**COMSATS University Islamabad,   
Park Road, Chak Shahzad, Islamabad Pakistan**

SOFTWARE DESIGN DESCRIPTION   
(SDD DOCUMENT)

for

**<Cloud Laundrette>**  
Version 1.0

***By***

**M Usman Awan CIIT/SP18-BSE-086/ISB**

**M Anwar Sameen CIIT/SP18-BSE-076/ISB**

***Supervisor*Ms Bushra Naz**

*Bachelor of Science in Computer Science (2020-2021)*

**Table of Contents**

Revision History iv

1. Introduction 1

2. Design methodology and software process model 1

3. System overview 2

3.1 Architectural design 2

3.2 Process flow/Representation 3

4. Design models [along with descriptions] 5

5. Data design 9

5.1 Data dictionary 9

6. Algorithm & Implementation 11

7. Software requirements traceability matrix 17

8. Human interface design 18

8.1 Screen images 19

8.2 Screen objects and actions 19

9. Appendix I 27

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason for changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

**Application Evaluation History**

|  |  |
| --- | --- |
| **Comments (by committee)**  **\*include the ones given at scope time both in doc and presentation** | **Action Taken** |
|  |  |
|  |  |

Supervised by

<Supervisor’s Name>

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 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.

# Design methodology and software process model

* **Design Methodology:**

The Design Methodology that we will be using is Object Oriented because data can be easily organized into objects and classes.

* **Software Process Model:**

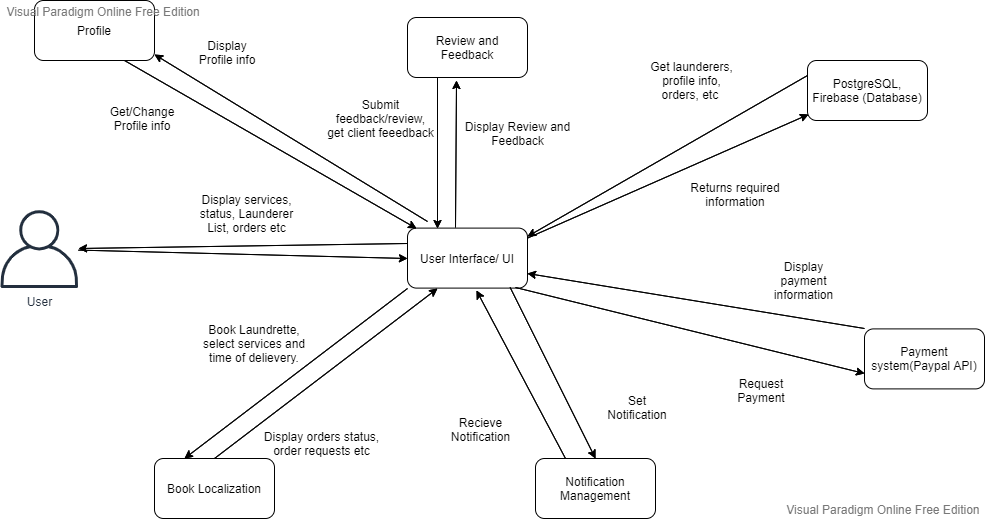
Incremental model is selected because this model make risk management easy as risky pieces are handled during the iterations and as our system requirements are clearly understood. The model generates working software quickly and early during the software life cycle and it is easier to test and debug. Move over, it is flexible and less expensive to change requirements and scope.

