | Achieved |
|---|---|---|
| The system shall be completely free for users to use. | M | ✓ |

### 4.3 Evaluation

As seen from the tables above, we have managed to meet all of our "Must" requirements and even all of the "Could" requirements. We met all of our "Should" requirements except for one ("The system shall send an email to users' email address to activate the account."). This was due to some technical issues with the mailer where the email was not being sent out properly. We are certain that if we had more time, we would have been able to implement this feature. However, we chose to focus on the main features such as the search engine and instant messaging platform which we felt were crucial components of the application.

# 5 Development Plan

## 5.1 Research

### 5.1.1 Possible Platforms

After conducting extensive research on ways to build our financial portal, we have narrowed our choices down to two different frameworks: Ruby on Rails and Django.

1. Ruby on Rails

Ruby on Rails is one of the most popular tools used for building dynamic web applications. It utilises the Ruby programming language. We will be using this for the framework of our application. It comes with gems, which are packages that contain code, documentation and can be installed to be used on projects. Different gems can be used to install different functions for use.

We have selected a few gems and other aspects of Ruby on Rails that will be useful for us in building the financial portal.

- **Backups (Backup Gem):** This gem helps to backup the database of the application. It can save the backup on different storages such as the local server, Dropbox and so on. In addition, the backups can also be compressed or encrypted using this gem.
  http://backup.github.io/backup/v4/
- **Scheduling Backups (Whenever Gem):** This gem will schedule automatic backups of the database as frequently as we want it to do so. This is to ensure that if anything happens to the application and it crashes, we can always manually restore the database to the last backup to minimize the amount of data lost.
  http://eewang.github.io/blog/2013/03/12/how-to-schedule-tasks-using-whenever/
- **Search Engine (Filterrific Plugin):** This Rails Engine plugin can be used to filter lists when searching for something to make it easy for users to look for what they want.
  http://filterrific.clearcove.ca/
- **User Authentication:** We can use Ruby on Rails to model users and allow users to sign up for an account on our web application. In addition, they will be able to log in, log out and update their profiles as well.
  https://www.railstutorial.org/book/beginning#code-cloud_server
- **Databases (PostgreSQL):** Ruby on Rails supports most of the databases (eg. SQLite, MySQL) but we have chosen to use Postgresql as we plan to build and deploy our app on Heroku, which Postgresql is compatible with.
  https://devcenter.heroku.com/articles/heroku-postgresql
- **Messaging Platform (Faye):** Faye can be used to build a real-time push server in our application, which will serve as our messaging platform for our users to communicate with each other.
  http://code.tutsplus.com/tutorials/how-to-use-faye-as-a-real-time-push-server-in-rails--net-22600

2. Django

   Django is another web application framework that uses Python as the programming language. Django is similar to Ruby on Rails as they both follow the Model-view-controller (MVC) principle where the modelling of the domain, the application's data and the user's interaction are all separate from one another.
   - **Backups (django-dbbackup Plugin):** We can use the django-dbbackup plugin to backup our database. There are two main management commands for this plugin which are dbbackup (which backups the database) and dbrestore (which restores the database to the latest backup).
     http://django-dbbackup.readthedocs.org/en/latest/
   - **Search Engine (Haystack):** Haystack is a modular search for Django which can be used to filter for search results in a database based on the criteria given.
     http://django-haystack.readthedocs.org/en/v2.4.0/
   - **User Authentication:** Django comes with a user authentication system. It handles both authentication, which verifies who the user is, and authorization, which determines what the user can do using his account.
     https://mayukhsaha.wordpress.com/2013/05/09/simple-login-and-user-registration-application-using-django/
     https://docs.djangoproject.com/en/1.9/topics/auth/
   - **Databases:** Django supports many different databases such as SQLite, MySQL and so on, which gives us the flexibility of choosing the most suitable database for our application.
     http://pythoncentral.io/writing-your-first-python-django-application/
   - **Messaging Platform (django-messaging Plugin):** The django-messaging plugin enables users to send each other messages privately. There are several functions of the plugin that will make it easy to put a "Send XYZ company a message link" on a profile page.
     https://github.com/arneb/django-messages

We ultimately decided to use Ruby on Rails over Django as we both have more experience in using Ruby on Rails after utilising it during our Scenario Week 1 project and would feel more comfortable programming in it. In addition, there is a wide variety of documentation available on Ruby on Rails on the internet with many online tutorials that we can access to help us with building the financial portal.

