Software Development Life Cycle

SWD505

Engineering Software Systems

Case Study 5

Solent Property Solutions

**Group Members**

Dean Leszman  
Mahtarr Jeng  
Fraser Provan  
Denils Kronberg  
Nicola Chigumburra

Contents

[Introduction 4](#_Toc481673411)

[Phase 1 Project Planning 5](#_Toc481673412)

[Belbin test 5](#_Toc481673413)

[Task management 6](#_Toc481673414)

[Planning and Design 6](#_Toc481673415)

[Implementation 6](#_Toc481673416)

[Concept Map 7](#_Toc481673417)

[Risk Analysis 8](#_Toc481673418)

[Gantt Chart 9](#_Toc481673419)

[Activity Networks 10](#_Toc481673420)

[Phase 2 Analysis & Design 12](#_Toc481673421)

[Software Development Methodologies 12](#_Toc481673422)

[System Requirements 13](#_Toc481673423)

[Structure Chart 14](#_Toc481673424)

[Domain Model 15](#_Toc481673425)

[Use-case 16](#_Toc481673426)

[Use-case texts 17](#_Toc481673427)

[Robustness Diagrams 26](#_Toc481673428)

[Sequence diagrams 35](#_Toc481673429)

[Class diagrams 43](#_Toc481673430)

[Phase 3 Design 45](#_Toc481673431)

[Wireframe Mockup 45](#_Toc481673432)

[FXML Mockup 47](#_Toc481673433)

[Phase 4 Implementation 49](#_Toc481673434)

[GitHub 49](#_Toc481673435)

[Testing 49](#_Toc481673436)

[System Test 49](#_Toc481673437)

[Client Tab (By Fraser) 49](#_Toc481673438)

[Employees Tab (By Fraser) 50](#_Toc481673439)

[Test case (By Mahtarr) 51](#_Toc481673440)

[Phase 5 Maintenance and Support 52](#_Toc481673441)

[Possible changes of this software in the future 52](#_Toc481673442)

[Users guide 53](#_Toc481673443)

[Introduction 53](#_Toc481673444)

[The System (Describing the system) 53](#_Toc481673445)

[Installing, Starting and Stopping the System 54](#_Toc481673446)

[Step by Step Instructions for using the system 55](#_Toc481673447)

[References 61](#_Toc481673448)

# Introduction

**‘BORIS**’ is a software development firm established since 2010, the organisation has an outstanding reputation for developing fantastic software systems. **‘BORIS’** won the contract to develop a property management system for Solent Building and Development Services Ltd.

**Solent Building and development Service Ltd** is a building development company that offers building development service to clients. This service incorporates the development, repair and renovation of clients’ buildings. The company is renowned for their outstanding services to their clients and has grown over the years. To effectively improve their services and the management of daily activities, the company will like to develop a “Property development management System”. Not only will this system seamlessly improve company-client relationships, it is also aimed to help the company shed any waste possibilities, which in turns increase the company’s profits.

Each section of the SDLC document will have authors listed in terms of their contribution to that section of the project. For example, the first name will have contributed the most to the given part of the project, second member the second most contribution.

# Phase 1 Project Planning

## Belbin test

Authors: All team members

This test determines the roles of group members on this project, on completion of the exercise it states each member’s results. Thoughts were discussed with the lecturer and we all agree to our allocated roles.

**Belbin Test Results**

|  |  |
| --- | --- |
| **Member** | **Role** |
| Nicky Chigumburra | Shaper |
| Fraser Provan | Completer-finisher |
| Denils Kronberg | Team worker |
| Dean Leszman | Implementer |
| Mahtarr Jeng | Implementer |

## Task management

Authors: Mahtarr Jeng, Dean Leszman, Fraser Provan

**Key**

* **Work not completed**
* **In progress**
* **Work completed**

### Planning and Design

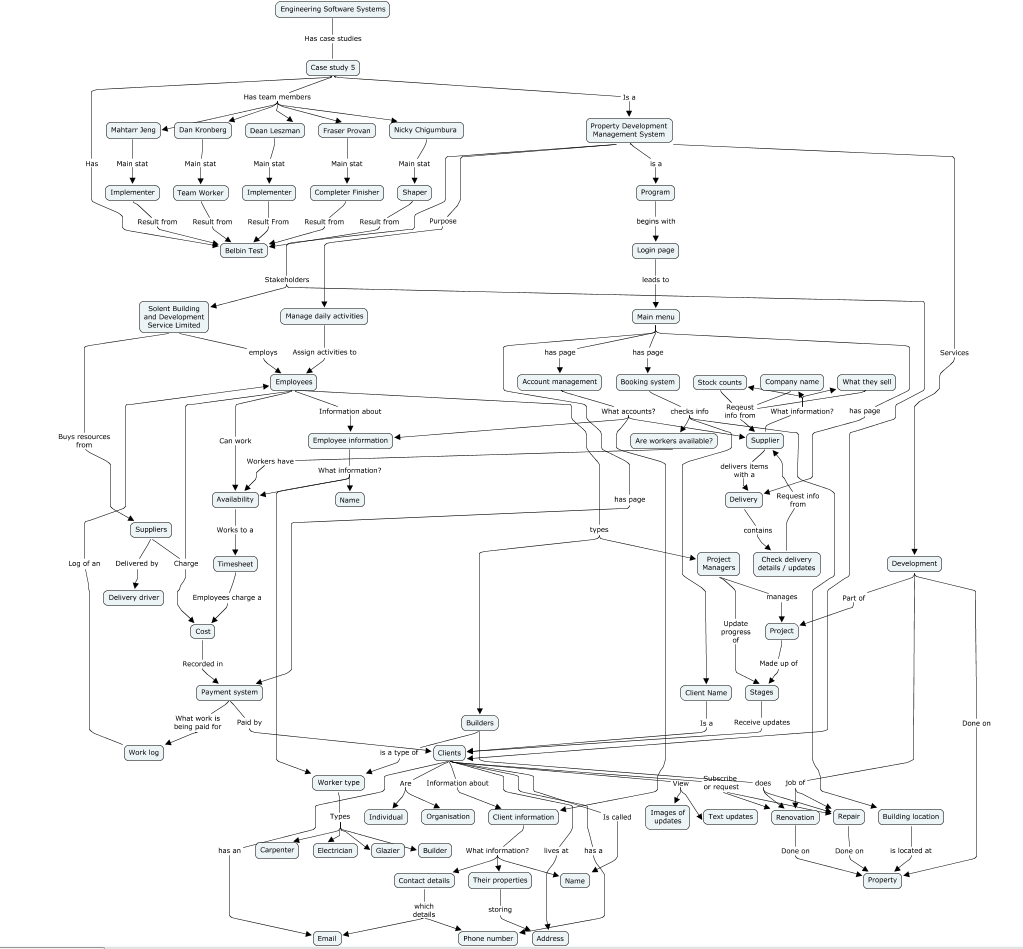
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Task No** | **Task Name** | **Mahtarr** | **Dean** | **Nicky** | **Dan** | **Fraser** |
|  | Belbin Test |  |  |  |  |  |
|  | Task Management |  |  |  |  |  |
|  | Concept Map |  |  |  |  |  |
|  | Work Breakdown Structure |  |  |  |  |  |
|  | Activities Network |  |  |  |  |  |
|  | Critical Path Analysis |  |  |  |  |  |
|  | Gantt Chart |  |  |  |  |  |
|  | Communication Groups |  |  |  |  |  |
|  | Github Setup |  |  |  |  |  |
|  | Risk Analysis |  |  |  |  |  |
|  | System Requirements |  |  |  |  |  |
|  | System Development Methodologies |  |  |  |  |  |
|  | Domain Model |  |  |  |  |  |
|  | Use Case Diagrams |  |  |  |  |  |
|  | Use Case Texts |  |  |  |  |  |
|  | Robustness Diagrams |  |  |  |  |  |
|  | Sequence Diagrams |  |  |  |  |  |
|  | Class Diagrams |  |  |  |  |  |
|  | Structure Chart |  |  |  |  |  |
|  | GUI Design |  |  |  |  |  |
|  | FXML Mockup |  |  |  |  |  |

### Implementation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Task Name** | **Mahtarr** | **Dean** | **Nicky** | **Den** | **Fraser** |
| 1 | Login page | ✓ |  |  |  |  |
| 3 | Suppliers / stock orders tab |  |  |  | ✓ |  |
| 4 | Employees |  |  | ✓ |  |  |
| 5 | Clients |  |  | ✓ |  | ✓ |
| 6 | Subscription & Booking |  | ✓ |  |  |  |
| 7 | Testing | ✓ | ✓ |  |  | ✓ |