# System overview

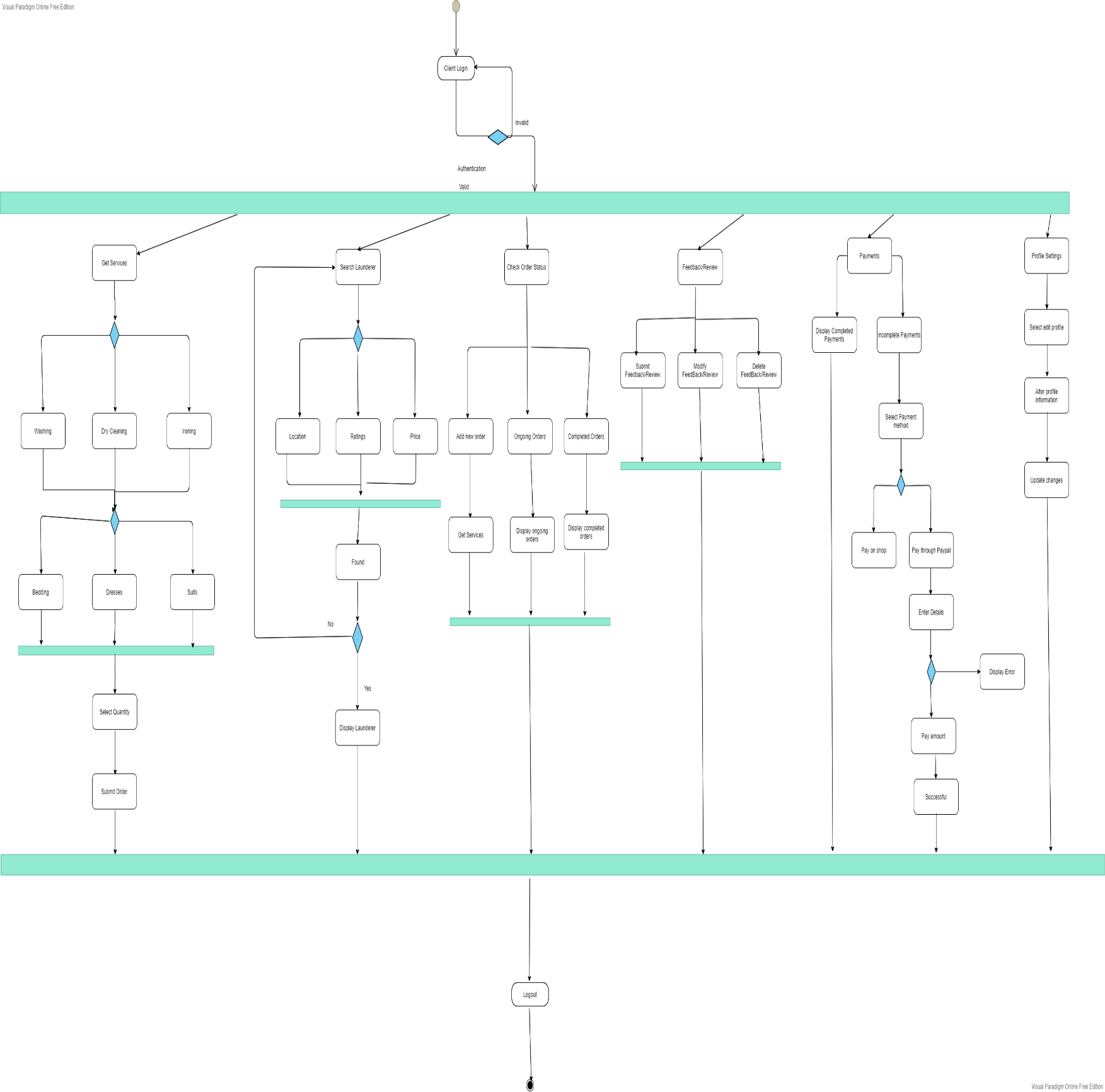
Many organizations such as hospitals, hotels etc are often faced by laundry problems as they don’t have the necessary equipment’s / tools or staff for cleaning of their laundry. As far as homes are concerned, people are so busy in their daily life that they don’t have time to take care of their laundry.

