

**DEPARTMENT OF**

**MASTER OF COMPUTER APPLICATIONS**

**

**Minor Project Report**

**On**

**“ALUMINATE”**

***Submitted in Partial Fulfillment of the Requirement for the 3 Semester MCA Academic Minor Project***

***20MCA36***

**By**

**ISHITA SARKAR (1RV20MC028)**

**ATUL GUPTA (1RV20MC119)**

Under the Guidance of

**Dr R. Savitha**

Assistant Professor

Department of MCA

Department of Master of Computer Applications

RV College of Engineering®,

RV Vidyanikethan Post, Mysore Road

Bengaluru – 560059

*January 2022*

**RV COLLEGE OF ENGINEERING**

**(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)**

**DEPARTMENT OF**

**MASTERS OF COMPUTER APPLICATIONS**

**Bengaluru– 560059**



**CERTIFICATE**

This is to certify that the project entitled “**ALUMINATE**” an Android Application submitted in partial fulfillment of **Minor Project-I (20MCA36)** of **III Semester MCA** is a result of the bonafide work carried out by **ISHITA SARKAR (1RV20MC028) and ATUL GUPTA (1RV20MC119)** during the Academic year

2020-21.

Dr. R Savitha Dr. Andhe Dharani

Assistant Professor Professor and Director

Department of MCA Department of MCA

RV College of Engineering RV College of Engineering

**RV COLLEGE OF ENGINEERING**

**(Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi)**

**DEPARTMENT OF**

**MASTERS OF COMPUTER APPLICATIONS**

**Bengaluru– 560059**

**DECLARATION**

It is to certify that **ISHITA SARKAR (1RV20MC028) and ATUL GUPTA (1RV20MC119)**, student of Third Semester MCA hereby declare that the Minor Project titled **“ALUMINATE**” has been carried out and completed successfully by me and is my original work.

**DATE OF SUBMISSION: 31/01/2022 ISHITA SARAKR**

**(1RV20MC028)**

**ATUL GUPTA**

**(1RV20MC119)**

**3rd SEMESTER STUDENT**

**MASTER OF COMPUTER APPLICATIONS**

**RV COLLEGE OF ENGINEERING**

**Acknowledgement**

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crowned the efforts with success.

I would like to profoundly thank Management of RV College of Engineering® for providing such a healthy environment for the successful completion of the project. I would like to express my thanks to the Principal **Dr. K.N. Subramanya**, RV College of Engineering® for his encouragement that motivated me for the successful completion of the Project. I would like to express my deepest sense of gratitude to **Dr. Andhe Dharani**, Professor and Director, Department of MCA, RV College of Engineering® for her constant support and guidance. I would like to express my deepest sense of gratitude to guide **Dr. R Savitha**, Professor and Director, Department of MCA, RV College of Engineering® for her complete project guidance and being a mentor constantly providing me with valuable feedback and corrections.

I thank all the teaching staff and technical staff of Master of Computer Applications, RV College of Engineering® for their help and support. Finally, I would like to express my sincere thanks to all staff members of the MCA Department for their valuable guidance and support.

**ISHITA SARKAR**

**(1RV20MC028)**

**ATUL GUPTA**

**(1RV20MC119)**

**3rd SEMESTER STUDENT**

**MASTER OF COMPUTER APPLICATIONS**

**RV COLLEGE OF ENGINEERING**

**Abstract**

A social networking application is an online platform that allows users to create a public profile and interact with each other. In this rapidly updating world, there’s a need of a network that helps the institution to keep track of their alumnae. And also, there might be a need of a friendly guidance to the students regarding their career. The main objective of this project is to build an alumni application and social networking site that establishes a link between the present students and the alumnae of the institute. This helps the institution keep a record of their old students. Also, it does not require much staff involvement and thus, will cleanly act as a network between the alumnae and the present students. This application provides a user-friendly interface that provides the user to understand the working of the application easily. The user will be able to sign up and log in to their user profile and interact with each other. The user can share ideas, digital photos and posts, chat with each other, and inform others about online or real-world activities and events.