### 5.1.2 Process, Context and Assumptions

From SMEs' Point of View

1. What motivates them to seek out the website?
   - Unable to secure loans from banks, thus these SMEs look to other means of financial funding
   - Looking for an investor who may be more or less involved in how the money is used depending on what the SMEs prefer

- Easy and convenient way to know of investors instead of having to attend countless networking sessions

2. What do they find on the website?
   - List of investors who have signed up for an account on the website as well
   - The background and overview of these investors
   - Contact details of the investors, the website acts as a platform for them to communicate with the investors as well
   - A search engine that narrows down a list of investors in accordance to the criteria the SMEs provide
3. What do they do once they have a set of 'matches' to follow up?
   - Information will be provided about each of their 'matches' for the SMEs to read up on and deem them suitable for their needs or not
   - A way to communicate with these investors will be provided, eg. through a platform much like a chatroom
   - Relevant contact details of these investors (email, contact number etc.) will be provided
4. How do they contact alternative finance lenders?
   - Either through the use of the contact details provided or through the communication platform integrated with the website which will be much like a chatroom/ instant messenger
5. What role, if any, does the portal owner play?
   - This depends on how involves the portal owner would want to be
   - The portal owner can just be responsible for the overall maintenance of the website or he can be involved in verifying the investors' background
6. How is the portal maintained?
   - Backups of the database can be done daily
   - Safeguards will be put in place to make sure that if the system crashes, the database will be able to be restored to the last working one

From Investors' Point of View

1. What would motivate them to subscribe?
   - Successful investments can result in very high returns for investors
   - Easy way to know about new and promising startups
   - Convenient to access information about these new startups as it can all be found on one single website
2. How would they subscribe?
   - They will be able to sign up for an account on the website while stating that they are an investor
   - They will then have to answer a series of questions that will form the criteria that will be used to help SMEs search for suitable investors
   - They will also have to enter a short paragraph profiling their background and what they do
3. What process checks would the portal need to make to verify the identity and creditworthiness of alternative business lenders?
   - Verifying the identities and creditworthiness of these alternative business lenders will be tricky as does it mean the portal owner will have to work with banks to do background checks?

- SMEs may have to exercise their own discretion when using this portal and be careful not to fall into a scam (similar websites are like Gumtree, Craiglist where there is no way to verify either parties)
- An alternative method will be for the portal owner to play a more involved role and only allow investors to have an account after they have shown proof of their financial capabilities to fund a SME

4. What, if any, role would the portal play in facilitating contact between potential lenders and potential borrowers?
   - Interested SMEs will be able to contact them through the website or through other means by the contact details they provide
   - They will also be able to contact SMEs through the same ways
5. What role, if any, would the portal play in anything other than facilitating initial contact between the parties?
   - Convenient and easy platform for SMEs and investors to discover more about each other before deciding to make initial contact

Assumptions Made

1. SMEs using this portal have been established less than five years ago
2. These SMEs have a turnover of less than £10 million per year

### 5.2 Prototypes

Both prototypes can be accessed on the website:
http://students.cs.ucl.ac.uk/2015/group5/

#### 5.2.1 Ruby on Rails Prototype

This prototype shows some basic functionalities of the financial portal. Users will be able to sign up for an account following which an email with an activation link will be sent to their email address to click on to activate their account. They will be able to view their profile page and view the list of all users once they are signed in.

Also, they can change their settings such as changing their password. If a user had forgotten his password, he can click on the “forgot password” link found on the login page which will bring him to another page that will ask him for his email associated with his account. An email will be sent to the email address with a link where he can click to reset his password.In addition, it contains links to the main Atos website as well as their news page. There are “About”, “Help” and “Contact Us” pages on the portal homepage that anyone can access.
This prototype was built with the Ruby on Rails framework in the Cloud 9 IDE and deployed using Heroku.

#### 5.2.2 User Interface Prototype

This prototype shows the planned user interface of the financial portal. It shows the layout of the website and the functionalities we aim to integrate into the portal such as the ability to create an account, login and logout of the account,

searching for candidates based on the selected criteria, being able to message other users directly and so on.

This prototype helped us to think about the portal would look from a user's point of view and how the layout can make the portal easier to use. For example, we realised that there should be a "New Search" link on every page so that users can carry out a new search at any point of time as after all, the search engine is our portal's main functionality. This prototype was built with HTML and deployed on apache server.

