CONTENTS

1. INTRODUCTION

Google's e-Commerce application is an online portal that facilitate online transactions of Google products through means of the transfer of information and funds over the Internet. In the early days, e-Commerce was done partially through emails and phone calls. Now, with a single websites, Android and iOS applications, anything and everything that a transaction needs, can be executed online.

With mobile devices taking over the planet, quite literally everyone uses their mobile devices more often than their laptops or desktops – to browse the internet. There has been a huge increase in transactions and bookings done via mobile devices in the last few years – so the benefits of mobile applications are significant.

Features:

With the use of mobile devices increasing every day, there are a lot of advantages of e-commerce applications like –

- Global market reach
- A global choice for consumers
- Short product/service distribution chain
- Lesser costs and pricing

Objectives:

e-Commerce application is an online portal that facilitate online transactions of products through means of the transfer of information and funds over the Internet. With mobile devices taking over the planet, quite literally everyone uses their mobile devices more often than their laptops or desktops – to browse the internet.

Modules:

1.1 Sign-in Page

There are 2 main users who operates the system:

- Google Dealer
- Customer

1.2 Sign-up Page

The idea of creation of an online sign-up form and online sign-up process completed by the individual to apply to register for the Google's e-Commerce App is implemented in this software. Online registrations improve efficiencies.

1.3 Home Page

There are 2 separate types of Home Page available. One is for the Google dealer and another is for the normal customer.

Google Dealer Home Page

Here the dealer will be provided with the following features -

- Add new Google Products
- Check Customer's Orders
- Logout Button

Customer's Home Page

Here the customers will be provided with the following features-

- Explore all the trending events going in Google
- Explore all the new products soon to be launched
- Place orders for Google products online
- Open other Google apps if required

1.4 Search Activity

Search activity helps the user to save their time by allowing the user to search a particular product for which he is looking for.

More-over it will list down all the related products of the searched one. By clicking on the products, customer will directly be shifted to the product details activity.

1.5 Products Activity

All the Google products which are added by the Google dealer will be listed down here. The customer can easily select their desired products and shift to the product details activity. These products are classified based on their category.

As there are four categories in the application, so the dealer gets the facility to add four different products in different categories. Some of the options are-

- Add new Phone
- Add new Accessories
- Add new Outfits
- Add new Smart Home devices

1.6 Product details Activity

This activity will show the details of the product which the customer has selected to see along with the product photo, product name, product price, product detailed and an option to add the product to the cart.

1.7 Cart Activity

The cart feature in any e-Commerce app helps the customer to place multiple number of orders collectively at the same time. This is time efficient in the side of the customer.

Similarly, here in Google's e-Commerce app, the customer can place multiple number of orders at the same time. But in case if the cart is empty, the user will not be allowed to place any order. For placing order, the user must have added at least one or more products to the cart.

1.8 Order Confirmation Activity

After the cart activity, the customer will be taken to the 'Order Confirmation Activity' which will show the user details including User Name, User Phone number, User Address and the total amount which needs to be paid. Once the confirm button is clicked, the customer will get an option to choose two payment methods – Cash on delivery and Razor-pay Online pay. Once this process is completed, the customer will be shifted to the order confirmation activity which will confirm that the order has been successfully paced.

1.9 Account Details Activity

The Account details activity will show the details of the customer who is currently logged in. The details include –

- Customer's name
- Customer's phone number
- Customer's address
- Customer's account level

It also allows the user to log himself/herself out from the application for some security reasons. Once the customer clicks the logout button, he/she will directly be shifted to the customer's login page.

There are basically four level which a customer can achieve. These level changes diametrically based on the number of orders placed.

Level 1	Number of orders <= 8
Level 2	Number of orders >= 9 but <= 16
Level 3	Number of orders >= 17 but <= 25
Level 4	Number of orders >= 26

1.10 Contact Us Activity

If the customer faces any problem while using the application then he/she can directly contact to our Google customer care for further help.

1.11 Opening related Google app

Other google apps like Google Maps, Google Mail, Google Play, Google Search can be opened via this application as per customer's requirements.

1.12 Check Customer's Orders [Only for Google Dealer]

The Google dealer can check the orders which are placed by the customers and then deliver the respective orders to the respective customers.

1.13 Order Details Fragment

Once the customer confirms the orders, he can check their order along with the order status in the order fragment. The status of the order gets changed dynamically as soon as the dealer changes it from the dealer panel.

1.14 About Us Activity

This activity provides details about the company to the user to build the trust between the company and the user. This also provides the Terms and conditions of the company.

1.15 Dark Theme

Since the new release of Android 10 and iOS 13, the dark theme is the most demanded feature which most of the user demand for. The Dark theme benefits the user in different ways like long lasting battery coverage, comfort for the eyes etc.

1.16 Recently Viewed Items

This feature keeps track of the products which the customer has seen recently. This feature is helpful in a scenario if a customer leaves the application for some reason and then again plans to place the order for the product which he viewed earlier. It shows the list of recently viewed items.

Functionalities:

Google Dealer Functionality:

- ❖ Add new Google products for the customers
- ❖ Check Customer's orders in details
- ❖ Process Customer's orders
- ❖ Log himself/herself out from the dealer account

Customer's Functionality:

- ❖ Browse all the latest technologies trending in Google
- ❖ Check product details for a specific product
- ❖ Add 'n' number of products to the cart
- ❖ Place 'n' number of orders once it is added to cart
- ❖ Search any products for saving time
- ❖ Check order process status
- ❖ Ask any help to the Google's customer care 24X7

2. LITERATURE SURVEY

2.1. Aim:

E-commerce is the route for purchasing goods and services online. The money transactions are done through online thus leads to the digital economy. One research data to know is that 80% of online users buy their products in an online store than a physical store. As per statistics, $4.5 trillion in sales are expected by 2021.The motto of e-commerce activity is to reach millions of customers easily and to increase sales in business. It generates a high revenue in the online industry as the viewers are turning into the audience every day.

2.2. Tools Survey:

2.2.0 Android Studio

Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ_IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Git-hub integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules. Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store. The software was first announced at Google I/O in May 2013, and the first stable build was released in December 2014. Android Studio is available for Mac, Windows, and Linux desktop platforms. It replaced Eclipse Android Development Tools (ADT) as the primary IDE for Android application development. Android Studio and the Software Development Kit can be downloaded directly from Google.

2.2.1 XML:

XML stands for **E**xtensible **M**ark-up **L**anguage. It is a text-based mark-up language derived from Standard Generalised Mark-up Language (SGML).

XML tags identify the data and are used to store and organise the data, rather than specifying how to display it like HTML tags, which are used to display the data. XML is not going to replace HTML in the near future, but it introduces new possibilities by adopting many successful features of HTML.

There are three important characteristics of XML that make it useful in a variety of systems and solutions −

- **XML is extensible** − XML allows you to create your own self-descriptive tags, or language, that suits your application.
- **XML carries the data, does not present it** − XML allows you to store the data irrespective of how it will be presented.
- **XML is a public standard** − XML was developed by an organisation called the World Wide Web Consortium (W3C) and is available as an open standard.

Applications of XML-

- XML can work behind the scene to simplify the creation of HTML documents for large web sites.
- XML can be used to exchange the information between organisations and systems.
- XML can be used for offloading and reloading of databases.
- XML can be used to store and arrange the data, which can customise your data handling needs.
- XML can easily be merged with style sheets to create almost any desired output.
- Virtually, any type of data can be expressed as an XML document.

2.2.2 JAVA:

Java is a high-level programming language originally developed by Sun Microsystems and released in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

It is used for:

- Mobile Android applications
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection

2.2.3 Firebase:

The Firebase Realtime Database is a cloud-hosted NoSQL database that lets you store and sync data between your users in real time.

It is a Backend-as-a-Service — BaaS — that started as a YC11 start-up and grew up into a next-generation app-development platform on Google Cloud Platform.

Firebase frees developers to focus crafting fantastic user experiences. You don’t need to manage servers. You don’t need to write APIs. Firebase is your server, your API and your datastore, all written so generically that you can modify it to suit most needs. Yeah, you’ll occasionally need to use other bits of the Google Cloud for your advanced applications. Firebase can’t be everything to everybody. But it gets pretty close.

Real-time data is the way of the future. Nothing compares to it. Most databases require you to make HTTP calls to get and sync your data. Most databases give you data only when you ask for it. When we connect your app to Firebase, you’re not connecting through normal HTTP. You’re connecting through a WebSocket. Web-Sockets are much, much faster than HTTP. You don’t have to make individual WebSocket calls, because one socket connection is plenty. All of your data syncs automagically through that single WebSocket as fast as your client’s network can carry it. Firebase sends new data as soon as it’s updated. When your client saves a change to the data, all connected clients receive the updated data almost instantly.

2.2.3 Fire-store:

Cloud Fire-store is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. Like Firebase Realtime Database, it keeps your data in sync across client apps through real time listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Fire-store also offers seamless integration with other Firebase and Google Cloud Platform products, including Cloud Functions.

3. REQUIREMENT ANALYSIS

3.1 Software Requirements Specification (SRS):

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system, we should have clear understanding of Software system. It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for specification.

