International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

FOOD ORDERING APPLICATION

Ruturaj Patil[a], Vishwajeet Patil[a], Rohini Bandgar[a], Pratik Patil[a], S.A.Salunkhe[a].

Student,Sanjay Ghodawat Polytechnic,Atigre,India
Faculty(C.S.E),Sanjay Ghodawat Polytechnic,Atigre,India

A B S T R A C T

Increased demand of restaurant-goers-food court generated the need for much attention for the hospitality industry. Providingmuch option with ease of ordering is the need of the hours. Technological interference has become mandatory to improve the quality of the service and business in this industry. Evidences are already existed for partial automation of food ordering process in the country; most of these technologies implemented are based on wireless technologies. This manuscript reports implementation and integration of android based technology for food court. A static database utility system was designed to fetchall the information from a centralized database. User utility was given importance during the development of this interface and efficiency, accuracy was the priority for better results and services and to reduce the majority of the human error. It was observedthat this system was successful in overcoming the shortcomings found in the previously developed similar systems.

***Keywords**: Android app*

1. INTRODUCTION

The earlier food ordering system was entirely a manual process which involved waiters, pen and paper. The waiter had to note down orders from customers, take these orders to kitchen, update them in records and again make bill. Even though this system is simple it may involve human errors in noting down the orders. There are many reasons leading to the feeling of dissatisfaction including being entertained late in terms of order taking by the waiter and meals serving. To overcome these limitations in manual system, multi-touchable restaurant management system is proposed in this paper to automate food ordering process. The restaurant with automated food ordering system will be equipped with a user- friendly touch screen, display screen in the kitchen, and software for completing the process at the back-end. For this system there will be a system administrator who will have the rights to enter the menu with their current prevailing prices. The system administrator can enter anytime in the system by a secured system password to change the menu contents by adding or deleting an item or changing its price.

The proposed system enables the end users to register on the application, select the food from menu card and order food by an android app. The user is given Username and Password to Login. The User can see the list of food items in food court. From that end user can able to select their food items and place order to cart.

2. LITERATURE REVIEW

Study of Online Food Delivery App like Zomato & Swiggy and their effect on Casual Dining:

Paper describes an online food menu is set up by the proposed food ordering system and as per their will customers can easily place the order. Also, customers can easily track the orders with the food menu. The management improve food delivery service and preserves customers database. Motivation to develop thesystem is from the restaurant management system. To get the services efficiently the users of the system provides various facilities. Restaurants as well as Mess facility is considered by our system for the customers.Mostly mess users are person who are shifted to new cities and this can be considered as a motivation to oursystem. Another motivation can be considered as the increasing use of smart phones by the customers, so thatany users of this system get all service of the system. The system will be designed to avoid users doing fatalerrors where users can change their own profile also where users can track their food items.

Automated Food Ordering System:

The research work aims to automate the food ordering process in restaurant and also improve the dining experience of customers. Design implementation of food ordering system for restaurants were discuss in this paper. This system implements wireless data access to servers. The android application on user’s mobile will have all the menu details. Kitchen and cashier receives the order details from the customer mobile wirelessly. These order details are updated in the central database. The restaurant owner can manage the menu modifications easily.

3. OBJECTIVE AND SCOPE

Objective of project:

- Develop the store and maintenance computerized system from the previously existingsystem.
- It handles the entire information about the users securely.
- Better error handling.
- To study food ordering system.

Scope of project:

This online food ordering system project aimed at developing an online food ordering system that canbe used in small places, and medium cities firstly and then on a large scale. It is developed to help food court to simplify their daily operational and managerial task as well as improve the dining experience of customers. And also helps food courts, restaurants develop healthy customer relationships by providing good services. The system enables staff to let update and make changes totheir food and beverage list information based on the orders placed and the orders completed.

4. CORE TECHNOLOGY

Project structure:

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modulesinclude:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays your project files in the Android project view. This view is organized by modules to provide quick access to your project's key source files. All the build files are visible at the top level under Gradle Scripts andeach app module contains the following folders:

- **Manifests**: Contains the AndroidManifest.xml file.
- **Java**: Contains the Java source code files, including JUnit test code.
- **Res**: Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

Features of Android Studio

- It has a flexible Gradle-based build system.
- It has a fast and feature-rich emulator for app testing.
- Android Studio has a consolidated environment where we can develop for all Android devices.
- Apply changes to the resource code of our running app without restarting the app.
- Android Studio provides extensive testing tools and frameworks.
- It supports C++ and NDK.
- It provides build-in supports for Google Cloud Platform. It makes it easy to integrate Google Cloud Messaging and AppEngine.

Gradle build system

Gradle build used as the foundation of the build system in Android Studio. It uses more Android-specific capabilities provided by the Android plugin for Gradle. This build system runs independently from the command line and integrated tool from the Android Studio menu. We can use build features for the following purpose:

- Configure, customize, and extend the build process.

