**Systems Analysis and Design Project**

**Mobile Phone Retailer and Distributor**



Jack Morrison 13127853

Colm O’Brien

William O’ Dwyer

Contents

[Project Plan and Allocated Roles 4](#_Toc445236001)

[Requirements 5](#_Toc445236002)

[Use Case Diagrams 5](#_Toc445236003)

[Customer Main Activities 5](#_Toc445236004)

[Administrator Main Activities 6](#_Toc445236005)

[Use Case Structured Descriptions 7](#_Toc445236006)

[Login 7](#_Toc445236007)

[Create Account 7](#_Toc445236008)

[Browse Catalogue 7](#_Toc445236009)

[Search Catalogue 7](#_Toc445236010)

[Add Product to Cart 8](#_Toc445236011)

[View Cart 8](#_Toc445236012)

[Remove Product from Cart 8](#_Toc445236013)

[Edit Product Quantity in Cart 8](#_Toc445236014)

[Add Product 9](#_Toc445236015)

[Remove Product 9](#_Toc445236016)

[Edit Product 9](#_Toc445236017)

[View Orders 9](#_Toc445236018)

[Change Customer Category 9](#_Toc445236019)

[Remove User 10](#_Toc445236020)

[Return Stock 10](#_Toc445236021)

[Ship Order 10](#_Toc445236022)

[Detailed Use Case Description 11](#_Toc445236023)

[Create Order 11](#_Toc445236024)

# Software Life Cycle

When deciding what software life cycle to use for our project we looked at three different life cycle models including Waterfall Life Cycle, V-Model and Agile Software Development.

**Waterfall Life Cycle**

The waterfall model is seen as a sequential design process that goes through different stages. It adheres to the principle that work cannot begin on the next stage until the previous stage has been completed. This can lead to difficulties where requirements have a risk of changing during the project. This is one of the reasons that we decided not to use this life cycle as it reduces the ability to adapt to change in requirements which would lead to high risk and cost if needed.

The model also follows a ‘big bang’ approach where integration is done at the very end of the cycle which doesn’t allow for identifying any technical or business issues that may have been apparent had integration been started earlier. As this would also result in a high risk and cost associative we decided to look at another software life cycle, the V-Model.

**V-Model**

The V-model process is similar to the waterfall life cycle in that it is a sequential design process; each stage must be completed before moving onto the next. The V-model is different to the waterfall cycle in that testing of the product is planned in parallel with the corresponding phase of development. This eliminates the ‘big bang’ issue that we had with the waterfall life cycle and would make the V-model more appealing for our project as some testing would have been completed before integration.

However the V-model is still a very rigid and non-flexible design process where it is difficult to implement changes to requirements. Once again we thought that this would not be suitable for us as we would like a more flexible design model that would allow us to easily adapt to change of requirements. We then looked at the Agile Software Development model.

**Agile Software Development Model**

The agile methodology is an alternative to models such as waterfall and v-model. It helps teams respond to unpredicted changes through incremental, iterative phases known as sprints. At the end of these sprints the team presents a potentially shippable product increment. This allows the team to continuously re-visit the product throughout the lifecycle and adjust to change if needed. As these sprints are usually short it allows time for the project to be steered in another direction. This model is appealing to our project as it would allow us to respond to change easily whilst continuously delivering a ‘version’ of the final product.

As online retailers similar to ours can very quickly become quite large and popular it is essential that we are able to respond to change in a quick and efficient way. For this reason we have decided to settle on the agile software development model.

