****

**COMSATS University Islamabad (CUI)**

**Project Name**

***By***

**Student Name 1 CIIT/SP09-BSE-xxx/ISB**

**Student Name 2 CIIT/SP09-BSE-xxx/ISB**

***Supervisor*Supervisor Name**

***Co-Supervisor*C0-Supervisor Name**

***Bachelor of Science in Software Engineering (20xx-20xx)***

**The candidate confirms that the work submitted is their own and appropriate  
 credit has been given where reference has been made to the work of others**.

****

**COMSATS University Islamabad (CUI)**

**CLAUNDERETE**

**A project presented to**

**COMSATS University Islamabad**

**In partial fulfillment**

**of the requirement for the degree of**

***Bachelor of Science in Software Engineering (20xx-20xx)***

**By**

**Student Name 1 CIIT/SP09-BSE-xxx/ISB**

**Student Name 2 CIIT/SP09-BSE-xxx/ISB**

**DECLARATION**

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other. We will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

|  |  |  |
| --- | --- | --- |
| Student Name1 | Student Name2 | Student Name3 |
| --------------------------- | --------------------------- | --------------------------- |

**CERTIFICATE OF APPROVAL**

It is to certify that the final year project of BS (CS) “Project title” was developed by   
**STUDENT 1 NAME (CIIT/FAXX-BSE-000)** and **STUDENT 2 NAME (CIIT/FAXX-BSE-000)** under the supervision of “SUPERVISOR NAME” and co supervisor “CO-SUPERVISOR NAME” and that in (their/his/her) opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

---------------------------------------

**Supervisor**

---------------------------------------

**Co-Supervisor**

---------------------------------------

**External Examiner**

---------------------------------------

**Head of Department**

**(Department of Computer Science)**

**Executive Summary**

In public places, there is often a need for monitoring people and different activities going on, which can be referred later for many reasons including security. Appointing humans for this task involves many problems such as increased employee hiring, accuracy problem, trust, no proof for later use, and also the fact that a human can remember things till a certain time limit. Talking about the current security system, they use dumb still cameras with a continuous recording facility irrespective of the fact that any event may happen or not. Moreover, they are usually pointing at a specific user defined location so more than one cameras are required to cover the entire region.

To prevent all these problems from prevailing, the CSCS is developed. It is a surveillance system, which provides solution to many of these problems. It is a stand-alone application which doesn’t require any computer to operate. It monitors different situations using a camera which is able to rotate intelligently based on sensor messages and captures the scene in the form of video or photos later reference as well.

**C**ustomizable **S**urveillance **C**ontrol **S**ystem **(CSCS)** is a surveillance system that can be assigned a sensor type as in our case a heat sensor is used, it works accordingly, rotates the camera upon event detection and perform user defined actions like capturing video and stores them, for the future use.

It is an embedded system consisting of Linux fox kit with embedded a running server application also a camera, USB storage device and a sensor node base station is attached with fox kit. LAN communication is used by user to download the videos and to operate the system manually.

**Acknowledgement**

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project supervisor “Dr. Majid Iqbal Khan” and our Co-Supervisor “Mr. Mukhtar Azeem”. Without their personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are deeply indebted to them for their encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us the values of honesty & hard work.

|  |  |  |
| --- | --- | --- |
| Student Name1 | Student Name2 | Student Name3 |
| --------------------------- | --------------------------- | --------------------------- |

**Abbreviations**

|  |  |
| --- | --- |
| **SRS** | Software Requirement Specification |
| **PC** | Personal Computer |
|  |  |
|  |  |
|  |  |

**Table of Contents**

[**1 Introduction** 1](#_Toc520773734)

[1.1 Brief Overview 1](#_Toc520773735)

[1.2 Relevance to Course Modules 1](#_Toc520773736)

[1.3 Project Background 1](#_Toc520773737)

[1.4 Literature Review 1](#_Toc520773738)

[1.5 Analysis from Literature Review 1](#_Toc520773739)

[1.6 Methodology and Software Lifecycle for this Project 1](#_Toc520773740)

[1.6.1 Rationale behind the Selected Methodology 1](#_Toc520773741)

[**2** **Problem Definition** 2](#_Toc520773742)

[2.1 Problem Statement 2](#_Toc520773745)

[2.2 Deliverables and Development Requirements 2](#_Toc520773746)

[2.3 Current Systems 2](#_Toc520773747)

[**3** **Requirement Analysis** 4](#_Toc520773748)

[3.1 Use Cases Diagram(s) 4](#_Toc520773750)

[3.2 Detailed Use Cases 4](#_Toc520773751)

[3.3 Functional Requirements 4](#_Toc520773752)

[3.4 Non-Functional Requirements 4](#_Toc520773753)

[**4 Design and Architecture** 5](#_Toc520773754)

[4.1 System Architecture 5](#_Toc520773759)

[4.2 Data Representation 5](#_Toc520773760)

[4.3 Process Flow/Representation 5](#_Toc520773761)

[4.4 Design Models 5](#_Toc520773762)

[**5** **Implementation** 7](#_Toc520773763)

[5.1 Algorithm 7](#_Toc520773765)

[5.2 External APIs 7](#_Toc520773766)

[5.3 User Interface 7](#_Toc520773767)

[**6 Testing and Evaluation** 8](#_Toc520773768)

[6.1 Manual Testing 8](#_Toc520773771)

[6.1.1 Unit testing 8](#_Toc520773772)

[6.1.2 Integration Testing 9](#_Toc520773773)

[6.1.3 Functional Testing 10](#_Toc520773774)

[6.1.4 System Testing 10](#_Toc520773775)

[6.2 Automated Testing 10](#_Toc520773776)

[**7** **Conclusion and Future Work** 12](#_Toc520773778)

[7.1 Conclusion 12](#_Toc520773780)

[7.2 Future Work 12](#_Toc520773781)

[**8** **References** 13](#_Toc520773782)

**List of Figures**

Fig 1.1 Block Diagram 1

Fig 2.1 Use Case Diagram 2

# 

# Introduction

People are often so busy in their daily life routine that they do not have enough time to get their laundry done as it is time so much time consuming. So, they want to get their laundry done in an affordable and convenient way and with no wastage of time. That is where the need of automation arises to provide the people with such laundry services.

An interactive platform for an online laundry cleaning system providing people with services such as washing, ironing and dry cleaning. This system has three main parts that is the client, Launderer and the admin that manages both the client and launderer. We have studied some systems that were either for android user or web users so we will be developing both web and android application and services will be provided. Client can place their appointment of their laundry pick-up and then delivered at their doorstep. Some people are too busy and sometimes they need their laundry to be done urgently so this service can also be provided.

The prime modules of the proposed system are as follows:

1. Profile Management.
2. Booking Localization.
3. Launderer Management.
4. Review Management.
5. Notification and Order Management.
6. Admin Customer Management.
   * + Admin can review the membership of new Launderer.
     + Admin will face a record for day-to-day sale.
     + Admin can review the complaints.
     + Admin can respond to those complaints and mark them as resolved/in-process.
     + Admin can block or unblock the client/ launderer.
     + Admin can look at the details of client/launderer.
7. Payment System.

## Brief Overview

An interactive platform for an online laundry cleaning system providing people with services such as washing, ironing and dry cleaning. This system has three main parts that is the client, Launderer and the admin that manages both the client and launderer. We have studied some systems that were either for android user or web users so we will be developing both web and android application and services will be provided. Client can place their appointment of their laundry pick-up and then delivered at their doorstep. Some people are too busy and sometimes they need their laundry to be done urgently so this service can also be provided.

## Relevance to Course Modules

Here is the table showing relevance to course modules

|  |  |
| --- | --- |
| **Course** | **Relevance to modules** |
| Web Engineering, Topics in software engineering I | Help in developing Web Application using language like JavaScript, HTML, CSS, Bootstrap. |
| Software Quality Engineering | Helped in ensuring the quality of the system. |
| Software Testing | Helped in testing the system. |
| Software Requirement Engineering | Helped in gathering requirements. |
| Software Design and Architecture | Helped in defining our Architecture. |

## Project Background

Technology has revolutionized almost every aspect of humans. It has the very ability to automate daily routine tasks. Often people do not find time to get their laundry done as they are too busy in their daily life routines and they face many problems in continuing their daily routines. The first thing is their time is so much wasted in doing their laundry. Secondly, many people do not know how wash or take good care of their clothes often resulting in damaging the fabric of their clothes.

## Literature Review

Following are some related existing systems.

• Dhobionline

• Elaundry

## Analysis from Literature Review

The related products are few and good apps are rarer which causes most launderettes to have standalone app or no online services at all. Analyzing those apps and websites have provided us to come up with the idea of submerging all launderettes or laundry shops under the same roof, helping in forging social media of launders. Products which are like our project are dhobionline.pk and elaundry.pk.

Table 1: Related System Analysis with Targeted Project Solution.

|  |  |  |
| --- | --- | --- |
| **Application Name** | **Weakness** | **Proposed Project Solution** |
| * dhobionline.pk * elaundry.pk | * No mobile application. * Limited audience of only Islamabad. * UI is not friendly. * No automation of laundry really occurs. * Limited for only Lahore residents. * Timing of services is specific. * No payment option available. * No Reviews and Feedback Feature | * We will develop mobile app for ease of access. * This service will be accessible for multiple cities of Pakistan. * Very friendly UI. * Providing full automation of laundry services. * Accessible for multiple cities. * This service will be provided 24/7 * Payment solution is available. * Product will use reviews and feedback features. |

## Methodology and Software Lifecycle for this Project

• **Design Methodology**

We will use function-oriented design, which inherits some properties from Structured Design, to develop our system. Reason behind selecting this method is because in function-oriented design, the system is comprised of many smaller sub-systems known as functions. This design mechanism divides the whole system into smaller functions, which provides means of abstraction by hiding the information and their operation. These functional modules can share information among themselves by means of information passing and using information available globally.

• **Process Methodology**

We will use Iterative and Incremental process model for our system, Main reason is that it breaks the system functionality into increments. We can evaluate system according to requirements after eachiteration.

### Rationale behind the Selected Methodology

Incremental Methodology is selected for this project due to its resonation with the nature of the work. In which the model is designed, implemented, and tested incrementally till product is finished. Changes are highly probable which might cause us to change or revise some modules in future. The project is flexible which resonates with the methodology and in the incremental approach, backtracking is easy.

# Problem Definition

This chapter would discuss the targeted problem in a precise manner. It should also need to clearly discuss the outcome of the targeted project.