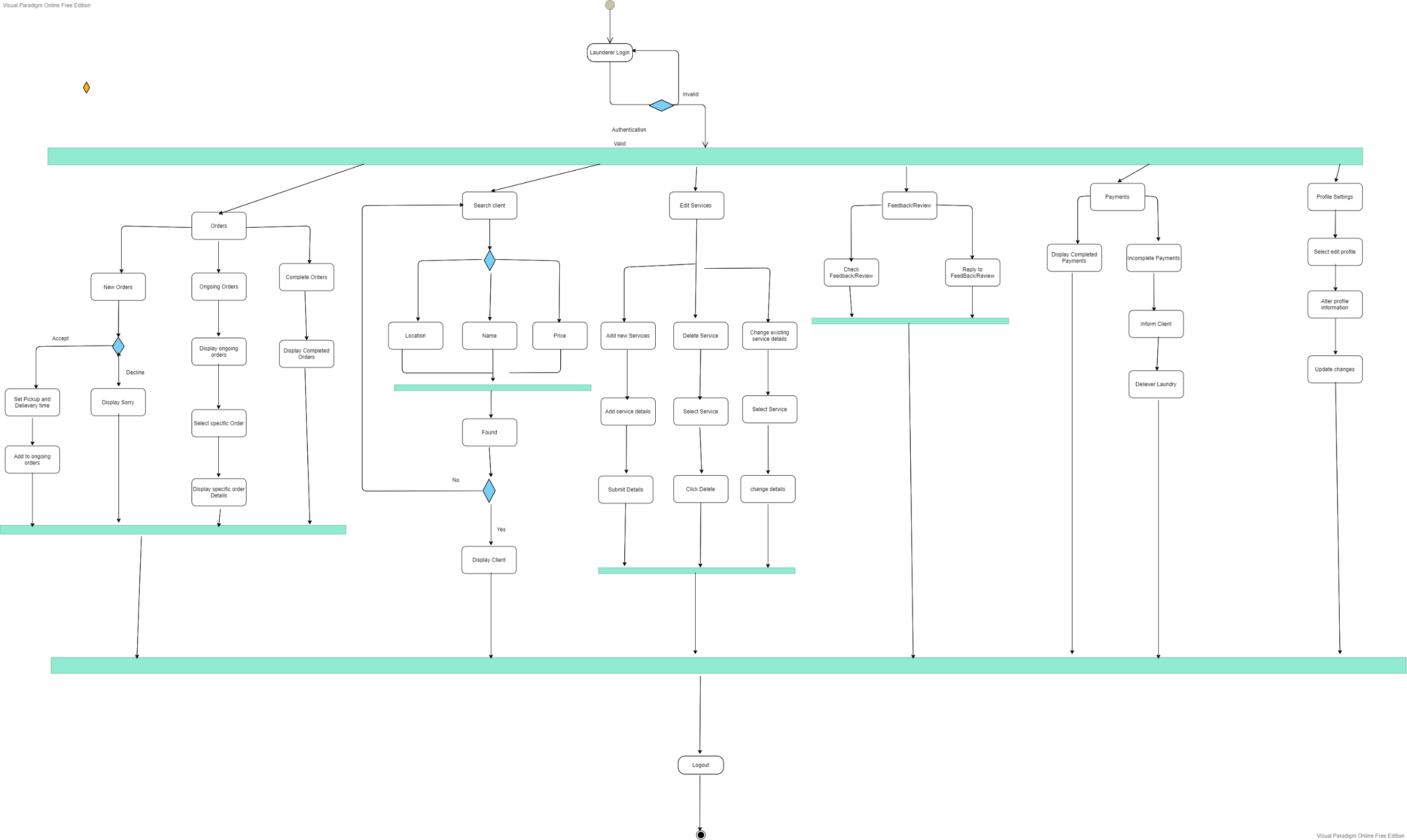
So the proposed system will be an online laundry service that will provide homes and organizations with laundry services such as washing, dry cleaning and ironing etc. They will set an appointment and the laundry will be picked from their doorsteps. The required services will be done and then again delivered at their doorsteps.

