VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JNANA SANGAMA", BELAGAVI-590018

A MINI PROJECT REPORT ON

COLLEGE MANAGEMENT SYSTEM

Submitted in partial fulfilment of the requirement

For the award of degree of

Bachelor of Engineering

In

Computer Science and Engineering

By

AKSHAY R

[1KS19CS115]

Under the guidance of

Mr. HARSHAVARDHAN JR

Associate Prof, Dept. Of CSE

Mr. PRASHANTH H S

Asst. Prof, Dept. Of CSE

Department of Computer Science & Engineering

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-560109

2021-22

K.S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-56010

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that mini project work entitled "COLLEGE MANAGEMENT SYSTEM" carried out by Mr. AKSHAY R bearing USN 1KS19CS115 bonafide student of **K.S. Institute of Technology** in the partial fulfilment for the award of the **Bachelor of Engineering** in **Computer Science & Engineering** of the **Visvesvaraya Technological University, Belagavi,** during the year 2022. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has beenapproved as it satisfies the academic requirements in respect of mini Project work prescribed forthe said degree for the 6th semester

Dr . Rekha B Venkatapur
Prof & HOD, CS & E Department

Dr. Dilip Kumar K
Principal/Director, KSIT

Mr. Harshavardhan J R
Asst. Prof, Dept. Of CSE

Mr. Prashanth H S
Asst. Prof, Dept. Of CSE

Name of the Examiners **Signature with date**

1.

2.

ACKNOWLEDGEMENT

The successful completion of mini project would be incomplete without the mention of the people who made it possible and whose constant guidance crowned my effort with success.

I take this opportunity to express my sincere gratitude to our Management **K S Institute of Technology**, Bengaluru for providing all the resources required for the mini project.

I would express my gratitude **to Dr. K.V.A. Balaji**, C.E.O. K.S. Institute of Technology, Bengaluru, for facilitating me to complete mini project.

I would like to extend my gratitude to **Dr. Dilip Kumar K**, Principal/Director, K.S. Institute of Technology, Bengaluru, for his encouragement and providing all facilities for the accomplishment of this project.

I thank **Dr. Rekha B. Venkatapur**, Professor and Head, Department of Computer Science and Engineering, K.S. Institute of Technology, Bengaluru, for her encouragement.

I whole heartedly thank project guides, **Mr. Harshavardhan J R** and **Mr. Prashanth H S**, Department of Computer Science and Engineering, K.S. Institute of Technology, Bengaluru, for their support and guidance.

Finally, I would like to thank all the teaching and non-teaching staff of the college for their co-operation. Moreover, I thank all my **family** and **friends** for their invaluable support and cooperation.

Akshay R

(1KS19CS115)

ABSTRACT

In this project, we will be building a college management system. This application will allow us to access and control the student database on cloud. For this project, we will be using Firebase Authentication to facilitate and authorize the user login. Firebase Authentication provides a simple and accessible way to login. We can perform basic CRUD operations (create, retrieve, update, delete) on each document in each collection stored in the Firestore Database.

In this project, we will build an app that will take in login credentials and based on the type of user (student, teacher, admin) it will give dashboard with all the functionalities available for that user. This project is implemented in Android Studio using Java language.

TABLE OF CONTENTS

1. INTRODUCTION **1-5**

1 1. Overview 1

1.2. Problem Statement 1

1.3. Mobile Application Development Need & Importance 1-3

1.4. Android Studio 3-5

2. SYSTEM REQUIREMENTS **6**

2.1. Hardware and Software Requirements 6

3. SYSTEM DESIGN **7-17**

3.1. XML Design 7

3.2. XML Code 8-11

3.3. Java Code 11-17

4. IMPLEMENTATION **18**

4.1. Description 18

5. RESULTS **19-21**

6. CONCLUSION, FUTURE ENHANCEMENTS AND REFERENCES **22**

Chapter 1

INTRODUCTION

1. Overview

With the advent of new mobile technologies, the mobile application industry is advancing rapidly. Consisting of several operating systems like Symbian OS, iOS, blackberry, etc., Android OS is recognized as the most widely used, popular and user- friendly mobile platform. This open-source Linux kernel-based operating system offers high flexibility due to its customization properties making it a dominant mobile operating system. Android applications are programmed in java language. Google android SDK delivers a special software stack that provides developers an easy platform to develop android applications. Moreover, developers can make use of existing java IDEs which provides flexibility to the developers. Java libraries are predominant in the process of third- party application development. Cross-platform approaches make sure that developers do not have to develop platform-dependent applications. With the help of these approaches, an application can be deployed to several platforms without the need for changes in coding.