## Concept Map

Author: Dean Leszman



## Risk Analysis

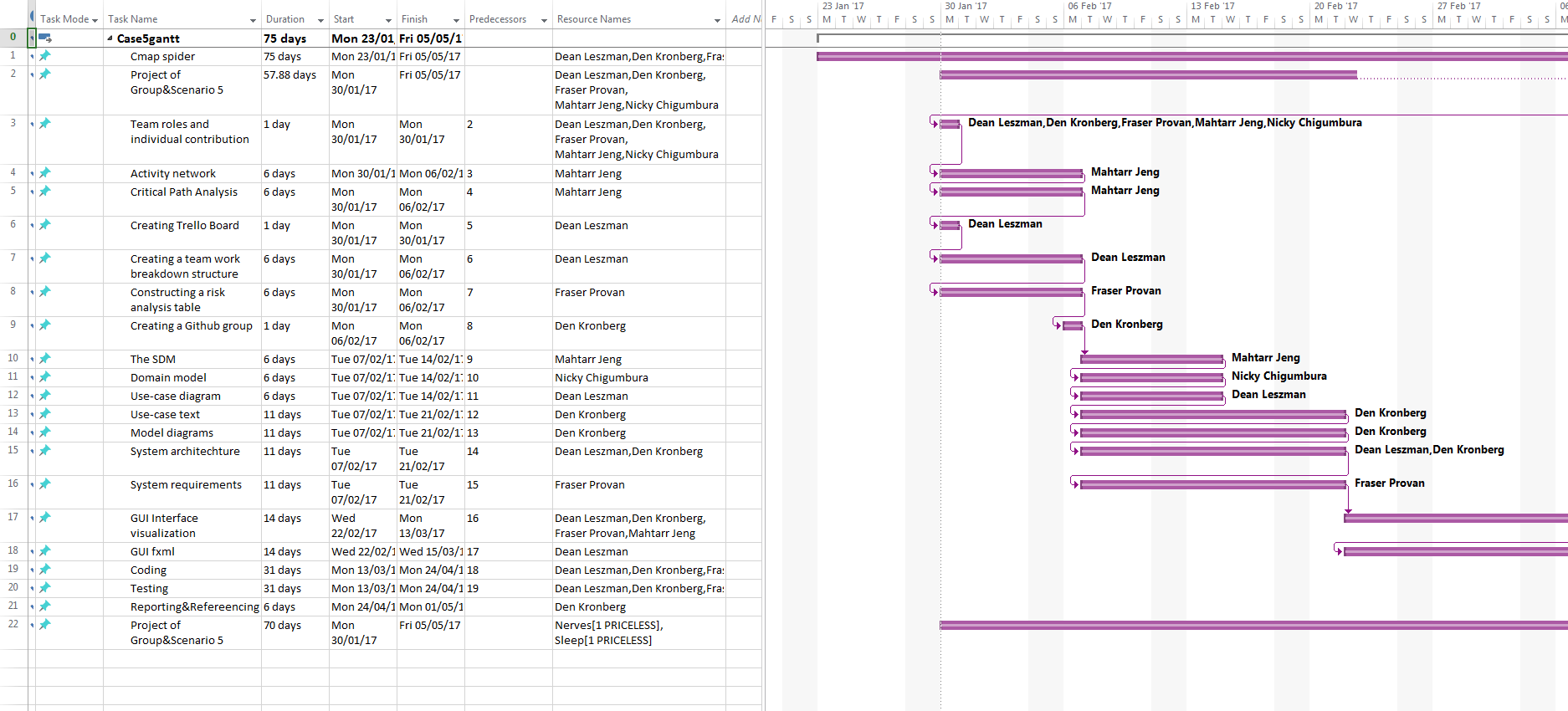
Authors: Fraser Provan

**Risk Analysis Table**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Risk event description and impact area | Impact rating | Probability | Risk Score | | Risk Response Description | Trigger |
| 1 Time management | | | | | | | |
| 1a | Not following design | High | High | 1 | Follow designs closely | | Inexperience |
| 1b | Working outside of the scope | High | Low | 2 | Complete the functionality to the specification | | Going off topic/ using time on not required tasks |
| 1c | Poor estimations | Low | High | 3 | Improve future estimations | | Incorrect estimations on tasks |
| 1d | Poor time allocation | Low | High | 3 | Use contingency time efficiently | | Not use Gantt chart appropriately |
| 1e | Too little Contingency time | Low | High | 3 | Not relying on contingency time | | Planning on using contingency time in advance |
| 2 Resource Management | | | | | | | |
| 2a | Lack of using resources (GitHub, Trello) | High | Low | 2 | Use all given resources appropriately | | Wasting time not using resources |
| 2b | Lack of Resources (No area to work) | Low | High | 3 | Find alternative ways to work (skype) | | Can’t find space to work (comps together) |
| n/a | |  |  |  |  | |  |
| 3 | Attendance of team members | High | Low | 2 | Persistent messaging | | Not wanting to come in (Illness + Other excused) |
| 4 | Plagiarism | High | Low | 2 | Not stealing code | | Using stolen code |
| 5 | Financial issues | Low | Low | 4 | Not being able tom complete task with our resources | | Only using resources we have |
| 6 | Errors in code | Low | Low | 4 | Give time to fix errors | | Not fixing errors when created |

## Gantt Chart

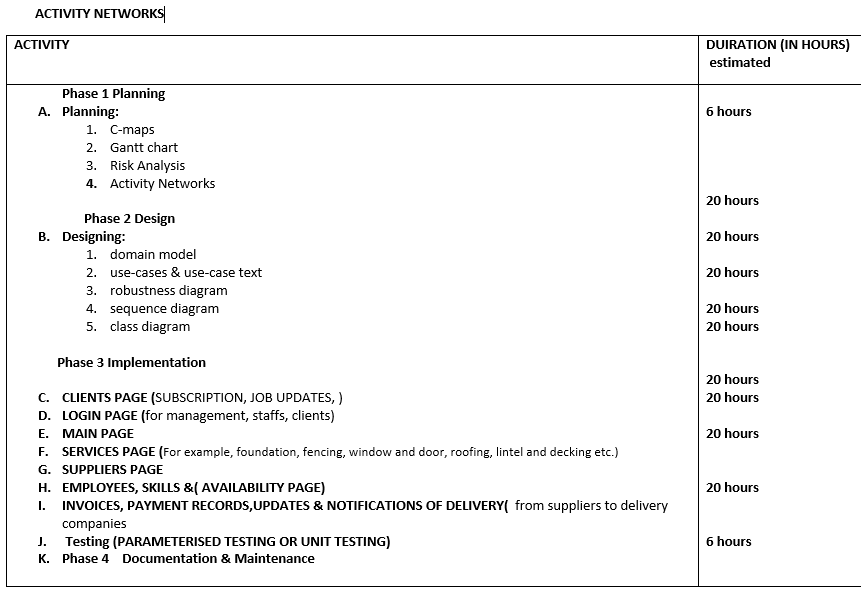
Authors: Denils Kronberg

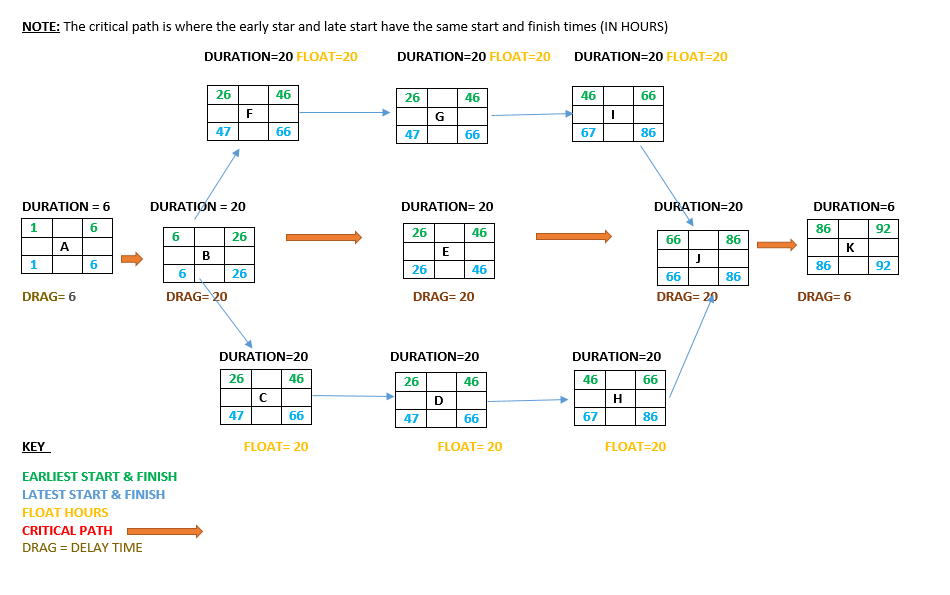


## Activity Networks

Author: Mahtarr Jeng, Dean Leszman

It is very essential as it illustrates all stages of the project and its duration process. The critical path of an activity is where the early start and early finish are the same of the late start and late finish. This means the activities that fall under the critical path have no flexible times and they need to be done within the allocated time frame. A pert chart was use to demonstrate all the activities, its early start and finish times, late start and finish times, drag time and critical path.





