DECLARATION

We are the undersigned **Harshith V Nayak, Yathiraj U, Nagendra V Kini** students of Bachelor of Computer Application, Mahatma Gandhi Memorial College, Kunjibettu, Udupi hereby declared by the project work, presented in this report is our own work and has been carried out under the guidance of project guide **Prof.Rekha N Chandra,** Mahatma Gandhi Memorial College, Kunjibettu, Udupi

As per our knowledge, this work has not been submitted previously to the university by any other student.

There is no doubt that in spite of our strenuous efforts error might remain in the project. Naturally; we take full responsibility for any lack of clarity, occasional erratum or inexactness that might occur.
We have not submitted the matter embodied in this project for the award of any other degree.

Date:
Place: Udupi.

Harshith V Nayak

Prof. Rekha N Chandra **Yathiraj U**

(Project Guide) **Nagendra V Kini**

ACKNOWLEDGEMENT

Behind every achievement, there is a sea of gratitude to those who have activated this project. The magnitude of this project demanded the co-operation, guidance and help of many people. We have been fortunate enough to have this in the entire task of completion of our project on "**Shreee Loistics Management System".**

We would like to thank our principal **Prof. Lakshmi Narayan Karanth** for giving us opportunity to carry out our project.

We thank **Mr. M. Vishwanath Pai**, a source of inspiration and encouragement, head of the Department of Computer Science, Mahatma Gandhi Memorial College, Udupi for having permitted us to carry out our project work.

We are extremely grateful to express our overwhelming gratitude to our guide **Prof. Rekha N Chandra** Lecturer of Computer Science Department, Mahatma Gandhi Memorial College Udupi for giving us valuable guidance to undertake this project.

Last but not the least, we are indebted to all teaching and non-teaching staff members, Mahatma Gandhi Memorial College Udupi for making this project successful.

Thanking you

Harshith V Nayak

Yathiraj U

Nagendra V Kini

SHREE LOGISTICS MANAGEMENT SYSYTEM

Index

1. Introduction

a. Introduction to the System:

A fully equipped App that alone manages the entire functionality of a Logistics organization. Logistics management consists of the process of planning, implementing and controlling the efficient flow of raw-materials, work-in-progress and finished goods and related information – from point of consumption with a view to providing satisfaction to the users. By, Implementing a Logistics Management system not only helps logistics-owners to run the business, but also keep track of the entire system by storing the required data in a database .

i. Project Title:

Shree Logistics Management System

ii. Category:

Data Base Management

iii. Overview:

Shree Logistics Management System is a project that handles the overall internal working of a logistic industry. It provides the separate working areas for different users like manager ,employee and supplier. This app provides the basic tracking facility of the parcels. We can get the current locations of different items that are dispatched. By using firebase cloud storage as backend, we can retrieve the information from database anytime.

b. Background:

Brief note on Existing System:

Shreee Logistics Management System is an Existing system includes features such as user registration and authentication, user management, It provides authorized functionalities for the various types of users. The system also has security measures in place to protect user data and ensure privacy.

c. Objectives of the System:

- To execute proper planning on transportation modes and inventory available to satisfy the customers.
- To a smooth freight moving process and timely delivery of products and goods.
- To mitigate product damage, the proper monitoring of all product movements.
- To achieve desired levels of delivered service and quality at lowest possible cost.

d. Scope of the System:

- Inventory control and warehouse management is provided
- Packing and material handling are integrated throughout a network

e. End Users:

End users are the users who have registered themselves with their Email, Username and Password or also by using Google Sign-in method. These users can use this application to handle the internal workings of logistic industry.

f. Software/Hardware used for the development:

Software:

- **Front End: Android studio**

  Android Studio is the official **Integrated Development Environment (IDE)** for **Google's Android operating system**, built on **JetBrains IntelliJ IDEA** software and designed specifically for android development. Android Studio provides more features that enhance our productivity while building Android apps. It has a flexible gradle-based build system and it also has a fast and feature-rich emulator for app testing. Android Virtual Device (Emulator) is used to run and debug apps in the Android Studio. Android Studio contains all the Android tools to design, test, debug, and profile the application. The Android Studio uses Gradle to manage the project, a Build Automation Tool. It provides a unified environment where we can build apps for Android phones, tablets, etc. The official language for Android Development is Java. The app module contains the following folders:

  - **manifests:** It contains the AndroidManifest.xml file. Most of the code is generated automatically in this file, we need not change anything. The AndroidManifest.xml file contains information about the package, including components of the application such as activities, services, broadcast receivers, content providers, etc.

- **java:** It contains the source code of java files including the JUnit test code.
- **res:** It contains all non-code resources, UI strings, XML Layouts, and bitmap images.

❖ Back End : Firebase

Firebase is a Google backed application development software that enables the developers to develop android apps. The advantage of using online databases is that there are less chances of data being lost. The insertion and retrieval of data makes use of Firebase. Firebase is a Backend-as-a-Service, and it is a real time database which is basically designed for mobile applications.

Authentication:

Firebase Authentication aims to make building secure authentication systems easy, while improving the sign-in and onboarding experience for end users. It provides an end-to-end solution supporting email and password accounts, phone auth and Google, Twitter login and more. Firebase Authentication provides backend services, easy-to-use SDKs, and readymade UI libraries to authenticate users to the app. Firebase Authentication uses Firebase Dynamic Links to send the email link to the mobile devices.

Cloud Firestore :

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through realtime 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 Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

Hardware:

Operating System: Windows 7 or above

RAM: Minimum 8GB or more

Processor: Intel – i3 or above

Hard Disk: Minimum 100Gb or more

g. Software/Hardware used for the implementation:

Software:

Our Android project, Shree Logistics Management System, uses common software tools like the Android operating system and Android Studio. Hence on the software side, we expect the user to use an Android Operating System based smartphone with minimum of Android Version 7.0 (Nougat).

Hardware:

To run our Android app smoothly, use any Android-compatible smartphone or tablet. Make sure it has enough processing power, memory, and storage space. It should also support Wi-Fi, Bluetooth, and mobile data for seamless connectivity.

RAM: Minimum 2GB

Storage: Minimum 8GB

2. SRS

a. Introduction:

A Software Requirement Specification (SRS) is the complete description about what the software will do and how the software is expected to perform it. It also describes the functionality that the software needs to fulfil all the user needs. It is a formal report that enables the users to review whether SRS is according to their requirements.

b. Overall Description:

i. Product Perspective:

Shreee Logistics Management System is a user-friendly Android app with comprehensive features for different perspective. It prioritises privacy and security through end-to-end encryption and secure user verification. Furthermore, this app is a user-centric management app for Android that emphasises intuitive design, privacy, collaboration, personalization.

ii. Product Functions:

- It enables people to experience the product starting from a simple Email/Password registration process.
- It functions in providing a seamless user-interface for various activities.
- It provides the facility to add, update , remove the employees in the organization.

iii. User Characteristics:

- HypeChat appeals to **Tech-Enthusiasts**, who prioritise effective communication and value their privacy. These users appreciate the app's customization options, allowing them to personalise their messaging experience.
- With a focus on mobile accessibility, HypeChat caters to users who prefer **Secure Messaging** and enjoy staying connected on the go.

iv. General Constraints:

App should run on any android device where the android version is more than 7.

v. Assumptions:

- This app assumes that users join the app with the intention of performing their respective activities, and reach up there needs.
- The project may be updated in the near future with new features like payment, effective ways of tracking etc.

c. Functional requirements:

i. Modules:

1. User Registration/Login
2. Admin module

3. Employee module
4. Supplier module

ii. Module Description:

1. **User Registration/Login:**

   This module describes the login functionalities for a user of this application. A new user can register using Email, Username and Password. An existing user can login using Email and Password. User can retain the account in case he/she forgets the password by password reset method through email.

2. **Admin module:**

   This module provides the facility for the admin to add, remove or update the records related to the organization. Admin can also have the authority to see through the all other internal activities of the organization.

3. **Employee module:**

   This module defines the activities of a employee who can perform the activities like receiving the order, tracking the parcel. Employee can also have the access to manage the payment details of the dispatched goods.

4. **Supplier module:**

   This module allows the supplier to update his current location which is stored in database and can be retrieved any time ,any place. This shows the current progress of a dispatched goods in delivary.

3. System Design

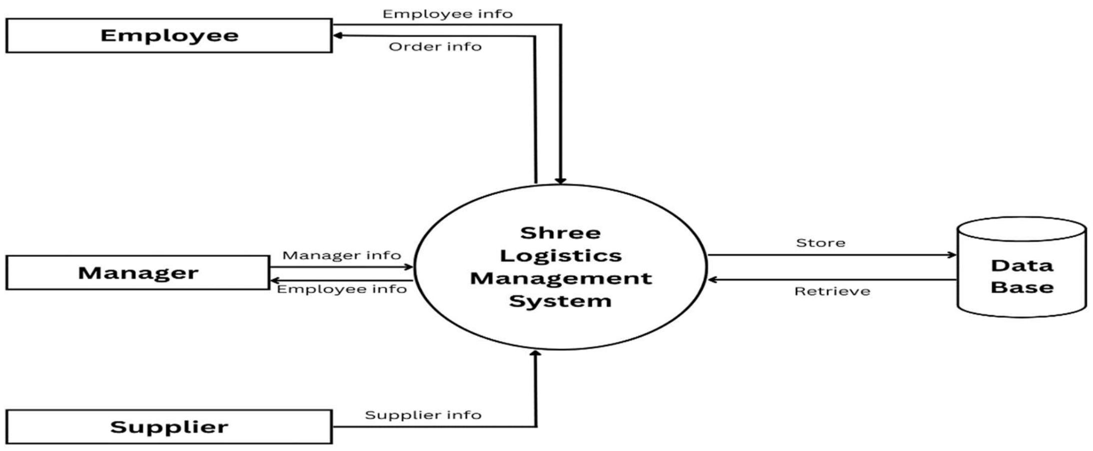
a. Introduction:

System design is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. It is the process of defining, developing and designing systems which satisfies the specific needs and requirements of a business or organisation.

b. <u>Description of Programs:</u>

i. Context flow Diagram:

CFD is one in which the entire system is treated as a single process and all its inputs, outputs, sinks and sources are identified. A context diagram, sometimes called a level 0 data-flow diagram, is drawn in order to define and clarify the boundaries of the software system. It identifies the flows of the information between the system and external entities. The entire software system is shown as a single process. It is used to establish the context and boundaries of the system to be modelled and identify the relationship of the system with the external entities.