2. Problem Statement

The aim of this project is to build a College Management System. This application will authenticate the user and allow user to perform authorized database operations.

In this project, we will build an app that will take in login credentials and based on the type of user (student, teacher, admin) it will give dashboard with all the functionalities available for that user. We can perform basic CRUD operations (create, retrieve, update, delete) on each document in each collection stored in the Firestore Database.

This application will allows us to access and control the student database on cloud.

3. Mobile Application Development Need & Importance

In the past few years mobile app development has become a booming industry.

Currently, it is estimated that there are 2.3 million mobile app developers who are devoted to keeping up with the industry demand.

In fact, according to Apple, in 2013 1.25 million apps were registered in the Apple app store and accounted for 50 billion downloads and $5 billion paid to developers.

With these types of industry numbers, it soon becomes clear that mobile app development is a key factor for business success.

With the growing number of people accessing the Internet via smartphones and tablets, mobile app development has the unique ability to access a large number of potential consumers.

Not only have the sales of smartphone and tablets increased, but the amount of mobile apps installed has also grown exponentially. The Pew Research Internet Project indicates that approximately 50 percent of all smartphone users have mobile apps installed; of this percentage, two-thirds of the individuals are regular mobile app users. These statistics show that mobile apps have a unique opportunity to engage with an entirely new type of customer, one whom is constantly connected to the Internet and the global commerce space. In essence, a mobile app allows you to have millions of new customers at your fingertips. All that is left for you to do, is to develop an effective app and reap the benefits of your labors.

There are multiple benefits to creating and distributing a mobile app. Below are a few of the top benefits for businesses across a wide-variety of industries.

- **Build Loyalty**

  Mobile apps work to consistently increase customer loyalty, especially in the retail sector.

- **Reinforce your Brand**

  Mobile apps offer the unique opportunity for brand reinforcement through a new channel. Through mobile apps, customers are encouraged to download the free branded version, where they can customize preferences to fit their specific needs.

- **Increase your Accessibility**

  Smartphone and tablet users are constantly on the go; this means that they don't always have time to sign into a mobile website. And these mobile websites are designed for readability and navigation, NOT for process management. Mobile apps allow users to have easy, functional access to information, products, services and processes that they need in real-time and are optimized for hands on interaction.

- **Increase Sell-Through**

  Recent analysis suggests that mobile app users spend more time on a company's mobile app, then they spend on the company's mobile website.

- **Reduce On-premise costs**

  Most of the services that you provide at your business premises can be provided through android mobile applications. This would put you in a position where you do not need to pay workers to do that particular job.

- **Scope for Innovation**

  With every year, Android brings up innovative ideas and trends that symbolize the future. The devices and technologies used by users to interact with business changes pertaining to users' behaviors and needs.

As we continue to evolve into a mobile-centric society, it comes as no surprise that mobile apps are at the center of the developmental push. Developing a mobile app can go a long way towards propelling your company into the hands of new customers and future business success.

4. Android Studio

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as:

1. A flexible Gradle-based build system.
2. A fast and feature-rich emulator.
3. A unified environment where you can develop for all Android devices.
4. Apply Changes to push code and resource changes to your running app without restarting your app.
5. Code templates and GitHub integration to help you build common app features and import sample code.
6. Extensive testing tools and frameworks.
7. Lint tools to catch performance, usability, version compatibility, and other problems.
8. C++ and NDK support

1.2.1 Project Structure

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

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

Each app module contains the following folders:

- Manifests - Contains the AndroidManifest.xml file.
- Java - Contains the Java source code files, including JUnit test code.
- Res – Contains all non-code resources, such as XML Layouts, UI Strings and bitmap images.