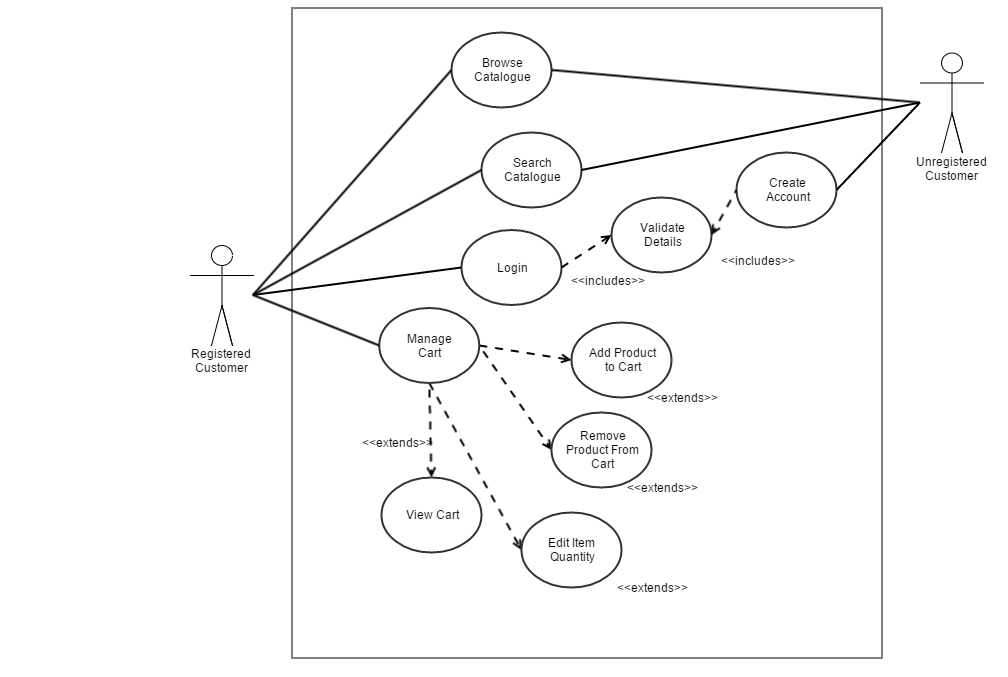
# Project Plan and Allocated Roles

|  |  |  |  |
| --- | --- | --- | --- |
| **Job** | **Job Description** | **Allocated To** | **Week** |
| Narrative | Narrative description of business scenarios | Jack | 6 |
| Presentation | Company Logo / Design Cover Page | Group | 10 |
| Software Life Cycle | Discussion of the software model used | Jack | 6 |
| Project Plan | Specifying jobs and roles | Jack | 5 |
| Requirements | Use Case Diagrams  Use Case Descriptions  Structured Use Case Description  Non-Functional Requirements  Tactics for handling Quality Attributes  Screenshots of GUI | Jack  Jack  Jack  Colm  Colm | 6  6  6  6  6  8 |
| System Architecture | Architecture Diagram with Interfaces | Group | 7 |
| Analysis Sketches | Identify Candidate Classes  Class Diagrams  Communication Diagrams  Entity Relationship Diagrams | William  William  Colm  Jack | 8  8  8  8 |
| Code | Code Implementation | Group | 7 to 12 |
| Design | Architectural Diagrams  Class Diagrams  Interaction Diagram  State Chart  Description of Patterns  Approach to Concurrency Support | Colm  Colm  William  William  Colm  Colm | 9  10  9  10  9  10 |
| Critique | Critique on quality of Analysis and Design | Group | 12 |
| References | Sources used for learning/information | Group | 12 |