CFD (DFD Level 0)

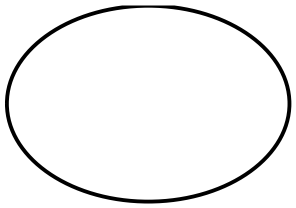
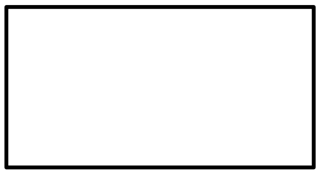
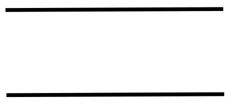
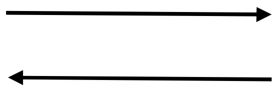
ii. Data Flow Diagram:

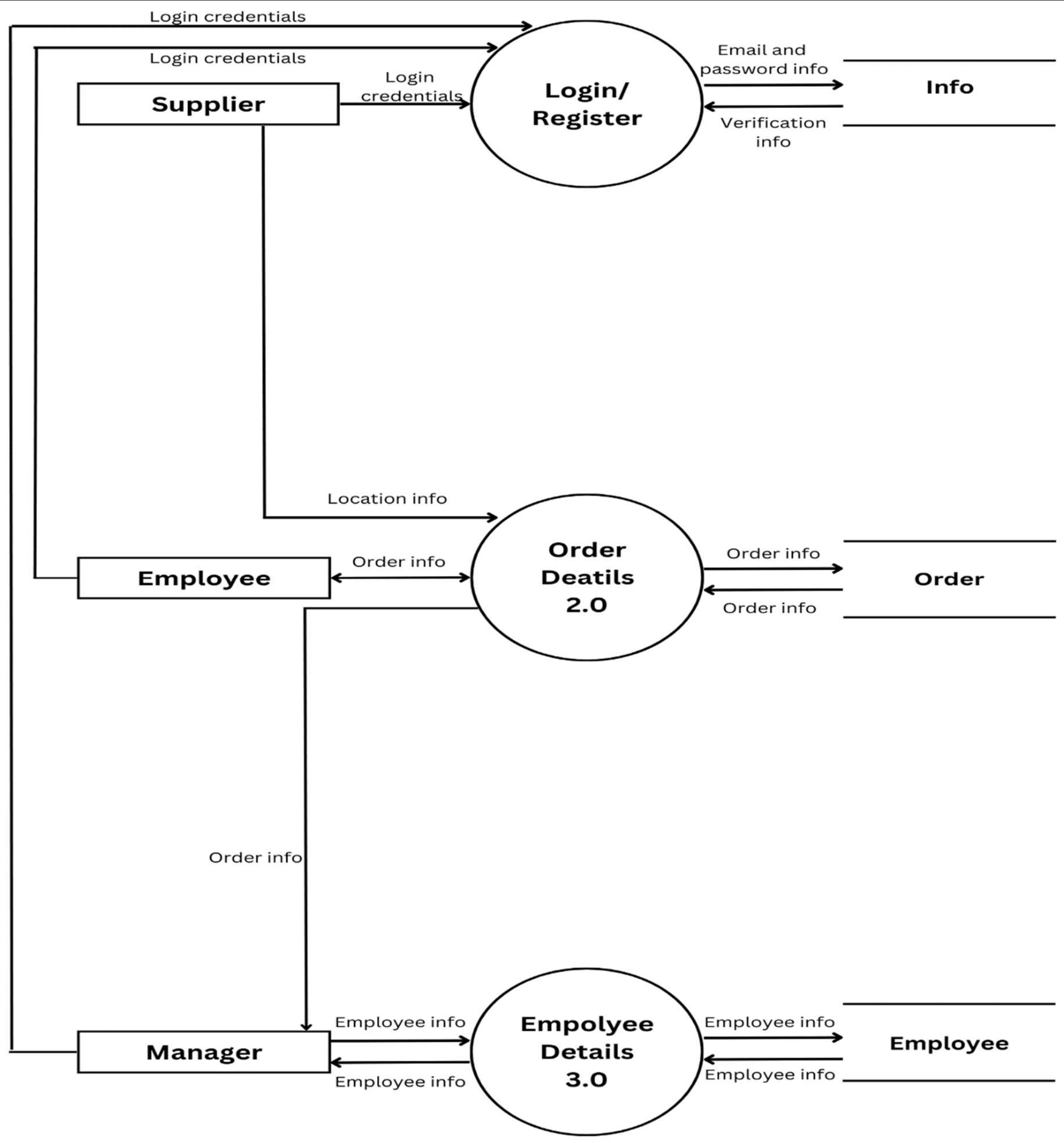
A data-flow diagram is a way of representing a flow through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data flow diagram has no control flow; there are no decision rules and no loops. Levels in DFD are numbered 0, 1, 2 and beyond.

The DFD server two purposes:

- To provide an indication of flow of data are transformed as they move through the system.
- To depict the function that transforms the data flow.

Notations for DFD Diagram:

Notations	Description
	A circle represents a **Process** that performs some transformation of input data to yield output data.
	A rectangle is used to represent an external entity i.e a source and a sink. System that produce information by the software is a **Source** and a system that receives or stores information produced by the system is a **Sink**.
	The open box represents a repository of data stored that is used by the software.
	An arrow represents one or more data items or data objects and is used to connect processes to each other or to sources and sinks. The arrow head indicates direction of data flow.



DFD Level 1

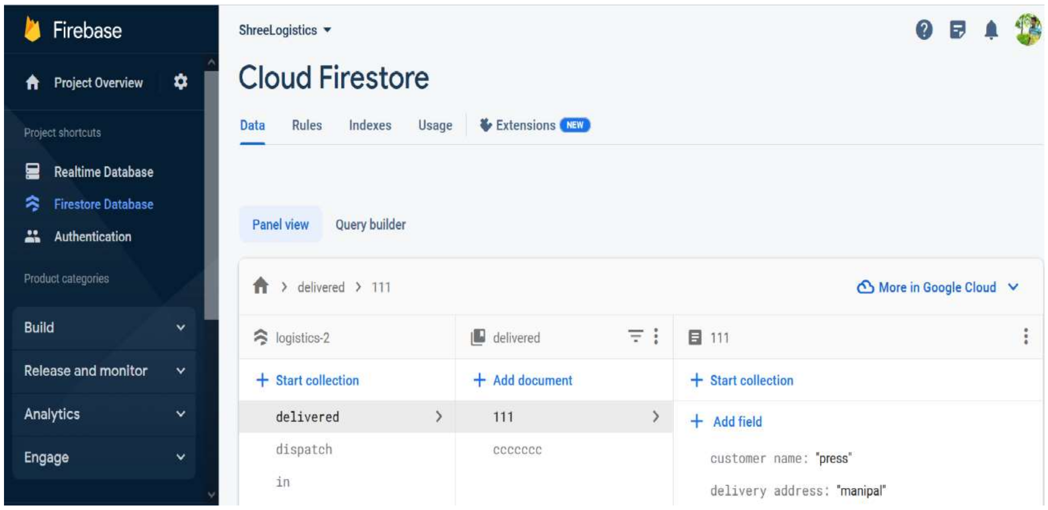
4. Database Design

a. Introduction:

Database design is the organisation of data according to database model.The designer determines what data must be stored and how the data elements interrelate.With this information,they can begin to fit the data to the database model.Database management system manages the data accordingly.Database design involves classifying data and identifying interrelationships.

b. Database Diagrams:

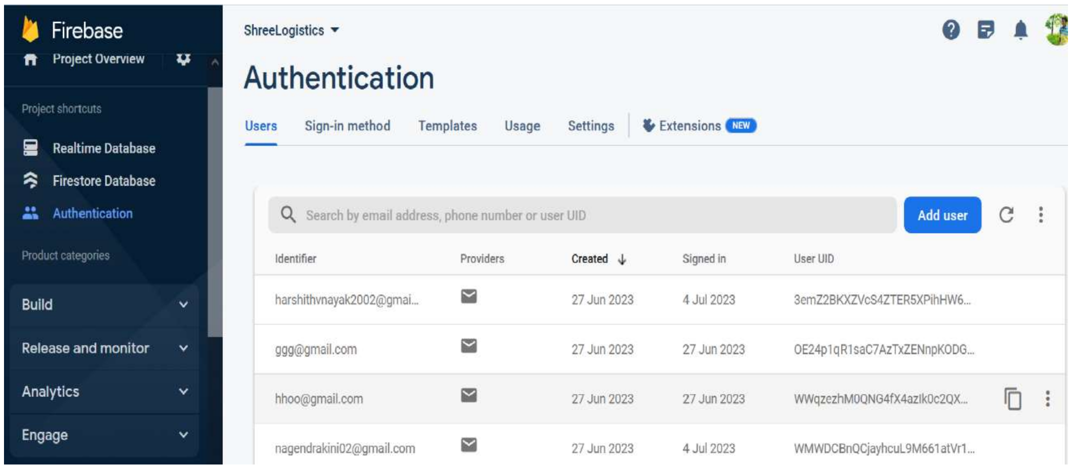
1. Firestore Database:



delivered > 111

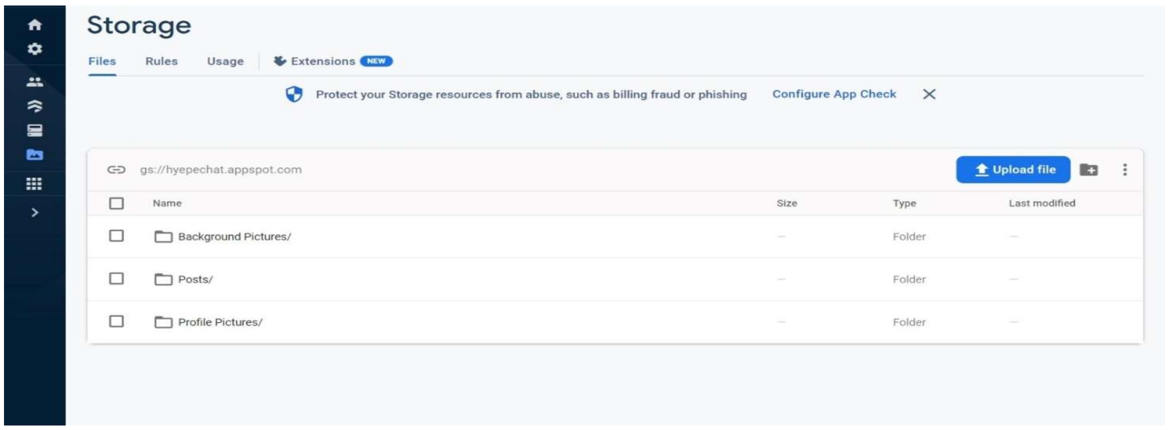
logistics-2	delivered	111
delivered	111	customer name: "press"
dispatch	ccccccc	delivery address: "manipal"
in		

2. Authentication:



Identifier	Providers	Created ↓	Signed in	User UID
harshithvnayak2002@gmai...	✉	27 Jun 2023	4 Jul 2023	3emZ2BKXZVcS4ZTER5XPihHW6...
ggg@gmail.com	✉	27 Jun 2023	27 Jun 2023	OE24p1qR1saC7AzTxZENnpKODG...
hhoo@gmail.com	✉	27 Jun 2023	27 Jun 2023	WWqzezhM0QNG4fX4azIk0c2QX...
nagendrakini02@gmail.com	✉	27 Jun 2023	4 Jul 2023	WMWDCBnQCjayhcuL9M661atVr1...