1.2.2 The User Interface

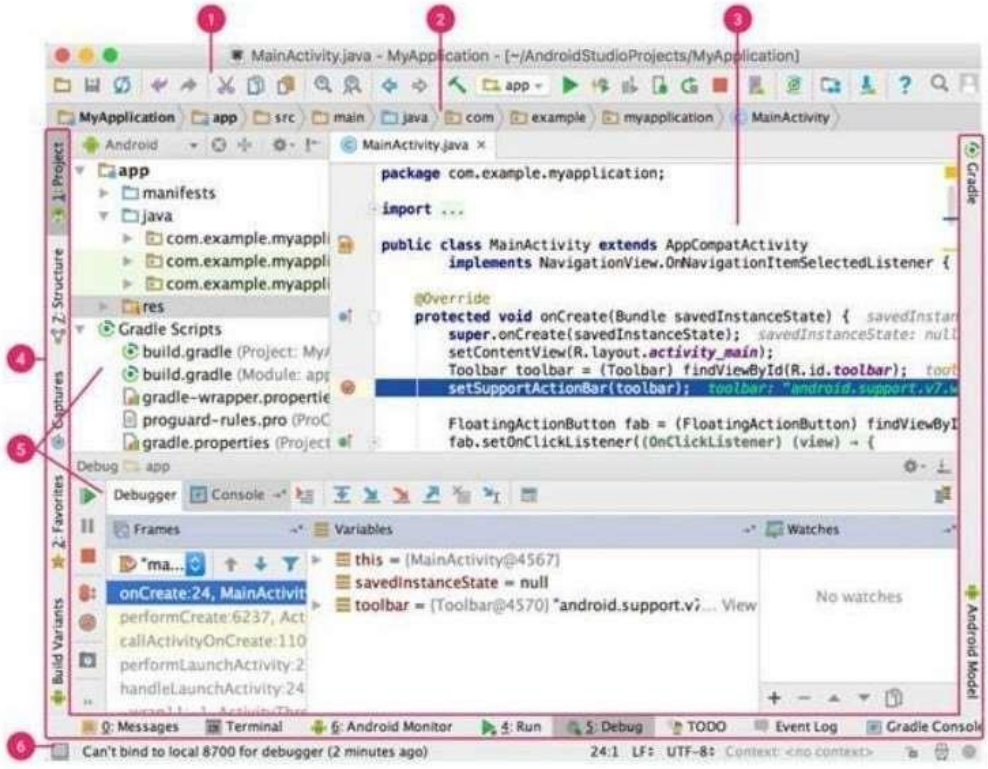

Figure 1.4.2. The Android Studio main window.

•The toolbar lets you carry out a wide range of actions, including running your app and launching Android tools

•The navigation bar helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.

•The Editor Window is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.

•The tool window bar runs around the outside of IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

•The tool windows give you access to specific tasks like project management, Search, version control, and more. You can expand them and collapse them.

•The status bar displays the status of your project and the IDE itself, as well as any warnings or messages.

4. Gradle build System

Android Studio uses Gradle as the foundation of the build system, with more Androidspecific capabilities provided by the Android plugin for Gradle. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for your app, with different features using the same projects and modules.
- Reuse code and resources across source sets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named build. Gradle. They are plain text files that use Groovy syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

Chapter 2

SYSTEM REQUIREMENTS

2.1 Software Requirements

Software requirements deal with defining software resource requirements and prerequisites that need to be installed on a computer to provide optimal functioning of an application. The following are the software requirements for the application:

- Operating System : Windows 10
- Java Development kit
- Android Studio

2.2 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware.

- CPU: Intel or AMD processor.
- Cores: Dual-Core (Quad-Core recommended).
- RAM: minimum 4GB (>4GB recommended).
- Graphics: Intel Integrated Graphics or AMD Equivalent.
- Display Resolution: 1366x768 (1920x1080 recommended).

