ACKNOWLEDGEMENT

We take this opportunity to express our sincere gratitude and respect to the Mount Carmel College Autonomous, Bengaluru for providing us an opportunity to carry out our project. We express our gratitude to Dr. Sr. Arpana, Principal and Dr. Charmaine Jerome**,** Vice Principal for providing the resources and support without which the completion of this project would have been a difficult task.

We extend our thanks to Dr. Regina L Suganthi, Dean of Computer Science and Ms.Renju K, Head of Department of Computer Science, for their encouragement and support throughout the Project.

With a profound sense of gratitude, we acknowledge the guidance and support extended by Ms.Jinu Sara Rajan, Project Guide. We thank her for constant supervision and valuable technical support which is been of immense help in developing this project.

We also extend our thanks to the entire faculty and Lab Assistant of Department of Computer Science who have encouraged us throughout the course.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 PREFACE

Extra curriculum plays a key tool in the personal development of a student. It not only provides entertainment but also helps in networking and improving skills. These activities offer opportunities for students to learn the value of teamwork, individual strength and group responsibility, competition, diversity, and a sense of communication.

Today a growing number of students rely on various social media platforms as their primary source of information to know about the college's activities. Students from different states were not able to know about the events happening around them hence this website plays a good in role overcoming it. The purpose of our application is to communicate information about the college events to the students of various combinations. Being an online platform, the event grasps a lot of students hence both the hosting colleges and students are benefitted.

The "MCC CONNECT" application, curates all the information on activities directly by the college authorities, and delivers it to all the students. This is an online platform where the various events are constantly being updated by the authentic source so that it reaches the broader spectrum of students. This application helps students by keeping them completely aware of the events happening around them, providing messaging feature, campus guidance, etc. that is very helpful and offers a great advantage to both student and the administration.

1.2 ABOUT THE PROJECT

The main objective of the application is to provide a platform for the administration to update notification of any sort and avoid circulation of spam information and facilitate students to connect with students/faculty/alumni of that institute by sharing, viewing and communicating about latest updates happening in institute. The application where the viewers will get campus guidance at one click. This application provides a user friendly interface that can be accessed by any student without hindrance and the main part is composed of the different tabs which segregates all the events so as to avoid chaos and confusion, search

bar that ensure quick accessibility, reducing the search time effectively, chat section for communication/interaction with other users while using our application, login portal that enhance the security, so that information is only from authentic source and no random people has access to it and an option to subscribe so that subscribers will get regular updates through email.

On the front end we have used XML, Java and Firebase on the back end of our project for reliability and most importantly, to ensure the accuracy of the information.

1.3 PROBLEM OVERVIEW

Colleges conduct various events, seminars, webinars and other competitions they face issue in reaching out the information to students due to several non- integrated social media platforms. It is difficult for College authorities to keep track whether all students have got the events information without spamming. In the other hand student may not have direct communication with students of other courses in college or university, which will be an issue and chance of missing out on connections making students disappointed.

The MCC CONNECT website helps to build a platform to overcome communication gap faced by students and college authorities and help them to make their job easy.

1.4 OBJECTIVES

1. The main intension of “MCC CONNECT” is to develop a mobile application for the upcoming activities in college through posts and to help students to connect with other students/faculty/alumni.
2. The project aims to develop an online platform that will allow administration to update their students/faculty in authenticated manner.
3. To provide a chat section as a user-friendly interface to for active connection.
4. To provide a facility of Campus Navigation guidance.

CHAPTER 2

PROBLEM ANALYSIS

2.1 PURPOSE

The purpose of developing MCC CONNECT mobile app is to provide an integrated platform for students, faculties and alumni of their college where they can connect to each other and known updates from authenticated source.

2.2 METHODOLOGY

Mobile application development is a process of development of compatible and suitable applications for mobile devices. With increasing internet consumption as well as awareness about technology mobile application development has become a well-known and well-needed domain of technological enhancement.

The development of a good mobile application with good user interface design is really important because it considers all the constraints like screen, input, context, and other outlines. The front end development tools for an integrated User experience are UI design tools, cross-platform accommodation/ support, and SDKs to access the features of devices. The development of progressive apps is really important for compatibility in different devices and software programs like Google Android and Apple iOS. Different methodologies are adopted for the development of mobile applications based on business requirements, software requirements, prototyping, application architecture, design and development integration testing, alpha and beta testing, and market analysis.

Agile methodology for mobile application development is an interactive approach that leads to the development of an application with the complete cycle process divided into multiple sub-modules or small parts known as mini-projects. Each mini project or some module is assigned to two different members of a team of Specialists that collaborate and work together to complete the development cycle starting from designing to development and testing after which the project is delivered under expert supervision and constant improvisation.

Some benefits of the agile mobile application development approach are faster development, reduction of risks, improved quality, uninterrupted project management, customization etc.

Each iteration is considered as a short time "frame" in the agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client. Agile assumes that the end users' needs are ever changing in a dynamic business and IT world. People and interactions are emphasized rather than process and tools in Agile Model.

For the app, the necessary features required were collected initially. Later taking the recommendations, the requirements were bought down to the services for specific needs. The information needed from users was finalized. When the requirements are defined, each module was taken by the team for the designing and developing phase which underwent various stages of improvement. The quality of the application's performance and the bugs were examined and rectified. As the last stage the app was issued for user's work environment. After releasing the app, the team receives feedback about the product and works through the feedback.

2.3 Data Flow Diagram

Data flow is used to graphically represent flow of data in the application system. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Data Flow Diagram symbols are standardized notations, like rectangles, circles, arrows, and short-text labels, that describe a system or process' data flow direction, data inputs, data outputs, data storage points, and it's various sub-processes.

2.3.1 Zero Level DFD

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

MCC CONNECT application consists of College Admin and student, faculty or alumni as a user who requests for various processes and admin from the server of MCC CONNECT gives the data information.

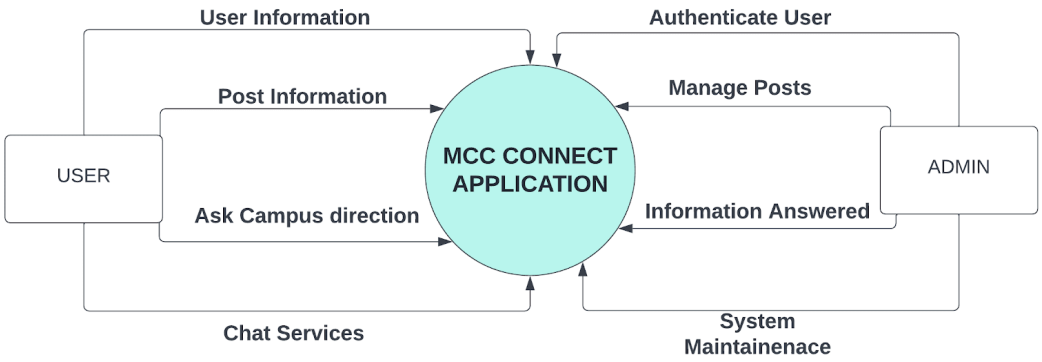


Figure 2.1 Zero level DFD

2.3.2 First Level DFD

First Level DFD denotes each of the main sub process that together forms the complete system. In this level, we highlight the main functions of the system.

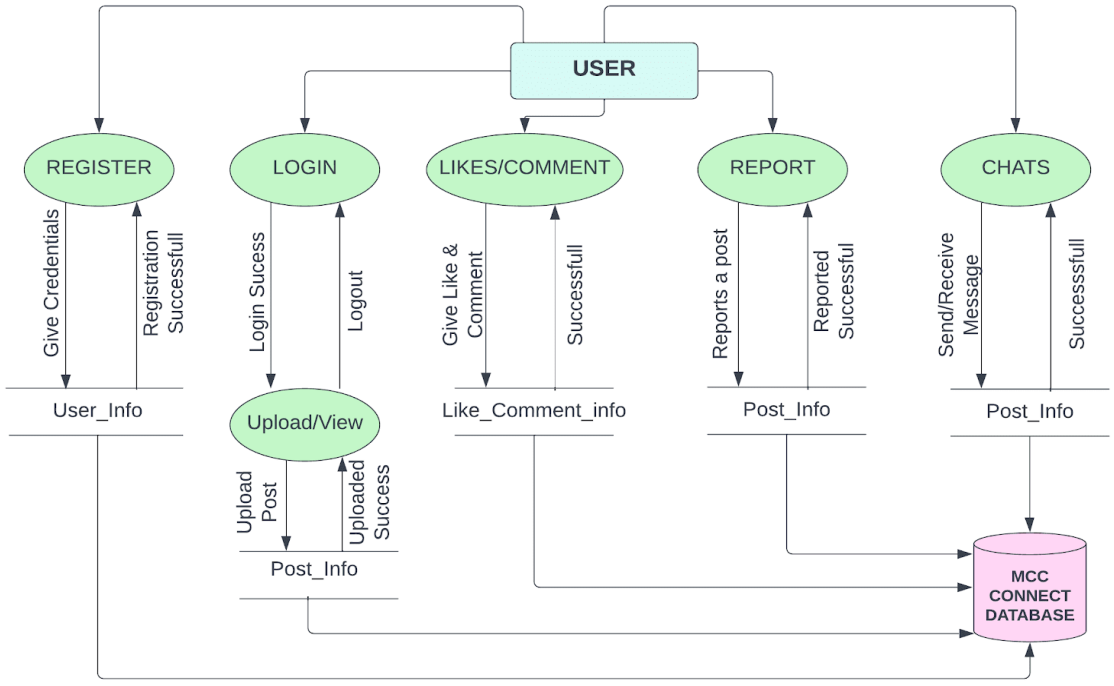


Figure 2.2 First level DFD

CHAPTER 3

PROJECT DESIGN

3.1 MODULES

I. PROFILE MANAGEMENT

By providing the valid college email ID, name and setting a password user will be registered successfully and will be redirected to the login page. In the login page users can login with the registered email and password and based on the input user is redirected to admin or dashboard page. Users can find a profile page where he/she can update their password, change username and profile picture.