## Problem Statement

Technology has revolutionized almost every aspect of humans. It has the very ability to automate daily routine tasks. Often people do not find time to get their laundry done as they are too busy in their daily life routines and they face many problems in continuing their daily routines. The first thing is their time is so much wasted in doing their laundry. Secondly, many people do not know how wash or take good care of their clothes often resulting in damaging the fabric of their clothes.

## Deliverables and Development Requirements

### Scope

In the start of the project we had to deliver the SCOPE of our project with the Problem statement and its proposed solution. The advantages the proposed solution would bring and a related system analysis of an existing system.

### SRS

As we moved forward with our project, we had to deliver the SRS (Software Requirement Specification) Document which would include all the FRs (Functional Requirements) and NFRs (Non-Functional Requirements) of the proposed solution along with some implementation of the solution.

### SDD

After delivering the SRS next we had to deliver the SDD (System Design Document) which would include our systems Design Methodology and Software process model. It would also include System Overview, Design Models, Data Design, Algorithm and how it would be implemented. We also had to deliver 40% implementation of the proposed system solution along with this document.

### 70% Implementation

After delivering the SDD of the system we had to show substantial work done on the proposed solution. The implementation of the modules we showed was more than 70%.

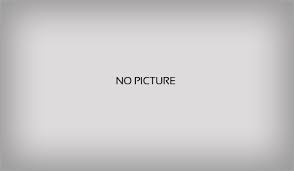
### 100% Implementation

At this stage all the Modules have been completely implemented and all the Functionality of the proposed solution has been achieved

## Current System

A brief description of existing system(s) should be provided in this section. You may add a Table to succinctly discusses the strengths and limitations of current system(s) in comparison with the targeted system.

The following figure is a sample figure, Figure 2.1. You are required to follow the same style of numbering and caption for the whole report. The discussion should be like “ Figure 2.1 presents the…”



**Figure 2.1: Sample Figure**

The following table (Table 2.1) is sample table; You are required to follow the same style of numbering and caption for the whole report.

**Table ‎2.1: Sample Table**

|  |  |  |
| --- | --- | --- |
| **Header 1** | **Header 2** | **Header 3** |
| Text | Text | Text |
|  |  |  |

# Requirement Analysis

The following parts of Software Requirements Specification (SRS) report should be included in this chapter.



## Use Cases Diagram(s)

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

## Detailed Use Case

|  |  |
| --- | --- |
| **Use Case ID:** | UC-1 |
|  |  |
| **Use Case Name:** | LogIn |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client, Launderer,Admin | |
| **Description:** | In order to access all the functionality of the Cloud Launderette user will have to login so they can use the system. |
| **Trigger:** | User clicks on login option |
| **Preconditions:** | PRE-1. User has registered their account. |
| **Postconditions:** | POST-1. User gets redirected to their home page after successful login. |
| **Normal Flow:** | 1. User enters their username/email.  2. User enters their password.  3. User presses login button.  4. Home screen is displayed to user. |
| **Alternative Flows:** | 1. User selects register account. |
| **Exceptions:** | 1. User enters wrong username/email.  2. User enters wrong password.  3. User is not registered. |
| **Business Rules** | BR-1: User must have internet connection. |
| **Assumptions:** | N/A |

|  |  |
| --- | --- |
| **Use Case ID:** | UC-2 |
|  |  |
| **Use Case Name:** | Forgot Password |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client, Launderer,Admin | |
| **Description:** | If user has forgotten the password the can use this use case to retrieve that password |
| **Trigger:** | User clicks on forgot password option |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User must be in UC-1 already. |
| **Postconditions:** | POST-1. User gets redirected to their home page after successful login. |
| **Normal Flow:** | 1. User enters their username/email.  2. A valid account relating to that username/email is shown.  3. User selects that account.  4. Code verification is sent to the relevant email.  5. User enters the verification code.  6. User presses enter button.  7. User is show new password interface.  8. User creates new password. |
| **Alternative Flows:** | 1. User enters invalid username/email. 2. No account found message is shown to the user. |
| **Exceptions:** | 1. User enters nothing in username/email field. |
| **Business Rules** | BR-1: User must have internet connection. |
| **Assumptions:** | N/A |

### UC-3: SignUp

|  |  |
| --- | --- |
| **Use Case ID:** | UC-3 |
| **Use Case Name:** | SignUp |
| **Actors:** | |  |  |  | | --- | --- | --- | | Primary Actor: | Client, Launderer,Admin |  | |
| **Description:** | User can create account if they don’t have already to use to the system |
| **Trigger:** | User indicates that he/she wants to use the system. |
| **Postconditions:** | POST-1. User account gets created.  POST-2. User gets automatically logged in. |
| **Normal Flow:** | 1. User enters their information.  2. User specify payment method.  3. User presses create account button.  3. User account gets created.  5. Home screen is displayed to user. |
| **Exceptions:** | 1. User enters invalid information.  . |
| **Business Rules** | BR-1: User must have stable internet connection. |
| **Assumptions:** | N/A |

### UC-4: View Profile

|  |  |
| --- | --- |
| **Use Case ID:** | UC-4 |
| **Use Case Name:** | View Profile |
| **Actors:** | |  |  |  | | --- | --- | --- | | Primary Actor: | Client, Launderer |  | |
| **Description:** | User wants to checks his profile to see how their bio looks and their details on that have been recorded on the system. |
| **Trigger:** | User click on View Profile Option. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User has internet connection. |
| **Postconditions:** | POST-1. User gets redirected to their or others profile page. |
| **Normal Flow:** | 1. User selects view profile option.  2. Relevant profile detail is show to the user. |
| **Alternative Flows:** | N/A |
| **Exceptions:** | N/A |
| **Business Rules** | BR-1: User must have stable internet connection.  BR-2: User must be logged in. |
| **Assumptions:** | N/A |

### UC-5: Profile Setting

|  |  |
| --- | --- |
| **Use Case ID:** | UC-5 |
| **Use Case Name:** | Profile Setting |
| **Actors:** | |  |  |  | | --- | --- | --- | | Primary Actor: | Client, Launderer, |  | |
| **Description:** | User can change their profile settings like changing the general information, contact information, password and can modify their payment settings. |
| **Trigger:** | User wants to change his profile information. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User has internet connection. |
| **Postconditions:** | POST-1. Success diaglog box is shown.  POST-2. User gets redirected to UC-4. |
| **Normal Flow:** | 1. User select the information he wants.  2. User modify the information he wants to change.  3. User presses save button.  4. User is redirected to CL-3. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 1. After the step 2 of normal flow, user clicks cancel. 2. Confirmation dialogue box appears. 3. User confirms it. 4. The information is not modified. 5. User gets redirected to CL-3 |
| **Exceptions:** | 1. User enters invalid information. 2. User doesn’t have a payment account. |
| **Business Rules** | BR-1: User must have stable internet connection. |
| **Assumptions:** | N/A |

### UC-6: Notifications

|  |  |
| --- | --- |
| **Use Case ID:** | UC-6 |
| **Use Case Name:** | Notifications |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client, Launderer,  Admin | |
| **Description:** | In notifications user can open their notification to check updates, they can mark the notification/notifications as read or clear them. |
| **Trigger:** | User wants to check any updates related to him or related to the system. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User has internet connection. |
| **Postconditions:** | POST-1. User gets redirected to specified page of the notification. |
| **Normal Flow:** | 1. User selects open notifications.  2. User reads the notifications  3. User can check a specific notification. |
| **Alternative Flows:** | 1. After step 2 of Normal Flow, user opens up a notification. 2. User is redirected to the related page. |
| **Exceptions:** | 1. User selects a notification whom reference page has been deleted or is unaccessible. |
| **Business Rules** | BR-1: User must have stable internet connection.  BR-2 Notification must be up to date with system. |
| **Assumptions:** | N/A |

#### UC-7: Mark As Read

|  |  |
| --- | --- |
| **Use Case ID:** | UC-7 |
| **Use Case Name:** | Mark As Read |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client, Launderer,  Admin | |
| **Description:** | When user wants to check single notification after reading/checking mark as read. User can also mark multiple notifications. |
| **Trigger:** | User selects notification and clicks on mark as read. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User is in UC-6 |
| **Postconditions:** | POST-1. Notifications get updated |
| **Normal Flow:** | 1. User selects open notifications.  2. User reads the notifications  3. User can check a specific/multiple notification as read. |
| **Alternative Flows:** | N/A |
| **Exceptions:** | 1. Notification has already been read. |
| **Business Rules** | BR-1: User must have stable internet connection.  BR-2: Notification must be up to date with system. |
| **Assumptions:** | N/A |

#### UC-8: Clear Notifications

|  |  |
| --- | --- |
| **Use Case ID:** | UC-8 |
| **Use Case Name:** | Clear Notifications |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client, Launderer,  Admin | |
| **Description:** | When user wants to clear all the or certain notification. |
| **Trigger:** | User selects notification and clicks clear. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User is in UC-6 |
| **Postconditions:** | POST-1. Notifications get updated |
| **Normal Flow:** | 1. User selects open notifications.  2. User clicks on clear.  3. User can clear a specific/multiple notification. |
| **Alternative Flows:** | N/A |
| **Exceptions:** | 1. Notification has already been read. |
| **Business Rules** | BR-1: User must have stable internet connection.  BR-2: Notification must be up to date with system. |
| **Assumptions:** | N/A |

### UC-9: Check Launderette Reviews

|  |  |
| --- | --- |
| **Use Case ID:** | UC-9 |
| **Use Case Name:** | Check Launderette Reviews |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client | |
| **Description:** | When client want to check reviews of a specific launderette. |
| **Trigger:** | Client clicks on Check Launderette Review option. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User has internet connection. |
| **Normal Flow:** | 1. Client opens up the Launderette.  2. Client checks the reviews.  3. List of reviews of specific Launderette is shown. |
| **Alternative Flows:** | N/A |
| **Exceptions:** | 1. Client is not logged in.  2. Client has no internet connection.  2. User is not registered. |
| **Business Rules** | N/A |
| **Assumptions:** | N/A |

#### UC-10: Make Review

|  |  |
| --- | --- |
| **Use Case ID:** | UC-10 |
| **Use Case Name:** | Make Reviews |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Client | |
| **Description:** | When client wants to give his feedback of recent interaction with specific launderette. Client can edit or delete this review. |
| **Trigger:** | Client clicks on Make Review option. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User has internet connection. |
| **Normal Flow:** | 1. Client opens up the Launderette.  2. Client checks the reviews.  3. Client makes a new review. |
| **Alternative Flows:** | N/A |
| **Exceptions:** | 1. Client review is too big. 2. Client review is empty. |
| **Business Rules** | BR-1: Client must be inside UC-9. |
| **Assumptions:** | N/A |