Chapter 3

SYSTEM DESIGN

3.1 XML DESIGN

3.2 XML Code

activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".LoginActivity">

    <View
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@drawable/plz" />



    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:fillViewport="true"
        android:paddingBottom="30dp"
        tools:layout_editor_absoluteX="-16dp"
        tools:layout_editor_absoluteY="-26dp">

        <androidx.constraintlayout.widget.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <ImageView
                android:id="@+id/imageView2"
                android:layout_width="120dp"
                android:layout_height="120dp"
                android:layout_alignParentEnd="true"
                android:layout_marginTop="23dp"
                android:src="@drawable/ksit_logo_edited"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintTop_toTopOf="parent"
                android:contentDescription="@string/ksit_logo"/>

            <TextView
                android:id="@+id/textView"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="30dp"
                android:fontFamily="@font/habibi"
                android:text="@string/login"
                android:textAllCaps="true"
                android:textColor="#FFFFFF"
                android:textSize="40sp"
                android:textStyle="bold"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView2" />

    <EditText
        android:id="@+id/password"
        android:layout_width="280dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:background="@drawable/round_corner"
        android:drawableStart="@drawable/pwd_logo"
        android:drawablePadding="5dp"
        android:ems="10"
        android:fontFamily="@font/habibi"
        android:hint="@string/pwd"
        android:inputType="textPassword"
        android:padding="12dp"
        android:textColor="@color/white"
        android:textColorHint="@color/white"
        android:importantForAutofill="no"
        app:layout_constraintEnd_toEndOf="@+id/emailId"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/emailId"
        app:layout_constraintTop_toBottomOf="@+id/emailId" />

    <TextView
        android:id="@+id/forgot_pwd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="4dp"
        android:text="@string/forgot_pwd"
        android:textColor="@color/white"
        android:textSize="15sp"
        android:textStyle="bold|italic"
        app:layout_constraintEnd_toEndOf="@+id/password"
        app:layout_constraintTop_toBottomOf="@+id/password"  />

    <Button
        android:id="@+id/login_btn"
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:layout_marginTop="120dp"
        android:background="@drawable/round_corner"
        android:fontFamily="@font/habibi"
        android:text="@string/login"
        android:textColor="@color/white"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.495"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/password" />


    <EditText
        android:id="@+id/emailId"
        android:layout_width="280dp"
```

```
                android:layout_height="wrap_content"
                android:layout_marginTop="69dp"
                android:background="@drawable/round_corner"
                android:drawableStart="@drawable/email_logo"
                android:drawablePadding="5dp"
                android:ems="10"
                android:fontFamily="@font/habibi"
                android:hint="@string/email"
                android:inputType="textEmailAddress"
                android:padding="12dp"
                android:textColor="@color/white"
                android:textColorHint="@color/white"
                android:importantForAutofill="no"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.496"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/textView" />

            <ProgressBar
                android:id="@+id/progressBar"
                style="?android:attr/progressBarStyle"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="40dp"
                android:indeterminate="true"
                android:indeterminateTint="@color/white"
                android:indeterminateTintMode="src_atop"
                android:visibility="invisible"
                app:layout_constraintEnd_toEndOf="parent"
                app:layout_constraintHorizontal_bias="0.498"
                app:layout_constraintStart_toStartOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/login_btn" />
        </androidx.constraintlayout.widget.ConstraintLayout>
    </ScrollView>


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

activity_view_documents.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:id="@+id/activity_view_documents"

    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".ViewDocuments">

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginHorizontal="10dp"
        android:layout_marginVertical="10dp"
        android:layout_marginTop="4dp"
        android:background="@color/black"
        android:text="@string/add"
        android:textAllCaps="true"
        android:textColor="@color/white"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.444"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="60dp"
        app:layout_constraintTop_toBottomOf="@+id/button"
        tools:layout_editor_absoluteX="5dp" />