### 5.3 User Interface Development

Please download the user interface development documentation from the website: http://students.cs.ucl.ac.uk/2015/group5/

# 6 Technical Achievements

## 6.1 User Authentication

We used the Devise gem for user authentication, so it took care of the routes and views for the user sign-in and sign-out processes. However, due to the nature of the alternative financing portal, the application would have two different types of users: SMEs and credit providers. Hence, we decided that the sign-up route should be different for these two users and we needed to customise the user registrations controller from the default one provided by the Devise gem. The customisation resulted in separate registration routes for the different users but one common login route. We set up a polymorphic association so that the User model will be associated with the rolable model. This created a SME and a credit provider model which were associated with the user model as well using the has_one association reference. The routes were then tweaked such that users can only sign up by either "/provider/sign_up" or "/sme/sign_up" and the type of user would be saved in the database under the column "rolable_type".

After creating separate sign-up routes for the different users, the registrations view was customised such that a different view will be rendered depending on whether the user chose to sign-up as a SME or credit provider. The view rendered would show a specific set of questions that the users would have to answer. Their responses would serve as the search criteria for the search engine. Most of the responses were saved as a string in the database. For responses where several options can be selected such as the nature of funding sought for SME users or the other form of support offered for credit provider users, they were saved as a string as well with commas separating the multiple options selected.

Also, *before_action :signed_in_user, only* was added to the User controller to restrict various actions solely to signed-in users. These include access to the search and instant messaging pages as well as the ability to edit settings and other information. Users who are not signed-in will not be able to view other users' profile pages as well. All these were done to ensure that people will sign up for an account before making use of the application.

## 6.2 Search Engine

We implemented the search engine using the Filterrific gem. We added Filterrific to the User model and then defined the ActiveRecord scopes for the available filters that we have chosen. Filterrific was added to the User controller as well. Using the code from the documentation available on Filterrific, we built the view which would render on first load the Filterrific filter form and a partial that shows the actual list of users. When the filter options are changed, the partial would be updated via AJAX using an additional Javascript template. We also enabled users to access other users' profile pages adding a link for each user in the "Name of Organisation" column of the list of users.

We decided that the main way to filter the users will be by column values of the database. For each user, their selection for a search criteria would be saved in a column in the database and Filterrific would access the values in order to correctly display the suitable users.

The search criteria was provided to us by our client and we changed it marginally to suit our search engine better.

### 6.3 Profile Page

As part of the alternative financing portal, each user will have a profile page where they can display information about themselves for other users to view, such as their responses to the various search criteria, their contact details and so on. Using a Ruby on Rails form, users will be able to edit their responses for the search criteria and methods were written in the User controller such that the changes of the various params will be reflected in the database. The user can only edit his own profile page.

Another feature we implemented for the profile page is a HTML5 editor. We used Ckeditor and paperclip gems for this functionality. The paperclip gem was used to manage file uploads while ActiveRecord was used as the backend for file upload support. The model for the CKEditor, which was autoloaded for Rails 4, was generated in the *app/models/ckeditor* directory. After that, we changed the routes to mount Ckeditor:: Engine and included Ckeditor javascripts in the *app/assets/javascripts/application.js* file. For deployment, the following *Rails.application.config.assets.precompile += %w( ckeditor/* )* was added to the *config/initializers/assets.rb* file.

### 6.4 Instant Messaging Platform

Faye, Thin and Private Pub gems were used to implement a real-time chat application in the alternative financing portal. We followed a tutorial found online to add such a functionality to our application. A user woud have many conversations and a conversation would have many messages. A conversation model was generated and a conversation would contain a sender_id and recipient_id (both of which are instances of a user). The sender_id and recipient_id would both have to be unique to ensure that there will not be more than one conversation between the same users. Also, two scopes were added: *involving* and *between*. *Involving* would retrieve all existing conversations of the currently signed-in user while *between* would check whether a conversation is already present between two users before a new one is generated.

A message model was generated as well which has a body, conversation_id and user_id columns. Messages that belong to the same conversation will have the same conversation_id.

On the messages page, the list of users is displayed with a send message button beside each user's name. The button contains the sender_id (which is the id of the currently signed-in user) and the recipient_id. If there is already an existing

conversation, that particular conversation_id will be returned by the server using jQuery which will request the respective show page of that conversation with all the associated messages. This conversation data will then be displayed as a pop-up. Conversely, if there is no existing conversation, a new conversation will be created with a new conversation_id. The conversations controller contains only *create* and *show* actions.

The servers used for the instant messaging platform are different for the application hosted locally and when it has been deployed on Heroku.