# Phase 2 Analysis & Design

## Software Development Methodologies

Author: Mahtarr Jeng

Waterfall Model was initially used on this project during the planning, analysis and design phase

Agile Development

|  |
| --- |
| Implementation |

This methodology IS used in the implementation stage, because it will save time and different tasks can be done without waiting on any individual to complete their

Maintenance

Design

Planning

AGILE DEVELOPMENT

## System Requirements

Authors: Fraser Provan, Mahtarr Jeng

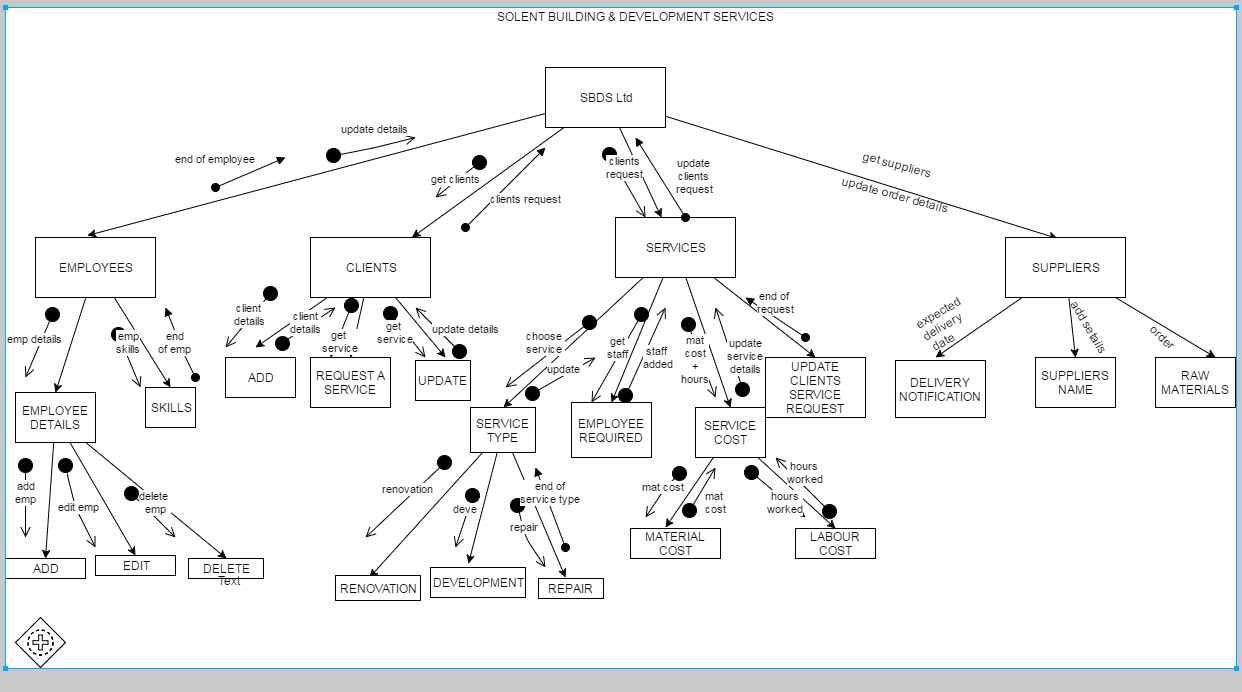
The table below outlines the requirements of the final software product.

|  |  |  |
| --- | --- | --- |
| Ranking | Requirement | Requirement Type |
| 1 | Subscribe to companies rendered services | Functional |
| 2 | Security – Different logins give different access | Non-functional |
| 3 | Create/Add Client | Functional |
| 4 | Request a service | Functional |
| 5 | - Subscribe to companies rendered services | Functional |
| 6 | Give clients updates using notifications (images) | Functional |
| 7 | Purchase building supplies from suppliers | Functional |
| 8 | - Allocate Jobs to staff depending on skills | Functional |
| 9 | Portability – Can be ran on portable device (e.g. USB, portable HDD) | Non-functional |
| 10 | Maintainability | Non-functional |
| 11 | Employee list that includes skills and availability | Functional |
| 12 | View payment records (Invoices) | Functional |
| 13 | Stability – Chance of crashing is low | Non-functional |
| 14 | Usability – Easy to navigate | Non-functional |
| 15 | Software allows altering size of stages (windows) | Non-functional |

## Structure Chart

Author: Mahtarr Jeng

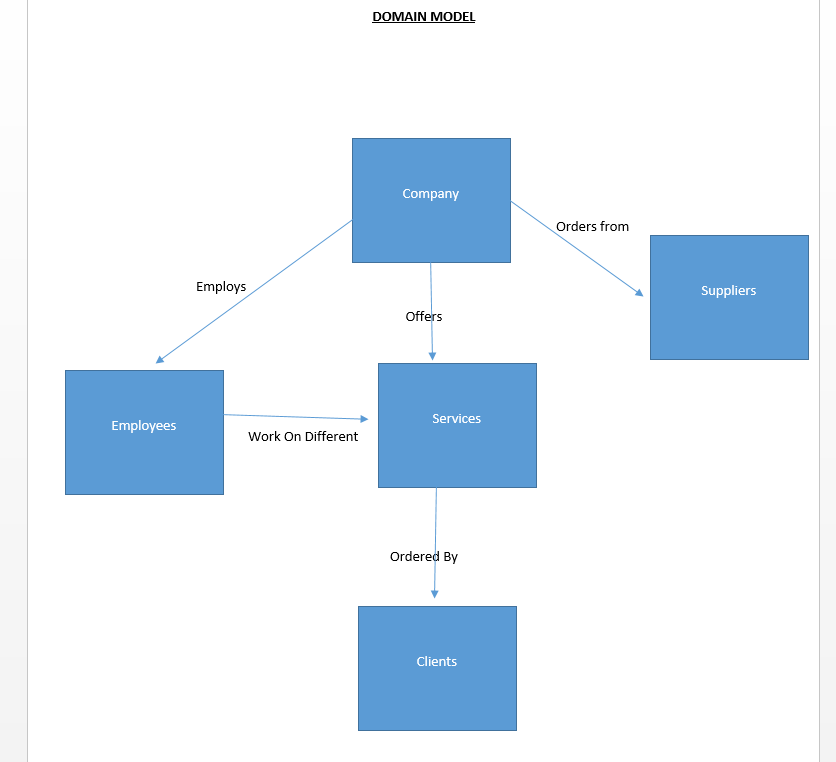
This outlines the whole program in drawing and it shows how many classes, methods or sub-classes your program will require and how they interact with each other.



## Domain Model

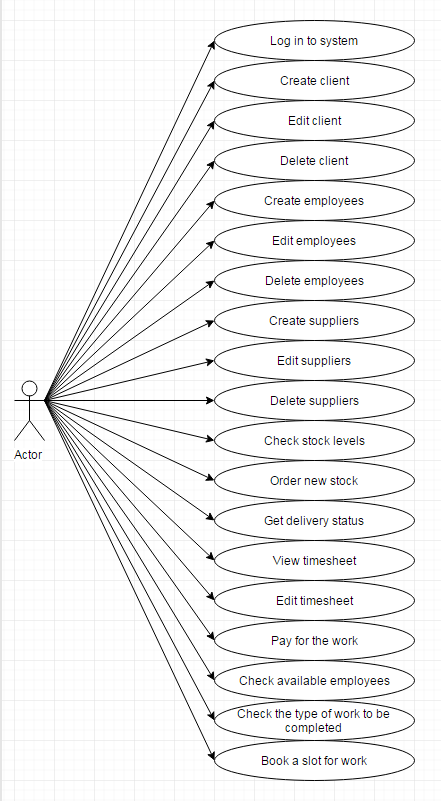
Author: Mahtarr Jeng

In this diagram, possible classes are shown and their interactions. They are extracted from the clients request scenario. This is done by identifying the nouns or main players in the passage.



## Use-case

This diagram considers things from the user’s perspective, (who are the user? How can they interact with system), our system must be able to do all this below.



## Use-case texts

The use-case texts breaks down the use-case in series of steps, describing user’s interactions with the system to complete a use-case.

**Create Employee (By Dean)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Step** | **User Actions** | **System Actions** | |
| User will be unable to access this use case if they are not logged in to an Admin account | | | |
| 1 | User selects to create new employee | |  |
| 2 |  | | System displays form to enter new employee |
| 3 |  | | System alerts user to enter new details |
| 4 | User enters details for the new employee | |  |
| 5 | User clicks ‘Save’ | |  |
| 6 |  | | Details are checked and validated to ensure they follow the standard format for employee details |
| 7 |  | | If all details are valid, go to step 11. |
| 8 |  | | If any details are invalid, the save is cancelled, the user is alerted which are invalid |
| 9 | User corrects any invalid details | |  |
| 10 | Return to step 5 | |  |
| 11 |  | | New ID is given to the employee |
|  |  | | Details are saved as a new employee record |
|  |  | | System confirms employee is created |

**Delete Employee (By Dean)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| User will be unable to access this use case if they are not logged in to an Admin account | | |
|  | User selects ‘Delete employee’ from list of existing employees |  |
|  |  | Employee details are loaded into a form |
|  |  | Systems alerts confirming they want to delete the employee |
|  | If user selects ‘Yes’ |  |
|  |  | Employee is deleted and confirmation is sent back to the user |
|  | If user selects ‘No’ |  |
|  |  | Confirm employee is not deleted and unselect the employee |

**Edit Employee (By Dean)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| User will be unable to access this use case if they are not logged in to an Admin account | | |
| 1 | User selects ‘Edit employee’ from list of existing employees |  |
| 2 |  | Employee details are loaded into the form |
| 3 | User edits existing details |  |
| 4 | User clicks ‘Save’ |  |
| 5 |  | Details are checked and validated to ensure they follow the standard format for employee details |
| 6 |  | If details are valid, go to step 10. |
| 7 |  | If any details are invalid, the save is cancelled and the user is alerted which details are invalid. |
| 8 | User corrects any invalid details |  |
| 9 | Return to step 4. |  |
| 10 |  | Employee details are saved |
| 11 |  | Confirmation is given to user |