3. Storage:


Storage
Files
Rules
Usage
Extensions NEW
Protect your Storage resources from abuse, such as billing fraud or phishing
Configure App Check
Upload file

gs://hyepechat.appspot.com

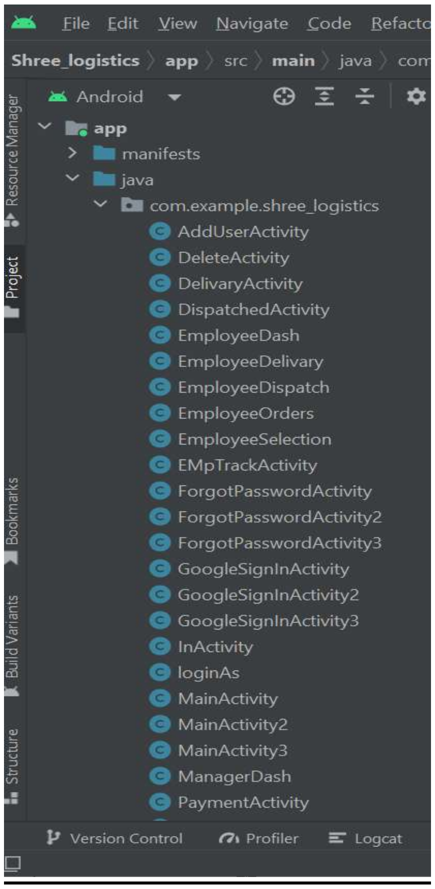
Name	Size	Type	Last modified
Background Pictures/	–	Folder	–
Posts/	–	Folder	–
Profile Pictures/	–	Folder	–

c. **Data Dictionary:**

1. Login @gmail + password
    - • Email={0-9|A-Z|a-z}*@
    - •Password="(?=.*[0-9])"+"(?=.*[a-z])"+"(?=.*[A-Z])"+"(?=.*[a-Z])"+"(?=.*[@#$%^&+=])"+"(?=\\S+$)"+".{10,}"+"$";
2. User @name + email + password
    - • Email={0-9|A-Z|a-z}*@
    - •Password="(?=.*[0-9])"+"(?=.*[a-z])"+"(?=.*[A-Z])"+"(?=.*[a-Z])"+"(?=.*[@#$%^&+=])"+"(?=\\S+$)"+".{10,}"+"$";

5. Program Code Listing

Project Structure:



```
app
  manifests
  java
    com.example.shree_logistics
      AddUserActivity
      DeleteActivity
      DelivaryActivity
      DispatchedActivity
      EmployeeDash
      EmployeeDelivary
      EmployeeDispatch
      EmployeeOrders
      EmployeeSelection
      EMpTrackActivity
      ForgotPasswordActivity
      ForgotPasswordActivity2
      ForgotPasswordActivity3
      GoogleSignInActivity
      GoogleSignInActivity2
      GoogleSignInActivity3
      InActivity
      loginAs
      MainActivity
      MainActivity2
      MainActivity3
      ManagerDash
      PaymentActivity
```

Project Code:

LoginActivity:

Design File:

```
<?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=".MainActivity2"

android:background="@drawable/bgmanager">

<View

android:id="@+id/view10"

android:layout_width="4000dp"

android:layout_height="4000dp"

android:background="#B8000000"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />
```

```
<ScrollView

android:layout_width="match_parent"

android:layout_height="match_parent"

android:fillViewport="true"

app:layout_constraintBottom_toBottomOf="parent"

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">

<androidx.constraintlayout.widget.ConstraintLayout

android:layout_width="match_parent"

android:layout_height="match_parent">

<TextView

android:id="@+id/textView"

android:layout_width="142dp"

android:layout_height="73dp"

android:layout_marginTop="84dp"

android:fontFamily="cursive"

android:text="Login"

android:gravity="center"

android:textColor="@color/white"

android:textSize="53sp"
```

```
android:textStyle="bold"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.498"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" /

<EditText

android:id="@+id/inputEmail"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_marginStart="24dp"

android:layout_marginTop="60dp"

android:layout_marginEnd="24dp"

android:background="@drawable/input_bg"

android:drawableLeft="@drawable/ic_baseline_email_24"

android:drawablePadding="10dp"

android:ems="10"

android:fontFamily="serif"

android:hint="Email"

android:inputType="textEmailAddress"

android:paddingLeft="20dp"

android:paddingTop="13dp"

android:paddingRight="20dp"

android:paddingBottom="13dp"
```

```
android:textColor="@color/white"

android:textColorHint="@color/white"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/textView" />

<EditText

android:id="@+id/inputPassword"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_marginTop="16dp"

android:background="@drawable/input_bg"

android:drawableStart="@drawable/ic_baseline_security_24"

android:drawableEnd="@drawable/ic_baseline_visibility_off_24"

android:drawablePadding="15dp"

android:ems="10"

android:fontFamily="serif"

android:hint="Password"

android:inputType="textPassword"

android:longClickable="false"

android:paddingLeft="20dp"

android:paddingTop="13dp"

android:paddingRight="20dp"
```

```
android:paddingBottom="13dp"

android:textColor="@color/white"

android:textColorHint="@color/white"

app:layout_constraintEnd_toEndOf="@+id/inputEmail"

app:layout_constraintStart_toStartOf="@+id/inputEmail"

app:layout_constraintTop_toBottomOf="@+id/inputEmail" />

<TextView

android:id="@+id/forgotPassword"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="12dp"

android:layout_marginEnd="4dp"

android:fontFamily="serif"

android:text="Forgot Password"

android:textColor="@color/white"

android:textSize="15sp"

app:layout_constraintEnd_toEndOf="@+id/inputPassword"

app:layout_constraintTop_toBottomOf="@+id/inputPassword" />

<Button

android:id="@+id/btnLogin"

android:layout_width="0dp"

android:layout_height="wrap_content"

android:layout_marginTop="32dp"
```

```
android:background="@drawable/btn_round"

android:fontFamily="serif"

android:text="Login"

android:textStyle="bold"

android:textColor="@color/white"

android:textSize="17sp"

app:layout_constraintEnd_toEndOf="@+id/forgotPassword"

app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toStartOf="@+id/inputPassword"

app:layout_constraintTop_toBottomOf="@+id/forgotPassword" />

<TextView

android:id="@+id/alreadyHaveaccount"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="20dp"

android:fontFamily="serif"

android:text="Create new Account?"

android:textColor="@color/white"

android:textSize="17sp"

app:layout_constraintEnd_toEndOf="@+id/btnLogin"

app:layout_constraintHorizontal_bias="0.507"

app:layout_constraintStart_toStartOf="@+id/btnLogin"

app:layout_constraintTop_toBottomOf="@+id/btnLogin" />
```

```
<TextView

android:id="@+id/textView4"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_marginTop="68dp"

android:fontFamily="serif"

android:text="OR LOGIN IN WITH GOOGLE"

android:textColor="@color/white"

android:textSize="17sp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.495"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/alreadyHaveaccount"

app:layout_constraintVertical_bias="0.211" />

<View

android:id="@+id/view8"

android:layout_width="0dp"

android:layout_height="2dp"

android:layout_marginEnd="16dp"

android:background="@color/white"

app:layout_constraintBottom_toBottomOf="@+id/textView4"

app:layout_constraintEnd_toStartOf="@+id/textView4"
```

```
app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toStartOf="@+id/btnLogin"

app:layout_constraintTop_toTopOf="@+id/textView4"

app:layout_constraintVertical_bias="0.588" />

<View

android:id="@+id/view9"

android:layout_width="0dp"

android:layout_height="2dp"

android:layout_marginStart="16dp"

android:background="@color/white"

app:layout_constraintBottom_toBottomOf="@+id/textView4"

app:layout_constraintEnd_toEndOf="@+id/btnLogin"

app:layout_constraintHorizontal_bias="0.0"

app:layout_constraintStart_toEndOf="@+id/textView4"

app:layout_constraintTop_toTopOf="@+id/textView4"

app:layout_constraintVertical_bias="0.588" />

<ImageView

android:id="@+id/btnGoogle"

android:layout_width="50dp"

android:layout_height="50dp"

android:layout_marginBottom="28dp"

app:layout_constraintBottom_toBottomOf="parent"

app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintHorizontal_bias="0.498"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/textView4"

app:layout_constraintVertical_bias="0.38"

app:srcCompat="@drawable/google_logo" />

</androidx.constraintlayout.widget.ConstraintLayout>

</ScrollView>

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

Java File:

```
package com.example.shree_logistics;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.app.AlertDialog;

import android.app.ProgressDialog;

import android.content.Intent;

import android.graphics.drawable.ColorDrawable;

import android.os.Bundle;

import android.text.TextUtils;

import android.text.method.HideReturnsTransformationMethod;

import android.text.method.PasswordTransformationMethod;

import android.util.Patterns;

import android.view.MotionEvent;
import android.view.View;

import android.view.WindowManager;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.TextView;

import android.widget.Toast;