II. POST MANAGEMENT

In the add post page users can add posts by giving title, post picture and post description. This post can be viewed by all other users. There is an option given to like a post and comment on a particular post. The post can be deleted by the user who uploaded it.

III. SEARCH

In this search module users can search for a particular post and search a particular user out of all registered users.

IV. COMMUNICATION

Users can open a chat page where he/she can start a conversation with each other personally. They can send a text message and also media. Delete message option is also available.

V. CAMPUS GUIDANCE

In Campus Guidance module users can know the information or guidance for the departments and lab room in the university. Information such as block name, floor name and directions will be specified.

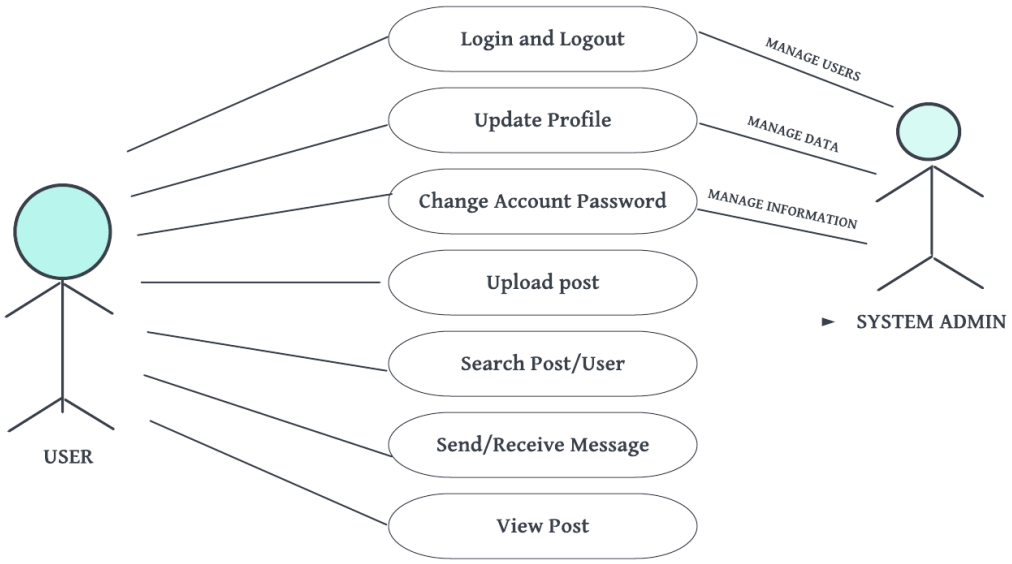
VI. ADMIN

In Admin module admin log in into the dashboard. View data analytics and registered user information. Admin has access to broad cast message to all the users at same time. Admin has access to feedback received from users under report section.

3.2 USE CASE DIAGRAM

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.



3.1 Use Case Diagram

3.3 DATA INTEGRITY & CONSTRAINTS

Data Integrity is having correct and accurate data in your database. When we are storing data in the database we don't want repeating values, incorrect values or broken relationships between tables.

Data Integrity can be maintained using constraints. These constraints define the rules according to which the operations like updating, deletion, insertions etc. have to be performed to maintain the data integrity. There are mainly four types of Data Integrity:

1. Domain Integrity

   Domain refers to the range of acceptable values. It refers to the range of values that we are going to accept and store in a particular column within a database.

2. Entity Integrity

   Each row for an entity in a table should be uniquely identified i.e. if some record is saved in the database then that record should be uniquely identified from others. This is done with the help of primary keys.

3. Referential Integrity

   Referential Integrity is used to maintain the data consistency between two tables. Rules are made in the database structure about how foreign keys should be used to ensure that changes, addition and deletion in the database maintain the data integrity.

4. User-Defined Integrity

   Sometimes domain, referential and entity integrity are not enough to maintain the data integrity. Such integrity is typically implemented through triggers and stored procedures. Triggers are a block of statements which executes automatically if any predefined events occur.

3.4 DATA DICTIONARY

Admin Message		
Field Name	**Type**	**Example**
Message	string	ESE Exams on 11th June.
Users		
Field Name	**Type**	**Example**
Email Id	string	Sumipandi104@gmail.com
Password	string	Sumithra@4.
Name	string	Sumithra
Profile picture	string	https://firebasestorage./myapp-Users_Profile_Cover_image
Designation	string	Student
Uid	string	65wZ2aDdfiduwGQ12JGNy9dxs7j2
Post		
Field Name	**Type**	**Example**
Title	string	Artificial intelligence
Description	string	Genomics, gene editing, and synthetic biology trend of 2022
Picture	string	https://firebasestorage./myapp-Users_Profile_Cover_image
Uid	string	65wZ2aDdfiduwGQ12JGNy9dxs7j2
pcomment	string	Fabulous
plike	numeric	5

Chat		
Field Name	**Type**	**Example**
Message	string	Hello
Receiver	string	85wZ2aDdfiduwGQ12JGNy9dxs7j2344
Sender	string	65wZ2aDdfiduwGQ12JGNy9dxs7j2

3.5 SYSTEM REQUIREMENTS

To know that we can start our Android application development of operating system on Microsoft Windows 10/8/7/Vista/2003 (32 or 64-bit) the required tools to develop Android applications are open source and can be downloaded from the Web. Following is the list of software's we need before we start of Android application programming.

- Java JDK5 or later version
- Java Runtime Environment (JRE) 6
- Android Studio

1. SOFTWARE TOOLS :

Frontend tools

Android Studio:

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on Jet Brains' IntelliJ IDEA software and designed specifically for Android development. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development. Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. On May 7,

2019, Kotlin replaced Java as Google's preferred language for Android app development. Java is still supported, as is C++

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGear integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Java:

Java is default language to code whereas C and C++ are in options. It supports native code also, so C and C++ are applicable to code for the mobile application. An SDK (Software Development Kit) is used for Android development which has a bunch of libraries instead of JVM.

XML:

XML stands for Extensible Mark-up Language. XML is a very popular format and commonly used for sharing data on the internet. This chapter explains how to parse the XML file and extract necessary information from it. Android provides three types of XML parsers which are DOM, SAX and XMLPullParser.

Backend tools

Firebase:

Google Firebase is Google-backed application development software that enables developers to develop iOS, Android and Web apps. Firebase provides tools for tracking analytics, reporting and fixing app crashes, creating marketing and product experiment. It is a Backend-as-a-Service (Baas).It provides developers with a variety of tools and services to help to develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

2. HARDWARE TOOLS :

1. Microsoft Windows 10 (64-bit)
2. 8 GB RAM recommended (plus 1 GB for the Android Emulator)
3. 2 GB of available disk space minimum, 4 GB recommended (500 MB for IDE plus 1.5 GB for Android SDK and emulator system image)
4. 1280 x 800 minimum screen resolution.
5. Hard disk: 80GB (or) Higher.

CHAPTER 4

CODING

4.1 Manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
/>
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
/>
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".ReportShow"
            android:exported="false" />
        <activity
            android:name=".AdminChatActivity"
            android:exported="false" />
        <activity
            android:name=".NavigationActivity"
            android:exported="false" />
        <activity
```

```
            android:name=".reportSplashActivity"
            android:exported="false" />
        <activity
            android:name=".ReportPostActivity"
            android:exported="false" />
        <activity android:name=".MainActivity" />
        <activity android:name=".ChatActivity" />
        <activity android:name=".PostLikedByActivity" />
        <activity android:name=".PostDetailsActivity" />
        <activity android:name=".EditProfilePage" />
        <activity android:name=".LoginActivity" />
        <activity android:name=".RegistrationActivity" />
        <activity android:name=".DashboardActivity" />
        <activity
            android:name=".SpalshScreen"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data
            android:name="preloaded_fonts"
            android:resource="@array/preloaded_fonts" />
    </application>
</manifest>
```

4.2 Splash Screen Java file

```
package com.example.myapp;
```

```
import androidx.appcompat.app.AppCompatActivity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.*;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
public class SpalshScreen extends AppCompatActivity {
    ImageView imageview;
    FirebaseUser currentUser;
    private FirebaseAuth mAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_spalsh_screen);
        imageview=findViewById(R.id.imageview);
        Animation animation =
AnimationUtils.loadAnimation(getApplicationContext(),R.anim.zoomanim);
        imageview.startAnimation(animation);
        mAuth=FirebaseAuth.getInstance();
        if (mAuth !=null) {
            currentUser = mAuth.getCurrentUser();
        }
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                FirebaseUser user=mAuth.getCurrentUser();
                if(user==null){
                    Intent intent = new Intent(SpalshScreen.this, LoginActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
```

```
                startActivity(intent);
                finish();
            }
            else {
                Intent mainIntent= new Intent(SpalshScreen.this, DashboardActivity.class);
                mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(mainIntent);
                finish();
            }
        }
    },3000);
  }
}
```

4.3 Registration Java file