**Login (By Mahtarr)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **Systems Response** |
| 1 | The use-case begins when user wants to login into the system |  |
| 2 |  | System prompts user to enter login details. (username, password ) |
| 3 | User enters details specified in step 2 |  |
| 4 |  | Systems checks validity (IF username and password are correct) |
| 5 |  | IF incorrect **alert** user that details entered are wrong. **GO BACK STEP 3** |
| 6 |  | System displays main page |

**Order New Stock (By Mahtarr)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **Systems Response** |
| 1 | The use-case begins when user is ordering new stock |  |
| 2 |  | System prompts user to select supplier and enter item details (item name, quantity) |
| 3 | User selects and enters details specified in step 2 |  |
| 4 |  | Order is completed (system displays a message) |

**Checking Stock (By Mahtarr)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **Systems Response** |
| 1 | The use-case begins when user is checking stock levels |  |
| **2** |  | Systems prompts user to select item |
| **3** | user selects a specific item |  |
| **4** |  | System checks if item is in stock (system displays quantity available) |
| **5** |  | If item is out of stock (system displays out of stock) |

**Create Supplier (By Dan)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| 1 | The use case begins when an admin selects create supplier option |  |
| 2 |  | System displays form to enter new supplier details |
| 3 |  | System asks user to enter new details |
| 4 | User enters details to create a new supplier |  |
| 5 | User clicks ‘Save’ |  |
| 6 |  | Details are checked and validated by system (e.g. valid address, etc) |
| 7 |  | If all details are valid, go to step 11. |
| 8 |  | If any details are invalid, the save is cancelled, the user is warned which are invalid |
| 9 | User corrects any invalid details |  |
| 10 | Return to step 5 |  |
| 11 |  | New ID was generated about supplier |
|  |  | Details are saved as a new supply record |
|  |  | System confirms supplier is created |

**Delete Supplier (By Dan)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| 1 | The use case begins when the admin selects to delete existing supplier details |  |
| 2 |  | Supplier details are loaded into a form |
| 3 |  | Systems warns confirming they want to delete the supplier |
| 4 | If user selects ‘Yes’ |  |
| 5 |  | Supplier is deleted and confirmation is sent back to the user |
| 6 | If user selects ‘No’ |  |
| 7 |  | Confirm supplier is not deleted and unselect the supplier |

**Edit Supplier (By Dan)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| 1 | The use case begins when the admin selects to modify existing supplier details |  |
| 2 |  | Supplier details are loaded into the form |
| 3 | User edits existing details |  |
| 4 | User clicks ‘Save’ |  |
| 5 |  | Details are checked and validated to ensure they follow the standard format for supplier details |
| 6 |  | If details are valid, go to step 10. |
| 7 |  | If any details are invalid, the save is cancelled and the user is warned which details are invalid. |
| 8 | User corrects any invalid details |  |
| 9 | Return to step 4. |  |
| 10 |  | Supplier details are saved |
| 11 |  | Confirmation is given to user |

**Create Client (By Dan)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| 1 | The use case begins when an admin selects create new client option |  |
| 2 |  | System displays form to enter new client details |
| 3 |  | System asks user to enter new details |
| 4 | User enters details to create a new client |  |
| 5 | User clicks ‘Save’ |  |
| 6 |  | Details are checked and validated by system (e.g. valid DOB, etc) |
| 7 |  | If all details are valid, go to step 11. |
| 8 |  | If any details are invalid, the save is cancelled, the user is warned which are invalid |
| 9 | User corrects any invalid details |  |
| 10 | Return to step 5 |  |
| 11 |  | New ID was generated about the client |
|  |  | Details are saved as a new client record |
|  |  | System confirms client is created |

**Edit Client (By Dan)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| 1 | The use case begins when the admin selects to modify existing client details |  |
| 2 |  | Client details are loaded into the form |
| 3 | User edits existing details |  |
| 4 | User clicks ‘Save’ |  |
| 5 |  | Details are checked and validated to ensure they follow the standard format for client details |
| 6 |  | If details are valid, go to step 10. |
| 7 |  | If any details are invalid, the save is cancelled and the user is warned which details are invalid. |
| 8 | User corrects any invalid details |  |
| 9 | Return to step 4. |  |
| 10 |  | Client details are saved |
| 11 |  | Confirmation is given to user |

**Delete Client (By Dan)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User Actions** | **System Actions** |
| 1 | The use case begins when the admin selects to delete existing client details |  |
| 2 |  | Client details are loaded into a form |
| 3 |  | Systems warns confirming they want to delete the client |
| 4 | If user selects ‘Yes’ |  |
| 5 |  | Client is deleted and confirmation is sent back to the user |
| 6 | If user selects ‘No’ |  |
| 7 |  | Confirm client is not deleted and unselect the client |

**Get delivery status (By Fraser)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **Systems Response** |
| 1 | User types login details into system | - |
| 2 | - | System logs into admin |
| 3 |  | System opens main stage |
| 4 | User clicks on delivery status under deliveries | - |
| 5 | - | System opens delivery status |

**Check available (By Fraser)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **Systems Response** |
| 1 | User types login details into system | - |
| 2 | - | System logs into admin |
| 3 |  | System opens main stage |
| 4 | User clicks on employee list | - |
| 5 | - | System opens employee list |
| 6 | User selects available employees from the employee list page | - |
| 7 | - | System opens list of available employees |

**Type of work completed (By Fraser)**

|  |  |  |
| --- | --- | --- |
| **Step** | **User’s Actions** | **Systems Response** |
| 1 | User types login details into system | - |
| 2 | - | System logs on to either admin or user login |
| 3 | - | System opens main stage |
| 4 | User can see the job type under subscription on the main menu | - |

**Subscribe to any of the company’s rendered services (By Nicky)**

|  |  |  |
| --- | --- | --- |
| Step | Actor action | System action |
| 1 | The use case begins when the Organisation finds new rendered services | - |
| 2 | - | System prompts the user for display of service details (Info, price, images) |
| 3 | The organisation subscribes on any company rendered services specified in step 2 | - |
| 4 | - | System checks that user is not subscribed already |
| 5 | - | System creates new subscriptions and saves in the system |
| 6 | - | System displays subscriptions details for user to see |

**Client notification (By Nicky)**

|  |  |  |
| --- | --- | --- |
| Step | Actor action | System action |
| 1 | The use case begins when the organisation selects to make updates for clients | - |
| 2 | - | System prompts the user for the client details (Login, name, etc.) |
| 3 | Organisation enters the details specified in step 2 | - |
| 4 | - | System checks existence of details |
| 5 | - | System displays agent details for updating |
| 6 | Organisation sends new deals and news to highlighted client or clients specified in step 5 | - |
| 7 | - | System checks validity of details |
| 8 | - | System updates all deals for client and saves it |
| 9 |  | System displays message that notifications were sent to clients |

**Purchasing supplies (By Nicky)**

|  |  |  |
| --- | --- | --- |
| Step | Actor action | System action |
| 1 | The use case begins when the organisation selects to purchase a new supply | - |
| 2 | - | System prompts the user for the supply details (Name, delivery date, price etc.) |
| 3 | The organisation enters the details specified in step 2 | - |
| 4 | - | System checks validity of details, e.g. date of delivery is sensible |
| 5 | - | System checks that order does not exist already (same name, same price) |
| 6 | - | System saves order details on system |
| 7 | - | System displays purchase for users and clients to see. |

**Payment records (By Nicky)**

|  |  |  |
| --- | --- | --- |
| Step | Actor action | System action |
| 1 | The use case begins when the organisation selects to record a purchase | - |
| 2 | - | System prompts the user for the supply details (Name, delivery date, price etc.) |
| 3 | The organisation enters the details specified in step 2 | - |
| 4 | - | System checks validity of the invoice |
| 5 | - | System checks that invoice does not exist already (same name, same price) |
| 6 | - | System saves invoice details on system |
| 7 | - | System displays payment invoice for user to see |

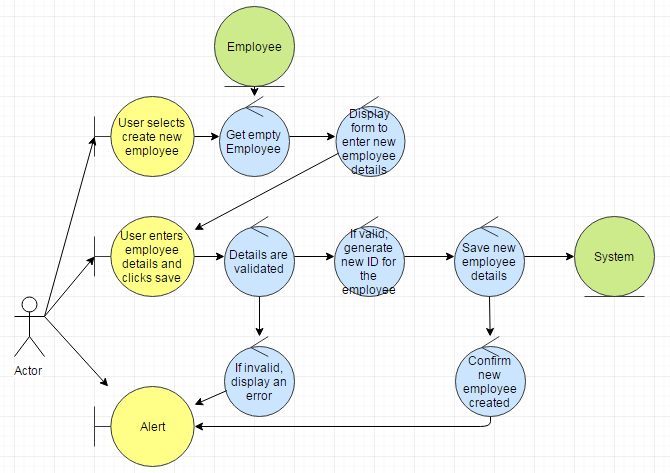
**Allocating jobs to staff (By Nicky)**

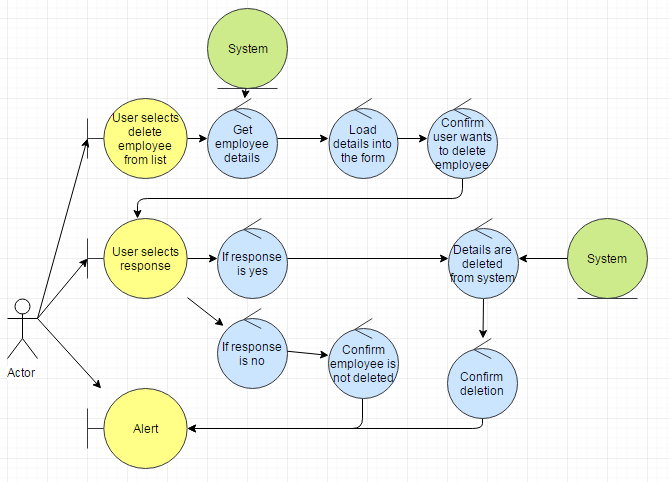
|  |  |  |
| --- | --- | --- |
| Step | Actor action | System action |
| 1 | The use case begins when the organisation selects to allocate job to staff | - |
| 2 | - | System prompts the user for the staff details (Login, name, etc.) |
| 3 | Organisation enters the details specified in step 2 | - |
| 4 | - | System checks existence of details |
| 5 | - | System displays staff details for allocating the job |
| 6 | Organisation allocates the job to staff depending on skills described in step 5 (like work experience and degree) | - |
| 7 | - | System checks validity of job allocated |
| 8 | - | System allocates new job to the staff |
| 9 |  | System displays message that job was allocated to selected staff member |

