“ DESIGN AND DEVELOPMENT OF MEGA FOOD DELIVERY ANDROID APP ”

Submitted

By

NAMAN MADIRATTA (1701410065)
SUGAM RAIZADA (1701410102)
TUSHAR SINGH (1701410107)

Submitted to the Department of Computer Science and Engineering

In partial fulfillment of the requirements

For the degree of

Bachelor of Technology

In

Computer Science & Engineering

Shri Ram Murti Smarak College of Engineering & Technology, Bareilly

Dr. A.P.J. Abdul Kalam Technical University, U.P., Lucknow

(July, 2021)

DECLARATION

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature………………………………	Signature………………………………
Name…………………………………..	Name…………………………………..
Roll No…………………………………	Roll No…………………………………
Date……………………………………	Date…………………………………….

Signature………………………………

Name………………………………….

Roll No…………………………………

Date…………………………………….

CERTIFICATE

This is to certify that the Project Report entitled “Design and Development of Mega Food Delivery Android App” which is submittedby Naman Madiratta (1701410065), Sugam Raizada (1701410102) and Tushar Singh (1701410107) is a record of the candidates own work carried out by them under my supervision. The matter embodied in this work is original and has not been submitted for the award of any other work or degree.

Dr. L. S. Maurya	Mr. Ashish Agarwal	Md. Shadab Hussain
HOD (CSE/IT)	**Project In charge (CS2)**	**Supervisor**

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Assistant Professor Md. Shadab Hussain, Computer Science & Engineering, S.R.M.S.C.E.T, Bareilly for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavors have seen light of the day.

We also take the opportunity to acknowledge the contribution of Dr. L. S. Maurya, Head, Department of Computer Science & Engineering/Information Technology, S.R.M.S.C.E.T, Bareilly for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature…………………………………	Signature………………………………
Name……………………………………..	Name…………………………………..
Roll No…………………………………..	Roll No…………………………………
Date……………………………………..	Date…………………………………….

Signature…………………………………

Name……………………………………..

Roll No…………………………………..

Date……………………………………..

ABSTRACT

With the increasing popularity of food delivery, the traditional telephone order food has inconvenience to the customers and food delivery stores. How to make the food delivery more quickly and conveniently has become a concern of people.So this report explores and develops new takeway apps that are easier and more object oriented than other apps.Food delivery app has easy and simple features , but now the food-delivery app is no more convenient for people, the overall is relatively old, not novel enough, cannot attract new users. Based on this feature, we decided to design a system for people.

The food-delivery should be timely, convenient and comprehensive, many food-delivery systems today, but the function is not comprehensive, and some did not meet the requirements of timely delivery, not friendly enough. Payment is too simple, layout is too rigid, and the update of information is not timely enough. While we are developing new systems, we keep the basic on-time features that take-out systems have to offer, along with delivered on time and more comprehensive recommendations. The new takeaway system makes it easier for people to meet the needs of most people, with a more innovative layout and more appealing to new users.

LIST OF FIGURES

LIST OF ABBREVIATIONS

SQL	Structured Query Language
SDK	Software Development Kit
JVM	Java Virtual Machine
IDE	Integrated Development environment
OS	Operating System
IOS	Iphone Operating System

TABLE OF CONTENTS

CHAPTER 1

1.1 Introduction

With the increasing popularity of food delivery in daily life , the traditional telephone order food has inconvenience to the customers and the food delivery store. The online food ordering system provides convenience for the customers. It overcomes the disadvantages of the traditional queuing system. This system increases the takeaway of foods than visitors.Therefore, this system enhances the speed and standardization of taking the order from the customer. It provides a better communication platform. the user’s details are noted electronically. Using this application, the customers need not go to the restaurant by themselves, but they can order the dishes through Android mobiles anywhere.

1.2 Current system

In an existing system for giving any orders, a user should visit Hotels or Restaurants to know about food items and then give orders and pay in advance or you need to select menu and place an order on call. In this method time and manual work is required. Maintaining critical information in the files and manuals is full of risk and tedious process. Tracking of Delivery is not available and Booking of a particular table in advance is also not available. Customization of Order, Current status of the order is not available. Some systems contain an outdated database that is Restaurant is closed, yet it shows on the application.

Drawbacks of Existing System

- Inconvenient to customer and takes longer time.
- Customer cannot view their previous orders.
- Nobody shows the current status of delivery.
- Less options to choose from.
- Customer cannot have a list of their favorite restaurants.
- Customer cannot select restaurants based on ratings given by other customers.

1.3 Proposed system

The online food ordering system provides convenience for the customers. It overcomes the disadvantages of the traditional queuing system. This system increases the takeaway of foods than visitors. Therefore, this system enhances the speed and standardization of taking the order from the customer. It provides a better communication platform. the user's details are noted electronically. Using this application, the customers need not go to the restaurant by themselves, but they can order the dishes through Android mobiles anywhere.The User can see the list of Restaurants on the basis of the User Ratings given. The user can see the different cuisines offered at the restaurants and the related food menus along with their prices. The User can place the order accordingly.

Advantages of proposed system

- One step registration with android application.
- Advance ordering.
- Customize food ordering.
- More convenient and easy to use.
- No need to waste time standing in long queues.
- More options to choose from.
- 24X7 availability.

1.4 Objective of the project

A food delivery app focuses on making the process of food ordering easy and convenient for the customers. The main aim of food ordering app is to develop food ordering system using mobile application which will help customers to manage their food ordering.

Main objectives include :

- The main objective is to create a system that allows customers to order food online.
- The next important objective is to make the process of ordering quick, easy and convenient.
- The System is User Friendly so that any person using it will not face any difficulties in operating it.
- The system has facilities that allow users to view menu card, select the items as required, add them to cart, view menu card, select the items as required, view special discounts on food items, select a food delivery, rate the items and services provided.
- Less time consuming hence increases the efficiency of the system.
- To allow users to view the food catalogue and search for food items.
- 0nable people to install the application on their smartphones.
- To register new users or customers.
- To allow customers to order using the application.
- To allow customers to view or check their delivery status.

1.5 Project advantages and limitations

Advantages

- For those who are disabled and elderly, who cannot get out of the house, people who don't have anybody to physically support them, the food ordering app is very useful.
- No need to waste time standing in long queues.

- When we order food more regularly and get familiar with all the deals and discounts, we can avail the benefits during special occasions.

Limitation

- This application requires active internet connection.
- Lack of internet knowledge to customers.
- Limited no of menu choices.
- No Bargaining.

CHAPTER 2

MOTIVATION

The current system for food ordering is to visit the restaurant manually and from the available list of menu items choose the item he wants and purchase the item by paying the price of the item.

- It is less user-friendly.
- User must go to restaurant and select items.
- Description of the restaurant is limited.
- It is a time consuming process.
- Not in reach of distant users .

In the proposed system customer need not go to the restaurant for placing order. He can order the items he wish to buy through the application in his Smartphone. The online food ordering system provides convenience for the customers. It overcomes the disadvantages of the traditional queuing system. This system increases the takeaway of foods than visitors. Therefore, this system enhances the speed and standardization of taking the order from the customer. It provides a better communication platform. the user's details are noted electronically. Using this application, the customers need not go to the restaurant by themselves, but they can order the dishes through Android mobiles anywhere.

CHAPTER 3

SECURITY AND PRIVACY PRACTICES IN ANDROID

Android is focused on helping users take advantage of the latest innovations, while making sure users' security and privacy are always a top priority.

Pay attention to permissions

Build trust with your users by being transparent and providing users control over how they experience your app. Request the minimum permissions that your feature needs. Whenever you introduce major changes to your app, review the requested permissions to confirm that your app's features still need them.

Minimize your use of location

If your app requests permission to access location, help users make an informed decision. If your app collects location information, explain to users how your app uses this information to deliver specific benefits to them. If your app can support its use cases without requiring any location data, don't request any location permissions.

Handle data safely

Be transparent and secure in how you handle sensitive data. Make users aware of when and why your app collects, uses, or shares sensitive data. Always use secure network connections. For your app's data at rest, use Android's built-in credential encryption. For data in transit, you should use TLS, the successor of SSL, for all data transmission regardless of sensitivity.

CHAPTER 4

LIST OF MODULES

Modules:

In this system there are two modules namely, Admin and User. Admin can login, manage restaurants by adding, updating and deleting, manage delivery person by adding, updating and deleting. Admin can also check register users and the orders total count. Users can register and login. User have option to choose the cuisine, hotels of nearby. User will get details of restaurant like name, location and reviews. User will can select the food from the menu list, can add to favorites and can processed further. User can view the history of their orders and the current orders status.

The system comprises of 2 major modules with their sub-modules as follows:

1. **Admin**

- **Manage Restaurants:** Admin can manage restaurants by adding, updating and deleting.
- **View Users:** Admin can view users
- **View Orders:** Admin can view orders.

2. **User**

- **Register:** User can register and get login
- **Login:** User can login using credentials.
- **Profile:** User can set their profile.
- **Change Password:** User can change their password.
- **Cuisine:** User can select list of cuisines. Can also list hotels and search for the desire restaurant.
- **Restaurant:**Restaurant Details been shown like address, location, reviews and ratings.

- **Menu:** list of Menu with price, details and photos. Filter menu by Kind of Food e.g.: Appetizers, Main course, Sides etc. Proceed to Order, Cart Page - Modify deleted items, Order Confirmation/Payment/COD etc.
- **Orders:** Users can view the pervious and current order history and also can track the order.
- **Favorites :** Favorites Food/remove from Favorites.

CHAPTER 5

TOOLS AND TECHNOLOGY

5.1 Tools

5.1.1 Software specifications

5.1.1.1 Tools Used

- Android Studio
- Android SDK

5.1.1.2 Technologies Used

- Android
- Java
- Kotlin
- SQL Lite

Android v5.0 or Higher

5.1.2 Hardware specifications

Laptop or PC

- i3 Processor Based Computer or higher
- 2GB RAM
- 5 GB Hard Disk

Android Phone or Tablet

- 1.2 Quad core Processor or higher
- 2 GB RAM

5.2 TECHNOLOGY

5.2.1 ANDROID

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on notebooks, game consoles, digital cameras, and other electronics.Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. As of July 2013, the Google Play store has had over one million Android applications ("apps") published, and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of developers create applications for Android, and a 2015 survey found that 40% of full-time professional developers see Android as their priority target platform.

Figure 5.1: Android Logo

ANDROID ARCHITECTURE

We studied the Android system architecture. Android system is a Linux-based system, Use of the software stack architecture design patterns . As shown in Figure 1, the Android architecture consists of four layers: Linux kernel, Libraries and Android runtime, Application framework and Applications. Each layer of the lower encapsulation, while providing call interface to the upper.