Developing mobile application is a crucial and essential process in any organization to improve the communication and productivity. Creating a mobile application can accomplish a complex and time-consuming task in an efficient way. In this project, an android application system called “Aluminate” is proposed to develop an effective mobile user interface design to connect and collaborate former students of institution. Smart organization of screen elements in the system helps the people to understand the interface easier and make communication quickly.

The system is developed in an Android platform using java for backend, xml to design user interface and firebase to store data, makes effective communication among different batches of graduate from the same Institution. The purpose of this application is to provide a mobile user interface, which facilitates a data storage, authenticate a user, and provide different services. The services include connect and communicate an individual alumnus, making effective interaction with individual user, share knowledge and publish newsletter, etc.

Advantages of an Alumni application consists of; firstly, the main advantage is to find friends and also to be contact with old friends, secondly, news regarding alumni meet can be easily meet can be easily known since this system Conway such information to the passed students once posted, lastly, this can post the various information regarding job vacancies, competitive exams and so on. So, one's circle becomes big. Thus, some of the disadvantages of this application are no video streaming, no audio calling, no Dynamic Grouping

***Contents Page No***

College Certificate ii

Undertaking by student iii

[Acknowledgement iv](#_heading=h.30j0zll)

[Abstract v](#_heading=h.1fob9te)

[Table of Contents vi](#_heading=h.3znysh7)

List of Figures vii

[Chapter 1: Introduction 1](#_heading=h.tyjcwt)

[1.1 Project Description 1](#_heading=h.3dy6vkm)

[Chapter 2: Literature Review 2](#_heading=h.1t3h5sf)

* 1. [Literature Survey 2](#_heading=h.4d34og8)
  2. Existing and Proposed System 5
  3. [Tools and Technologies used 6](#_heading=h.2s8eyo1)
  4. Hardware and Software Requirements 6

[Chapter 3: Software Requirement Specifications 7](#_heading=h.17dp8vu)

* 1. [Introduction 7](#_heading=h.3rdcrjn)
  2. General Description 8
  3. [Functional Requirement 9](#_heading=h.26in1rg)
  4. [External Interfaces Requirements 1](#_heading=h.lnxbz9)2
  5. [Non-Functional Requirements 1](#_heading=h.35nkun2)3

[Chapter 4: System Design 1](#_heading=h.1ksv4uv)4

* 1. System Perspective /Architectural Design 14
  2. [Module Specification 1](#_heading=h.44sinio)6

[Chapter 5: Detailed Design 17](#_heading=h.2jxsxqh)

[5.1 System Design 17](#_heading=h.z337ya)

[Chapter 6: Implementation 2](#_heading=h.3j2qqm3)1

* 1. Code Snippets 21
  2. [Screenshots](#_heading=h.1y810tw) 43

[Chapter 7: Testing](#_heading=h.4i7ojhp) 47

[Chapter 8: Conclusion](#_heading=h.2xcytpi) 51

[Chapter 9: Future Enhancements](#_heading=h.1ci93xb) 52

[Chapter 10: References](#_heading=h.3whwml4) 53

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure No | Figure Label | Page No |
| 3.1 | Abbreviation and Explanation | 7 |
| 3.3.1 | Login/Registration | 9 |
| 3.3.2 | Follow Request | 10 |
| 3.3.3 | Post Upload | 11 |
| 3.4.1 | Splash Screen | 12 |
| 3.4.2 | Android Device | 12 |
| 4.1.1 | Architectural Design | 14 |
| 4.1.2 | Architectural Design with Merger | 15 |
| 5.1.1 | Use Case Diagram | 17 |
| 5.1.2 | Sequence Diagram | 18 |
| 5.1.3 | Activity Diagram | 19 |
| 5.1.4 | Data Diagram Of Following a User | 20 |
| 6.2.1 | Login | 42 |
| 6.2.2 | Login with ID | 42 |
| 6.2.3 | Post | 42 |
| 6.2.4 | Homepage | 42 |
| 6.2.5 | Comment | 43 |
| 6.2.6 | Notification | 43 |
| 6.2.7 | Chat | 43 |
| 6.2.8 | Check If User is Online | 43 |
| 6.2.9 | Chat box | 44 |
| 6.2.10 | Profile | 44 |
| 6.2.11 | Friends | 45 |
| 7.1 | Test Case 1 | 46 |
| 7.2 | Test Case 2 | 47 |
| 7.3 | Test Case 3 | 48 |
| 7.4 | Test Case 4 | 49 |