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

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

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

public class MainActivity2 extends AppCompatActivity {

TextView createnewAccount;

EditText inputEmail,inputPassword;

Button btnLogin;

String emailPattern="[a-zA-Z0-9._-]+@gmail+\\.+com";

ProgressDialog progressDialog;

FirebaseAuth mAuth;

FirebaseUser mUser;

ImageView btnGoogle;

TextView forgotPassword;

boolean passwordVisible;
```

```
@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main2);

createnewAccount=findViewById(R.id.alreadyHaveaccount);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManage
r.LayoutParams.FLAG_FULLSCREEN);

inputEmail=findViewById(R.id.inputEmail);

inputPassword=findViewById(R.id.inputPassword);

btnLogin=findViewById(R.id.btnLogin);

btnGoogle=findViewById(R.id.btnGoogle);

forgotPassword=findViewById(R.id.forgotPassword);

progressDialog=new ProgressDialog(this);

mAuth=FirebaseAuth.getInstance();

mUser=mAuth.getCurrentUser();

createnewAccount.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

startActivity(new Intent(MainActivity2.this,RegisterActivity2.class));

finish();

}

});

btnLogin.setOnClickListener(new View.OnClickListener() {
```

```
@Override

public void onClick(View view) {

perforLogin();

}

});

btnGoogle.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Intent intent=new Intent(MainActivity2.this,GoogleSignInActivity.class);

startActivity(intent);

}

});

inputPassword.setOnTouchListener(new View.OnTouchListener() {

@Override

public boolean onTouch(View view, MotionEvent motionEvent) {

final int Right=2;

if(motionEvent.getAction()==MotionEvent.ACTION_UP){

if (motionEvent.getRawX()>=inputPassword.getRight()-
inputPassword.getCompoundDrawables()[Right].getBounds().width()){

int selection=inputPassword.getSelectionEnd();

if (passwordVisible)

{

inputPassword.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic_baseline
_visibility_off_24,0);
inputPassword.setTransformationMethod(PasswordTransformationMethod.getInstance());
passwordVisible=false;
}
else
{
inputPassword.setCompoundDrawablesRelativeWithIntrinsicBounds(0,0,R.drawable.ic_baseline
_visibility_24,0);
inputPassword.setTransformationMethod(HideReturnsTransformationMethod.getInstance());
passwordVisible=true;
}
inputPassword.setSelection(selection);
return true;
}
}
return false;
}
});
forgotPassword.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
AlertDialog.Builder builder=new AlertDialog.Builder(MainActivity2.this);
View dialogView=getLayoutInflater().inflate(R.layout.activity_forgot_password,null);
EditText emailBox=dialogView.findViewById(R.id.emailBox);
```

```
builder.setView(dialogView);

AlertDialog dialog=builder.create();

dialogView.findViewById(R.id.btnReset).setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String userEmail=emailBox.getText().toString();

if(TextUtils.isEmpty(userEmail)&&
!Patterns.EMAIL_ADDRESS.matcher(userEmail).matches()){

Toast.makeText(MainActivity2.this,"Enter your registered email
id",Toast.LENGTH_SHORT).show();

return;

}

mAuth.sendPasswordResetEmail(userEmail).addOnCompleteListener(new
OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isSuccessful()){

Toast.makeText(MainActivity2.this,"Check your email",Toast.LENGTH_SHORT).show();

dialog.dismiss();

}

else

{

Toast.makeText(MainActivity2.this,"Unable to send,failed",Toast.LENGTH_SHORT).show();

}
}
});
}
});
dialogView.findViewById(R.id.btnCancel).setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
dialog.dismiss();
}
});
if(dialog.getWindow() !=null){
dialog.getWindow().setBackgroundDrawable(new ColorDrawable(0));
}
dialog.show();
}
});
}
private void perforLogin() {
String email=inputEmail.getText().toString();
String password=inputPassword.getText().toString();
if(!email.matches(emailPattern))
{
inputEmail.setError("Enter Correct Email");
inputEmail.requestFocus();

}

else if(password.isEmpty() || password.length()<6)

{

inputPassword.setError("Enter Proper Password");



}

else

{

progressDialog.setMessage("Please Wait While Login...");

progressDialog.setTitle("Login");

progressDialog.setCanceledOnTouchOutside(false);

progressDialog.show();

mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {

@Override

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

if(task.isSuccessful())

{

progressDialog.dismiss();

sendUserToNextActivity();

Toast.makeText(MainActivity2.this,"Login Successful",Toast.LENGTH_SHORT).show();

}
else

{

progressDialog.dismiss();

Toast.makeText(MainActivity2.this,"Login Unsuccessful,\nAccount Doesn't
Exist",Toast.LENGTH_SHORT).show();

}



}

});

}

}

private void sendUserToNextActivity() {

Intent intent=new Intent(MainActivity2.this,ManagerDash.class);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TA
SK);

startActivity(intent);

finish();

}

}
```

SignupEmailActivity:

Java File:

```
package com.example.shree_logistics;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Log;

import android.widget.Toast;



import com.google.android.gms.auth.api.signin.GoogleSignIn;

import com.google.android.gms.auth.api.signin.GoogleSignInAccount;

import com.google.android.gms.auth.api.signin.GoogleSignInClient;

import com.google.android.gms.auth.api.signin.GoogleSignInOptions;

import com.google.android.gms.common.api.ApiException;

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

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

import com.google.firebase.auth.AuthCredential;

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.auth.GoogleAuthProvider;

public class GoogleSignInActivity extends MainActivity {

private static final int RC_SIGN_IN = 101;

GoogleSignInClient mGoogleSignInClient;

FirebaseAuth mAuth;
```

```
FirebaseUser mUser;

ProgressDialog progressDialog;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

progressDialog=new ProgressDialog(this);

progressDialog.setMessage("Google Sign In...");

progressDialog.show();

GoogleSignInOptions gso=new
GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)

.requestIdToken(getString(R.string.default_web_client_id))

.requestEmail()

.build();

mAuth=FirebaseAuth.getInstance();

mUser=mAuth.getCurrentUser();

mGoogleSignInClient= GoogleSignIn.getClient(this,gso);

Intent signInIntent=mGoogleSignInClient.getSignInIntent();

startActivityForResult(signInIntent,RC_SIGN_IN);

}

@Override

public void onActivityResult(int requestCode,int resultCode,Intent data){

super.onActivityResult(requestCode,resultCode,data);

if(requestCode==RC_SIGN_IN){
```

```
Task<GoogleSignInAccount> task=GoogleSignIn.getSignedInAccountFromIntent(data);

try {

GoogleSignInAccount account=task.getResult(ApiException.class);

firebaseAuthWithGoogle(account.getIdToken());

}

catch (ApiException e){



Toast.makeText(this,""+e.getMessage(),Toast.LENGTH_SHORT).show();

progressDialog.dismiss();

finish();

}

}

}

private void firebaseAuthWithGoogle(String idToken) {

AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);

mAuth.signInWithCredential(credential)

.addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {

@Override

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

if (task.isSuccessful()) {

progressDialog.dismiss();

Toast.makeText(GoogleSignInActivity.this,"Login
Successful",Toast.LENGTH_SHORT).show();
FirebaseUser user = mAuth.getCurrentUser();

updateUI(user);

}

else {

progressDialog.dismiss();

Toast.makeText(GoogleSignInActivity.this,""+task.getException(),Toast.LENGTH_SHORT).sh
ow();

finish();



}

}

});

}

private void updateUI(FirebaseUser user) {

Intent intent=new Intent(GoogleSignInActivity.this,EmployeeDash.class);

intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TA
SK);

startActivity(intent);

finish();

}

}
```

<u>Mnager Module Code:</u>

<u>ManagerDash.java:</u>

```
package com.example.shree_logistics;

import android.annotation.SuppressLint;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Toast;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.cardview.widget.CardView;

import androidx.recyclerview.widget.RecyclerView;

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

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

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

import com.google.android.material.floatingactionbutton.FloatingActionButton;

import com.google.firebase.FirebaseApp;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QueryDocumentSnapshot;

import com.google.firebase.firestore.QuerySnapshot;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class ManagerDash extends AppCompatActivity {

CardView card;
```

```
UserAdapter adapter;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_managerdash);

FirebaseFirestore db = FirebaseFirestore.getInstance();

RecyclerView recyclerView = findViewById(R.id.recycler);

FloatingActionButton add = findViewById(R.id.addUser);

add.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

startActivity(new Intent(ManagerDash.this,AddUserActivity.class));

}

});

db.collection("users").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

@SuppressLint("NotifyDataSetChanged")

@Override

public void onComplete(@NonNull Task<QuerySnapshot> task) {

if (task.isSuccessful()) {

ArrayList<User> arrayList = new ArrayList<>();

for (QueryDocumentSnapshot doc : task.getResult()) {

User user1 = doc.toObject(User.class);
//user.setId(doc.getId());

//User user = new User();

Map<String, Object> user = new HashMap<>();

user.get("id");

user.get("name");

user.get("dob");

user.get("age");

user.get("phone");

user.get("degree");

user.get("designation");

arrayList.add(user1)

//u.setAge();

}

adapter = new UserAdapter(ManagerDash.this, arrayList);

recyclerView.setAdapter(adapter);

adapter.notifyDataSetChanged();

}

}

}

).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {
```

```
Toast.makeText(ManagerDash.this, "failed to get all the user list",
Toast.LENGTH_SHORT).show();

}

});

FloatingActionButton refresh = findViewById(R.id.refresh);

refresh.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Toast.makeText(ManagerDash.this, "dddfdg", Toast.LENGTH_SHORT).show();

db.collection("users").get().addOnCompleteListener(new
OnCompleteListener<QuerySnapshot>() {

@Override

public void onComplete(@NonNull Task<QuerySnapshot> task) {

if (task.isSuccessful()) {

ArrayList<User> arrayList = new ArrayList<>();

for (QueryDocumentSnapshot doc : task.getResult()) {

User user = doc.toObject(User.class);

//user.setId(doc.getId());

arrayList.add(user);