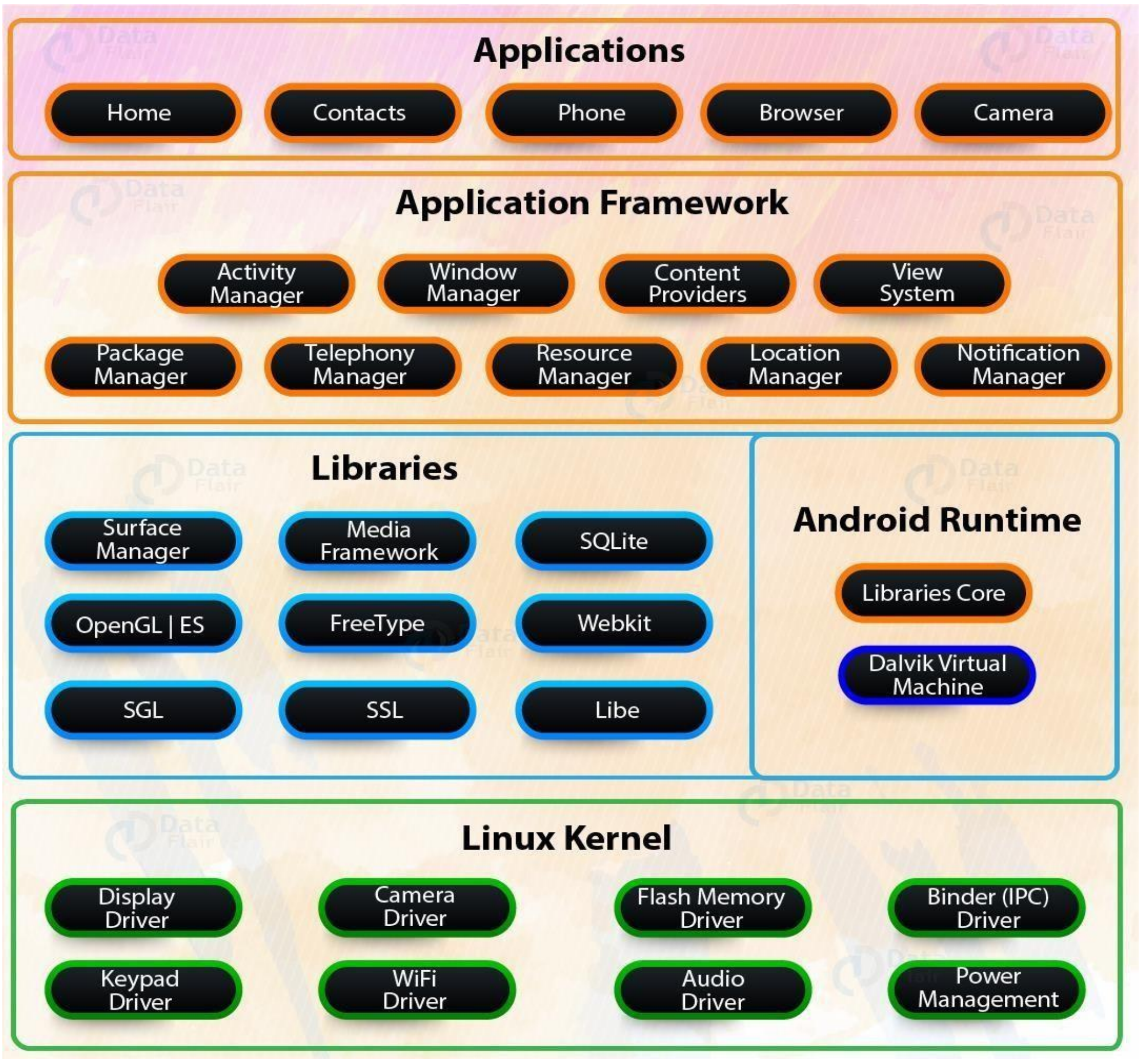


Figure 5.2: Android Architecture

- **Applications:** Android app will be shipped with a set of core applications including client, SMS program, calendar, maps, browser, contacts, and others. All these application programs are developed in Java.
- **Application Framework :** The developer is allowed to access all the API framework of the core programs. The application framework simplifies the reuse of its components. Any other app can release its functional components and all other apps can access and use this component (but have to follow the security of the framework). Same as the users can be able to substitute the program components with this reuse mechanism.
- **Libraries and Android Runtime :** The library is divided in to two components: Android Runtime and Android Library. Android Runtime is consisted of a Java Core Library and Dalvik virtual machine. The Core Library provides Java core library with most functions. Dalvik virtual machine is register virtual machine and makes some specific improvements for mobile device. Android system library is support the application framework, it is also an important link connecting between application framework and Linux Kernel. This system library is developed in C or C++ language. These libraries can also be utilized by the different components in the Android system. They provide service for the developers through the application framework.
- **Linux Kernel :** The kernel system service provided by Android inner nuclear layer is based on Linux 2.6 kernel, Operations like internal storage, process management, internet protocol, bottom drive and other core service are all based on Linux kernel.

5.2.2 JAVA

Our core Java programming tutorial is designed for students and working professionals. Java is an object-oriented, class-based, concurrent, secured and general-purpose computer-programming language. It is a widely used robust technology. Java is a programming language and a platform. Java is a high level, robust, object-oriented and secure programming language. Java was developed by *Sun Microsystems* (which is now the subsidiary of Oracle) in the year 1995. *James Gosling* is known as the father of Java. Before Java, its name was *Oak*. Since Oak was already a registered company, so James Gosling and his team changed the Oak name to Java.

Figure 5.3: JAVA Logo

5.2.3 KOTLIN

Kotlin is a cross-platform, statically typed, general-purpose programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of Kotlin's standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript (e.g., for frontend web applications using React) or native code (via LLVM); e.g., for native iOS apps sharing business logic with Android apps.Language development costs are borne by JetBrains, while the Kotlin Foundation protects the Kotlin trademark. Google announced that the Kotlin programming language is now its preferred language for Android app developers.Since the release of Android Studio 3.0 in October 2017, Kotlin has been included as an alternative to the standard Java compiler. The Android Kotlin compiler produces Java 8 bytecode by default (which runs in any later JVM), but lets the programmer choose to target Java 8 up to 16, for optimization, or allows for more features, e.g. Java 8 related with Kotlin 1.4,and has experimental record class support for Java 16 compatibility. Kotlin support for JavaScript (i.e. classic back-end) is considered stable in Kotlin 1.3 by its developers, while Kotlin/JS (IR-based) in version 1.4, is considered alpha. Kotlin/Native Runtime (for e.g. Apple support) is considered bet.

Figure 5.4 : KOTLIN Logo

5.2.4 SQLITE

SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. It is the most used database engine in the world. It is an in-process library and its code is publicly available. It is free for use for any purpose, commercial or private. It is basically an embedded SQL database engine. Ordinary disk files can be easily read and write by SQLite because it does not have any separate server like SQL. The SQLite database file format is cross-platform so that anyone can easily copy a database between 32-bit and 64-bit systems. Due to all these features, it is a popular choice as an Application File Format.

Figure 5.5: SQLite Logo

5.3 TOOLS

5.3.1 ANDROID STUDIO

Android Studio is an integrated development environment (IDE) for developing for the Android platform. It was announced on May 16, 2013 at the Google I/O conference. Android Studio is freely available under the Apache License 2.0. Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The firststable build was released in December 2014, starting from version 1.0. Based on JetBrains' IntelliJ IDEA software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS X and Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

Figure 5.6:Android Studio

5.3.2 ANDROID SDK

A software development kit (SDK or "devkit") is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. To create applications you have to download this software development kit. For example, if you want to create an Android app you require an SDK with java programming, for iOS apps you require an iOS SDK with swift language, and to develop MS Windows apps you require the .net language. There are also SDKs that are installed in apps to provide analytics and data about activity. Prominent examples include Google and Facebook.

Figure 5.7: Android SDK

CHAPTER 6

METHODOLOGY

As mentioned before one goal of this project is to provide hustle free environment for customers for online food ordering by viewing list of restaurants , their menus and order food online.These conditions are bound to many variables, and the method used here in this project is the Agile methodology. Agile software development describes an approach to software development under which requirements and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s). It advocates adaptive planning, evolutionary development, early delivery, and continual improvement, and it encourages rapid and flexible response to change. The term agile (sometimes written Agile) was popularized, in this context, by the Manifesto for Agile Software Development. While there is significant anecdotal evidence that adopting agile practices and values improves the agility of software
professionals, teams and organizations; some empirical studies have found no scientific evidence.

The Agile approach seeks alternatives to traditional project management. This project is based on the AGILE APPROACH which includes the following:

• Understanding of problem
• Module development
• Integration and testing
• Customer feedback
• Pass/rebuild

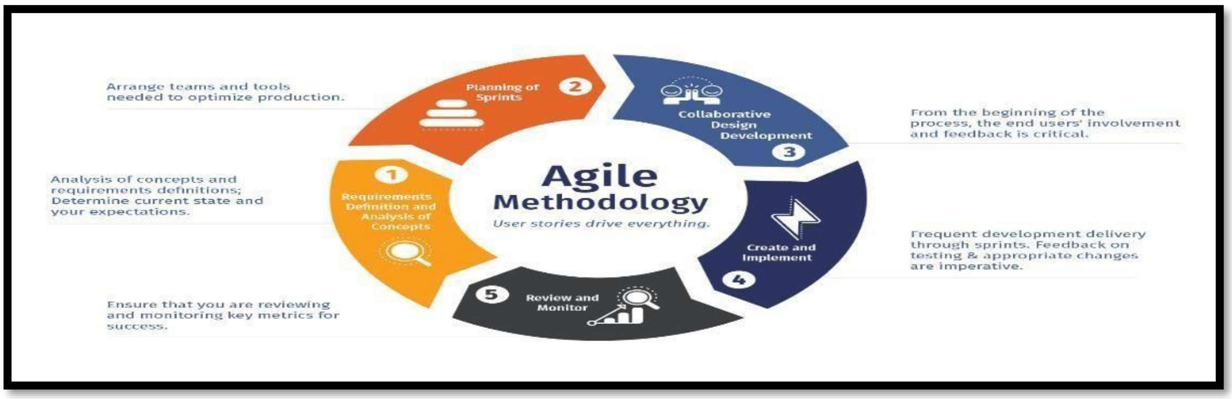

Arrange teams and tools needed to optimize production.
Planning of Sprints
2
Collaborative Design Development
3
From the beginning of the process, the end users' involvement and feedback is critical.
Analysis of concepts and requirements definitions; Determine current state and your expectations.
1
Requirements Definition and Analysis of Concepts
Agile Methodology
User stories drive everything.
Create and Implement
4
Frequent development delivery through sprints. Feedback on testing & appropriate changes are imperative.
Ensure that you are reviewing and monitoring key metrics for success.
5
Review and Monitor

Figure 6.1: Agile Methodology

Scrum is the most popular way of introducing Agility due to its simplicity and flexibility. Scrum is founded on empiricism and lean thinking. Empiricism asserts that knowledge comes from experience and making decisions based on what is observed. Lean thinking reduces waste and focuses on the essentials.Scrum employs an iterative, incremental approach to optimize predictability and to control risk. Scrum engages groups of people who collectively have all the skills and expertise to do the work and share or acquire such skills as needed.