**Chapter 1: Introduction**

The title of the project is Aluminate. This project idea came to us while we were discussing about what would happen if there would be no restrictions on chatting over internet and that too something like a safe for people to freely interact with their alumni that’s when it hit us why not we ourselves consider it as a project idea and go further with it. Thus, the objective of Aluminate is to let people freely connect to their alumni over an application where they can directly chat or interact with their alumni without any restriction and one can even made posts related to recruitments, else than this it will have video and voice calling facility along with the option to create a group with unlimited members.

* 1. **Project Description**

The existing applications have multiple drawbacks starting from:

* + - Limitations on adding n number of people in a group
    - If the account is private such as in LinkedIn we cannot message directly
    - It is difficult to connect to our alumni if we don’t have their username or details to

search for their account

Scope of the project:

* Aluminate will help students to connect to their alumni without any restrictions
* So that they can directly discuss with them regarding any company placements or ask for their help and guidance through messages
* Else than that the alumni can also make posts related to anything that is helpful for the students

# Chapter 2: Literature Review

A literature review is a comprehensive summary of previous research on a topic. This literature review surveys scholarly articles, books, and other sources relevant to the project’s area of research. This chapter review enumerates, describe, summarize, objectively evaluate and clarify the previous research and gives a brief outcome of the main survey done.

## **Literature Survey**

* **Alumni Database Management System [1]**

**Authors :** Tarun Kumar, Yeeshu Prateek, Prajwal Atharga, Dr. Rajashekarappa, Prof. V. K. Parvati,

Alumni can communicate to the students regarding job opportunities and the students can share the department technology activities to the alumni. The alumni and the student can communicate only through the admin permission.

* **Design and Implementation of Student and Alumni Web Portal [2]**

**Authors:** Shaimaa Q. Sabri,Akeela M. Ahmad,Maiwan B. Abdulrazzaq,

Science Journal of University of Zakho 5(2):272

The Information and Communication Technology (ICT) has witnessed great development in the recent years. Therefore, the design of Students and Alumni Web Portal (SAWP) involves the analysis of the internal and external environment of the three universities.

* **CAs Based Student-Alumni Management System [3]**

**Authors:** D.V. Nishanth; Satish; Niteesh S. Narasimha,

Students and alumni management system of any university is a collaborative framework which integrates the knowledge between the students who are pursuing education and the students who have already successfully established themselves after graduation.

* **Centralized Alumni Management System (CAMS) - A Prototype Proposal [4]**

**Author:** Aritra Mukherjee; Adrita Roy,

The importance of a centralized alumni application is highlighted it this paper.

* **Alumni Info-Com with Distinct Classification of Data using Support Vector Machine Algorithm [5]**

**Author:** R. Sasikumar, B. Haritha, T. Borshiya Vincy, M.Kamali, S. Deva Priya,

International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-8 Issue-6, March 2020

This paper presents to create a centralized alumni network for betterment of institutions and upcoming student's community

* **Alumni Interaction System [6]**

**Authors:** Subashini.S , Sowndarya.A,

International Journal of Computer Science Trends and Technology (IJCST) – Volume 5 Issue 2, Mar – Apr 2017,

The main aim of the project is to build an interaction between alumni, admin and the students. A system that will be able to manage alumni data of a college and provide easy access to the same. The alumni will also be interested to maintain relations with their institutions.