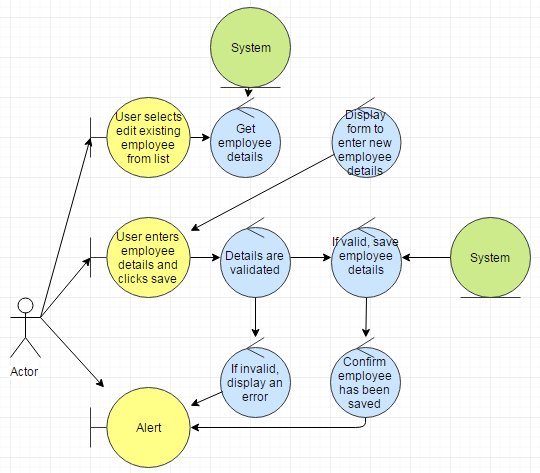
## Robustness Diagrams

* "Sanity check" on use cases - drawing out the use-case in a graphical form can help us check that the original use case text was sensible (Rosenberg and Stephens, p104)
* It also enables you to discover new classes, which we have not previously identify in the system (Rosenberg and Stephens, p106)
* Gives good understanding of how the steps of a use-case interacts with the classes in the domain model

**Create Employee (By Dean)**



**Delete Employee (By Dean) **

**Edit Employee (By Dean)** 

**Login (By Mahtarr)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Login_Robustness_Diagram.png

**Order New Stock (By Mahtarr)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\order_new_stock_robustness_Diagram.png

**Checking Stock (By Mahtarr)** C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\checking_stock_robustness_Diagram_(1).png

**Create Supplier (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Create_supplier2.png

**Delete Supplier (By Dan)** *C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Delete_supplier.png*

**Edit Supplier (By Dan)**

*C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Edit_supplier.png*

**Create Client (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Create_client2.png

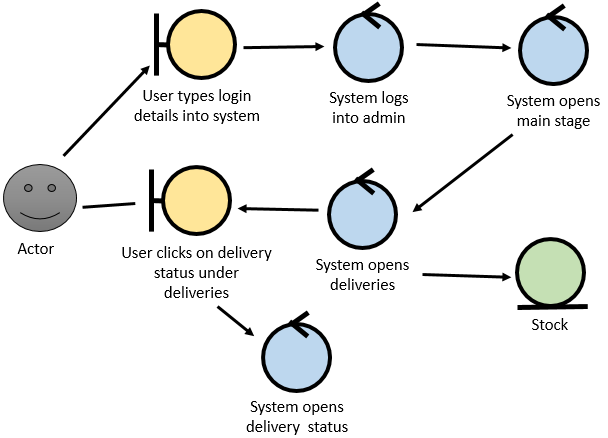
**Edit Client (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Edit_client2.1.png

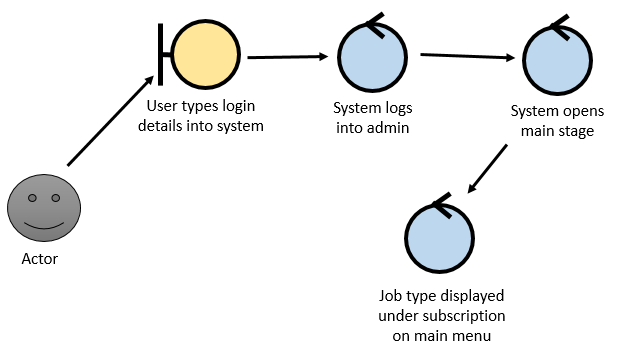
**Delete Client (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Delete_client2.png

**Get delivery status (By Fraser)**

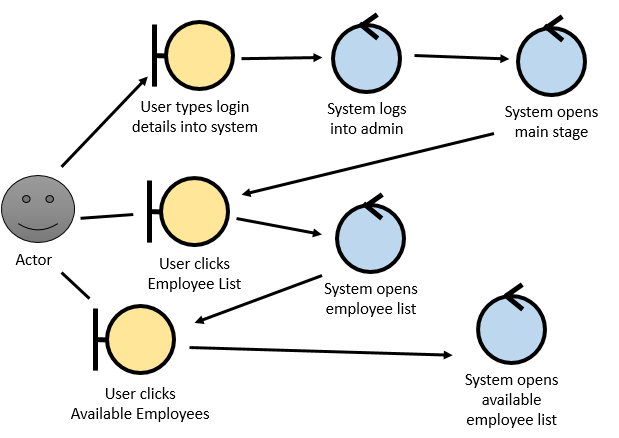




**Check available (By Fraser)** 



**Type of work completed (By Fraser)**

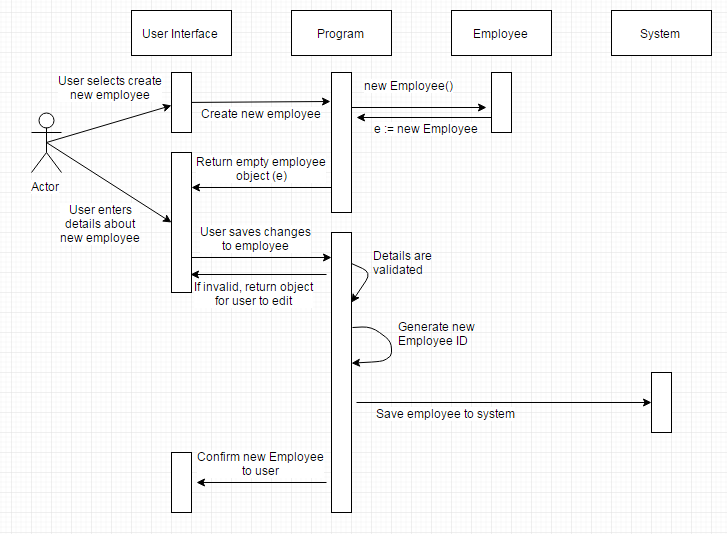




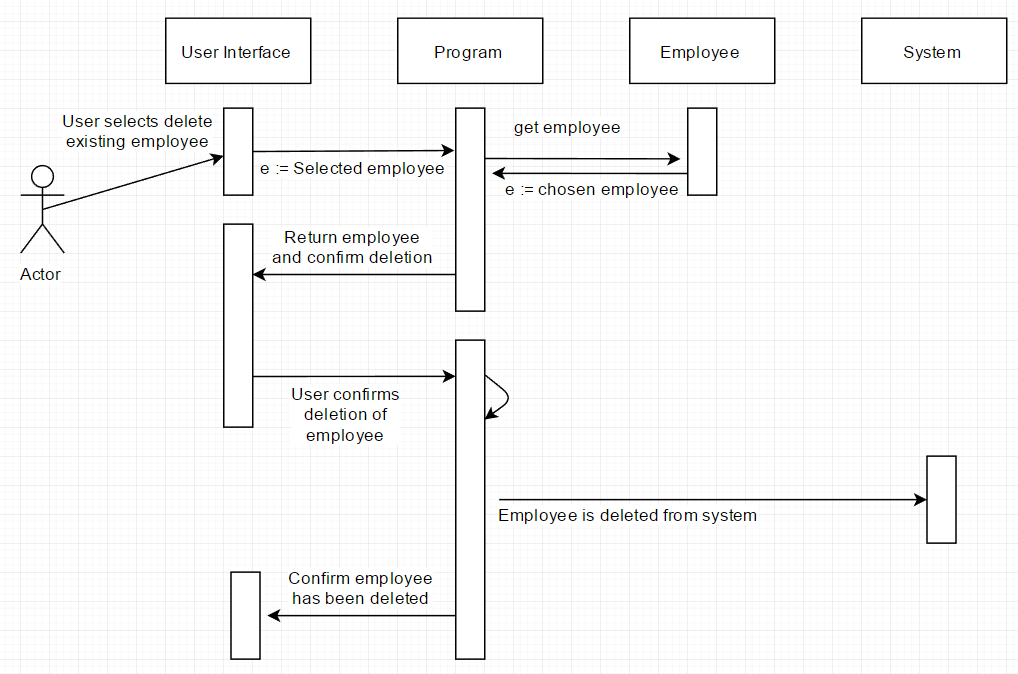
## Sequence diagrams

In here the system is designed in detail, classes interactions are shown in the system in the form of methods. Once completed we can fill in the class diagram and implementation can begin.