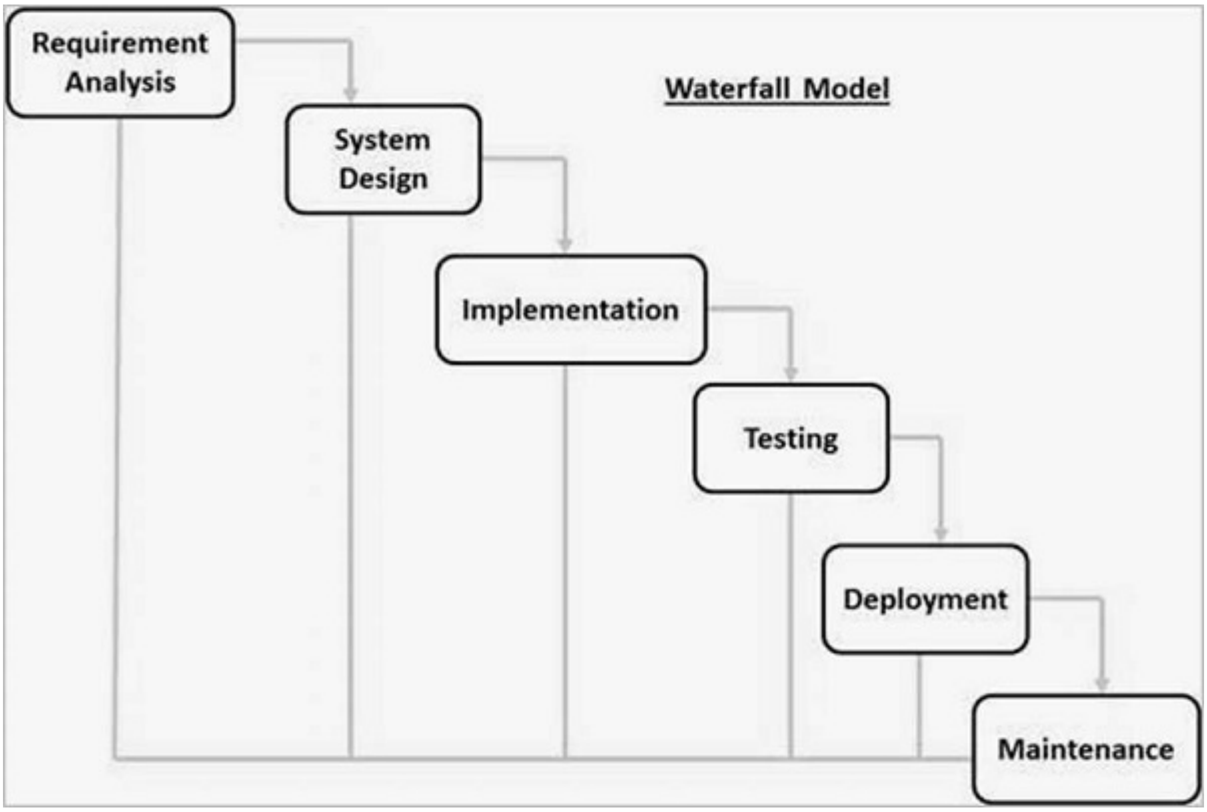
SRS are typically developed during the first stages of “Requirements Development” which is the initial product development phase in which information is gathered about what requirements are needed and what are not needed. This information gathering stage can include onsite visits, questionnaires, surveys and perhaps a needs analysis of the customer or client’s current business environment. The actual specification is then written after the requirements have been gathered and analysed.

User Interface	Android Studio XML Java
Database	Firebase Firestore (Google Cloud)

3.1.1 Specification of SDLC Model used:

The Google’s e-Commerce is a software application which is developed using the waterfall software developing model. It is also referred to as a linear-sequential life cycle model. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development. It illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are –

- Requirement Analysis – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- System Design – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- Implementation – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- Integration and Testing – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- Deployment of system – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- Maintenance – There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Applications of Waterfall Model:

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.
- The project is short.

Advantages of Waterfall Model:

The advantages of waterfall development are that it allows for departmentalisation and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

Some of the major advantages of the Waterfall Model are as follows −

- Simple and easy to understand and use.
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Clearly defined stages.
- Well understood milestones.
- Easy to arrange tasks.
- Process and results are well documented.

Disadvantages of Waterfall Model:

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

The major disadvantages of the Waterfall Model are as follows −

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.

- Poor model for long and ongoing projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty are high with this process model.
- It is difficult to measure progress within stages.
- Cannot accommodate changing requirements.
- Adjusting scope during the life cycle can end a project.
- Integration is done at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

3.2 Hardware Requirements:

Processor	Snapdragon / Media-Tech any version
RAM	1 GB or above
OS	Android [Above 6.0]

3.3 Feasibility Analysis:

3.3.1 Technical Feasibility:

Technical feasibility is about risk assessment and solutions. The evaluation of technological and system requirements basically refers to issues related to the type, availability and quality of the following, used in the project:

- data sources
- methods of data processing, analysing, and visualising
- software

The development process of payroll management system would be advantageous to the organisation because we would make use of only the currently available resources of the organisation. All the tools needed for the development are already available with the organisation and the organisation does not have to acquire any new resources. The technical feasibility is also attributed to the fact that the system does not make use of any additional or external third-party components which can lead to increased load on the system.

3.3.2 Operational Feasibility:

Operational feasibility is based on control, efficiency and services. Allowing for the level of knowledge and skills necessary for data acquisition, and later to process, analyses and visualise them, as well as to implement the results, is of paramount importance in the evaluation of operational requirements. The payroll management system is intended to provide every user-friendly and easy to use interface which is beneficial for both the employees as well as the operators who help in providing support for the system. This system would also be easily acceptable among the employees and administrator as there is no need of any special skill set for using the application. This system also benefits the users as they do not have to download anything on their terminals increasing their efficiency and ease of use.

3.3.3 Economic Feasibility:

Economic feasibility is the cost-benefit analysis. Three components are important in the evaluation of economic requirements:

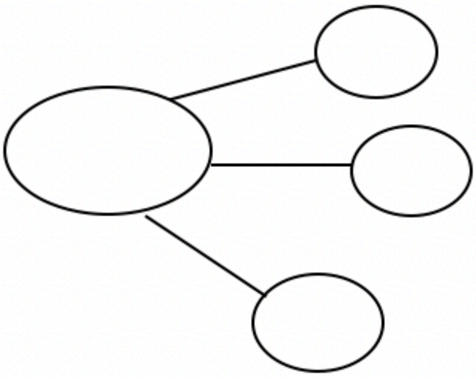
- The cost of data or software procurement
- The cost of employing workforce for specific tasks
- The cost of possible improvement of qualifications of the workforce employed

The payroll management system has a very low development cost. The low cost is attributed to the usage of the existing resources of the organisation. As the software is very user-friendly and easy to use, there is no need to provide special training to the users of the software, thus saving valuable time and money.

3.3.4 Scheduled Feasibility:

Scheduled feasibility is about timeline estimations and optimising resources. It is the process of assessing the degree to which the potential time frame and completion dates for all major activities within a project meet organisational deadlines and constraints for affecting change. The evaluation of requirements related to the schedule concerns the estimation of time necessary to complete respective parts of a project, e.g. data procurement and processing.

The payroll management system project was successfully completed on specified time based on the Control Decentralised team organisation and the requirements were fulfilled.

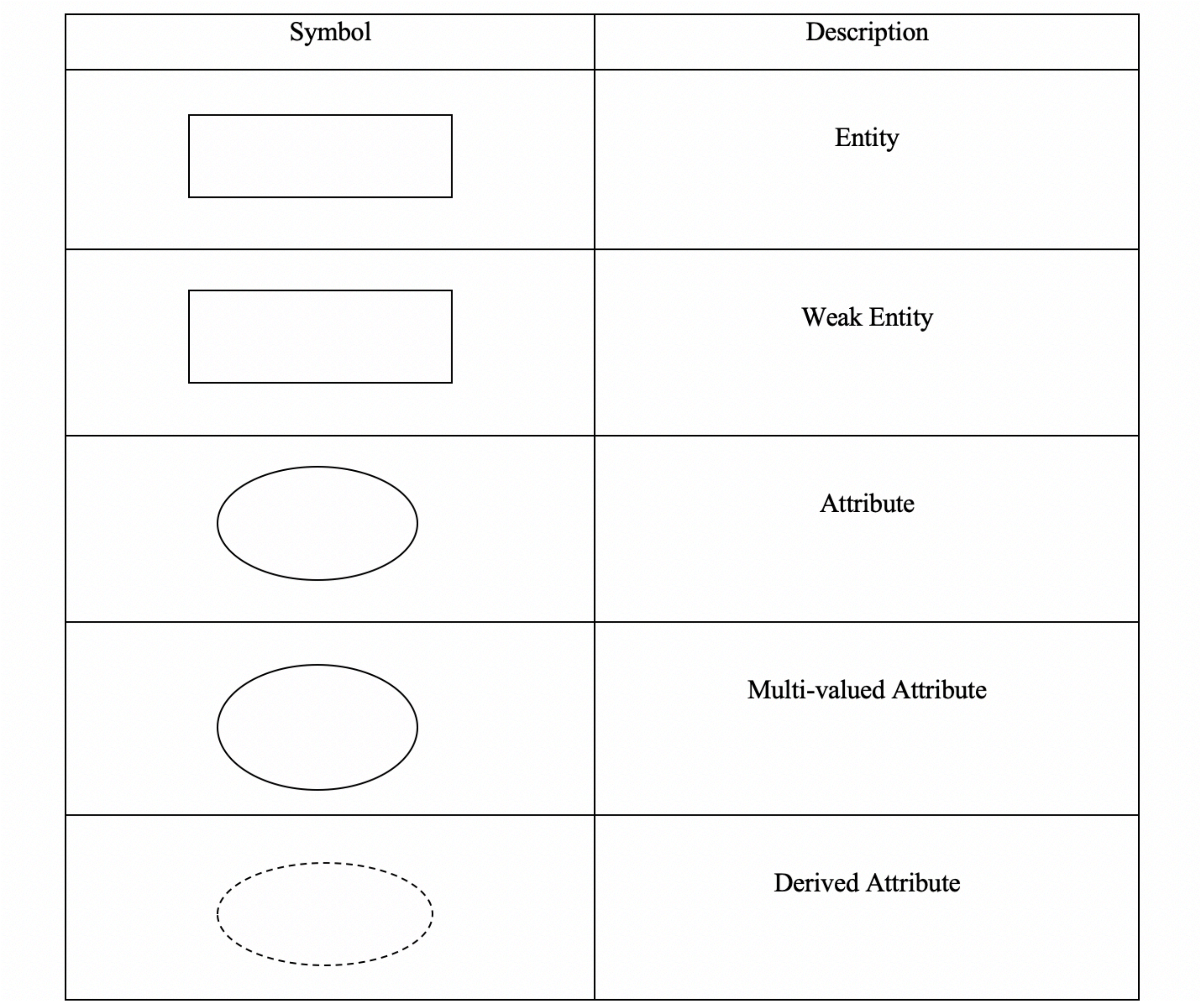
——	Link
◇	Relationship
◇	Weak Relationship
	Composite Attribute
⬭	Key Attribute
1:N	Cardinality Ratio
═	Total Participation

