



BSIT-F21-009

03-135182-009 Muhammad Abubaker

03-135182-012 Muhammad Saad

Restaurant Hygiene Regulator Application

In partial fulfilment of the requirements for the degree of
Bachelor of Science in Information Technology

Supervisor: Numan Aslam

Department of Computer Sciences
Bahria University, Lahore Campus

June 2022

Certificate



We accept the work contained in the report titled
“Restaurant Hygiene Regulator Application”

Written by
Muhammad Abubaker
Muhammad Saad

As a confirmation to the required standard for the partial fulfilment of the degree of
Bachelor of Science in Information Technology.

Approved by:

Supervisor: Numan Aslam

(Signature)

June 14, 2022

DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Enrolment	Name	Signature
03-135182-009	Muhammad Abubaker	
03-135182-012	Muhammad Saad	

Date : June 14, 2022

Specially dedicated to
My beloved grandmother, mother and father
(Muhammad Abubaker)
My beloved grandmother, mother and father
(Muhammad Saad)

ACKNOWLEDGEMENTS

We would like to thank everyone who had contributed to the successful completion of this project. We would like to express our gratitude to our research supervisor, Mr. Numan Aslam for his invaluable advice, guidance and his enormous patience throughout the development of the research.

In addition, we would also like to express our gratitude to our loving parent and friends who had helped and given encouragement to us.

Muhammad Abubaker

Muhammad Saad

Restaurant Hygiene Regulator Application

ABSTRACT

Hygiene Regulator is a mobile-based centralized platform that allows users to view and write the hygiene rating and review of a restaurant. Health is a very important factor for living life. People often ignore food hygiene because of their busy schedules in new technologies, changing lifestyles, and hectic schedules. This has led to health-related issues in all age groups.

A mobile device is the only thing that everyone is using on daily basis with that routine. Therefore, developing an app that identifies food hygiene in any restaurant is needed of the time. Youth is most vulnerable to health issues since their irregular schedule makes it difficult to keep track of the nutrition of their food intake and manually maintain daily health. With the advancement of time and the environment. There are numerous restaurants that provide healthy food to their customers, but most people are unaware of these restaurants (location).

We propose to develop a user-friendly application which solved this issue. We provide a platform that assists individuals in finding hygienic food spots. They can simply select restaurants from categories by using the location and hygiene value of the restaurant.

TABLE OF CONTENTS

DECLARATION	ii
ACKNOWLEDGEMENTS	iv
ABSTRACT	v
TABLE OF CONTENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF SYMBOLS / ABBREVIATIONS	xii
LIST OF APPENDICES	xiii

CHAPTERS

1	INTRODUCTION	1
1.1	Background	1
1.2	Problem Statements	2
1.3	Aims and Objectives	2
1.4	Scope of Project	3
	1.4.1 Optional Scope	3
2	Software Requirements Specification	4
2.1	Use Case Description	4
2.2	User classes/characteristics	4
2.3	Operating Environment	4
2.4	Other Non-Functional Requirements	5
	2.4.1 Safety Requirement	5
	2.4.2 Security Requirement	5
	2.4.3 Software quality attributes	5
2.5	Admin (U1)	6
2.6	User (U2)	7
2.7	Sign up (U3)	8
2.8	Login (U4)	9
2.9	Location (U5)	10
2.10	Main Category (U6)	11
2.11	Sub Category (U7)	12
2.12	View Restaurant (U8)	13
2.13	Update Restaurant detail (U9)	14
2.14	Feedback (U10)	15
2.15	Get direction. (U11)	16
2.16	Hygiene (U12)	17
2.17	Menu bar (U13)	18
2.18	Search (U14)	19
2.19	Logout (U15)	20

3	DESIGN AND METHODOLOGY	21
3.1	Design	21
3.2	Use Case Diagram	22
3.3	Sequence Diagram	22
3.3.1	Admin	22
3.3.2	User	25
3.4	Domain Model	28
3.5	Collaborative Diagram	29
3.5.1	Admin	29
3.5.2	Role Assign/Controlling/Maintaining	30
3.5.3	User	31
3.5.4	View Restaurant	32
3.5.5	View/Feedback/Comment	33
3.6	Entity Relationship Diagram	34
3.7	Design Class diagram	35
3.8	Data Model	36
3.9	Methodology	37
3.9.1	Develop the overall model:	37
3.9.2	Build the feature list:	37
3.9.3	Plan by feature:	37
3.9.4	Design By future:	38
3.9.5	Build by future:	38
3.9.6	Diagram:	38
3.9.7	Working of Application:	39
4	IMPLEMENTATION	40
4.1	Dataset	40
4.2	Languages used for Implementation	40
4.2.1	Dart	40
4.3	Framework	41
4.3.1	Flutter	41
4.3.2	Adobe XD	41
4.3.3	Android studio	41

4.4	Methodology	42
4.4.1	FDD	42
4.4.2	Diagram:	42
4.5	Implementation	43
5	USER MANUAL	44
5.1	Welcoming Screen	44
5.2	Login & Sign up Screen	45
5.3	Profile Screen	46
5.4	Home Screen	47
5.5	Categories Screen	48
5.6	Specific Restaurant Screen	49
5.7	Hygiene Reviews Screen	50
6	CONCLUSION AND RECOMMENDATIONS	51
6.1	Conclusion	51
	REFERENCES	52
	APPENDICES	53

LIST OF TABLES

TABLE	TITLE	PAGE
TABLE 2. 1:	ADMIN-U1	6
TABLE 2. 2:	USER-U2	7
TABLE 2. 3:	SIGN UP-U3	8
TABLE 2. 4:	LOGIN -U4	9
TABLE 2. 5:	LOCATION -U5	10
TABLE 2. 6:	MAIN CATEGORY -U6	11
TABLE 2. 7:	SUB CATEGORY -U7	12
TABLE 2. 8:	VIEW RESTAURANT -U8	13
TABLE 2. 9:	UPDATE RESTAURANT DETAIL -U9	14
TABLE 2. 10:	FEEDBACK -U10	15
TABLE 2. 11:	GET DIRECTION-U11	16
TABLE 2. 12:	HYGIENE-U12	17
TABLE 2. 13:	MENU BAR-U13	18
TABLE 2. 14:	SEARCH -U14	19
TABLE 2. 15:	LOGOUT -U15	20

LIST OF FIGURES

FIGURE	TITLE	PAGE
FIGURE 3. 1:	USE CASE DIAGRAM	22
FIGURE 3. 2:	LOGIN	23
FIGURE 3. 3:	ADD/REMOVE/UPDATE/EDIT/RESTAURANT	24
FIGURE 3. 4:	CREATE ACCOUNT/SIGNUP	25
FIGURE 3. 5:	LOGIN	26
FIGURE 3. 6:	VIEW/FEEDBACK/RATING/COMMENT	27
FIGURE 3. 7:	DOMAIN MODEL	28
FIGURE 3. 8:	ADMIN.....	29
FIGURE 3. 9:	ROLE ASSIGN/CONTROLLING/MAINTAINING	30
FIGURE 3. 10:	USER	31
FIGURE 3. 11:	VIEW RESTAURANT	32
FIGURE 3. 12:	VIEW/FEEDBACK/COMMENT	33
FIGURE 3. 13:	ERD.....	34
FIGURE 3. 14:	CLASS DIAGRAM.....	35
FIGURE 3. 15:	DATA MODEL.....	36
FIGURE 5. 1	WELCOMING SCREEN	44
FIGURE 5. 2:	LOGIN & SIGN UP SCREEN	45
FIGURE 5. 3:	PROFILE SCREEN	46
FIGURE 5. 4:	HOME SCREEN	47
FIGURE 5. 5:	CATEGORIES SCREEN.....	48
FIGURE 5. 6:	SPECIFIC RESTAURANT SCREEN	49
FIGURE 5. 7:	HYGIENE REVIEWS SCREEN	50

LIST OF SYMBOLS / ABBREVIATIONS

AFL	Flutter
FB	Firebase
DT	Dart
SRS	Software Requirements Specification
ERD	Entity-Relationship Diagram
AR	Architectural Diagram
UML	Unified Modelling Language
JSON	JavaScript Object Notation
DM	Domain Model
SD	Sequence Diagram
FDD	Feature-Driven Development
APK	Android Application Package
IOS	IPhone Operating System

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
	APPENDICES 1: MAIN SCREEN:	53
	APPENDICES 2: HOME SCREEN:.....	53
	APPENDICES 3: LOGIN SCREEN:.....	54
	APPENDICES 4: CATEGORIES:.....	54
	APPENDICES 5: ADMIN.....	55

CHAPTER 1

INTRODUCTION

1.1 Background

Mobile applications are becoming increasingly important in our daily lives. Mobile phones and technologies have grown so ingrained in our everyday routines that it is hard to even imagine life without them. We commonly see people searching the internet for the best hygiene restaurants in Pakistan, and then browsing websites to select their preferred hygiene restaurant in a given place. The proposed mobile application is based on hygiene ratings of restaurants across Pakistan, based on different techniques and skills. Because there is no such platform for the people to make sure whether the food, they are eating is hygienic or not? Here we will provide an application that will allow the users to go through different restaurants to check their rating and hygiene level. It is planned to make the user interface design more user-friendly.

The range of mobile app development has broadened so an application for the Hygiene ratings will be developed to solve this problem for the general public. This Hygenator application will assist them in managing their Hygiene food restaurant (ISO 22000, 14001).[1] The difficulties stated above largely affect both indigenous and foreign residents of Pakistan. To achieving this goal, we used a variety of streamlining processes and procedures.

1.2 Problem Statements

Nowadays, there are numerous problems with hygienic food and places. People are unable to locate a healthy eating place in a pleasant environment. Daily, we read many headlines about restaurants being banned for not providing hygienic food and places to their clients. Even the PFA (Pakistan Food Authority) is not providing us with the hygiene ratings and the food ratings of restaurants across the country. [2]

But why is it necessary to develop this application? Here in Pakistan, there is no platform available, which tells you hygienic information regarding food. As a result, we can solve this problem by developing an application to help users pick food that is suited for their health and to assist them in selecting their favorite restaurant. This application will be developed in Flutter and we will be creating its UI in Adobe XD.

1.3 Aims and Objectives

The following are the thesis objectives:

- i. To allow users to check the hygiene rating of the restaurant.
- ii. To allow users to find a good healthy restaurant nearby them.
- iii. To allow users to give a suggestion/rating reading the restaurant.
- iv. Providing an IOS and Android-based platform.
- v. Allow restaurants to hygiene inspection.
- vi. To allow users to give orders by using the application.

1.4 Scope of Project

Our application will allow its users to look into the rating and decide where to eat near them. The main purpose of this application is to inform our people/users which restaurants are providing a healthy environment and food. We will create a user-friendly product that is dependent on usability. The Usability concept is "the extent to which a product can be used by specified users to achieve specific goals with effectiveness, efficiency, and satisfaction in a specified context of use." The beneficiaries to this project will be the public and restaurants with healthy environments.

1.4.1 Optional Scope

- If restaurants want to be listed on our app for their healthy atmosphere, an inspections team module will be established.
- A separate screen/module for video advertisements can be set up for a certain restaurant.
- On-demand restaurants can also add online food delivery services to our application.

CHAPTER 2

Software Requirements Specification

2.1 Use Case Description

A use case statement is written in the form of a representation how a user will perform his/her task on our application. It shows from a user side point of view, a system application behaviour as it responds to a request.

2.2 User classes/characteristics

In the Hygenator application we have two classes/characteristics:

- i. Admin
- ii. User

2.3 Operating Environment

The Hygenator platform we develop is a flutter, and the flowing software is used.

Name	Description
Operating system	Windows 10.
Language	Dart.
Tool	Android Studio, Visual studio.

2.4 Other Non-Functional Requirements

The non-functional requirement follows are as:

2.4.1 Safety Requirement

In The developing stage we faced some issues:

- i. Data storage well maintained
- ii. Backup
- iii. Schedule to implement
- iv. Reliability
- v. Requirement well defined
- vi. Meet the requirement
- vii. End-user testing

2.4.2 Security Requirement

The application contains the following functions and elements are below:

- i. Confidentiality
- ii. Integrity
- iii. Availability
- iv. Audit
- v. Authentication /authorization
- vi. Network
- vii. Data security

2.4.3 Software quality attributes

Following quality attributes that will give to be considered while designing and developing the Hygenator regulator application are below:

- i. The application is easily editable
- ii. The following application we developed according to a universal principle.
- iii. The application is easy to use for everyone due to its interface. However, the user must have a basic knowledge of using mobile.

2.5 Admin (U1)

Name	Admin	
Use Case ID	U1	
Priority	High	
Primary Actor	Admin	
Other Participating Actor	Editor	
Description	This Use Case explains the process of the relevant actor who's controlling and maintaining the Hygenator application.	
Trigger	This use case initiates the Backside of the application.	
Typical Flow of Event	User Action	System Response
	Admin can maintain the database.	The system can maintain the User information and restaurant details are saved.
	Admin can enter data.	Details are shown.
	Admin controls the flow.	Working on the application.
	Admin gives the role.	The system will access the user role (Editor, Data Entry, etc.)
Post condition	Controlling and maintaining.	

Table 2. 1: Admin-U1

2.6 User (U2)

Name	User	
Use Case ID	U2	
Priority	High	
Primary Actor	User	
Other Participating Actor	None	
Description	This Use Case describes the process of a relevant actor who can use the application.	
Precondition	Part of Hygenator family /Guest.	
Trigger	This use case initiates the use of the application.	
Typical Flow of Event	User Action	System Response
	Users can create an account or as a guest.	The system can show the relevant profile.
	Users click on login, signup, or as a guest.	Screens are shown.
Alternative Flow of Events	User login / Guest.	
Post condition	Login.	
Alternative Post condition	Unsuccessful / Guest	

Table 2. 2: User-U2

2.7 Sign up (U3)

Name	Sign up	
Use Case ID	U3	
Priority	Medium	
Primary Actor	User	
Other Participating Actor	None	
Description	This Use Case describes the process of relevant actors Signing up.	
Precondition	The user installs the application.	
Trigger	This use case is initiated when clicking on the signup/Register button.	
Typical Flow of Event	User Action	System Response
	Users can create an account.	Sign up/Register screen appears.
	The user enters the personal data for registered the sign-up.	The system will show the information is successful or not.
	The user will be registered.	The system shows the profile.
Alternative Flow of Events	User register / failed.	
Post condition	Register successfully.	
Alternative Post condition	Register Unsuccessful.	