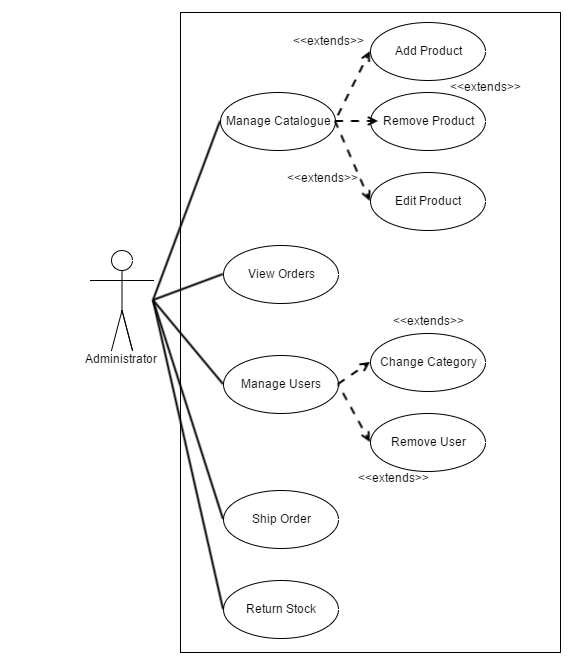
# Requirements

# Use Case Diagrams

## Customer Main Activities

****

## Administrator Main Activities



# Use Case Structured Descriptions

## Login

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects Login from initial screen. | 1. Login form is displayed. |
| 1. Customer enters Username and password and clicks ‘Login’ button. | 1. Login details entered correctly, customer is logged in. |
| **ALTERNATIVE ROUTE** | |
| 3a. Customer enters wrong login details. | 4a. Login details are incorrect, error message with ‘Unable to Login’ is shown. |

## Create Account

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘Create Account’ from initial screen. | 1. Register form is displayed. |
| 1. Customer enters Username, email address and desired password and clicks ‘Login’ button. | 1. System checks if desired username or email address is already associated with an existing account, they are available therefore an ‘Account created successfully.’ message is shown. |
| **ALTERNATIVE ROUTE** | |
| 3a. Desired username is already taken. | 4a. Customer is unable to create account as desired username or email address is already taken, error message with ‘Username already taken, please choose another.’ is shown. |

## Browse Catalogue

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘Browse Catalogue’ from initial screen. | 1. Catalogue is displayed. |
| 1. Customer clicks on a product in the catalogue. | 1. Product info page is shown. |

## Search Catalogue

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘Search Catalogue’ from initial screen. | 1. Search Catalogue screen is displayed. |
| 1. Customer enters search query in text box. | 1. Search results are shown. |
| **ALTERNATIVE ROUTE** | |
| 3a. Search results not found. | 4a. Message ‘No results found.’ shown. |

Non-Functional Requirements:

* Search should not take longer than 5 seconds.

## Add Product to Cart

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘Add Product to Cart’ on Product Info screen. | 1. Product is added to cart successfully with message ‘Product added to cart’ shown. |
| **ALTERNATIVE ROUTE** | |
| 3a. Product is out of stock. | 4a. Message ‘Sorry, this product is not in stock.’ is shown. Product is not added to cart. |
| **ALTERNATIVE ROUTE** | |
| 3b. Cart already contains selected product. | 4b. Product in cart is updated with desired quantity. |

## View Cart

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘View Cart’ on initial screen. | 1. Cart and its contents are shown on screen. |
| **ALTERNATIVE ROUTE** | |
| 3a. Cart is empty. | 4a. Message ‘Your cart is empty.’ is shown. |

## Remove Product from Cart

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘Remove product from Cart’ on View Cart screen. | 1. Product is removed from cart successfully with message ‘Product removed from cart’ shown. |

## Edit Product Quantity in Cart

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Customer selects ‘Edit Product Quantity’ on View Cart screen. | 1. Product quantity is updated successfully and new quantity is shown on View Cart Screen. |
| **ALTERNATIVE ROUTE** | |
| 3a. There is not enough stock to fulfil quantity change. | 4a. Error message ‘Sorry there are not enough products in stock’ is shown. |

## Add Product

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘Add Product’ on the administrator screen. | 1. ‘Add Product’ page is shown. |
| 1. Administrator enters Product name, price and stock. | 1. System checks if product ID already exists, it doesn’t therefore the product is added successfully. |
| **ALTERNATIVE ROUTE** | |
| 3a. Administrator enters Product name, price and stock. | 1. Product ID already exists in the system, its details are updated with the entered values. |