## Architectural design



## Process flow/Representation

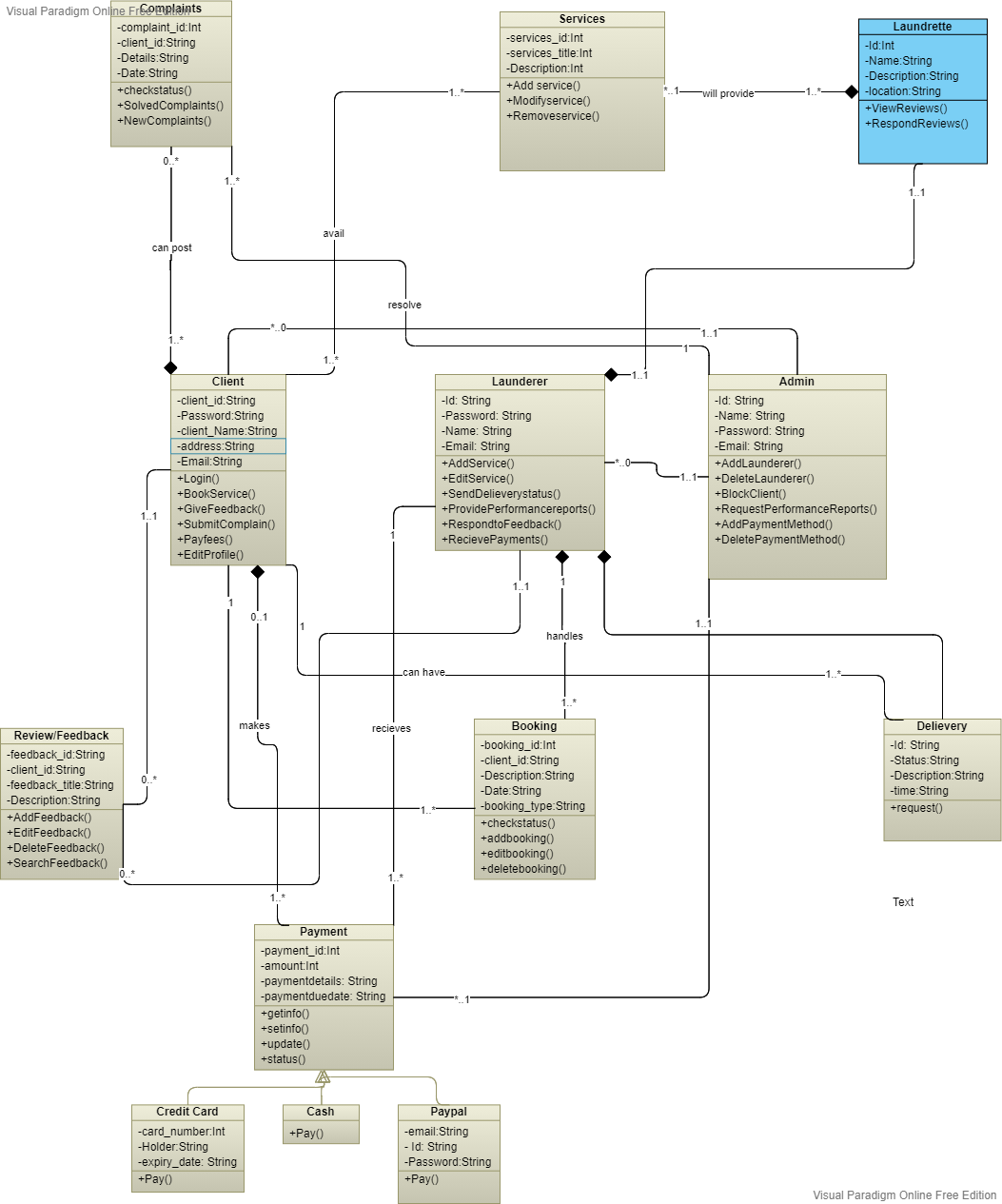




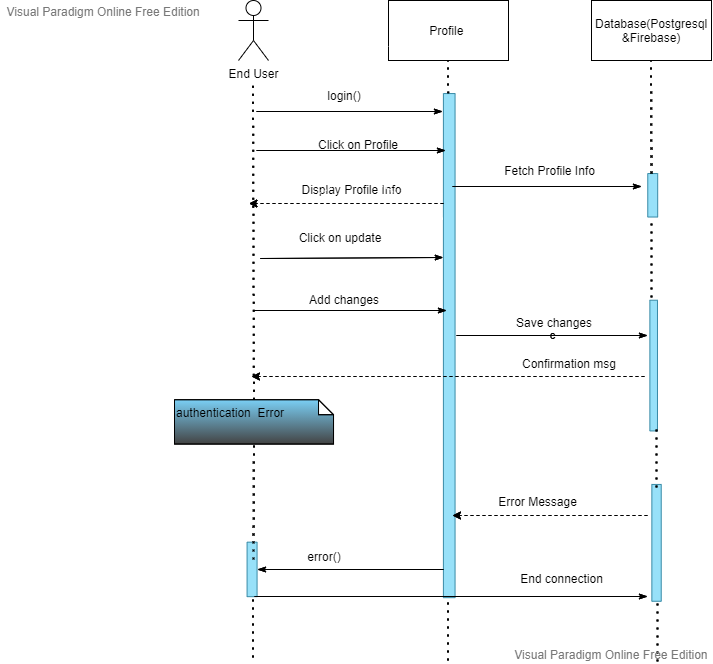
# Design models

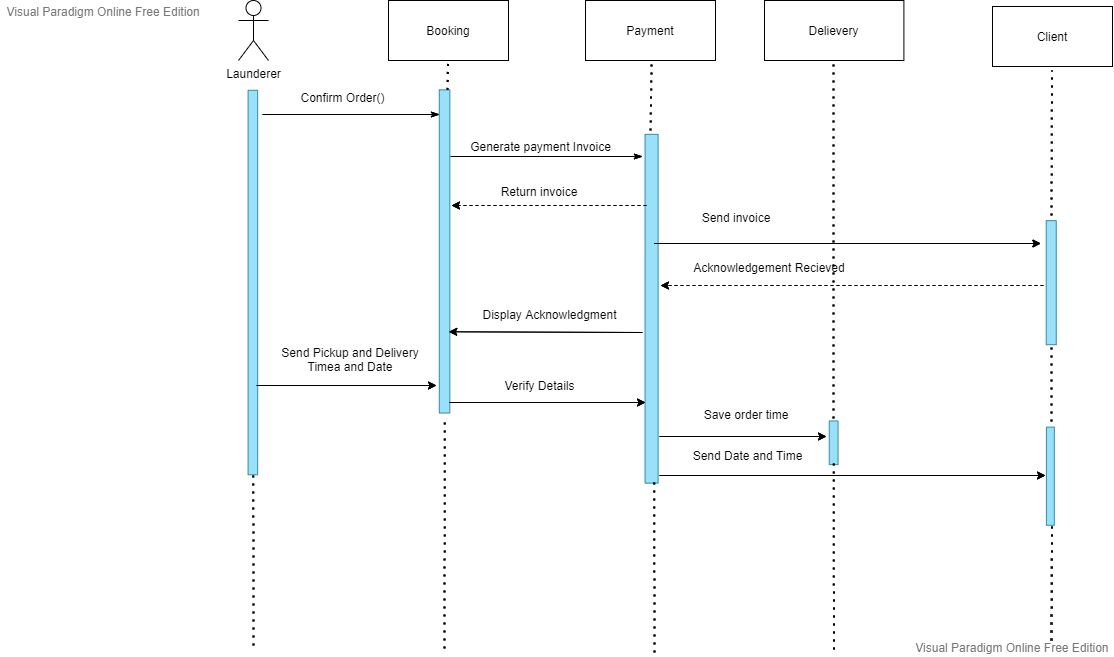
**Following are the design models :**

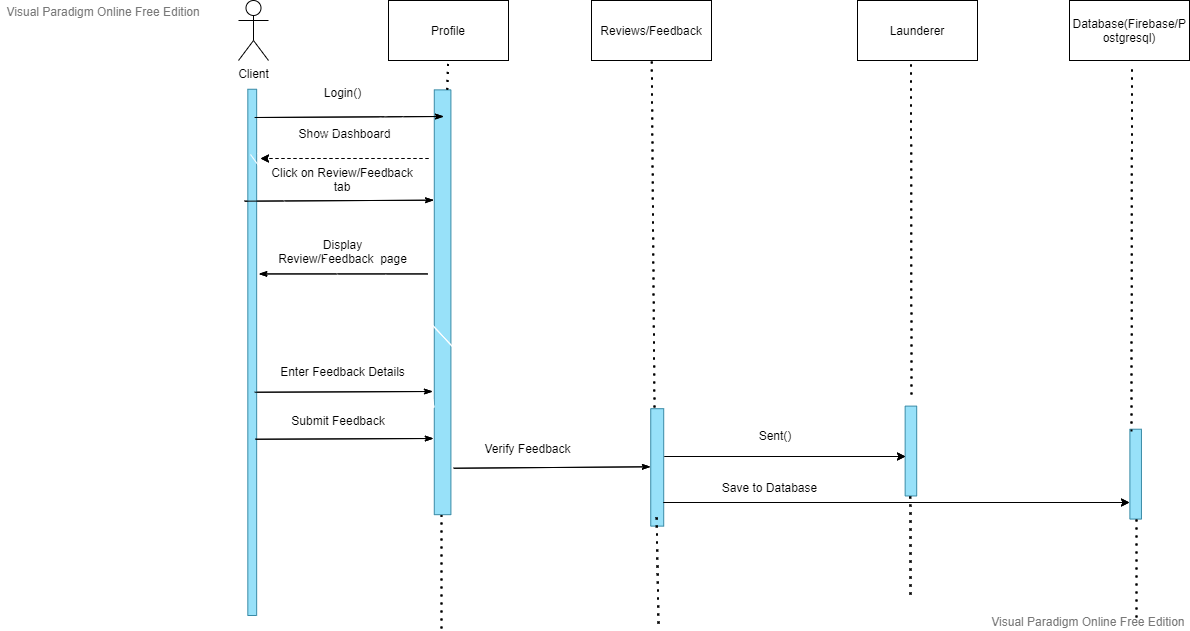