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

3.3 JAVA Code

AddCollection.java

```
import …
public class AddCollection extends AppCompatActivity {
```

```
    private EditText dob;
    Button submitBtn;
    EditText firstName, lastName, dateOfBirth, address, salary, mail;
    String fName, lName, dOb, add, sal, email;

    FirebaseFirestore db = FirebaseFirestore.getInstance();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_add_collection);

        dob = findViewById(R.id.dob);
        final Calendar calendar = Calendar.getInstance();
        final int year = calendar.get(Calendar.YEAR);
        final int month = calendar.get(Calendar.MONTH);
        final int day = calendar.get(Calendar.DAY_OF_MONTH);
        dob.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                DatePickerDialog       datePickerDialog       =       new
DatePickerDialog(view.getContext(),                                    new
DatePickerDialog.OnDateSetListener()  {
                    @Override

                    public  void  onDateSet(DatePicker  datePicker,  int
year, int month, int day) {
                        month = month+1;
                        String date = day+"-"+month+"-"+year;
                        dob.setText(date);
                    }
                },year,month,day);
                datePickerDialog.show();
            }
        });


        Intent i = getIntent();
        String collection = i.getStringExtra("collection");


db.collection(collection).addSnapshotListener((documentSnapshots, e) ->
{
            if (e != null)
                Toast.makeText(getApplicationContext(),  e.getMessage(),
Toast.LENGTH_SHORT).show();
        });

        submitBtn = findViewById(R.id.submitBtn);
        submitBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                firstName = findViewById(R.id.fName);
                fName = firstName.getText().toString();
                lastName = findViewById(R.id.lName);
                lName = lastName.getText().toString();
                address = findViewById(R.id.address);
                add = address.getText().toString();
```

```
                dateOfBirth = findViewById(R.id.dob);
                dOb = dateOfBirth.getText().toString();
                salary =  findViewById(R.id.salary);
                sal = salary.getText().toString();
                mail = findViewById(R.id.mail);
                email = mail.getText().toString();

                Map<String, Object> teacher = new HashMap<>();
                teacher.put("address", add);
                teacher.put("dob", dOb);
                teacher.put("fname", fName);
                teacher.put("lname", lName);
                teacher.put("salary", sal);
                teacher.put("email", email);
                teacher.put("uid","");

                db.collection("teacher")
                        .add(teacher)
                        .addOnSuccessListener(new
OnSuccessListener<DocumentReference>() {
                            @Override
                            public    void    onSuccess(DocumentReference
documentReference) {
                                Toast.makeText(getApplicationContext(),
"Added successfully", Toast.LENGTH_SHORT).show();
                                finish();
                            }
                        })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e)
{
                                Toast.makeText(getApplicationContext(),
"Error adding", Toast.LENGTH_SHORT).show();
                            }
                        });
            }
        });
    }
}
```