## Remove Product

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘Remove Product’ on the administrator screen. | 1. List of products currently in catalogue is shown. |
| 1. Administrator selects product to remove. | 1. Product is removed successfully, catalogue is updated successfully. |

## Edit Product

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘Edit Product’ on the administrator screen. | 1. List of products currently in catalogue is shown. |
| 1. Administrator selects product to edit. | 1. Product edit page is shown. |
| 1. Administrator edits product. | 1. Product is updated successfully. |

## View Orders

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘View Orders’ on the administrator screen. | 1. List of all Orders is shown. |

## Change Customer Category

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘View Customers’ on administrator screen. | 1. List of all customers is shown. |
| 1. Administrator selects customer to update. | 1. Customer details screen is shown. |
| 1. Administrator selects change customer category | 1. Customer category is updated correctly. |

## Remove User

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘View Customers’ on administrator screen. | 1. List of all customers is shown. |
| 1. Administrator selects customer to remove. | 1. Customer details screen is shown. |
| 1. Administrator selects ‘Remove Customer’ | 1. Message ‘Are you sure you would like to remove customer?’ is shown. |
| 1. Administrator selects ‘Yes’. | 1. Customer is removed and customer list is updated successfully. |
| **ALTERNATIVE ROUTE** | |
| 7a. Administrator selects ‘No’. | 8a. Customer is not removed from system and the administrator is brought back to customer details screen |

## Return Stock

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘Return Stock’ on administrator screen. | 1. Return Stock screen is shown. |
| 1. Administrator selects supplier from drop-down to return stock to. | 1. Supplier is selected successfully. |
| 1. Administrator selects product to return to supplier. | 1. Product is selected successfully; available stock to return is shown. |
| 1. Administrator selects quantity of stock to return, selects ‘Return Stock’. | 1. Message ‘Stock returned successfully’ is shown. |
| **ALTERNATIVE ROUTE** | |
| 7a. Administrator selects ‘No’. | 8a. Customer is not removed from system and the administrator is brought back to customer details screen |

## Ship Order

|  |  |
| --- | --- |
| **ACTOR’S ACTION** | **SYSTEM RESPONSE** |
| 1. Administrator selects ‘View Orders’ on the administrator screen. | 1. List of all Orders is shown. |
| 1. Administrator selects Order to Ship. | 1. Order details screen is shown. |
| 1. Administrator selects ‘Ship Order’. | 1. Message ‘Order shipped successfully’ is shown. |
| **ALTERNATIVE ROUTE** | |
| 1. Product on order is not in stock. | 1. Message ‘Product is not in stock, order cannot be shipped’ is shown. |

# Detailed Use Case Description

## Create Order

|  |  |  |
| --- | --- | --- |
| **USE CASE: CREATE ORDER** | | |
| **DESCRIPTION:** | | User creates an order for their desired products. |
| **PRE-CONDITIONS:** | | * User must be logged in. * User must have products in their cart. * Desired products must be in stock. |
| **SUCCESS END CONDITION:** | | A new order is created successfully. |
| **FAILED END CONDITION:** | | Order is not created. |
| **ACTORS:** | | Customer. |
| **TRIGGER:** | | User clicks ‘Confirm Order’. |
| **FLOW** | | |
| **DESCRIPTION** | **STEP NUMBER** | **ACTION** |
|  | **1** | ‘Create Order’ procedure called by the user through the site GUI. |
|  | **2** | User permissions and credentials are verified by the system. |
|  | **3** | Order is created and order information is saved. |
| **EXTENSION** | **STEP NUMBER** | **BRANCHING ACTION** |
|  |  |  |
| **VARIATION** | **STEP NUMBER** | **BRANCHING ACTION** |
|  | **1** |  |
|  | **2** |  |

# Identifying Candidate Classes

The group has decided upon a Data Driven Design approach to list any possible objects to implement.

**List of Candidate Objects**

To find appropriate candidate objects, the noun identification technique has been applied to the described use cases. Simple heuristics have been used to identify potential objects.

