

# 4th Year Project – Technical Architecture

Don Valino

Contents

[4th Year Project – Technical Architecture 1](#_Toc466933742)

[1. Use Case Diagram & Descriptions – Jake Valino 3](#_Toc466933743)

[(i) Use Case Diagram 3](#_Toc466933744)

[A. Register: 4](#_Toc466933745)

[B. Manage User Account: 5](#_Toc466933746)

[C. Handle Complaints: 5](#_Toc466933747)

[D. Post Jobs: 6](#_Toc466933748)

[E. Search Jobs: 7](#_Toc466933749)

[F. Place an Offer: 7](#_Toc466933750)

[G. Accept Offer: 7](#_Toc466933751)

[H. Pay Deposit: 7](#_Toc466933752)

[I. Pay Balance: 7](#_Toc466933753)

[(ii) Use Case Descriptions 8](#_Toc466933754)

[Register 8](#_Toc466933756)

[Manage User Account 9](#_Toc466933758)

[Handle Complaints 10](#_Toc466933760)

[Post Jobs 11](#_Toc466933762)

[Search Jobs 12](#_Toc466933764)

[Place an Offer 13](#_Toc466933766)

[Accept an Offer 14](#_Toc466933768)

[Pay Deposit 15](#_Toc466933770)

[Pay Balance 16](#_Toc466933772)

[2. Technical Architecture – Jake Valino 17](#_Toc466933773)

[Software Components: 17](#_Toc466933774)

[Platform libraries: 17](#_Toc466933775)

[Distribution and Deployment: 17](#_Toc466933776)

[Risk: 18](#_Toc466933777)

[3. Prototype – Jake Valino 18](#_Toc466933778)

[Testing Strategy: 18](#_Toc466933779)

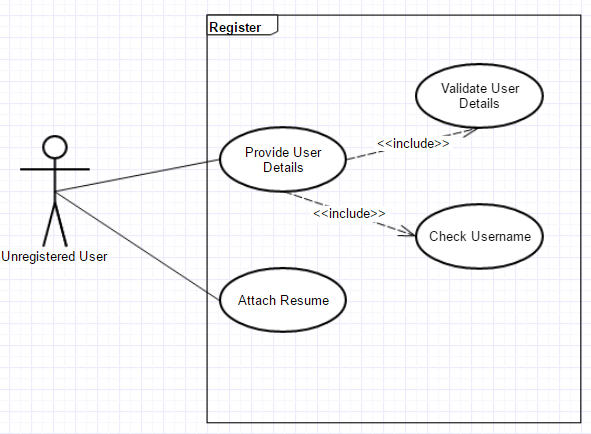
[Prototype Deliverable for week 8: 19](#_Toc466933780)

[Prototype Deliverable for week 11: 19](#_Toc466933781)

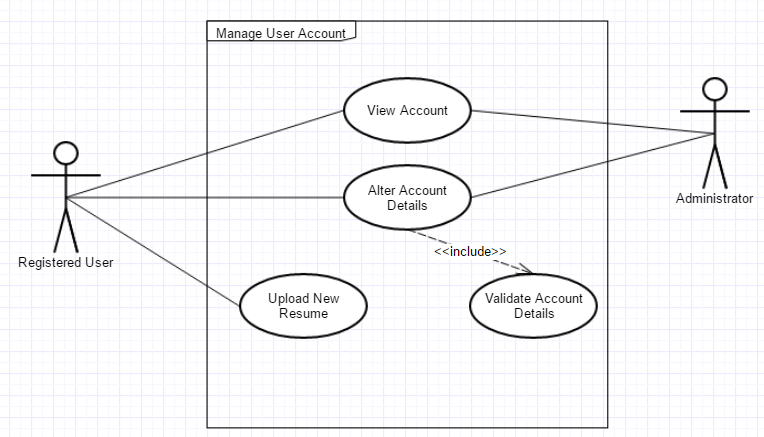
# Use Case Diagram & Descriptions – Jake Valino