Local Host
Using the command *rails g private_pub:install,* a configuration file and a Rackup file were generated to start the Faye server. Next, the command *rackup private_pub.ru –s thin –E production* was used to start up the Rack server on a separate terminal window. Private_pub was added to the application's Javascript manifest file as well. The show view was then created for the conversations controller. The subscribe_to method in the show view will subscribe to the conversation's path which will be used to publish update notifications. This method is provided by the private_pub javascript required in the application.js manifest file.

Following this, we created the messages controller which contains the create action which will render a javascript template. The conversation's path is stored in the @path variable and this path will be used to publish push notifications to the view. In addition, two helper methods *self_or_other* and *message_interlocutor* were defined and both take a message as an argument.

Deployment
A new application was created on Heroku to stand in as a second server. In the *private_pub.yml* file of this new application, the production server url would be the url of this Heroku application while the secret_token would be retrieved from the *private_pub.yml* file of the alternative financing portal application which was previously generated when the Private Pub gem was installed for the portal application. Private Pub, Thin and Foreman gems were installed for this new application and further configurations were made for the Procfile. In addition, a private_pub.ru file was created as well and deploying this new Heroku app would set up the Faye server required for the instant messaging platform.

For the *private_pub.yml* file in the alternative financing portal application, the production server url would have to be configured to the url of this new application.

### 6.5 Styling

For styling, our team made use of existing styles from the internet and customised it to suit our website better.

The styling for our website is designed such that user would have a good idea of what the website's functionalities are. At the same time, when the user is using

our website, our styling will not distract the user from the application's functionalities so that the user can carry out his tasks efficiently.

The styling of the website basically contains of the following aspects:

- Header <h> tags
- Paragraph<p> tags
- Text fields and HTML forms
- Buttons, check-boxes
- Fonts sizes, styles
- Navigation bars

**Styling for specific pages:**

**Sign-up pages:**
When the user is trying to sign up for an account, he would need to enter his details in text fields provided. We adjusted the length of the text fields so that it will fit the approximate length of the details the user needs to enter, thus saving space for a less cluttered look.

**Search page:**
For the search page, our group used a similar style as the sign-up pages. In addition, our team also included some "margin attribute" for each element so that the layout would be clearer. Also, we boxed up the specific set of search questions to search for a SME or credit provider so that users will understand on one look which set of questions to use to filter the SME or credit provider users.

# 7 Testing

For our testing of our project in COMP205P, after comparing the possible testing tools, our group decided to use "Selenium" as our testing tools for the following reasons.

1. It can be used on various platforms.
2. Tests can be automatically run on web browsers.
3. It has a "Firefox IDE" which can generate scripts easily by writing commands.
4. By combining several test cases together, a test suite can be produced and tested. This means that all the test cases can run together using a single command.

The details for the testing tool could be found on this website:
http://docs.seleniumhq.org/

Our team has decided to use two different testing strategies. They are as follows:

1. Unit Testing
   After considering the complexity of our project, our team believed that doing it manually would be sufficient and timesaving.
2. Regression Testing
   Since we are combining functionalities together to test them, we decided to use the automated testing tool. Doing it in this manner is more efficient as compared to doing it by hand. In addition, it results in a more thorough testing process.

## 7.1 Testing Plans

**Unit testing: (manually)**

1. Test that all individual buttons are clickable.
2. Test that all the pages in our application can be rendered correctly and viewed by the user.
3. Test to see if the CSS styling is working for each individual page by using the "inspect page source" functionality in the web browser.
4. Test to see if the information by the user is stored in a database by checking the database file.

**Integration testing: (using script)**

1. Test to see if all the links in the home page are working
   - Clicking any link or button on the homepage should direct the user to another page with the exception of the "Home" button which would direct the user back to the homepage.
   - Clicking on the sign-up buttons as either SME or Credit Provider should direct the user to two different sign-up pages.
   - If the search button is clicked when the user is not signed in, the user should be directed to the page requesting him to sign up for an account.
2. Test to see if user can register for an account by entering the relevant details.

    - Signing up as a SME or Credit Provider should require the user to enter different details depending on whether the user is signing up as a SME or Credit Provider.
    - All details need to be entered in order to register for an account.
3. Test to see if the user can log in to the website using the details that were just entered by the user when signing up.
    - Test for both SME and Credit Provider accounts.
4. Test to see if all the links are working after user has signed into the website.
    - Used a similar checking process as '1'.
5. Test to see if the user can edit his profile page.
    - Make changes to existing information. After clicking on the 'Save Changes' button, the information should be updated and saved.