**From User Use Cases:**

|  |  |  |
| --- | --- | --- |
| **Potential Class** | **Attributes** | **Event** |
| User | User Name | Login |
| Catalogue | Password | Browse |
| Cart | Email | Search |
| Product | Category | Validate |
| Orders |  | Create |
| Stock |  | Manage |
|  |  | View |
|  |  | Remove |
|  |  | Add |
|  |  | Ship |
|  |  | Return |

Based on the above we have decided upon these candidate classes.

Customer/User, Product, Cart, Catalogue, Orders and Stock.

**Customer/User**

A customer or user is a user of the system outside of the company. This would represent one individual creating an order over the internet.

**Product**

A product is a phone or accessory that the company has in stock and that a customer can add to an order. Products are ordered from suppliers and put into stock.

**Cart**

The cart contains a list of products and their quantities for a potential order. The cart can be viewed and edited by a customer and contents can be saved for future use.

**Catalogue**

The catalogue is the list of all products that the system makes available to a customer. This displays product details and whether the product is in stock and available to order or not.

**Orders**

An order contains a list of all products and their quantities and also whether it has shipped or not. This can be viewed by the customer and is stored for future reference.

**Stock**

Stock is a list of all products available for immediate shipping. This stores a list of the current quantity of each product and the supplier the product is ordered from.

# Non-Functional Requirements

**Security:**

Our system needs to adhere to a certain level of security when dealing with customers log in credentials (e-mail and passwords) and their back details. The system must be protected from any potential hacking/security threats and allows the user to use our system with complete trust and confidence.

**Performance:**

Response times such as application loading, screen open and refresh times must be very minimal and fluid. Processing times should also be fast as well and Query and Reporting times. Noticeable dips in performance may cause users to be frustrated with the system and in turn may cause them to go else-where.

**Capacity:**

The system should never be nearing its capacity limits; the system should be able to deal with hundreds and thousands of transactions per minute and also must never reach its storage limits.

**Availability:**

As an online application, the system should be available 24 hours a day, 7 days a week. The system does not conform to traditional 9-5 business hours.

**Reliability:**

Users should be able to rely on the system, and not be worried about the site crashing in the middle of a transaction and being charge more than once. If the system does crash, the mean time to recovery should be kept at a minimum.

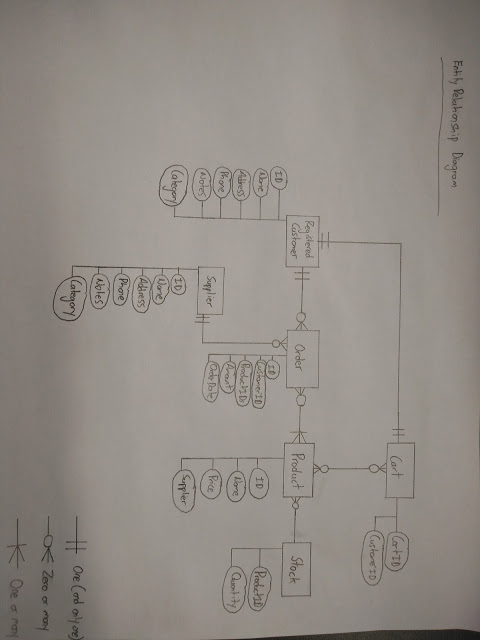
**Maintainability:**

Our system must be easily maintained. We should be able to update a small portion of the system without the change propagating throughout the entire system. We should be able to change the user interface without affecting functionality. The code must be easily understood to new members of the development team (low cost of comprehension).