LoginActivity.java

```
import …
public class LoginActivity extends AppCompatActivity {

    public static String USER_TYPE;

    Button login;
    ProgressBar circularPB;
    EditText inputEmail, inputPass;
    TextView forgotPass;
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    private FirebaseAuth mAuth;
```

```
    final String emailRegEx = "^[a-zA-Z0-9_+&*-]+(?:\\." +
            "[a-zA-Z0-9_+&*-]+)*@" +
            "(?:[a-zA-Z0-9-]+\\.)+[a-z" +
            "A-Z]{2,7}$";
    final String passwordRegEx = "^(?=.*[0-9])"
            + "(?=.*[a-z])(?=.*[A-Z])"
            + "(?=.*[@#$%^&+=])"
            + "(?=\\S+$).{8,20}$";


    @Override
    public void onStart() {
        super.onStart();
        //  Check if  user  is  signed in  (non-null) and     update  UI
accordingly.
        FirebaseUser currentUser = mAuth.getCurrentUser();
        updateSignIn(currentUser);

    }

    private void updateSignIn(FirebaseUser user) {
        if (user != null)
            db.collection("admin")
                    .document(user.getUid())
                    .get()
                    .addOnCompleteListener(t -> {
                        if (t.getResult().exists()) {
                            Intent  i  =  new  Intent(LoginActivity.this,
AdminPage.class);
                            startActivity(i);
                        }
                        else
                            db.collection("teacher")

.addSnapshotListener((documentSnapshots, e) -> {
                                        if (documentSnapshots != null) {
                                            for  (DocumentSnapshot  doc:
documentSnapshots) {
                                                Map<String, Object> data
= doc.getData();
                                                String item = "";
                                                for(Map.Entry<String,
Object> entry: Objects.requireNonNull(data).entrySet()) {
                                                    item               =
item.concat(entry.getKey() + ": " + entry.getValue() + "\n");

if(item.contains("uid: " + user.getUid())){
                                                        Intent  i  =  new
Intent(LoginActivity.this, TeacherPage.class);


startActivity(i);
                                                    }
                                                }
                                            }
                                        });
                        });
    }
```

```
    private void checkForUser(String toString, String toString1) {
        db.collection("teacher")
                .addSnapshotListener((documentSnapshots, e) -> {
                    if (documentSnapshots != null) {
                        for (DocumentSnapshot doc: documentSnapshots) {
                            Map<String, Object> data = doc.getData();
                            String item = "";
                            for(Map.Entry<String,     Object>     entry:
Objects.requireNonNull(data).entrySet()) {
                                item = item.concat(entry.getKey() + ": "
+ entry.getValue() + "\n");
                                if(item.contains("email: " + toString)){
                                    userRegister(toString,    toString1,
doc.getId());

                                }
                                }}
                        });}
    private void userRegister(String toString, String toString1, String
docId) {
        mAuth.createUserWithEmailAndPassword(toString, toString1)
                .addOnCompleteListener(task -> {
                    if(task.isSuccessful()){
                        FirebaseUser user = mAuth.getCurrentUser();
                        assert user != null;
                        String uid = user.getUid();
                        db.collection("teacher").document(docId)
                         .update("uid", uid);
                        Intent   i   =   new   Intent(LoginActivity.this,
TeacherPage.class);
                        startActivity(i);
                        finish();
                    }});
    }


    private boolean validation(String string, String regEx) {

        Pattern pattern = Pattern.compile(regEx);
        return pattern.matcher(string).matches();
    }

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


        mAuth = FirebaseAuth.getInstance();

        inputEmail = findViewById(R.id.emailId);
        inputPass = findViewById(R.id.password);
        forgotPass = findViewById(R.id.forgot_pwd);
        login = findViewById(R.id.login_btn);
        circularPB = findViewById(R.id.progressBar);

        forgotPass.setOnClickListener(v -> {
            if (validation(inputEmail.getText().toString(), emailRegEx))
```

```
mAuth.sendPasswordResetEmail(inputEmail.getText().toString());
            else
                Toast.makeText(this,    "Enter    a    valid    Email_id",
Toast.LENGTH_SHORT).show();
        });



        login.setOnClickListener(view -> {

            boolean                    emailValidated                    =
validation(inputEmail.getText().toString(), emailRegEx);
            boolean                  passwordValidated                   =
validation(inputPass.getText().toString(), passwordRegEx);

            if (emailValidated && passwordValidated) {
                circularPB.setVisibility(View.VISIBLE);

mAuth.signInWithEmailAndPassword(inputEmail.getText().toString(),
inputPass.getText().toString())
                        .addOnCompleteListener(this, task -> {
                            if (task.isSuccessful()) {
                                FirebaseUser           user           =
mAuth.getCurrentUser();
                                updateSignIn(user);
                            } else {

checkForUser(inputEmail.getText().toString(),
inputPass.getText().toString());
                                Toast.makeText(this,    "Login    failed",
Toast.LENGTH_SHORT).show();
                            }
                        });

            } else if (!emailValidated) Toast.makeText(this, "Invalid
Email Id", Toast.LENGTH_SHORT).show();
            else      Toast.makeText(this,      "Invalid      Password",
Toast.LENGTH_SHORT).show();
        });
    }}
```