4. DESIGN

4.1 E-R Diagram:

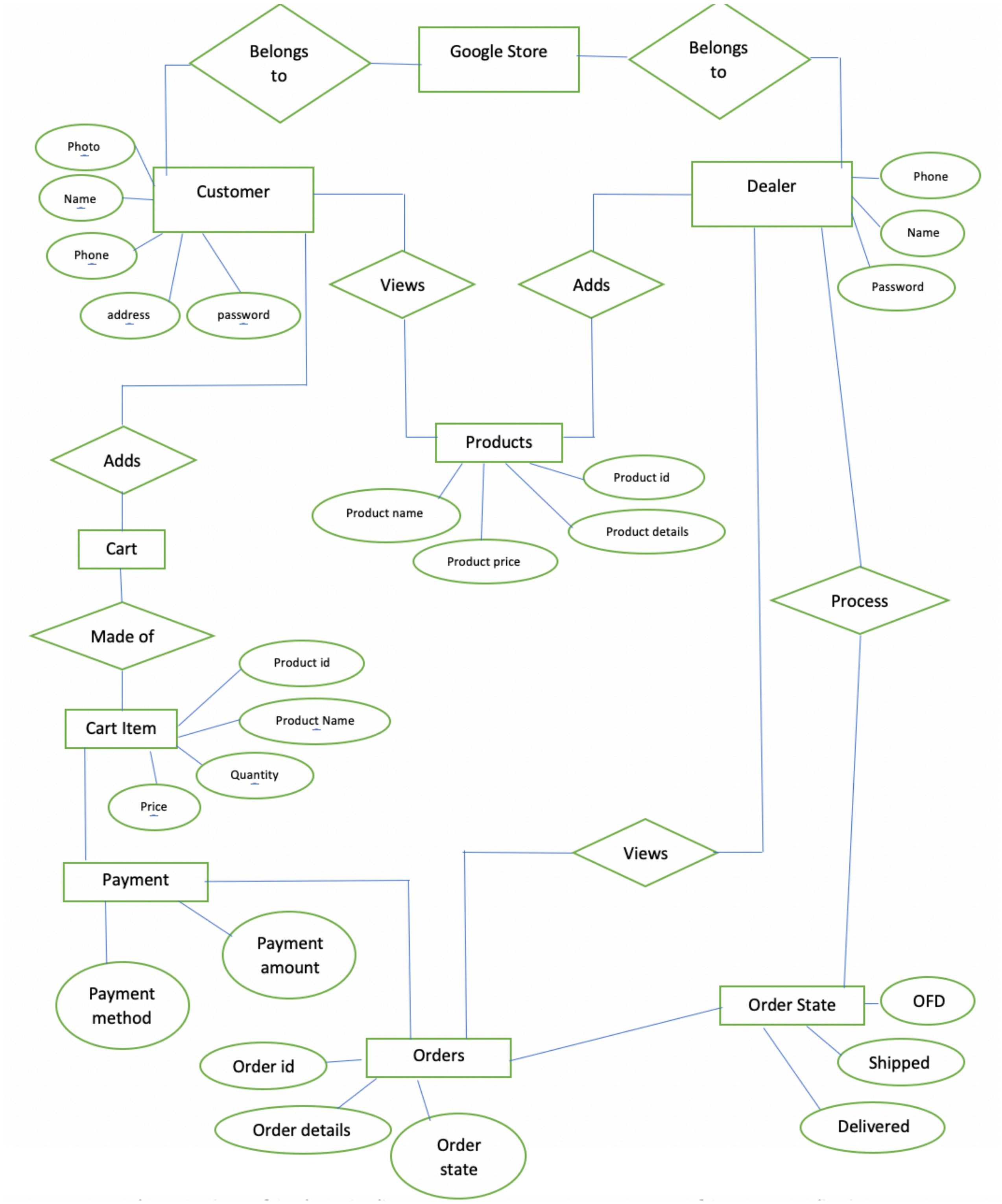
An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database.



Symbol	Description
▭	Entity
▭	Weak Entity
⬭	Attribute
⬭	Multi-valued Attribute
⬭	Derived Attribute

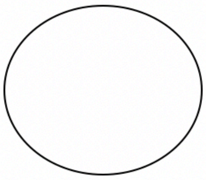
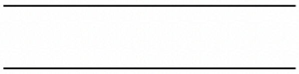
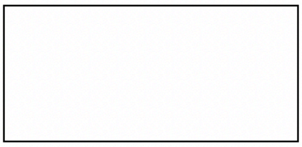
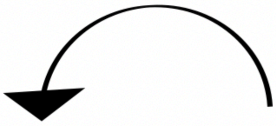
4.1.1 ER Diagram:



4.2 DFD:

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no controlled flow; there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

In Software engineering DFD (data flow diagram) can be drawn to represent the system of different levels of abstraction.

	Process (Function)
	Warehouse (File/Database)
	External Entity (Input/output)
	Data Flow

Process:

The process (function, transformation) is part of a system that transforms inputs to outputs. The symbol of a process is a circle, an oval, a rectangle or a rectangle with rounded corners. The process is named in one word, a short sentence, or a phrase that is clearly to express its essence.

Data Flow:

Data flow (flow, data-flow) shows the transfer of information (sometimes also material) from one part of the system to another. The symbol of the flow is the arrow. The flow should have a name that determines what information (or what material) is being moved. Exceptions are flows where it is clear what information is transferred through the entities that are linked to these flows. Material shifts are modelled in systems that are not merely informative. Flow should only transmit one type of information (material). The arrow shows the flow direction (it can also be bi-directional if the information to/from the entity is logically dependent - e.g. question and answer), flow link processes, warehouses and terminators.

Warehouse:

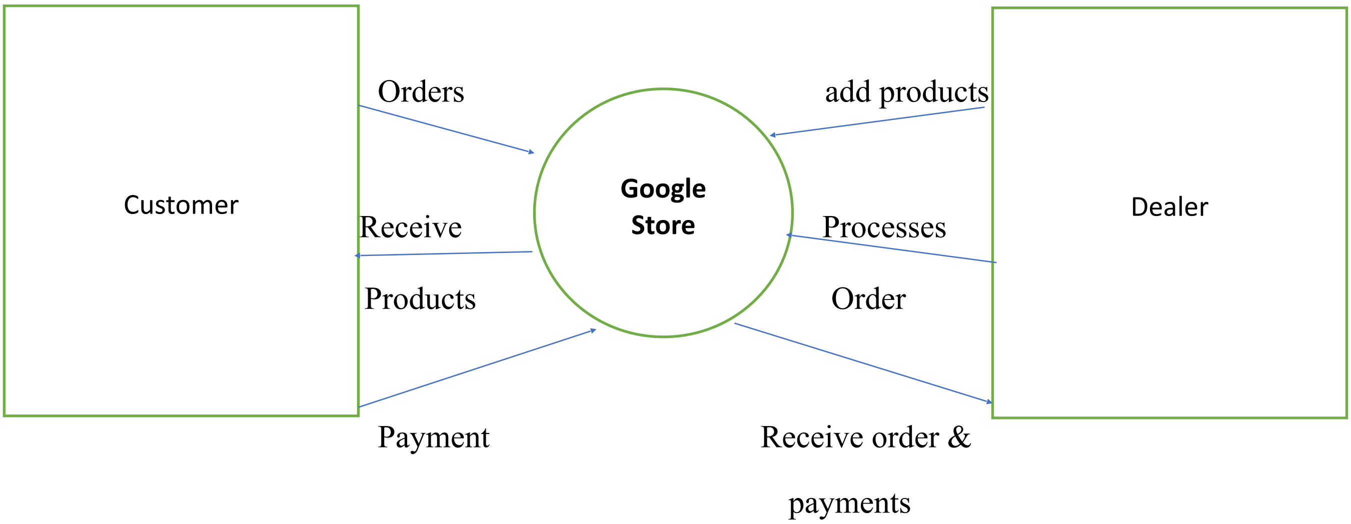
The warehouse (data store, file, and database) is used to store data for later use. The symbol of the store is two horizontal lines; the other way of view is shown in the DFD Notation. The name of the warehouse is a plural noun (e.g. orders) - it derives from the input and output streams of the warehouse. The warehouse does not have to be just a data file, for example, a folder with documents, a filing cabinet, and optical discs. Therefore, viewing the warehouse in DFD is independent of implementation. The flow from the warehouse usually represents the reading of the data stored in the warehouse, and the flow to the warehouse usually expresses data entry or updating (sometimes also deleting data). Warehouse is represented by two parallel lines between which the memory name is located (it can be modelled as a UML buffer node).

Terminator:

The Terminator is an external entity that communicates with the system and stands outside of the system. It can be, for example, various organisations, groups of people (e.g. customers), authorities or a department of the same organisation, which does not belong to the model system. The terminator may be another system with which the modelled system communicates.

4.2.1 DFD Diagram:

It is also known as context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as single bubble with input and output data indicated by incoming/outgoing arrows.