Table 2. 3: Sign up-U3

2.8 Login (U4)

Name	Login	
Use Case ID	U4	
Priority	Medium	
Primary Actor	User	
Other Participating Actor	None	
Description	This Use Case describes the process of relevant actor Logging in.	
Precondition	Users must register/sign up for the application.	
Trigger	This use case is initiated when clicking on the Login button.	
Typical Flow of Event	User Action	System Response
	The user enters the log in detail.	The system displays a profile.
	The user clicks on the login button.	The screen appears to the next user portal.
Alternative Flow of Events	User Login / Login failed.	
Post condition	Login was successful.	
Alternative Post condition	Login Unsuccessful.	

Table 2. 4: Login –U4

2.9 Location (U5)

Name	Location.	
Use Case ID	U5	
Priority	Medium	
Primary Actor	Location define.	
Other Participating Actor	None	
Description	This Use Case describes the Location of the relevant actor whose Logging in.	
Precondition	The user must log in.	
Trigger	This use case initiates after clicking on the Login button.	
Typical Flow of Event	User Action	System Response
	A user enters the Location after login.	The system set the location.
	Users can set the location according to their needs.	The system will filter the location.
Alternative Flow of Events	Location on/ Location off	
Post condition	Location on successful.	
Alternative Post condition	Location off.	

Table 2. 5: Location –U5

2.10 Main Category (U6)

Name	Main Category.	
Use Case ID	U6	
Priority	High	
Primary Actor	Maintain the Restaurant Category type.	
Other Participating Actor	None	
Description	This Use Case describes the Main Category of a restaurant.	
Precondition	Internet must be connected.	
Trigger	This use case initiate when clicking on Category.	
Typical Flow of Event	User Action	System Response
	Admin set the main Category of the restaurant type. (Eating/Drinking)	The system will appear in the relevant restaurant list.
	Users can filter the Category.	The system will filter the Category. (Eating/Drinking)
Alternative Flow of Events	None.	
Post condition	Select the Category.	
Alternative Post condition	Unselected.	

Table 2. 6: Main Category –U6

2.11 Sub Category (U7)

Name	Sub Category.	
Use Case ID	U7	
Priority	High	
Primary Actor	Maintain the restaurant Sub Category inside the Main Category.	
Other Participating Actor	None	
Description	This Use Case describes the after the Main Category of a restaurant.	
Precondition	Inside the main Category.	
Trigger	This use case initiate when entering inside the main Category.	
Typical Flow of Event	User Action	System Response
	Admin set the Sub Category of the restaurant type inside the main Category.	The system will appear in the relevant restaurant list.
	Users can filter the sub Category.	The system will filter the Category. (taste)
Alternative Flow of Events	None.	
Post condition	Select the Category.	
Alternative Post condition	Unselected.	

Table 2. 7: Sub Category –U7

2.12 View Restaurant (U8)

Name	View Restaurant.	
Use Case ID	U8	
Priority	Medium	
Primary Actor	Administrator.	
Other Participating Actor	None	
Description	This Use Case describe the after the hygiene rating and detail of restaurant.	
Precondition	View restaurant.	
Trigger	This use case initiate when enter inside the restaurant detail.	
Typical Flow of Event	User Action	System Response
	Admin maintain the detail of the restaurant.	System will show the hygiene rating of the restaurant.
	User can view the hygiene rating and detail of the restaurant.	System will shows the detail.
Alternative Flow of Events	None.	
Post condition	Click on the restaurant.	
Alternative Post condition	Unclick able.	

Table 2. 8: View Restaurant –U8

2.13 Update Restaurant detail (U9)

Name	Update Restaurant Detail.	
Use Case ID	U9	
Priority	Medium	
Primary Actor	Administrator.	
Other Participating Actor	None	
Description	This Use Case describe process of a relevant actor updating the restaurant detail.	
Precondition	Restaurant must be added.	
Trigger	This use case initiate when the administrator update the detail.	
Typical Flow of Event	User Action	System Response
	Administrator will be able to update and changing the restaurant information.	System will update the database.
Alternative Flow of Events	None.	
Post condition	Detail will be updated.	
Alternative Post condition	No action performed.	

Table 2. 9: Update Restaurant detail –U9

2.14 Feedback (U10)

Name	Feedback.	
Use Case ID	U10	
Priority	High	
Primary Actor	Administrator	
Other Participating Actor	User.	
Description	This Use Case describe process of a relevant actor when administrator maintain the database of the feedback of restaurant hygiene rating.	
Precondition	Restaurant must be added.	
Trigger	This use case initiate when the administrator/user write the feedback	
Typical Flow of Event	User Action	System Response
	Administrator will be able to write the restaurant hygiene information.	System will update the database.
	User can write the feedback after login the profile	System will show the feedback.
Alternative Flow of Events	None.	
Post condition	Write the feedback.	
Alternative Post condition	Not found.	

Table 2. 10: Feedback –U10

2.15 Get direction. (U11)

Name	Get direction.	
Use Case ID	U11	
Priority	Medium.	
Primary Actor	User.	
Other Participating Actor	None.	
Description	This Use Case describe process of a relevant actor when user click on the direction.	
Precondition	Restaurant detail must be open.	
Trigger	This use case initiate when the user click the direction button.	
Typical Flow of Event	User Action	System Response
	User will able to check the location of the restaurant.	System will shows the location.
Alternative Flow of Events	None.	
Post condition	Get direction.	
Alternative Post condition	Not found.	

Table 2. 11: Get direction–U11

2.16 Hygiene (U12)

Name	Hygiene	
Use Case ID	U12	
Priority	High	
Primary Actor	User	
Other Participating Actor	none	
Description	This Use Case describe process of a relevant actor when user open a restaurant detail to check the hygiene level of the restaurant.	
Precondition	Restaurant must be added.	
Trigger	This use case initiate when the user check the restaurant detail button.	
Typical Flow of Event	User Action	System Response
	User will able to check the Hygiene level of the restaurant.	System will shows the hygiene level in restaurant detail.
Alternative Flow of Events	None.	
Post condition	Hygiene level/rating shown.	
Alternative Post condition	None.	

Table 2. 12: Hygiene–U12

2.17 Menu bar (U13)

Name	Menu bar	
Use Case ID	U13	
Priority	Medium	
Primary Actor	User	
Other Participating Actor	Administrator	
Description	This Use Case describe process of a relevant actor when user click on the menu bar button of the application.	
Precondition	Application must be run.	
Trigger	This use case initiate when the user click on menu bar button.	
Typical Flow of Event	User Action	System Response
	User will able to click on the menu bar button.	System will shows the menu bar of the application.
Alternative Flow of Events	Main screen option.	
Post condition	Menu bar information display.	
Alternative Post condition	Not click.	

Table 2. 13: Menu bar–U13

2.18 Search (U14)

Name	Search	
Use Case ID	U14	
Priority	Medium	
Primary Actor	User	
Other Participating Actor	None	
Description	This Use Case describe process of a relevant actor when user click on the search option of the application to search the restaurant.	
Precondition	Application must be run.	
Trigger	This use case initiate when the user search.	
Typical Flow of Event	User Action	System Response
	User will able to click on the search option.	System will shows the restaurant search result.
Alternative Flow of Events	Main screen option.	
Post condition	Search the restaurant.	
Alternative Post condition	None.	

Table 2. 14: Search –U14

2.19 Logout (U15)

Name	Logout	
Use Case ID	U15	
Priority	Low	
Primary Actor	Administrator/User	
Other Participating Actor	None	
Description	This Use Case describe process of a relevant actor when user click on the Logout button of the currently using account.	
Precondition	User must be login.	
Trigger	This use case initiate when the user click on Logout button.	
Typical Flow of Event	User Action	System Response
	User will able to click on the Logout button.	System generate a confirmation dialogue box.
	User click on confirm button.	System end the user session and logout the user.
Alternative Flow of Events	None.	
Post condition	User logout successfully.	
Alternative Post condition	None.	

Table 2. 15: Logout –U15

CHAPTER 3

DESIGN AND METHODOLOGY

3.1 Design

This chapter provides an overview of the application and its design. The system architecture design provides a comprehensive view of the system. Developers and clients will be able to see and check the design plan in greater detail as a result of this.

- i. Use Case Diagram
- ii. Sequence Diagram
- iii. Domain Model
- iv. Collaborative Diagram
- v. Entity Relationship Diagram
- vi. Design class diagram
- vii. Data Model

3.2 Use Case Diagram

Use case diagram of the whole application system are given below:

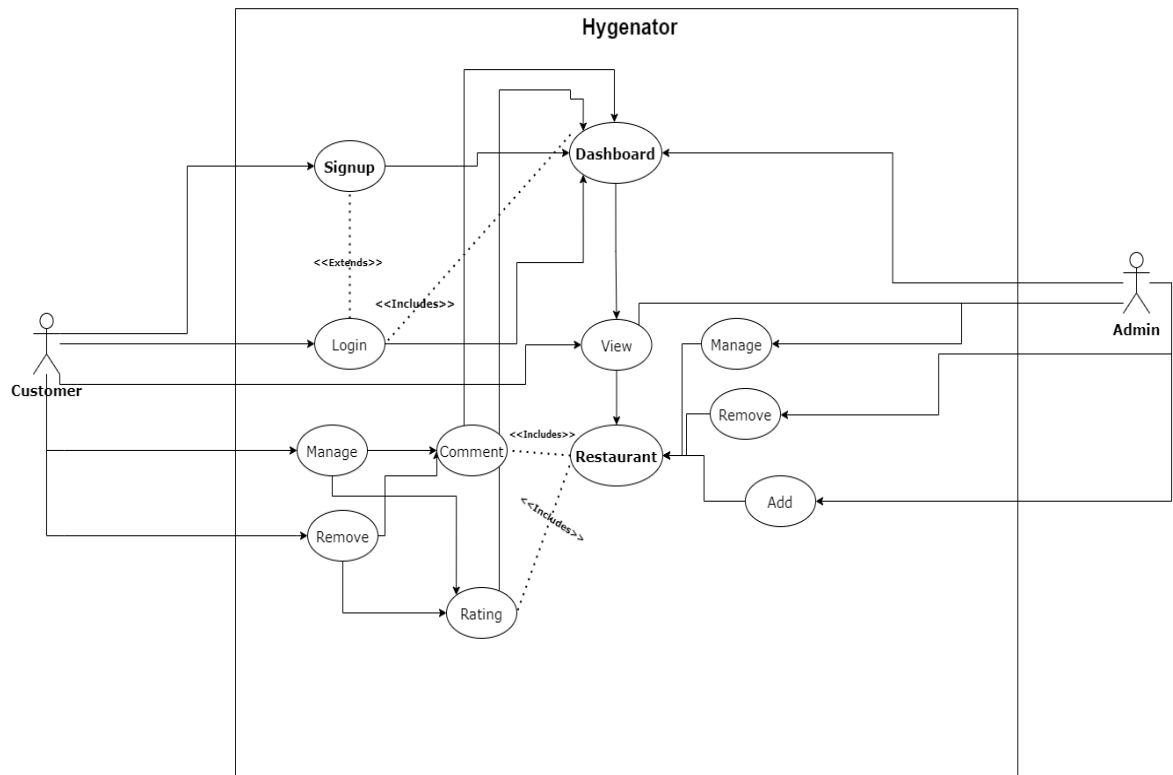


Figure 3. 1: Use Case Diagram

3.3 Sequence Diagram

We divide Sequence diagram into two categories the following are bellow.

3.3.1 Admin

The Admin Sequence diagram are following.