UpdateDocument.java

```
public class UpdateDocument extends AppCompatActivity {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_update_document);
        Intent update = getIntent();
        String collection = update.getStringExtra("collection");
        String docId = update.getStringExtra("docId");
        LinearLayout updateDocList = findViewById(R.id.updateDocList);
        LinkedList<Map.Entry   <String,   EditText>>   editTexts   =   new
LinkedList<>();
```

```
db.collection(collection).document(docId).get().addOnCompleteListener(t
ask -> {
            if (task.isSuccessful()){
                DocumentSnapshot doc = task.getResult();
                Map<String, Object> data = doc.getData();
                if (data != null) {
                    for      (Map.Entry<String,      Object>      entry:
data.entrySet()) {
                        TextView textView = new TextView(this);
                        textView.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                        textView.setTextSize(20f);
                        textView.setText(entry.getKey());
                        updateDocList.addView(textView);

                        EditText editText = new EditText(this);
                        editText.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                        editText.setTextSize(20f);

editText.setInputType(InputType.TYPE_CLASS_TEXT);
                        editText.setText(entry.getValue().toString());
                        editTexts.add(new
AbstractMap.SimpleEntry<>(entry.getKey(), editText));
                        updateDocList.addView(editText);
                    }}
            }
            else {
                Log.d("GetDoc", "Get Failed with ", task.getException());
            }
        });

        Button updateBtn= findViewById(R.id.updateBtn);
        updateBtn.setOnClickListener(v -> {
            for (Map.Entry<String, EditText> entry : editTexts)

db.collection(collection).document(docId).update(entry.getKey(),
((EditText)entry.getValue()).getEditableText().toString());
            Toast.makeText(UpdateDocument.this, "Updated Successfully",
Toast.LENGTH_SHORT).show();
            finish();
        });
    }
}
```

ViewDocuments.java

```
public class ViewDocuments extends AppCompatActivity {

    RecyclerDocumentAdapter adapter;
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    ArrayList<Map.Entry<String, String>> list = new ArrayList<>();
    String collection;
    Button addBtn;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_view_documents);
        Intent intent = getIntent();
        collection = intent.getStringExtra("collection");
        RecyclerView recyclerView = findViewById(R.id.recyclerView);
        addBtn = findViewById(R.id.button);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        addBtn.setOnClickListener(view -> {
            Intent    i    =    new    Intent(getApplicationContext(),
AddCollection.class);
            i.putExtra("collection",collection);
            startActivity(i);
        });
        db.collection(collection)
                .addSnapshotListener((documentSnapshots, e) -> {
                    if (e != null)
                        Toast.makeText(getApplicationContext(),
e.getMessage(), Toast.LENGTH_SHORT).show();
                    list.clear();
                    if (documentSnapshots != null) {
                        for (DocumentSnapshot doc: documentSnapshots) {
                            Map<String, Object> data = doc.getData();
                            String item = "";
                            for(Map.Entry<String,     Object>     entry:
Objects.requireNonNull(data).entrySet())
                                item = item.concat(entry.getKey() + ": "
+ entry.getValue() + "\n");

                            list.add(new
AbstractMap.SimpleEntry<>(doc.getId(), item));
                        }
                    }
                    adapter = new RecyclerDocumentAdapter(this, list);
                    recyclerView.setAdapter(adapter);
                });

        ItemTouchHelper          itemTouchHelper          =          new
ItemTouchHelper(simpleCallback);
        itemTouchHelper.attachToRecyclerView(recyclerView);
    }

    ItemTouchHelper.SimpleCallback       simpleCallback       =       new
ItemTouchHelper.SimpleCallback(0,         ItemTouchHelper.LEFT         |
ItemTouchHelper.RIGHT) {
        @Override

        public   boolean   onMove(@NonNull   RecyclerView   recyclerView,
@NonNull       RecyclerView.ViewHolder       viewHolder,       @NonNull
RecyclerView.ViewHolder target) {
            return false;

        }
        @Override
        public     void     onSwiped(@NonNull     RecyclerView.ViewHolder
viewHolder, int direction) {
```

```
            switch (direction) {
                case ItemTouchHelper.LEFT:
                    new
AlertDialog.Builder(viewHolder.itemView.getContext())
                            .setMessage("Are you sure?")
                            .setPositiveButton("Yes", (dialog, which) ->
{
                                int                pos                =
viewHolder.getAdapterPosition();
                                Toast.makeText(ViewDocuments.this,

"Deleted:"+ pos, Toast.LENGTH_SHORT).show();

db.collection(collection).document(adapter.getAdapterId(pos)).delete().
addOnSuccessListener(aVoid                                              ->
Toast.makeText(getApplicationContext().getApplicationContext(),   "  Data
deleted successfully ", Toast.LENGTH_SHORT).show())

                                            .addOnFailureListener(e       ->
Toast.makeText(getApplicationContext().getApplicationContext(), " Error:
"+e.getMessage(), Toast.LENGTH_LONG).show());
                                adapter.notifyItemRemoved(pos);
                            })
                            .setNegativeButton("No", (dialog, which) ->
adapter.notifyItemChanged(viewHolder.getAdapterPosition()))
                            .create()
                            .show();
                    break;
                case ItemTouchHelper.RIGHT:
                    Intent update = new Intent(getApplicationContext(),
UpdateDocument.class);
                    update.putExtra("collection", collection);
                    update.putExtra("docId",
adapter.getAdapterId(viewHolder.getAdapterPosition()));
                    startActivity(update);
                    break;
            }
        }};
}
```