```
public class RegistrationActivity extends AppCompatActivity {
   private EditText email,password,name;
   private Button mRegister;
   private TextView existaccount;
   private ProgressDialog progressDialog;
   private FirebaseAuth mAuth;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_registration);
      ActionBar actionBar=getSupportActionBar();
      actionBar.setTitle("Create Account");
      actionBar.setDisplayShowHomeEnabled(true);
      actionBar.setDisplayHomeAsUpEnabled(true);
      email=findViewById(R.id.register_email);
```

```
name=findViewById(R.id.register_name);
password=findViewById(R.id.register_password);
mRegister=findViewById(R.id.register_button);
existaccount=findViewById(R.id.homepage);
mAuth=FirebaseAuth.getInstance();
progressDialog=new ProgressDialog(this);
progressDialog.setMessage("Register");
mRegister.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      String emaill=email.getText().toString().trim();
      String uname=name.getText().toString().trim();
      String pass=password.getText().toString().trim();
      String regex = "[A-Za-z\\s]{1,}[\\.]{0,1}[A-Za-z\\s]{0,}$";
      if(!Patterns.EMAIL_ADDRESS.matcher(emaill).matches()){
         email.setError("Invalid Email");
         email.setFocusable(true); }
         else if (pass.length()<6){
         password.setError("Length Must be greater than 6 character");
         password.setFocusable(true);
      }
      else if(name.getText().toString().isEmpty()){
         name.setError("No Name");
         name.setFocusable(true);
      else {
         registerUser(emaill,pass,uname);
      } } });
existaccount.setOnClickListener(new View.OnClickListener() {
   @Override
```

```
        public void onClick(View v) {
            startActivity(new Intent(RegistrationActivity.this, LoginActivity.class));
        }
    });
  }
private void registerUser( String emaill, final String pass,final String uname) {
    progressDialog.show();
    mAuth.createUserWithEmailAndPassword(emaill, pass).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()){
                progressDialog.dismiss();
                FirebaseUser user=mAuth.getCurrentUser();
                String email=user.getEmail();
                String uid=user.getUid();
                HashMap<Object,String> hashMap=new HashMap<>();
                hashMap.put("email",email);
                hashMap.put("uid",uid);
                hashMap.put("name",uname);
                hashMap.put("onlineStatus","online");
                hashMap.put("typingTo","noOne");
                hashMap.put("image","");
                hashMap.put("designation","");
                FirebaseDatabase database= FirebaseDatabase.getInstance();
                DatabaseReference reference=database.getReference("Users");
                reference.child(uid).setValue(hashMap);
                Toast.makeText(RegistrationActivity.this,"Registered User "
+user.getEmail(),Toast.LENGTH_LONG).show();
                Intent mainIntent=new Intent(RegistrationActivity.this,
DashboardActivity.class);
```

```
                mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
                startActivity(mainIntent);
                finish();
            }
            else {
                progressDialog.dismiss();
                Toast.makeText(RegistrationActivity.this,"Error",Toast.LENGTH_LONG).show
();
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            progressDialog.dismiss();
            Toast.makeText(RegistrationActivity.this,"Error
Occured",Toast.LENGTH_LONG).show();
        }
    });
  }
  @Override
  public boolean onSupportNavigateUp() {
    onBackPressed();
    return super.onSupportNavigateUp();
  }
}
```

4.4 Sign in Java file

```
public class LoginActivity extends AppCompatActivity {
  private EditText email,password,name;
  private Button mlogin;
```

```
    private TextView newdnewaccount,reocverpass;
    FirebaseUser currentUser;
    private ProgressDialog loadingBar;
    private FirebaseAuth mAuth;
    String uname="connect@gmail.com";
    String pwd="04142001";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        ActionBar actionBar=getSupportActionBar();
        actionBar.setTitle("Create Account");
        actionBar.setDisplayShowHomeEnabled(true);
        actionBar.setDisplayHomeAsUpEnabled(true);
        email=findViewById(R.id.login_email);
        password=findViewById(R.id.login_password);
        newdnewaccount=findViewById(R.id.needs_new_account);
        reocverpass=findViewById(R.id.forgetp);
        mAuth=FirebaseAuth.getInstance();
        mlogin=findViewById(R.id.login_button);
        loadingBar=new ProgressDialog(this);
        mAuth=FirebaseAuth.getInstance();
        if (mAuth !=null) {
            currentUser = mAuth.getCurrentUser();
        }
gin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String emaill=email.getText().toString().trim();
                String pass=password.getText().toString().trim();

                if(!Patterns.EMAIL_ADDRESS.matcher(emaill).matches()){
                    email.setError("Invalid Email");
                    email.setFocusable(true);
```

```
            }
              else if (email.getText().toString().equals(uname) &&
password.getText().toString().equals(pwd)) {
                 Toast.makeText(getApplicationContext(), "Welcome " +
email.getText().toString(), Toast.LENGTH_LONG).show();
                 startActivity(new Intent(LoginActivity.this, MainActivity.class));
              }
              else {
                 loginUser(emaill,pass);
              }
           }
        });
        newdnewaccount.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
              startActivity(new Intent(LoginActivity.this, RegistrationActivity.class));
           }
        });
        reocverpass.setOnClickListener(new View.OnClickListener() {
           @Override
           public void onClick(View v) {
              showRecoverPasswordDialog();
           }
        });

     }
     private void showRecoverPasswordDialog() {
        AlertDialog.Builder builder=new AlertDialog.Builder(this);
        builder.setTitle("Recover Password");
        LinearLayout linearLayout=new LinearLayout(this);
        final EditText emailet= new EditText(this);
        emailet.setText("Email");
        emailet.setMinEms(16);
        emailet.setInputType(InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS);
```

```
        linearLayout.addView(emailet);
        linearLayout.setPadding(10,10,10,10);
        builder.setView(linearLayout);
        builder.setPositiveButton("Recover", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                String emaill=emailet.getText().toString().trim();
                beginRecovery(emaill);
            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
        builder.create().show();
    }
    private void beginRecovery(String emaill) {
        loadingBar.setMessage("Sending Email....");
        loadingBar.setCanceledOnTouchOutside(false);
        loadingBar.show();
        mAuth.sendPasswordResetEmail(emaill).addOnCompleteListener(new
OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                loadingBar.dismiss();
                if(task.isSuccessful())
                {
                    Toast.makeText(LoginActivity.this,"Done
sent",Toast.LENGTH_LONG).show();
                }
                else {
```

```
                Toast.makeText(LoginActivity.this,"Error
Occured",Toast.LENGTH_LONG).show();
            }
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            loadingBar.dismiss();
            Toast.makeText(LoginActivity.this,"Error
Failed",Toast.LENGTH_LONG).show();
        }
    });
  }
  private void loginUser(String emaill, String pass) {
    loadingBar.setMessage("Logging In....");
    loadingBar.show();
    mAuth.signInWithEmailAndPassword(emaill, pass).addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                loadingBar.dismiss();
                FirebaseUser user = mAuth.getCurrentUser();
                if (task.getResult().getAdditionalUserInfo().isNewUser()) {
                    String email = user.getEmail();
                    String uid = user.getUid();
                    HashMap<Object, String> hashMap = new HashMap<>();
                    hashMap.put("email", email);
                    hashMap.put("uid", uid);
                    hashMap.put("name", "");
                    hashMap.put("onlineStatus", "online");
                    hashMap.put("typingTo","noOne");
                    hashMap.put("phone", "");
                    hashMap.put("image", "");
```

```
                hashMap.put("cover", "");
                FirebaseDatabase database = FirebaseDatabase.getInstance();
                DatabaseReference reference = database.getReference("Users");
                reference.child(uid).setValue(hashMap);
            }
            Toast.makeText(LoginActivity.this, "Registered User " + user.getEmail(),
Toast.LENGTH_LONG).show();
            Intent mainIntent = new Intent(LoginActivity.this, DashboardActivity.class);
            mainIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
Intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(mainIntent);
            finish();
          }
          else {
            loadingBar.dismiss();
            Toast.makeText(LoginActivity.this,"Login
Failed",Toast.LENGTH_LONG).show();
          }
        }
      }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
          loadingBar.dismiss();
          Toast.makeText(LoginActivity.this,"Error
Occured",Toast.LENGTH_LONG).show();
        }
      });
  }
  @Override
  public boolean onSupportNavigateUp() {
    onBackPressed();
    return super.onSupportNavigateUp();
  }
}
```

4.5 Profile Java file

```
public class ProfileFragment extends Fragment {
    private FirebaseAuth firebaseAuth;
    FirebaseUser firebaseUser;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference databaseReference;
    ImageView avatartv,covertv;
    TextView nam,email,phone,design;
    RecyclerView postrecycle;
    StorageReference storageReference;
    String storagepath="Users_Profile_Cover_image/";
    FloatingActionButton fab;
    List<ModelPost> posts;
    AdapterPosts adapterPosts;
    String uid;
    ProgressDialog pd;
    private static final int CAMERA_REQUEST=100;
    private static final int STORAGE_REQUEST=200;
    private static final int IMAGEPICK_GALLERY_REQUEST=300;
    private static final int IMAGE_PICKCAMERA_REQUEST=400;
    String cameraPermission[];
    String storagePermission[];
    Uri imageuri;
    public ProfileFragment() {
        // Required empty public constructor
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
```