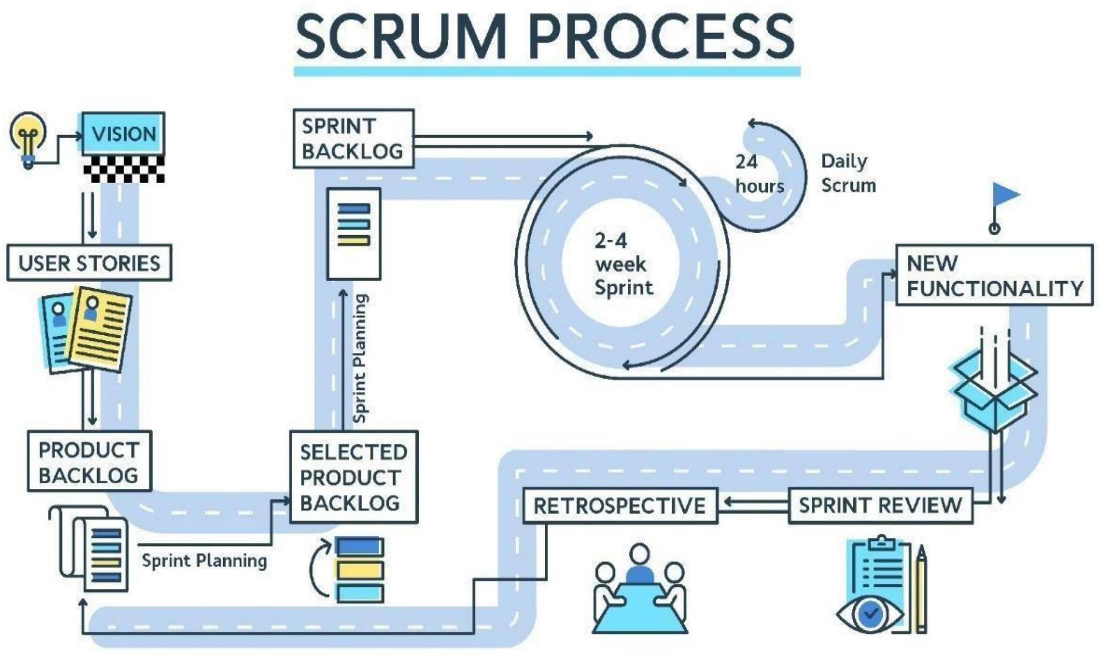


Figure 6.2: Scrum Process

❖ USE CASE DIAGRAM

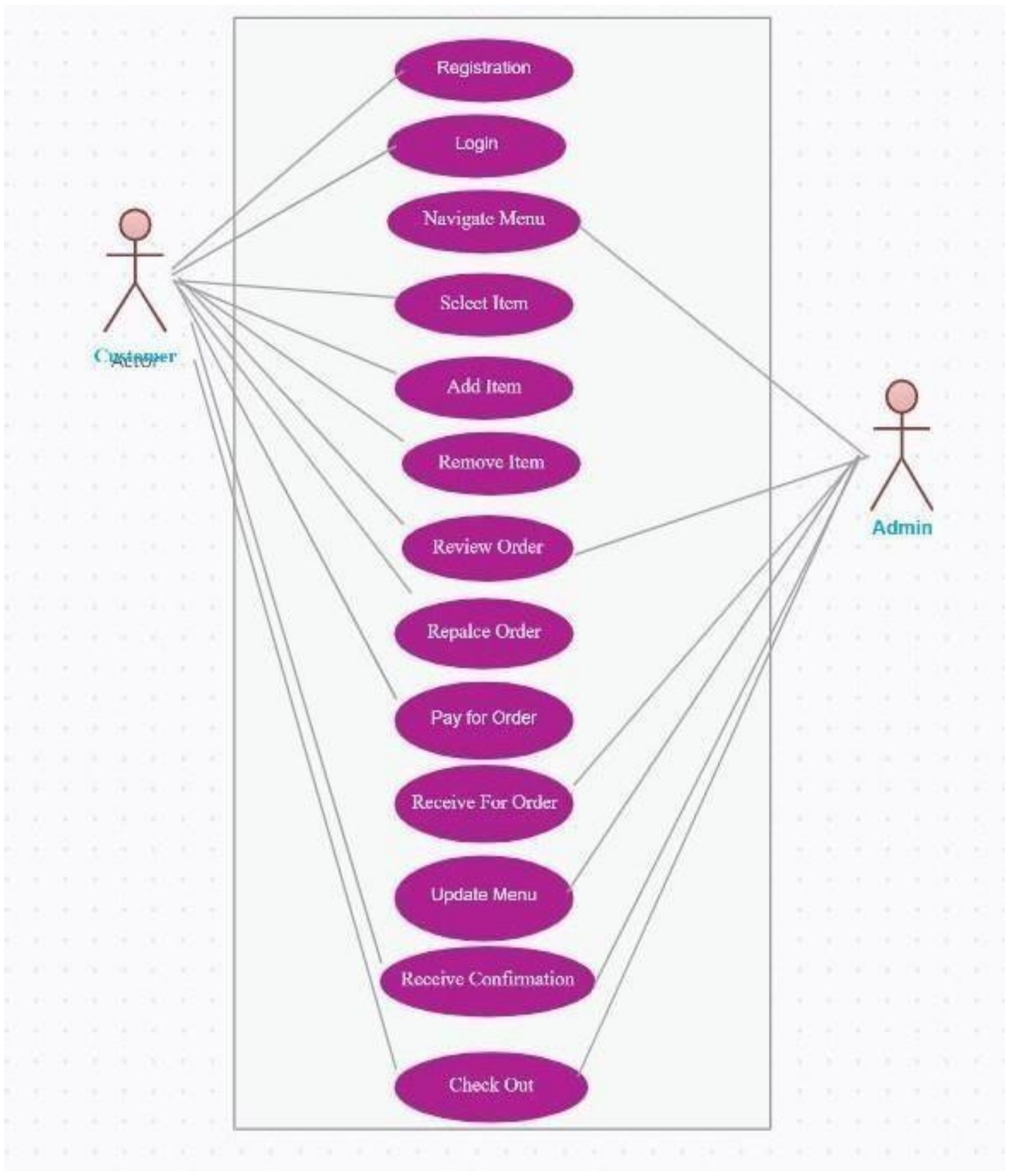


Figure 6.3: Use Case Diagram

❖ DATA FLOW DIAGRAM

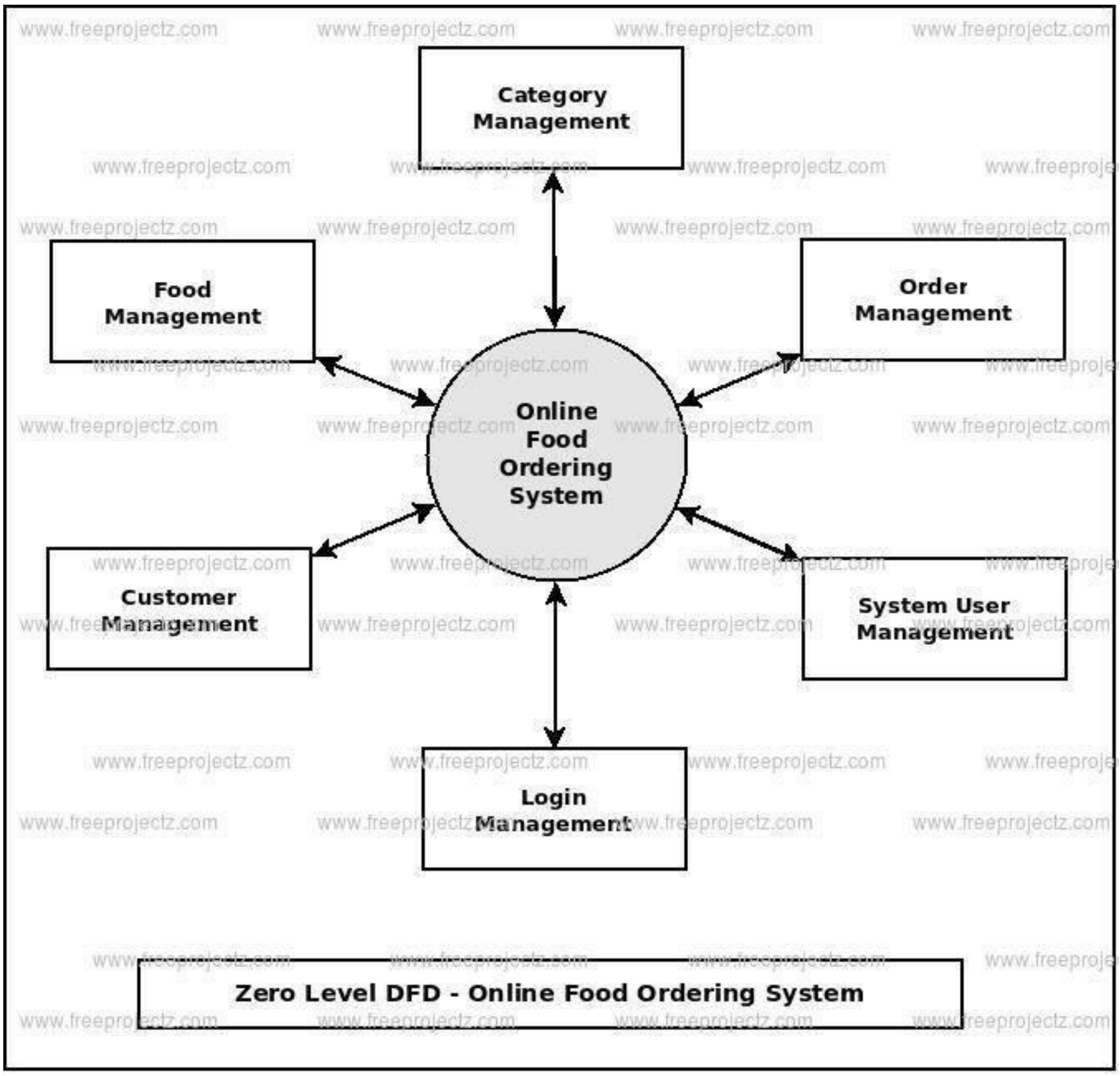


Figure 6.4 : Data Flow Diagram

❖ Activity Diagram

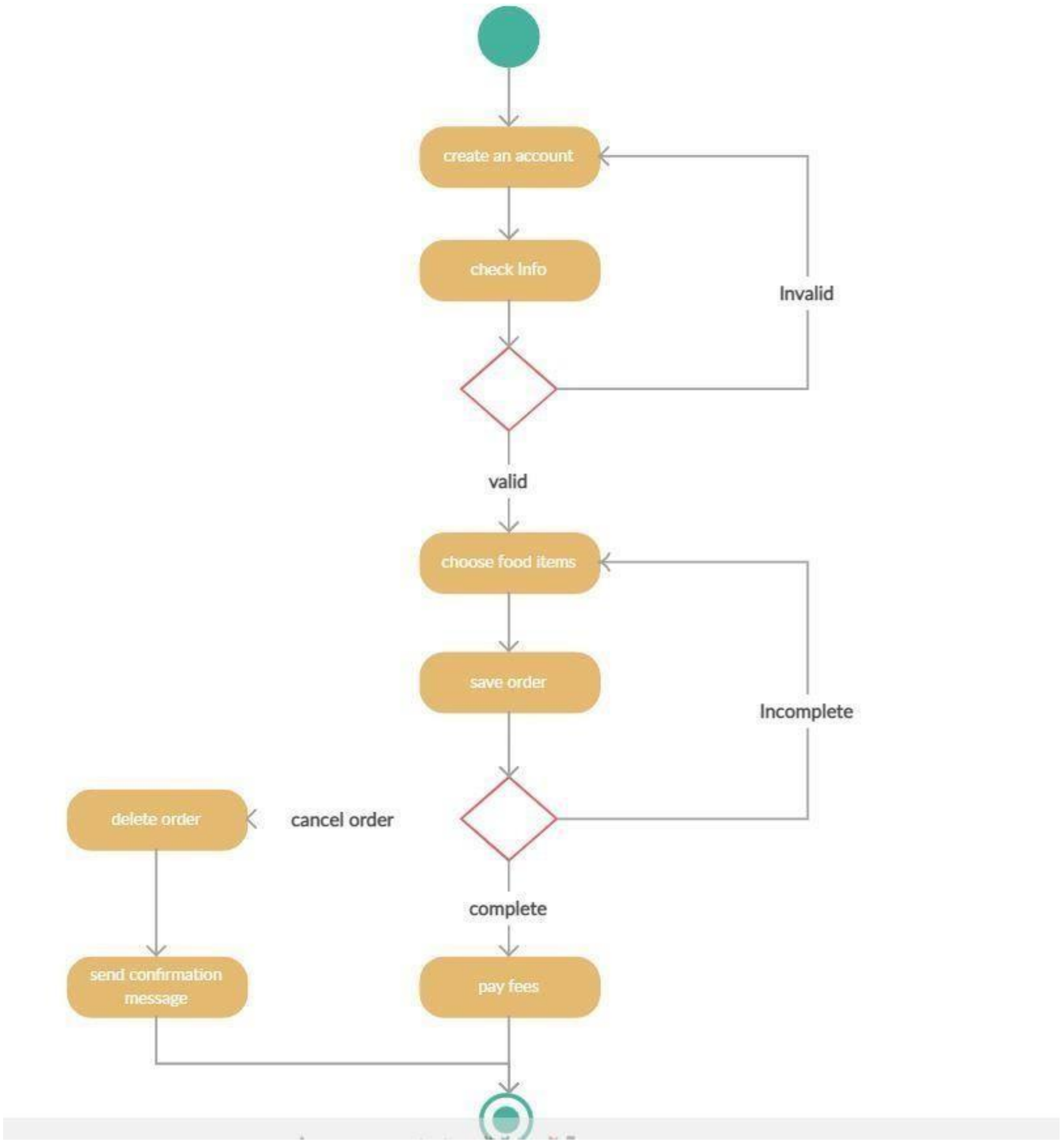


Figure 6.5 : Activity Diagram

❖ Block Diagram

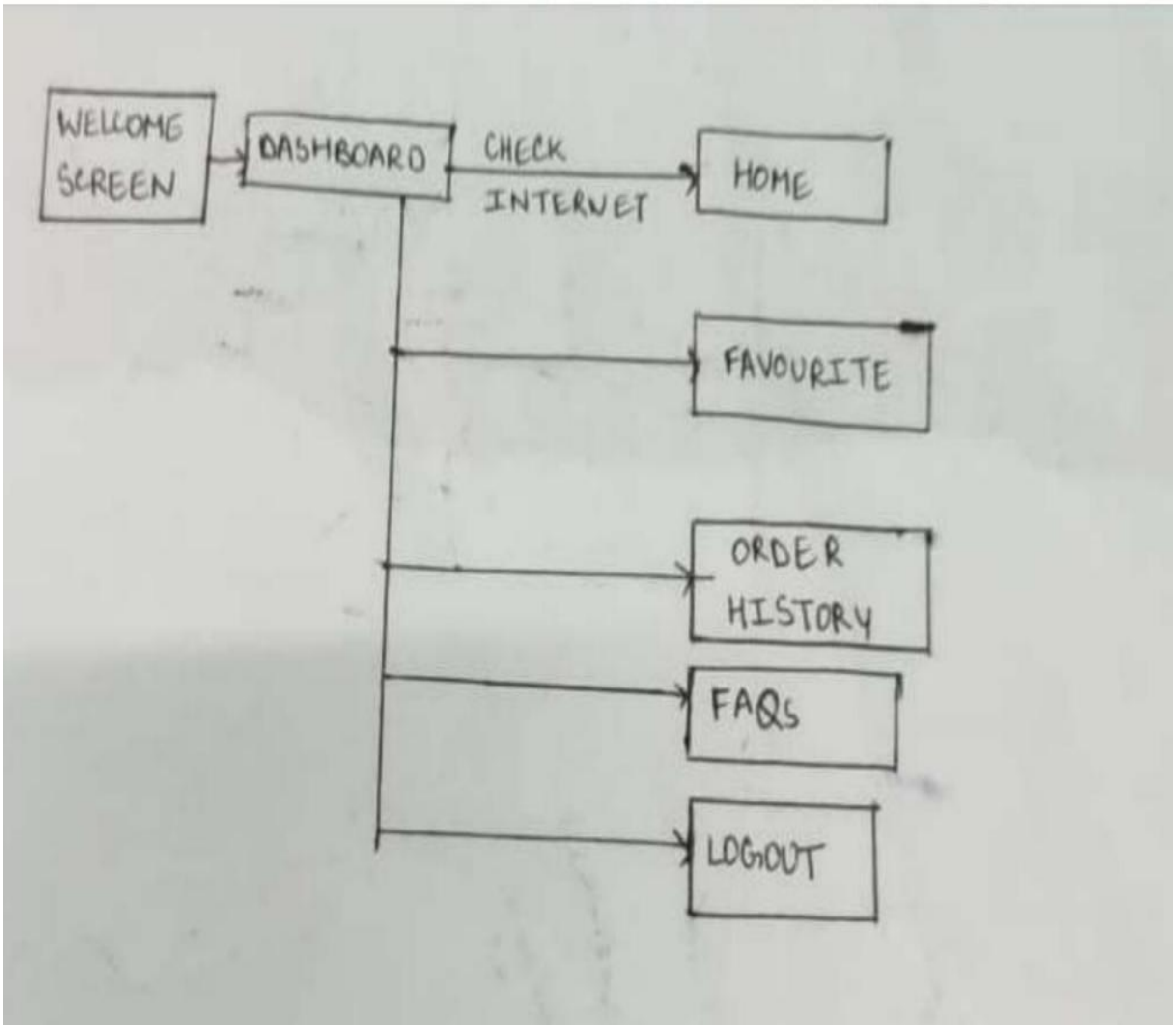


Figure 6.6 : Block Diagram

CHAPTER 7

PROJECT SNAPSHOT

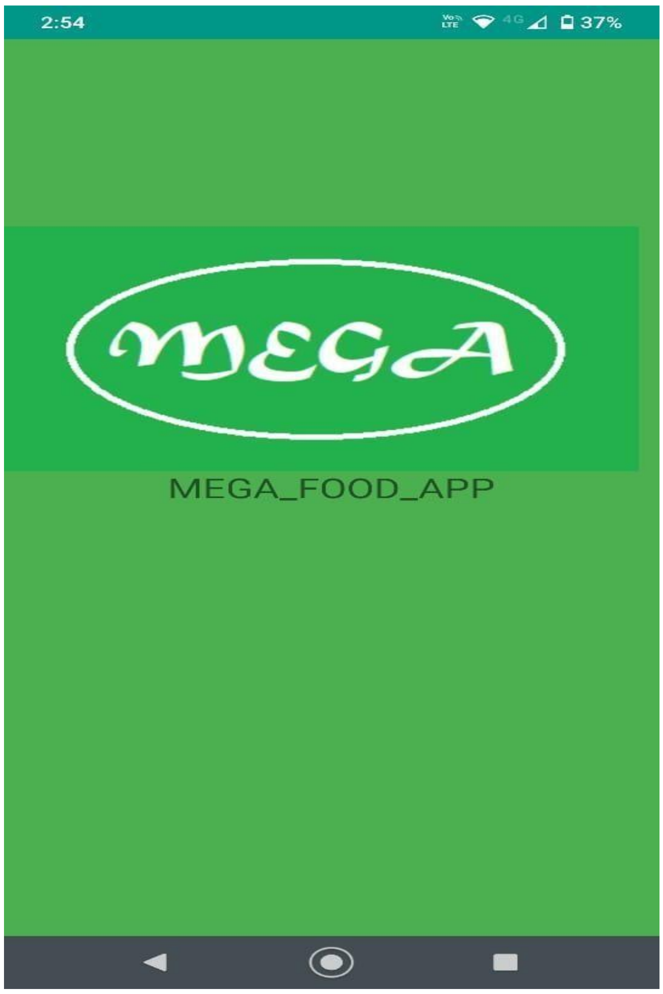