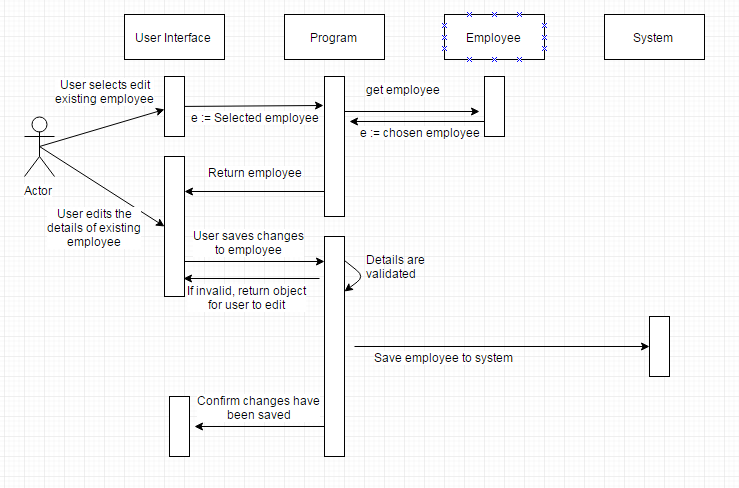
**Create Employee (By Dean)**



**Delete Employee (By Dean)**



**Edit Employee (By Dean)**



**Login (By Mahtarr)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Login.png

**Order New Stock (By Mahtarr)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\order_new_stock.png

**Check Stock (By Mahtarr)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\checking_stock.png

**Create Supplier (By Dan)C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Create_supplierSeq2.png**

**Delete Supplier (By Dan)C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Delete_supplierSeq.png**

**Edit Supplier (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Edit_supplierSeq2.png

**Create Client (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Create_clientSeq.png

**Edit Client (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Edit_clientSeq.png

**Delete Client (By Dan)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Delete_clientSeq.png

**C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\1.getdelivery.pngGet delivery status (By Fraser)**

**Check available (By Fraser)**

C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2.checkavailableemployees.png

**Type of work completed (By Fraser)** C:\Users\FraserGamingPC\AppData\Local\Microsoft\Windows\INetCache\Content.Word\3.typeofworkcompleted.png

## Class diagrams

This is a diagram that supports view of the evolving system, it shows classes and their relationships. Class diagram holds and processes the systems information.

**Items (By Mahtarr)**

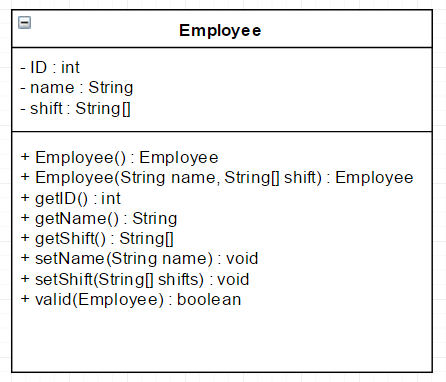
C:\Users\mahtarrjeng\AppData\Local\Microsoft\Windows\INetCache\Content.Word\item class.png