```
// Inflate the layout for this fragment
View view= inflater.inflate(R.layout.fragment_profile, container, false);
firebaseAuth=FirebaseAuth.getInstance();
firebaseUser=firebaseAuth.getCurrentUser();
firebaseDatabase=FirebaseDatabase.getInstance();
databaseReference=firebaseDatabase.getReference("Users");
avatartv=view.findViewById(R.id.avatartv);
nam=view.findViewById(R.id.nametv);
email=view.findViewById(R.id.emailtv);
uid=FirebaseAuth.getInstance().getUid();
design=view.findViewById(R.id.designtv);
fab=view.findViewById(R.id.fab);
postrecycle=view.findViewById(R.id.recyclerposts);
posts=new ArrayList<>();
pd=new ProgressDialog(getActivity());
loadMyPosts();
pd.setCanceledOnTouchOutside(false);
Query
query=databaseReference.orderByChild("email").equalTo(firebaseUser.getEmail());
query.addValueEventListener(new ValueEventListener() {
   @Override
   public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
      for (DataSnapshot dataSnapshot1:dataSnapshot.getChildren()){
         String name=""+dataSnapshot1.child("name").getValue();
         String emaill=""+dataSnapshot1.child("email").getValue();
         String image=""+dataSnapshot1.child("image").getValue();
         String designn=""+dataSnapshot1.child("designation").getValue();
         nam.setText(name);
         email.setText(emaill);
         design.setText(designn);
```

```
                try {
                    Glide.with(getActivity()).load(image).into(avatartv);
                }catch (Exception e){
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {
            }
        });
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(getActivity(),EditProfilePage.class));
            }
        });
        return view;
    }
    private void loadMyPosts() {
        LinearLayoutManager layoutManager=new LinearLayoutManager(getActivity());
        layoutManager.setReverseLayout(true);
        layoutManager.setStackFromEnd(true);
        postrecycle.setLayoutManager(layoutManager);
        DatabaseReference
databaseReference=FirebaseDatabase.getInstance().getReference("Posts");
        Query query=databaseReference.orderByChild("uid").equalTo(uid);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                posts.clear();
                for (DataSnapshot dataSnapshot1:dataSnapshot.getChildren()){
                    ModelPost modelPost=dataSnapshot1.getValue(ModelPost.class);
                    posts.add(modelPost);
```

```
                adapterPosts=new AdapterPosts(getActivity(),posts);
                postrecycle.setAdapter(adapterPosts);
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            Toast.makeText(getActivity(),databaseError.getMessage(),Toast.LENGTH_LONG
).show();
        }
    });
  }
  @Override
  public void onCreate(@Nullable Bundle savedInstanceState) {
    setHasOptionsMenu(true);
    super.onCreate(savedInstanceState);
  }
  @Override
  public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu,inflater);
  }
  @Override
  public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    if(item.getItemId()==R.id.logout){
        firebaseAuth.signOut();
        startActivity(new Intent(getContext(), SpalshScreen.class));
        getActivity().finish();
    }
    else if(item.getItemId()==R.id.report){
        Intent in = new Intent(getContext(), ReportPostActivity.class);
        startActivity(in);
```

```
        //Toast.makeText(getActivity(),"Reported",Toast.LENGTH_LONG).show();
      }
      return super.onOptionsItemSelected(item);
   }
}
```

4.6 ChatPage Java file

```
public class ChatActivity extends AppCompatActivity {
    Toolbar toolbar;
    RecyclerView recyclerView;
    ImageView profile,block;
    TextView name,userstatus;
    EditText msg;
    ImageButton send,attach;
    FirebaseAuth firebaseAuth;
    String uid,myuid,image;
    ValueEventListener valueEventListener;
    List<ModelChat> chatList;
    AdapterChat adapterChat;
    private static final int IMAGEPICK_GALLERY_REQUEST = 300;
    private static final int IMAGE_PICKCAMERA_REQUEST = 400;
    private static final int CAMERA_REQUEST = 100;
    private static final int STORAGE_REQUEST = 200;
    String cameraPermission[];
    String storagePermission[];
    Uri imageuri = null;
    FirebaseDatabase firebaseDatabase;
    DatabaseReference users;
    boolean notify=false;
    boolean isBlocked=false;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        LinearLayoutManager linearLayoutManager=new LinearLayoutManager(this);
```

```
        linearLayoutManager.setStackFromEnd(true);
        recyclerView=findViewById(R.id.chatrecycle);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(linearLayoutManager);
        uid=getIntent().getStringExtra("uid");
        firebaseDatabase= FirebaseDatabase.getInstance();
        cameraPermission = new String[]{Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE};
        storagePermission = new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}
checkUserStatus();
        users=firebaseDatabase.getReference("Users");
attach.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showImagePicDialog();
            }
        });
        send.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                notify=true;
                String message=msg.getText().toString().trim();
                if (TextUtils.isEmpty(message)){
                    Toast.makeText(ChatActivity.this,"Please Write Something
Here",Toast.LENGTH_LONG).show();}
        Query userquery=users.orderByChild("uid").equalTo(uid);
        userquery.addValueEventListener(new ValueEventListener() {

    private void readMessages() {

        chatList=new ArrayList<>();
        DatabaseReference dbref=
FirebaseDatabase.getInstance().getReference().child("Chats");
```

```
    dbref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            chatList.clear();
            for (DataSnapshot dataSnapshot1:dataSnapshot.getChildren()){
                ModelChat modelChat=dataSnapshot1.getValue(ModelChat.class);
                if(modelChat.getSender().equals(myuid)&&
                        modelChat.getReceiver().equals(uid)||
                        modelChat.getReceiver().equals(myuid)
                                && modelChat.getSender().equals(uid)){
                    chatList.add(modelChat);
                }
                adapterChat=new AdapterChat(ChatActivity.this,chatList,image);
                adapterChat.notifyDataSetChanged();
                recyclerView.setAdapter(adapterChat);
        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
private void showImagePicDialog() {
    String options[]={ "Camera","Gallery"};
    AlertDialog.Builder builder=new AlertDialog.Builder(ChatActivity.this);
    builder.setTitle("Pick Image From");
    builder.setItems(options, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {

            if(which==0){
                if(!checkCameraPermission()){
                    requestCameraPermission();
                }
                else {
                    pickFromCamera();
                }
            }else if(which==1){
```

```
                if(!checkStoragePermission()){
                    requestStoragePermission();
                }
                else {
                    pickFromGallery();
}}});       builder.create().show();
    }
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {

        switch (requestCode){
            case CAMERA_REQUEST:{
                if(grantResults.length>0){
                    boolean
camera_accepted=grantResults[0]==PackageManager.PERMISSION_GRANTED;
                    boolean
writeStorageaccepted=grantResults[1]==PackageManager.PERMISSION_GRANTE
D;
                    if(camera_accepted&&writeStorageaccepted){
                        pickFromCamera();
                    }
                    else {
                        Toast.makeText(this,"Please Enable Camera and Storage
Permissions",Toast.LENGTH_LONG).show();
                    }
                }
            }
            break;
            case STORAGE_REQUEST:{
                if(grantResults.length>0){
                    boolean
writeStorageaccepted=grantResults[0]==PackageManager.PERMISSION_GRANTE
D;
                    if(writeStorageaccepted){
```

```
                pickFromGallery();
            }
            else {
                Toast.makeText(this,"Please Enable Storage
Permissions",Toast.LENGTH_LONG).show();}}}          break;
    }   }
  @Override
  public void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
    if(resultCode == RESULT_OK){
      if(requestCode==IMAGEPICK_GALLERY_REQUEST){
        imageuri=data.getData();
        try {
          sendImageMessage(imageuri);
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
      if(requestCode==IMAGE_PICKCAMERA_REQUEST){
        try {
          sendImageMessage(imageuri);
        } catch (IOException e) {
          e.printStackTrace();
        }
      }
    }
    super.onActivityResult(requestCode, resultCode, data);
  }

  private void sendImageMessage(Uri imageuri) throws IOException {
    notify=true;
    final ProgressDialog dialog=new ProgressDialog(this);
    dialog.setMessage("Sending Image");
    dialog.show();
```

```
final String timestamp=""+System.currentTimeMillis();
String filepathandname="ChatImages/"+"post"+timestamp;
Bitmap
bitmap=MediaStore.Images.Media.getBitmap(this.getContentResolver(),imageuri);
ByteArrayOutputStream arrayOutputStream=new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.PNG,100,arrayOutputStream);
final byte[] data=arrayOutputStream.toByteArray();
StorageReference ref=
FirebaseStorage.getInstance().getReference().child(filepathandname);
ref.putBytes(data).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
    @Override
    public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
      dialog.dismiss();
      Task<Uri> uriTask = taskSnapshot.getStorage().getDownloadUrl();
      while (!uriTask.isSuccessful()) ;
      String downloadUri = uriTask.getResult().toString();
      if(uriTask.isSuccessful()){
        DatabaseReference re= FirebaseDatabase.getInstance().getReference();
        HashMap<String,Object> hashMap=new HashMap<>();
        hashMap.put("sender",myuid);
        hashMap.put("receiver",uid);
        hashMap.put("message",downloadUri);
        hashMap.put("timestamp",timestamp);
        hashMap.put("dilihat",false);
        hashMap.put("type","images");
        re.child("Chats").push().setValue(hashMap);
        final DatabaseReference ref1=
FirebaseDatabase.getInstance().getReference("ChatList").child(uid).child(myuid);
        ref1.addValueEventListener(new ValueEventListener() {
          @Override
          public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(!dataSnapshot.exists()){
```