## (i) Use Case Diagram

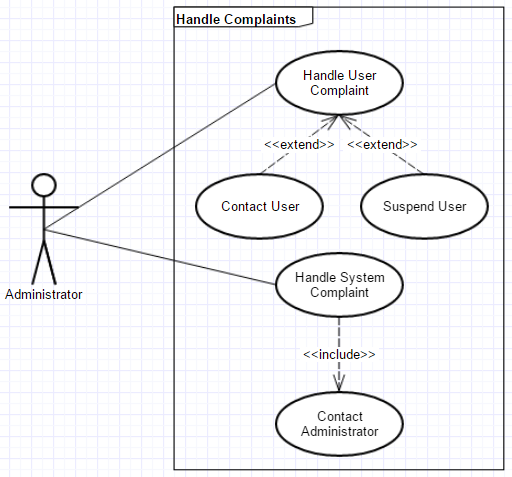
## **Register:**



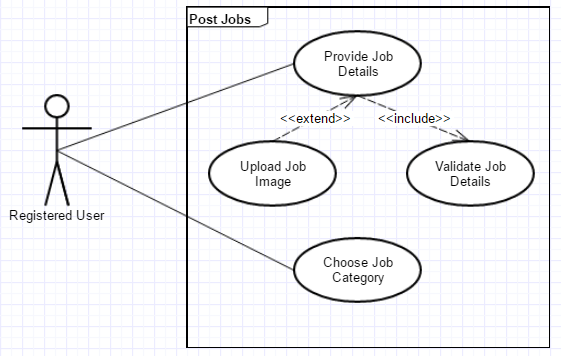
## **Manage User Account:**



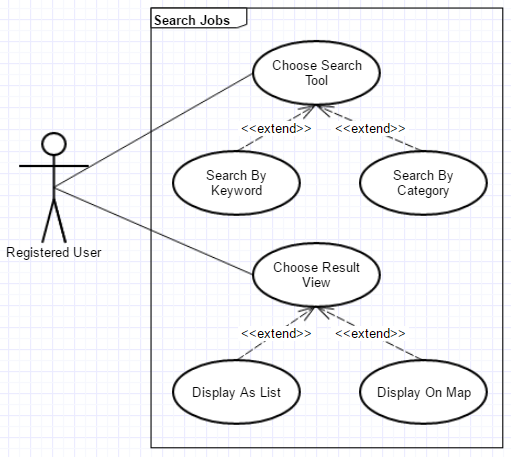
## **Handle Complaints:**



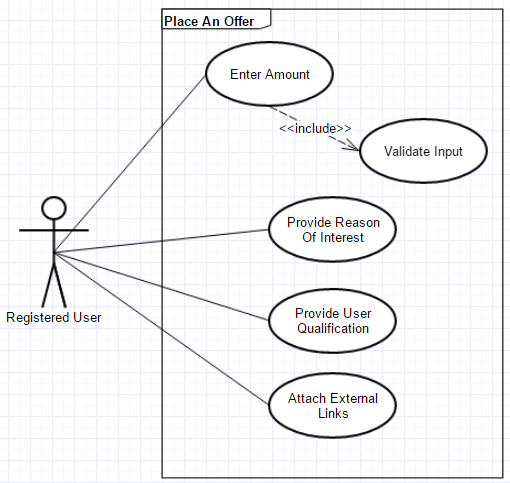
## **Post Jobs:**



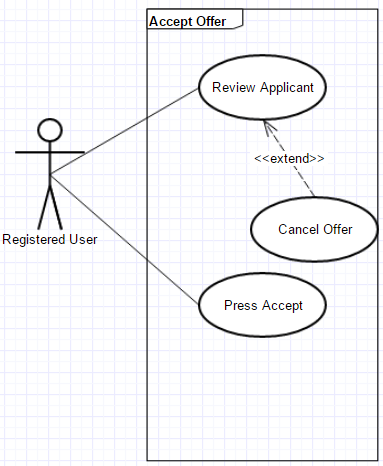
## **Search Jobs:**



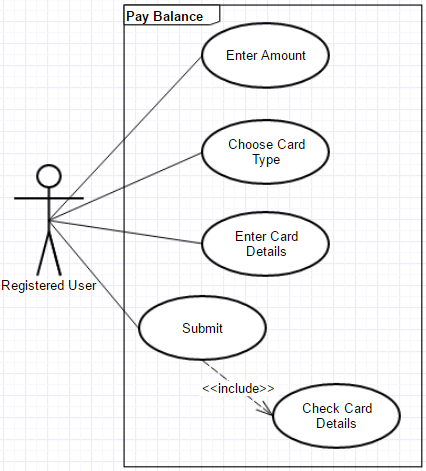
## **Place an Offer:**



## **Accept Offer:**



## **Pay User:**



## **(ii) Use Case Descriptions**

|  |  |
| --- | --- |
| **Use Case Description 1** | |
| **Use Case:** | Register |
| **Actor (s):** | Unregistered User |
| **Goal:** | The goal of this use case is to register an unregistered user. By registering, the user can access the various functionalities of the system such as to search or post jobs. |
| **Overview:** | This use case implements a form in which the user can fill the information required to register. The form allows the user to type their first & last name, age, unique username, password, email and phone number. It also allows them to upload their resume. The data will then be validated and the user will be directed to the home page. |
| **Type** |  |
| **Pre-Conditions** | User must be new to the system (i.e. Must be an Unregistered User). |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. Unregistered User clicks the ‘Register A New Account’ link. This redirects the user to a page containing a form in which they can register. |
|  | Step 2. The user enters their first & last name, age, unique username, password, email as well as their phone number. |
| Step 3. If they wish, the user can choose to select a file attachment from their computer that contains their resume. |
| Step 4. The data will be validated. |
| Step 5. Validation is successful and the user is redirected to the home page. |
| **Alternative Scenario (s)** | The username already exists in the system and therefore the Unregistered User is asked to select a different username. |
|  | The user refuses to register because of trust related issues (e.g. they don’t trust the system to handle their bank details). |
|  | Validation of data provided fails and the user is asked to amend the data |
|  |  |
| **Use Case Description 2** | |
| **Use Case:** | Manage User Account |
| **Actor (s):** | Registered User, Administrator |