- We can create multiple APKs from our app, with different features using the same project and modules.
- Reuse resource and code across source sets.

5. XML LANGUAGE

XML stands for Extensible Markup Language, which gives us a clue to what it does. A markup language is slightly different from a programming language. Whereas a programming language (C#, C++, Java, Kotlin, Python, BASIC) will allow you to define behaviors, interacti ons, and conditions; a markup language is used more to describe data, and in this case, layouts. Programming languages create dynamic interactions, whereas markup languages generally handle things like static user interfaces. When you create a new project in Android Studio, you will be greeted by a hierarchy of different files and folders, which can be a little daunting for complete beginners. It's a rather jarring introduction to XML.

Xml in Android App

XML describes the views in your activities, and Java tells them how to behave. To make changes to the layout of your app then, you have two main options. The first is to use the Design view. Open up the activity main.xml file in Android Studio and get your first introduction to XML. You'll notice there are two tabs at the bottom of that window: Design and Text. The Text view will show you the actual XML code, but the Design view will let you manually edit the layout by dragging and dropping elements into the render of your activity. XML files can also help store strings. Using the Design view is easier for beginners, though it can lead to complications. For one, you will run into the limitations of XML early on when the designer refuses to let you drop items into specific places. Without the knowledge of why, this can make designing your app an exercise in frustration.

Used Layouts and Containers

Android Layout is used to define the user interface that holds the UI controls or widgets that will appear on the screen of anandroid application or activity screen. Generally, every application is a combination of View and ViewGroup. As we know,an android application contains a large number of activities and we can say each activity is one page of the application. Allthe elements in a layout are built using a hierarchy of View and ViewGroup objects.

- **RelativeLayout:** RelativeLayout is a view group that displays child views in relative positions. The position of each viewcan be specified as relative to sibling elements (such as to the left-of or below another view) or in positions relative to the parent RelativeLayout area (such as aligned to the bottom, left or center).
- **Linear Layout**: Linear Layout is a view group that aligns all children in a single direction, vertically or horizontally. You can specify the layout direction with the android:orientation attribute.
- **androidx.cardview.widget.CardView**: Apps often need to display data in similarly styled containers. These containers are often used in lists to hold each item's information. The system provides the CardView API as an easy way for you to show information inside cards that have a consistent look across the platform.
- **FrameLayout**: FrameLayout is a ViewGroup subclass, used to specify the position of View elements it contains on the topof each other to display only a single View inside the FrameLayout.
- **ConstraintLayout**: ConstraintLayout is a ViewGroup subclass, used to specify the position of layout constraints for everychild View relative to other views present. A ConstraintLayout is similar to a RelativeLayout, but having more power.
- **View**: A View is defined as the user interface which is used to create interactive UI components such as TextView, ImageView, EditText, RadioButton, etc., and is responsible for event handling and drawing. They are Generally Called Widgets.

Used Components

- **TextView**: A TextView displays text to the user and optionally allows them to edit it. A TextView is a complete text editor,however the basic class is configured to not allow editing.
- **TextInputLayout**: TextInputLayout is a view container that is used to add more features to an EditText. It acts as a wrapper for EditText and has some features like: Floating hint. Animation that can be disabled or enabled.
- **EditText**: A EditText is an overlay over TextView that configures itself to be editable. It is the predefined subclass of TextView that includes rich editing capabilities.
- **Button**: A button consists of text or an icon (or both text and an icon) that communicates what action occurs when theuser touches it.

- **ImageButton**: Image Button is a user interface control that is used to display a button with an image and to perform anaction when a user clicks or taps on it.
- **ImageView**: Displays image resources, for example Bitmap or Drawable resources. ImageView is also commonly used toapply tints to an image and handle image scaling

6. DEVELOPMENT

Java

Android development is a popular buzz in the Java programming world. It’s Android, which keeps Java at the forefront of the last couple of years. Now, How important is it to understand or learn Android for Java programmers? Well, it depends on, if you like application development and wants to reach a mass, Android offers that opportunity to you. Millions of Android phones are available and they keep increasing, with pace, much higher than iPhone or iOS. What all this means is,it does make a lot of sense to learn Android programming for Java programmers, and this article is about that, but this is also one of the good reasons to learn Java programming. This tutorial will give you a basic idea of How Android works? not detailed but a good overview.