5. CODING

Splash Screen

XML:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_height="match_parent"
    tools:context=".MainActivity"/>
```

JAVA:

```
package com.example.googlestore;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.os.SystemClock;
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        SystemClock.sleep(2000);
        Intent intent = new Intent(MainActivity.this, WelcomeActivity.class);
        startActivity(intent);
        finish();
    }
}
```

Welcome Activity

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=".WelcomeActivity">

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/nextPage"
        app:layout_constraintTop_toBottomOf="@+id/imageView2"
        app:rippleColor="#CDDC39"
        tools:ignore="RelativeOverlap" />
    </androidx.constraintlayout.widget.ConstraintLayout>
```

JAVA:

```
package com.example.googlestore;
public class WelcomeActivity extends AppCompatActivity {

    private FloatingActionButton nextPage;
    private ProgressDialog loadingBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_welcome);
        Paper.init(this);
        loadingBar = new ProgressDialog(this);
        nextPage.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(WelcomeActivity.this, UserLogin.class);
                startActivity(intent);
            }
        });
        if (UserPhoneKey != "" && UserPasswordKey != ""){
            if (!TextUtils.isEmpty(UserPhoneKey) && !TextUtils.isEmpty(UserPasswordKey)){
                loadingBar.setMessage("We are directly logging you in");
                loadingBar.setCanceledOnTouchOutside(false);
                loadingBar.show();
                AllowAccess(UserPhoneKey ,UserPasswordKey);
            }
        }
        if (AdminPhoneKey != "" && AdminPasswordKey != ""){
            if (!TextUtils.isEmpty(AdminPhoneKey) && !TextUtils.isEmpty(AdminPasswordKey)){
                loadingBar.setMessage("We are directly logging you in");
                loadingBar.setCanceledOnTouchOutside(false);
                loadingBar.show();
                AllowAccessForAdmin(AdminPhoneKey ,AdminPasswordKey);
            }
        }
    }
    private void AllowAccessForAdmin(final String phone, final String password) {
        final DatabaseReference RootRef;
        RootRef = FirebaseDatabase.getInstance().getReference();


        RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
            }
        });
    }

    private void AllowAccess(final String phone, final String password) {
        final DatabaseReference RootRef;
        RootRef = FirebaseDatabase.getInstance().getReference();
        RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                }else {
                    Toast.makeText(WelcomeActivity.this, phone + " doesn't exist",
Toast.LENGTH_SHORT).show();
                    loadingBar.dismiss();
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}});}}
```

User’s Login Activity

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"
    android:background="@drawable/signin"
    tools:context=".UserLogin">
    <EditText
        android:id="@+id/Login_password"
        android:layout_width="260dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Password"/>
    <CheckBox
        android:id="@+id/RememberMe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Remember Me"
        app:layout_constraintBottom_toTopOf="@+id/textView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.085"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/Login_password" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

JAVA:

```
package com.example.googlestore;


import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
```

```
        LoginButton = (FloatingActionButton) findViewById(R.id.LoginButton);
        InputNumber = (EditText) findViewById(R.id.Login_phoneNumber);
        InputPassword = (EditText) findViewById(R.id.Login_password);
        loadingBar = new ProgressDialog(this);
    private void LoginUser() {
        String phone = InputNumber.getText().toString();
        String password = InputPassword.getText().toString();
        if (TextUtils.isEmpty(phone)){
            Toast.makeText(this, "Phone number is required", Toast.LENGTH_SHORT).show();
        }
        RootRef = FirebaseDatabase.getInstance().getReference();

        RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                if (dataSnapshot.child("Users").child(phone).exists()){
                    Users usersData = dataSnapshot.child("Users").child(phone).getValue(Users.class)
                    }
                }else {
                    Toast.makeText(UserLogin.this, phone + " doesn't exist",
Toast.LENGTH_SHORT).show();
                    loadingBar.dismiss();}}
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {}});}}
```

<u>User’s Registration Activity</u>

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"
```

```
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.571"
        app:srcCompat="@drawable/logo" />
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.178"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
        android:hint="Password"
        android:inputType="textPassword"
        app:layout_constraintBottom_toTopOf="@+id/SignupTextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.178"
        app:layout_constraintStart_toStartOf="parent" />

    <EditText
        android:id="@+id/address"
        android:layout_width="260dp"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Address"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@+id/SignupTextView"
        app:layout_constraintEnd_toEndOf="@+id/Password"
        app:layout_constraintStart_toStartOf="@+id/Password"
        app:layout_constraintTop_toBottomOf="@+id/Password" />

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

JAVA:

```
  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_register);
      AlreadyAUser = (TextView) findViewById(R.id.alreadyAUser);
      String name = InputName.getText().toString();
      String phone = InputPhoneNo.getText().toString();
      String password = InputPassword.getText().toString();
      String address = InputAddress.getText().toString();
  private void ValidatePhoneNo(final String name, final String phone, final String password, final
String address) {
                        Intent intent = new Intent(RegisterActivity.this, UserLogin.class);
                        startActivity(intent);
                    }else {
                        loadingBar.dismiss();
                        Toast.makeText(RegisterActivity.this, "Error: Network Error in Database",
Toast.LENGTH_SHORT).show();
                    }
                  }
              });
          }else {
              Toast.makeText(RegisterActivity.this, "This " + phone + " already exists" ,
Toast.LENGTH_SHORT).show();
              loadingBar.dismiss();
          }
      }
      @Override
      public void onCancelled(@NonNull DatabaseError databaseError) {}});  }}
```

Dealer’s Login Page

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"
        android:maxLength="10"
        android:inputType="phone"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView5" />

    <EditText
        android:id="@+id/password"
        android:layout_width="256dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:ems="10"
        android:hint="Password"
        android:inputType="textPassword"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.503"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/phone" />



        android:textAllCaps="false"
        android:textColor="#FFFFFF"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/password"
        app:layout_constraintVertical_bias="0.31" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

JAVA:

```
package com.example.googlestore;

  @Overrid
        Intent intent = new Intent(Admin_LoginActivity.this, Admin_Register_Activity.class);
        startActivity(intent);
      }
    });}
  private void LoginAdmin() {
    String phone = Admin_phone.getText().toString();
      Paper.book().write(Prevalent.AdminPhoneKey, phone);
      Paper.book().write(Prevalent.AdminPasswordKey, password);
    }
    final DatabaseReference RootRef;
    RootRef = FirebaseDatabase.getInstance().getReference();

    RootRef.addListenerForSingleValueEvent(new ValueEventListener() {
      @Override
      public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.child("Admin").child(phone).exists()){
          Users usersData = dataSnapshot.child("Admin").child(phone).getValue(Users.class);
          if (usersData.getPhone().equals(phone)){
            if (usersData.getPassword().equals(password)){
              loadingBar.dismiss();
```

```
                    Toast.makeText(Admin_LoginActivity.this,  "Welcome Back Google Dealer",
Toast.LENGTH_SHORT).show();
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}});}}
```

Dealer Register Activity

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=".Admin_Register_Activity">

   <Button
      android:id="@+id/CreateDealerAccount"
      android:layout_width="250dp"
      android:layout_height="wrap_content"
      android:background="@drawable/admin_button"
      android:text="Create Dealer Account"
      android:textAllCaps="false"
      android:textColor="#FFFFFF"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.5"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/Password"
      app:layout_constraintVertical_bias="0.31" />
```

```
 <EditText
     android:id="@+id/Password"
     android:layout_width="256dp"
     android:layout_height="wrap_content"
     android:layout_marginTop="20dp"
     android:ems="10"
     android:hint="Password"
     android:inputType="textPassword"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintHorizontal_bias="0.5"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintTop_toBottomOf="@+id/PhoneNo" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

JAVA:

```
  private Button CreateDealerAccount;
 private ProgressDialog loadingBar;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
 private void createDealerAccount() {
     String name = AdminName.getText().toString();
     String phone = AdminPhone.getText().toString();
     String password = AdminPassword.getText().toString();
                 public void onComplete(@NonNull Task<Void> task) {
                     if (task.isSuccessful()){
                         loadingBar.dismiss();
                         Toast.makeText(Admin_Register_Activity.this, "Dealer account created",
Toast.LENGTH_SHORT).show();
                         Intent intent = new Intent(Admin_Register_Activity.this,
Admin_LoginActivity.class);
```

```
            }else {
                loadingBar.dismiss();
                Toast.makeText(Admin_Register_Activity.this, "This " + phone + " already exists" ,
Toast.LENGTH_SHORT).show();
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {}});}}
```

Home Fragment

XML:

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

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#FFFFFF">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <androidx.constraintlayout.widget.ConstraintLayout
        android:id="@+id/const1"
        android:layout_width="match_parent"
        android:background="@drawable/grid_main_backgroung">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="verti
            <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/accessories_list"
                android:layout_width="match_parent"
                android:paddingLeft="5dp"
                android:paddingRight="5dp"
                android:layout_height="match_parent" />
        </LinearLayout>
    </RelativeLayout>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="25dp"
        android:layout_marginBottom="15dp"
        android:fontFamily="@font/comfortaa"
        android:text="Showcase 2020"
        android:layout_gravity="center"
        android:gravity="center"
        android:textColor="#FC494949"
        android:textSize="25dp"
        android:textStyle="bold" />
```

```
<pl.droidsonroids.gif.GifTextView
  android:layout_width="330dp"
  android:layout_height="210dp"
  android:layout_marginTop="15dp"
  android:layout_gravity="center"
  android:elevation="15dp"
  android:background="@drawable/proo"
  android:layout_marginBottom="15dp" /
<RelativeLayout
  android:layout_width="match_parent"
  android:layout_height="300dp"
  android:layout_marginTop="10dp"
  android:paddingTop="20dp"
  android:layout_marginBottom="10dp"
  android:background="@drawable/smarthomebackground"
  >
  <LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
   <TextView
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginLeft="10dp"
      android:layout_marginTop="10dp"
      android:layout_marginBottom="15dp"
      android:fontFamily="@font/comfortaa"
      android:paddingBottom="10dp"
      android:text="Make your home Smarter"
      android:textColor="#FC393939"
```

JAVA:

```
import com.firebase.ui.database.FirebaseRecyclerAdapter;
import com.firebase.ui.database.FirebaseRecyclerOptions;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.squareup.picasso.Picasso;

import java.util.ArrayList;
import java.util.List;

import Model.Accessories;
import Model.Fabrics;
import Model.Home;
import Model.Products;
import ViewHolder.AccessoriesViewHolder;
import ViewHolder.ProductViewHolder;


/**
 * A simple {@link Fragment} subclass.
 */
public class home extends Fragment {

    public home() {
        // Required empty public constructor
    }

    private ViewPager2 viewPager2;
    private Handler sliderHandler = new Handler();
    private ImageView ph, stadia ,daydream, pixelBook, google, assistant;
    private RecyclerView accessories_list, fabric_list, home_list;
    private DatabaseReference AccessoriesRef, FabricRef, HomeRef;


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_home, container, false);
//        +++++++++++++++++++++++++++++++++ Assign ID to variables +++++++++++++++++++++++
+++++++++++++++++++
        ph = view.findViewById(R.id.ph);
//        stadia = view.findViewById(R.id.stadia);
//        daydream = view.findViewById(R.id.daydream);
        pixelBook =view.findViewById(R.id.pixelBookGo);
        google = view.findViewById(R.id.google);
        assistant = view.findViewById(R.id.assistant);
        accessories_list = view.findViewById(R.id.accessories_list);
        fabric_list = view.findViewById(R.id.fabric_list);
        home_list= view.findViewById(R.id.home_list);
//        +++++++++++++++++++++++++++++++++ Assign ID to variables +++++++++++++++++++++++
+++++++++++++++++++
        google.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = 
getActivity().getPackageManager().getLaunchIntentForPackage("com.google.android.googlequicks
earchbox");
                startActivity(i);
            }
        });
        assistant.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent i = 
getActivity().getPackageManager().getLaunchIntentForPackage("com.android.vending");
                startActivity(i);
            }
        });
        ph.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getActivity(), Products_List.class);
                startActivity(intent);
            }
        });

        pixelBook.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getActivity(), PixelBookGo.class);
                startActivity(intent);
            }
        });



        //        ============================ Banner Slider 
====================================
        viewPager2 = view.findViewById(R.id.viewPagerImageSlider);
        List<SliderItem> sliderItems = new ArrayList<>();
        sliderItems.add(new SliderItem(R.drawable.p4));
        sliderItems.add(new SliderItem(R.drawable.ghome));
        sliderItems.add(new SliderItem(R.drawable.stadia));
        sliderItems.add(new SliderItem(R.drawable.pixel3xl));
        sliderItems.add(new SliderItem(R.drawable.reg));
        viewPager2.setAdapter(new SliderAdapter(sliderItems, viewPager2));
        viewPager2.setClipToPadding(false);
        viewPager2.setClipChildren(false);
        viewPager2.setOffscreenPageLimit(3);
        viewPager2.getChildAt(0).setOverScrollMode(RecyclerView.OVER_SCROLL_NEVER);
        CompositePageTransformer compositePageTransformer = new CompositePageTransformer();
        compositePageTransformer.addTransformer(new MarginPageTransformer(40));
        compositePageTransformer.addTransformer(new ViewPager2.PageTransformer() {
            @Override
            public void transformPage(@NonNull View page, float position) {
                float r = 1 - Math.abs(position);
                page.setScaleY(1.00f + r * 0.10f);
            }
        });
        viewPager2.setPageTransformer(compositePageTransformer);
        viewPager2.registerOnPageChangeCallback(new ViewPager2.OnPageChangeCallback() {
            @Override
            public void onPageSelected(int position) {
                super.onPageSelected(position);
                sliderHandler.removeCallbacks(slideRunnable);
                sliderHandler.postDelayed(slideRunnable, 2000);
            }
        });
        //        ========================== Banner Slider
====================================

        accessories_list.setHasFixedSize(true);
        LinearLayoutManager linearLayoutManager = new LinearLayoutManager(getContext(),
LinearLayoutManager.HORIZONTAL,false);
//        linearLayoutManager.setStackFromEnd(true);
//        linearLayoutManager.setReverseLayout(true);
        accessories_list.setLayoutManager(linearLayoutManager);
        fabric_list.setHasFixedSize(true);
        LinearLayoutManager linearManager = new LinearLayoutManager(getContext(),
LinearLayoutManager.HORIZONTAL,false);
        linearLayoutManager.setStackFromEnd(true);
        linearLayoutManager.setReverseLayout(true);
        fabric_list.setLayoutManager(linearManager);
        home_list.setHasFixedSize(true);
        LinearLayoutManager linear = new LinearLayoutManager(getContext(),
LinearLayoutManager.HORIZONTAL,false);
        linearLayoutManager.setStackFromEnd(true);
        linearLayoutManager.setReverseLayout(true);
        home_list.setLayoutManager(linear);
        AccessoriesRef =
FirebaseDatabase.getInstance().getReference().child("Products").child("Accessories");
        FabricRef = FirebaseDatabase.getInstance().getReference().child("Products").child("Fabrics");
        HomeRef = FirebaseDatabase.getInstance().getReference().child("Products").child("Home");
        return view;
    }
    @Override
    public void onStart() {
        super.onStart();
// ++++++++++++++++++++++++++++++++++++++++ For Accessories ++++++++++++++++
++++++++++++++++++++++++++++++++++++++++
        FirebaseRecyclerOptions<Accessories> options = new
FirebaseRecyclerOptions.Builder<Model.Accessories>()
                .setQuery(AccessoriesRef, Model.Accessories.class)
                .build();
        FirebaseRecyclerAdapter<Accessories, AccessoriesViewHolder> adapter =
                new FirebaseRecyclerAdapter<Model.Accessories, AccessoriesViewHolder>(options) {
                    @Override
                protected void onBindViewHolder(@NonNull final AccessoriesViewHolder holder, int
position, @NonNull final Model.Accessories model) {

                    holder.txtProductName.setText(model.getProduct_Name());

                    holder.txtProductPrice.setText(model.getPrice());
Glide.with(holder.imageView).load(model.getProduct_Image()).into(holder.imageView);

                    final TextView txV = holder.txtProductName;

                    holder.itemView.setOnClickListener(new View.OnClickListener() {

                        @Override

                        public void onClick(View v) {

                            Intent intent = new Intent(getContext(), ProductDetailsActivity.class);

                            intent.putExtra("pid", model.getPid());

                            startActivity(intent);

                        }

                    });

                }

                @NonNull

                @Override

                public AccessoriesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {

                    View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.accessories_layout, parent,false);

                    AccessoriesViewHolder holder = new AccessoriesViewHolder(view);

                    return holder;

                }

            };

        accessories_list.setAdapter(adapter);

        adapter.startListening();

// +++++++++++++++++++++++++++++++++++++++++++++++ For Accessories ++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++


//        ++++++++++++++++++++++++++++++++++++++++++++++++ For Fabric List ++++++++++++++
+++++++++++++++++++++
    FirebaseRecyclerOptions<Fabrics> option = new
FirebaseRecyclerOptions.Builder<Model.Fabrics>()
        .setQuery(FabricRef, Model.Fabrics.class)
        .build();
    FirebaseRecyclerAdapter<Fabrics, AccessoriesViewHolder> adapters =
        new FirebaseRecyclerAdapter<Model.Fabrics, AccessoriesViewHolder>(option) {
          @Override
          protected void onBindViewHolder(@NonNull AccessoriesViewHolder holder, int
position, @NonNull final Model.Fabrics model) {
            holder.txtProductName.setText(model.getProduct_Name());
            holder.txtProductPrice.setText(model.getPrice());
            Picasso.get().load(model.getProduct_Image()).into(holder.imageView);
            holder.itemView.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View v) {
                Intent intenta = new Intent(getContext(), ProductDetailsActivity.class);
                intenta.putExtra("pid", model.getPid());
                startActivity(intenta);
              }
            });
          }
          @NonNull
          @Override
          public AccessoriesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
            View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.accessories_layout, parent,false);
            AccessoriesViewHolder holder = new AccessoriesViewHolder(view);
            return holder;
          }
        };
        fabric_list.setAdapter(adapters);
        adapters.startListening();
//      ++++++++++++++++++++++++++++++++++++++++++++++ For Fabric List ++++++++++++++
++++++++++++++++++++
        //      ++++++++++++++++++++++++++++++++++++++++++++++ For Home List ++++++++++
++++++++++++++++++++++++
        FirebaseRecyclerOptions<Home> optiona = new
FirebaseRecyclerOptions.Builder<Model.Home>()
                .setQuery(HomeRef, Model.Home.class)
                .build();
        FirebaseRecyclerAdapter<Home, AccessoriesViewHolder> homeadapter =
                new FirebaseRecyclerAdapter<Model.Home, AccessoriesViewHolder>(optiona) {
                    @Override
                    protected void onBindViewHolder(@NonNull AccessoriesViewHolder holder, int
position, @NonNull final Model.Home model) {
                        holder.txtProductName.setText(model.getProduct_Name());
                        holder.txtProductPrice.setText(model.getPrice());
                        Picasso.get().load(model.getProduct_Image()).into(holder.imageView);
                        holder.itemView.setOnClickListener(new View.OnClickListener() {
                            @Override
                            public void onClick(View v) {
                                Intent intenta = new Intent(getContext(), ProductDetailsActivity.class);
                                intenta.putExtra("pid", model.getPid());
                                startActivity(intenta);
                            }
                        });
                    }
                    @NonNull
                    @Override
                    public AccessoriesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
```