| **Goal:** | The goal of this use case is to allow Registered Users and Administrators manage their personal account. |
| **Overview:** | The user is displayed a page where they are provided with an overview of their account. The overview displays the status of the jobs in which they placed an offer on, the jobs they accepted and the jobs they posted. Also present is their profile picture, a copy of their resume and other links (e.g. LinkedIn links). The user can also alter information about their account such as to change their email. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. User views their account to check the status of the jobs in which they had placed an offer on or the jobs they posted. |
|  | Step 2. User changes some data about their account (e.g. User changes their phone number / email address). A new user resume file attachment can also be uploaded. |
| Step 3. The data will be validated. |
| Step 4. Validation is successful and all the changes is saved. |
| Step 5. The user signs out or returns to the home page. |
| **Alternative Scenario (s)** | Validation of data provided fails and the user is asked to amend the data. |
|  | File attached for the user’s resume is too large and cannot be uploaded. |

|  |  |
| --- | --- |
| **Use Case Description 3** | |
| **Use Case:** | Handle Complaints |
| **Actor (s):** | Administrator |
| **Goal:** | The goal of this use case is to enable the administrators to handle user complaints that are related to the operation of the system. |
| **Overview:** | The administrator can receive complaints relating to the system (e.g. a problem with one of the website’s functionality) or to a user. The administrator is not required to reply if the complaint is related to the system. Instead, the administrator must consult the developer about it. If the problem is related to a user complaint, proper action must be taken (e.g. suspend a user account). |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. Administrator receives complaints about the system or about a user |
|  | Step 2. If it is a system complaint, the administrator contacts the developer for consultation and to resolve the issue. |
| Step 3. If it is about a user complaint, the administrator must investigate (e.g. contact the user for proof) and take proper actions. |
| Step 4. |
| Step 5. |
| **Alternative Scenario (s)** | The complaint was cancelled by the user who created it. |