Used Listener

- **OnClickListener**: In Android, the OnClickListener() interface has an onClick(View v) method that is called when theview (component) is clicked. The code for a component’s functionality is written inside this method, and the listener is set using the setOnClickListener() method.
- **ValueEventListener**: public interface ValueEventListener. Classes implementing this interface can be used to receive events about data changes at a location. Attach the listener to a location us addValueEventListener(ValueEventListener) .
- **OnSuccessListener**: public abstract void onSuccess (ResultT result). Called when the Task completes successfully.
- **addOnFailureListener**: Listener called when a Task fails with an exception OnCompleteListener: public abstract void onComplete (Task¡ResultT¿ task). Called when the Task completes

7. FIREBASE DATABASE

INTRODUCTION TO FIREBASE

Firebase is a toolset to “build, improve, and grow your app”, and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don’t really want to build, because they’d rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer. The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtimeto every connected client. When you build cross-platform apps with our Apple platforms, Android, and JavaScript SDKs, allof your clients share one Realtime Database instance and automatically receive updates with the newest data.

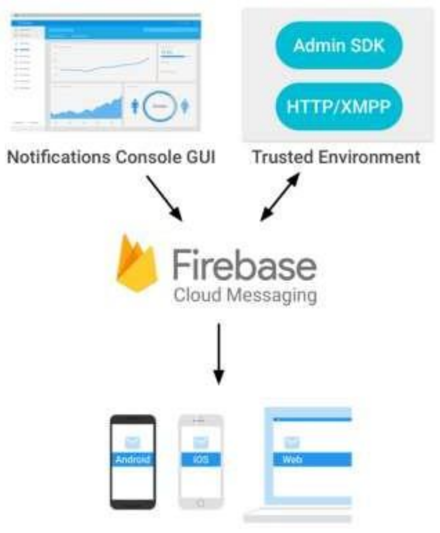


Figure 7.1. Firebase Diagram

8. UML DIAGRAMS

FLOWCHART:

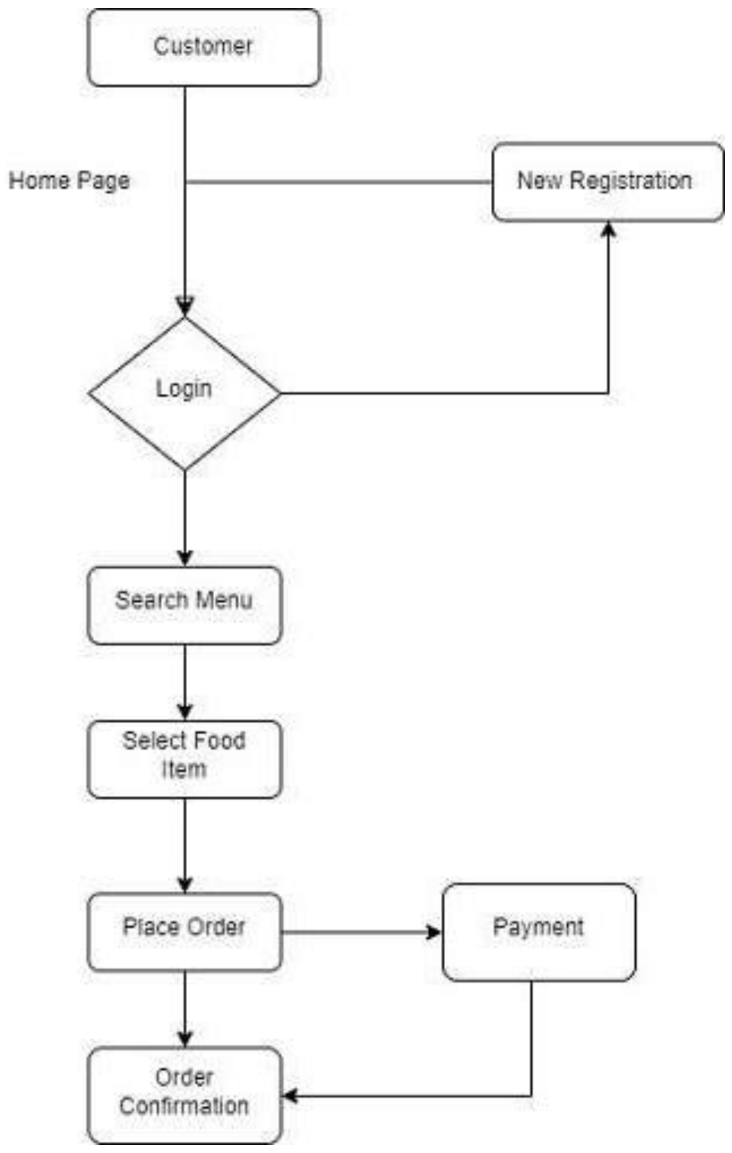
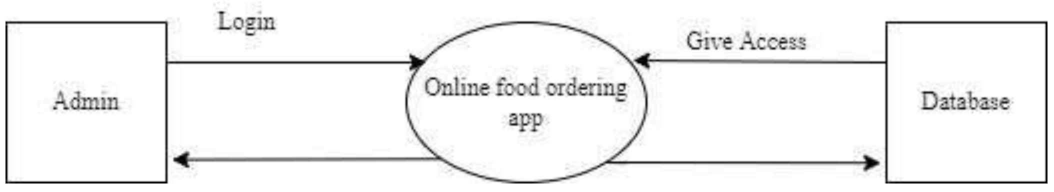
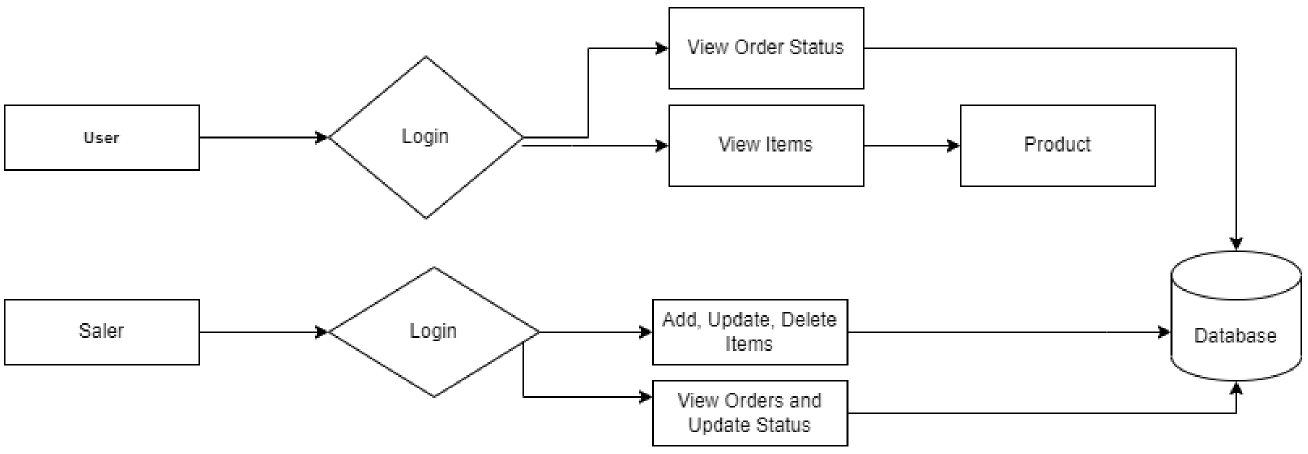