### UC-11: Find Launderette

|  |  |
| --- | --- |
| **Use Case ID:** | UC-11 |
| **Use Case Name:** | Find Launderer |
| **Actors:** | |  |  |  |  | | --- | --- | --- | --- | | Primary Actor: | Client | Secondary Actors: | Location System | |
| **Description:** | For finding a launderer quickly based on location and furthermore price and rating who can provide laundry service for you. |
| **Trigger:** | Clients wants to book a launderette for their laundry work. |
| **Preconditions:** | PRE-1. Client has registered their account.  PRE-2. Client is logged in.  PRE-3. Client has internet connection.  PRE-4. Client has location turned on. |
| **Postconditions:** | POST-1. Clients sends laundry order to Launderer.  POST-2. Launderer either accepts or rejects the order. |
| **Normal Flow:** | 1. Client choses find launderer. 2. Client is show of list of launderettes close to them. 3. Client selects launderettes based on the price/rating/location. 4. Client selects the services they want to get. 5. Client order is sent to that launderette in UC-8. |
| **Alternative Flows:** | 1. Client cancels the order request. 2. Order request is removed the launderer portal. 3. Order notification is removed from launderer portal. |
| **Exceptions:** | 1. Client does not have location turned on. 2. Client order request doesn’t get sent. 3. Launderer doesn’t receive order request. |
| **Business Rules** | BR-1: Client can access this use case.  BR-2: Client must have location turned on.  BR-3: Client should be logged in.  BR-4: If location is turned off then last know location will be used as address. |
| **Assumptions:** | N/A |

### UC-12: Book Launderette

|  |  |
| --- | --- |
| **Use Case ID:** | UC-12 |
| **Use Case Name:** | Book Launderer |
| **Actors:** | |  |  |  |  | | --- | --- | --- | --- | | Primary Actor: | Launderer | Secondary Actors: | Location System | |
| **Description:** | All the order request will be received in this use case and launderer will be able to accept these request or reject them. If launderer accept theses request he can add time frame for the completion of order and and will quote price for the work. |
| **Trigger:** | Launderer wants to check for order request to work with. |
| **Preconditions:** | PRE-1. Launderer has registered their account.  PRE-2. Launderer is logged in.  PRE-3. Launderer has internet connection.  PRE-4. Launderer has location turned on. |
| **Postconditions:** | POST-1. Order approvel notification is sent to client.  POST-2. Launderer picks up the clothes. |
| **Normal Flow:** | 1. Launderer opens Requests. 2. Launderer is shown request based on time. 3. Launderer accepts a specific request. 4. Launderer sets the time frame for the job. 5. Launderer set price quota for the client. |
| **Alternative Flows:** | 1. Launderer checks the requests. 2. Launderer is busy so they reject the request. |
| **Exceptions:** | 1. Launderer payment account is expired. 2. Launderer location is turned off. 3. Launderer internet connection gets interrupted. 4. Launderer login is expired. |
| **Business Rules** | 1. BR-1: Launderer must be logged in. 2. BR-2: Launderer payment account should be active. 3. BR-3: Launderer should have valid internet connection. |
| **Assumptions:** | N/A |

### UC-13: Payment Method

|  |  |
| --- | --- |
| **Use Case ID:** | UC-13 |
| **Use Case Name:** | Payment Method |
| **Actors:** | |  |  |  | | --- | --- | --- | | Primary Actor: | Client, Launderer | | | Secondary Actors: | | PayPal System, Credit Card System | | |
| **Description:** | In order to get service clients needs to pay and launderer can use payment method to get the payments by clients. |
| **Trigger:** | Client wants to pay through paypal.  Launderer wants to get payment through paypal. |
| **Preconditions:** | PRE-1. User has registered their account.  PRE-2. User is logged in.  PRE-3. User has internet connection.  PRE-4. User has PayPal account. |
| **Postconditions:** | POST-1. Payment method should get updated. |
| **Normal Flow:** | 1. User selects payment method.  2. User selects payment through paypal account.  3. User enters paypal account information.  4. User saves and updates the payment method. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 1. User selects payment method.  2. User selects payment through physical cash.  3. User saves and updates the payment method. |
| **Exceptions:** | 1. User specifies invalid information.  2. PayPal api error. |
| **Business Rules** | 1. BR-1: Payment account must be valid. 2. BR-2: Payment account should be active. |
| **Assumptions:** | N/A |

### UC-14: Admin Dashboard

|  |  |
| --- | --- |
| **Use Case ID:** | UC-14 |
| **Use Case Name:** | Admin Dashboard |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Admin | |
| **Description:** | After the admin has logged in, the admin dashboard will appear. The main elements that will be appear are Launderers, Clients, Complaints, Sales Records and logout. |
| **Trigger:** | The admin wants to use the system. |
| **Preconditions:** | 1. The admin has registered his account. 2. The admin has logged in into the system. |
| **Postconditions:** | N/A |
| **Normal Flow:** | 1. The admin logs in. 2. The admin can clicks on the Launderer. 3. The launderer windows appear displaying it’s information. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 1. The admin logs in. 2. The admin can clicks on the Launderer. 3. The launderer windows appear displaying it’s information. |
| **Exceptions:** | 1. User has already logged out. 2. Server is down. 3. Api error. |
| **Business Rules** | N/A |
| **Assumptions:** | N/A |

#### UC-15: Sales Report

|  |  |
| --- | --- |
| **Use Case ID:** | UC-15 |
| **Use Case Name:** | Sales Report |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Admin | |
| **Description:** | Admin wants to check reports relating to sales. |
| **Trigger:** | Admin clicks on sales report. |
| **Preconditions:** | 1. The admin has logged in into the system. |
| **Postconditions:** | N/A |
| **Normal Flow:** | 1. The admin logs in. 2. The admin can clicks on the Sales Report. 3. Sales record based on current date is shown. 4. Admin can change this date to any previous date and sort on that. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | N/A |
| **Exceptions:** | 1. Admin has already logged out. 2. Server is down. 3. Reports doesn’t exist. |
| **Business Rules** | N/A |
| **Assumptions:** | N/A |

#### UC-16: Complaints

|  |  |
| --- | --- |
| **Use Case ID:** | UC-16 |
| **Use Case Name:** | Complaints |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Admin | |
| **Description:** | Admin can check list of complaints, can respond to complaint and change complaint status. |
| **Trigger:** | Admin clicks on Complaints. |
| **Preconditions:** | 1. The admin has logged in into the system. |
| **Postconditions:** | 1. Complaints List is shown. |
| **Normal Flow:** | 1. The admin logs in. 2. The admin can clicks on the Complaints. 3. List of Complaints is shown to admin. 4. Admin selects a complaint. 5. Admin responds to the complaint. 6. Admin changes the status of complaint. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | N/A |
| **Exceptions:** | 1. Admin has already logged out. 2. Server is down. |
| **Business Rules** | BR-1: Complaints should get updated after admin responds to complaint.  BR-2: Complaints should get updated when admin changes the status of complaints. |
| **Assumptions:** | N/A |

### UC-17: Launderer Dashboard

|  |  |
| --- | --- |
| **Use Case ID:** | UC-15 |
| **Use Case Name:** | Launderer Dashboard |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Launderer | |
| **Description:** | After the launderer has logged in, the launderer dashboard will appear. The main elements that will be appear are orders, laundrette, Reviews, performance reports and logout. |
| **Trigger:** | The launderer wants to use the system. |
| **Preconditions:** | 1. The launderer has registered his account. 2. The launderer has logged in into the system. |
| **Postconditions:** | N/A |
| **Normal Flow:** | 1. The launderer logs in. 2. The launderer can clicks on the order. 3. The Orders windows appear displaying it’s information. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 1. The launderer logs in. 2. The launderer clicks on the Review. 3. The Review window appear displaying the sales information. |
| **Exceptions:** | 1. User has already logged out. 2. Server is down. 3. Api error. |
| **Business Rules** | N/A |
| **Assumptions:** | N/A |

#### UC-18: Services

|  |  |
| --- | --- |
| **Use Case ID:** | UC-18 |
| **Use Case Name:** | Services |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Launderer | |
| **Description:** | Launderer wants to see the services they provided in the launderette.  Launderer can modify these services, delete them and add new service. |
| **Trigger:** | The launderer clicks on Services |
| **Preconditions:** | 1. The launderer has registered his account. 2. The launderer has logged in into the system. |
| **Postconditions:** | Launderer gets redirected to services page |
| **Normal Flow:** | 1. The launderer logs in. 2. The launderer can clicks on the Launderette. 3. Launderer clicks on services. 4. List of services is displayed. 5. Launderer adds a new services. 6. Launderer specify the price of the service. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 1. After step 4 of normal flow. Launderer deletes a service. 2. Updated services are shown. |
| **Exceptions:** | 1. User has already logged out. 2. Server is down. 3. Api error. |
| **Business Rules** | N/A |
| **Assumptions:** | N/A |

#### UC-19: Orders

|  |  |
| --- | --- |
| **Use Case ID:** | UC-19 |
| **Use Case Name:** | Orders |
| **Actors:** | |  |  | | --- | --- | | Primary Actor: | Launderer | |
| **Description:** | Launderer wants to see the orders they finished which are ongoing or declined. Launderer can check details and payment of the order and can change the status of order. |
| **Trigger:** | The launderer clicks on orders. |
| **Preconditions:** | 1. The launderer has registered his account. 2. The launderer has logged in into the system. |
| **Postconditions:** | N/A |
| **Normal Flow:** | 1. The launderer logs in. 2. The launderer can clicks on the Launderette. 3. Launderer clicks on orders. 4. List of orders is displayed. 5. Launderer changes the state of order. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | 1. After step 4 of normal flow. Launderer clicks on order detail. 2. Order details are displayed. |
| **Exceptions:** | * User has already logged out. * Server is down. * Api error. |
| **Business Rules** | N/A |
| **Assumptions:** | N/A |

### UC-20: LogOut

|  |  |
| --- | --- |
| **Use Case ID:** | UC-16 |
| **Use Case Name:** | LogOut |
| **Actors:** | |  |  |  | | --- | --- | --- | | Primary Actor: | Client, Launderer,Admin |  | |
| **Description:** | When the actors are finished on working on the job they logged in for. They can click on log out to terminate the session. |
| **Trigger:** | User has finished up using the application. |
| **Preconditions:** | User has logged in and and he is done using the application |
| **Postconditions:** | The user is logged out and redirected to the login page. |
| **Normal Flow:** | 1. User is done using the application  2.The user clicks on the logout.  3. The user is redirected to the login page. |
| **Alternative Flows:**  **[Alternative Flow 1 – Not in Network]** | None |
| **Exceptions:** | None |
| **Business Rules** | N/A |
| **Assumptions:** | So that no one uses the system with the user credentials other than him. |

## Functional Requirements

The Functional Requirements needs to be mentioned in this section.