* **BVDUCOE College Alumni** [7]

**Authors:** Portal Rohit Singh, Rajat Singh Parmar, Saurabh Tripathi Prof. Rohini Khalkar, Prof. Sheetal Patil

International Journal of Computer Science Trends and Technology (IJCST) – Volume 5 Issue 2, Mar – Apr 2017

The aim of Alumni portal application is to allow old and current students of the college to have interaction. This allows students/alumni to know about themselves and their current activities. This portal focuses the feature of interaction/communication, which will enable the current students to have interaction with the alumni of the college or organization for getting various updates on current industry trends, Internship opportunities, project sponsorship and various referrals opening in the corporate world.

* **Personalized expertise search at LinkedIn** [8]

**Authors:** Viet Ha-Thuc, Ganesh Venkataraman, Mario Rodriguez, Shakti Sinha, Senthil Sundaram, Lin Guo

How LinkedIn helps one to personalize there search results.

* 1. **Existing and Proposed Systems**

**EXISTING**

* Many institutions and universities maintain the information manually about present and past students. This does not allow efficient data management and retrieval process. Most of the apps to accomplish a complex and time-consuming task. It is developed to an efficient mobile user interface design to connect and collaborate former students. It developed on Android platform and it facilitates various services to connect with each other. This allows the user to post messages, displays recent news and photos, live conversation with their friends.
* The existing system is built with numberless excel sheets that are created by each user. These sheets may be collated by an alumni organization and shared with all the alumni but this activity may not be frequent. The system is difficult to maintain on a regular process and it also have a privacy issue.

**PROPOSED**

* The application will allow a student to message, follow an alumni or another student from his/her own college.
* Android operation system has more than 70% of worldwide mobile operating system market. That is why Aluminate is based on Android so that more and more users will be able to use it.
* Firebase Realtime database makes this application light, fast and real time. it stores the data in JSON format which makes the access to database easy.
* Realtime one to one chat helps users to chat. When a user sends a message to another user the message will first go to Firebase database server and then from their recipients UID will be fetched and message will be delivered to that user.
* User can add stories on their home page which will be disappeared within 24 hours.
* This application also offers option to add post so one can post something on their home page and it will be displayed on every user’s home page.
* Likes, comments on each post will be updated in dynamically, no of likes and comments with timestamps are stored in the database.

## **Tools and Technologies Used**

Android Studiois the official Integrated Development Environment (IDE) for android application development. Android Studio provides more features that enhance our productivity while building Android apps. Android Studio was announced on 16th May 2013 at the Google I/O conference as an official IDE for Android app development. It started its early access preview from version 0.1 in May 2013. The first stable built version was released in December 2014, starts from version 1.0.

Javais a general-purpose, class-based, object-oriented programming language designed for having lesser implementation dependencies. It is a computing platform for application development. Java is fast, secure, and reliable, therefore. It is widely used for developing Java applications in laptops, data centers, game consoles, scientific supercomputers, cell phones, etc.

XML is used **to implement UI-related data,** and it’s a lightweight markup language that doesn’t make layout heavy. XML only contains tags, while implementing they need to be just invoked. MySQL/Mongo DB/Firebase to save the credentials in the database.

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grows 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.

* 1. **Hardware and Software Requirements**
     1. **Hardware Requirements**
* System- Smartphone running on or above Android 6.0
* Internet connection
  + 1. **Software Requirements:**
* XML for user interface
* Java for backend
* Firebase for database
* Dependencies-Picasso
* Android operating system

# Chapter 3: Software Requirement Specifications

## **3.1 Introduction**

The application can be used by students and professors/staff of the college. Application can be easily installed from Google play store and it will work on or above Android version 6.0. The user interface of the website has been kept simple and plain for easy usage of all the features. This application will help students to connect alumni of the college without any hussle.

|  |  |
| --- | --- |
| **Abbreviation** | **Explanation** |
| XML | Extensible Markup Language |
| AVD | Android Virtual Device |
| Android Studio | **Android Studio** provides the fastest tools for building apps on every type of Android device. |
| SDK | Software Development Kit |
| API | Application Programming Interface |
| APK | Android Package Kit |