Figure 8.1.1 Flowchart

DATA FLOW DIAGRAMS:



DFD Level 0



DFD Level 1

9. SYSTEM CODING

LoginActivity.java :

package com.example.foodcourt.activities;import androidx.annotation.NonNull;

import androidx.appcompat.app.AlertDialog;

import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;import android.content.Intent;

import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.EditText;

import android.widget.ProgressBar;import android.widget.TextView; import android.widget.Toast;

import com.example.foodcourt.MainActivity;import com.example.foodcourt.R;

import com.google.android.gms.tasks.OnCompleteListener;import com.google.android.gms.tasks.OnFailureListener;

import com.google.android.gms.tasks.OnSuccessListener; import com.google.android.gms.tasks.Task;

import com.google.firebase.auth.AuthResult; import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends AppCompatActivity {Button loginbutton;

TextView register,forgetpassword;EditText username, password; FirebaseAuth fAuth;

ProgressBar progressBar;

@Override

protected void onCreate(Bundle savedInstanceState) {super.onCreate(savedInstanceState); setContentView(R.layout.activity_login);

loginbutton = findViewById(R.id.logb1); register = findViewById(R.id.logtv2); username = findViewById(R.id.resusername); password = findViewById(R.id.respassword);

forgetpassword = findViewById(R.id.logforgetpass);