3.3.1.1 Login

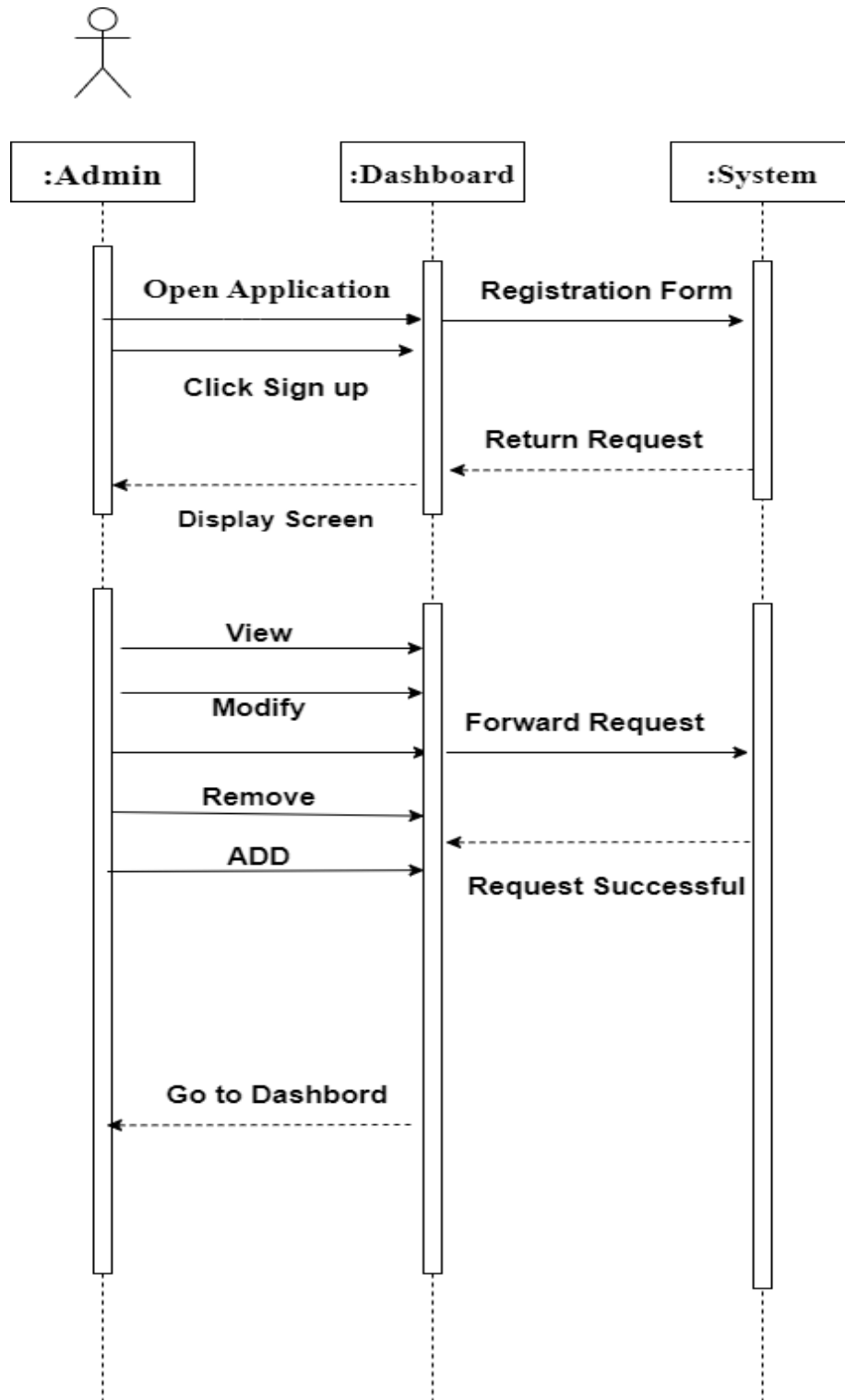


Figure 3. 2: Login

3.3.1.2 Add/Remove/Update/View/Rating and comment

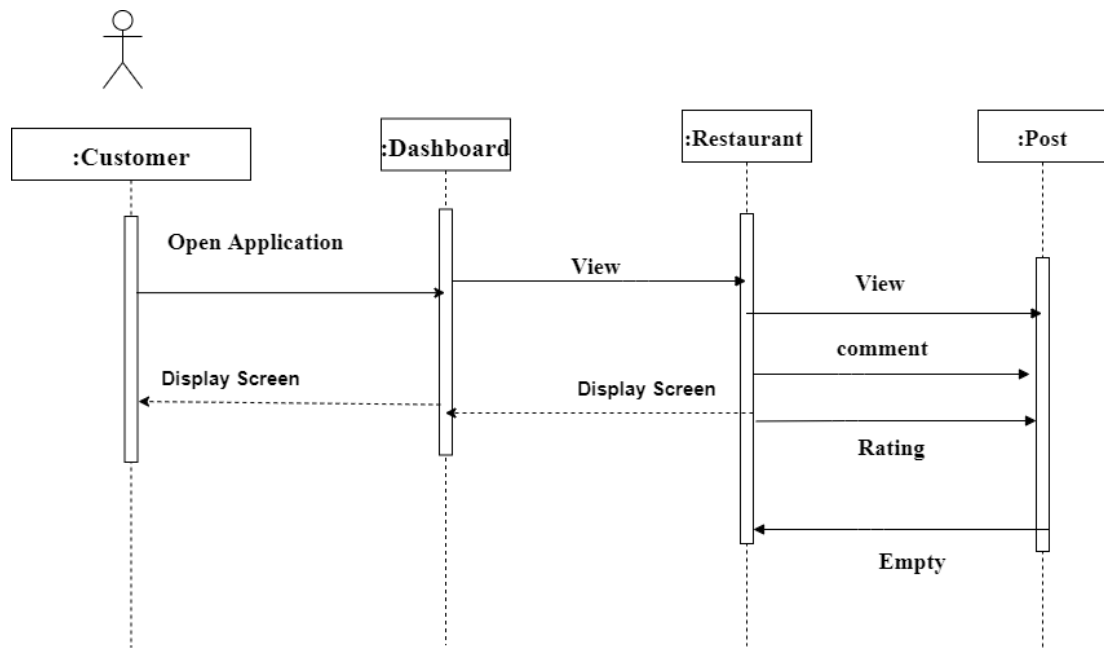


Figure 3. 3: Add/remove/update/edit/restaurant

3.3.2 User

The user sequence diagram are following.