Figure 7.1 : Splash Screen

- Opening of the App with a Splash Screen for 1.5sec, presenting the App Logo and App Name.
- Login by entering suitable details.
- If already Logged-in, Home Page is opened directly!
- Option to register or change password are present on page.

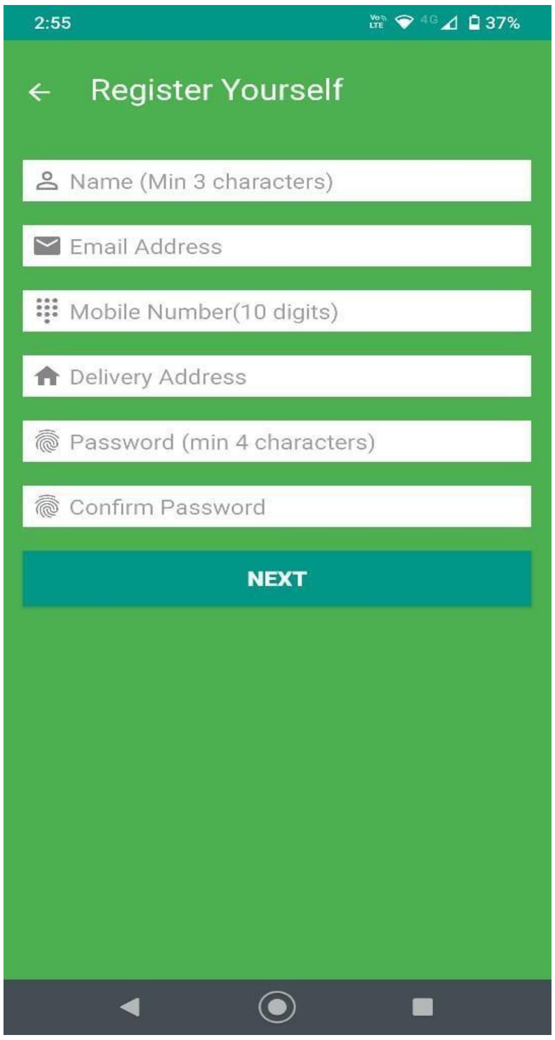
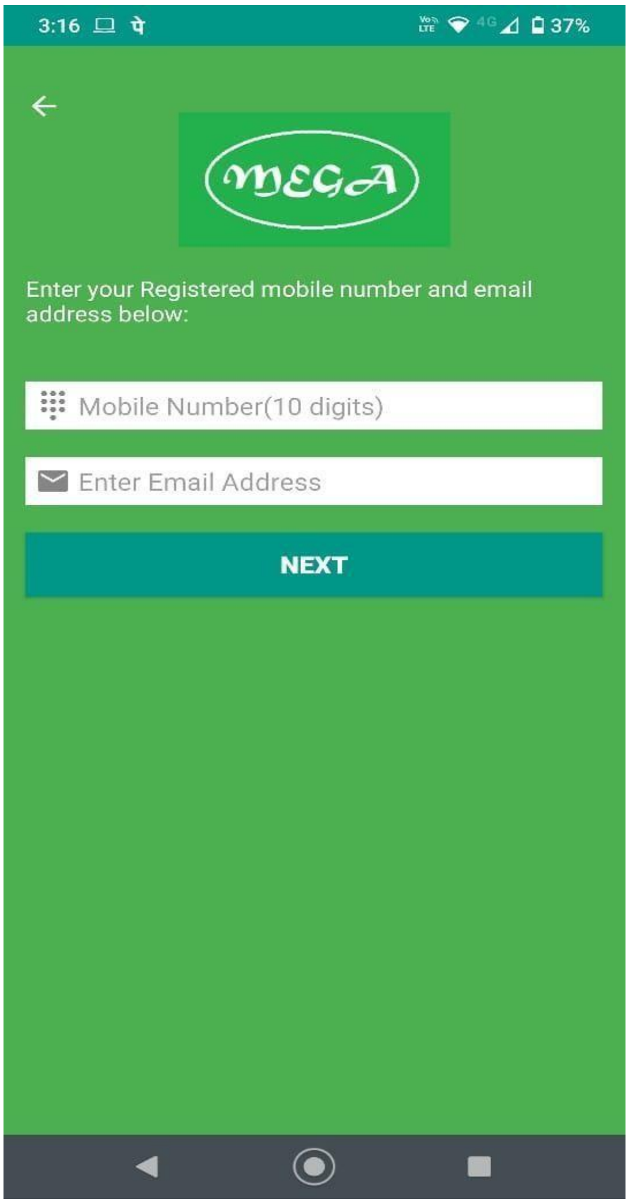


Figure 7.2 : Registration Page

- Register onto the Database by providing your details.
- If successfully registered, you're forwarded to Home Page.