*Table 3.1 : Abbreviation and Explanation*

## **3.2. General Description**

## **3.2.1.Product Perspective**

* Displays the whole list of students present in the application at that moment of time
* The Android application enables easier searching and is super portable
* The Android Application is easy to download, one can download it from play store and create an easy login and password
* Easy access in other user’s profile and easy messaging

**3.2.2. Product Functions**

* The Android application is less than 25mb and stores all the user data over cloud
* The Android Application makes searching a user far easier and messaging them quicker
* User can make job related posts which can be helpful for other students

**3.2.2. User Characteristics**

**User of the system can perform following functions:**

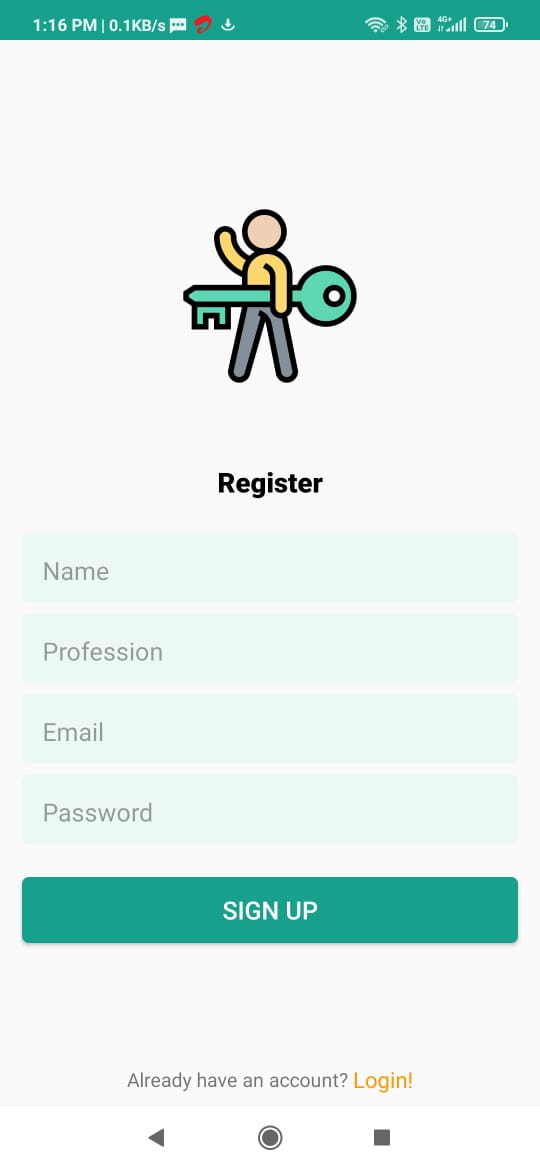
* Search for an alumni or existing student
* The user can easily login into the application using a username and password stored securely in the database
* The user can view all the students currently using the application
* Use the chat section to chat with other users

**3.3 Functional Requirements**

* USER LOGIN / REGISTRATION

The user can simply sign in using Google account or create a new account by providing required personal information about themselves. After creating the account it will display account created successfully.

* + - Input- sign in using Google account or create a new account
    - Process-authenticate the user’s mail id
    - Output- display account created