3.3.2.1 Signup

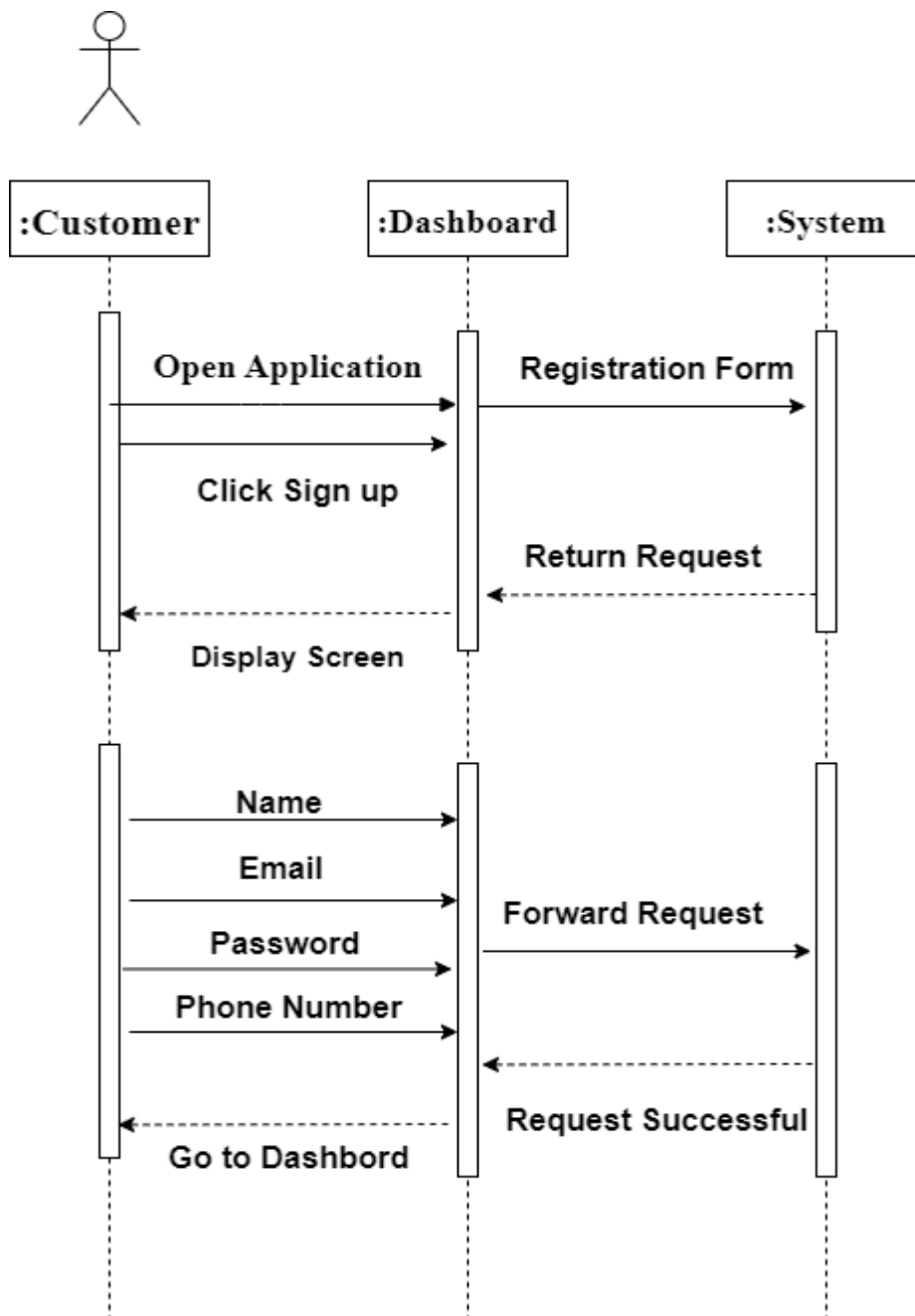


Figure 3. 4: Create account/Signup

3.3.2.2 Login

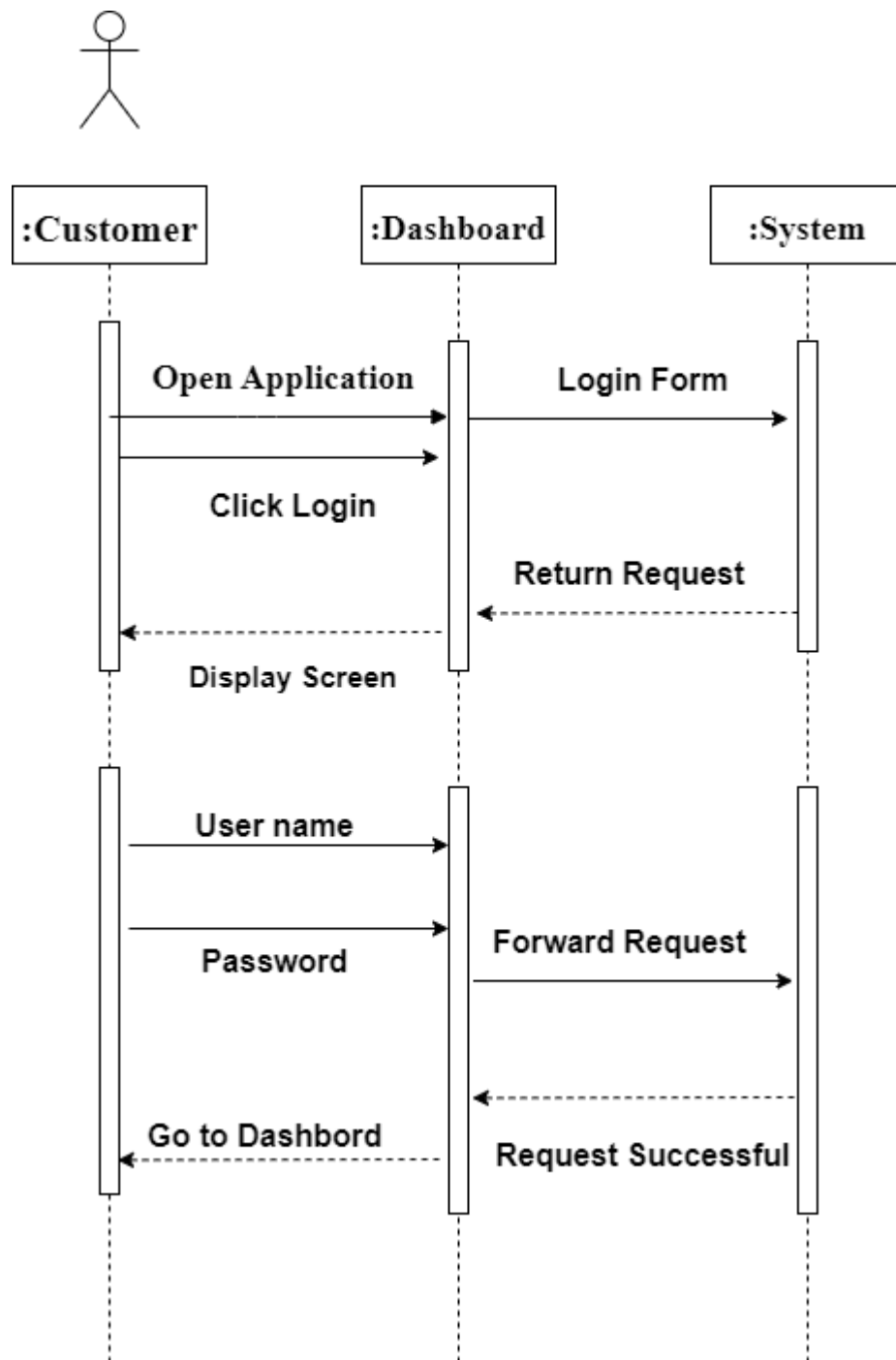


Figure 3. 5: Login

3.3.2.3 View/Feedback/Rating/Comment

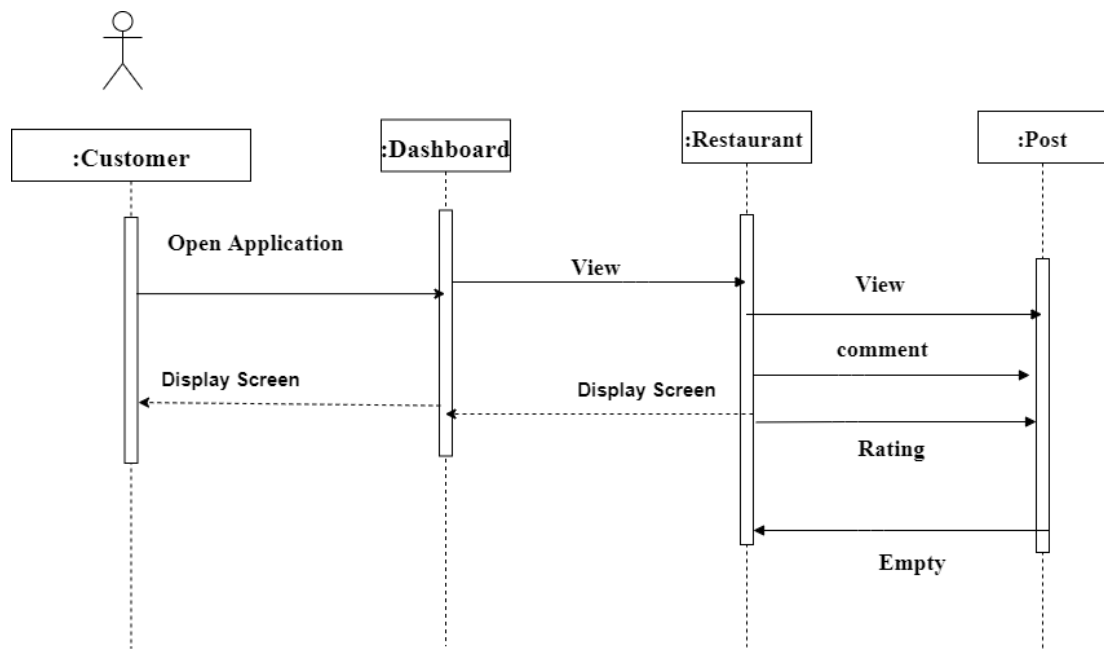


Figure 3. 6: View/Feedback/Rating/Comment

3.4 Domain Model

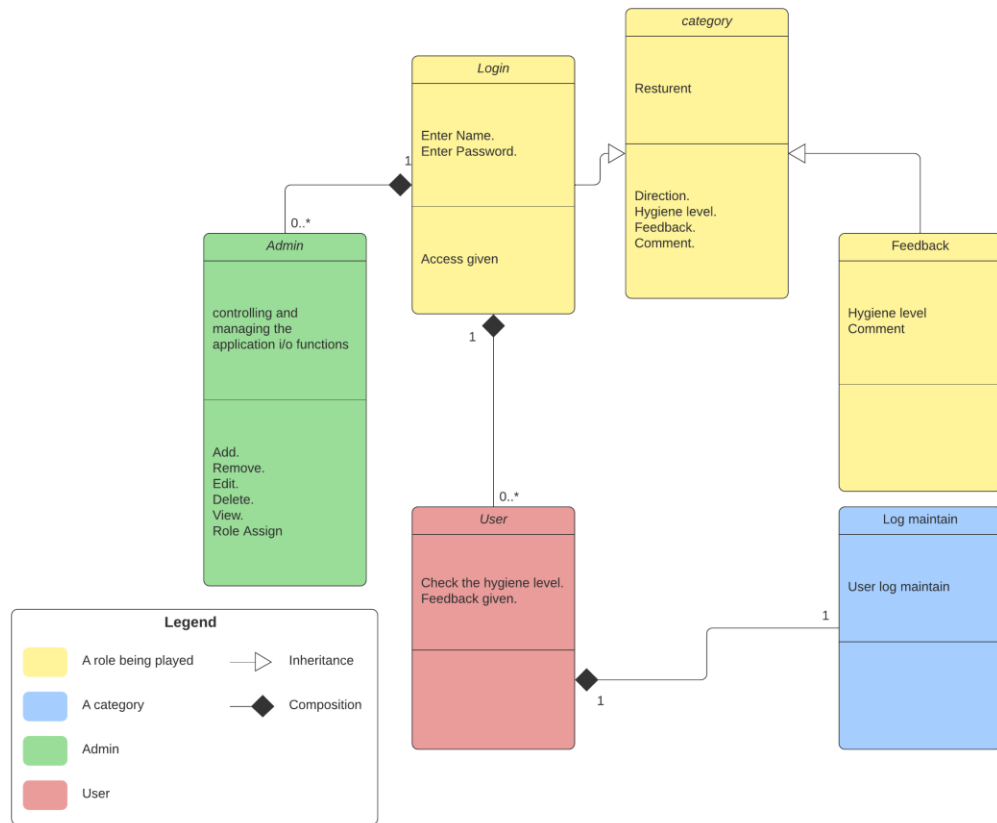


Figure 3. 7: Domain Model

3.5 Collaborative Diagram

The collaborative diagram is as follows.