**Usability:**

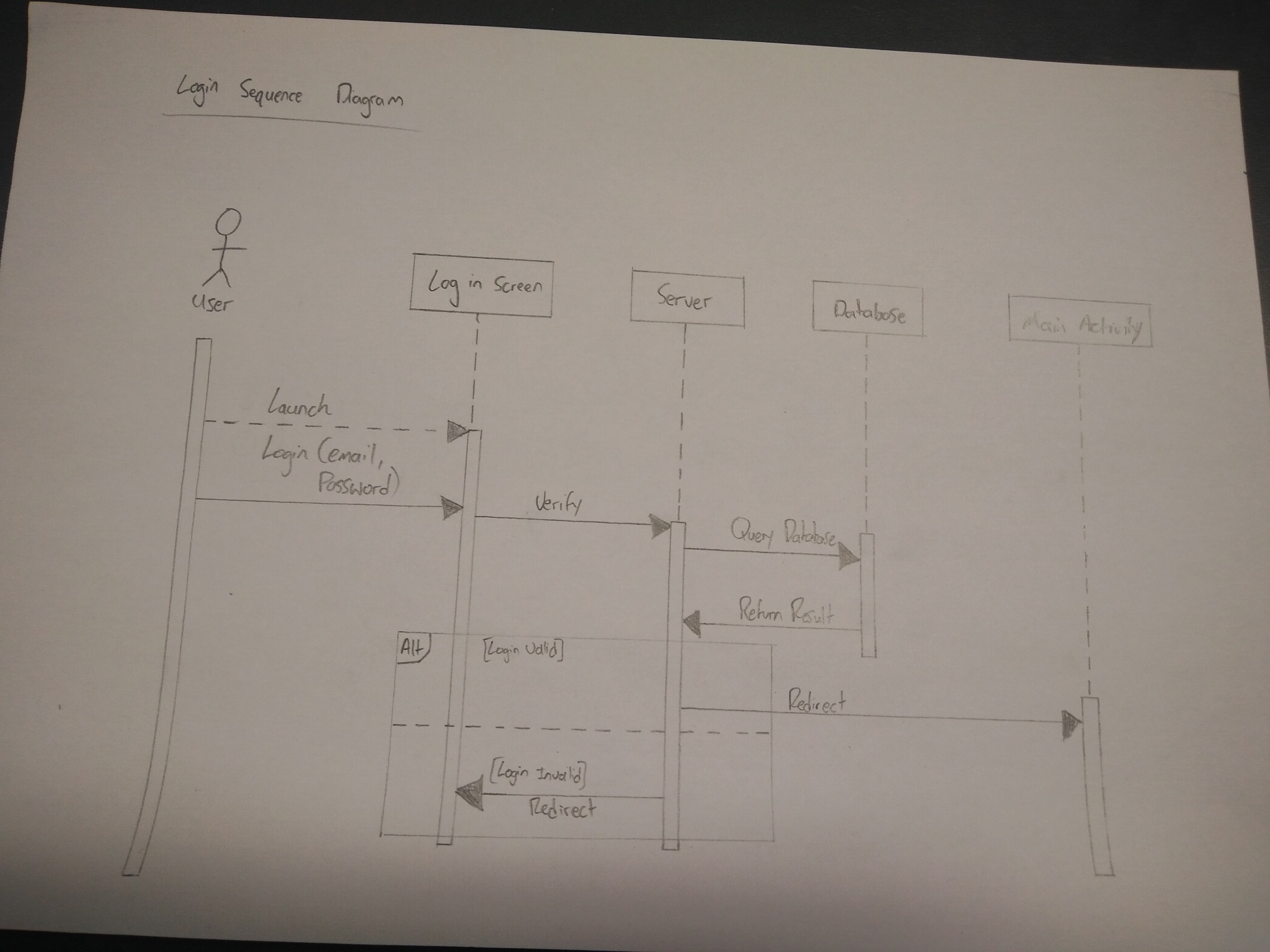
The look and feel of the system must be aesthetically pleasing; it should be an enjoyable experience to use. The layout of the UI should be well spaced out and the UI elements should not be cramped together. The system should localized so people of various nationalities and languages can use the system.

# Entity Relationship Diagram



# Sequence Diagrams

## Login Sequence Diagram



## Create Order Sequence Diagram