Chapter 4

IMPLEMENTATION

1. Description

- **MainActivity.java**

  It is used to display Splash Screen and to check if a user has logged in or not. If user is logged in, then dashboard page will be displayed after 1 second, Else Login page is displayed.

- **LoginActivity.java**

  It takes in login from user and checks if user is present in the database. If user is there, then they will be redirected to their dashboard page, else a Toast is displayed.

- **AddCollection.java**

  This is accessible only to admins to add Teachers. It takes in the input from user and store it in Firestore. Email is given as login credentials for teachers, using which a teacher can login. First login of teacher will set the password of that user too.

- **UpdateDocument.java**

  Admin can update teacher details. Teacher's ID is taken and matched to update data in firebase.

- **ViewDocuments.java**

  Recycler view is used to display real time data from database. An adapter is linked to this file to display data. Data fetching from firestore is done here.

Chapter 5

RESULTS

Loading Page

Login Page

The app opens with a splash screen followed by a Login page, where Admin/Teacher can login. If wrong credentials are given, a Toast message is displayed saying “Invalid Password”.

If the user gives correct credentials and their account is registered, then they are taken to their respective Dashboard pages (Admin/Teacher dashboard).

Admin Page

Admin Dashboard has a grid view which displays different features. User can click on it if they want to CRUD any data.

Update Page

Sign Up page

Admin can update Teacher's details in the Update Teacher form. Admin can add new Teacher and their login credentials in Add user page.

Teacher List Page

User can swipe left to Update and swipe right to Delete content.

Teacher's Dashboard **Subject List Page**

When a teacher login, they are taken to Teacher's Dashboard. They can update marks, attendance of students. Subject list shows subject name and the subject code.

Chapter 6

CONCLUSION, FUTURE ENHANCEMENT AND REFERENCES

Conclusion

Android as a full, open and free mobile device platform, with its powerful function and good user experience rapidly developed into the most popular mobile operating system.

Firebase Authentication provides a simple and accessible way to login. We will build an app that will take in login credentials and based on the type of user (student, teacher, admin) it will give dashboard with all the functionalities available for that user. It is simple and clear API. It is free API which is provided and the weather is updated in every four hours. Firestore Database provided by Firebase allows a quick and reliable way for storing and accessing data.

This application will allows us to access and control the student database on cloud. We can perform basic CRUD operations (create, retrieve, update, delete) on each document in each collection stored in the Firestore Database.

Future Enhancement

1. To add the feature for teachers to share study materials with students.
2. To add in-app option to read college circulars and updates.
3. To add more interactive features like feedback system, assignments and quizzes.

References

- https://www.geeksforgeeks.org/
- https://www.javatpoint.com/
- https://www.tutorialspoint.com/
- https://stackoverflow.com/