3.5.1 Admin

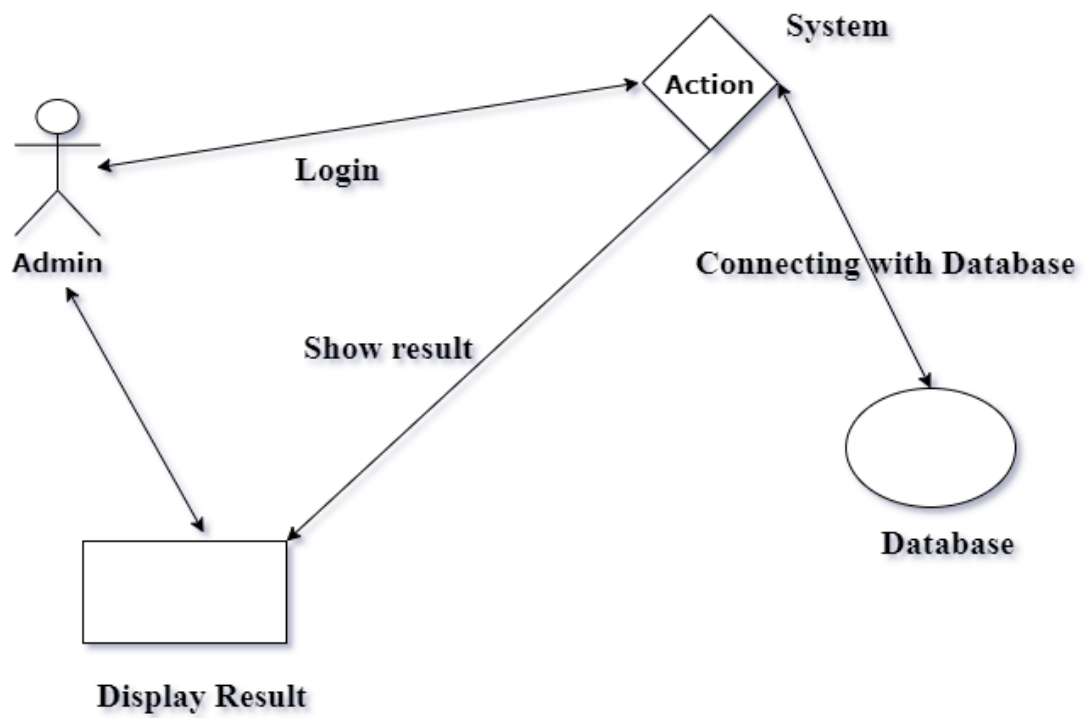


Figure 3. 8: Admin

3.5.2 Role Assign/Controlling/Maintaining

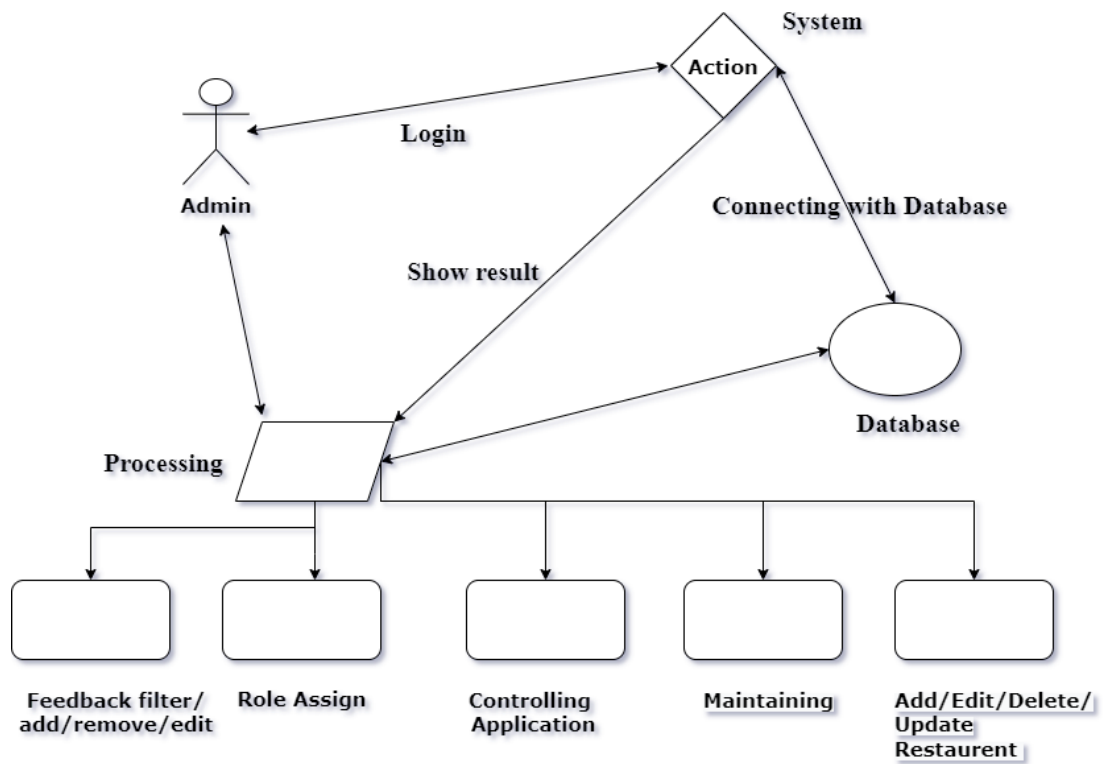


Figure 3. 9: Role Assign/Controlling/Maintaining

3.5.3 User

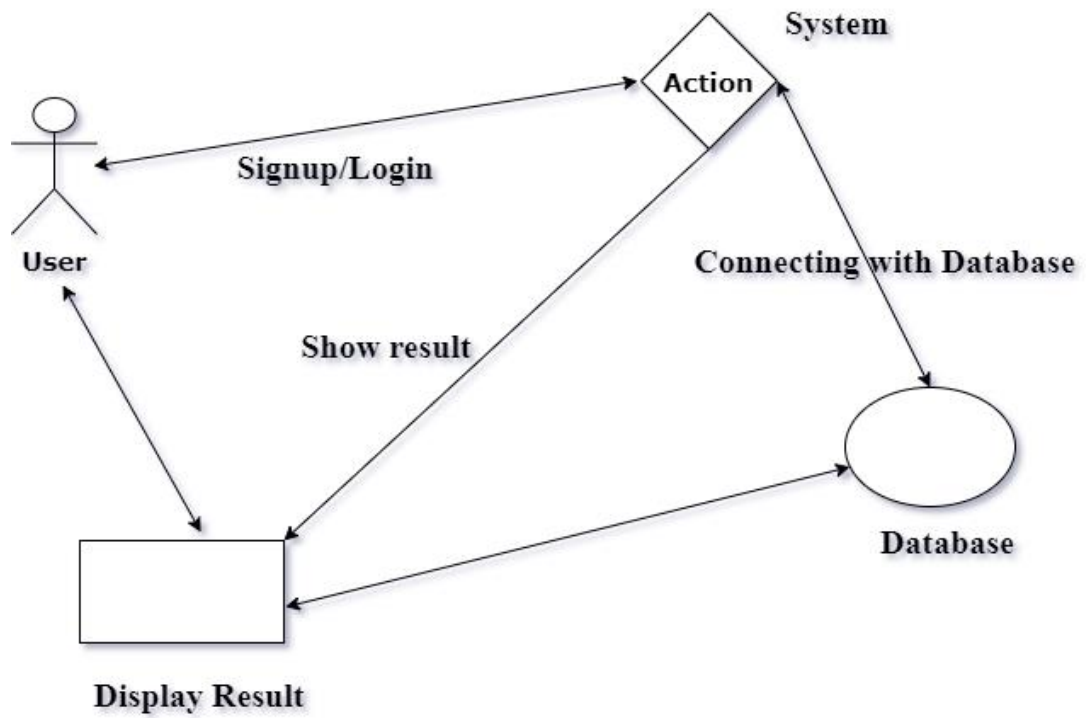


Figure 3. 10: User

3.5.4 View Restaurant

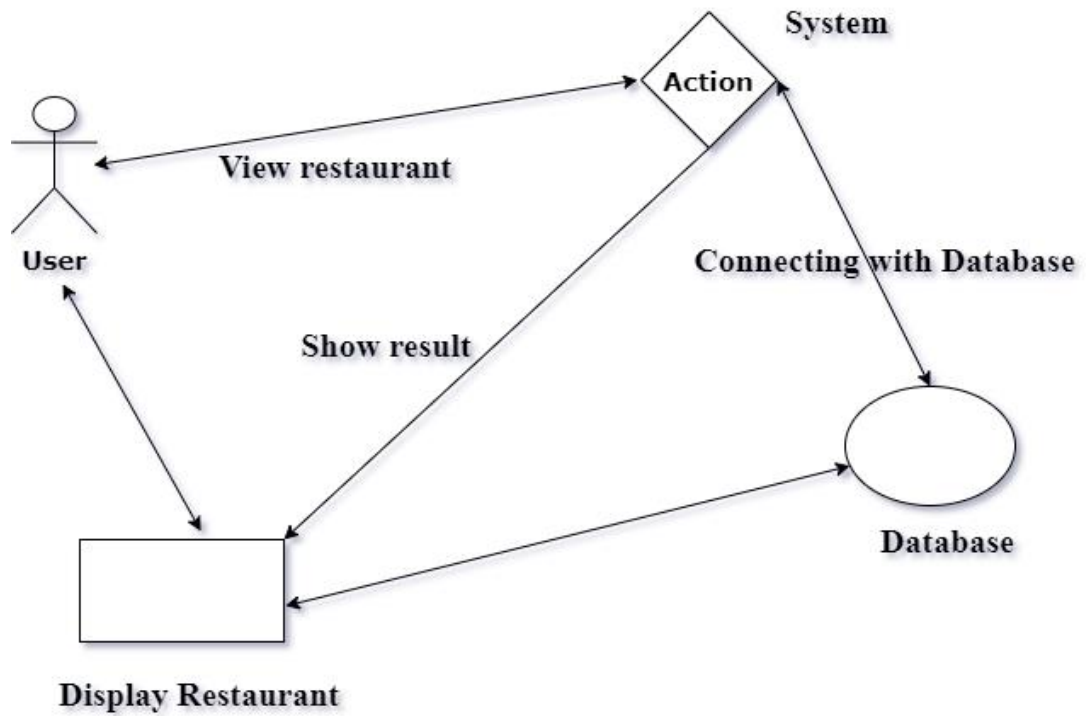


Figure 3. 11: View Restaurant

3.5.5 View/Feedback/Comment

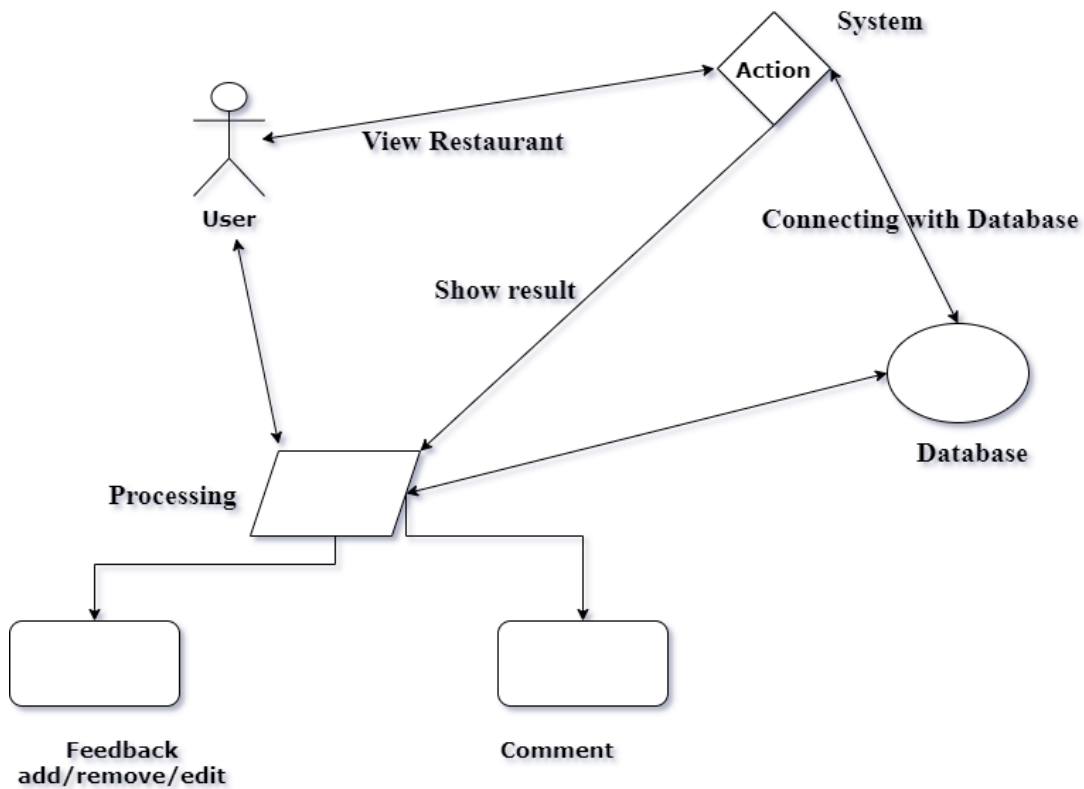


Figure 3. 12: View/Feedback/Comment

3.6 Entity Relationship Diagram

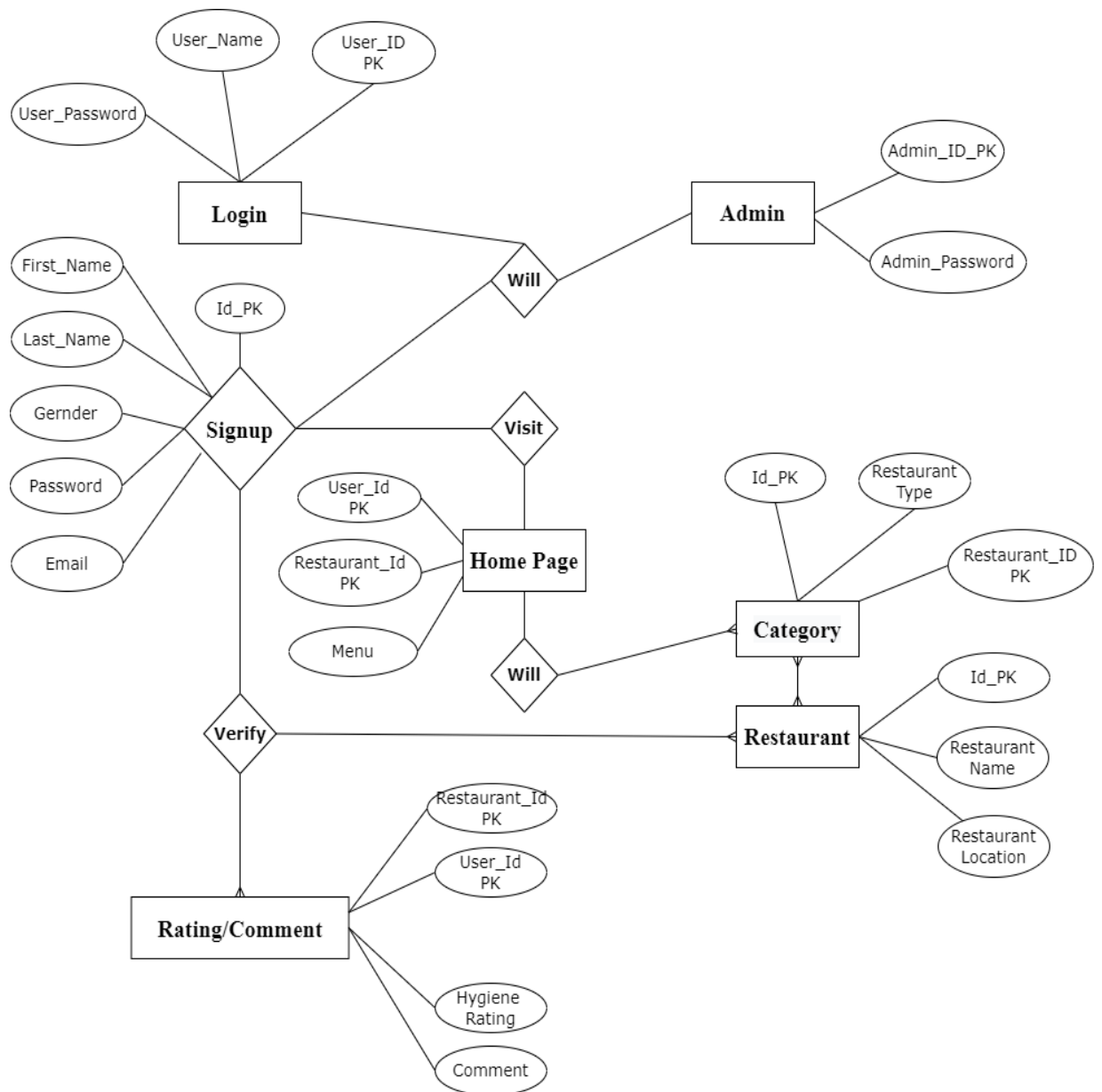


Figure 3. 13: ERD

3.7 Design Class diagram

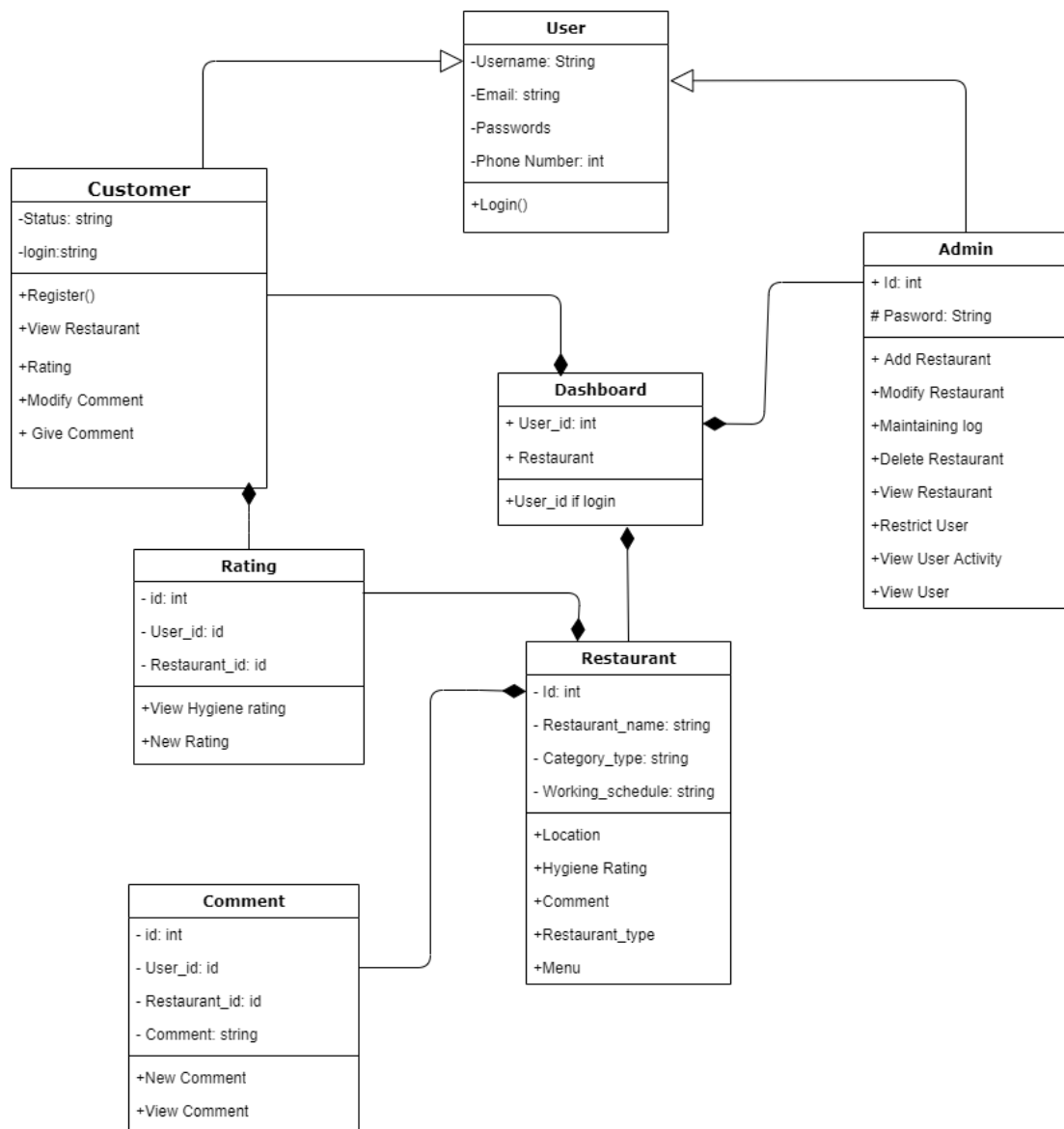


Figure 3. 14: Class diagram

3.8 Data Model

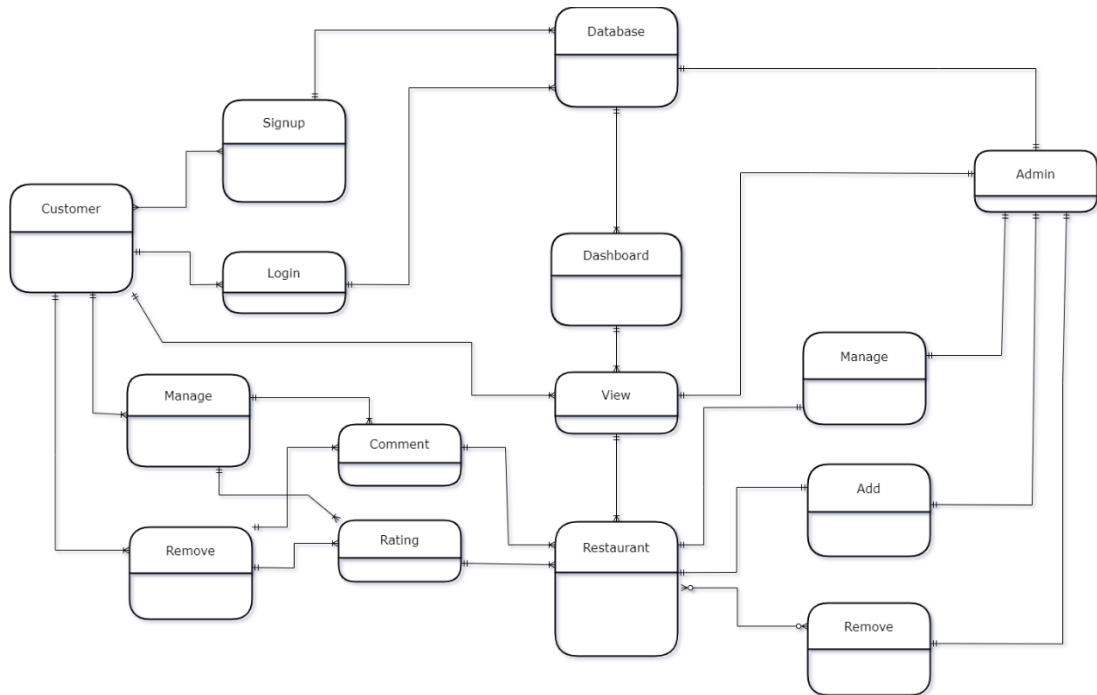


Figure 3. 15: Data Model

3.9 Methodology

The methodology we used is **Agile** and the framework is **FDD (Feature-Driven Development)**. In this methodology, the framework focuses on building functional software with features that fulfil client expectations, as the names suggest. Under the Agile Manifesto's ideals and principles, FDD attempts to ensure consistent and on-time delivery to consumers.[3]

Feature-Driven Development consists of five significant developments:

- Develop the overall model
- Build the features list
- Plan by feature
- Design by future
- Build by future

Furthermore, explain these five major developments.

3.9.1 Develop the overall model:

In the FDD team we determine the overall scope and project consequence to take a decision that develops an overall application.

3.9.2 Build the feature list:

The team members will map out the customer-focused features that will be developed. They will be minor tasks that can be accomplished in a short amount of time.

3.9.3 Plan by feature:

The team will assess the individual elements of the list and arrange them in the correct order.