3.3.1 REGISTRATION

* Searching a friend

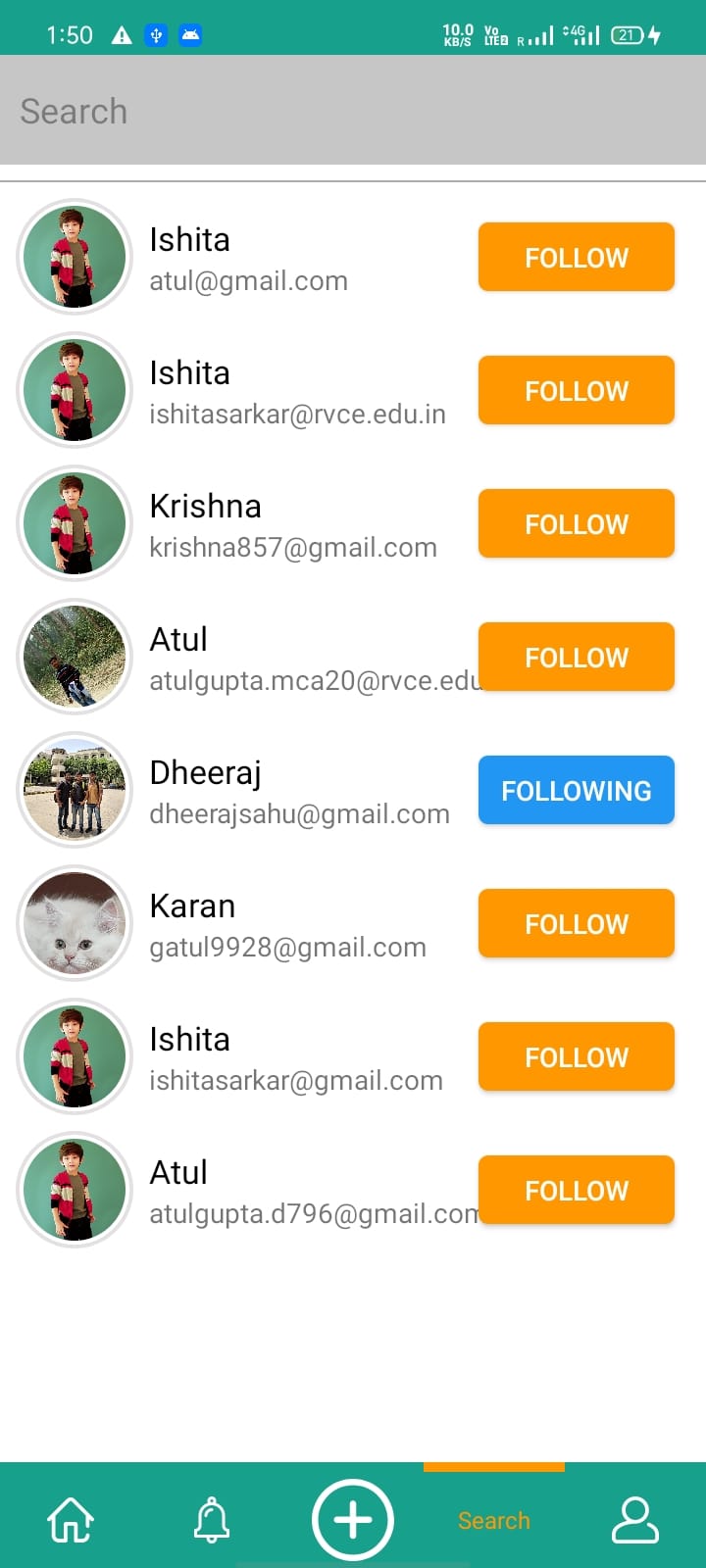
Search for those people whom the user wants to follow.

* + - Input- enter any name in search box
    - Process-search through database
    - Output- display searched result on screen
* Sending Follow request

After finding the people the user can send him or her the follow request and other

can also send follow request to the user.

* + - * Input- follow button
      * Process-either send follow request or directly follows if the account is public
      * Output- shows user as followed in user profile