```
                View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.accessories_layout, parent,false);
                AccessoriesViewHolder holder = new AccessoriesViewHolder(view);
                return holder;
            }
        };
    home_list.setAdapter(homeadapter);
    homeadapter.startListening();
//      ++++++++++++++++++++++++++++++++++++++++++ For Home List +++++++++++++
+++++++++++++++++++
  }
  private Runnable slideRunnable = new Runnable() {
    @Override
    public void run() {
      viewPager2.setCurrentItem(viewPager2.getCurrentItem() + 1);
    }
  };
  @Override
  public void onPause() {
    super.onPause();
    sliderHandler.removeCallbacks(slideRunnable);
  }
  @Override
  public void onResume() {
    super.onResume();
    sliderHandler.postDelayed(slideRunnable, 2000);
  }
}
```

Customer Order Details Fragment

XML:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"


        </LinearLayout>
      <androidx.recyclerview.widget.RecyclerView
          android:id="@+id/list"
          android:paddingTop="15dp"
          android:paddingLeft="10dp"
          android:paddingRight="10dp"
          android:layout_gravity="center"
          android:layout_width="match_parent"
          android:layout_height="match_parent" />
    </LinearLayout>
</RelativeLayout>
```

JAVA:

```
package com.example.googlestore;

public class OrdersFragment extends Fragment {

    private RecyclerView list;
    RecyclerView.LayoutManager layoutManager;
    private DatabaseReference ordersRef;
    private  String state;


    public OrdersFragment() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_orders, container, false);
        list = view.findViewById(R.id.list);


    @Override
    public void onStart() {
        super.onStart();


        FirebaseRecyclerOptions<UserOrders> options = new
FirebaseRecyclerOptions.Builder<UserOrders>()
                .setQuery(ordersRef, UserOrders.class)
                .build();


        list.setAdapter(adapter);
        adapter.startListening();
    }


    private void checkProgress(String state) {
    }
}
```

6. TESTING

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. According to ANSI/IEEE 1059 standard, Testing can be defined as - A process of analysing a software item to detect the differences between existing and required conditions (that is defects/ errors/bugs) and to evaluate the features of the software item.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both large- and small-scale systems.

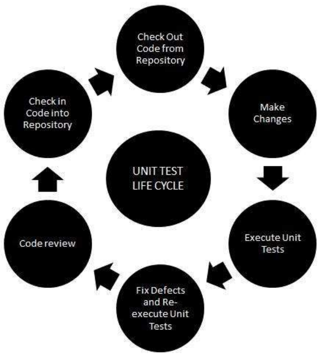
6.1 Functional Testing:

6.1.1 Unit Testing:

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there are any issue by the developer himself. It is correlated with functional correctness of the independent modules.

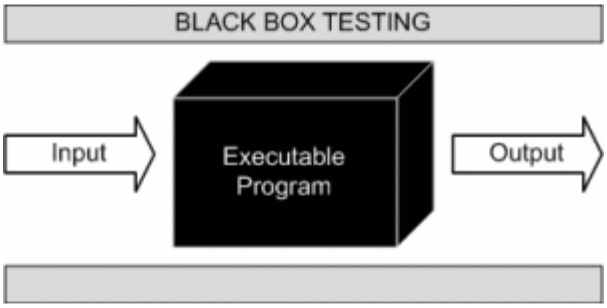
It is defined as a type of software testing where individual components of software are tested. Unit testing of software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit testing is typically performed by the developer.

Unit testing lifecycle-


Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Unit Tests
Code review
Check in Code into Repository
UNIT TEST LIFE CYCLE

6.1.1.1 Black Box Testing:

Black Box Testing, also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the software being tested is not known to the tester. Black box testing is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications.



In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program. Data are entered into the application and the outcome is compared with the expected results; what the program does with the input data or how the program arrives at the output data is not a concern for the tester performing black box testing. All that is tested is the behaviour of the functions being tested. This is why black box testing is also known as functional testing which tests the functionality of a program.

6.1.1.2 White Box Testing:

White Box Testing is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system.

White box testing involves the testing of the software code for the following:

- Internal security holes.
- Broken or poorly structured paths in the coding processes.
- The flow of specific inputs through the code.
- Expected output.
- The functionality of conditional loops.
- Testing of each statement, object, and function on an individual basis.

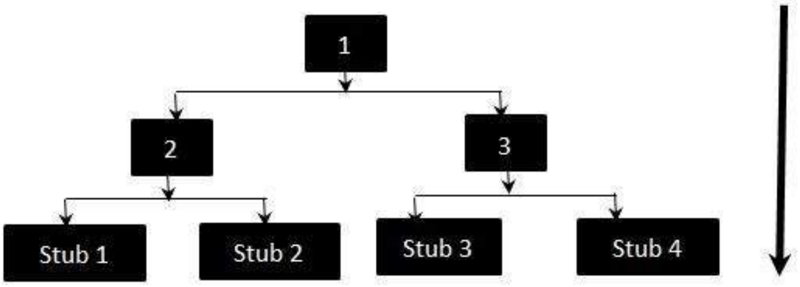
6.1.2 Integration Testing:

Integration testing is the process of testing the interface between two software units or module. It focuses on determining the correctness of the interface. The purpose of the integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

6.1.2.1 Top-down Integration Testing:

Top-down integration testing is an integration testing technique used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended. Stubs are the modules that act as temporary replacement for a called module and give the same output as that of the actual product. The replacement for the 'called' modules is known as 'Stubs' and is also used when the software needs to interact with an external system.

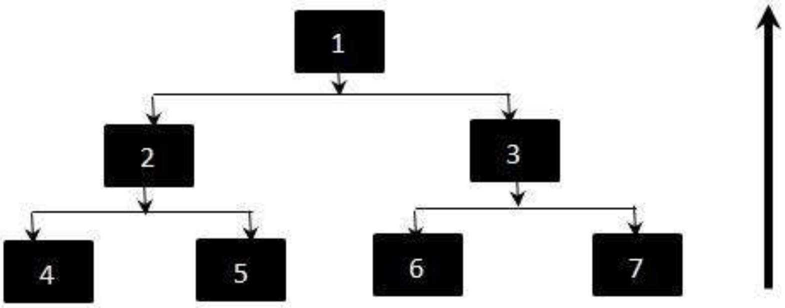
Top-down Integration - Flow Diagram:



6.1.2.2 Bottom Up Integration Testing:

Each component at lower hierarchy is tested individually and then the components that rely upon these components are tested.

Bottom Up Integration - Flow Diagram:



6.1.3 System Testing:

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

In system testing, integration testing passed components is taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behaviour of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).

System Testing Process:

- Test Environment Setup- Create testing environment for the better-quality testing.
- Create Test Case- Generate test case for the testing process.
- Create Test Data- Generate the data that is to be tested.
- Execute Test Case- After the generation of the test case and the test data, test cases are executed.
- Defect Reporting- Defects in the system are detected.
- Regression Testing- It is carried out to test the side effects of the testing process.
- Log Defects- Defects are fixed in this step.
- Retest- If the test is not successful then again test is performed.

6.2 Non-Functional Testing:

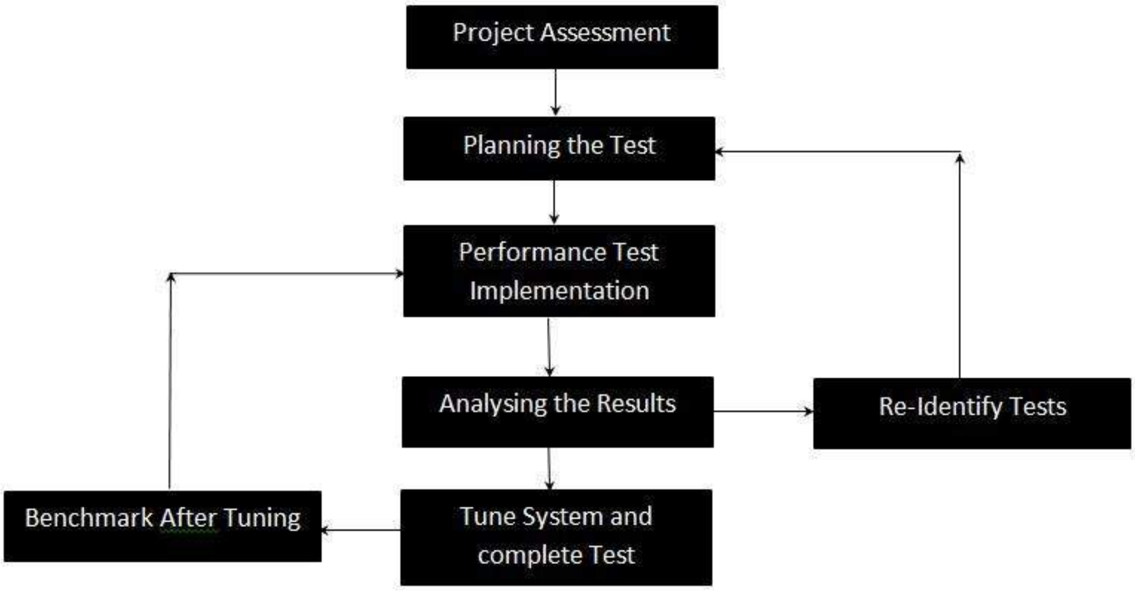
6.2.1 Performance Testing:

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

Attributes of Performance Testing:

- Speed
- Scalability
- Stability
- reliability

Performance Testing Process-

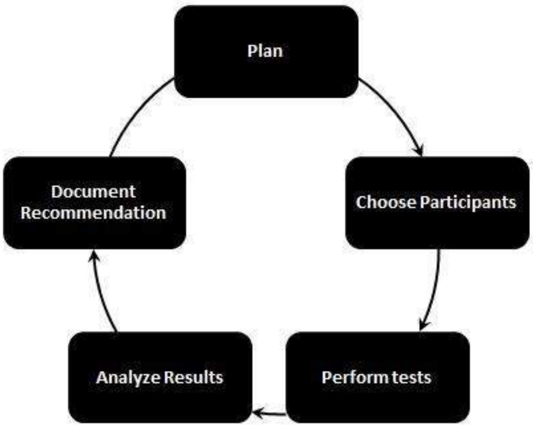


6.2.2 Usability Testing:

Usability testing, a non-functional testing technique that is a measure of how easily the system can be used by end users. It is difficult to evaluate and measure but can be evaluated based on the below parameters:

- Levels of Skill required to learn/use the software. It should maintain the balance for both novice and expert user.
- Time required to get used to in using the software.
- The measure of increase in user productivity if any.
- Assessment of a user's attitude towards using the software.