3.9.4 Design By future:

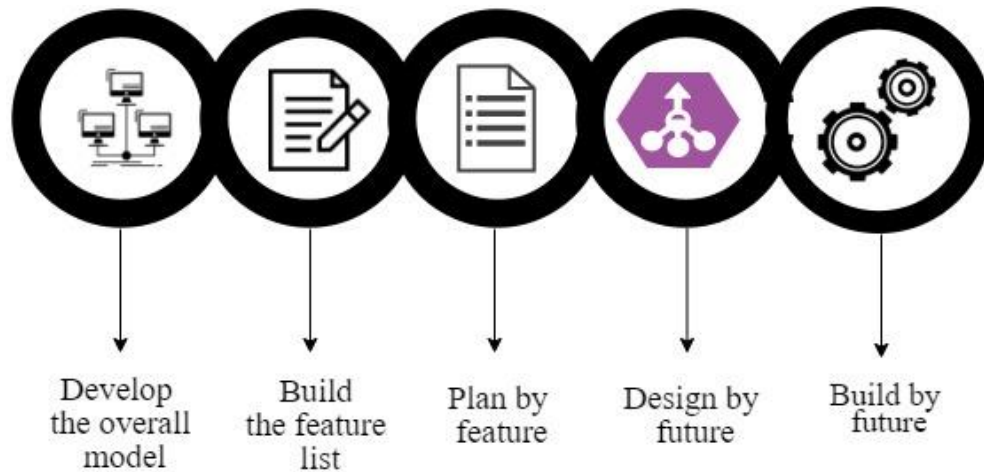
At this point, the team programmer will decide which features to create over the next two weeks. Before construction begins, a design package for each feature will be developed, and team members will undertake a review

3.9.5 Build by future:

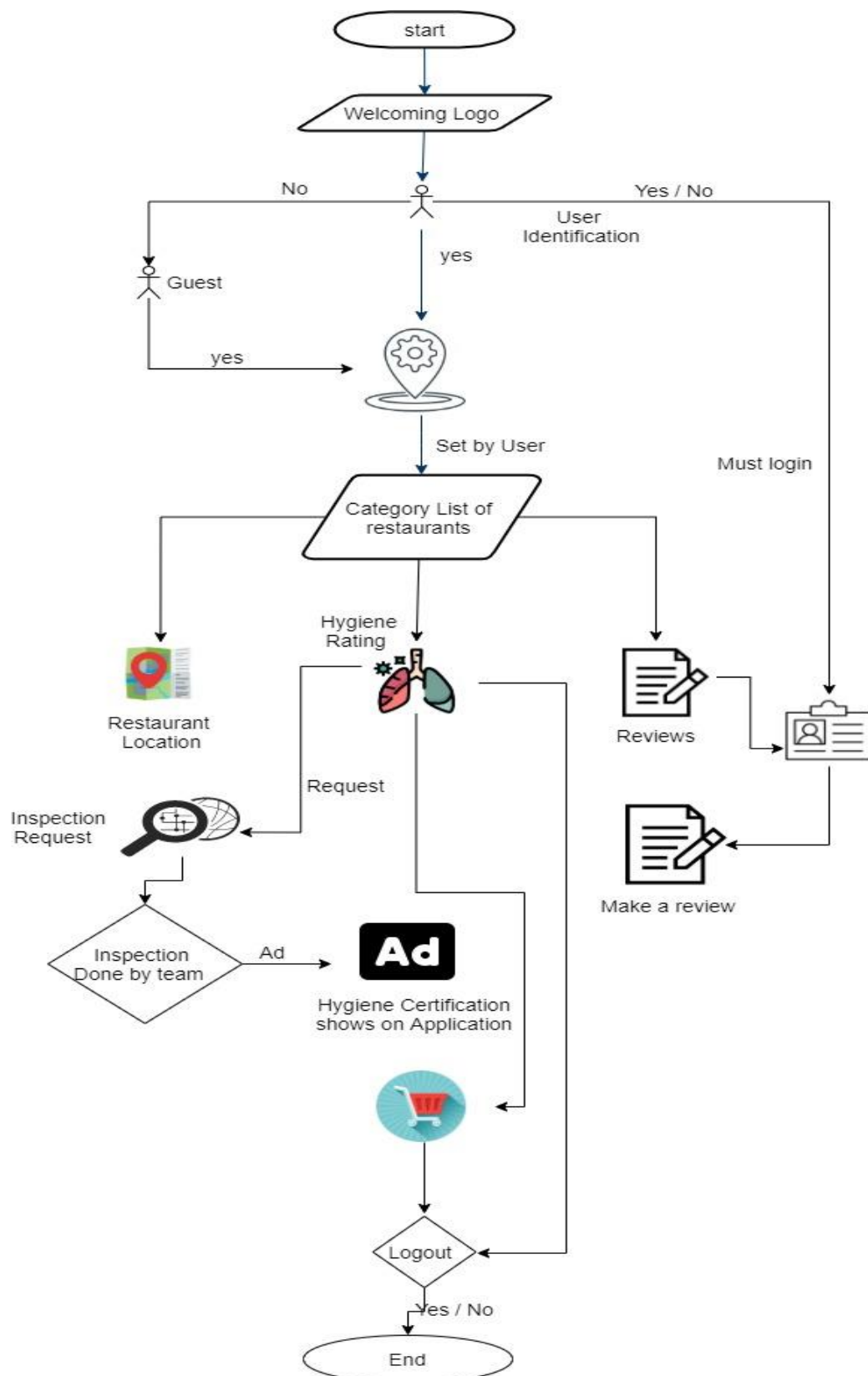
Developers are hard at work creating the code for the aforementioned functionalities. Before the final version is developed, this code will be tested.

3.9.6 Diagram:

In this diagram, we shows the working how FDD works:[4]



3.9.7 Working of Application:



In the above Flowchart, we shows how our application worked with the Methodology of FDD.[4]

CHAPTER 4

IMPLEMENTATION

4.1 Dataset

The Dataset is to fulfil our review suggestions and train the Hygenator application, we require a dataset with records of different people of varied ages and tastes. The dataset that we received contains data that will be used as dummy data when we run the application at this time. After that, we launch the application as a beta version to collect user feedback, and on the backend, we create a database to store the data from the reviews and restaurants.

4.2 Languages used for Implementation

The following languages were used to implement the application:

4.2.1 Dart

Dart is a client-oriented programming language that lets you build apps quickly for any platform. Dart is comparable to flutter in that it aims to be the most productive programming language for multi-platform application development, a language built for quick programming on any platform.

Dart features a large set of core libraries that provide essentials with a wide range of functionality for a variety of platforms.[5]

4.3 Framework

4.3.1 Flutter

Flutter is a Google open-source framework for creating multi-platform native apps from a single codebase. The Flutter supports several platforms, which allowing developers to create two distinct versions for IOS and Android. Deploy to various platforms from a single codebase, including mobile, web, desktop, and embedded devices.[6]

4.3.2 Adobe XD

It was designed from the ground up with performance in mind. Adobe XD allows users to create prototypes that feel and look real, allowing them to effectively convey their design concept while keeping their team on track. It's a simple and powerful vector-based experience design platform that gives teams the tools and resources they need to work build the best experiences in the world.[7]

4.3.3 Android studio

Android Studio, which is based on IntelliJ IDEA, is the official Integrated Development Environment (IDE) for Android app development. Android Studio, in addition to IntelliJ's strong code editor and developer tools, provides additional capabilities that improve efficiency when developing Android apps, such as:[8]

1. A build system that is customizable and based on Gradle.
2. A feature-rich and speedy emulator.
3. Constructed on the Google Cloud Platform and many more.

4.4 Methodology

4.4.1 FDD

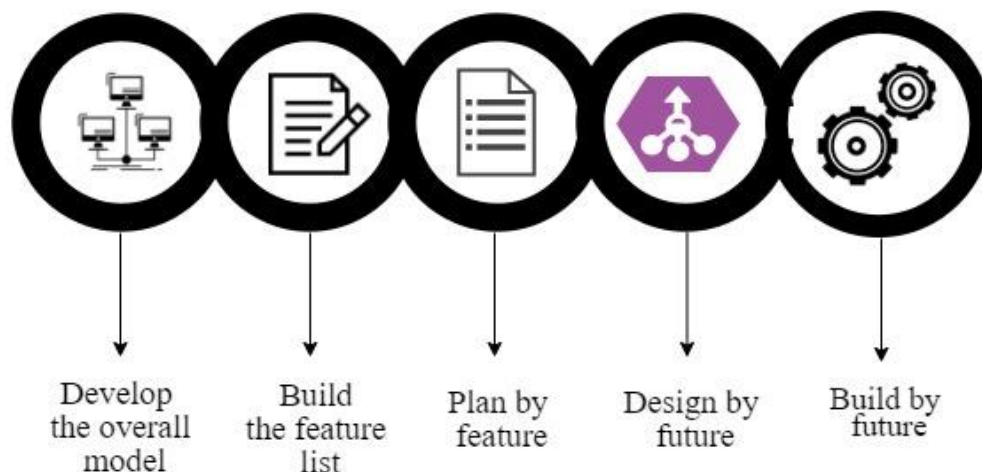
The methodology we will be using is Agile and the framework will be FDD (Feature-Driven Development). It focusses on building functional software with features that fulfil client expectations, as the names suggest. In accordance with the Agile Manifesto's ideals and principles, FDD attempts to ensure consistent and on-time delivery to consumers.[9]

Feature-Driven Development consists of five significant developments::

1. Develop the overall model
2. Build the features list
3. Plan by feature
4. Design by future
5. Build by future

4.4.2 Diagram:

In this diagram, we shows the working how FDD works:[4]



4.5 Implementation

First, the End User must install the application on their device. When they start the application after installing they will be welcomed by the "Hygenator" logo screen, after that the Sign In or sign up screen are pop up. If the End User is new on the application, they will first create an account by sharing the information necessary by the application, such as their "name, password, email, and so on".

After successfully making an account, the End User can sign in and be redirected to the home screen. The End User can search for "Restaurant" by their name on their needs from the home screen by clicking on the search bar.

The End User will be redirected to the specific Restaurant after selecting the major category via search or scrolling down on the home screen.

The End User will see Restaurant details such as "Direction, Restaurant Type, Menu, Hygiene Related Reviews and Suggestions" on a specific Restaurant screen. At the end this application allows the End User to share their Restaurant experience and suggestions in the form of "feedback or reviews".

CHAPTER 5

USER MANUAL

5.1 Welcoming Screen

When the user open the application the welcoming screen welcome the users with a Hygenator logo.



Figure 5.1 Welcoming Screen

5.2 Login & Sign up Screen

The Login and Sign up screen allow user to enter their identity to if the user not new they simply enter their confidential information to login their session, or if the user is new then they sign up the application.

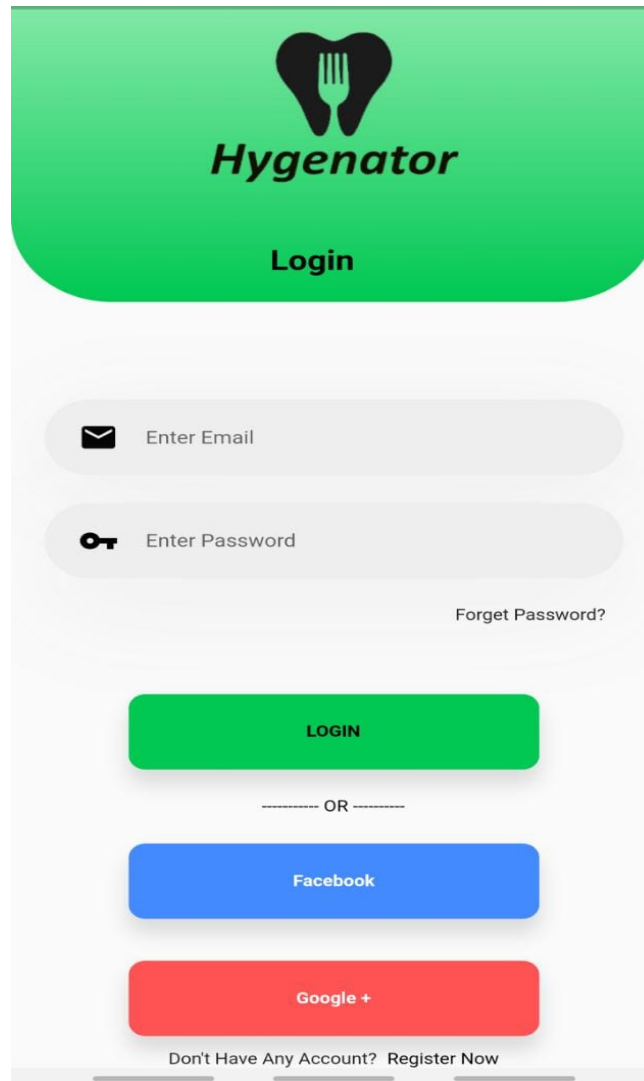


Figure 5. 2: Login & Sign up Screen

5.3 Profile Screen

After the login The Hygenator Application the User can see their information and also check his/her profile and modify their profile according to their need.