}

adapter = new UserAdapter(ManagerDash.this, arrayList);

recyclerView.setAdapter(adapter);

recyclerView.setVisibility(View.VISIBLE);

adapter.notifyDataSetChanged();
}

}

}).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(ManagerDash.this, "failed to get all the user list",
Toast.LENGTH_SHORT).show();

}

});

}



});

FloatingActionButton update = findViewById(R.id.update);

update.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

startActivity(new Intent(ManagerDash.this, UpdateActivity.class));

}

});

FloatingActionButton delete = findViewById(R.id.delete);

delete.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {
```

```
startActivity(new Intent(ManagerDash.this, DeleteActivity.class));

}

});

FloatingActionButton search = findViewById(R.id.search);

search.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

startActivity(new Intent(ManagerDash.this, SearchActivity.class));

}

});

}

}
```

AddUserActivity.java:

```
package com.example.shree_logistics;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.app.DatePickerDialog;

import android.os.Bundle;

import android.view.View;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import com.example.shree_logistics.R;
import com.google.android.gms.tasks.OnFailureListener;

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

import com.google.android.material.button.MaterialButton;

import com.google.android.material.textfield.TextInputEditText;

import com.google.firebase.firestore.DocumentReference;

import com.google.firebase.firestore.FirebaseFirestore;

import java.util.Calendar;

import java.util.HashMap;

import java.util.Map;

import java.util.Objects;

public class AddUserActivity extends AppCompatActivity {

DatePickerDialog datePicker;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_add_user);

FirebaseFirestore db = FirebaseFirestore.getInstance();

EditText firstname = findViewById(R.id.firstNameET);

EditText lastname = findViewById(R.id.lastNameET);

EditText id = findViewById(R.id.idET);

EditText dob = findViewById(R.id.dobET);

EditText age = findViewById(R.id.ageET);

EditText phone = findViewById(R.id.phoneET);
EditText degree = findViewById(R.id.degreeET);

EditText designation = findViewById(R.id.designationET);

MaterialButton addUser = findViewById(R.id.addUser);

dob.setOnClickListener(v -> {

final Calendar cldr = Calendar.getInstance();

int day = cldr.get(Calendar.DAY_OF_MONTH);

int month = cldr.get(Calendar.MONTH);

int year = cldr.get(Calendar.YEAR);

datePicker = new DatePickerDialog(AddUserActivity.this, (View, year1, monthOfYear,
dayOfMonth)

-> dob.setText(dayOfMonth + "/" + (monthOfYear + 1) + "/" + year1), year, month, day);

datePicker.show();



});

addUser.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Map<String, Object> user = new HashMap<>();

user.put("firstName", Objects.requireNonNull(firstname.getText().toString()));

user.put("lastName", Objects.requireNonNull(lastname.getText().toString()));

user.put("id", Objects.requireNonNull(id.getText().toString()));

user.put("dob", Objects.requireNonNull(dob.getText().toString()));

user.put("age", Objects.requireNonNull(age.getText().toString()));
user.put("phone", Objects.requireNonNull(phone.getText().toString()));

user.put("degree", Objects.requireNonNull(degree.getText().toString()));

user.put("designation", Objects.requireNonNull(designation.getText().toString()));

db.collection("users").document(id.getText().toString()).set(user).addOnSuccessListener(new
OnSuccessListener<Void>() {

@Override

public void onSuccess(Void avoid) {

Toast.makeText(AddUserActivity.this, "user added succussfully",
Toast.LENGTH_SHORT).show();

firstname.setText("");

lastname.setText("");

id.setText("");

dob.setText("");

age.setText("");

phone.setText("");

degree.setText("");

designation.setText("");

}

}).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(AddUserActivity.this, "failed to add user", Toast.LENGTH_SHORT).show();

}

});
```

```
}

});

}

}
```

DeleteActivity:

```
package com.example.shree_logistics;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

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.firestore.DocumentReference;

import com.google.firebase.firestore.DocumentSnapshot;

import com.google.firebase.firestore.FirebaseFirestore;

import com.google.firebase.firestore.QueryDocumentSnapshot;

import com.google.firebase.firestore.QuerySnapshot;
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class DeleteActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_delete);

FirebaseFirestore db = FirebaseFirestore.getInstance();

EditText searchid = findViewById(R.id.searchid);

Button btnsearch = findViewById(R.id.search);

Button btndelete=findViewById(R.id.delete);

EditText fnamed = findViewById(R.id.firstNameET);

EditText lnamed=findViewById(R.id.lastNameET);

EditText idd=findViewById(R.id.idET);

EditText dobd=findViewById(R.id.dobET);

EditText phoned=findViewById(R.id.phoneET);

EditText aged=findViewById(R.id.ageET);

EditText degreed=findViewById(R.id.degreeET);

EditText designationd=findViewById(R.id.designationET);

btnsearch.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {
```

```
db.collection("users").document(searchid.getText().toString()).get().addOnCompleteListener(ne
w OnCompleteListener<DocumentSnapshot>() {
@Override
public void onComplete(@NonNull Task<DocumentSnapshot>task) {
if(task.isSuccessful()) {
DocumentSnapshot doc = task.getResult();
if (doc.exists()) {
String docfname=doc.getString("firstName");
String doclname=doc.getString("lastName");
String docid=doc.getString("id");
String docdob=doc.getString("dob");
String docage=doc.getString("age");
String docphone=doc.getString("phone");
String docdegree=doc.getString("degree");
String docdesig=doc.getString("designation");
fnamed.setText(docfname);
lnamed.setText(doclname);
idd.setText(docid);
dobd.setText(docdob);
aged.setText(docage);
phoned.setText(docphone);
degreed.setText(docdegree);
designationd.setText(docdesig);
} else {

Toast.makeText(DeleteActivity.this, "record does not exists", Toast.LENGTH_SHORT).show();

}

}

else {

Toast.makeText(DeleteActivity.this, "failed", Toast.LENGTH_SHORT).show();

}

}

});

}

});

btndelete.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

db.collection("users").document(idd.getText().toString()).delete().addOnCompleteListener(new
OnCompleteListener<Void>() {

@Override

public void onComplete(@NonNull Task<Void> task) {

if(task.isSuccessful()){

Toast.makeText(DeleteActivity.this, "employee record deleted",
Toast.LENGTH_SHORT).show();

searchid.setText("");

fnamed.setText("");

lnamed.setText("");
```

```
idd.setText("");

dobd.setText("");

aged.setText("");

phoned.setText("");

degreed.setText("");

designationd.setText("");

}

else{

Toast.makeText(DeleteActivity.this, "failed", Toast.LENGTH_SHORT).show();

}

}

});

}

});

}

}
```

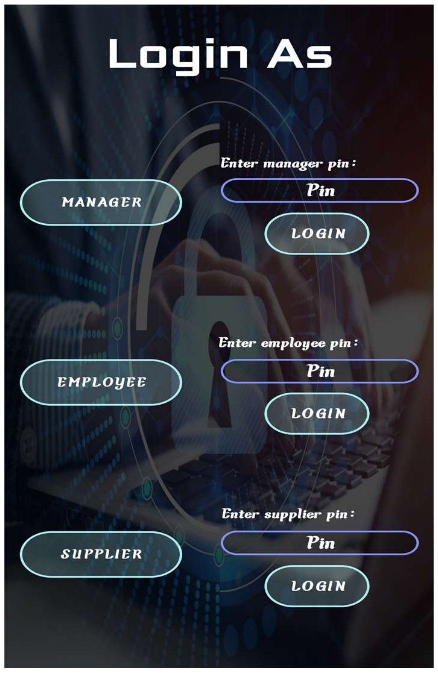
UpdateActivty:

```
package com.example.shree_logistics;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;
import android.widget.EditText;

import android.widget.Toast;

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.firestore.DocumentSnapshot;

import com.google.firebase.firestore.FirebaseFirestore;

import java.util.HashMap;

import java.util.Map;

public class UpdateActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity_update);

FirebaseFirestore db = FirebaseFirestore.getInstance();

EditText searchid = findViewById(R.id.searchid);

Button btnsearch = findViewById(R.id.search);

Button btnupdate=findViewById(R.id.update);

EditText fnamed = findViewById(R.id.firstNameET);

EditText lnamed=findViewById(R.id.lastNameET);

EditText idd=findViewById(R.id.idET);

EditText dobd=findViewById(R.id.dobET);
EditText phoned=findViewById(R.id.phoneET);

EditText aged=findViewById(R.id.ageET);

EditText degreed=findViewById(R.id.degreeET);

EditText designationd=findViewById(R.id.designationET);

btnsearch.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

db.collection("users").document(searchid.getText().toString()).get().addOnCompleteListener(ne
w OnCompleteListener<DocumentSnapshot>() {

@Override

public void onComplete(@NonNull Task<DocumentSnapshot> task) {

if(task.isSuccessful()) {

DocumentSnapshot doc = task.getResult();

if (doc.exists()) {

String docfname=doc.getString("firstName");

String doclname=doc.getString("lastName");

String docid=doc.getString("id");

String docdob=doc.getString("dob");

String docage=doc.getString("age");

String docphone=doc.getString("phone");

String docdegree=doc.getString("degree");

String docdesig=doc.getString("designation");

fnamed.setText(docfname);
lnamed.setText(doclname);

idd.setText(docid);

dobd.setText(docdob);

aged.setText(docage);

phoned.setText(docphone);

degreed.setText(docdegree);

designationd.setText(docdesig);

} else {

Toast.makeText(UpdateActivity.this, "record does not exists", Toast.LENGTH_SHORT).show();

}

}

else {

Toast.makeText(UpdateActivity.this, "failed", Toast.LENGTH_SHORT).show();

}

}

});

}

});

btnupdate.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

Map<String,Object> data= new HashMap<>();

data.put("firstName",fnamed.getText().toString());
data.put("lastName",lnamed.getText().toString());

data.put("id",idd.getText().toString());

data.put("dob",dobd.getText().toString());

data.put("age",aged.getText().toString());

data.put("phone",phoned.getText().toString());

data.put("degree",degreed.getText().toString());

data.put("designation",designationd.getText().toString());

db.collection("users").document(searchid.getText().toString()).set(data).addOnSuccessListener(n
ew OnSuccessListener<Void>() {