|  |  |
| --- | --- |
| **Use Case Description 4** | |
| **Use Case:** | Post Jobs |
| **Actor (s):** | Registered User |
| **Goal:** | The goal of this use case is to allow Registered Users to post jobs into the website for advertisement. |
| **Overview:** | This use case implements a form in which the user can fill the information required to post a new job advertisement. The form allows the user to type the job’s name, description, the minimum accepted rate for offer, qualification requirements and the duration. The data will then be validated and the job advertisement will be posted. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. Registered User clicks the ‘Post A New Job’ link. This redirects the user to a page containing a form in which they can provide information about the job. |
|  | Step 2. The user types the job’s name, description, the minimum accepted rate for offer, qualification requirements and the duration. A category relating to the job is also selected by the user. |
| Step 3. If they wish, the user can choose to select an image file attachment from their computer to display an image about the job. |
| Step 4. The data will be validated. |
| Step 5. Validation is successful and the user is redirected to the home page. |
| **Alternative Scenario (s)** | The user account contains to many active posted jobs and therefore the new job advertisement cannot be posted. |
|  | User’s account is suspended for some reason and therefore cannot upload any new advertisement. |
|  | Validation of data provided fails and the user is asked to amend the data. |

|  |  |
| --- | --- |
| **Use Case Description 5** | |
| **Use Case:** | Search Jobs |
| **Actor (s):** | Registered User |
| **Goal:** | The goal of this use case is to enable the Registered Users search for jobs advertisements in the website. |
| **Overview:** | This use case provides various search tools. From the homepage, the user can choose to use the search bar and search for jobs based on the keywords. The user can also search jobs based on the category or location of a job. The search result can be viewed using various preferences ranging from a map that displays the jobs based on location or as a webpage, containing the results organised into a list. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. User searches for jobs using the various search tools such as the keyword in the search bar and searching based on the category or location. |
|  | Step 2. The user selects their preference on how they would like to view the search result (e.g. View as an organised list or as a map) |
| Step 3. If the user views the search result as a list, they are provided with the functionality to sort it alphabetically (A – Z or Z – A). |
| Step 4. If the search result is displayed on a map, the user can view the jobs based on their location |
| Step 5. The user selects a job and review the description about it. |
| **Alternative Scenario (s)** | No interesting job was found by the user. |
|  | Relevant jobs were not found because of inconsiderate search (e.g. user searches for a job that belongs to another category). |
|  | The keyword specified by the user does not match any jobs advertisement. |

|  |  |
| --- | --- |
| **Use Case Description 6** | |
| **Use Case:** | Place an Offer |
| **Actor (s):** | Registered User |
| **Goal:** | The goal of this use case is to enable Registered Users place an offer into a job in which they are interested in. |
| **Overview:** | In the job description page a place an offer button will be provided. Upon clicking this button, a dialog appears. Here, the user can type the amount for the offer and the reason why they are interest and / or qualified for the job. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. User presses the place an offer button. A dialog appears. |
|  | Step 2. The user enters the amount in euro for the offer. This indicates how much the user is willing to do the job for. |
| Step 3. The reason of user’s interest and / or qualification is typed below in the dialog text box. |
| Step 4. Any external links is pasted by the user in the text box (e.g. LinkedIn Profile) |
| Step 5. The user presses submit button. |
| **Alternative Scenario (s)** | User types an incorrect input for the amount in euro. This causes the program to crash. |
|  | User accidentally places an offer in the wrong job. |
|  | The user account is suspended due to some violation and cannot place any new offer. |