fAuth = FirebaseAuth.getInstance();

```
progressBar = findViewById(R.id.lprogressBar);

loginbutton.setOnClickListener(new View.OnClickListener() {@Override

public void onClick(View view) {

String username_ = username.getText().toString();String password_ = password.getText().toString();

if (!username_.isEmpty()) {

if (username_.matches("^(.+)@(.+)$")) {username.setError(null);

username.setEnabled(false);if (!password_.isEmpty()) {

if (password_.length() >= 8) {password.setError(null); password.setEnabled(false);

progressBar.setVisibility(View.VISIBLE);

fAuth.signInWithEmailAndPassword(username_,          password_).addOnCompleteListener(newOnCompleteListener<AuthResult>() {

@Override

public void onComplete(@NonNull Task<AuthResult> task) {if (task.isSuccessful()) {

Toast.makeText(getApplicationContext(), "Login Successfully", Toast.LENGTH_SHORT).show();progressBar.setVisibility(View.GONE);

Intent intent = new Intent(getApplicationContext(), MainActivity.class);startActivity(intent);

finish();

} else {

Toast.makeText(getApplicationContext(), "Login unsuccessfully", Toast.LENGTH_SHORT).show();progressBar.setVisibility(View.GONE);

startActivity(new Intent(LoginActivity.this, RegistrationActivity.class));

}

}

});

} else {

password.setError("please enter at lest 8 charactor password");

}

} else {

password.setError("please enter the  password");

}

} else {

username.setError("Invalid email");

}

} else {

username.setError("please enter the  username");

}

}
});

forgetpassword.setOnClickListener(new View.OnClickListener() {@Override

public void onClick(View view) {

EditText resetMail = new EditText(view.getContext());

AlertDialog.Builder passworedResetDialog = new AlertDialog.Builder(view.getContext()); passworedResetDialog.setTitle("Reset Password ?");
passworedResetDialog.setMessage("Enter Your Email To Received Reset Password Linnk.");passworedResetDialog.setView(resetMail);

passworedResetDialog.setPositiveButton("Yes", new DialogInterface.OnClickListener() {@Override

public void onClick(DialogInterface dialogInterface, int i) {

String mail = resetMail.getText().toString(); fAuth.sendPasswordResetEmail(mail).addOnSuccessListener(new OnSuccessListener<Void>() {
@Override

public void onSuccess(Void unused) {

Toast.makeText(getApplicationContext(), "Reset Link Sent To Your Email.", Toast.LENGTH_SHORT).show();

}

}).addOnFailureListener(new OnFailureListener() {@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(getApplicationContext(), "Error...! Reset Link is not Sent."+ e.getMessage(),
Toast.LENGTH_SHORT).show();

}

});



}

});

passworedResetDialog.setNegativeButton("No", new DialogInterface.OnClickListener() {@Override

public void onClick(DialogInterface dialogInterface, int i) {

}

});

passworedResetDialog.create().show();

}

});

}

public void register(View view) {

startActivity(new Intent(LoginActivity.this, RegistrationActivity.class));

}

}
```

activity_login.xml:

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".activities.LoginActivity">


<ImageView
    android:id="@+id/
    logbg"

    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:foreground="@drawable/foreground_design"
    android:scaleType="centerCrop"
    android:src="@drawable/loginbg"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="1.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.0" />


<ImageView
    android:id="@+id/logfclogo"
    android:layout_width="wrap_conte
    nt"
    android:layout_height="wrap_cont
    ent"

    app:layout_constraintBottom_toBottomOf="@+id/logbg"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"

    app:layout_constraintVertical_bias="0.0"
    app:srcCompat="@drawable/food_court"
    />

<ImageView
    android:id="@+id/loglog
    o"
    android:layout_width="3
    19dp"
    android:layout_height="9
    2dp"

    app:layout_constraintBottom_toBottomOf="@+id/logfclogo"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.674"
    app:srcCompat="@drawable/login" />


<EditText
    android:id="@+id/resusername"
    android:layout_width="0dp"
    android:layout_height="wrap_cont
    ent"
    android:backgroundTint="#CCCC
    CC"
    android:drawableLeft="@drawable/ic_baseline_email_24"
    android:hint="Email Id"

    android:minHeight="48dp" android:textColor="@color/white"
    android:textColorHint="@color/white"
    app:layout_constraintBottom_toBottomOf="@+id/logbg"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/loglogo"
    app:layout_constraintVertical_bias="0.022"
    app:layout_constraintWidth_percent=".8" />

<EditText
    android:id="@+id/respassword"
    android:layout_width="0dp"
    android:layout_height="wrap_cont
    ent"
    android:backgroundTint="#CCCC
    CC"
    android:layout_marginTop="10sp"

    android:drawableLeft="@drawable/ic_baseline_vpn_key_24"
    android:hint="@string/password"
    android:inputType="textPassword" android:minHeight="48dp"
    android:textColorHint="@color/white"
    android:textColor="@color/white"
    app:layout_constraintBottom_toBottomOf="@+id/logbg"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.493"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/resusername"
    app:layout_constraintVertical_bias="0.0"
    app:layout_constraintWidth_percent=".8" />

<ProgressBar
    android:id="@+id/lprogressBar"

    style="?android:attr/progressBarStyleHorizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:max="100"

    android:progress="50"
    android:progressTint="#F44336"
    android:visibility="invisible"
    app:layout_constraintBottom_toTopOf="@+id/logb1"
    app:layout_constraintEnd_toEndOf="parent"


    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/logfclogo"
    app:layout_constraintVertical_bias="0.881" />



<Button

    android:id="@+id/logb1"
    android:layout_width="2
    29dp"
    android:layout_height="4
    8dp"
    android:onClick="Login"
    android:text="Log In"
    android:textAllCaps="fal
    se"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/respassword"
    app:layout_constraintVertical_bias="0.607" />


<TextView
    android:id="@+id/l
    ogtv1"

    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Don,t Have An Account ?"
    android:textAlignment="center"
    android:textColor="#FBFBFB"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.34"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/logb1"
    app:layout_constraintVertical_bias="0.136" />