**Acceptance testing:**

- Our team has asked a group of students to use our website in order to get their feedback on the website's functionalities and visual aesthetics.
- Our team let students navigate the website on their own to get feedback on how easy it is to understand and learn how to use the functionalities of the application.

**Regression testing:**

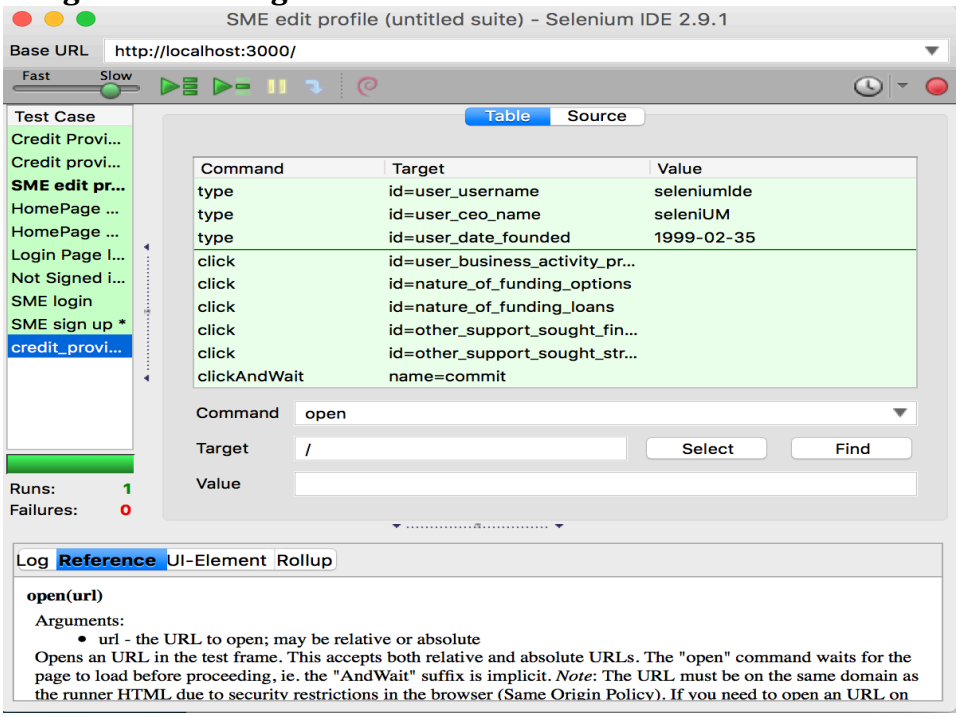
- The functionalities for our website have not changed after the website has been developed, although there were some changes in the 'CSS'. However, 'CSS' shouldn't affect the testing code since it only alters the appearance of the website.
- If there will be changes in functionalities in the future, the testing can be simply carried out by editing the commands in the script.

### 7.2 Testing Results

**Unit Testing:**

The website passed all the unit tests successfully. We did it 20 times manually in total and all the tests were successful for every round.

## Integration Testing:




Green indicates that the test case is automatically tested, and there were no errors during the test.
From the picture shown above, the website successfully passed all the test cases and no error occurred.

## Acceptance Testing:

The general feedbacks from the testing are the following:

General Comments:

- The layout of the home page is clear. The links shown allowed the user to easily understand which pages they will be directed to depending on the links they clicked.
- User could enter various details about their company in their profile page.
- The styling on the whole website is appropriate as it does not distract users from their tasks.
- The sequence of search criteria on the search page was easy to comprehend.

Suggestions for Improvements:

- The website could have more interactive features (e.g one user could add another user as friend and voice chat with each other).
- The website could have responsive designs such that the layout would be displayed correctly for all screen sizes.
- It was not clear to users that search results are refreshed automatically without the need to press a search button.
- When editing some selections eg. nature of funding for the user's profile page, the previous selection was not highlighted which would lead to some confusion for the user on what his selection is.

- It would be more convenient for users if there was a way to search for a specific user in the messages page.

**Regression Testing:**
Not applicable as no changes were made to the functionalities.

### 7.3 Evaluation of PoC

Based on our testing results, all functionalities of the application are working well concurrently and do not have any known bugs. Also, with our user acceptance tests, feedback has shown that those who had the chance to use our application generally agreed that it was easy to use and navigate. We have improved the application based on the feedback we have receieved as well, such as stating on top of the search page that search results are refreshed automatically.