## Non-Functional Requirements

Non-Functional requirements are required to be explicitly mention in this section.

## FR-1: View Profile

|  |  |
| --- | --- |
| **Identifier** | FR-1 |
| **Title** | View Profile. |
| **Requirement** | If the client or Laundere are registered in the system then the admin can view their profiles.  The admin can view all the profiles which either be a Launderer or a client registered in the system. |
| **Source** | Admin class. |
| **Rationale** | The admin can view everyone that is registered to the system using the system and utilizing the services. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-2: Block/Unblock Client

|  |  |
| --- | --- |
| **Identifier** | FR-2 |
| **Title** | Block/unblock Client. |
| **Requirement** | If the client has some pending payment issues or other issues then the admin can block the client and after the issues are cleared then the admin can unblock him/her. |
| **Source** | Admin class. |
| **Rationale** | The admin can block/unblock client if there is and issue and can also unblock him. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-3: Complaints

|  |  |
| --- | --- |
| **Identifier** | FR-3 |
| **Title** | Complaints |
| **Requirement** | If a user has registered a complaint then the admin will be the one to resolve the all the complaints regarding the system or the services.  The client can also see the status of the complaint. |
| **Source** | Admin class. |
| **Rationale** | Problem in the system or services can occur os the admin must able to remove or resolve all those problems. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-4: Launderer List

|  |  |
| --- | --- |
| **Identifier** | FR-4 |
| **Title** | Launderer List |
| **Requirement** | The launderer list will contain all the launderers that are registered in the system and their respected information. |
| **Source** | Admin class. |
| **Rationale** | A list of all the launderers that have registered themselves in the system so that the admin can view it. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-5: Launderer Details

|  |  |
| --- | --- |
| **Identifier** | FR-5 |
| **Title** | Launderer Details |
| **Requirement** | The launderer details contains all the information regarding a particular launderer and the information can also include the review and performance reports. |
| **Source** | Admin class. |
| **Rationale** | The admin can view a specific launderer and see that launderer details. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-6: Create Account

|  |  |
| --- | --- |
| **Identifier** | FR-6 |
| **Title** | Create Account. |
| **Requirement** | If the client wants use the system he must have to create an account first.  Creating account is the first step and after that the user can avail all the services that he requires. |
| **Source** | Client class. |
| **Rationale** | The user/client cannot use the services until he had created account in the system. |
| **Business Rule (if required)** | None |
| **Dependencies** |  |
| **Priority** | High |

## FR-7: Select Service

|  |  |
| --- | --- |
| **Identifier** | FR-7 |
| **Title** | Select Service. |
| **Requirement** | After the users has registered himself onto the system then he can select from list of services that he wants such washing,ironing or dry cleaning. |
| **Source** | Client class. |
| **Rationale** | To use the available services he must have to select from a list of those services. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-8: Launderettes Profile

|  |  |
| --- | --- |
| **Identifier** | FR-8 |
| **Title** | Launderettes Profile |
| **Requirement** | If the user are unaware of the Launderette details then he can look into their profile and check details like services prices in price class to check if he can afford their services or not. |
| **Source** | Client class. |
| **Rationale** | To let the client know of the charges for the services so that he can apply knowing he can afford it or not. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Low |

## FR-9: Find Launderer.

|  |  |
| --- | --- |
| **Identifier** | FR-9 |
| **Title** | Find Launderer |
| **Requirement** | The find launderer will look for professionals who can get their laundry in the best way possible and the way they want.It will look for the launderer in that specific area and display them so that they can get their services done through them. |
| **Source** | Client class. |
| **Rationale** | As different areas will have different launderers, so the find launderer will find it and list launderers in that specific area. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-10: Client Profile

|  |  |
| --- | --- |
| **Identifier** | FR-10 |
| **Title** | Client Profile |
| **Requirement** | The profile requirement will contain all the personal details of a specific that the client provided during the creation of the account. These details can be changed when through the profile settings and updated.These details can be the user name,name, address etc. |
| **Source** | Client class. |
| **Rationale** | To contain the details of client so that they can be used and changed later,a profile FR is required. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-11: Booking

|  |  |
| --- | --- |
| **Identifier** | FR-12 |
| **Title** | Booking |
| **Requirement** | The Booking will enable the client to make the appointment for the pick-up of their laundry from their door-step. The booking will contain all the details that is address etc. |
| **Source** | Client class. |
| **Rationale** | For the client to set the appointment so that the laundry can be picked-up and delivered , a booking functional requirement is required. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-12: Launderer Dashboard.

|  |  |
| --- | --- |
| **Identifier** | FR-12 |
| **Title** | Launderer Dashboard |
| **Requirement** | The Launderer dashboard will contain both the services that are provided and the status of payment, orders. The Services are washing,ironing and dry cleaning etc and orders will contain will contain information about the laundry that is it done or not. It will also contains new offers provided. |
| **Source** | Client class. |
| **Rationale** | The home page meaning dashboard of the client will contain services and status and new offers functionality. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-13: Client Notification

|  |  |
| --- | --- |
| **Identifier** | FR-13 |
| **Title** | Client Notification. |
| **Requirement** | The notification requirement will give client the information like their laundry is done, their laundry is done and like their laundry is about to be delivered. |
| **Source** | Client class. |
| **Rationale** | To make the client updated regarding their laundry and new offers, a notification system is required |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-14: Feedbackand Review

|  |  |
| --- | --- |
| **Identifier** | FR-14 |
| **Title** | Feedback and Review |
| **Requirement** | The feedback will provide the client with the opportunity to let the launderer know how their experience was and how the can improve it. |
| **Source** | Client class. |
| **Rationale** | To let the launderer improve their services, a feedback system is required so that the launderer knows about their shortages and can overcome |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-15: Payment

|  |  |
| --- | --- |
| **Identifier** | FR-15 |
| **Title** | Payment |
| **Requirement** | The payment will contain all the payment history including all the payments that’s been paid, pending etc. |
| **Source** | Client class. |
| **Rationale** | The payment will contain all the payment history including all the payments that’s been paid, pending etc. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-16: Provide Services

|  |  |
| --- | --- |
| **Identifier** | FR-16 |
| **Title** | Provide Services. |
| **Requirement** | If the client requires any services then those services will be provided by the launderer. The Launderer will communicate with the client and then deliver the services. |
| **Source** | Launderer class. |
| **Rationale** | The Launderer will be the one managing all the services and client. He will the delieverer of the services. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High. |

## FR-17: Pick-up and Delivery

|  |  |
| --- | --- |
| **Identifier** | FR-17 |
| **Title** | Pick-up and Delivery |
| **Requirement** | When the client set’s up an appointment at a specific time then the launderer will be responsible for the pick-up of the laundry and after cleaning services done they are to be delivered to the destination. |
| **Source** | Launderer class. |
| **Rationale** | The Launderer will be the one managing the pick-up and delivery of the laundry. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-18: Communication

|  |  |
| --- | --- |
| **Identifier** | FR-18 |
| **Title** | Communication |
| **Requirement** | The Launderer will be managing the communication with the client.The Launderer will be responsible for getting the details from the client and in case of any query, he will be the one resolving that query. |
| **Source** | Launderer class. |
| **Rationale** | In case of any query or help, the launderer will be the one who resolves it. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-19: Accept/Decline

|  |  |
| --- | --- |
| **Identifier** | FR-20 |
| **Title** | Accept/Decline Client |
| **Requirement** | The Launderer can accept or decline a particular client based on the conditions. That is maybe the available appointment time is already fixed for another client or is from an area where we don’t provide these services. |
| **Source** | Launderer Class |
| **Rationale** | Launderer can accept or decline a particular client based on the conditions. That is maybe the available appointment time is already fixed for another client or is from an area where we don’t provide these services. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-20: Order Requests

|  |  |
| --- | --- |
| **Identifier** | FR-20 |
| **Title** | Order Requests |
| **Requirement** | The order requests will include all the order request sent by different clients to the launderer. It will includes all the order details. |
| **Source** | Launderer Class |
| **Rationale** | The Launderer can see all the order requests and all the details regarding those orders |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-21: Order request list

|  |  |
| --- | --- |
| **Identifier** | FR-21 |
| **Title** | Order request list |
| **Requirement** | A list of all the order is to be displayed when the launderer clicks on the order requests and then he will have have the option of selecting order requests |
| **Source** | Launderer Class |
| **Rationale** | When the launderer clicks on the order request, a list of all the order requests will be displayed and then launderer can accordingly. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-22: View Order Details

|  |  |
| --- | --- |
| **Identifier** | FR-22 |
| **Title** | View Order Details |
| **Requirement** | The view order details will display all the details of a specific order. The details will include the appointment timing, client details and launderer can accept and reject it. |
| **Source** | Launderer Class |
| **Rationale** | All the details will be displayed related to a single order when a clients clicks on it. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-23: Accept Order

|  |  |
| --- | --- |
| **Identifier** | FR-23 |
| **Title** | Accept Order |
| **Requirement** | The accept order will accept the clients order and then the launderer will pick-up the laundry and will make a schedule to this particular laundry and will dropped off at the destination. |
| **Source** | Launderer Class |
| **Rationale** | The launderer will agree to the appointment set by the client and then the laundry will picked up and dropped after the laundry services being done. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-24: Reject Order

|  |  |
| --- | --- |
| **Identifier** | FR-24 |
| **Title** | Reject Order |
| **Requirement** | The order will be reject by the launderer because the laundry cannot be delivered at the appointed time. |
| **Source** | Launderer Class |
| **Rationale** | The launderer can reject the order request sent by a particular client. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Low |