```
                    ref1.child("id").setValue(myuid);
                }
            }
            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
        final DatabaseReference ref2=
FirebaseDatabase.getInstance().getReference("ChatList").child(myuid).child(uid);
        ref2.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                if(!dataSnapshot.exists()){
                    ref2.child("id").setValue(uid);
                }
            }
```

4.7 PostDetails Java file

```
private void loadPostInfo() {

DatabaseReference databaseReference =
FirebaseDatabase.getInstance().getReference("Posts");

        Query query=databaseReference.orderByChild("ptime").equalTo(postId);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for(DataSnapshot dataSnapshot1:dataSnapshot.getChildren()){
                    String ptitle=dataSnapshot1.child("title").getValue().toString();
                    String
descriptions=dataSnapshot1.child("description").getValue().toString();
                    uimage=dataSnapshot1.child("uimage").getValue().toString();
```

```
                hisdp=dataSnapshot1.child("udp").getValue().toString();
                hisuid=dataSnapshot1.child("uid").getValue().toString();
                String uemail=dataSnapshot1.child("uemail").getValue().toString();
                hisname=dataSnapshot1.child("uname").getValue().toString();
                ptime=dataSnapshot1.child("ptime").getValue().toString();
                plike=dataSnapshot1.child("plike").getValue().toString();
                String
commentcount=dataSnapshot1.child("pcomments").getValue().toString();
                Calendar calendar=Calendar.getInstance(Locale.ENGLISH);
                calendar.setTimeInMillis(Long.parseLong(ptime));
                String timedate= DateFormat.format("dd/MM/yyyy hh:mm
aa",calendar).toString();
                name.setText(hisname);
                title.setText(ptitle);
                description.setText(descriptions);
                like.setText(plike +" Likes");
                time.setText(timedate);
                tcomment.setText(commentcount + " Comments");
                if(uimage.equals("noImage")){
                  image.setVisibility(View.GONE);
                }
                else {
                   image.setVisibility(View.VISIBLE);
                   try {
                      Glide.with(PostDetailsActivity.this).load(uimage).into(image);
                   }
                try {
                   Glide.with(PostDetailsActivity.this).load(hisdp).into(picture);
                }
                catch (Exception(e){
}}}
```

4.8 Admin Java file

```
public class MainActivity extends AppCompatActivity {
    String val;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void MessageALl(View view) {
        Intent in = new Intent(MainActivity.this, AdminChatActivity.class);
        startActivity(in);
    }
    public void ReportShow(View view) {
        Intent in = new Intent(MainActivity.this, ReportShow.class);
        startActivity(in);  }
}
```

4.9 Navigation Java file

```
public class NavigationActivity extends AppCompatActivity implements
AdapterView.OnItemClickListener {
    AutoCompleteTextView navsearch;
    TextView txtinfo;
    Button btn1;
    String[] rooms ={"Department of CS",
        "Department of Maths",
        "Department of Kannada",
        "Department of Hindi",
        "Department of English",
        "Department of MCA",
        "MCA LAB",
        "Mathematics Lab"};
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_navigation);
        navsearch = findViewById(R.id.navserach);
        txtinfo=findViewById(R.id.txtinfo);
        ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.select_dialog_item,rooms);
        navsearch.setThreshold(1);//will start working from first character
        navsearch.setAdapter(adapter);//setting the adapter data into the
AutoCompleteTextView
        navsearch.setOnItemClickListener(this);
    }
    @Override
    public void onItemClick(AdapterView<?> adapterView, View view, int i, long l) {
        String item = adapterView.getItemAtPosition(i).toString();
        if (item.equals(rooms[0])){
            txtinfo.setText("GJB BLOCK\nFirst Floor");
        }
        else if(item.equals(rooms[1])){
            txtinfo.setText("GJB BLOCK\nFirst Floor\nNext to CS Department");
        }
        else if(item.equals(rooms[2])){
            txtinfo.setText("ADMIN BLOCK\nFirst Floor\nOpposite to Triple C");
        }
        else if(item.equals(rooms[3])){
            txtinfo.setText("GJB BLOCK\nFirst Floor\nTurn left to find");
        }
        else if (item.equals(rooms[4])){
            txtinfo.setText("MCA BLOCK\nGround Floor");
        }
        else if (item.equals(rooms[5])){
            txtinfo.setText("MCA BLOCK\nThird  Floor\nTurn to right");
        }
```

```
        else if(item.equals(rooms[6])){
            txtinfo.setText("MTB BLOCK\nSecond Floor\nNext to Chemistry Lab");
        }
    }
}
```

4.10 Add Post Java file

```
public AddBlogsFragment() {
    // Required empty public constructor
}
FirebaseAuth firebaseAuth;
EditText title, des;
private static final int CAMERA_REQUEST = 100;
private static final int STORAGE_REQUEST = 200;
String cameraPermission[];
String storagePermission[];
ProgressDialog pd;
ImageView image;
String edititle, editdes, editimage;
private static final int IMAGEPICK_GALLERY_REQUEST = 300;
private static final int IMAGE_PICKCAMERA_REQUEST = 400;
Uri imageuri = null;
String name, email, uid, dp;
DatabaseReference databaseReference;
Button upload;
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    firebaseAuth = FirebaseAuth.getInstance();
    View view= inflater.inflate(R.layout.fragment_add_blogs, container, false);
```

```
        title = view.findViewById(R.id.ptitle);
        des = view.findViewById(R.id.pdes);
        image = view.findViewById(R.id.imagep);
        upload = view.findViewById(R.id.pupload);
        uid=FirebaseAuth.getInstance().getUid();
        pd = new ProgressDialog(getContext());
        pd.setCanceledOnTouchOutside(false);
        email=FirebaseAuth.getInstance().getCurrentUser().getEmail();
        Intent intent = getActivity().getIntent();

        databaseReference = FirebaseDatabase.getInstance().getReference("Users");
        Query query = databaseReference.orderByChild("email").equalTo(email);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                for (DataSnapshot dataSnapshot1 : dataSnapshot.getChildren()) {
                    name = dataSnapshot1.child("name").getValue().toString();
                    email = "" + dataSnapshot1.child("email").getValue();
                    dp = "" + dataSnapshot1.child("image").getValue().toString();
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
        cameraPermission = new String[]{Manifest.permission.CAMERA,
Manifest.permission.WRITE_EXTERNAL_STORAGE};
        storagePermission = new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE};

        image.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                showImagePicDialog();
            }
        });
        upload.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String titl = ""+title.getText().toString().trim();
                String description = ""+des.getText().toString().trim();
                if (TextUtils.isEmpty(titl)) {
                    title.setError("Title Cant be empty");
                    Toast.makeText(getContext(), "Title can't be left empty",
Toast.LENGTH_LONG).show();
                    return;
                }
                if (TextUtils.isEmpty(description)) {
                    des.setError("Description Cant be empty");
                    Toast.makeText(getContext(), "Description can't be left empty",
Toast.LENGTH_LONG).show();
                    return;
                }
                if(imageuri==null){
                    Toast.makeText(getContext(), "Select an Image",
Toast.LENGTH_LONG).show();
                    return;
                }else{
                    uploadData(titl,description);
        return view;
    }
    private void showImagePicDialog() {
        String options[]={ "Camera","Gallery"};
        AlertDialog.Builder builder=new AlertDialog.Builder(getContext());
        builder.setTitle("Pick Image From");
        builder.setItems(options, new DialogInterface.OnClickListener() {
```

```
        @Override
        public void onClick(DialogInterface dialog, int which) {

            if(which==0){
                if(!checkCameraPermission()){
                    requestCameraPermission();
                }
                else {
                    pickFromCamera();
                }
            }else if(which==1){
                if(!checkStoragePermission()){
                    requestStoragePermission();
                }
                else {
                    pickFromGallery();
                }
             builder.create().show();
    }
    private Boolean checkStoragePermission(){
        boolean result=
ContextCompat.checkSelfPermission(getContext(),Manifest.permission.WRITE_EX
TERNAL_STORAGE)
                ==(PackageManager.PERMISSION_GRANTED);
        return result;
    }
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {

        switch (requestCode){
            case CAMERA_REQUEST:{
                if(grantResults.length>0){
                    boolean
camera_accepted=grantResults[0]==PackageManager.PERMISSION_GRANTED;
```

```
                boolean
writeStorageaccepted=grantResults[1]==PackageManager.PERMISSION_GRANTE
D;
                if(camera_accepted&&writeStorageaccepted){
                   pickFromCamera();
                }
                else {
                   Toast.makeText(getContext(),"Please Enable Camera and Storage
Permissions",Toast.LENGTH_LONG).show();
                }
             }
          }
          //function end
          break;
          case STORAGE_REQUEST:{
             if(grantResults.length>0){
                boolean
writeStorageaccepted=grantResults[0]==PackageManager.PERMISSION_GRANTE
D;
                if(writeStorageaccepted){
                   pickFromGallery();
                }
                else {
                   Toast.makeText(getContext(),"Please Enable Storage
Permissions",Toast.LENGTH_LONG).show();
                }
             }
          }
          break;
       }

    }
    private void requestStoragePermission(){
       requestPermissions(storagePermission,STORAGE_REQUEST);
```