Enter your Registered mobile number and email address below:

Mobile Number(10 digits)

Enter Email Address

Figure 7.3: Forgot Password Page

- An OTP(valid for 24 hours) is sent to registered Email address by providing Email and Phone Details.
- If successful, you're directed to Change Password Page. Change your password by providing new password along with the sent OTP within 24hrs.
- If successful, you're directed to Login Page.

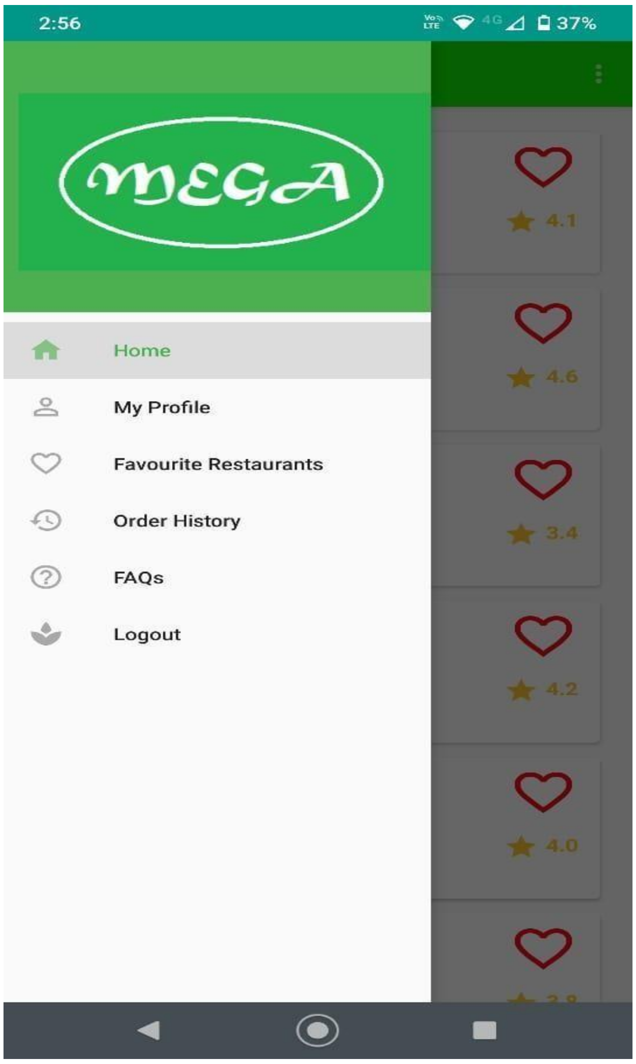
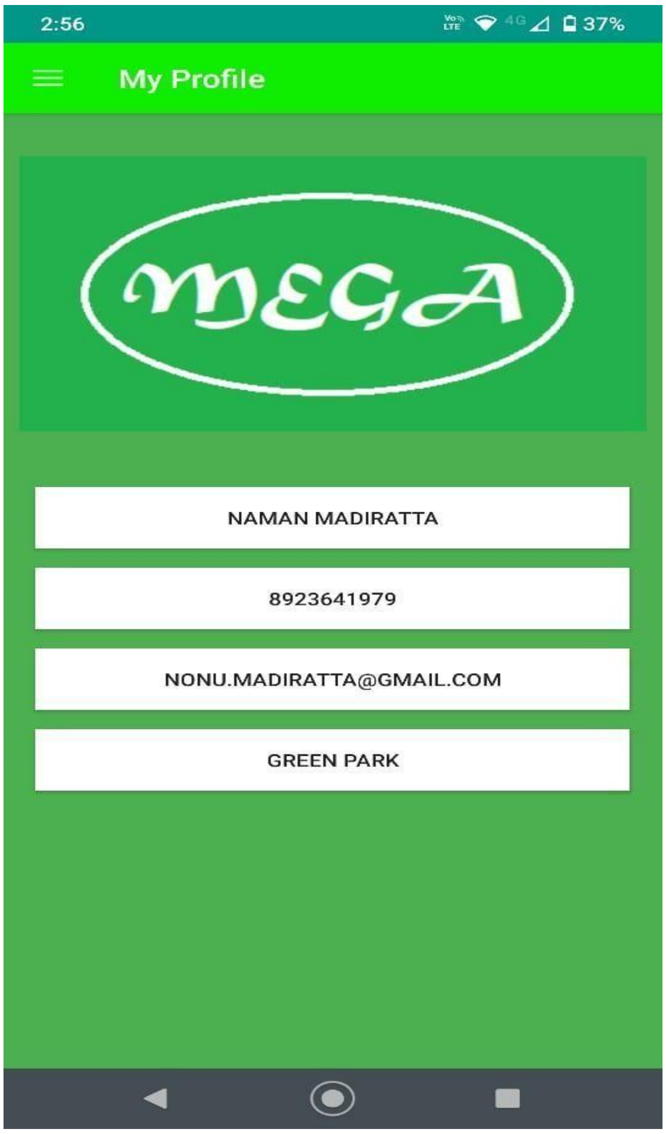


Figure 7.4 : Navigation Drawer

- Home Screen consists of a Drawer Menu, displaying App Name with menu item.
- Clicking on Menu items will open corresponding fragment.



2:56 37%

My Profile

NAMAN MADIRATTA

8923641979

NONU.MADIRATTA@GMAIL.COM

GREEN PARK

Figure 7.5: Profile

- Profile fragment displaying User's Name, Number, registered email address and delivery address.

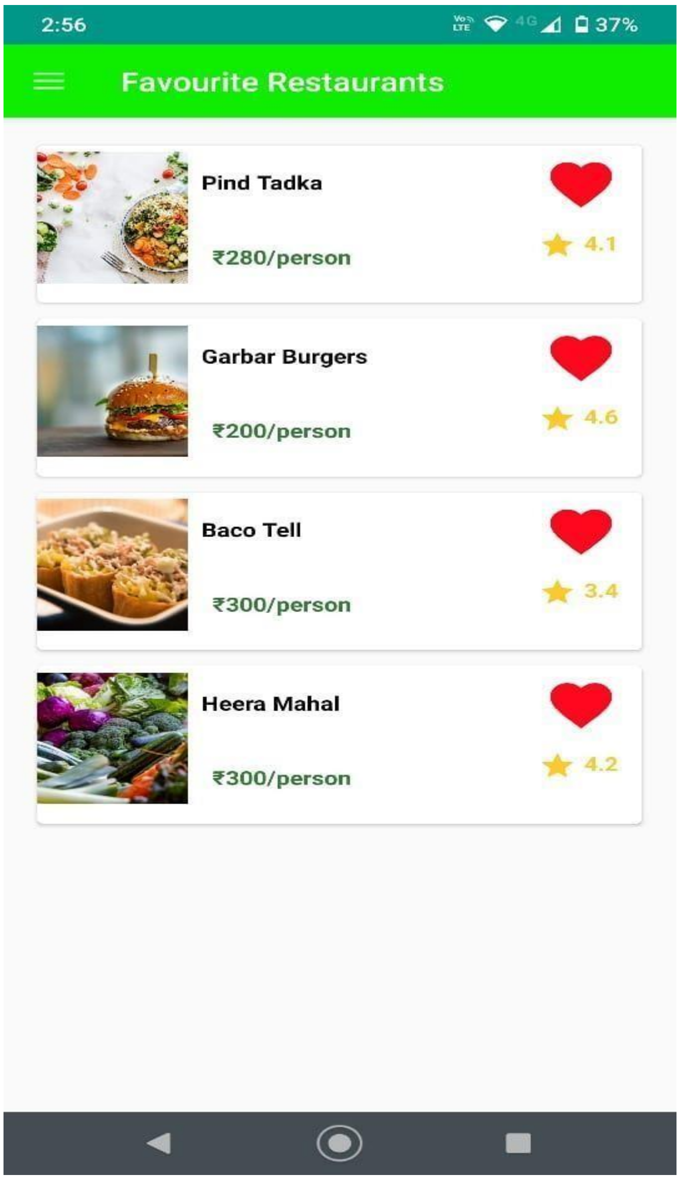
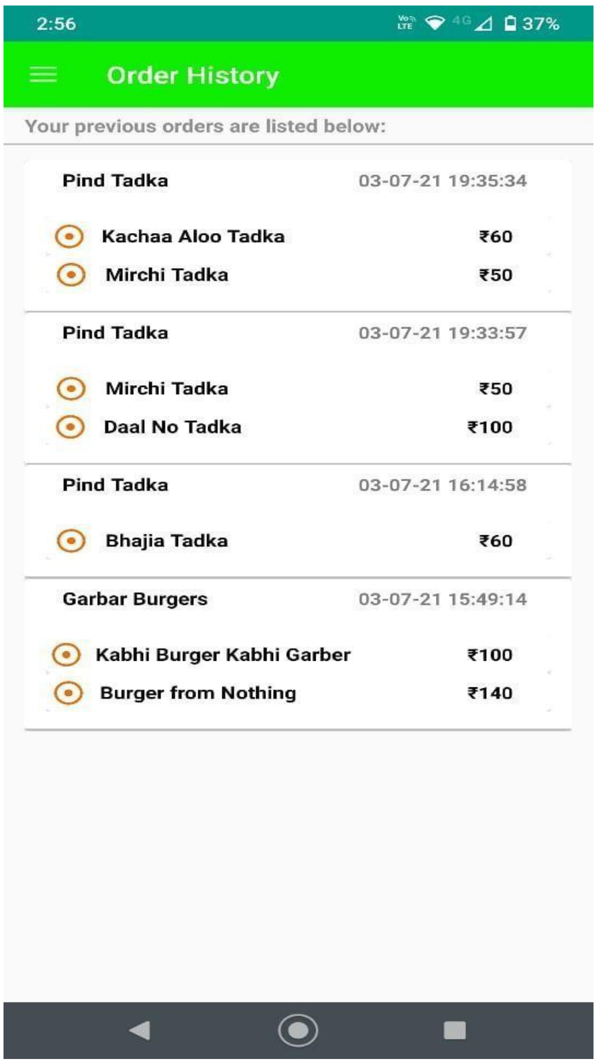


Figure 7.6 : Favourites

- Favourites fragment displays all your favourite restaurants in a list format.
- Restaurants when clicked, displays its information on Description Page.
- Option to remove a restaurant from favourites is also present.



2:56 37%

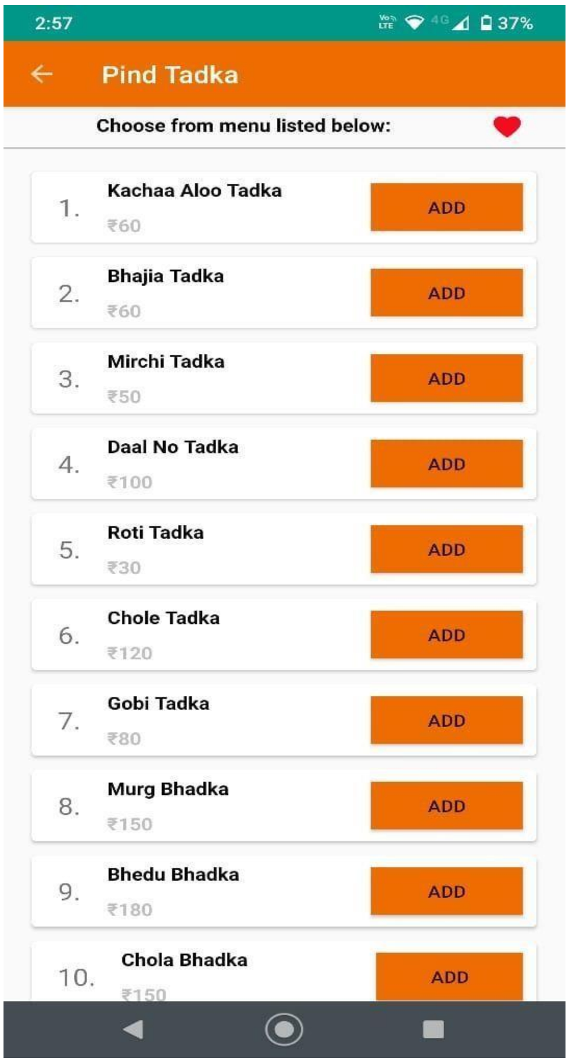
Order History

Your previous orders are listed below:

Pind Tadka	03-07-21 19:35:34
Kachaa Aloo Tadka	₹60
Mirchi Tadka	₹50
Pind Tadka	03-07-21 19:33:57
Mirchi Tadka	₹50
Daal No Tadka	₹100
Pind Tadka	03-07-21 16:14:58
Bhajia Tadka	₹60
Garbar Burgers	03-07-21 15:49:14
Kabhi Burger Kabhi Garber	₹100
Burger from Nothing	₹140

Figure 7.7 : Order History

- Displays a list of all the Previous orders, along with Restaurant Name and order time, with price of each ordered item.