## FR-25: Find Laundrette

|  |  |
| --- | --- |
| **Identifier** | FR-25 |
| **Title** | Find Laundrette |
| **Requirement** | The find launderette requirement will look for the launderette in the nearby location. The client can also search the launderettes like by price, ratings and location. The list of all the launderettes will be displayed. |
| **Source** | Client Class |
| **Rationale** | If the client requires services after registering to the system then he can search for the launderettes through different ways such as by location,price and ratings. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-26: View Order Details

|  |  |
| --- | --- |
| **Identifier** | FR-26 |
| **Title** | View Order Details |
| **Requirement** | The view order details will display all the details of a specific order. The details will include the appointment timing, client details and launderer can accept and reject it. |
| **Source** | Client Class |
| **Rationale** | All the details will be displayed related to a single order when a clients clicks on it. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-27: Laundrette List

|  |  |
| --- | --- |
| **Identifier** | FR-27 |
| **Title** | Laundrette List |
| **Requirement** | The launderette list will display all the launderette available and the client can select the suitable one. He can then send request to him. |
| **Source** | Client Class |
| **Rationale** | The launderette list will contain the all the launderettes and client can select from it. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-28: Sort Laundrettes

|  |  |
| --- | --- |
| **Identifier** | FR-28 |
| **Title** | Sort Laundrettes |
| **Requirement** | The sort launderette provides client with the choice of displaying launderettes through ratings, prices and by location. |
| **Source** | Client Class |
| **Rationale** | The launderettes can be sorted in three different ways. They are as follows:   * Sort by location. * Sort by Price. * Sort by Ratings. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-29: Sort launderettes by location

|  |  |
| --- | --- |
| **Identifier** | FR-29 |
| **Title** | Sort launderettes by location |
| **Requirement** | The sort by location will display all the launderettes that are close at the top and display the farther ones below them. |
| **Source** | Client Class |
| **Rationale** | To provide clients with the convenience of locating the launderettes that are near and farther, sort by locating is used. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-30: Sort launderettes by Rating

|  |  |
| --- | --- |
| **Identifier** | FR-30 |
| **Title** | Sort launderettes by Ratings |
| **Requirement** | This will sort the launderettes in the descending order with highleu rated at the top and then coming down to the low rated ones. |
| **Source** | Client Class |
| **Rationale** | Some clients are looking for the best laundry services, so they look for the ratings and selects the launderettes. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-31: Sort launderettes by Price

|  |  |
| --- | --- |
| **Identifier** | FR-31 |
| **Title** | Sort launderettes by Price |
| **Requirement** | This will sort the launderettes with the low price launderettes at the top and ascending down in increasing order. |
| **Source** | Client Class |
| **Rationale** | A launderettes will be displayed in ascending order with launderette having low rates at the top and increasing as going down the list of launderettes. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-32: Book launderette

|  |  |
| --- | --- |
| **Identifier** | FR-32 |
| **Title** | Book launderette |
| **Requirement** | The client will select a particular laundretted that is providing the desired services as required by the client. Then the client can select the services and enter the booking details. |
| **Source** | Client Class |
| **Rationale** | The book launderette will select the desired launderette and the details will be provided to the launderette and request will be sent to the launderette. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-33: Select Services

|  |  |
| --- | --- |
| **Identifier** | FR-33 |
| **Title** | Select Services |
| **Requirement** | The client will have to select the laundry services that he requires and provided by the launderette. These services can be washing,ironing and dry cleaning. |
| **Source** | Client Class |
| **Rationale** | The select service will allow the client to select the services that he requires provided by particular launderer. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-34: Booking details

|  |  |
| --- | --- |
| **Identifier** | FR-34 |
| **Title** | Booking details |
| **Requirement** | The client will have to enter the booking details that can be address and time of appointment. |
| **Source** | Client Class |
| **Rationale** | The client will provide the necessary details as required by the launderer. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-35: Order history

|  |  |
| --- | --- |
| **Identifier** | FR-35 |
| **Title** | Order history |
| **Requirement** | The order history will include the all the details of a particular order such as the date of booking, pick-up and delivery information etc. |
| **Source** | Launderer Class |
| **Rationale** | A launderer can see the the details of a particular and all the reviews that the launderer provided about it. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-36: Laundrette Reviews

|  |  |
| --- | --- |
| **Identifier** | FR-36 |
| **Title** | Laundrette Reviews |
| **Requirement** | It will provided the launderer about the clients opinion and the quality of the services that he provides and then he can improve the system keeping in mind the review. |
| **Source** | Launderer Class |
| **Rationale** | To collect the clients opinion about how he felt about the launderer a reviews systems is important. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

## FR-37: Ongoing orders

|  |  |
| --- | --- |
| **Identifier** | FR-37 |
| **Title** | Ongoing orders |
| **Requirement** | The Lauderer can view all the ongoing orders and check the status of those orders. |
| **Source** | Launderer Class |
| **Rationale** | To look into the status of the ongoing orders the launderer can view the ongoing orders. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-38: Add service

|  |  |
| --- | --- |
| **Identifier** | FR-38 |
| **Title** | Add service |
| **Requirement** | If the launderette want to provide more services to the clients then he can add more services and the clients can get those services |
| **Source** | Launderer Class |
| **Rationale** | Add service will add more services to the launderette profile and then the clients can achieve those services. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-39: Delete Service

|  |  |
| --- | --- |
| **Identifier** | FR-39 |
| **Title** | Delete Service |
| **Requirement** | If the launderer are no longer providing a particular services then he can deletes those services so that they are no longer showing up on the launderer profile. |
| **Source** | Launderer Class |
| **Rationale** | The launderer can delete a services if they are no providing it no more. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-40: Performance Reports

|  |  |
| --- | --- |
| **Identifier** | FR-40 |
| **Title** | Performance Reports |
| **Requirement** | The performance report will contain the launderet sales report and the launderette review report. |
| **Source** | Launderer Class |
| **Rationale** | If the launderer wants to know about their performance then can view the performance report. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-41: Sales Report

|  |  |
| --- | --- |
| **Identifier** | FR-41 |
| **Title** | Sales Report |
| **Requirement** | The sales report will contain all the launderet sales that all the orders that are completed uptill now. |
| **Source** | Launderer Class |
| **Rationale** | The launderer can check their sales report through the sales report. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | Medium |

## FR-42: Review Report

|  |  |
| --- | --- |
| **Identifier** | FR-42 |
| **Title** | Review Report |
| **Requirement** | The review report will contain all the information regarding the services that a particular launderer provided and he can view it. |
| **Source** | Launderer Class |
| **Rationale** | The launderer can check how people see their services and how they can improve it by checking the review report. |
| **Business Rule (if required)** | None |
| **Dependencies** | None |
| **Priority** | High |

# Non-Functional Requirements

1. Performance:
   * The response time of the system shall be made as low as possible.
   * Responses to view information shall take no longer than 5 seconds to appear on the screen.
2. Security Requirements
   * System will use secured database.
   * Password encryption shall be used.
   * System will have different types of users and every user has access constraints.
3. Compatibility:

* The system shall be able to support all android versions from 7 to 10.
* The web shall also be supporting all the latest versions of web browsers.

1. Usability:
   * The system shall be very friendly.
   * The system shall be easy to use.
2. Sacalbility:
   * The system shall be able to accommodate max number of users without any fault such as system slowing down etc.

# Design and Architecture

The following parts of Software Design Description (SDD) report should be included in this chapter.



## System Architecture

Diagram

Description automatically generated

## Data Representation

**Booking**

This part of the application would be used to store the orders data that is new orders data, ongoing orders and completed orders. This module would encapsulate the information and send it over to the server until it reaches the required destination and is stored on the receiver end.

**Services**

The services part of the application would include all the existing services, new and modified services details. All the data related to the services will be stored in the database.

**Review/Feedback**

This part of the system will include data related to feedback system of the application such as data of all the feedbacks submitted including person who submitted, when submitted, and feedbacks details. It will also include the launderer responding to the review.

**Payments**

The part of the application stores the data related to payments such as all the payments and payments details added by the client or the launderer.

**Log in**

This part of the application deals with that when the application is installed on the mobile device the user is prompted to log in to their account that is already registered and holds the record in the system database.

**Orders Data**

This part of the application deals with keeping track of the order history of the user. It also includes data of ongoing orders. It also includes data related to the order status and is stored in postgresql and firebase.

**Profile**

Every user needs to have a profile before using the application and its services, the profile information is stored on the database, every user is identified by a unique username and email.

**Complaints**

This part of the application contains data related to the complaints such data about submitted complaints, resolved complaints. The data is stored in the database and retrieved when needed.

**Search**

The application has several search options to select from, the user can search the launderer and apply various filtering and sorting techniques to rearrange the data. The user can also search for the products on the other websites within the application the data would be retrieved, through the API call to the scraper script running on the cloud. The scraped data won’t be stored in the firebase or postgresql database.

**Sign up**

Before using the application and its services the user needs to be registered into the application by providing their email, password name etc.

**Client:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Name | String | 50 | User name |
| Bio | String | 50 | User description |
| address | String | 20 | User location |

**Launderer:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Name | String | 50 | User name |
| Bio | String | 50 | User description |
| address | String | 20 | User location |

**Payment:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| User | String | 50 | User name |
| Type | String | 20 | Payment type |

**Laundrette:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| launderer | String | 50 | Launderer name |
| name | String | 50 | User description |
| description | String | 50 | Launderette details |
| Available\_time | Int | 20 | availability |
| location | String | 20 | address |

**Services:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Launderette | String | 50 | Launderette name |
| Title | String | 50 | services name |
| description | String | 50 | services details |
| price | Int | 20 | Amount charge for service |

**Order:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Launderette | String | 50 | Launderette name |
| client | String | 50 | client name |
| amount | Int | 20 | Total amount |
| price | Int | 20 | Amount charge for service |
| Status | String | 50 | Status of order |
| date\_start | String | 50 | Order starting date |
| date\_end | String | 50 | Order finishing date |
| Services | String | 50 | Services availed |

**Delivery:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| launderette | String | 50 | Launderette name |
| client | String | 50 | client name |
| description | String | 50 | services details |
| Status | String | 50 | Status of order |
| date | String | 20 | Delivery date |

**Transactions:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| launderette | String | 50 | Launderette name |
| client | String | 50 | client name |
| Details | String | 50 | transaction details |
| Amount | Int | 20 | Services fees |
| Available\_balance | Int | 20 | Balance in account |
| date | String | 50 |  |

**Review:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| launderette | String | 50 | Launderette name |
| Client | String | 50 | client name |
| Review | String | 50 | services details |
| date | String | 20 | Date of review |

**Complaint:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Size** | **Description** |
| Client | String | 50 | client |
| Subject | String | 50 | Subject of complain |
| Date | String | 20 | Date of complain |
| complain | String | 50 | Complain text |

## Process Flow/Representation

Diagram, schematic

Description automatically generated

## 

## Design Models

In this section, present the UML diagrams (for object-oriented methodology). Sections 4.4.1 and 4.4.2 present the UML diagrams for OO methodology.