```
    }
    private Boolean checkCameraPermission(){
        boolean result=
ContextCompat.checkSelfPermission(getContext(),Manifest.permission.CAMERA)
                ==(PackageManager.PERMISSION_GRANTED);
        boolean result1=
ContextCompat.checkSelfPermission(getContext(),Manifest.permission.WRITE_EX
TERNAL_STORAGE)
                ==(PackageManager.PERMISSION_GRANTED);
        return result && result1;
    }
    private void requestCameraPermission(){
        requestPermissions(cameraPermission,CAMERA_REQUEST);
    }
    private void pickFromCamera(){
        ContentValues contentValues=new ContentValues();
        contentValues.put(MediaStore.Images.Media.TITLE,"Temp_pic");
        contentValues.put(MediaStore.Images.Media.DESCRIPTION,"Temp
Description");
        imageuri=getActivity().getContentResolver().insert(MediaStore.Images.Media.E
XTERNAL_CONTENT_URI,contentValues);
        Intent camerIntent=new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        camerIntent.putExtra(MediaStore.EXTRA_OUTPUT,imageuri);
        startActivityForResult(camerIntent,IMAGE_PICKCAMERA_REQUEST);
    }
    private void pickFromGallery(){
        Intent galleryIntent = new Intent(Intent.ACTION_PICK);
        galleryIntent.setType("image/*");
        startActivityForResult(galleryIntent, IMAGEPICK_GALLERY_REQUEST);
    }
    private void uploadData(final String titl, final String description) {

        pd.setMessage("Publishing Post");
        pd.show();
```

```
        final String timestamp=String.valueOf(System.currentTimeMillis());
        String filepathname=  "Posts/" + "post" +timestamp;
            Bitmap bitmap=((BitmapDrawable)image.getDrawable()).getBitmap();
            ByteArrayOutputStream byteArrayOutputStream=new
ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.PNG,100,byteArrayOutputStream)
;
            byte[] data=byteArrayOutputStream.toByteArray();

            StorageReference storageReference1=
FirebaseStorage.getInstance().getReference().child(filepathname);
            storageReference1.putBytes(data).addOnSuccessListener(new
OnSuccessListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
                    Task<Uri> uriTask=taskSnapshot.getStorage().getDownloadUrl();
                    while (!uriTask.isSuccessful());
                    String downloadUri=uriTask.getResult().toString();
                    if(uriTask.isSuccessful()){
                        HashMap<Object,String > hashMap=new HashMap<>();
                        hashMap.put("uid",uid);
                        hashMap.put("uname",name);
                        hashMap.put("uemail",email);
                        hashMap.put("udp",dp);
                        hashMap.put("title",titl);
                        hashMap.put("description",description);
                        hashMap.put("uimage",downloadUri);
                        hashMap.put("ptime",timestamp);
                        hashMap.put("plike","0");
                        hashMap.put("pcomments","0");
                        DatabaseReference
databaseReference=FirebaseDatabase.getInstance().getReference("Posts");
                        databaseReference.child(timestamp).setValue(hashMap)
                                .addOnSuccessListener(new OnSuccessListener<Void>() {
```

```
                        @Override
                        public void onSuccess(Void aVoid) {
                            pd.dismiss();
                            Toast.makeText(getActivity().getApplicationContext(),"Publ
ished",Toast.LENGTH_LONG).show();
                            title.setText("");
                            des.setText("");
                            image.setImageURI(null);
                            imageuri=null;
                            startActivity(new
Intent(getContext(),DashboardActivity.class));
                            getActivity().finish();
                        }
                    }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    pd.dismiss();
                    Toast.makeText(getContext(),"Failed",Toast.LENGTH_LONG).s
how();
        }).addOnFailureListener(new OnFailureListener() {
            @Override
            public void onFailure(@NonNull Exception e) {
                pd.dismiss();
                Toast.makeText(getContext(),"Failed",Toast.LENGTH_LONG).show();


    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
        if(resultCode == getActivity().RESULT_OK){
            if(requestCode==IMAGEPICK_GALLERY_REQUEST){
                imageuri=data.getData();
                image.setImageURI(imageuri);
            }
            if(requestCode==IMAGE_PICKCAMERA_REQUEST){
```

```
                image.setImageURI(imageuri);
              }
            }
            super.onActivityResult(requestCode, resultCode, data);
          }}
```

4.11 Strings XML file

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#6495ED</color>
    <color name="colorPrimaryDark">#6495ED</color>
    <color name="colorAccent">#0A527E</color>
    <color name="colorBlack">#000000</color>
    <color name="colorWhite">#fff</color>
    <color name="colorGray">#F5F1F1</color>
    <color name="colorgray01">#959595</color>
    <color name="colorGray02">#f5f0f0</color>
    <color name="colorGreen">#17581A</color>
    <color name="colorRed">#F4511E</color>
    <color name="btnRed">#E61313</color>
    <color name="btnedit">#2CA7E0</color>
    <color name="neutral">#0a1b97</color>
    <color name="colordrawer">#2B2C2E</color>
</resources>
```

CHAPTER 5

SOFTWARE TESTING

Software testing is the process of evaluating and verifying that the product does what it is supposed to do. The benefit of testing includes preventing bugs, reducing development costs and improving performance. The purpose of software testing is to evaluate the quality before it is released. Tests are designed to ensure that the program meets its requirements, that it performs as well as expected, or that there are no known errors.

5.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and the errors those tests uncover is limited by the constrained scope established for unit testing. It also focuses on the internal processing logic and data structures within the boundaries of a component. This type of testing can be conducted in parallel for multiple components.

5.2 Integration Testing

In this type of testing we test various integration of the project module by providing the input. The primary objective is to test the module interfaces in order to ensure that no errors are occurring when one module invokes the other module.

5.3 White Box Testing

White Box testing is one of the techniques in which internal structure, design and coding of the application are tested to verify the flow of input-output and to improve usability and security.

1. All logical decisions were checked for the true and falsity of the values.
2. All independent paths in the modules have been exercised.

5.4 Regression Testing

Regression testing is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change. This testing makes sure that the whole components works properly even after adding components to complete the application.

Test Case No. 1			
Test Title : Splash Screen Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Running Splash Screen	Takes to Login Page	Successful
2.	Click on Register button	Takes to registration page	Successful
3.	Give Valid login credentials	Takes to home page	Successful
4.	Invalid Login details	App does not take to the home page.	Successful

Table 5.1 Splash Screen Page Test

Test Case No. 2			
Test Title : Forgot Password Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Click on forgot password link	Takes to Forgot password recovery page	Successful
2.	After filling valid Email ID click on send password button	Email from MCC Connect received with the reset- password link in it.	Successful

Table 5.2 Forgot Password Page Test

Test Case No. 3			
Test Title : Profile Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Click on Edit button	Takes to Edit profile Data page	Successful
2.	Click on Update profile picture	Able to change profile picture	Successful
3.	Click on Update name	Able to change name	Successful
4.	Click on change password	Able to change password	Successful
5.	Click on Update designation	Able to change designation	Successful
6.	Click on Logout button	Logged out from the App	Successful

Table 5.3 Profile Page Test

Test Case No. 4			
Test Title : Chat Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Send Message to user A	Message received by User A	Successful

Table 5.4 Chat Page Test

Test Case No. 5			
Test Title : Add Blogs Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Click on Upload button	Post published message	Successful
2.	Click on report a post button	User email reported to admin	Successful

Table 5.5 Add Blogs Page Test

Test Case No. 6			
Test Title : Home Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Click on like post button	Like number increased by one	Successful
2.	Click on comment and write comment on post	Comment Added to the post	Successful

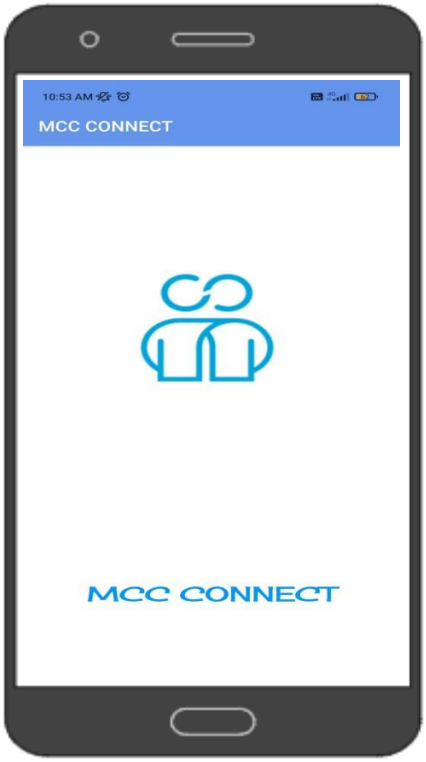
Table 5.6 Home Page Test

Test Case No.8			
Test Title : Admin Dashboard Page Test			
SL No.	**Test Case**	**Expected Result**	**Test Result**
1.	Click on Broadcast Message	Able to send message	Successful
2.	Click on report query	Able to view all reported queries	Successful
3.	Click on user Analytics	Able to see Analytics of app	Successful