<TextView
    android:id="@+id/l
    ogtv2"

    android:layout_width="wrap_content
    "
    android:layout_height="wrap_conten
    t" android:layout_marginTop="12dp"
    android:onClick="register"
    android:text="register"
    android:textAlignment="center"
    android:textColor="@color/purple_5
    00"android:textSize="16sp"
    android:textStyle="bold"

    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.104"
    app:layout_constraintStart_toEndOf="@+id/logtv1"
    app:layout_constraintTop_toBottomOf="@+id/logb1"
    app:layout_constraintVertical_bias="0.03"
    tools:ignore="TouchTargetSizeCheck" />


<TextView
    android:id="@+id/logforgetpass"
    android:layout_width="wrap_conten
    t"
    android:layout_height="wrap_conte
    nt"
    android:layout_marginTop="11dp"
    android:layout_marginBottom="187
    dp"android:text="Forget Password
    ?"
    android:textColor="@color/white"

    app:layout_constraintBottom_toTopOf="@+id/logb1"
    app:layout_constraintEnd_toEndOf="parent"


    app:layout_constraintHorizontal_bias="0.141"
    app:layout_constraintStart_toStartOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/respassword"
        app:layout_constraintVertical_bias="1.0" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

9.2 Screenshots:

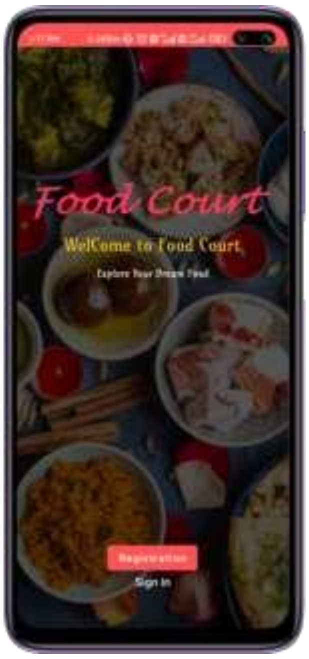


Figure 9.5.1 Starting Screen

This is a Starting Screen

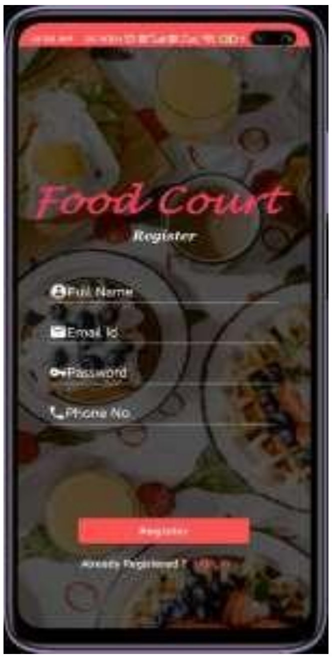


Figure 9.5.2 Registration

This page Provides Registration for User or Customer.

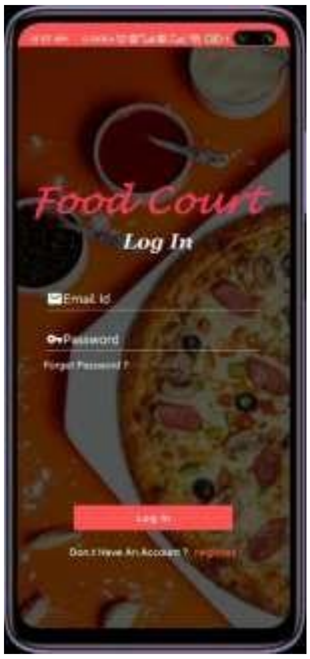


Figure 9.5.3 Login Page

This page Provides Login for Registered User or Customer.

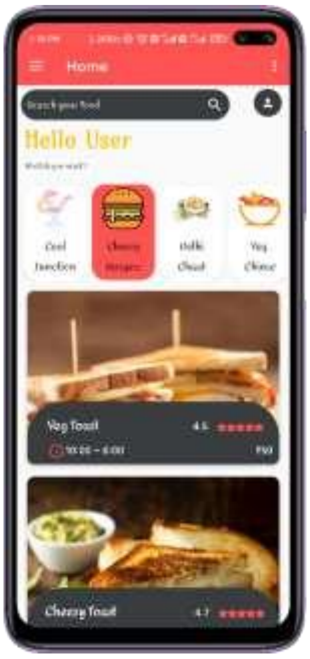


Figure 9.5.4 Home page

This Page Provides short information about our food-court and its products.