Note, in case of structured (procedural) methodology, you are required to present the respective set of diagrams, including Data-Flow Diagram, Event-Flow Diagram, Stat Diagram, and Activity Diagram.

### Structural Diagrams

This section would present the static structure of the system, its parts on different abstraction and implementation levels, and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

#### Class diagram

Diagram, schematic

Description automatically generated

#### Object diagram

#### Component diagram

#### Package diagram

#### Deployment diagram

### Behavioral Diagrams

This section would present the behavior diagrams that show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

#### Activity diagram

Diagram, schematic

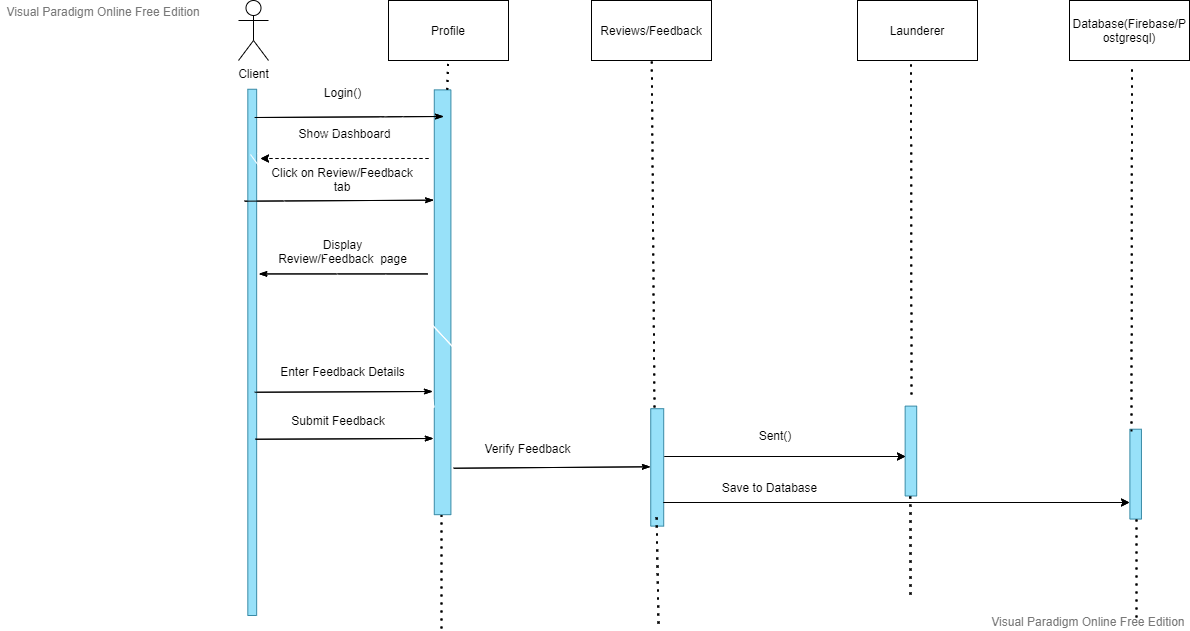
Description automatically generated

#### Diagram, schematic Description automatically generated

#### Sequence diagram

Diagram

Description automatically generated



Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

# Implementation

from django.db import models

from django.contrib.auth.models import User

import uuid

# Create your models here.

class Client(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE, null=True)

name = models.CharField(max\_length=200, null=True)

bio = models.CharField(max\_length=500, null=True, blank=True)

profile\_pic = models.ImageField(null=True, blank=True, default="default-profile.png")

address = models.models.CharField(max\_length=200, null=True)

def \_str\_(self):

return str(self.name)

class Launderer(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE, null=True)

name = models.CharField(max\_length=200, null=True)

bio = models.CharField(max\_length=500, null=True, blank=True)

profile\_pic = models.ImageField(null=True, blank=True, default="default-profile.png")

address = models.models.CharField(max\_length=200, null=True)

def \_str\_(self):

return str(self.name)

class Payment(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE, null=True)

type = models.CharField(max\_length=200, null=True)

def \_str\_(self):

return str(self.username)

class Launderette(models.Model):

launderer = models.ForeignKey(Launderer, null=True, on\_delete= models.SET\_NULL)

name = models.CharField(max\_length=200, null=True)

description = models.CharField(max\_length=500, null=True, blank=True)

available\_time = models.CharField(max\_length=500, null=True, blank=True)

cover\_photo = models.ImageField(null=True, blank=True, default="default-profile.png")

location = models.models.CharField(max\_length=200, null=True)

def \_str\_(self):

return str(self.name)

class Services(models.Model):

launderette = models.ForeignKey(Launderette, null=True, on\_delete= models.SET\_NULL)

title = models.CharField(max\_length=200, null=True)

description = models.CharField(max\_length=500, null=True, blank=True)

price = models.ImageField(null=True, blank=True, default="default-profile.png")

def \_str\_(self):

return str(self.title)

class StatusChoice1(models.TextChoices):

PENDING = 'pending', 'Pending'

FINISHED = 'finished', 'Finished'

ONGOING = 'ongoing', 'Ongoing'

DECLINED = 'declined', 'Declined'

class Order(models.Model):

client = models.ForeignKey(Client, null=True, on\_delete= models.SET\_NULL)

launderette = models.ForeignKey(Launderette, null=True, on\_delete= models.SET\_NULL)

price = models.IntegerField(default=0)

amount = models.IntegerField(default=0)

status =models.CharField(max\_length=50, blank=True, null=True,choices=StatusChoice1.choices,default=StatusChoice1.PENDING)

date\_started = models.DateTimeField(auto\_now\_add=True, null=True)

date\_end = models.DateTimeField(null=True)

services = models.ManyToManyField(Services, null=True, on\_delete= models.SET\_NULL)

def \_str\_(self):

return str(self.client.name)

class StatusChoice2(models.TextChoices):

PENDING = 'pending', 'Pending'

DELIVERED = 'delivered', 'Delivered'

SENT = 'sent', 'Sent'

class Delivery(models.Model):

client = models.OneToOneField(Client, null=True, on\_delete= models.SET\_NULL)

launderette = models.OneToOneField(Launderette, null=True, on\_delete= models.SET\_NULL)

description = models.CharField(max\_length=500, null=True, blank=True)

status =models.CharField(max\_length=50, blank=True, null=True,choices=StatusChoice2.choices,default=StatusChoice2.PENDING)

date = models.DateTimeField(auto\_now\_add=True, null=True)

def \_str\_(self):

return str(self.client.name)

class Transactions(models.Model):

client = models.ForeignKey(Client, null=True, on\_delete= models.SET\_NULL)

launderette = models.ForeignKey(Launderette, null=True, on\_delete= models.SET\_NULL)

details = models.CharField(max\_length=200, null=True)

amount = models.FloatField(default=0)

available\_balance = models.FloatField(default=0)

date = models.DateTimeField(auto\_now\_add=True, null=True)

def \_str\_(self):

return str(self.client.name)

class Review(models.Model):

launderette = models.ForeignKey(Launderette, null=True, on\_delete= models.SET\_NULL)

client = models.ForeignKey(Client, null=True, on\_delete= models.SET\_NULL)

review = models.CharField(max\_length=500, null=True)

date = models.DateTimeField(auto\_now\_add=True, null=True)

def \_str\_(self):

return str(self.client.name)

class Complaint(models.Model):

client = models.ForeignKey(Client, null=True, on\_delete= models.SET\_NULL)

subject = models.CharField(max\_length=200, null=True)

complain = models.CharField(max\_length=500, null=True)

date = models.DateTimeField(auto\_now\_add=True, null=True)

def \_str\_(self):

return str(self.client.name)

## Algorithm

**Algorithm 1: Login/ Sign up**

**Step 1:** The user can access the web application using any web browser and the mobile application needs to be installed on the user smart phones.

**Step 2:** The user would be prompted to enter their valid credentials to use the application and those credentials will be matched in the database. If the credentials are incorrect the user will be prompted with an error message.

**Step 3:** After verification from the database the user would be able to see and use the services offered by Cloud Laundrette.

**Algorithm 2: Get Services**

**Step 1:** After the student logs in, he/she can access the services section by clicking on the services tab. It will be displayed.

**Step 2:** All the services will be displayed. The client can choose a particular laundry service from a list of services.

**Step 3:** After choosing the services, Client will have to add laundry type that he/she wants to and the respective price will be displayed.

**Step 4:** After that a request send to the launderer for approval. If the request gets accepted then client will be notified with time and date of pickup and delivery.

**Step 5:** If the request gets cancelled then client will have to try some other time.

**Algorithm 3: Review/Feedback**

**Step 1:** The user needs to login to get to the feedback system. He/she can access the Feedback/Review section by clicking on the Reviews tab. All the reviews against a particular launderer will be displayed.

**Step 2:** He can then submit his own review by clicking on the add review button.

**Step 3:** The user will need to type the review and click on the submit button.

**Step 4:** After submitting the review, it will be stored in the database and becomes visible in the reviews section and the launderer can respond to individual’s reviews.

**Step 5:** When the launderer opens the web or mobile application and opens reviews section, the reviews will be displayed to the launderer and can respond to it then.

**Algorithm 4: Search**

**Step 1:** After the student logs in, he/she can access the search section by clicking on the search launderer tab. It will be displayed.

**Step 2:** The client can search through three different different methods. They are (a)search by location (b) by pricing (c) by ratings.

**Step 3:** Search by location will search for launderer in locations described by the client. All the available launderers will be displayed.

**Step 4:** Search by pricing will search for launderer and display a list of launderer from low pricing to high pricing.

**Step 5:** Search by rating will display all the top rated launderers at the top a followed by low rated launderers.

**Step 6:** The client can then choose the launderer that best suites him.

**Algorithm 5: Check order information**

**Step 1:** The client will have to click on the order tab to check order related information.

**Step 2:** A page will be displayed that will display new, ongoing and completed orders

**Step 3:** When clicked on the ongoing orders, a list will be displayed containing all the incomplete orders and related information will be displayed like start date, finishing date or delivery date.

**Step 4:** When clicked on the completed orders, a list will be displayed containing a history of all the complete orders.

**Step 5:** When clicked on the new order, client can place a new order and send an order request to the launderer.

**Algorithm 6: Payments**

**Step 1:** Click on the payments tab to display payments that are complete and those that are to be completed.

**Step 2:** By clicking on the completed payments, a list of all the completed and related information. All the data will be retrieved from the database and displayed.