2:57 37%

Pind Tadka

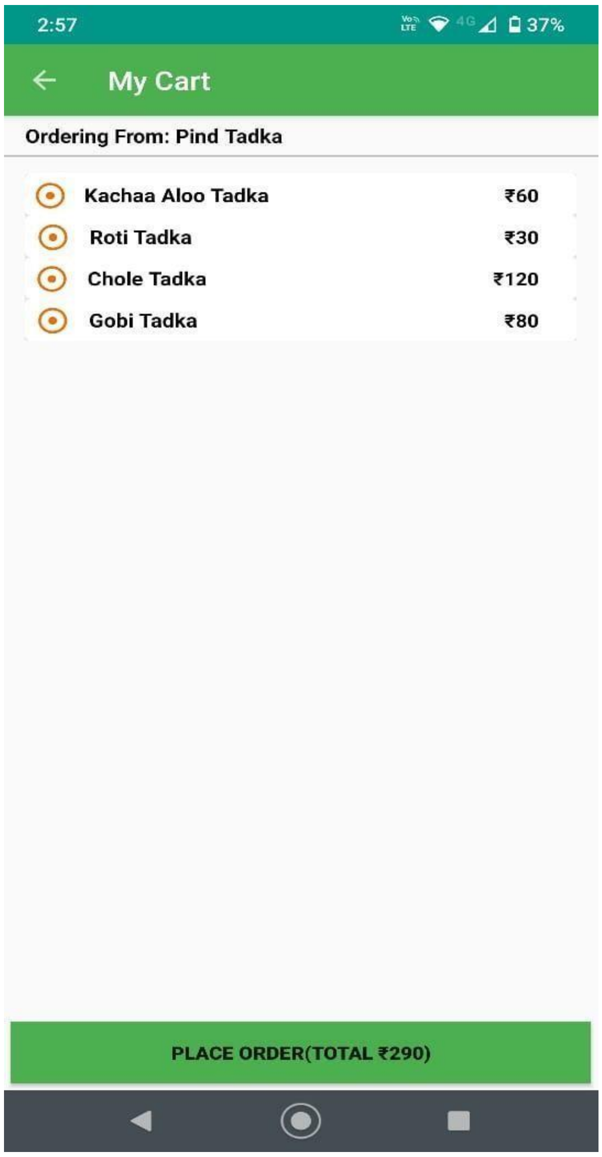
Choose from menu listed below:

1.	Kachaa Aloo Tadka	₹60	ADD
2.	Bhajia Tadka	₹60	ADD
3.	Mirchi Tadka	₹50	ADD
4.	Daal No Tadka	₹100	ADD
5.	Roti Tadka	₹30	ADD
6.	Chole Tadka	₹120	ADD
7.	Gobi Tadka	₹80	ADD
8.	Murg Bhadka	₹150	ADD
9.	Bhedu Bhadka	₹180	ADD
10.	Chola Bhadka	₹150	ADD

Figure 7.8: Restaurant Description

- Displays the restaurant's menu with the Dish's name and price.
- Option to add dish on cart and proceed to order is present along with to add restaurant in Favourites.

Displays the current order with its details and when ordered displays order confirmation page.


2:57 37%

My Cart

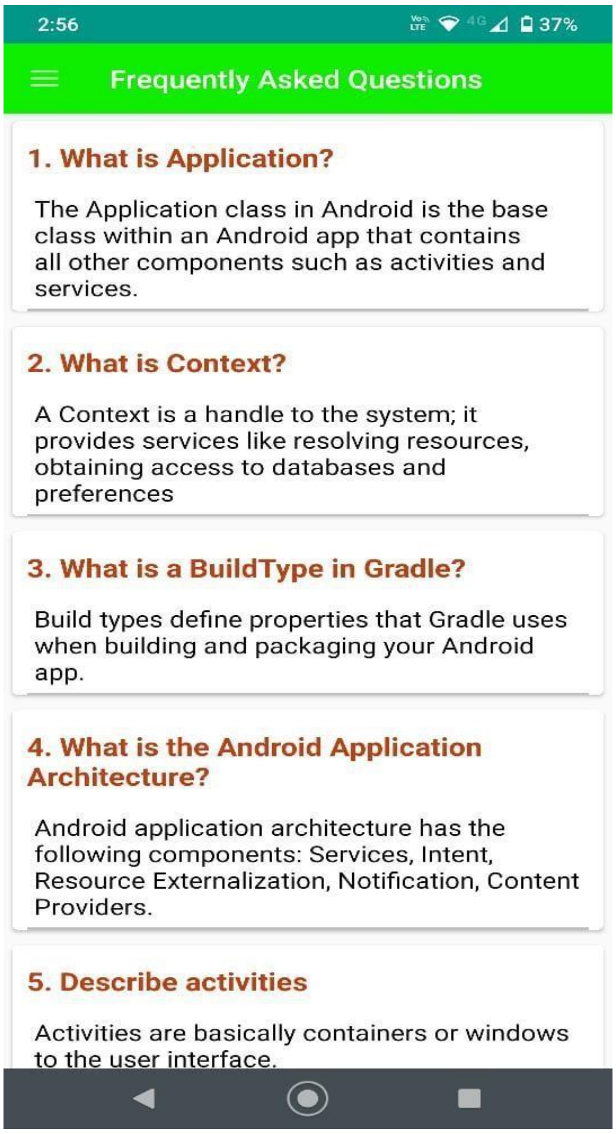
Ordering From: Pind Tadka

Kachaa Aloo Tadka	₹60
Roti Tadka	₹30
Chole Tadka	₹120
Gobi Tadka	₹80

PLACE ORDER(TOTAL ₹290)

Figure 7.9: Cart and Confirmation

- Displays the current order with its details and when ordered displays order confirmation page.


2:56 37%

Frequently Asked Questions

1. What is Application?

The Application class in Android is the base class within an Android app that contains all other components such as activities and services.

2. What is Context?

A Context is a handle to the system; it provides services like resolving resources, obtaining access to databases and preferences

3. What is a BuildType in Gradle?

Build types define properties that Gradle uses when building and packaging your Android app.

4. What is the Android Application Architecture?

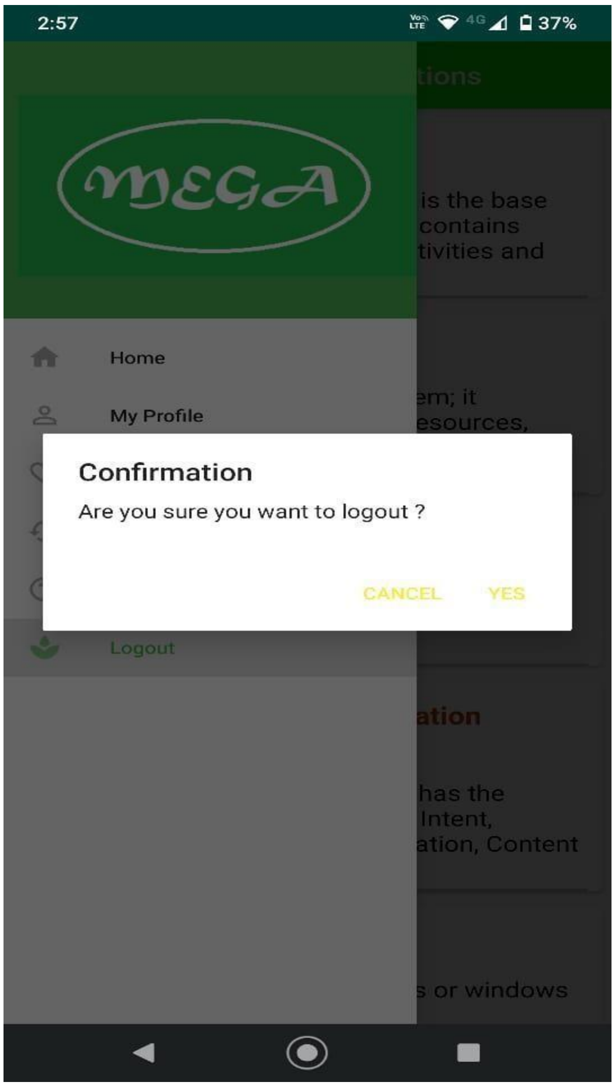
Android application architecture has the following components: Services, Intent, Resource Externalization, Notification, Content Providers.

5. Describe activities

Activities are basically containers or windows to the user interface.

Figure 7.10: FAQ’s

- Displays Frequently Asked Questions in a list manner.



Confirmation

Are you sure you want to logout ?

Figure 7.11: Logout Confirmation

- Displays an Alert Dialog for confirmation to logout from the App

CHAPTER 8

PROJECT\SOURCE CODE

❖ Splash Screen

```
package com.Mega_App.Food_Delivery_App.activity


import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.provider.Settings
import androidx.appcompat.app.AlertDialog
import com.Mega_App.Food_Delivery_App.R
import com.Mega_App.Food_Delivery_App.util.ConnectionManager


class SplashActivity : AppCompatActivity() {


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)


        Handler().postDelayed({
            if(ConnectionManager().checkConnectivity(this)){
                val startAct = Intent(this@SplashActivity,
                    LoginActivity::class.java)
                startActivity(startAct)
                finish()
            } else {
                val dialog = AlertDialog.Builder(this)
                dialog.setTitle("ERROR")
                dialog.setMessage("Internet Connection Not Found")
            dialog.setPositiveButton("Open Settings"){text, listener ->
                val settingIntent = Intent(Settings.ACTION_WIRELESS_SETTINGS)
                startActivity(settingIntent)
                finish()
            }
            dialog.setNegativeButton("Cancel"){text, listener ->
                finish()
            }
            dialog.create().show()
        }
    },1500)
}
  }
```