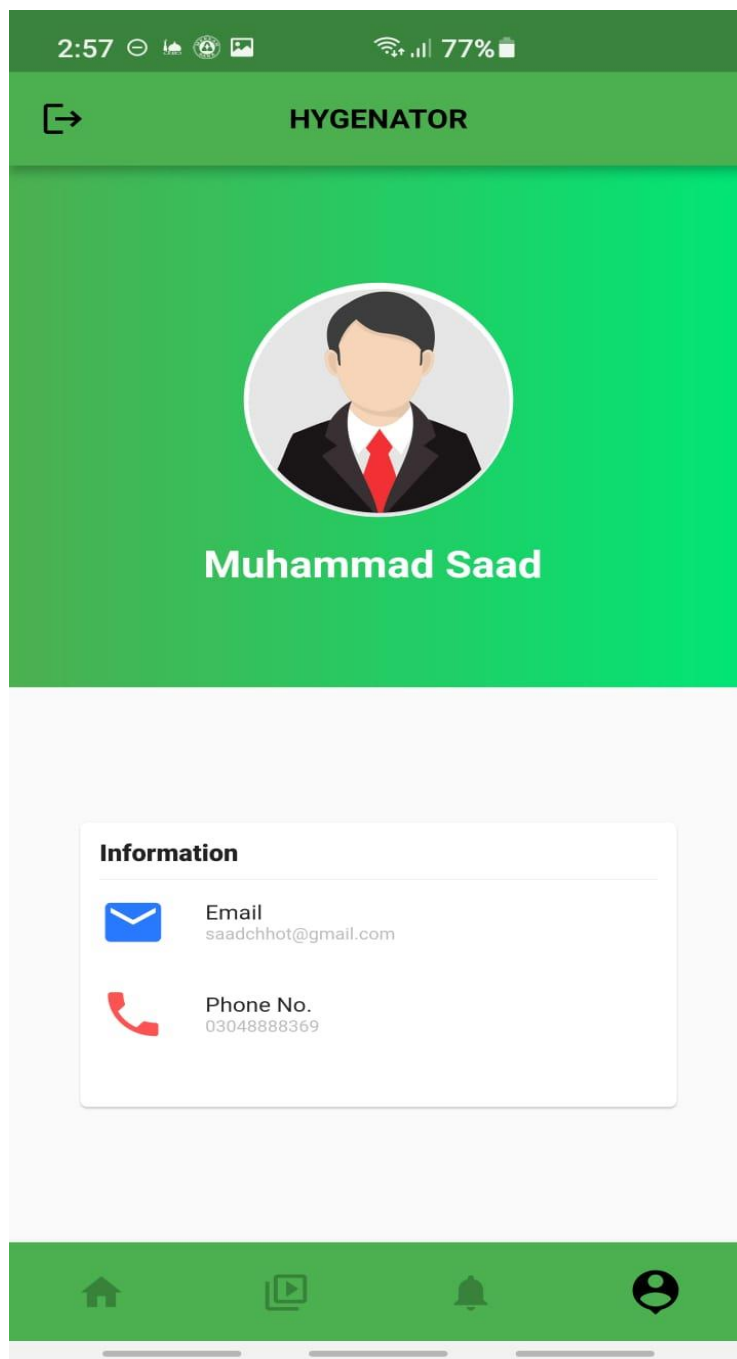


Figure 5. 3: Profile Screen

5.4 Home Screen

After the user login or sign up the application the user directly go to Home screen.

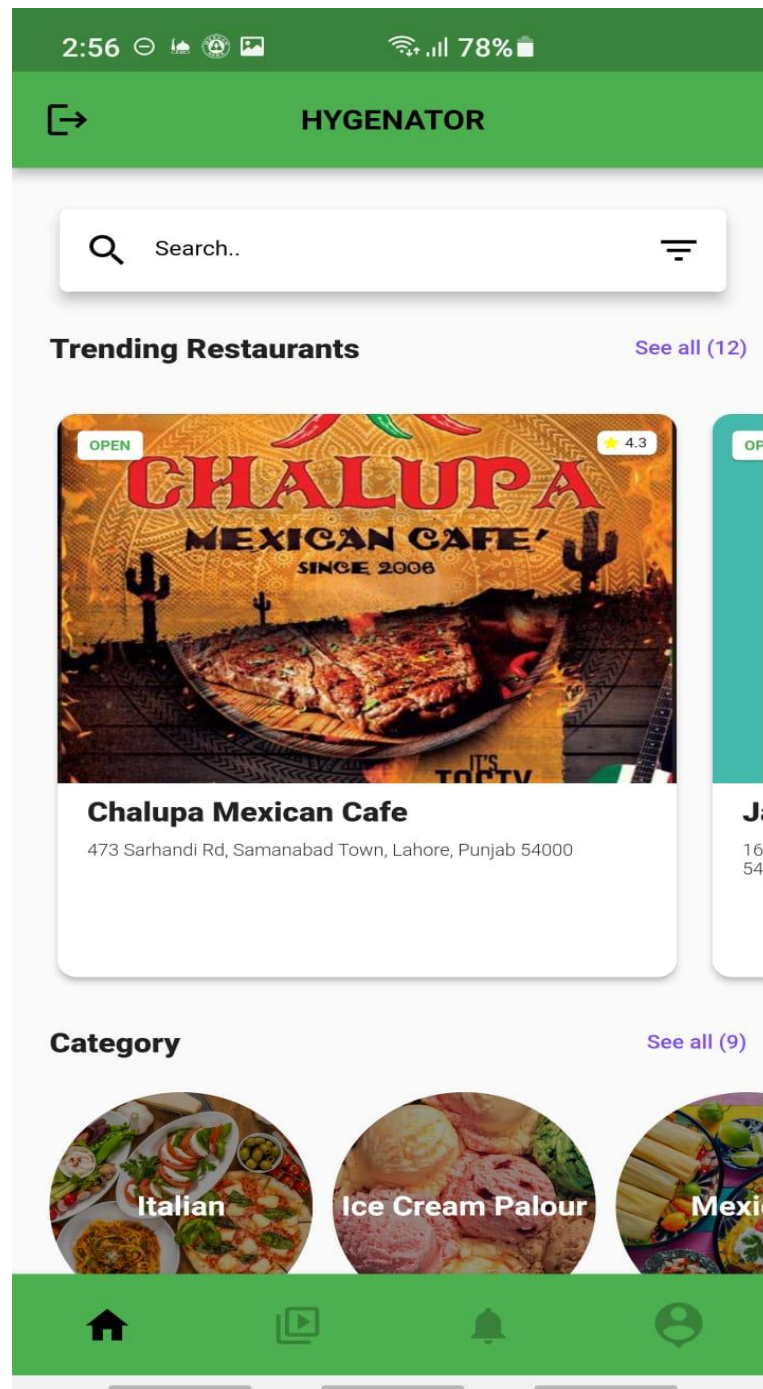


Figure 5. 4: Home Screen

5.5 Categories Screen

In this screen the user can check the categories of the restaurants and find restaurants easily by just one click.

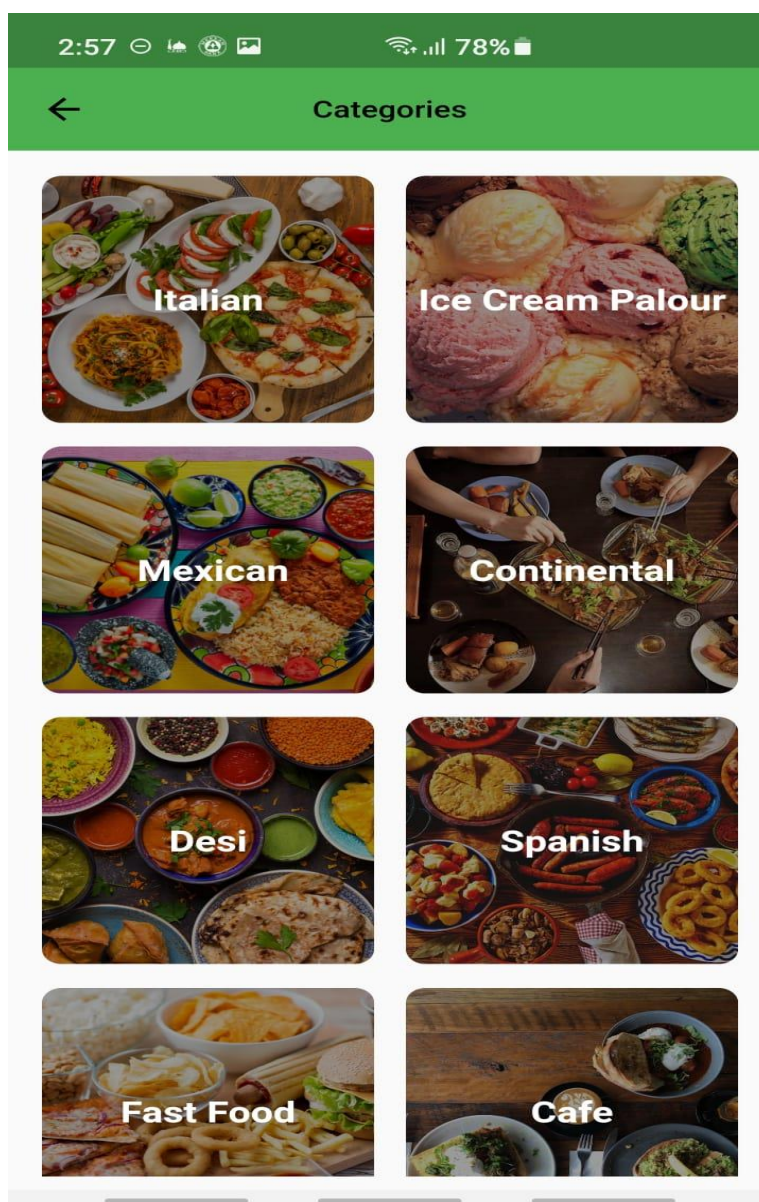


Figure 5. 5: Categories Screen

5.6 Specific Restaurant Screen

In the Specific Restaurant Screen the application shows the specific restaurant details such as: Direction, Menu, Hygiene Review, etc.

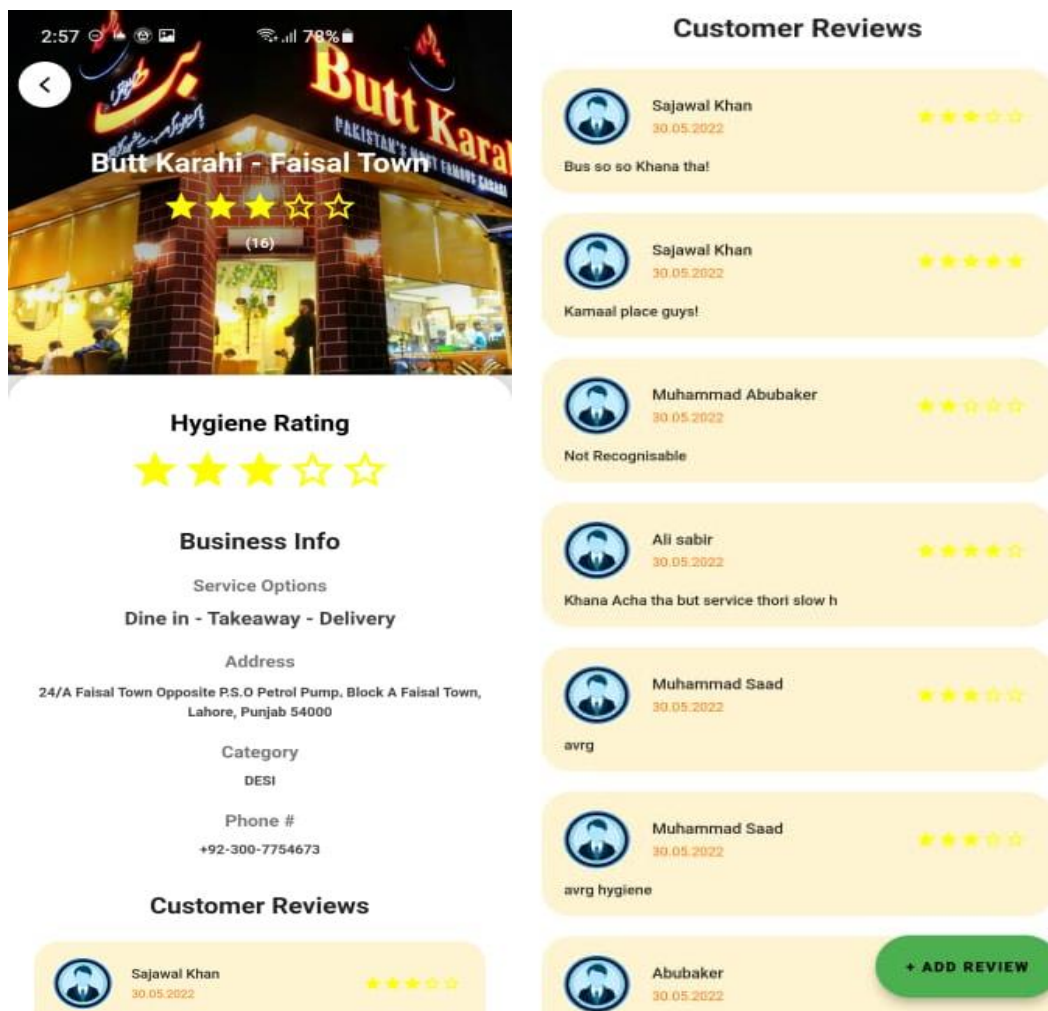


Figure 5. 6: Specific Restaurant Screen

5.7 Hygiene Reviews Screen

In this screen the user give reviews and hygiene rating.