Also, with reference to section 2.5 (Criteria for Success), we feel that we have met all the criteria for success. However, we do acknowledge that there is still room for improvement but with our limited capabilities, we are pleased with the PoC that we have built. We have detailed the potential future developments for this application in the section 9 of this report.

# 8 Project Management

## 8.1 Tools Used

1. Trello

Trello is a website application. It keeps track of the progress of a project using cards. First, we created a Trello board for our COMP205P project. Under that board, we created multiple lists as follows:

- Programming Tasks
- Project Documentation
- Client Liaison
- Styling
- In Progress
- Done
- Discussion

We added cards under each list as tasks we needed to perform for each area. Throughout our project, if we had more things to do for any of those areas, we would create a new card for it and add it in the respective list.

When we were performing a certain task, we would drag the card under the list "In Progress" so that one team member would be aware of the other was currently doing for the project. When a task was completed, it will be dragged and placed into the "Done" list. Also, if there was anything we were unsure of and wanted the other person's opinion, we would drag it into the "Discussion" list. We always referred to the "Discussion" list during our meetings to talk through the issues we were facing and come up with possible solutions.

As Trello automatically sends an email to the team members if any changes has been made to the cards, we were kept up-to-date on each other's individual progress without having to constantly ask each other what was happening. It also helped to keep track of the progress of our project and what issues we were facing.

2. Github

Github was used to manage our application source code. As a version control system, it was essential in keeping our code updated whenever a team member made a change. It also serves as a backup for our latest working version.

3. SourceTree

As we were not completely comfortable with navigating around Github using the command-line, we downloaded and installed SourceTree in our local machines. SourceTree is a free Git client for Windows or Mac and serves as a Graphical User Interface for Git. By using the SourceTree application's interface, it was much easier to see the Git-flow of our project visually. We could see which files have been changed before commiting the changes. The branches were also shown

graphically. This application was essential to us as it allowed us to use Git with ease.

### 8.2 Work Packages Completed

Heidi

- Wrote biweekly reports
- Programmed the application and implemented all features
- Liaisoned with Mr Michael Davison and Mr John Davenport (clients from ATOS)
- Wrote documentation for project website

Bill

- Wrote biweekly reports
- In charge of styling for both the application and project website
- Tested the application
- Wrote testing documentation for project website

# 9 Possible Future Developments

## 9.1 Technical Aspects

1. Search Function

There could be more options to filter the users better so that a more refined and specific search result will be returned. For example, SMEs could input the amount of funding they are requesting for and credit providers can utilise that to search for SMEs based on the amount they are able to fund.

For options where multiple selections could be selected by a user (eg. Nature of Funding, Other Support Offered), users could also search by selecting multiple selections for those options. Currently, users can only search by one selection (eg. Loans), but as a future development, users should be able to search by more than one selection (eg. Loans and Options). We were unable to implement such a functionality successfully due to time constraints but have explored the possibility extensively. A potential solution could be Harvest's chosen multi-select widget.

2. Messages Function

A search function could be implemented to allow users to search for a specific user to message. We understand that being presented with a long list of registered users can seem daunting and scrolling through the whole column to find one user is not the ideal way. One suggestion we can make for now is to use the browser's search function to find the user. However, we are well aware that there is definitely room for improvement for this functionality but due to a lack of time, we were unable to search for a possible solution for this issue. Thus, this would be a future development that would certainly result in a more polished instant messaging platform for the application.

## 9.2 Project Development Aspects

1. Mobile Application

The Alternative Financing Portal was programmed to be web application. However, smartphones usage has proliferated in recent years. People of all walks of life spend a substantial amount of time on their smartphone, with this being significantly true for businessmen. Hence, we propose, as a future development, the Alternative Financing Portal can be programmed as a mobile application for both OS and Android users.

This move into the mobile application market will make it easier for users to access the functionalities of the application at anytime and anywhere. Such increased convenience will appeal greatly to the people of today where time is precious.

2. Verification of Account

Currently, anyone can sign up for an account on the Alternative Financing Portal. This means that there is no way to verify whether the credit providers or the SMEs on the portal are genuine. As online scams are becoming increasingly common, people may shy away from utilising the portal to find alternative sources of finance and this will result in a wasted opportunity for SMEs in finding funding or investing in a SME full of potential for credit providers.

As such, we suggest that if users would want to increase their credibility, they can be verified by the administrator of the application and this verification will be displayed on their profile page. Such a service can be offered at no cost or at a small fee to ATOS.