❖ Registration

```
package com.Mega_App.Food_Delivery_App.activity

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.Settings
import android.widget.Button
import android.widget.EditText
import android.widget.ImageView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.core.app.ActivityCompat
import com.android.volley.Response
import com.android.volley.toolbox.JsonObjectRequest
import com.android.volley.toolbox.Volley
import com.Mega_App.Food_Delivery_App.R
import com.Mega_App.Food_Delivery_App.util.ConnectionManager
import org.json.JSONException
import org.json.JSONObject

class RegistrationActivity : AppCompatActivity() {

    lateinit var etName: EditText
    lateinit var etEmail: EditText
    lateinit var etMobileNumber: EditText
    lateinit var etAddress: EditText
    lateinit var etPassword: EditText
    lateinit var etCnfPassword: EditText
    lateinit var btnNext : Button
    lateinit var imgBack : ImageView
    lateinit var sharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_registration)

        title = "Registration Page"

        sharedPreferences =
getSharedPreferences(getString(R.string.preference_file_name),
Context.MODE_PRIVATE)
        btnNext = findViewById(R.id.btnNext)
        imgBack = findViewById(R.id.imgBackArrow)
        etEmail = findViewById(R.id.etEmail)
        etName = findViewById(R.id.etName)
        etAddress = findViewById(R.id.etAddress)
        etPassword = findViewById(R.id.etPassword)
        etCnfPassword = findViewById(R.id.etConfirmPwd)
        etMobileNumber = findViewById(R.id.etMobileNumber)

        btnNext.setOnClickListener {
            val mobileNumber = etMobileNumber.text.toString()
            val email = etEmail.text.toString()
            val name = etName.text.toString()
            val address = etAddress.text.toString()
            val pwd = etPassword.text.toString()
            val cnfPwd = etCnfPassword.text.toString()

            val intent = Intent(this@RegistrationActivity, MainActivity::class.java)
            val queue = Volley.newRequestQueue(this)
            val url = "http://13.235.250.119/v2/register/fetch_result/"
            val jsonParams = JSONObject().put("name",name).put("mobile_number",
mobileNumber).put("password",pwd).put("address",address).put("email",email)
            if(ConnectionManager().checkConnectivity(this)){

                val jsonObjectRequest = object : JsonObjectRequest(Method.POST, url,
jsonParams, Response.Listener {

                    try {
                        val res = it.getJSONObject("data")
                        val success = res.getBoolean("success")
                        if(success){
                            val data = res.getJSONObject("data")
                            savePreferences(data)
                            startActivity(intent)
                            finish()
                        } else {
                            Toast.makeText(this, "Registration Error! Please try again.",
Toast.LENGTH_SHORT).show()
                        }
                    } catch (e: JSONException){
                        Toast.makeText(this, "Some unexpected error occurred!!!",
Toast.LENGTH_SHORT).show()
                    }

                }, Response.ErrorListener {
                    Toast.makeText(this, "Volley error occurred!!!",
Toast.LENGTH_SHORT).show()
            }) {
                override fun getHeaders(): MutableMap<String, String> {
                    val headers = HashMap<String,String> ()
                    headers["Content-type"] = "application/json"
                    headers["token"] = "64dd3ad5877c86"
                    return headers
                }
            }
            queue.add(jsonObjectRequest)

        } else {
            val dialog = AlertDialog.Builder(this)
            dialog.setTitle("ERROR")
            dialog.setMessage("Internet Connection Not Found")
            dialog.setPositiveButton("Open Setting"){text, listener ->
                val settingIntent = Intent(Settings.ACTION_WIRELESS_SETTINGS)
                startActivity(settingIntent)
                finish()
            }
            dialog.setNegativeButton("Exit"){text, listener ->
                ActivityCompat.finishAffinity(this)
            }
            dialog.create().show()
        }
    }

    imgBack.setOnClickListener {
        val intent = Intent(this, LoginActivity::class.java)
        startActivity(intent)
        finish()
    }
}

fun savePreferences(data: JSONObject) {
```

```
            sharedPreferences.edit().putBoolean("isLoggedIn", true).apply()
            sharedPreferences.edit().putString("user_id",data.getString("user_id")).apply()
            sharedPreferences.edit().putString("name",data.getString("name")).apply()
            sharedPreferences.edit().putString("email",data.getString("email")).apply()

        sharedPreferences.edit().putString("mobile_number",data.getString("mobile_number
        ")).apply()
            sharedPreferences.edit().putString("address",data.getString("address")).apply()
        }
    }
```

❖ Main Activity

```
package com.Mega_App.Food_Delivery_App.activity

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.MenuItem
import android.widget.FrameLayout
import androidx.appcompat.app.ActionBarDrawerToggle
import androidx.appcompat.app.AlertDialog
import androidx.coordinatorlayout.widget.CoordinatorLayout
import androidx.core.view.GravityCompat
import androidx.drawerlayout.widget.DrawerLayout
import com.google.android.material.navigation.NavigationView
import com.Mega_App.Food_Delivery_App.*
import com.Mega_App.Food_Delivery_App.fragment.*

class MainActivity : AppCompatActivity() {

    lateinit var sharedPreferences: SharedPreferences
    lateinit var drawerLayout : DrawerLayout
    lateinit var coordinatorLayout : CoordinatorLayout
    lateinit var toolbar : androidx.appcompat.widget.Toolbar
    lateinit var frameLayout: FrameLayout
    lateinit var navigationView: NavigationView

    var previousMenuItem : MenuItem? = null

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        sharedPreferences = getSharedPreferences(getString(R.string.preference_file_name),
Context.MODE_PRIVATE)
        drawerLayout = findViewById(R.id.drawerLayout)
        coordinatorLayout = findViewById(R.id.coordinatorLayout)
        toolbar = findViewById(R.id.toolbar)
        frameLayout = findViewById(R.id.frame)
        navigationView = findViewById(R.id.navigationView)
        setUpToolbar()
        openHome()

        val actionBarDrawerToggle = ActionBarDrawerToggle(this, drawerLayout,
            R.string.open_drawer,
            R.string.close_drawer
        )
        drawerLayout.addDrawerListener(actionBarDrawerToggle)
        actionBarDrawerToggle.syncState()

        navigationView.setNavigationItemSelectedListener {

            if(previousMenuItem != null){
                previousMenuItem?.isChecked = false
            }
            it.isCheckable = true
            it.isChecked = true
            previousMenuItem = it
```

```
when(it.itemId){
    R.id.itemHome -> {
        openHome()
        drawerLayout.closeDrawers()
    }
    R.id.itemProfile -> {
        supportFragmentManager.beginTransaction()
            .replace(
                R.id.frame,
                ProfileFragment()
            )
            .commit()
        supportActionBar?.title = "My Profile"
        drawerLayout.closeDrawers()
    }
    R.id.itemFavourites -> {
        supportFragmentManager.beginTransaction()
            .replace(
                R.id.frame,
                FavouritesFragment()
            )
            .commit()
        supportActionBar?.title = "Favourite Restaurants"
        drawerLayout.closeDrawers()
    }
    R.id.itemHistory -> {
        supportFragmentManager.beginTransaction()
            .replace(
                R.id.frame,
                HistoryFragment()
            )
            .commit()
        supportActionBar?.title = "Order History"
        drawerLayout.closeDrawers()
```

```
            }
            R.id.itemFAQs -> {
                supportFragmentManager.beginTransaction()
                    .replace(
                        R.id.frame,
                        FAQsFragment()
                    )
                    .commit()
                supportActionBar?.title = "Frequently Asked Questions"
                drawerLayout.closeDrawers()
            }
            R.id.itemLogout -> {

                val dialog = AlertDialog.Builder(this)
                dialog.setTitle("Confirmation")
                dialog.setMessage("Are you sure you want to logout ?")
                dialog.setPositiveButton("Yes"){text, listener ->
                    sharedPreferences.edit().clear().apply()
                    val intent = Intent(this, LoginActivity::class.java)
                    startActivity(intent)
                    finish()
                }
                dialog.setNegativeButton("Cancel"){text, listener ->

                }
                dialog.create().show()
            }
        }
        return@setNavigationItemSelectedListener true
    }
}

fun setUpToolbar(){
    setSupportActionBar(toolbar)
    supportActionBar?.title = sharedPreferences.getString("title", "Mega_Food_App")
```

```
        supportActionBar?.setHomeButtonEnabled(true)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        val id = item.itemId
        if(id == android.R.id.home){
            drawerLayout.openDrawer(GravityCompat.START)
        }
        return super.onOptionsItemSelected(item)
    }

    fun openHome(){
        val fragment =
            HomeFragment()
        val transaction = supportFragmentManager.beginTransaction()
        transaction.replace(R.id.frame,fragment).commit()
        supportActionBar?.title = "All Restaurants"
        navigationView.setCheckedItem(R.id.itemHome)
    }

    override fun onBackPressed() {
        val fragment = supportFragmentManager.findFragmentById(R.id.frame)
        when(fragment){
            !is HomeFragment -> openHome()
            else -> super.onBackPressed()
        }
    }

    override fun onPause() {
        super.onPause()
        finish()
    }
}
```

❖ Login

```
package com.Mega_App.Food_Delivery_App.activity

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.provider.Settings
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import androidx.appcompat.app.AlertDialog
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import com.android.volley.Response
import com.android.volley.toolbox.JsonObjectRequest
import com.android.volley.toolbox.Volley
import com.Mega_App.Food_Delivery_App.R
import com.Mega_App.Food_Delivery_App.util.ConnectionManager
import org.json.JSONException
import org.json.JSONObject

class LoginActivity : AppCompatActivity() {

    lateinit var etMobileNumber: EditText
    lateinit var etPassword: EditText
    lateinit var btnLogin: Button
    lateinit var txtForgotPassword: TextView
    lateinit var txtRegisterYourself: TextView
    lateinit var sharedPreferences: SharedPreferences

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        sharedPreferences = getSharedPreferences(getString(R.string.preference_file_name),
Context.MODE_PRIVATE)
        val isLoggedIn = sharedPreferences.getBoolean("isLoggedIn", false)
        setContentView(R.layout.activity_login)

        if (isLoggedIn) {
            val intent = Intent(this@LoginActivity, MainActivity::class.java)
            startActivity(intent)
            finish()
        }

        title = "Log In"

        etMobileNumber = findViewById(R.id.etMobileNumber)
        etPassword = findViewById(R.id.etPassword)
        btnLogin  =  findViewById(R.id.btnLogin)
        txtForgotPassword = findViewById(R.id.txtForgotPassword)
        txtRegisterYourself = findViewById(R.id.txtRegister)

        btnLogin.setOnClickListener {

            val mobileNumber = etMobileNumber.text.toString()
            val password = etPassword.text.toString()

            val intent = Intent(this@LoginActivity, MainActivity::class.java)

            val queue = Volley.newRequestQueue(this)
            val url = "http://13.235.250.119/v2/login/fetch_result/"
            val jsonParams = JSONObject().put("mobile_number",
mobileNumber).put("password",password)
            if(ConnectionManager().checkConnectivity(this)){

                val jsonObjectRequest = object : JsonObjectRequest(Method.POST, url,
jsonParams, Response.Listener {
```

```
                try {
                    val res = it.getJSONObject("data")
                    val success = res.getBoolean("success")
                    if(success){
                        val data = res.getJSONObject("data")
                        savePreferences(data)
                        startActivity(intent)

                    } else {
                        Toast.makeText(this, "Login Error! Please try again.",
Toast.LENGTH_SHORT).show()
                    }
                } catch (e: JSONException){
                    Toast.makeText(this, "Some unexpected error occurred!!!",
Toast.LENGTH_SHORT).show()
                }

            }, Response.ErrorListener {
                    Toast.makeText(this, "Volley error occurred!!!",
Toast.LENGTH_SHORT).show()
            }) {
                override fun getHeaders(): MutableMap<String, String> {
                    val headers = HashMap<String,String> ()
                    headers["Content-type"] = "application/json"
                    headers["token"] = "64dd3ad5877c86"
                    return headers
                }
            }
            queue.add(jsonObjectRequest)

        } else {
            val dialog = AlertDialog.Builder(this)
            dialog.setTitle("ERROR")
            dialog.setMessage("Internet Connection Not Found")
            dialog.setPositiveButton("Open Setting"){text, listener ->
```

```
                val settingIntent = Intent(Settings.ACTION_WIRELESS_SETTINGS)
                startActivity(settingIntent)
                finish()
            }
            dialog.setNegativeButton("Exit"){text, listener ->
                ActivityCompat.finishAffinity(this)
            }
            dialog.create().show()
        }

    }

    txtForgotPassword.setOnClickListener {
        val intent = Intent(this@LoginActivity, PasswordActivity::class.java)
        startActivity(intent)
    }

    txtRegisterYourself.setOnClickListener {
        val intent = Intent(this@LoginActivity, RegistrationActivity::class.java)
        startActivity(intent)
    }

}

override fun onPause() {
    super.onPause()
    finish()
}

fun savePreferences(data: JSONObject) {
    sharedPreferences.edit().putBoolean("isLoggedIn", true).apply()
    sharedPreferences.edit().putString("user_id",data.getString("user_id")).apply()
    sharedPreferences.edit().putString("name",data.getString("name")).apply()
    sharedPreferences.edit().putString("email",data.getString("email")).apply()
sharedPreferences.edit().putString("mobile_number",data.getString("mobile_number")).appl
y()
        sharedPreferences.edit().putString("address",data.getString("address")).apply()
    }
}
```