**Supplier (By Denils)**

Supplier.png

**Client (By Denils)**  
Client2.png

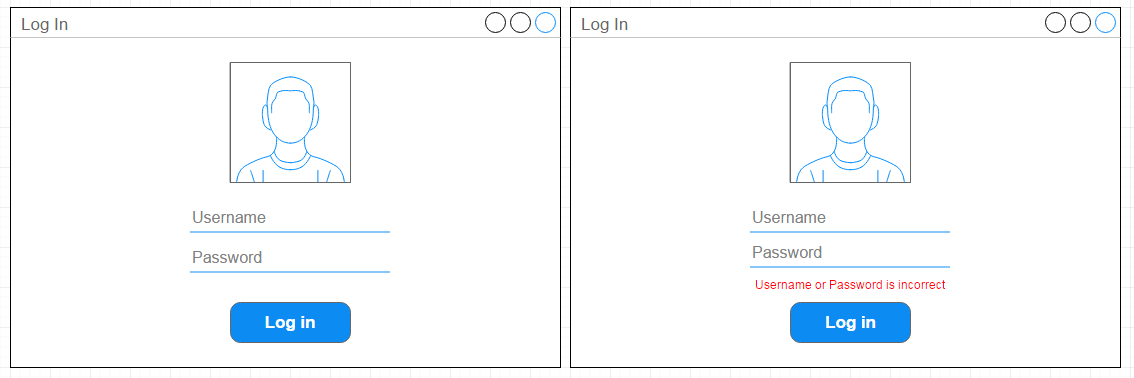
**Employees (By Dean)**



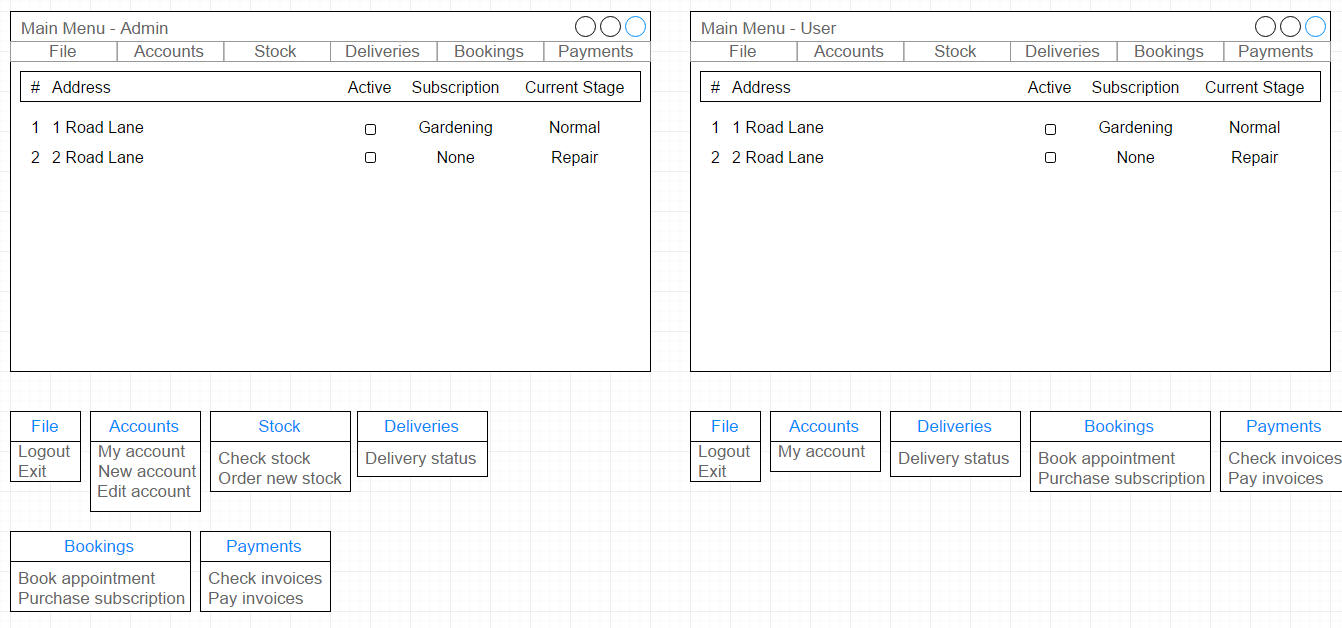
# Phase 3 Design

## Wireframe Mockup

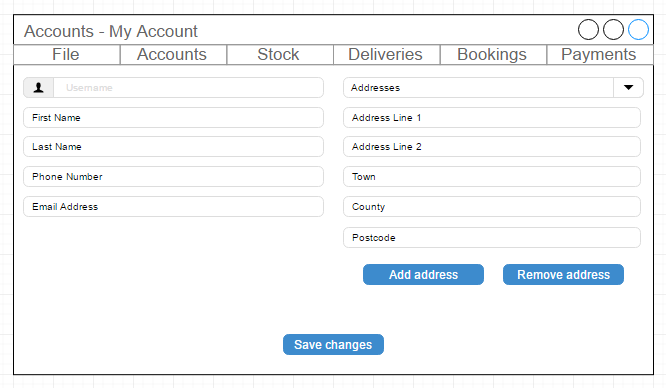
**Login page (By Dean)**



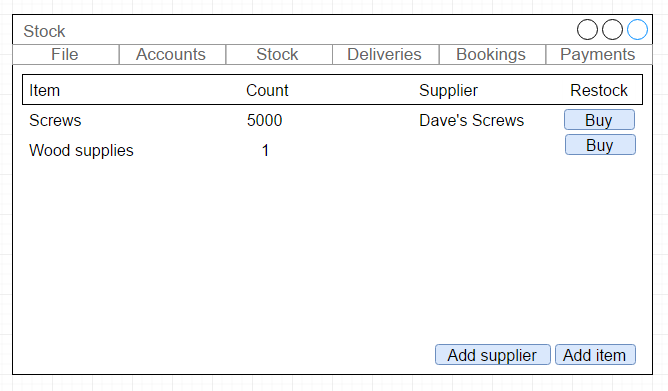
**Main page (By Dean)**



**Accounts page (By Dean)**

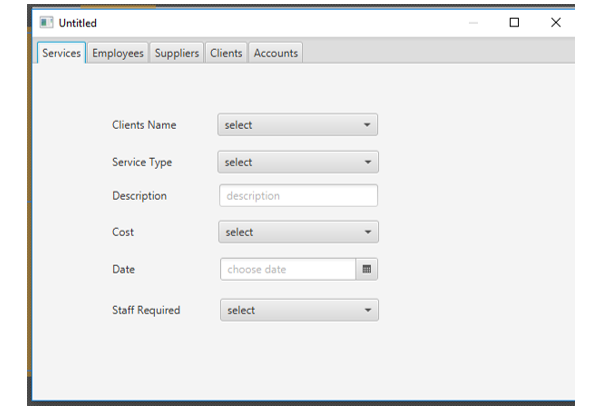


**Stock page (By Dean)**

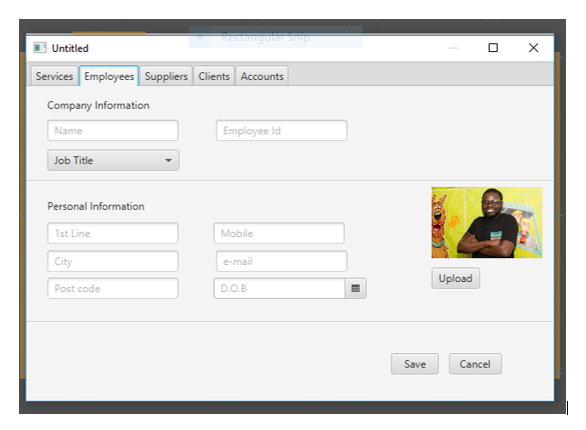


## FXML Mockup

**Service page (By Mahtarr)**



**Employees page (By Mahtarr)**



# Phase 4 Implementation

## GitHub

Github is a development platform that allows version control between various team members.

The Boris Github repository can be found [here](https://github.com/SolentComputing/ESS_Boris).

## Testing

### System Test

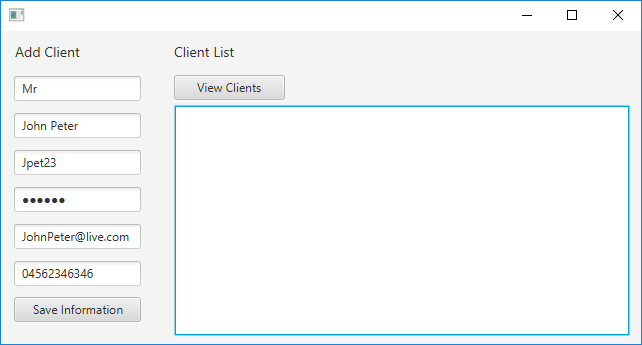
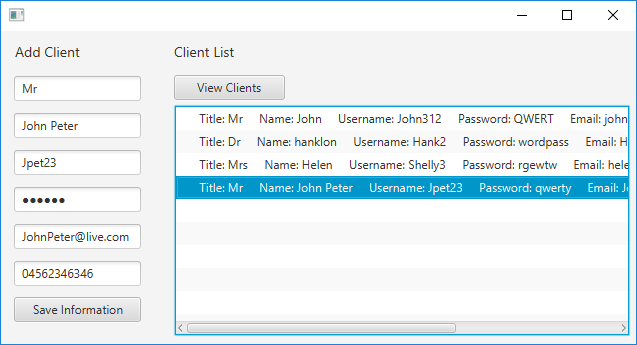
“*System test, it’s also known as the black box. It tests the entire system on error and bugs”*

The diagrams below shows the testing of:

### Client Tab (By Fraser)

Test type: Black Box testing

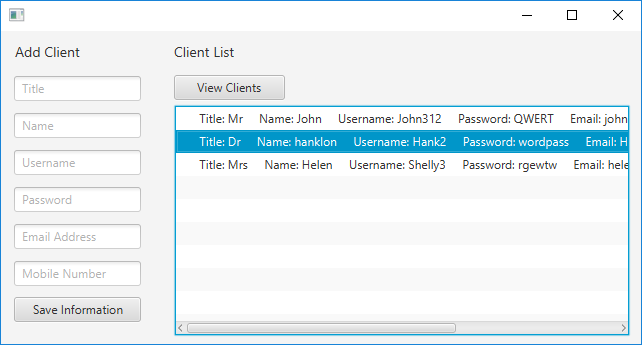
Add Client



When you save the information the saved fields are succesfully added to a string and added the list view. When the view clients button is clicked it successfully refreshes list of clients.

Test Result: Success

Edit Client



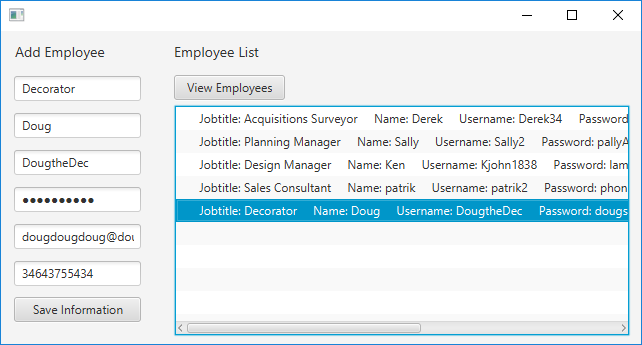
When selecting a client in the list view the user’s information should be opened in the text fields to the left. This does not work and client’s details cannot be edited.

Test Result: Failure

### Employees Tab (By Fraser)

Test type: Black Box testing

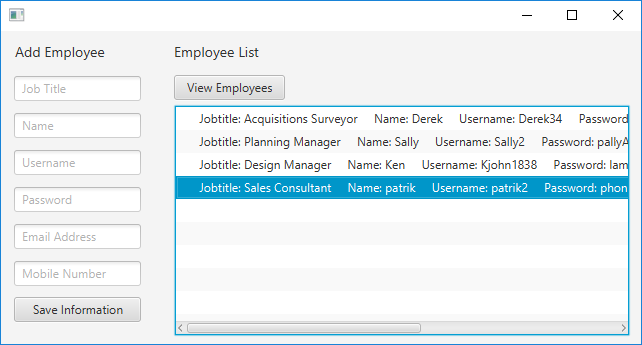
Add Employee



The Employees information is saved and viewable by click the view employees button. The string is successfully made and entered into the list view.

Test Result: Success

Edit Employee



When you select an employee in the list view it should add the clients information into the text fields so you can edit the employee, this does not work.

Test Result: Failure

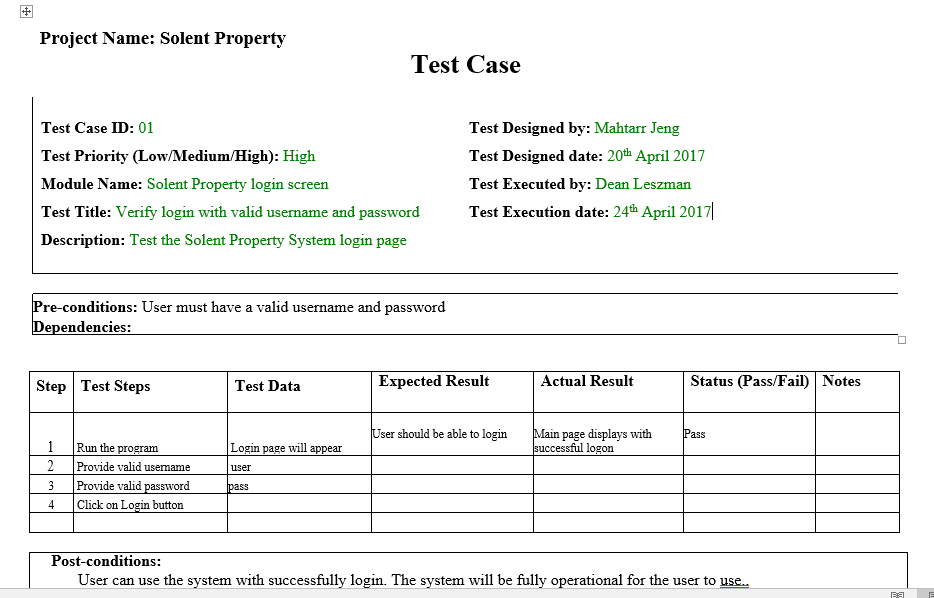
#### ***Unit test***

*“Is the testing of individual software components and it is usually written by the software developer in the test driven environment*.”

### Test case (By Mahtarr)

Serves the purposes of finding errors and bugs in software applications, showing how application should be executed if perform correctly.

#### Login test case



# Phase 5 Maintenance and Support

## Possible changes of this software in the future

At the moment this software is mainly use by Solent Building and Development Services Ltd, but improvements can be made and codes can be redeveloped to allow other users such as clients and suppliers.

This can be done in several ways by having alternative logins for different users. In this case restrictions can be in place within the system and access can be denied if user doesn’t have authority for example, to view confidential data. This will definitely make our system more rigid and it will boost the systems security.

Reduction of technical debt is well considered and applied on this project of this system by commenting codes and making them more readable, this helps a lot when new developers come on board.