Usability Testing Process-



6.3 Verification:

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfils the requirements that we have.

The following are the various features and aspects on which verification was performed-

- It includes checking documents, design, codes and programs.
- Verification is the static testing
- It does *not* include the execution of the code.
- Methods used in verification are reviews, walkthroughs, inspections and desk-checking.
- It checks whether the software conforms to specifications or not.
- It can find the bugs in the early stage of the development.
- The goal of verification is application and software architecture and specification.
- It comes before validation.

6.4 Validation:

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. It is validation of actual and expected product.

The following are the various features and aspects on which validation was performed-

- It includes testing and validating the actual product.
- Validation is the dynamic testing.
- It includes the execution of the code.
- Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
- It checks whether the software meets the requirements and expectations of a customer or not.
- It can only find the bugs that could not be found by the verification process.
- The goal of validation is an actual product.
- It comes after verification.

7. CONCLUSION

This Google's e-Commerce android application consists of everything which can satisfy the customer to buy any google products online. The application is both available for customers as well as Google dealers. For the case of customers, they are allowed to browse and place orders and receive them at their doorstep and for the Google dealers, they get the option to add new upcoming google products in the application and get the live status of the orders placed by the customers and process them by their convenience.

The Google's e-Commerce app is an essential application which helps all the customers to place orders easily and receive them at their doorsteps.

8. FUTURE ENHANCEMENT

This android application can be further enhanced in following aspects:

- Introducing AI chatbot
- Improved UI
- Order Confirmation receipt can be sent to respective E-mail
- Size of the application is to be compressed below 50MB

9. BIBLIOGRAPHY

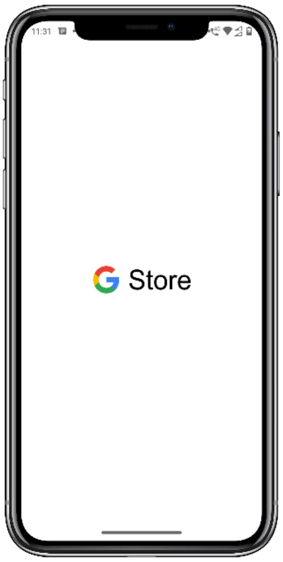
Websites Referred:

- www.medium.com
- www.firebase.google.com/docs/.com
- www. developer.android.com/studio/intro

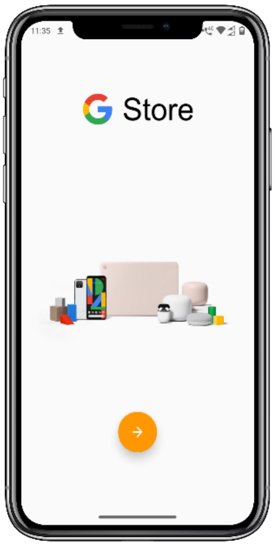
10. USER MANUAL

Customer Side Application

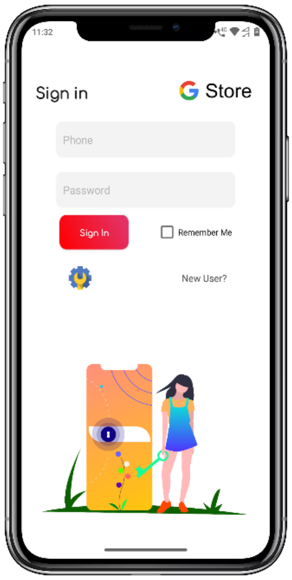
Splash Screen:



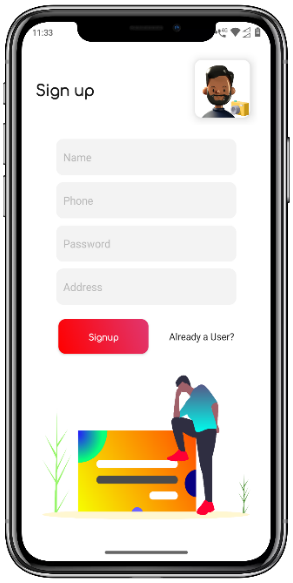
Welcome Screen:



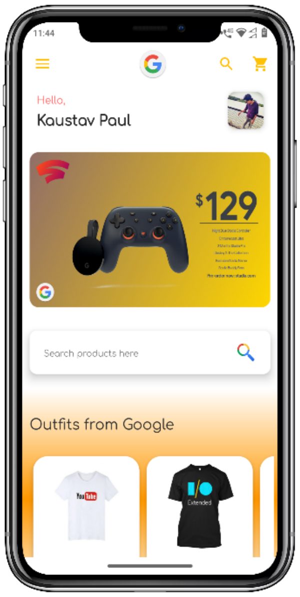
Login Activity(for customers):



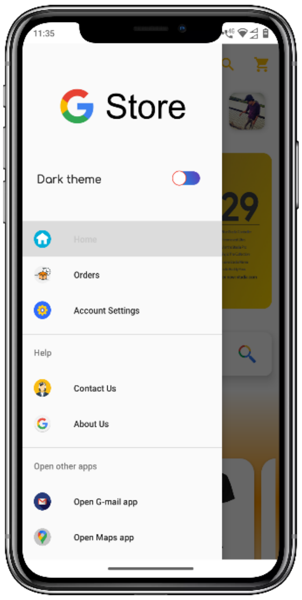
Register Activity:



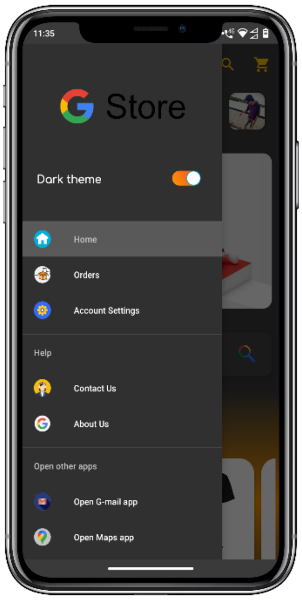
Home Activity:



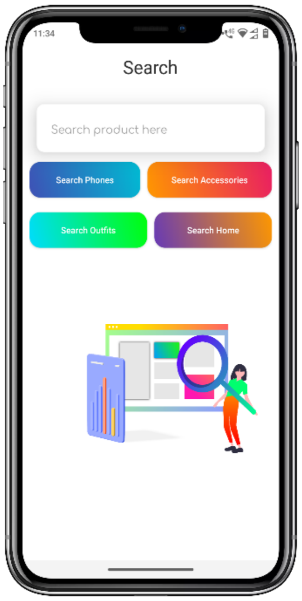
Navigation Drawer Activity:



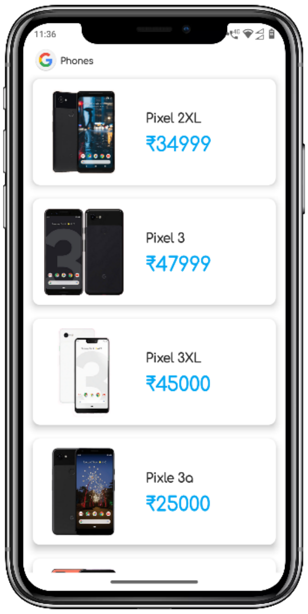
Navigation Drawer Activity (Dark):



Search Activity:



Phone List Activity:



11:36

Phones

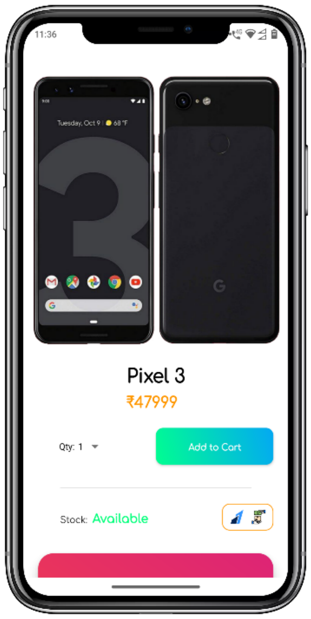
Pixel 2XL
₹34999

Pixel 3
₹47999

Pixel 3XL
₹45000

Pixle 3a
₹25000

Product Details Activity:



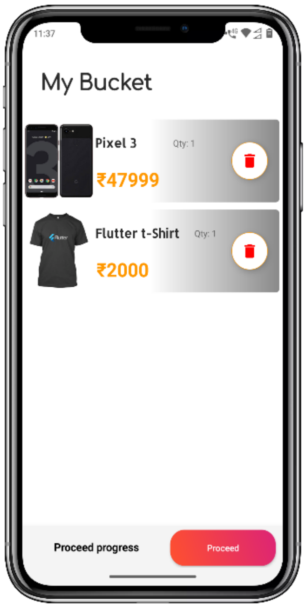
11:36

Pixel 3
₹47999

Qty: 1

Stock: Available

Cart Activity:



11:37

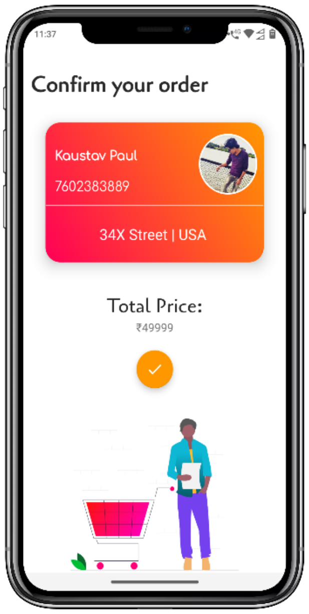
My Bucket

Pixel 3 Qty: 1
₹47999

Flutter t-Shirt Qty: 1
₹2000

Proceed progress

Confirm Order Activity:



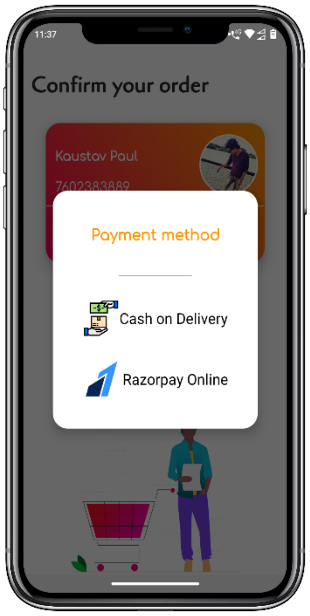
11:37

Confirm your order

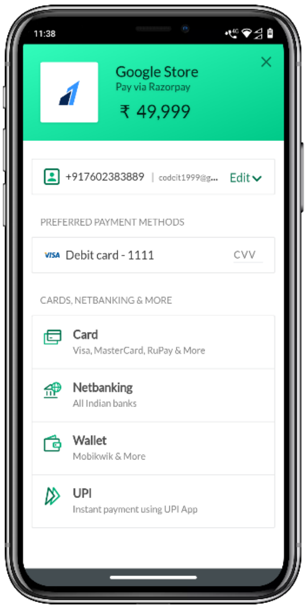
Kaustav Paul
7602383889
34X Street | USA

Total Price:
₹49999

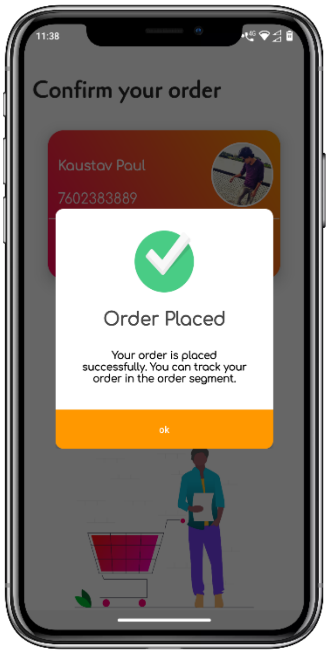
Choose payment method:



Razor-pay payment gateway:

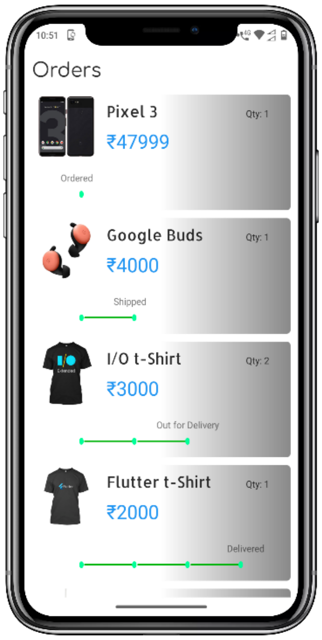


Order Confirmation Dialog:

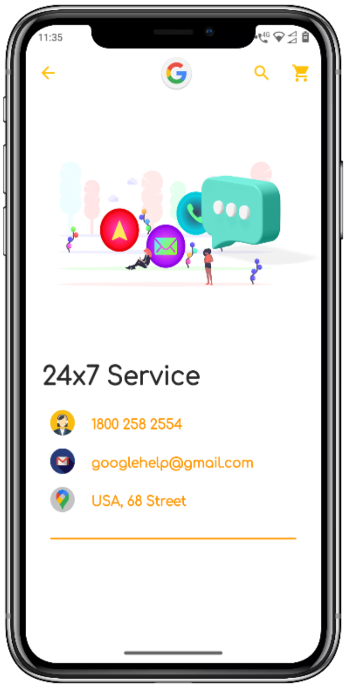


Your order is placed successfully. You can track your order in the order segment.

Order Fragment:



Contact Us Fragment:



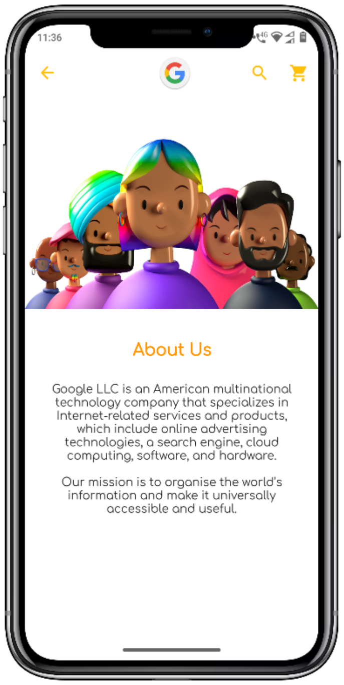
24x7 Service

1800 258 2554

googlehelp@gmail.com

USA, 68 Street

About Us Fragment:

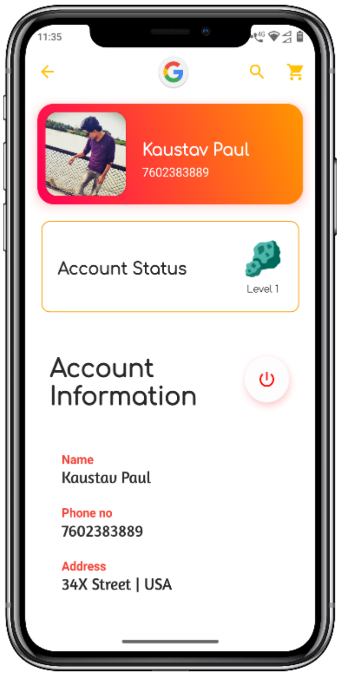


About Us

Google LLC is an American multinational technology company that specializes in Internet-related services and products, which include online advertising technologies, a search engine, cloud computing, software, and hardware.

Our mission is to organise the world's information and make it universally accessible and useful.

Account Settings Fragment:



Kaustav Paul
7602383889

Account Status
Level 1

Account Information

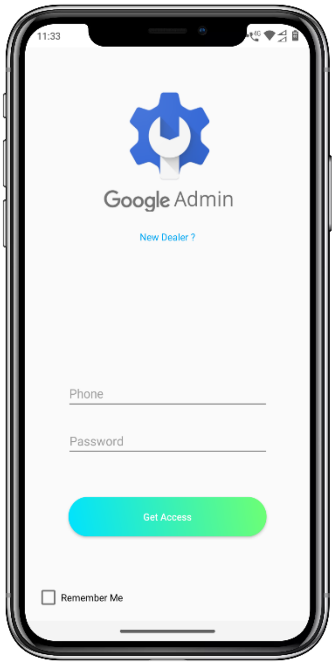
Name
Kaustav Paul

Phone no
7602383889

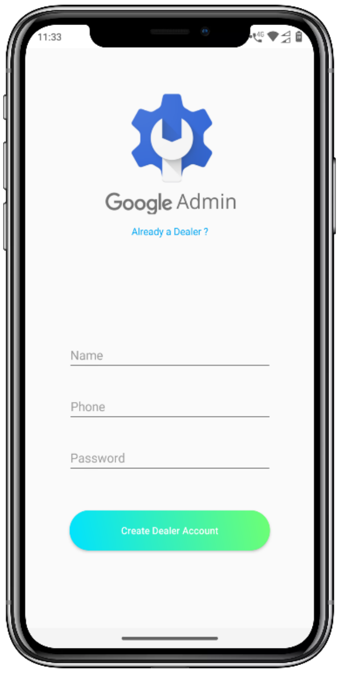
Address
34X Street | USA

Dealer Side Application

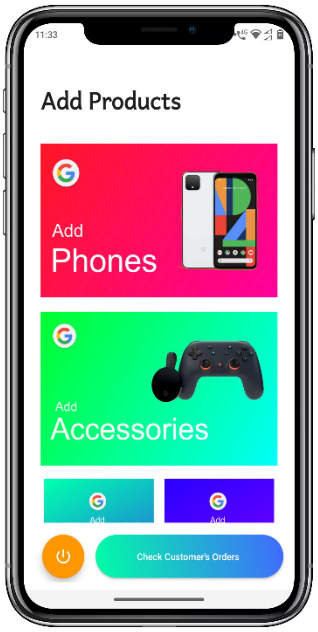
Login Activity(for Google dealer):



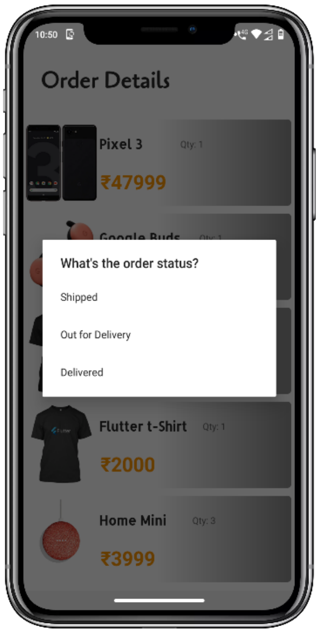
Register Activity (Dealer Side):



Dashboard Activity(Dealer):



Order Status Change Activity(Dealer):


10:50
Order Details
Pixel 3
Qty: 1
₹47999
Google Buds
Qty: 1
What's the order status?
Shipped
Out for Delivery
Delivered
Flutter t-Shirt
Qty: 1
₹2000
Home Mini
Qty: 3
₹3999