# 10 Manuals

## 10.1 System Manual

**System Requirements:**

- Ruby Language (version 1.9.3 or newer)
- RubyGems Packaging System (installed with Ruby versions 1.9 and later)
- A working installation of SQLite3 Database

**Databases Used:**

- SQLite3 (Development)
- PostGres (Deployment)

**How to Run the Application Locally:**

1. Install Rails

   Refer to http://guides.rubyonrails.org/getting_started.html (Section 3.1 Installing Rails) on how to install Ruby on Rails and get it up and running.

   There are various tools available to install Ruby on Rails quickly on different operating systems. Windows users can use Rails Installer (http://railsinstaller.org/en) and Mac users can use Tokaido (https://github.com/tokaido/tokaidoapp).

   For a more detailed guide, refer to http://railsapps.github.io/installing-rails.html to install Ruby on Rails depending on your operating system.

2. Obtain Source Code

   Go to the Github Repository and download the source code. The Github link is as follows: https://github.com/heiditan/comp205p-afp.git

3. Start up Rails Server

   Unzip the file downloaded from Github. Open a command line prompt. Navigate to the downloaded folder using command line. Run the command "`rails server`". This may result in a prompt stating that some gems have not been installed. Follow the instructions shown on the terminal window to install the necessary gems. Once installed, run the command again. The terminal window should display something like the image as shown below.

```
dhcp-13-7:sysengg aktan$ rails s
=> Booting Thin
=> Rails 4.2.2 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
=> Ctrl-C to shutdown server
Thin web server (v1.6.4 codename Gob Bluth)
Maximum connections set to 1024
Listening on localhost:3000, CTRL+C to stop
```

Keep this server running and access the application in your web browser using the link: http://localhost:3000

4. Start up Rack Server

   This server is essential for the instant messaging platform to work on your local machine. Open a new terminal window. Run the command "`rackup private_pub.ru –s thin –E production`" to start the Rack server. It is crucial to keep this serving while running the application or else the instant messaging platform will not work.

5. Done! The application should be running smoothly on your local machine. If you encounter any problems, contact the team at heidi.tan.14@ucl.ac.uk or jiaao.ma.14@ucl.ac.uk

## 10.2 User Manual

### 1. Signing up for an account

Anyone can sign up as a user on the Alternative Financing Portal. There are two types of users:
- Small and medium enterprises (SMEs)
- Credit providers

On the homepage of the portal, there are two links "Sign up now as a credit provider!" and "Sign up now as a small and medium enterprise!". Click either one of these links depending on what type of user you are signing up as. This will then take you the sign-up page where you will have to enter your email, address, contact details and answer a list of questions as well. Click the "Submit" button once you are done.

You will then be directed to the homepage with a banner stating "Welcome! You have signed up successfully."

### 2. Changing account settings

Click on the "Settings" tab found on top of the webpage after you have signed in. You will be able to change your email, address and contact number here. Enter your new information for any of those fields. For the fields you would want to remain the same, leave the information unchanged. Enter your password and confirm your password, then click "Save Changes".

If your changes were successfully saved, a banner would appear stating "Settings have been successfully saved."

### 3. Searching for a user

Click on the "Search" tab found on top of the webpage after you have signed in. **Please note that search results are displayed automatically and there is no "Search" or "Submit" button to press for results to be shown.**

Enter any phrase in the first search field that is titled "Search". You can also sort the users based on:

- Email (a-z)
- Registration date (newest first)
- Registration date (oldest first)
- Name of Organisation (a-z)
- Organisation Type (SME or Credit Provider)

The next search criteria "Are you looking for a SME or credit provider?" will display users depending on the organisation type. If "SME" is chosen, then only users who have signed up as SMEs will be displayed and vice versa if "Credit Provider" is chosen.

Following this, there are options to narrow down the SMEs or credit providers even further. If "Credit Provider" was previously selected, then the questions in box A will help to refine the search for a suitable credit provider. If "SME" was previously selected, then the question in box B will help to refine the search for a suitable SME.

If you would like to learn more about a specific user, just click on the name of the organisation which serves as a link to that specific user' profile page.

Should you try to access the search function without signing in, the page will direct you to sign up for an account. This is to ensure only users with an account can utilise this function.

### 4. Editing profile

Click on the "Profile Page" tab found on top of the webpage after you have signed in. This page displays all your information including your email, address and contact number. Other users will be able to view this page.