|  |  |
| --- | --- |
| **Use Case Description 7** | |
| **Use Case:** | Accept an Offer |
| **Actor (s):** | Registered User |
| **Goal:** | The goal of this use case is to enable Registered Users accept an offer made by another user into their advertised job. |
| **Overview:** | In the job description page a new offer in the comment section appears. This is an offer made by another user to do the job that is being advertised. To accept this offer, the user who advertised the job can simply click the accept button. This will redirect them to a page where they can pay the deposit. To decline the offer, they can click the decline button. A dialog appears where they can state the reason for declining the offer. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. User is presented a new offer. |
|  | Step 2. The applicant for the job is reviewed by the user. |
| Step 3. If the applicant is qualified, the user presses the accept button and they are then redirected into a page where they can pay the deposit. |
| Step 4. If the applicant is not fit for the job, the user presses the decline button. |
| Step 5. A dialog then appears where the user can type their reason for declining the applicant’s offer |
| **Alternative Scenario (s)** | No offer was made to the job being advertised. |
|  | User declines the wrong applicant. |
|  | The offer was cancelled by the applicant. |

|  |  |
| --- | --- |
| **Use Case Description 8** | |
| **Use Case:** | Pay Deposit |
| **Actor (s):** | Registered User |
| **Goal:** | The goal of this use case is to enable Registered Users pay the deposit for the job after accepting an offer from another Registered user. |
| **Overview:** | After clicking the accept button, the user is redirected to a page where they can pay the 20% deposit. In this page, the user enters their credit / debit card details. They then press the submit button to successfully pay the deposit. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. The amount in which the user wishes pay for the deposit is typed (i.e. minimum is 20 % deposit). |
|  | Step 2. Users choses their card type |
|  | Step 3. The card number and name of the user is typed. |
| Step 4. The start and end date of the card is selected. |
| Step 5. The ccv of the card is typed. |
| Step 6. The user presses a submit button and a confirmation is shown. |
| **Alternative Scenario (s)** | User selects the incorrect card type. |
|  | The credit / debit card number and / or name is incorrect. |
|  | The ccv of the card is incorrect. |
|  | User changes their mind and presses the cancel button instead. |

|  |  |
| --- | --- |
| **Use Case Description 9** | |
| **Use Case:** | Pay Balance |
| **Actor (s):** | Registered User |
| **Goal:** | The goal of this use case is to enable Registered Users pay the balance of 80 % to another user after the job has been completed. |
| **Overview:** | Once the deed has been settled and the job has been completed, the Registered user is required to pay the balance left to the other user who did the job. The process here is like paying a deposit. The user enters their credit / debit card details. They then press the submit button to successfully pay the deposit. |
| **Type** |  |
| **Pre-Conditions** |  |
| **Post-Conditions** |  |
| **Special Requirements** |  |
| **Successful Scenario:** | Step 1. User types the amount to pay the balance. Note: user can give tips by paying more than the balance. |
|  | Step 2. Users choses their card type |
|  | Step 3. The card number and name of the user is typed. |
| Step 4. The start and end date of the card is selected. |
| Step 5. The ccv of the card is typed. |
| Step 6. The user presses a submit button and a confirmation is shown. |
| **Alternative Scenario (s)** | User selects the incorrect card type. |
|  | The credit / debit card number and / or name is incorrect. |
|  | The ccv of the card is incorrect. |
|  | User changes their mind and presses the cancel button instead. |

## Technical Architecture – Jake Valino

## Software Components:

**Development:** I will be using **PHP** and following the Model, View and Controller (**MVC**) model. Using PHP will enable different implementation possibilities for various features such us the login / register page or displaying the list of users in a table.

**Database:** For data definition (creating/altering tables, creating indexes etc.) and data manipulation (adding/removing columns, creating constraints etc.), I plan to use **MYSQL** with **PHPMYADMIN** as the Integrated Development Environment (IDE).