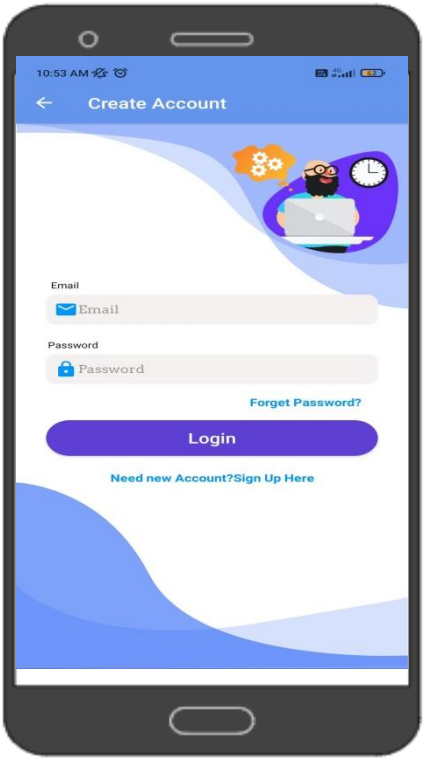
Table 5.7 Admin Dashboard Page Test

CHAPTER 6

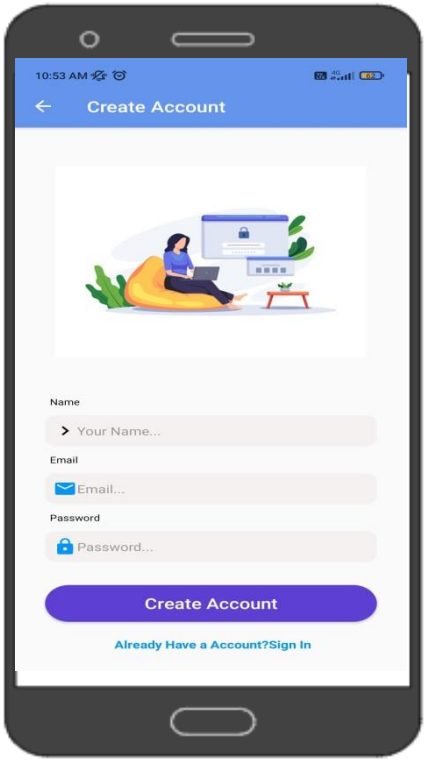
SAMPLE SCREENSHOTS



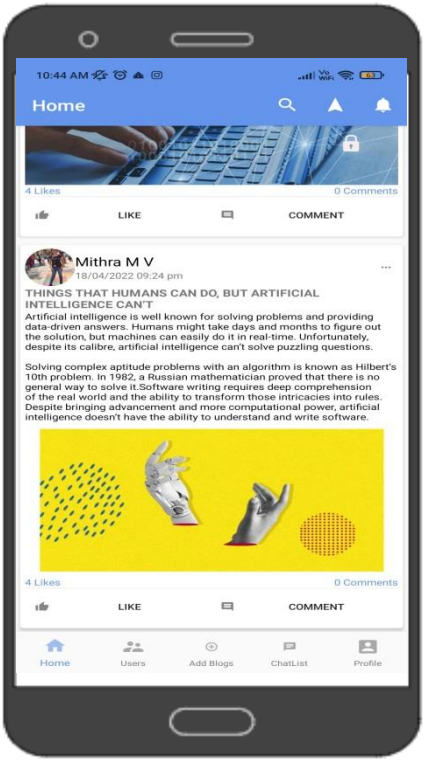
Screenshot 6.1 Splash Screen page



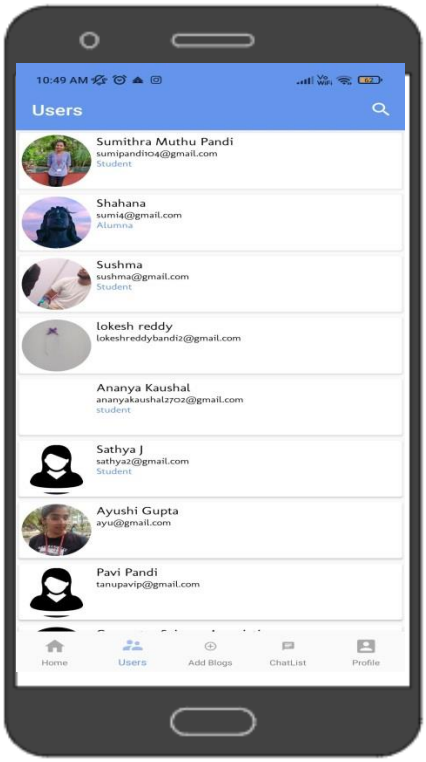
Screenshot 6.2 Login page



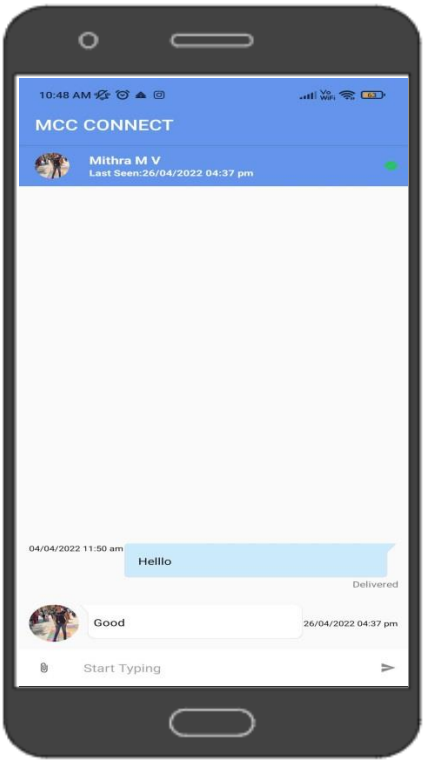
Screenshot 6.3 Registration Page



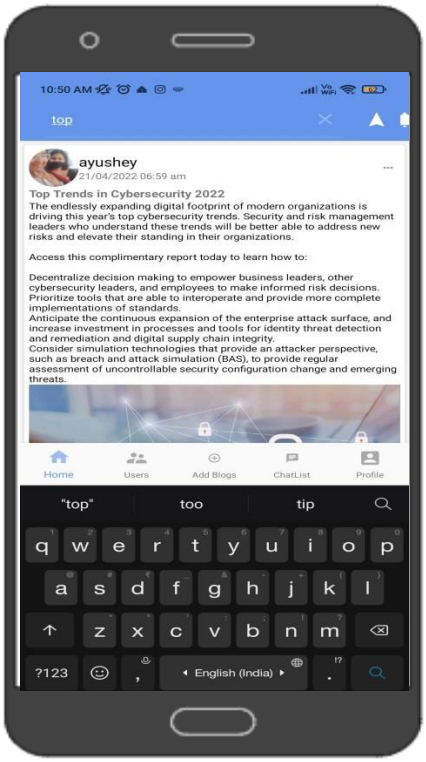
Screenshot 6.4 Home Page



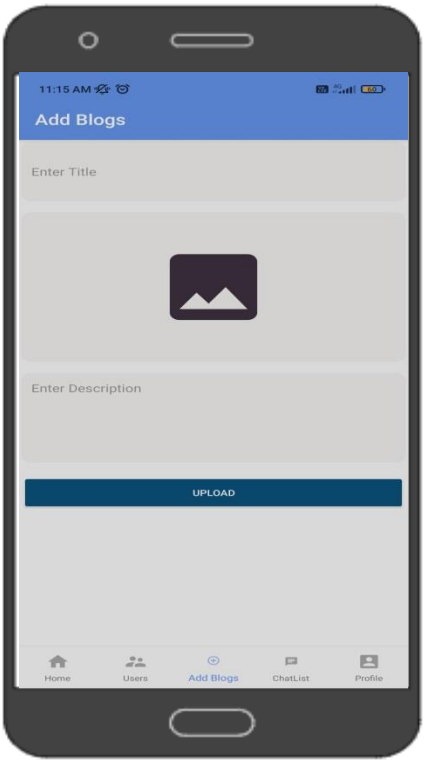
Screenshot 6.5 Users Page



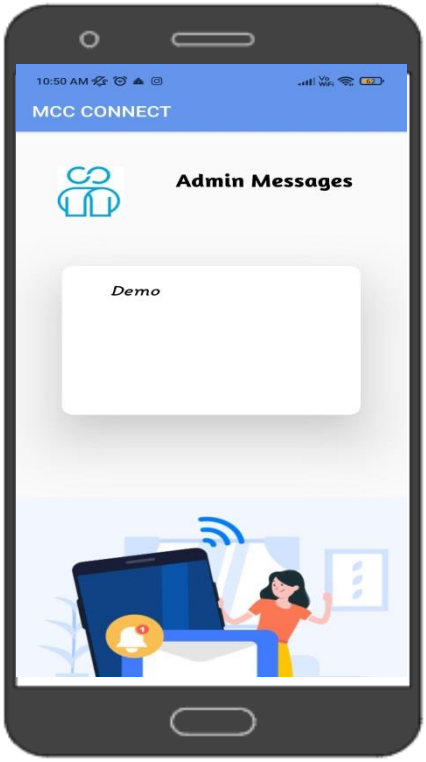
Screenshot 6.6 Chat Page



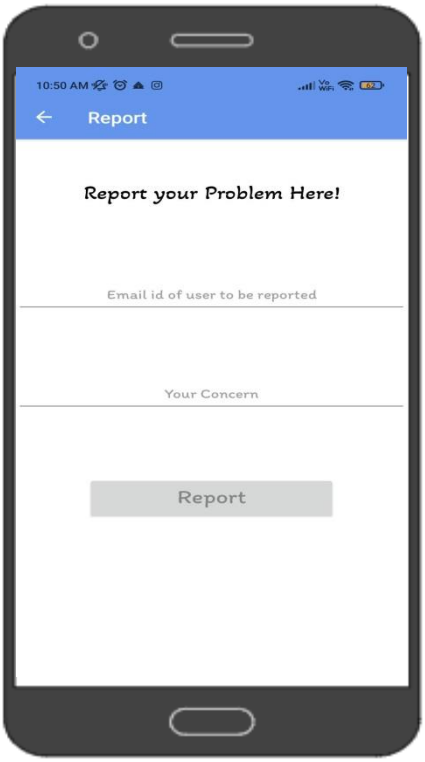
Screenshot 6.7 Search Page



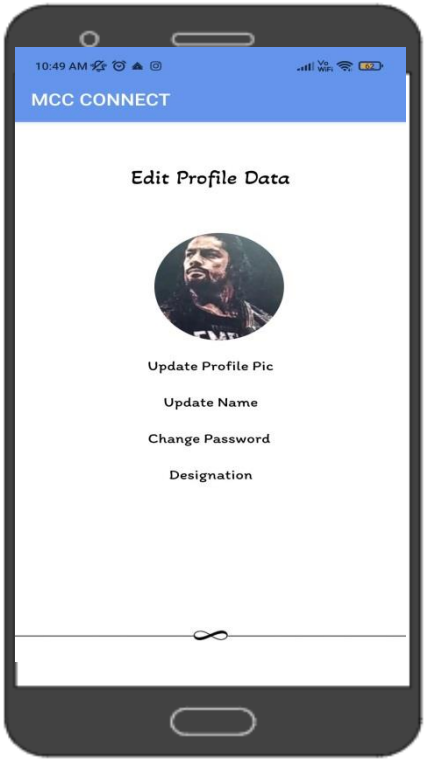
Screenshot 6.8 Add Post Page



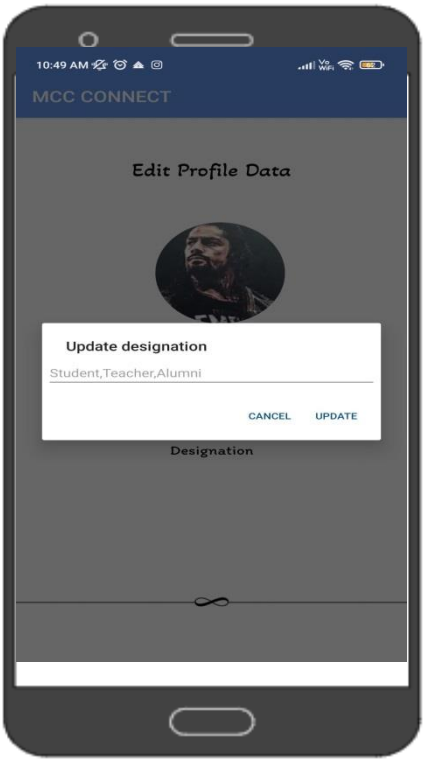
Screenshot 6.9 Admin Message receive Page



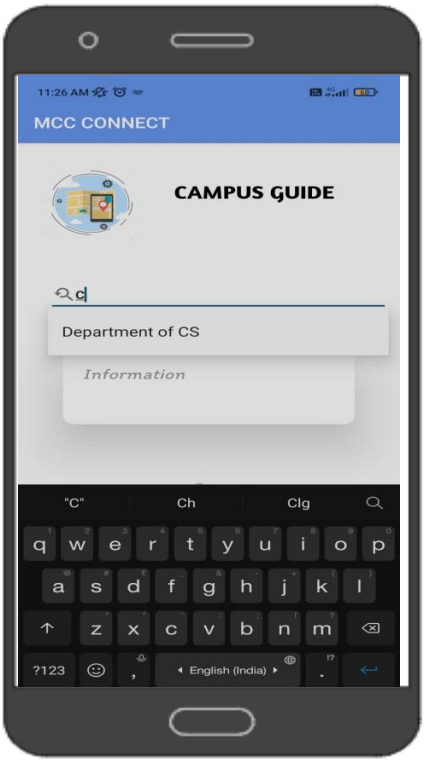
Screenshot 6.10 Report user Page



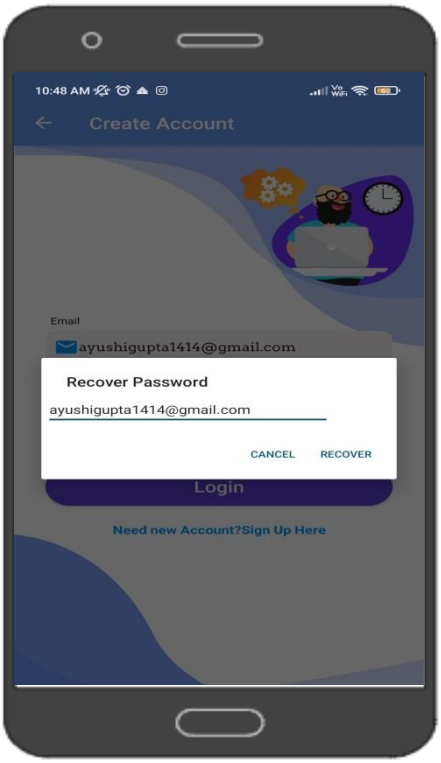
Screenshot 6.11 Edit profile data Page



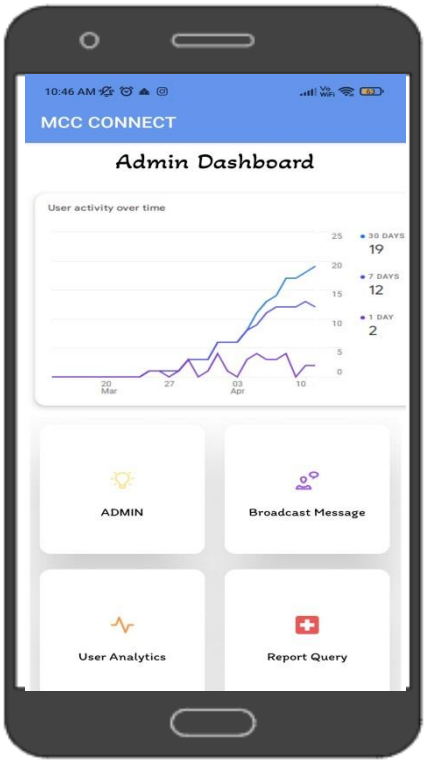
Screenshot 6.12 Update data Page



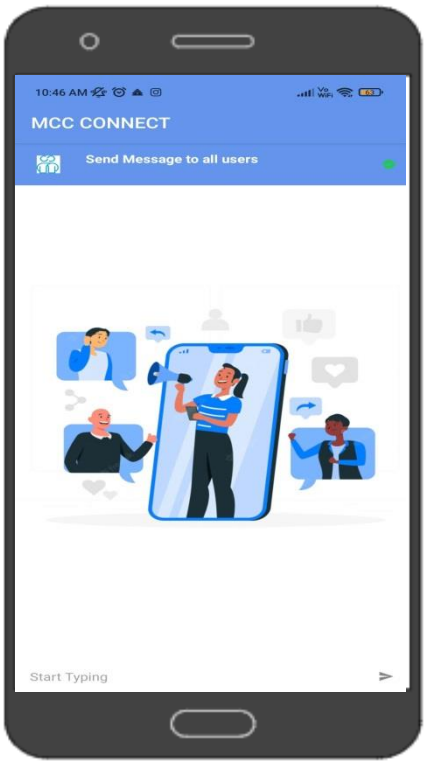
Screenshot 6.13 Campus guide Page



Screenshot 6.14 Recover password Page



Screenshot 6.15 Admin Page



Screenshot 6.16 Admin broadcast message Page

CHAPTER 7

CONCLUSION & SCOPE ENHANCEMENT

1. CONCLUSION

The Project entitled "MCC Connect" is a mobile application that satisfies the main objective to help students to know about current and upcoming updates on various activities and events organized by various departments and students and provide fast passes of information through verified applications. Also, it will be of greater advantage for college authorities to spread the broadcast message at one click this taking place to a wider number of students. This application will be a great advantage for the institute in having an integrated platform for all the necessary purposes like Campus guidance, Chat Activity, etc. Students can connect with each other no matter what is the designation of a person, may it be student, teacher, alumni etc. and share thoughts. Providing an easy way to convey updates by the institute.