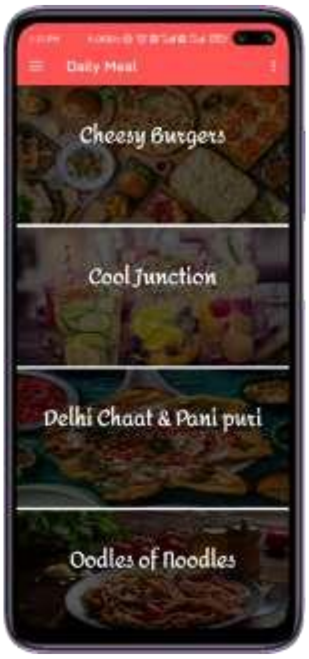


Figure 9.5.5 Daily Meal

Here User or Customer can see daily meals with different junctions available in food-court.

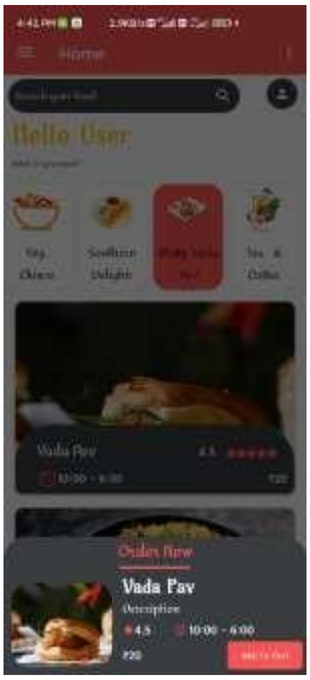


Figure 9.5.6 Product Detail Page

Here User or Customer can see the products price and able to add in cart.

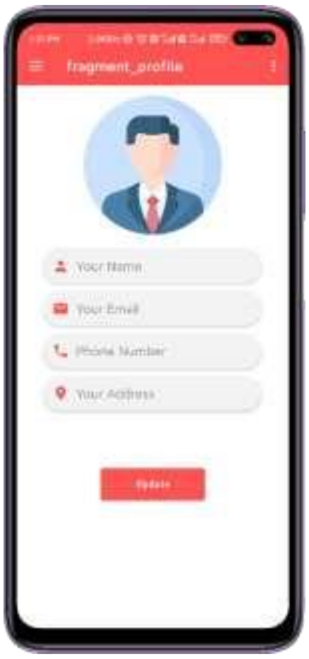


Figure 9.5.7 Update Page

Here user or customer can change or update his/her data.

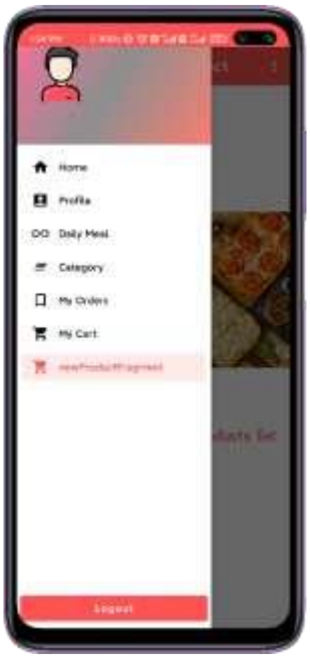


Figure 9.5.8 Log out

Here User or Customer is able to logout from system .

10. TESTING AND RESULTS OF PROJECT

TESTING METHODOLOGIES

1. Black box Testing
2. White box Testing

LEVELS OF TESTING

1. Unit Testing
2. Integration Testing
3. System Testing

TESTING METHODOLOGIES

1. Black box Testing

It is the testing process in which tester can perform testing on an application without having any internal structural knowledgeof application. Usually Test Engineers are involved in the black box testing.

2. White box Testing

It is the testing process in which tester can perform testing on an application with having internal structural knowledge. UsuallyThe Developers are involved in white box testing.

LEVELS OF TESTING

1. Unit Testing

Unit Testing concentrates on the verification of the smallest element of the program i.e. Module. In this testing all controlpaths are tested to identify errors within the bounds of the module. The important goal of unit testing is to isolate each partof the program and show individual parts are correct. It is very easy to perform and requires less amount of time because the modules are smaller in size. In unit testing it is possible that the outputs produced by one unit become input for anotherunit hence, if incorrect output produced by one unit is provided as input to the second unit then it also produces wrong output. If this process is not corrected, the entire software may produce unexpected outputs. To avoid this, all the units in the software are tested independently using unit testing. In unit testing, the units are tested to ensure that they operate correctly. In software engineering the unit testing is not just performed once during software development, but repeated whenever the software is modified.

2. Integration Testing

When unit testing is complete, integration testing begins. In integration testing the tested units are combined together to form system as whole. The aim of this testing is to ensure that all modules are working properly according to user’s requirements when they are combined. The integration test takes all tested individual modules, integrate them, test them again and develop the software. It ensures that all modules work together properly and transfer accurate data across their interfaces.