**Server:** During the development phase, I plan to deploy my project into a localhost test server called **‘Wamp’**. By deploying my project into Wamp, I can test whether the features of my website are working properly or if a bug exist. If there are implementation errors, Wamp will identify these and display the line number with the description of the error in the webpage of the browser. **Using Wamp will also enable access to PHPMYADMIN for MYSQL.**

## Platform libraries:

**Twitter Bootstrap (CDN) Library:** To improve the user interface or to implement complex graphical components, I will be using the libraries of Twitter Bootstrap (CDN). With Twitter Bootstrap, I can also distinctly align components in my page easier using grid classes.

**Stripe PHP Library:** I plan to use Stripe for payment integration. Stripe is an American technology company that allows both private individuals and businesses to accept payments over the Internet.

## Distribution and Deployment:

1. **Session:** This website will implement the use of PHP sessions to keep track of the user’s activity and to ensure that the user is recognised. By being able to recognise the user, we can ensure that they can stay logged in as the session will remain valid in the duration. It is only when the timeout in the session expires or when the user logs out of their account that the session is destroyed.

**Deployment:** For localhost deployment, I plan to use **‘Wamp’**. This should enable me to test the features that I have implemented and ensure that there are no errors. If there are implementation errors, Wamp will identify and display these in the browser making it easier to amend it. For live deployment, I plan to use **‘Window Azure’** With Azure, I can deploy my project using Git, set the PHP version, use a start file that is not in the root application directory and access environment-specific variables.

1. **Security:** To improve security, I plan to carefully structure the implementation of my website using the Model, View, Controller (MVC) model. With this model, there are layers of protection as implementation is subdivided into different parts, making it harder for anomalies to hack the website. In my project files, there are 4 implementation directories:

* **View:** The PHP webpages will be stored in this directory. The webpage will provide the graphical user interface that will enable the user to interact and perform operations on the website. The views will interact with the controller classes to perform the user operations.
* **Controller:** This directory contains all the controller classes which will be used to implement the graphical component of the view and to also implement methods that the view can invoke. The methods in the controller classes will communicate with the methods inside the Model classes.
* **Model:** This directory contains all the model classes and the database credentials. The methods inside a model class is used to connect to the database using the database credentials and to perform operations such as inserts or updates to database data. Only the methods of the controller classes can invoke the methods stored in the model classes. The view cannot directly communicate with the model.
* **Entities:** This directory contains the entities of the project which is mapped to the table structure of the database. The entities are classes in the project with contains variables that matches the columns of a table stored in the database. Entities are used by the model classes to create an instance which will contain values of data from a table in the database.

## Risk:

The criteria that I consider as risks to my project is the inability to deploy my website into Window Azure or to implement payment integration using Stripe into the system. Deploying a PHP website into Windows Azure involves multiple vital steps which tends to increase the complexity involved. While on my placement, I deployed Maven projects into the company’s live Tomcat Server which is located overseas in Sweden. However, I have never deployed a project into any Cloud Service Provider. I will therefore need to carefully examine Windows Azure to investigate the type of services for my requirements and to determine if there will be any unexpected cost that comes with deploying a PHP website into Azure.

I have also never implemented a system that incorporate a payment integration functionality. Therefore, I will also need to study Stripe to discover how to implement and integrate it as part of my system.

## Prototype – Jake Valino

## Testing Strategy:

* **Test Case Matrix**: All the features that need to be implemented are listed into test cases. All the test cases will be initially configured as failed. After testing the features individually, the test cases in the matrix will be updated appropriately to state whether it had passed or failed.
* **PHPUNIT:** Created by Sebastian Bergmann on GitHub, PHPUnit is a unit testing framework for the PHP programming language. It is an instance of the xUnit architecture for unit testing frameworks that originated with SUnit and became popular with JUnit.

## Prototype Deliverable for week 8:

1. Register
2. Manage User Account

## Prototype Deliverable for week 11:

1. Search Jobs
2. Post Jobs