To edit this page, scroll to the bottom of the page and click on the link "Edit Profile". This will then direct you to a page where you can make any changes to the information displayed on your profile page except for your **address, email and contact number** (these will have to be changed through the "Settings" tab).

For some search criteria (eg. Nature of Funding, Nature of Financing, Other Support Offered, Other Support Sought) where multiple selections can be made, the current selection will be shown in **red and bold**. To make changes to these criteria, simply tick the relevant checkboxes to choose your new selection or leave it empty to keep it unchanged.

Click on the "Save Changes" at the bottom of the page once you are satisfied with your edits. A banner would appear stating that "Profile has been successfully updated." if your changes were saved successfully.

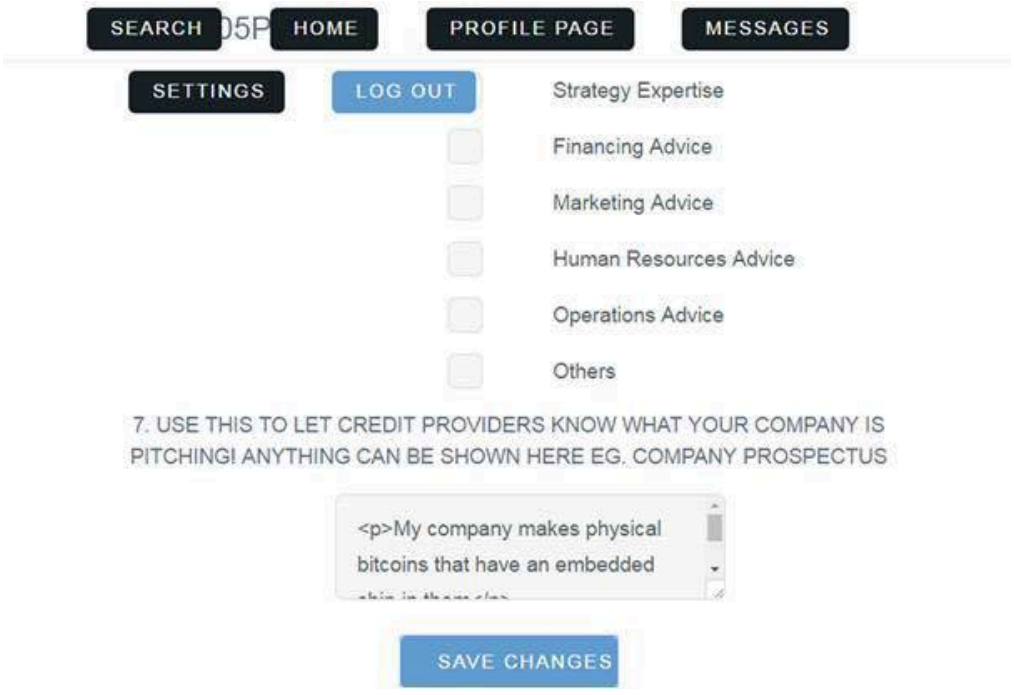
## 5. Messaging a user

Click on the "Messages" tab found on top of the webpage after you have signed in. There is a list of "Registered Users" where you can find any user. Click on the "Send Message" button next to their names to send them a message. A chat window will pop up on the webpage with the user's name as the header. There is a text box at the bottom of the window. Type your message in that and press the enter button on your keyboard to send the message. It will appear instantly in the chat window.

There is another column titled "Your Conversations" below the list of "Registered Users". This shows your current and previous conversations with any user you have spoken to. Simply click on any of the conversation to view it.

## 6. Application view

Depending on the zoom view of your web browser, the layout of the application may look disordered as seen as the image below.



A quick fix would be to press the command button and the – button for Mac users to zoom out so that the buttons in the header will be arranged in a single line. The same can be achieved for Windows users by pressing the Windows button and – button.

# 11 Materials Referenced

1. Devise Gem: https://github.com/plataformatec/devise
2. Multiple sign-up routes but one login route customisation: http://stackoverflow.com/questions/7299618/multiple-user-models-with-ruby-on-rails-and-devise-to-have-separate-registration http://stackoverflow.com/questions/9472852/devise-and-multiple-user-models
3. Filterrific: http://filterrific.clearcove.ca/
4. Ckeditor: https://github.com/galetahub/ckeditor
5. Tutorial for instant messaging platform: http://josephndungu.com/tutorials/gmail-like-chat-application-in-ruby-on-rails
6. Deployment on Heroku for instant messaging: http://www.thegreatcodeadventure.com/deploying-private-pub-on-heroku/