2. SCOPE AND ENHANCEMENT

Though the MCC Connect application is efficient and user friendly, the real situation will be more challenging. However we constantly upgrade the project on a basis as per the current scenario as there will be new requirements growing day-to-day. Reusability of this application is also possible.

In future, as an enhancement this application could be made available to a broader spectrum of people by considering generalized construction of applications for more institutes which will require the same basic apk with least and best modification for making it as per institute requirements thus making this application available worldwide.. The event can be updated with more useful information for college students.

CHAPTER 8

APPENDIX

Figures	Description
Figure 2.1	Zero Level DFD
Figure 2.2	First Level DFD
Figure 3.1	Use Case Diagram
Table 5.1	Splash Screen Page Test
Table 5.2	Forgot Password Page Test
Table 5.3	Profile Page Test
Table 5.4	Chat Page Test
Table 5.5	Add Blogs Page Test
Table 5.6	Home Page Test
Table 5.7	Admin Dashboard Page Test

CHAPTER 9

BIBILOGRAPHY

Textbook references:

[1] Grant Allen, Beginning Android 4, 5th Edition, Apress, 2012.

[2] Wei-Meng Lee, Beginning Android 4 Application Development, John Wiley & Sons, 2012

[3] Dr. S. Regina Lourdhu Suganthi, Mobile Application Development, Subhas Publication, 2019.

[4] David Griffith, Head First Android Development : A Brain Friendly, 7th Edition, 2019.

[5] Houssem Yahiaoui, Firebase Cookbook, Packt Publication, 2nd Edition, 2017

Website references:

[1] https://www.w3schools.com/

[2] https://www.studentstutorial.com/

[3] https://www.stackoverflow.com/

[4] https://www.guru99.com/

[5] http://library.ucmerced.edu/

[6] https://firebase.google.com/docs