* Class Diagram

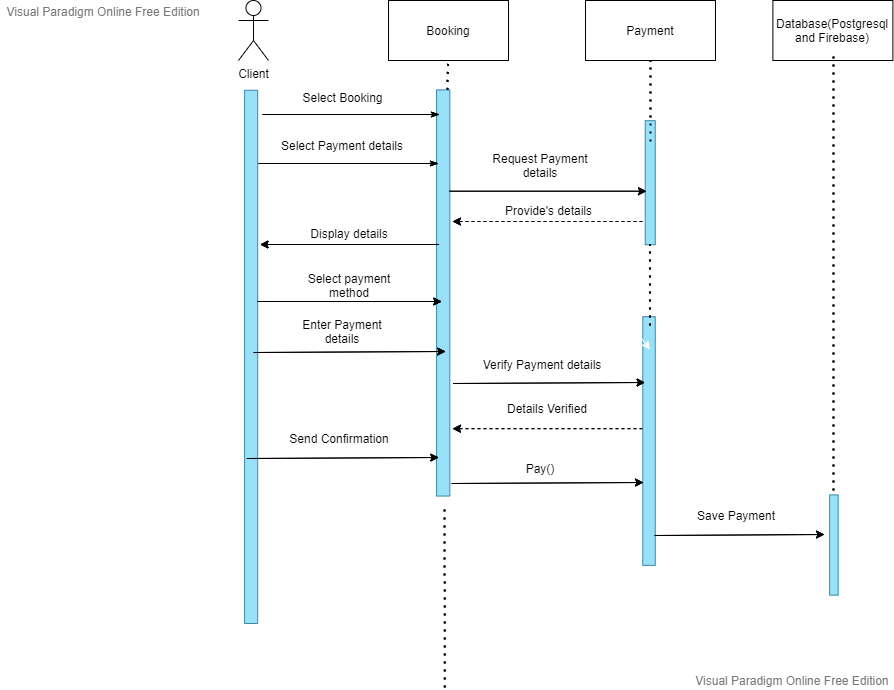


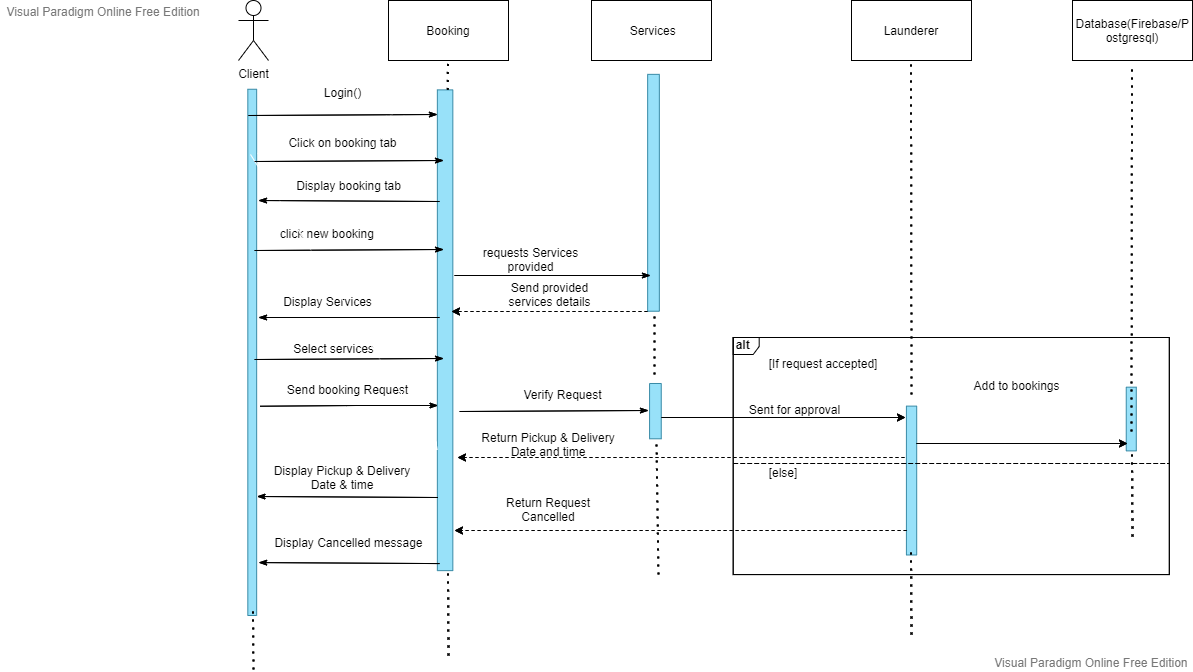
* Sequence Diagram











# Data design

The application is very rich in the terms of the data, the application is dependent upon various data sources, there is versatile kind of data coming from all dimensions from user logins, orders, services.. The data would be stored in a backend database using Google’s powerful cloud structure of firebase and firestore and Postgresql which is also a powerful database.

Some of the data would be stored locally on the user's mobile phone because the local storage is always available, the files saved on the local storage are only accessible to the applications.