@Override

public void onSuccess(Void unused) {

Toast.makeText(UpdateActivity.this, "success", Toast.LENGTH_SHORT).show();

searchid.setText("");

fnamed.setText("");

lnamed.setText("");

idd.setText("");

dobd.setText("");

aged.setText("");

phoned.setText("");

degreed.setText("");

designationd.setText("");

}

}).addOnFailureListener(new OnFailureListener() {
```

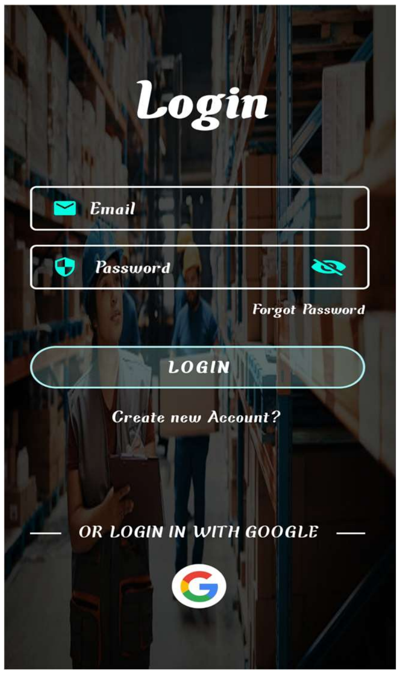
```
@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(UpdateActivity.this, "failed", Toast.LENGTH_SHORT).show();

}

});

}

});