**Step 3:** By clicking on the incomplete payments, client will be asked to select payment method.

**Step 4:** After selecting the payment, client will be prompted with to enter details.

**Step 5:**The payment API will verify those details and a confirmation message will appear for payment.

**Step 6:**If the details are incorrect then the client will be asked again to enter the details.

**Step 7:** By clicking on the confirm button, transaction will start and payment will be done in a moment then both client and launderer will be notified.

**Algorithm 7: Launderer new order request**

**Step 1:** After the launderer logs in, The launderer will have to click on the new orders to go to the new orders section. All the request from different clients will be displayed.

**Step 2:** After clicking on the approve button, launderer will be asked to add time and date of pickup and delivery for the laundry

**Step 3:** Client will be notified.

**Step 4:** The order will appear in the ongoing orders.

**Step 5:** If the orders is declined, client will be notified and asked to get back at later time.

**Algorithm 8: Launderer edit services**

**Step 1:** The launderer will have to click on the services tab to edit services.

**Step 2:** Click will select a service that he want to modify.

**Step 3:** After selecting the service, launderer will click on the detail that he want to modify.

**Step 3:** Then launderer will have to fill the details and click on the submit button. Service will get modified.

**Algorithm 9: Launderer delete service**

**Step 1:** The launderer will have to click on the services tab to delete services.

**Step 2:** Click will select a service that he want to delete.

**Step 3:** After selecting the service, launderer will click on the delete button and the service will get deleted.

**Algorithm 10: Launderer add new service**

**Step 1:** The launderer will have to click on the services tab to add services.

**Step 2:** Then launderer will have to add the details of the new service.

**Step 3:** After adding the details, launderer will click in the add button and it will get added.

## External APIs

Describe the APIs used in the following table.

**Table 5.1: Details of APIs used in the Project**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name of API** | **Description of API** | **Purpose of Usage** | **List down the function/class name in which it is used** |
|  |  |  |  |
|  |  |  |  |

## User Interface

Details about user interface with descriptions will be presented in this section.

# Testing and Evaluation

This chapter may include the following sections. (Students are required to perform the testing both manually and automatedly).



## Manual Testing

### Unit Testing

|  |  |
| --- | --- |
| Test Case ID | 1.1 |
| Test Case Name | View Profile |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Profile” button in the drawer. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the profile information is viewed successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Profile’ button. | 1. System displays the all the details regarding the user profile.  2. System displays the profile show his information. |
| Result | Pass The profile is viewed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.2 |
| Test Case Name | Change Profile Picture |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Change Profile Pic” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the profile information is viewed successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Change Profile Pic’ button.  3. The user uploads the pictures and click on save changes. | 1. System displays the all the details regarding the user profile.  2. Changes are displayed on the screen. |
| Result | Pass The profile picture is changedsuccessfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.3 |
| Test Case Name | Change Profile Picture |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Change Profile Pic” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the profile picture is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Change Profile Pic’ button.  3. The user uploads the pictures and click on save changes. | 1. System displays a form with current profile picture with option to upload a new one.  2. Changes are displayed on the screen. |
| Result | Pass The profile picture is changed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.4 |
| Test Case Name | Edit general Info |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Edit general Info” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the general information is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Edit general Info’ button.  3. The user fills the data that he want to change and click on save changes button. | 1. System displays a form with current profile general information with option to change name and address.  2. Changes are displayed on the screen. |
| Result | Pass The profile picture is changed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.5 |
| Test Case Name | Change Email |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Email” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the email is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change email’ button.  3. The user enters the email that he want to change and click on save changes button. | 1. System displays a form with an email change option.  2. The system checks if the email is unique then changes will be reflected otherwise the email will stay the same.  3. Changes are displayed on the screen. |
| Result | Pass The email is changed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.5.1 |
| Test Case Name | Change Email |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Email” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the email is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change email’ button.  3. The user enters the email that he want to change and click on save changes button. | 1. System displays a form with an email change option.  2. The system checks if the email is not unique the email will stay the same.  3. Changes are not displayed on the screen. |
| Result | Pass The email will not be updated. |

|  |  |
| --- | --- |
| Test Case ID | 1.5.1 |
| Test Case Name | Change Email |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Email” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the email is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change email’ button.  3. The user enters the email that he want to change and click on save changes button. | 1. System displays a form with an email change option.  2. The system checks if the email is not unique the email will stay the same.  3. Changes are not displayed on the screen. |
| Result | Pass The email will not be updated. |

|  |  |
| --- | --- |
| Test Case ID | 1.5.2 |
| Test Case Name | Change Email |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Email” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the email is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change email’ button.  3. The user enters the email that he want to change and click on save changes button. | 1. System displays a form with an email change option.  2. The system checks if the email is incorrect then email will stay the same.  3. Changes are not displayed on the screen. |
| Result | Pass The email will not be updated. |

|  |  |
| --- | --- |
| Test Case ID | 1.6 |
| Test Case Name | Change Password |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Password” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the password is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change Password’ button.  3. The user enters the password that he want to change and click on save changes button. | 1. System displays a form with old password , new password and confirm password Textfields.  2. The system checks if the new password is different than the old password.  3. Changes are displayed on the screen. |
| Result | Pass The password is updated successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.6.1 |
| Test Case Name | Change Password |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Password” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the password is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change Password’ button.  3. The user enters the password that he want to change and click on save changes button. | 1. System displays a form with old password , new password and confirm password Textfields.  2. The system checks if the new password is same as the old password.  3. Changes are not displayed on the screen. |
| Result | Pass The password is not updated successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.6.2 |
| Test Case Name | Change Password |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Password” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the password is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change Password’ button.  3. The user enters the password that he want to change and click on save changes button. | 1. System displays a form with old password , new password and confirm password Textfields.  2. The system checks if the new password meets the requirements that new password is has a minimum of 6 characters and one capital alphabets and also contains numbers or special character.  3. Changes are displayed on the screen. |
| Result | Pass The password is updated successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.6.3 |
| Test Case Name | Change Password |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “change Password” button in the profile section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that the password is updated successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘change Password’ button.  3. The user enters the password that he want to change and click on save changes button. | 1. System displays a form with old password , new password and confirm password Textfields.  2. The system checks if the new password is same matches the previous password. The password is not changed.  3. Changes are not displayed on the screen. |
| Result | Pass The password is not updated successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.1 |
| Test Case Name | View Order Requests |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Order request” button in the order section. |
| Pre-condition | The launderer must be logged in. |
| Test Verification | Test will verify that all the orders are displayed successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order request’ button. | 1. If there are orders system displays a list of orders else display no new orders. |
| Result | Pass All the orders are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.2 |
| Test Case Name | Accepts Requests |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Accept Order” button in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that the orders are accepted successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user clicks on the accept button. | 1.System displays a list of orders.  2. The order is placed in the ongoing orders. |
| Result | Pass The order is accepted successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.3 |
| Test Case Name | Decline Requests |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Decline Order” button in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that all the orders are declined successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user clicks on the decline button. | 1.System displays a list of orders.  2. The order is placed in the declined orders. |
| Result | Pass The order is declined successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.4 |
| Test Case Name | Ongoing Orders |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Ongoing Orders” button in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that all the ongoing orders are displayed successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user clicks on the ongoing orders button. | 1.System displays a list of ongoing orders. |
| Result | Pass The ongoing orders are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.5 |
| Test Case Name | Search Order |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that the searched order is displayed successfully. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user clicks on the search order button. | 1.System displays the searched order. |
| Result | Pass The searched order is displayed successfully. |
| Test Case ID | 2.5.1 |
| Test Case Name | Search Order by client name |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that the order search by name filter works correctly. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user clicks on the search order by name button. | 1.System displays the order searched by name. |
| Result | Pass The searched order by name is displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.5.2 |
| Test Case Name | Search Order by price |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button and select the price filter in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that the order search by price filter works correctly. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user enters the price.  4. The user clicks on the search order by name button. | 1. System will check for orders with prices from a specific range to a specific range.  2.System displays the order searched by prices. |
| Result | Pass The searched order/orders by prices is displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.5.3 |
| Test Case Name | Search Order by date |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button and select the date filter in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that the order search by date filter works correctly. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user enters the date.  4. The user clicks on the search order by date button. | 1. System will check for orders with date from a specific range to a specific range.  2.System displays the order searched by date. |
| Result | Pass The searched order/orders by date is displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.5.4 |
| Test Case Name | Check Details |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “ongoing” button and select the check details button. |
| Pre-condition | The launderer must be logged in and orders must be available. |
| Test Verification | Test will verify that the launderer can view the order details. |
| Actions | System response |
| 1. User opens the logs in by providing the credentials and clicking in the sign in button.  2. User clicks on ‘Order’ button.  3. The user clicks on the check details button. | 1.System displays the details of that specific order. |
| Result | Pass The details of the order are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.5.5 |
| Test Case Name | Change ongoing order status. |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Already registered order, status value, order data. |
| Pre-condition | The launderer must be logged in.  The relevant order should already exist.  The user should have privilege to update order data. |
| Test Verification | Test will verify that the order status is changed to “finished” successfully. |
| Actions | System response |
| 1. User will click details of order against ongoing order table. 2. User will click finished button to change the status. | 1. System displays the details of that specific order. 2. System will update order status to “finished”. |
| Result | Pass the order’s status is changed “finished” successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.5.6 |
| Test Case Name | Change ongoing order status. |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Already registered order, status value, order data. |
| Pre-condition | The launderer must be logged in.  The relevant order should already exist.  The user should have privilege to update order data. |
| Test Verification | Test will verify that the order status is changed to “cancel” successfully. |
| Actions | System response |
| 1. User will click details of order against ongoing order table. 2. User will click “cancel” button to change the status. | 1. System displays the details of that specific order. 2. System will update order status to “cancel”. |
| Result | Pass the order’s status is changed “cancel” successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.6 |
| Test Case Name | List Order History. |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Order History” button to view all orders. |
| Pre-condition | The launderer must be logged in.  The user should have privilege to view orders data. |
| Test Verification | Test will verify that all the orders are viewed successfully |
| Actions | System response |
| 1. User will click orders from side nav bar. 2. User will click “Order History” to view all orders. | 1. System displays dropdown order menu. 2. System displays all orders in table. |
| Result | Pass all orders are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.6.1 |
| Test Case Name | Search Order by client name |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Valid text for client name. |
| Pre-condition | The launderer must be logged in and orders must be available.  The user must have privilege to search order. |
| Test Verification | Test will verify that Orders or Order containing the entered words in search field will be displayed in table. |
| Actions | System response |
| 1. User will click orders from side nav bar. 2. User will click “Order History” to view all orders. 3. User will enter keywords for name in search field. | 1. System displays dropdown order menu. 2. System displays all orders in table. 3. System will search the entered words in table’s client column. Orders or Order containing keywords will be displayed in table. |
| Result | Pass all the Orders or Order containing the search Keywords searched successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.6.2 |
| Test Case Name | Search Order by price |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button and select the price filter in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available.  The user must have privilege to search order. |
| Test Verification | Test will verify that the order search by price filter works correctly. |
| Actions | System response |
| 1. User will click orders from side nav bar. 2. User will click “Order History” to view all orders. 3. The user enters the price. 4. The user clicks on the search order by name button. | 1. System displays dropdown order menu. 2. System displays all orders in table. 3. System will check for orders with prices from a specific range to a specific range. 4. System displays the order searched by prices. |
| Result | Pass The searched order/orders by prices is/are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.6.3 |
| Test Case Name | Search Order by date |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button and select the date filter in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available.  The user must have privilege to search order. |
| Test Verification | Test will verify that the order search by date filter works correctly. |
| Actions | System response |
| 1. User will click orders from side nav bar. 2. User will click “Order History” to view all orders. 3. The user enters the date. 4. The user clicks on the search order by date button. | 1. System displays dropdown order menu. 2. System displays all orders in table. 3. System will check for orders with date from a specific range to a specific range. 4. System displays the order searched by date. |
| Result | Pass The searched order/orders by date is/are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.6.4 |
| Test Case Name | Search Order by status |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “search” button and select the status filter in the order section. |
| Pre-condition | The launderer must be logged in and orders must be available.  The user must have privilege to search order. |
| Test Verification | Test will verify that the order search by status filter works correctly. |
| Actions | System response |
| 1. User will click orders from side nav bar. 2. User will click “Order History” to view all orders. 3. The user enters the date. 4. The user clicks on the search order by date button. | 1. System displays dropdown order menu. 2. System displays all orders in table. 3. System will check for orders with date from a specific range to a specific range. 4. System displays the order searched by date. |
| Result | Pass The searched order/orders by status is/are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 2.6.5 |
| Test Case Name | Check Details |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “Order History” button and select the check details button. |
| Pre-condition | Test will verify that the launderer can view the order details. |
| Test Verification | System response |
| Actions | 1.System displays the details of that specific order. |
| 1. User will click orders from side nav bar. 2. User will click “Order History” to view all orders. 3. The user clicks on the check details button. | 1. System displays dropdown order menu. 2. System displays all orders in table. 3. Pass The details of the order are displayed successfully. |
| Result | Pass The details of the order are displayed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.2 |
| Test Case Name | Register new complaint |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Makki Anjum | Date: 06 August 2021 |
| Test Data | Valid data for complaint’s title, details, choose severity, choose launderer, launderer details. |
| Pre-condition | The user must be client to register new complaint. |
| Test Verification | Test will verify that the complaint is registered successfully |
| Actions | System response |
| 1. User will click on new complaint button.  2. User will add information for the fields and click register button. | 1. System will open a dialog box.  2. System will show corresponding invalid fields errors. |
| Result | Pass The complaint not added successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.3 |
| Test Case Name | View detail of complaint |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “details” button in table against the complaint. |
| Pre-condition | Test will verify that the complaint is viewed successfully. |
| Test Verification | Test will verify that the complaint is viewed successfully. |
| Actions | System response |
| 1. User will click Details button in complaints table. | 1. System will display a dialog containing all the details. |
| Result | Pass The complaint is viewed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.4 |
| Test Case Name | View all of complaints |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Click on “complaints” button on the tab in the dashboard. |
| Pre-condition | The user must have privilege to view the complaints and database has some complaints. |
| Test Verification | Test will verify that all the complaints are viewed successfully. |
| Actions | System response |
| 1. User will have to be on Dashboard page to view all complaints. | 1. System will display a dialog containing all the details. |
| Result | Pass All complaints are viewed successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.5 |
| Test Case Name | Give Feedback |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Valid input fields Feedback remarks, feedback tags i.e. satisfied or not satisfied |
| Pre-condition | The user role must be complainer and that complaint’s status is not ‘in-progress’. |
| Test Verification | Test will verify that the complaint’s feedback is given successfully. |
| Actions | System response |
| 1. User opens the complaint details page by clicking details button.  2. User clicks on ‘Give feedback’ button.  3. User enters all the input fields and clicks ‘submit’ | 1. System displays the complaints details dialog box to user.  2. System displays feedback dialog box to enter the feedback fields.  3. System displays the message that complaint feedback is given successfully. |
| Result | Pass The feedback is given successfully. |