The user data that includes the text-based data would be stored on the Firebase cloud. The interfaces of the application would have the data coming from the cloud API calls, all of the data would be gathered in the application and would be used in the required places throughout the application. The ListArrrays would be used locally to store the data within the application this goes both for Dart and python-based programs. The dictionary data structure would also be used to store the key-value pair of the data and the results would be retrieved through the use of API.

## Data dictionary

**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 |

# Algorithm & Implementation

**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.

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)

# Software requirements traceability matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Req. Number** | **Ref. Item** | **Design Component** | **Component Items** |
| FR01 | Class Diagram | Launderer | Register(), Login(), |
| FR02 | Class Diagram | Launderer,admin,client | UpdateProfile() |
| FR03 | Class Diagram | Launderer | AddService(),Editservice() |
| FR04 | Class Diagram | Launderer | SendDelieverystatus() |
| FR05 | Class Diagram | Launderer | RespondtoFeedback() |
| FR06 | Class Diagram | Admin | Addlaunderer(),RemoveLaunderer() |
| FR07 | Class Diagram | Client | Submitfeedback(),Submitcomplain() |
| FR08 | Class Diagram | Client | Payfees() |
| FR09 | Class Diagram | Client | BookService() |
| FR10 | Class Diagram | Launderer | RecievePayments(), |
| FR11 | Class Diagram | Admin | AddPaymentMethod() |
| FR12 | Class Diagram | Client | SearchLaunderer() |
| FR13 | Class Diagram | Booking | Checkstatus() |
| FR14 | Class Diagram | Complaint | Checkcomplaintstatus() |
| FR15 | Class Diagram | ServiceCategory | CreateC |
| FR16 | Class Diagram | Laundrette | ViewReviews, |
| FR17 | Class Diagram | Launderer | ProvidePerformanceReports |

# Human interface design

Every application should follow some standard guidelines so that the user experience can be as smooth as possible, just like that Cloud Laundrette focuses on the ease of the user by providing them with easy to use and navigate UI in which they do not feel like that they are stuck at using some feature.

Few of the guidelines in this regard that needs to be followed are

* The interface should be intuitive.
* The icons used should be self-explanatory that by looking at the icon the user can judge what they would do, so standard icons that are used in the other application should be used.
* The font size, color, and family should not be too overwhelming.
* The UI should be consistent throughout the application.
* The color scheme should match the dark theme of the application.

## Screen images

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 7.2 Screen objects and actions

|  |  |
| --- | --- |
| **Screen object** | **Action** |
|  | This object refers to the launderer dashboard. It include orders, services availed, reviews and feedback etc. By clicking on this button, it will redirect to the clients dashboard |
|  | Launderer can click on this object to to check their profile information and can edit it. |
|  | It refers to the launderer performance. Launderer can create and submit the performance to the admin. |
|  | Launderer can click on this icon to check their orders history, ongoing orders, completed orders and all new orders requests. |
|  |  |
|  | Launderer can check the reviews submitted by the client and can respond to those feedbacks. |
|  | It refers to the services provided to the client by the launderer. Launderer can add, modify and delete services. |
|  | This icon refers to the notification. Launderer can receive different type of notifications like new offers etc. |

## 

# Appendix I

* How to design using UML (OOP): For guidance please follow the instructions mentioned in the link: http://agilemodeling.com/artifacts/
* How and when to design ER diagrams: For guidance please follow the instructions mentioned in the link:

<http://people.inf.elte.hu/nikovits/DB2/Ullman_The_Complete_Book.pdf>

* Data flow diagrams: For guidance please follow the instructions mentioned in the link and book:
  + http://www.agilemodeling.com/artifacts/dataFlowDiagram.htm
  + Software Engineering –A Practitioner’s approach by Roger Pressman
* Architecture diagram: For guidance please follow the instructions mentioned in the link and book:
  + Ian Sommerville – Software Engineering 9th Edition– Chapter 6