}

}
```

6. User Interfaces

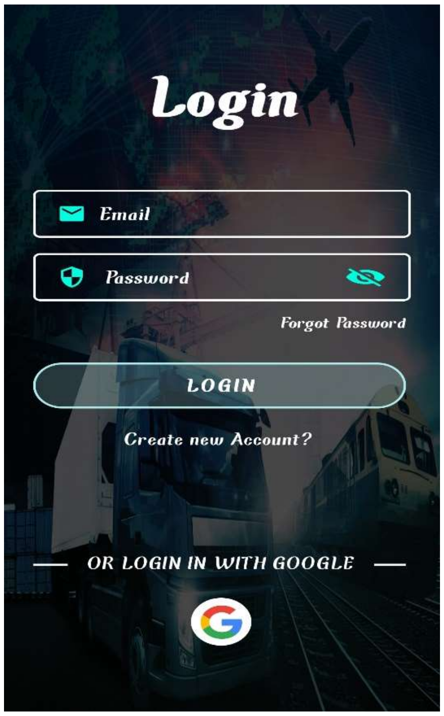
LoginAs Screen:



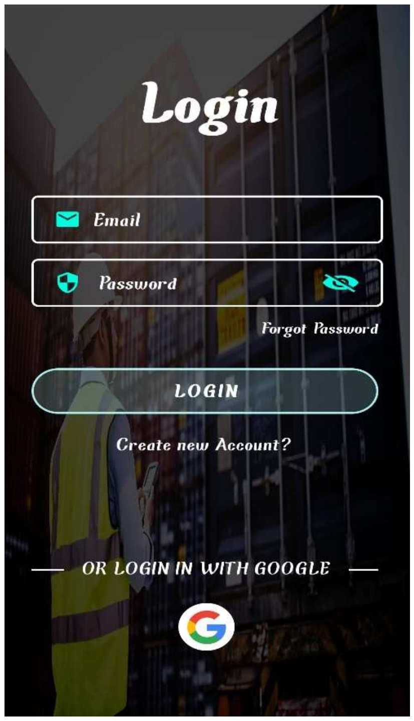
Manager Login:



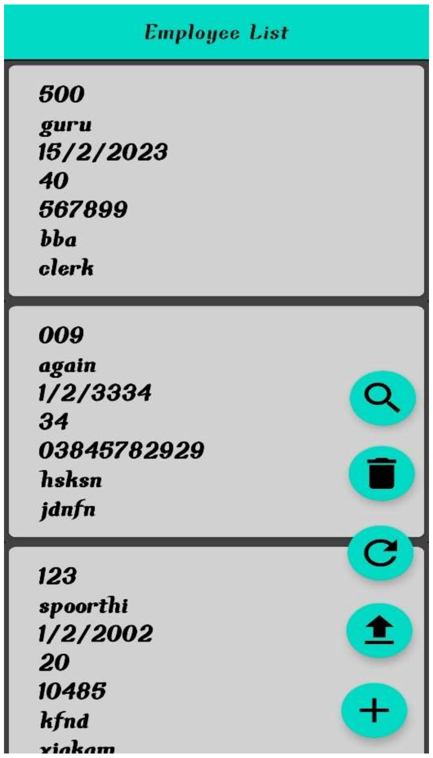
Employee Login:



Supplier Login:



Manager home screen:



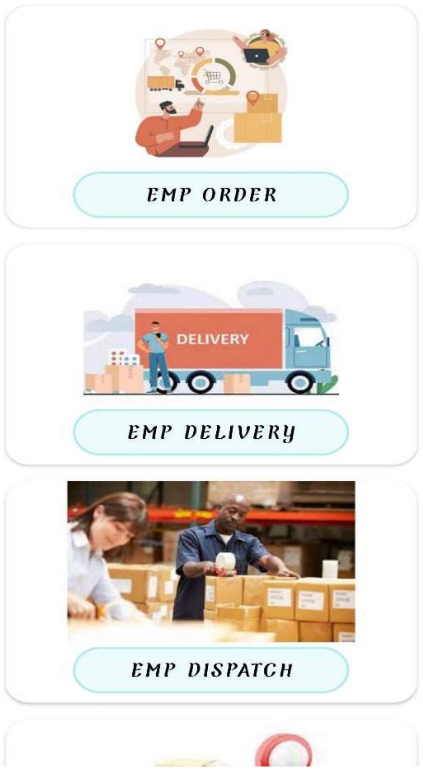
Employee List

500
guru
15/2/2023
40
567899
bba
clerk

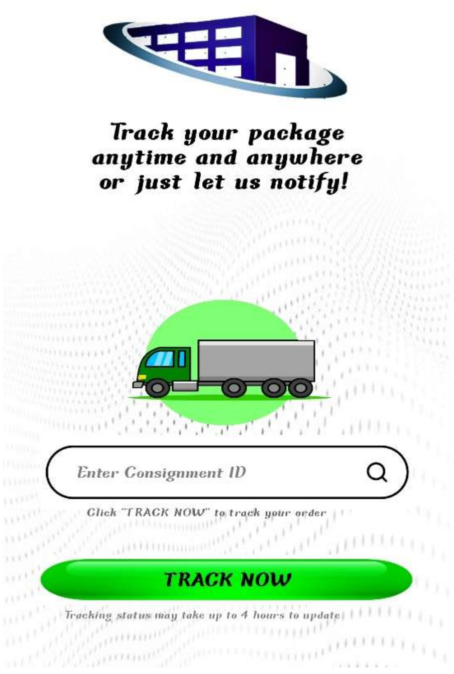
009
again
1/2/3334
34
03845782929
hsksn
jdnfn

123
spoorthi
1/2/2002
20
10485
kfnd

Employee home page:



Supplier home page:


Track your package anytime and anywhere or just let us notify!

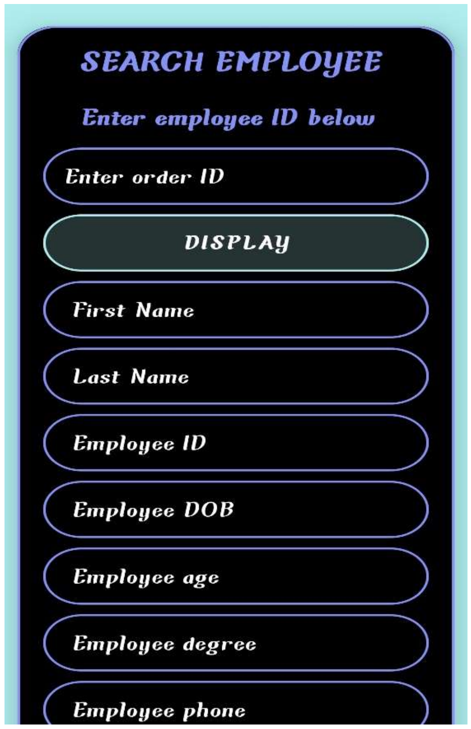
Enter Consignment ID

Click "TRACK NOW" to track your order

TRACK NOW

Tracking status may take up to 4 hours to update

Search employee page:


SEARCH EMPLOYEE

Enter employee ID below

Enter order ID

DISPLAY

First Name

Last Name

Employee ID

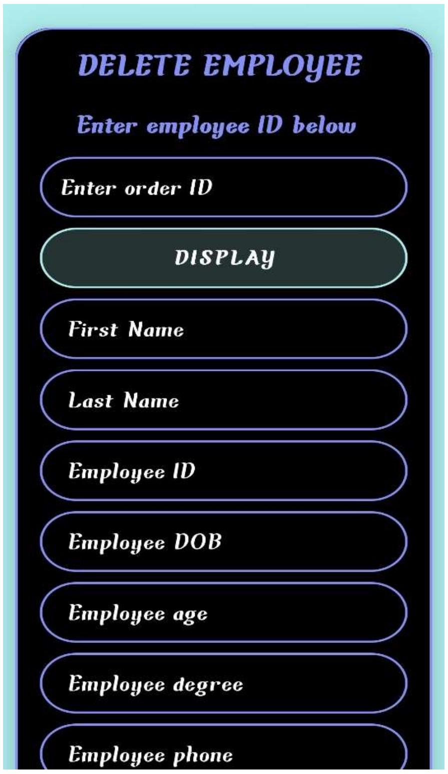
Employee DOB

Employee age

Employee degree

Employee phone

Delete employee page:


DELETE EMPLOYEE

Enter employee ID below

Enter order ID

DISPLAY

First Name

Last Name

Employee ID

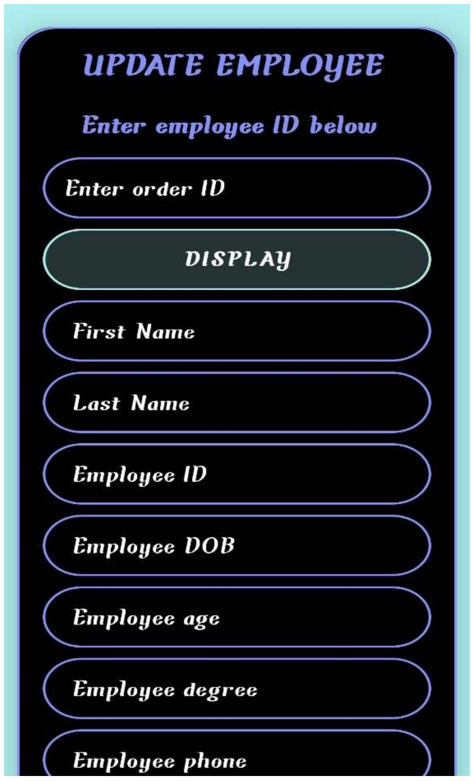
Employee DOB

Employee age

Employee degree

Employee phone

Update employee page:


UPDATE EMPLOYEE

Enter employee ID below

Enter order ID

DISPLAY

First Name

Last Name

Employee ID

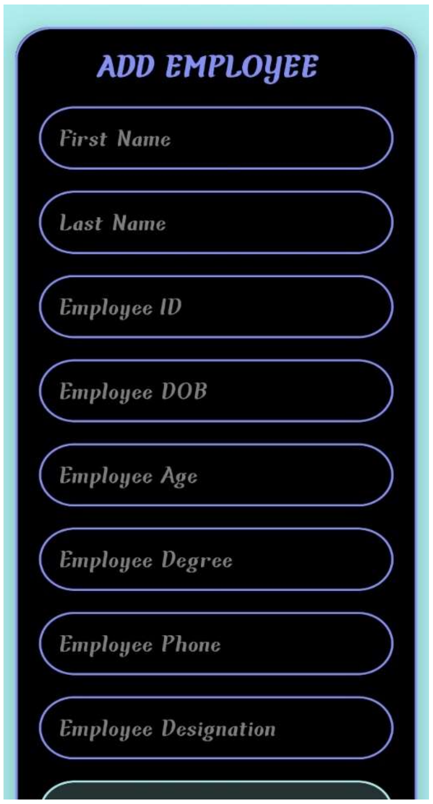
Employee DOB

Employee age

Employee degree

Employee phone

Add employee page:



7. Testing

a. Introduction:

Testing and implementation is the process, which tells the reality of efficiency and the flexibility of the system design. Reliability means how much the user is expecting from the system. Flexibility tells how much the user is comfortable and hopes for additional facilities with the system.

It is vital to the success of the system. System testing makes the logical assumption that if all parts of the system are correct, the goal will be successively achieved. It is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Testing presents an interesting anomaly of the software. The testing phase involves testing a system using various test data.

The first test of the system is to see whether it produces the correct output. When the software is tested the actual output is tested with the expected output. If there is discrepancy the sequence of instruction must be traced to determine the problem. Breaking the program down into self-contained portions, each of which can be checked at certain key points, facilitates the process.

The best program is worthless if it does not meet needs. The first step in system testing is to prepare a plan that will test all aspects of the system in a way that promotes its credibility among potential users. The development of software systems involves a series of production activities where opportunities for injunction of human error are enormous. Error may occur at the very imperfectly specified as well as later design and development stages.

b.Test Reports:

i. Unit Testing:

Unit testing focuses on verification efforts on the smallest unit of software design module. Using the unit test plans, prepared in the design phase of the system as guide, important control paths are tested to uncover errors within the boundary of the module. The interfaces of each of the modules under consideration are tested. Boundary condition was checked. All independent phases were exercised to ensure the error handling path was tested. Each unit was thoroughly tested to check if it might fail in any possible situation. This testing was carried out during the programming itself. At the end of the testing phase, each unit was found to be working satisfactorily, as regarded to the expected output from the module

ii. Integration Testing:

Integration testing is another aspect of testing that is generally done in order to uncover errors associated with the flow of data across interfaces. The unit-tested modules are grouped together and tested in small segments, which makes it easier to isolate and correct errors. This approach is continued until we have integrated all modules to form the system as a whole.

iii. System Testing:

Software testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. The testing phase involves the testing of a system using various test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested. Those test data, error were found and corrected by using some testing steps. Thus a series testing is performed on the system before it is ready for implementation.

c. Test Plan:

Test Plan	Test Objective
1	**Test for Email and Password**
2	**Test for Forgot Password**
3	**Test for Add Employee**
4	**Test for Remove Employee**
5	**Test for Update Employee**

d. Test Cases:

Test case 1:

Test Objective: Test for Email and Password

Test Data : Valid: Enter valid email and password, click on Log In.

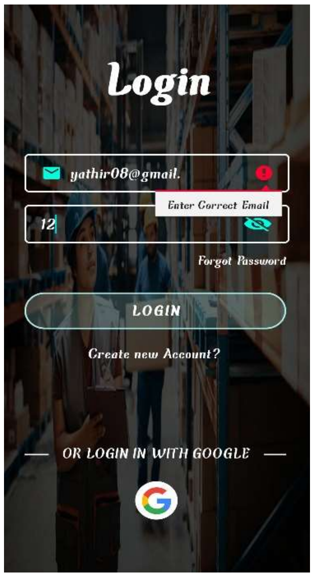
Invalid: Enter invalid email or email in wrong format and password.

Output : Valid: User is logged in and allowed to enter the App

Invalid: Alert message is shown regarding invalid entry.

Result : Valid: User is navigated to the Home Page of the App.

Invalid: Sign In page is retained and user is allowed to re-enter the login info.



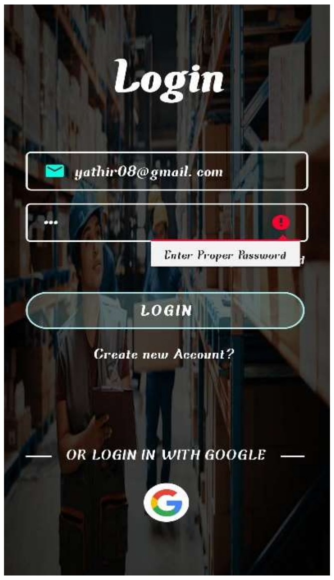
Login

yathir08@gmail.

Enter Correct Email

12

Forgot Password

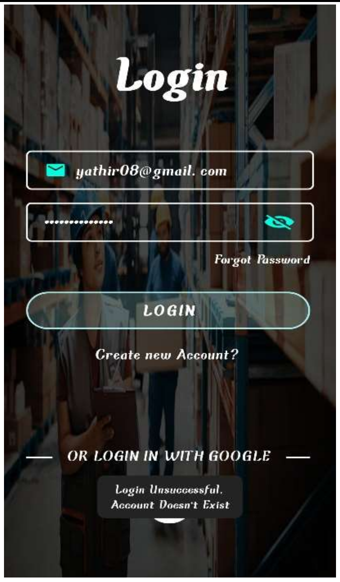


Login

yathir08@gmail. com

•••

Enter Proper Password



Login

yathir08@gmail. com

••••••••••••••

Forgot Password

Login Unsuccessful.
Account Doesn't Exist

Test case 2:

Test Objective: Test for Forgot Password

Test Data : Valid: Enter valid email to receive recovery password mail.

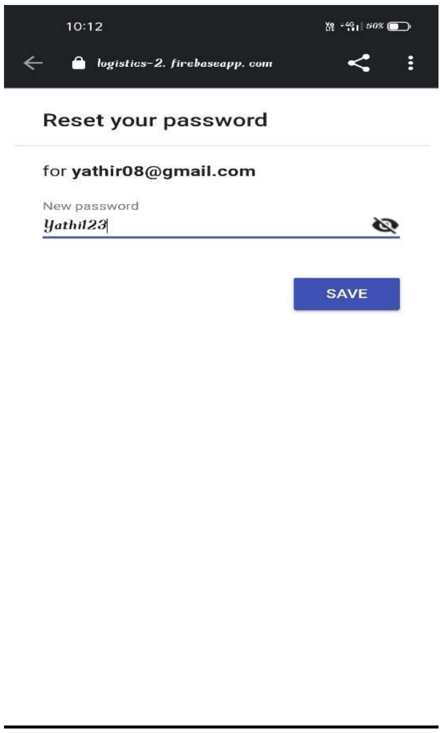
Invalid: Enter invalid email or email in wrong format.

Output : Valid: Valid email gets the recovery password mail.

Invalid: Toast message is shown regarding invalid entry.

Result : Valid: Password is updated.

Invalid: Sign In page is retained and user is allowed to re-enter the email id.


10:12

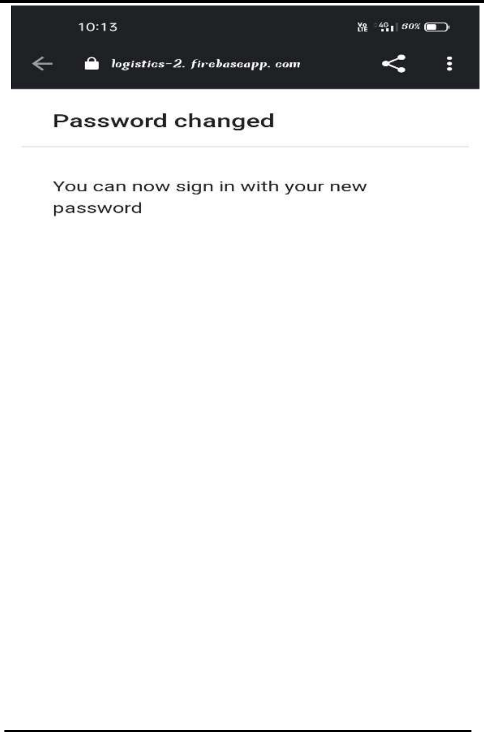
logistics-2. firebaseapp. com

Reset your password

for **yathir08@gmail.com**

New password

Yathi123


10:13

logistics-2. firebaseapp. com

Password changed

You can now sign in with your new password

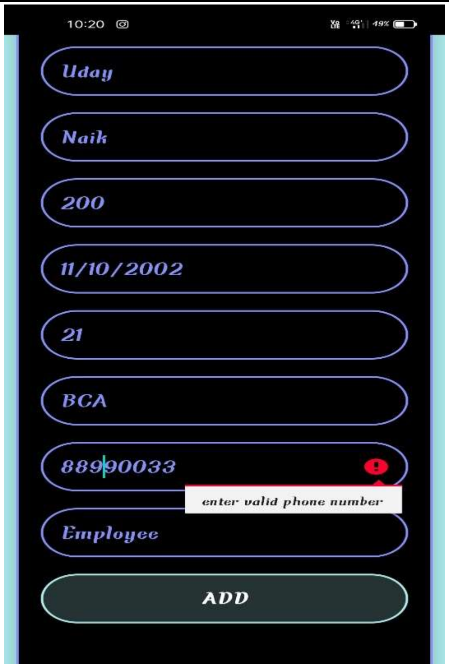
Test case 3:

Test Objective: Test for Add Employee

Test Data : Required data should be inserted

Output : Toast message is showed for success/fail attempts

Result : Employee detail is successfully added


Uday

Naik

200

11/10/2002

21

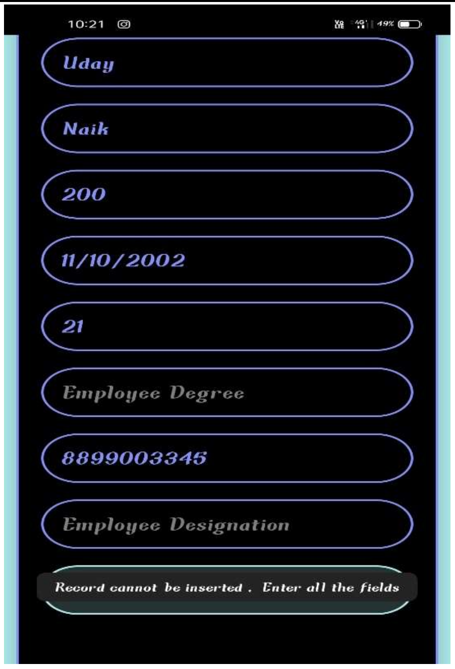
BCA

88990033

enter valid phone number

Employee

ADD


Uday

Naik

200

11/10/2002

21

Employee Degree

8899003345

Employee Designation

Record cannot be inserted . Enter all the fields

Test case 4:

Test Objective: Test for Remove Employee

Test Data : Valid: Valid Employee ID should be inserted

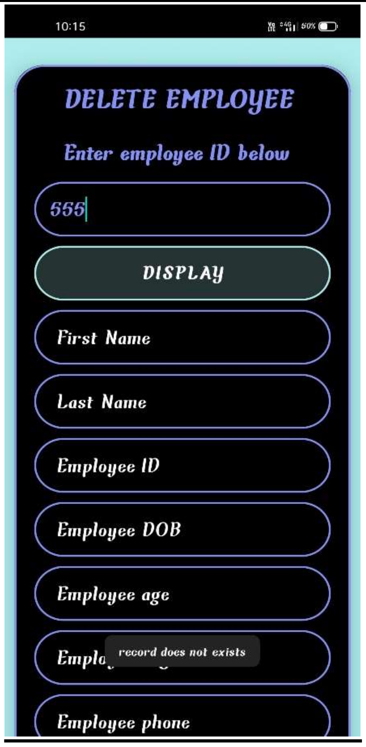
Invalid: Enter Wrong Employee ID

Output : Valid: Required Employee Detail Displayed

Invalid: Toast message is shown.

Result : Valid: Employee information is deleted

Invalid: Toast message is shown and information is not Deleted.


10:15

DELETE EMPLOYEE

Enter employee ID below

555

DISPLAY

First Name

Last Name

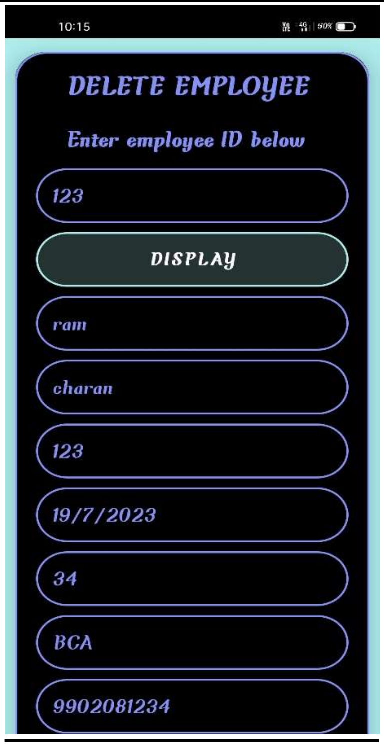
Employee ID

Employee DOB

Employee age

Emplo record does not exists

Employee phone


10:15

DELETE EMPLOYEE

Enter employee ID below

123

DISPLAY

ram

charan

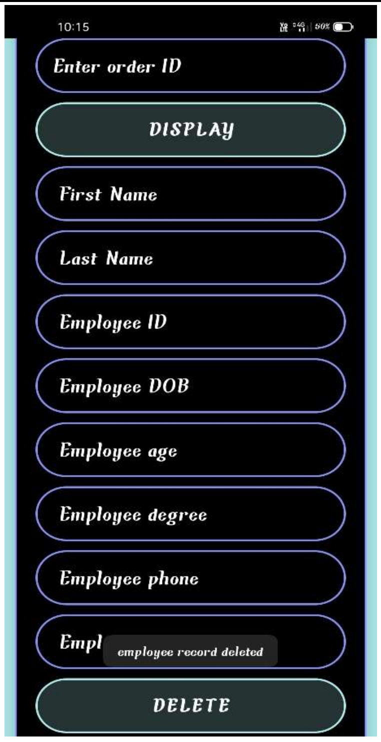
123

19/7/2023

34

BCA

9902081234


10:15

Enter order ID

DISPLAY

First Name

Last Name

Employee ID

Employee DOB

Employee age

Employee degree

Employee phone

Empl employee record deleted

DELETE

Test case 5:

Test Objective: Test for Update Employee

Test Data : Valid: Valid Employee ID should be inserted

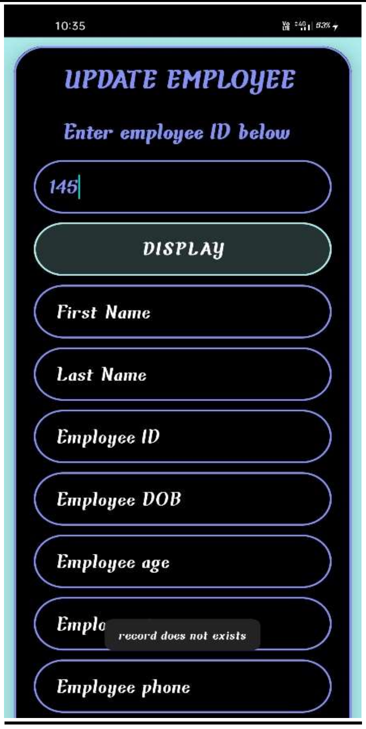
Invalid: Enter Wrong Employee ID

Output : Valid: Required Employee Detail Displayed

Invalid: Toast message is shown.

Result : Valid: Employee information is updated

Invalid: Toast message is shown and information is not updated.


10:35

UPDATE EMPLOYEE

Enter employee ID below

145

DISPLAY

First Name

Last Name

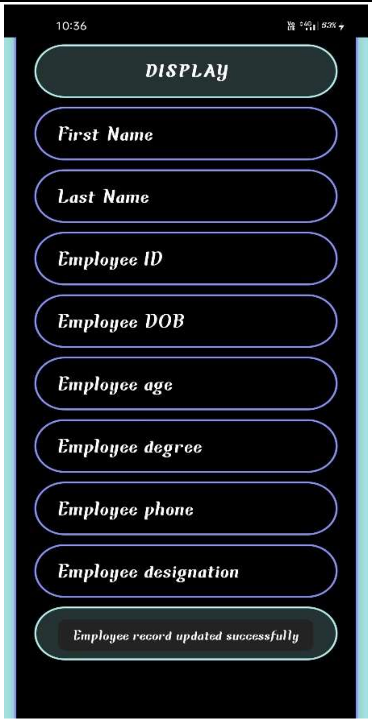
Employee ID

Employee DOB

Employee age

Emplo record does not exists

Employee phone


10:36

DISPLAY

First Name

Last Name

Employee ID

Employee DOB

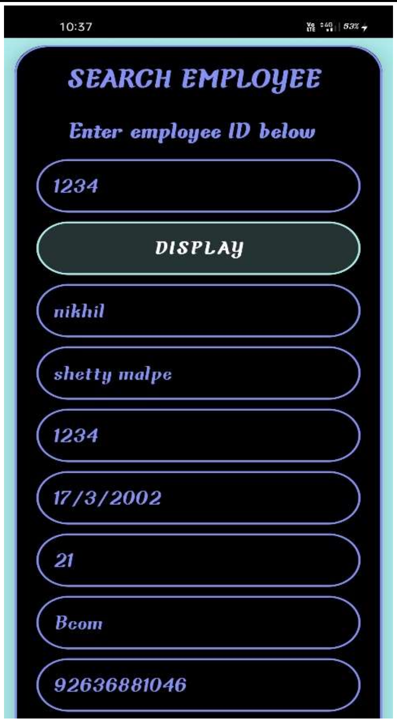
Employee age

Employee degree

Employee phone

Employee designation

Employee record updated successfully


10:37

SEARCH EMPLOYEE

Enter employee ID below

1234

DISPLAY

nikhil

shetty malpe

1234

17/3/2002

21

Bcom

92636881046

8. <u>Limitation</u>

- Only compatible with android OS.

9. <u>Scope for Enhancements</u>

- It can be developed in such a way that it can be supported for other OS.

10. <u>Abbreviation and acronyms</u>

Database Reference

A Firebase Reference represents a particular location in your database and can be used for reading or writing data to that Database location.

Data Snapshot

A DataSnapshot instance contains the data from a Firebase Database location. Any time you read data from the Database, you receive the data as a DataSnapshot.

CFD: Context Flow Diagram

DFD: Data Flow Diagram

SDK: Software Development Kit

API: Application programming interface

JDK: Java Development Kit

11. Bibliography

The content for this project has been taken from the following sources:

(www.stackoverflow.com

www.geeksforgeeks.org)

//Insert the reference links

THANK YOU