|  |  |
| --- | --- |
| Test Case ID | 1.6 |
| Test Case Name | Give Feedback |
| Testing Environment | Visual Studio Code 1.40.2 |
| Tested By: Usman Awan | Date: 08 August 2021 |
| Test Data | Valid input fields Feedback remarks, feedback tags i.e. satisfied or not satisfied |
| Pre-condition | The user role must be complainer and that complaint’s status is not ‘in-progress’. |
| Test Verification | Test will verify that the complaint’s feedback is given successfully. |
| Actions | System response |
| 1. User opens the complaint details page by clicking details button.  2. User clicks on ‘Give feedback’ button.  3. User enters all the input fields and clicks ‘submit’ | 1. System displays the complaints details dialog box to user.  2. System displays feedback dialog box to enter the feedback fields.  3. System displays the message that complaint feedback is given successfully. |
| Result | Pass The feedback is given successfully. |

**Unit Testing 1:** Login as FYP Committee

**Testing Objective:** To ensure the login form is working correctly.

**Test Case Id:** BU\_001

**Test Case Description:** Test the login functionality.

**Test Scenario:** Verify on entering valid username and password, the user can login.

**Table 6.1: Test Cases for Login**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail/Not Executed/ Suspended** |
| 1. | Verify user login after click on the ‘Login’ button on login form with correct input data | Username:  L001  Password:  1234 | Successfully log into the main page of the system as FYP Committee member. | As Expected | Pass |
| 2. |  |  |  |  |  |

**Unit Testing 2:** Edit Profile

**Testing Objective:** To ensure the edit profile form is working properly.

**Test Case Id:** BU\_002

**Test Case Description:** Test the edit profile functionality.

**Test Scenario:**

**Table 6.2: Test Cases for Edit Profile**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail/Not Executed/ Suspended** |
| 1. | Verify user login after click on the ‘Login’ button on login form with correct input data | Username:  L001  Password:  1234 | Successfully log into the main page of the system as FYP Committee member. | As Expected | Pass |
| 2. |  |  |  |  |  |

### Functional Testing

The functional testing will take place after the unit testing. In this functional testing, the functionality of each of the module is tested. This is to ensure that the system produced meets the specifications and requirements.

**Functional Testing 1:** Login with different roles

**Test Objective**: To ensure that the correct page with the correct navigation bar is loaded.

**Test Case Id:** BU\_003

**Test Case Description:**

**Test Scenario:**

**Table 6.3: Test Cases for Login with different Roles**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail/Not Executed/ Suspended** |
| 1. | Login as the ‘Admin’. | Username: beyondtalkadmin  Password: admin12345 | Home page for the admin is loaded. | Login as a ‘Admin’. | Pass |
| 2. | Login as a ‘Customer’. | Username:samia  Password:abc123@ | Home page for the customer is loaded. | Login as a ‘Customer’. | Pass |
| 3. | Login as an ‘Employee’. | Username:nushmia  Password:abc123@ | Home page for the employee is loaded. | Login as an ‘Employee’ | Pass |

### Integration Testing

Please specify the type of applied integration strategy, i.e. top down, bottom up. Also, elaborate the applied integration strategy.

**Table 6.4: Test Cases for Integration Testing of Unit X**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Test Case/Test Script** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail/Not Executed/ Suspended** |
| 1. | Login as “FYP Committee” member | Username: L001  Password: 1234 | Login successful and the FYP Committee page with its navigation bar is loaded and in the view profile page | As Expected | Pass |
| 2. | Upload student record for Project 1 | - | File successfully uploaded and return to the upload page. Student records are updated. | As Expected | Pass |
| 3. | View supervising student | - | The list of supervisees shown on the screen. | As Expected | Pass |
| 4. |  |  |  |  |  |

### System Testing

Once the system has been successfully developed, testing has to be performed to ensure that the system working as intended. This is also to check that the system meets the requirements stated earlier. Besides that, system testing will help in finding the errors that may be hidden from the user. There are few types of testing which includes the unit testing, functional testing and integration testing. The testing must be completed before it is being deploy for user to use.

## Automated Testing:

This section will discuss the testing tools used to automatically test the targeted project.

**Table 6.5: Tools employed for Automated Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tool Name** | **Tool Description** | **Applied on [list of related Test Cases / FR / NFR]** | **Results** |
|  |  |  |  |
|  |  |  |  |

# Conclusion and Future Work

This chapter concludes the project and highlights future work.



## Conclusion

The proposed project is a both an android and web application and will be used for providing online laundry services and managing all the clients. This system will have complaints system and review/feedback system. Some system providing laundry services were either web-based or android based applications, so we are developing a system that will be both web-based or android-based.

## Future Work

We are looking forward to add google maps to the android application and for the web application an API will be created. Then that API will be used in our android application and all the data will be fetch through that API.

# References

References to any book, journal paper or website should properly be acknowledged. Please consistently follow the style. The following are few examples of different resources i.e. journal article, book, and website.

* 1. Lyda M.S. Lau, Jayne Curson, Richard Drew, Peter Dew and Christine Leigh, (1999), Use Of VSP Resource Rooms to Support Group Work in a Learning Environment, ACM 99, pp-2. **(Journal paper example)**
  2. Hideyuki Nakanishi, Chikara Yoshida, Toshikazu Nishmora and TuruIshada, (1996), FreeWalk: Supporting Casual Meetings in a Network, pp 308-314 **(paper on web)** http://www.acm.org/pubs/articles/proceedings/cscw/240080/p308-nakanishi.pdf
  3. Ali Behforooz& Frederick J.Hudson, (1996), Software Engineering Fundamentals, Oxford University Press. Chapter 8, pp255-235. **(book reference example)**
  4. Page Author, Page Title, http://www.bt.com/bttj/archive.htm, Last date accessed**. (web site)**