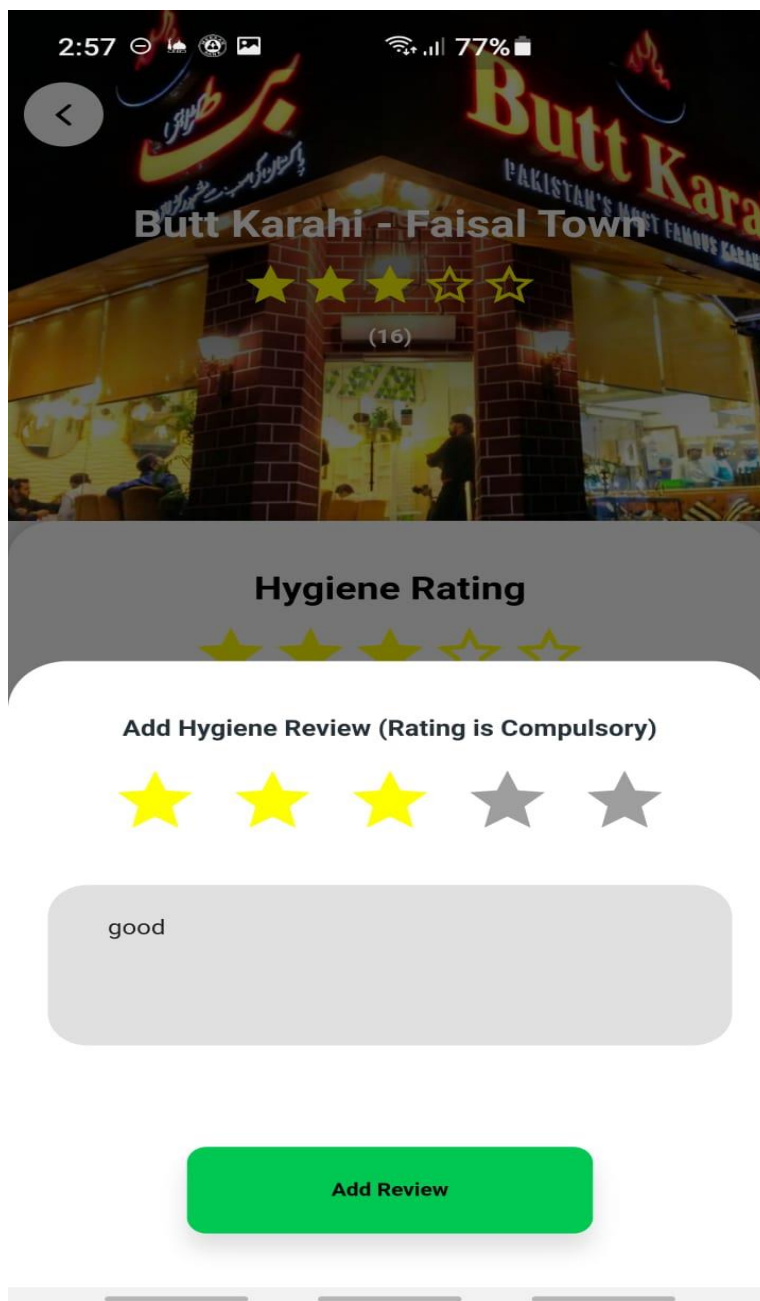


Figure 5. 7: Hygiene Reviews Screen

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

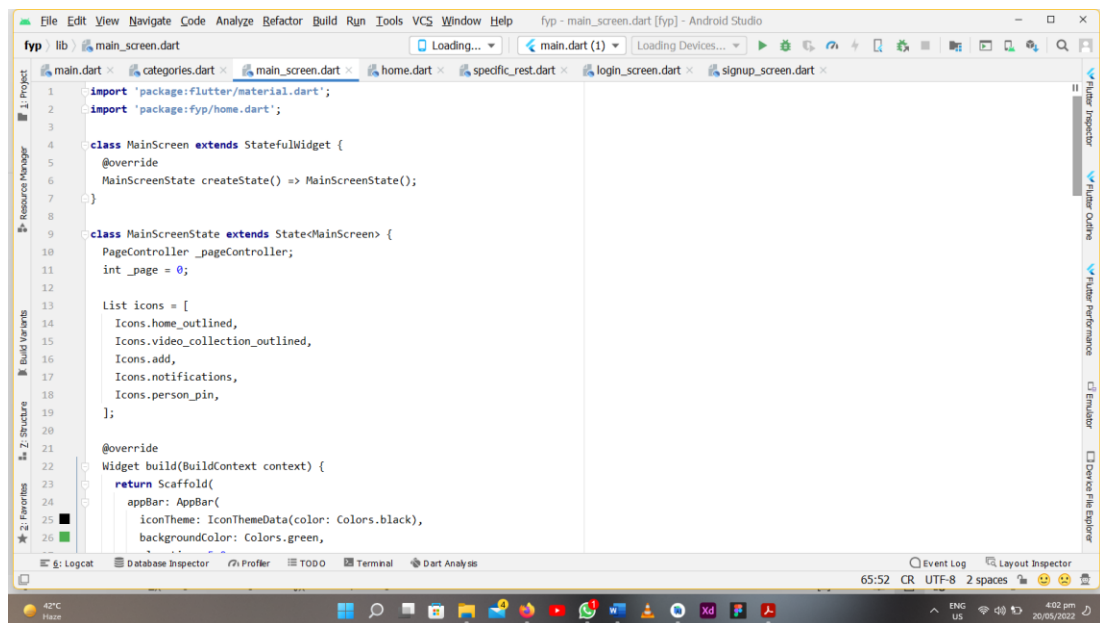
Our application allows the user to explore restaurant reviews and ratings. Restaurant reviews and ratings are not provided by most food platforms. The main goal of the app is to make life easier for both users and proprietors. It will lower your chances of picking the wrong restaurant. The issues mentioned above impact both locals and tourists. An application that allows users to browse different places and verify their ratings and hygiene levels. It was intended to improve the usability of the user interface design. The primary goal of this application is to tell our people/users of which restaurants offer a healthy environment and cuisine.

REFERENCES

- [1] “ISO Certification for Restaurants | ISO Certification for Hotel.” <https://siscertifications.co.in/iso-certification-for-restaurants-hotels/> (accessed Nov. 17, 2021).
- [2] S. Mortlock and S. Mortlock, “Hygiene Behaviour and Food Quality in Lahore , Pakistan,” vol. 3, no. Photograph 3, pp. 1539–1542, 2015.
- [3] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta, “Agile software development methods: Review and analysis,” *CoRR*, vol. abs/1709.0, 2017.
- [4] “Flowchart Maker & Online Diagram Software.” https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1&title=Untitled%20Diagram.drawio
- [5] “Dart overview | Dart.” <https://dart.dev/overview> (accessed May 20, 2022).
- [6] “Flutter - Build apps for any screen.” https://flutter.dev/?gclid=Cj0KCQjw1ZeUBhDyARIsAOzAqQIFeVQBQ0mgmEuPdz0dOL1sGiU_0wOoOYTHjwxMvwZGfgI-74mcvLwaAv8AEALw_wcB&gclidsrc=aw.ds (accessed May 20, 2022).
- [7] “What is Adobe XD and What is it Used for?” <https://www.adobe.com/products/xd/learn/get-started/what-is-adobe-xd-used-for.html> (accessed May 20, 2022).
- [8] “Meet Android Studio | Android Developers.” <https://developer.android.com/studio/intro> (accessed May 20, 2022).
- [9] “What is Feature Driven Development (FDD)? | Definition.” <https://www.productplan.com/glossary/feature-driven-development/> (accessed May 20, 2022).

APPENDICES

Appendices 1: Main screen:

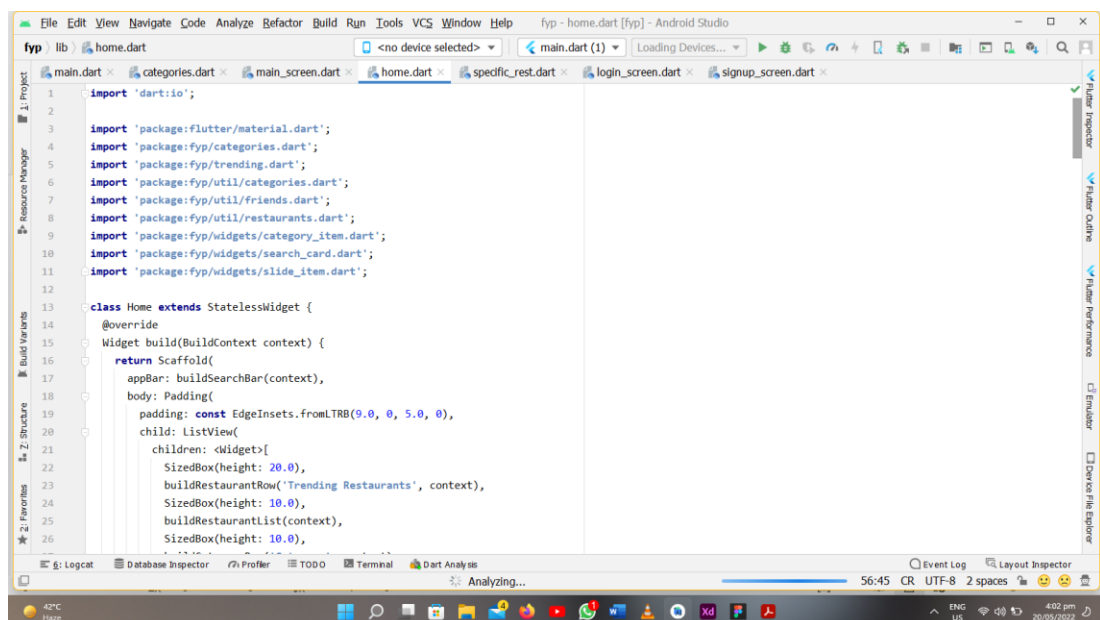


```

1  import 'package:flutter/material.dart';
2  import 'package:fyp/home.dart';
3
4  class MainScreen extends StatefulWidget {
5    @override
6    MainScreenState createState() => MainScreenState();
7  }
8
9  class MainScreenState extends State<MainScreen> {
10   PageController _pageController;
11   int _page = 0;
12
13   List icons = [
14     Icons.home_outlined,
15     Icons.video_collection_outlined,
16     Icons.add,
17     Icons.notifications,
18     Icons.person_pin,
19   ];
20
21   @override
22   Widget build(BuildContext context) {
23     return Scaffold(
24       appBar: AppBar(
25         iconTheme: IconThemeData(color: Colors.black),
26         backgroundColor: Colors.green,

```

Appendices 2: Home screen:



```

1  import 'dart:io';
2
3  import 'package:flutter/material.dart';
4  import 'package:fyp/categories.dart';
5  import 'package:fyp/trending.dart';
6  import 'package:fyp/util/categories.dart';
7  import 'package:fyp/util/friends.dart';
8  import 'package:fyp/util/restaurants.dart';
9  import 'package:fyp/widgets/category_item.dart';
10 import 'package:fyp/widgets/search_card.dart';
11 import 'package:fyp/widgets/slide_item.dart';
12
13 class Home extends StatelessWidget {
14   @override
15   Widget build(BuildContext context) {
16     return Scaffold(
17       appBar: buildSearchBar(context),
18       body: Padding(
19         padding: const EdgeInsets.fromLTRB(9.0, 0, 5.0, 0),
20         child: ListView(
21           children: <Widgets>[
22             SizedBox(height: 20.0),
23             buildRestaurantRow('Trending Restaurants', context),
24             SizedBox(height: 10.0),
25             buildRestaurantList(context),
26             SizedBox(height: 10.0),

```

Appendices 3: Login Screen:

```

1  import 'package:flutter/material.dart';
2  import 'package:fyp/controller/authentication.dart';
3  import 'package:fyp/main_screen.dart';
4  import 'package:fyp/specific_rest.dart';
5  import 'package:get/get.dart';
6  import 'package:fyp/signup_screen.dart';
7
8  import 'package:get/get.dart';
9
10 import 'categories.dart';
11
12 class LoginScreen extends StatefulWidget {
13   @override
14   State<StatefulWidget> createState() => StartState();
15 }
16
17 class StartState extends State<LoginScreen> {
18   final _formKey = GlobalKey<FormState>();
19   Map<String, String> userLoginData = {"email": "", "password": ""};
20
21   @override
22   Widget build(BuildContext context) {
23     AuthContorll controller = Get.put(AuthContorll());
24     authentication() {
25       if (_formKey.currentState.validate()) {
26         print("form is valid");

```

Appendices 4: Categories:

```

1  import 'package:flutter/material.dart';
2  import 'package:fyp/categories.dart';
3  import 'package:fyp/util/categories.dart';
4  import 'package:fyp/widgets/category_item.dart';
5
6  class Categories extends StatefulWidget {
7   @override
8   _CategoriesState createState() => _CategoriesState();
9 }
10
11 class _CategoriesState extends State<Categories> {
12   @override
13   Widget build(BuildContext context) {
14     return Scaffold(
15       appBar: AppBar(
16         iconTheme: IconThemeData(color: Colors.black),
17         backgroundColor: Colors.green,
18         elevation: 0.0,
19         title: Text(
20           'Categories',
21           style: TextStyle(color: Color(0xffff0000)),
22         ),
23         centerTitle: true,
24       ),
25       body: Padding(
26         padding: EdgeInsets.all(

```

Appendices 5: Admin

The screenshot shows the Firebase Cloud Firestore console. The left sidebar contains navigation options: Project Overview, Build (Authentication, App Check, Firestore Database, Realtime Database, Extensions, Storage, Hosting, Functions, Machine Learning), and Release & Monitor (Crashlytics, Performance, Test Lab). The main area displays the 'restaurants' collection under the 'EkNidUu6jfo...' project. A document with ID 'EkNidUu6jfoEzBLcmF1' is selected, showing its fields: address, category, contact, id, image, open_till, and reviews.

Field	Value
address	"H Block Market, Street 144, Defence H Block Sector H Dha Phase 1, Lahore, Punjab"
category	"fast food"
contact	"..."
id	"EkNidUu6jfoEzBLcmF1"
image	"https://assets.simpleviewinc.com/simpleview/image/upload/crm/nacogdoches/maxresdefault-1-2ffb717c5056b3a_2ffb7308-5056-b3a8-493fce3b4e62589c.jpg"
open_till	"12:00AM"
reviews	0

The screenshot shows the Firebase Cloud Firestore console. The left sidebar is the same as in the previous image. The main area displays the 'users' collection under the '2x2e4c0h1qh...' project. A document with ID '2x2e4c0h1qhxTqcGHRbWvyxrQ52' is selected, showing its fields: email, phone_no, and uid.

Field	Value
email	"syedhamzaarif20@gmail.com"
phone_no	"03338313260"
uid	"2x2e4c0h1qhxTqcGHRbWvyxrQ52"

hygenerator

ORIGINALITY REPORT

9%

SIMILARITY INDEX

3%

INTERNET SOURCES

0%

PUBLICATIONS

9%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to President University Student Paper	5%
2	Submitted to United Colleges Group - UCG Student Paper	1%
3	dspace.calstate.edu Internet Source	1%
4	Submitted to Glasgow Caledonian University Student Paper	<1%
5	Submitted to Manchester Metropolitan University Student Paper	<1%
6	Submitted to South Bank University Student Paper	<1%
7	Submitted to Kookmin University Student Paper	<1%
8	elib.uni-stuttgart.de Internet Source	<1%
9	www.upi-yptk.ac.id Internet Source	<1%

10

lib.buet.ac.bd:8080

Internet Source

<1 %

11

myfik.unisza.edu.my

Internet Source

<1 %

12

opus.lib.uts.edu.au

Internet Source

<1 %

Exclude quotes On

Exclude bibliography On

Exclude matches < 4 words