## Users guide

### Introduction

This property development and management system is a platform the organization will use for the development, repair and renovations of clients’ properties.

The system effectively improves the management of their services, company-client relationship and the company’s daily activities, it is also designed to help the company to cut cost on any waste possibilities and in return it will maximize company’s profits

### The System (Describing the system)

The key features for this system is to:

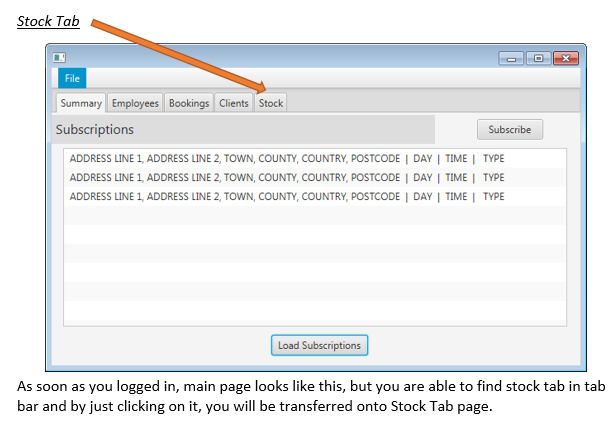
* Allow an administrator or a manager to login into the system using the correct username and password assigned to them.
* Subscribe clients to company’s services, log their requested jobs and select the right staff for the job in demand. Service cost may vary and it depends on the services wanted by the customer and a date will be allocated for work to start.
* Records of all jobs will be on the system with the name of the client and it will illustrate if the work is completed or on going.
* Employees can be added, edited and deleted to the system, and on the employees tab their personal details and skills will be displayed
* Suppliers can be added, edited and deleted on this system and dropdown list of all suppliers used by the company is automatically added on the system for future use.
* List of items ordered by the company from suppliers will also be on the system for reuse, i.e. (raw materials).

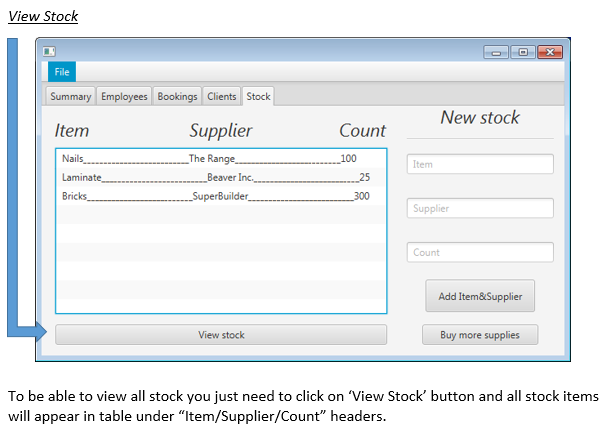
### Installing, Starting and Stopping the System

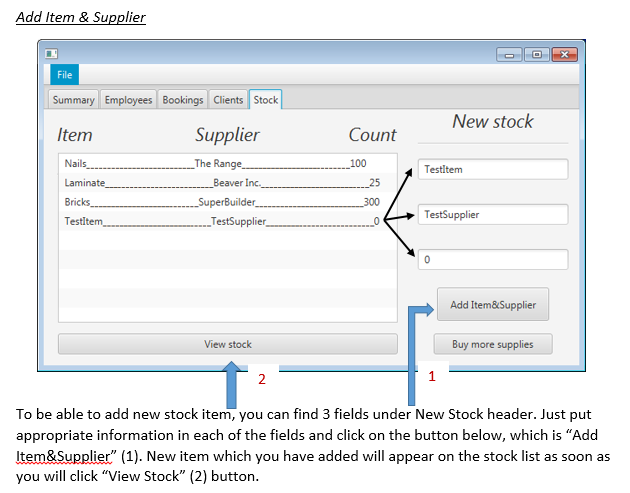
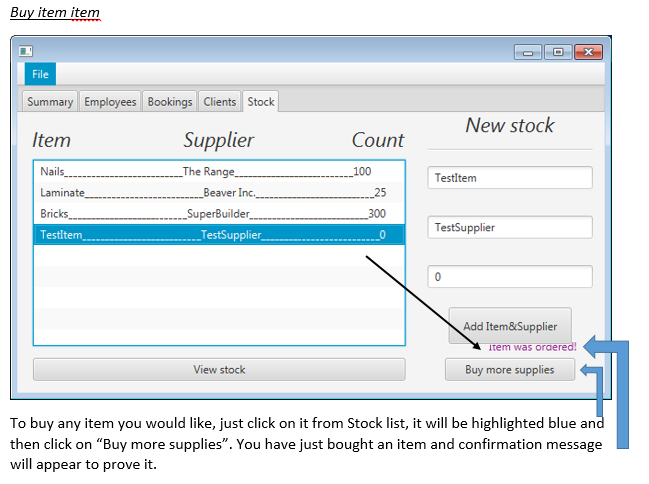
Process of installing, starting, stopping and suspending the system. This will take in consideration the first time users and expert users.

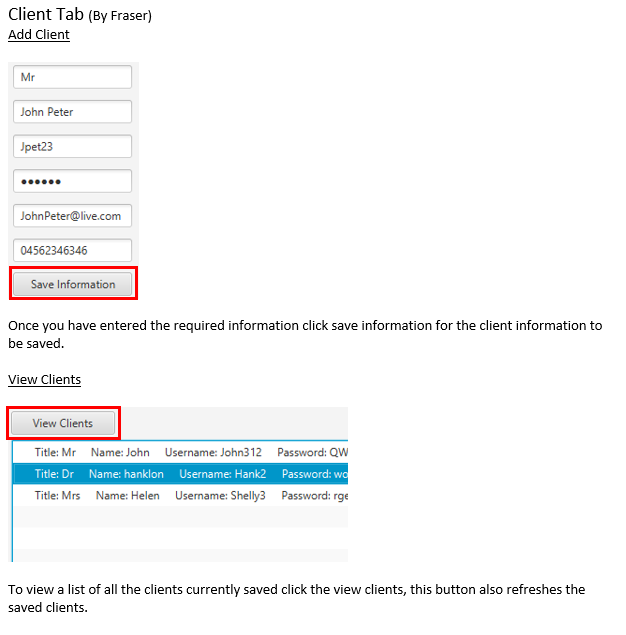
### Step by Step Instructions for using the system

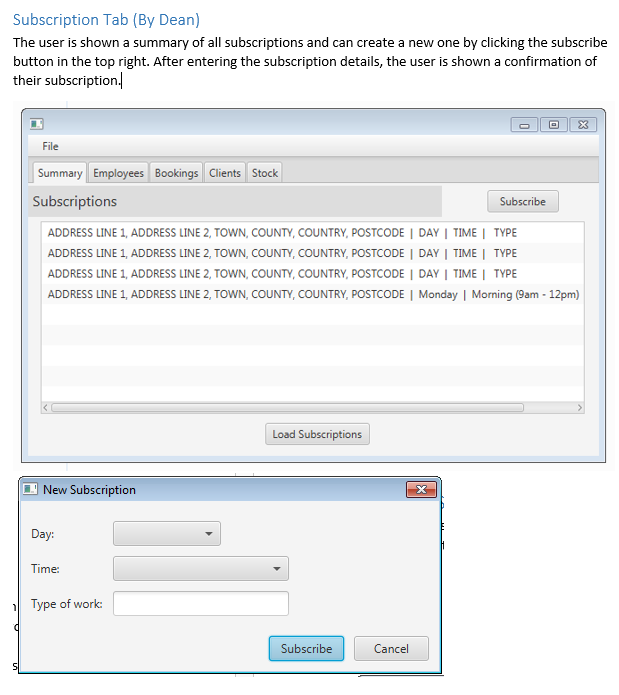
This include the instructions on how to use the system. In addition are the conventions, errors, malfunctions, emergencies and messages.

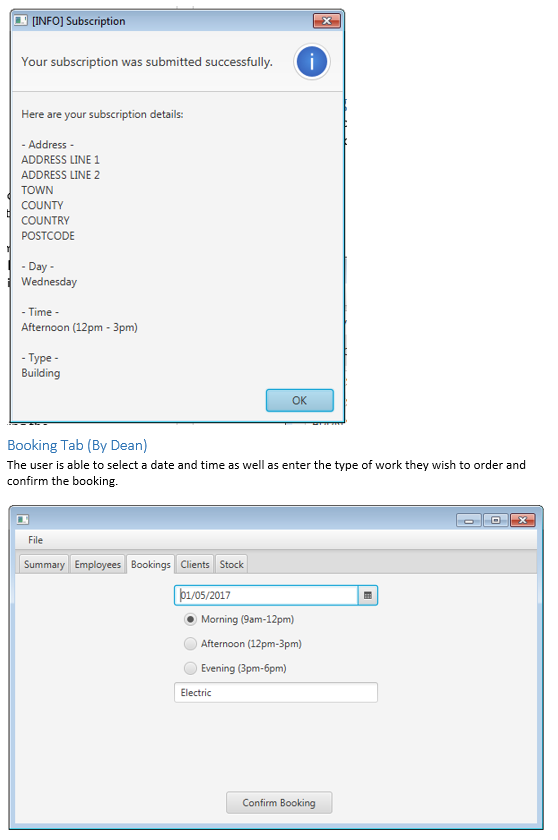










### References

Engineering Software systems case scenario 5

Trello board