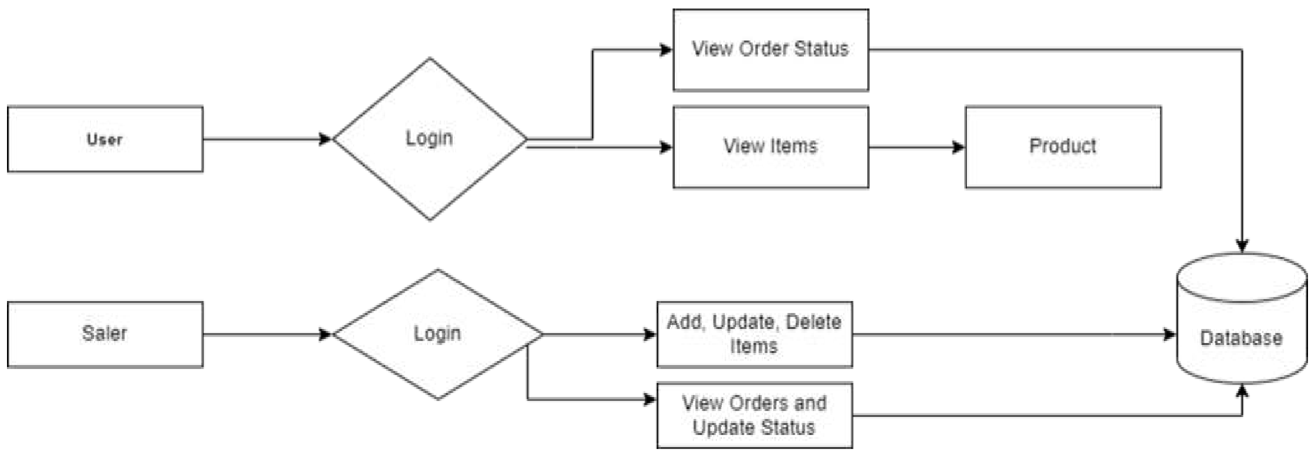
3. System Testing

System testing is the next level in the testing and tests the system as a whole. Once, all the components are integrated, theapplication as a whole is tested to see that it meets Quality Standards. This type of testing is performed by a specialized testing team. System testing can be defined as ”a testing conducted on a complete, integrated system to ensure that the system is according to its specified requirement”.

11. METHODOLOGY

SYSTEM ARCHITECTURE

In this system user orders the food by using android based touch pad. Figure shows the system architecture. After loginto system user can able to see the food items available in food-court. User is able to add food item to cart, view order status, deleteorder , update his profile which stored in database.



- **Login:** Module will authenticate the admin or customer which has to be login. The Login Module is a portal module that allowsusers to type a Email and password to log in.

- **Registration:** In this module the customer can register himself. He needs to provide his details like name, address, phone number andhe needs to create a username and password.

- **Search:** A search module is for search required products using search bar. search module helps user to find his product he wants.

12. SYSTEM CONFIGURATION

Hardware Requirement:

- Android Device
- RAM - 2 GB (min)
- Processor – snapdragon 660

Software Requirement

- Operating System -Windows 10+ , Android 8+
- Languages - Java, XML
- Tool - Android Studio

Database Requirement

- Firebase

13. ADVANTAGES AND DISADVANTAGES OF PROPOSED SYSTEM

Advantages:

- Make the ordering process easier.
- Efficient user/customer and order management.
- Better user’s data.
- It saves time.
- Freedom of choice.

Disadvantages:

- You need a secure internet connection
- Spending too much time online.
- Data security.
- Mismatch data of order, user.
- Risk of losing customer data.

14. CONCLUSION

The application is based on user’s requirement and is user centered. All issues related to all user which are included in this system are developed by this system. If people know how to operate android smart phone wide variety of people can use the application. This system will solve the various issues related to Mess/Tiffin service. To help and solve important problems of people implementation of Online Food Ordering system is done. It can be concluded that, based on the application: Orders are made easily by this system; Information needed in making order to customer is provided by the system. Receiving orders and modifying its data is possible through the application and it also helps admin in controlling all the food system

REFERENCES

[1] Prathamesh Jagannath Mane” Study of Online Food Delivery App like Zomato & Swiggy and their effect on Casual Dining.” International Journal of Scientific Research and Engineering Development.

[2] Abhishek Singh1, Adithya R2, Vaishnav Kanade3, Prof. Salma Pathan4 “ONLINE FOOD ORDERING SYSTEM “International Research Journal of Engineering and Technology (IRJET).

[3] Resham Shinde, Priyanka Thakare, Neha Dhomne, Sushmita Sarkar, ”Design and Implementation of Digital dining in Restaurants using Android”, International Journal of Advance Research in ComputerScience and Management Studies 2014.

[4] Patel Krishna, Patel Palak, Raj Nirali, Patel Lalit,” Automated Food Ordering System”, International Journal of Engineering Research and Development (IJERD) 2015.