❖ Description

```
package com.Mega_App.Food_Delivery_App.activity

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.provider.Settings
import android.view.MenuItem
import android.view.View
import android.widget.*
import androidx.appcompat.app.AlertDialog
import androidx.core.app.ActivityCompat
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.android.volley.Response
import com.android.volley.toolbox.JsonObjectRequest
import com.android.volley.toolbox.Volley
import com.Mega_App.Food_Delivery_App.R
import com.Mega_App.Food_Delivery_App.adapter.DescriptionRecyclerAdapter
import com.Mega_App.Food_Delivery_App.adapter.HomeRecyclerAdapter
import com.Mega_App.Food_Delivery_App.database.RestaurantEntity
import com.Mega_App.Food_Delivery_App.model.Dish
import com.Mega_App.Food_Delivery_App.util.ConnectionManager
import org.json.JSONException

class DescriptionActivity : AppCompatActivity() {

    var restaurantId: String? = "0"
    var restaurantName: String? = "0"
```

```
lateinit var recyclerDescription : RecyclerView
lateinit var layoutManager: LinearLayoutManager
var dishList = arrayListOf<Dish>()
var orderList = arrayListOf<Dish>()
lateinit var recyclerAdapter: DescriptionRecyclerAdapter
lateinit var progressLayout : RelativeLayout
lateinit var progressBar: ProgressBar
lateinit var restaurantEntity: RestaurantEntity

lateinit var toolbar : androidx.appcompat.widget.Toolbar
lateinit var imgHeart : ImageView
lateinit var btnProceed : Button

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_description)

    recyclerDescription = findViewById(R.id.recyclerDescription)
    layoutManager = LinearLayoutManager(this)
    progressLayout = findViewById(R.id.progressLayout)
    progressBar = findViewById(R.id.progressBar)

    toolbar = findViewById(R.id.toolbar)
    imgHeart = findViewById(R.id.imgHeart)
    btnProceed = findViewById(R.id.btnProceedToCart)
    setUpToolbar()

    if(intent != null){
        restaurantId = intent.getStringExtra("id")
        restaurantName = intent.getStringExtra("name")
        toolbar.title = intent.getStringExtra("name")
        restaurantEntity = RestaurantEntity(
            restaurantId = intent.getStringExtra("id")!!,
            restaurantName = intent.getStringExtra("name")!!,
            restaurantPrice = intent.getStringExtra("price")!!,
            restaurantRating = intent.getStringExtra("rating")!!,
            restaurantImage = intent.getStringExtra("image")!!
        )

    } else {
        finish()
        Toast.makeText(this, "Some unexpected error
occurred!!!",Toast.LENGTH_SHORT).show()
    }
    if(restaurantId == "0"){
        finish()
        Toast.makeText(this, "Some unexpected error
occurred!!!",Toast.LENGTH_SHORT).show()
    }

    putImage()
    imgHeart.setOnClickListener {
        clickHeart()
    }

    val queue = Volley.newRequestQueue(this)
    val url = "http://13.235.250.119/v2/restaurants/fetch_result/$restaurantId/"

    if(ConnectionManager().checkConnectivity(this)){

        val jsonObjectRequest = object : JsonObjectRequest(Method.GET, url, null,
Response.Listener {

            try {
                progressLayout.visibility = View.GONE

                val res = it.getJSONObject("data")
                val success = res.getBoolean("success")
                if(success){
                    val data = res.getJSONArray("data")
                    for( i in 0 until data.length()){
                        val dishJsonObject = data.getJSONObject(i)
                        val dishObject = Dish(
                            dishJsonObject.getString("id"),
                            dishJsonObject.getString("name"),
                            dishJsonObject.getString("cost_for_one"),
                            dishJsonObject.getString("restaurant_id")
                        )
                        dishList.add(dishObject)
                    }
                    recyclerAdapter = DescriptionRecyclerAdapter(this, dishList, orderList)

                    recyclerDescription.adapter = recyclerAdapter
                    recyclerDescription.layoutManager = layoutManager
//
recyclerDescription.addItemDecoration(DividerItemDecoration(recyclerDescription.conte
xt, layoutManager.orientation))
                } else {
                    Toast.makeText(this, "Some Error Occurred",
Toast.LENGTH_SHORT).show()
                }
            } catch (e: JSONException){
                Toast.makeText(this, "Some unexpected error occurred!!!",
Toast.LENGTH_SHORT).show()
            }


        }, Response.ErrorListener {
            Toast.makeText(this, "Volley error occurred!!!",
Toast.LENGTH_SHORT).show()
        }) {
            override fun getHeaders(): MutableMap<String, String> {
                val headers = HashMap<String,String> ()
                headers["Content-type"] = "application/json"
                headers["token"] = "64dd3ad5877c86"
                return headers
            }
        }
        queue.add(jsonObjectRequest)

    } else {
        val dialog = AlertDialog.Builder(this)
        dialog.setTitle("ERROR")
        dialog.setMessage("Internet Connection Not Found")
        dialog.setPositiveButton("Open Setting"){text, listener ->
            val settingIntent = Intent(Settings.ACTION_WIRELESS_SETTINGS)
            startActivity(settingIntent)
            finish()
        }
        dialog.setNegativeButton("Exit"){text, listener ->
            ActivityCompat.finishAffinity(this)
        }
        dialog.create().show()
    }

    btnProceed.setOnClickListener {
        if(orderList.size > 0){
            val intent = Intent(this, CartActivity::class.java)
            intent.putParcelableArrayListExtra("order",
orderList).putExtra("resName",restaurantName)
            startActivity(intent)
        } else {
            Toast.makeText(this, "Select atleast one dish on menu!",
Toast.LENGTH_SHORT).show()
        }
    }
}

fun setUpToolbar(){
    setSupportActionBar(toolbar)
    supportActionBar?.setHomeButtonEnabled(true)
    supportActionBar?.setDisplayHomeAsUpEnabled(true)
}

fun putImage(){
    val checkFav = HomeRecyclerAdapter.DBAsyncTask(this, restaurantEntity,
1).execute()
    val isFav = checkFav.get()
    if(isFav) {
        imgHeart.setImageResource(R.drawable.ic_heart)
    } else {
        imgHeart.setImageResource(R.drawable.ic_heart_border)
    }
}

fun clickHeart(){
    if(!HomeRecyclerAdapter.DBAsyncTask(this, restaurantEntity, 1).execute().get()){
        val result = HomeRecyclerAdapter.DBAsyncTask(this, restaurantEntity,
2).execute().get()
        if(result) {
            Toast.makeText(this,"Added to favourites",Toast.LENGTH_SHORT).show()
            imgHeart.setImageResource(R.drawable.ic_heart)
            recyclerAdapter.notifyDataSetChanged()
        } else {
            Toast.makeText(this,"Some error occurred",Toast.LENGTH_SHORT).show()
        }
    } else {
        val async = HomeRecyclerAdapter.DBAsyncTask(this, restaurantEntity,
3).execute()
        val result = async.get()
        if(result) {
            Toast.makeText(this,"Removed from
favourites",Toast.LENGTH_SHORT).show()
            imgHeart.setImageResource(R.drawable.ic_heart_border)
            recyclerAdapter.notifyDataSetChanged()
        } else {
            Toast.makeText(this,"Some error occurred",Toast.LENGTH_SHORT).show()
        }
    }
}

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    val id = item.itemId
    if(id == android.R.id.home){
        onBackPressed()
    }
    return super.onOptionsItemSelected(item)
}

override fun onBackPressed() {
    super.onBackPressed()
    startActivity(Intent(this, MainActivity::class.java))
    finish()
}
}
```

CHAPTER 9

CONCLUSION

Technology has made significant progress over the years to provide customers a better online food ordering experience and will continue to do so for years to come. With the rapid growth of online food delivery market, people have speculated that online food ordering will overtake in-restaurant food ordering. While this has been the case in some areas, there is still demand for restaurants in market areas where the consumer feels more comfortable seeing and touching the product being bought. However, the availability of online food ordering has produced a more educated consumer that can order food around with relative ease without having to spend a large amount of time. In exchange, online food delivery has opened up doors to many small food vendors that would never be in business if they had to incur the high cost of owning a restaurant. At the end, it has been a win-win situation for both customer and restaurant owners or small food vendors. For potential customers, the app enables them to browse before they order food, and to research about the restaurant from which they are ordering food online. It overcomes the disadvantages of the traditional queuing system. This system increases the takeaway of foods than visitors.Therefore, this system enhances the speed and standardization of taking the order from the customer. It provides a better communication platform. the user’s details are noted electronically. Using this application, the customers need not go to the restaurant by themselves, but they can order the dishes through Android mobiles anywhere.

CHAPTER 10

FUTURE ENHANCEMENT

Payment Gateway

With the online mobile payment feature ordering food using restaurant based apps has become easier these days. There occurs no requirement to make use of cash. One can order food online using online payment modes right from the restaurant ordering app.Customers can also save up payment-related information in their profiles. Hence, ordering repeatedly is hassle-free; there is no need to add their account details, again and again. With a single button, one can order food online!

Food Pre-Ordering

There is a feature called ‘Advance Order’ or ‘Food Pre-Ordering’ which allows users to schedule their order’s delivery time. With the help of the food pre-ordering feature, customers get the freedom of choosing delivery or pickup time, at the time of placing their orders. Customers can select their usual order to be delivered immediately or set a particular time for future delivery. The restaurant is immediately notified about your customers’ preferred schedule.

Push Notifications

With a restaurant app, you can improve your customer engagement by sending push-notifications. This feature allows you to post updates on the latest deals, offers, and discounts through the right to your customers’ mobile phones.Push notifications are also a great marketing tool as it comes handy in updating your customers about new items, new product updates, and new combo, etc., introduced to the restaurant menu.

Chat Bot

Food Delivery Chatbot is deployed on food delivery app to help food lovers to order delicious food blissfully from their nearby restaurants and cafes. Chatbot will start the conversation by asking user a set of questions to discover restaurants around him on the basis of the food, rating, price and review. Further chatbot will ask customer’s details such as delivery address, name along with payment details to complete the order. Post delivery chatbot can be used to take feedback on the food, taste, quality service etc. to measure the quality of service. The bot can also answer commonly asked customer queries instantly.

Tracking Order

An order tracking system is one that tracks foods from the moment the order is placed to when they are physically delivered to the destination location. Through this feature customers can check themselves where their order is and when it will be delivered.

REFERNCES

[1] https://nevonprojects.com/android-food-order-and-delivery-app/

[2] https://developer.android.com/

[3] https://www.tutorialspoint.com/android/index.htm

[4] https://www.geeksforgeeks.org/android-tutorial/

[5] https://www.javatpoint.com/android-tutorial