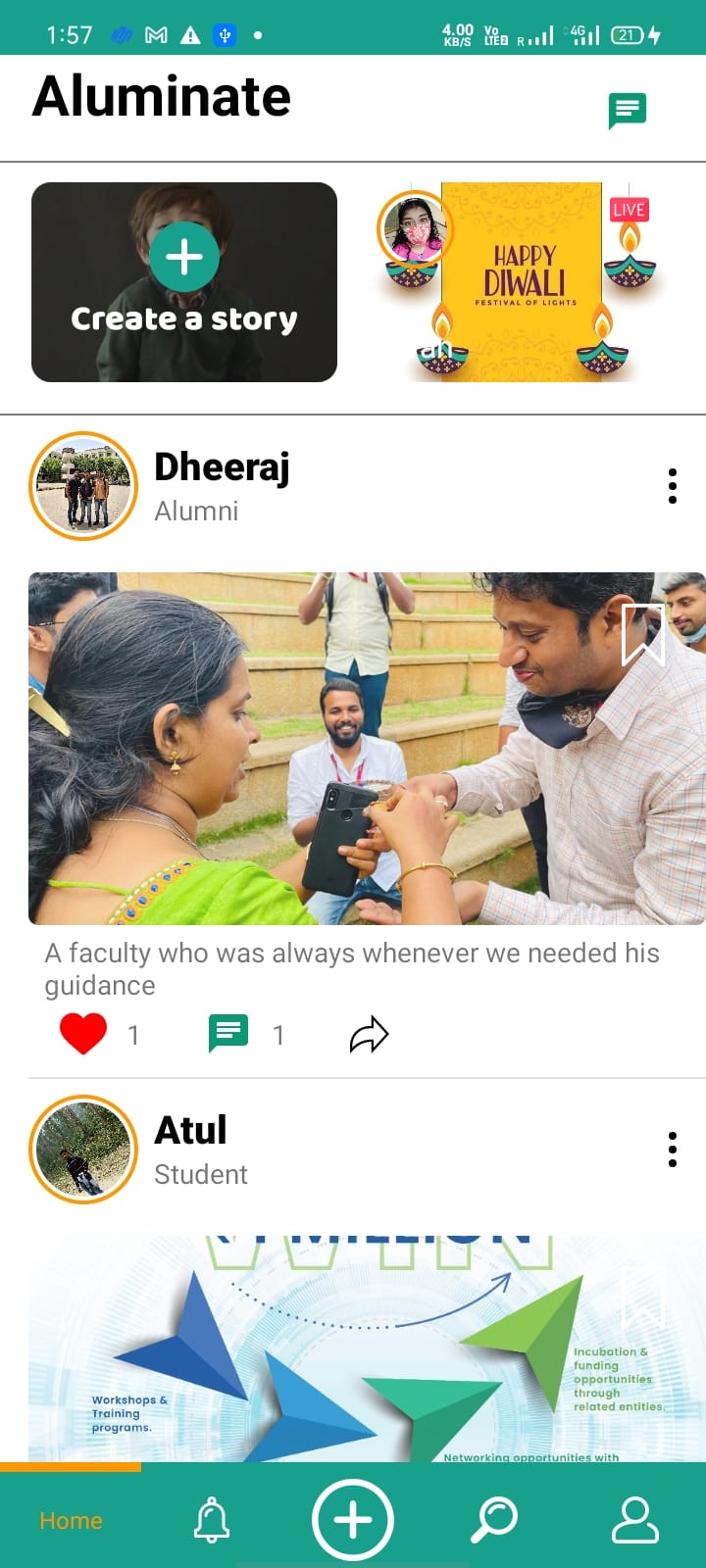


3.3.2 FOLLOW REQUEST

* Uploading Photos, Videos

To upload any photo or video first select the respective file from the gallery if someone wants to edit they can. After editing the can simply share the photo or the video or any other text posts.

* + - * Input- access photos through gallery
      * Process-uploads the photo in the application
      * Output- photos can be viewed in user profile



3.3.3. POST UPLOAD

**3.4 External Interfaces Requirements**

**3.4.1. Software Interfaces**

* The device should have Android Operating System
* Firebase database to track all the posts, images and chat

****

3.4.1 SPLASH SCREEN

**3.4.2. Hardware Interfaces**

* Any mobile device to use the Android Application



**3.4.2 ANDROID DEVICE**

## **3.5. Non-Functional Requirements**

* **Security**
* The system use SSL(secured socket layer) in all transactions that includes any other confidential passenger information, it's the standard technology for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems, preventing criminals from reading and modifying any information transferred, including potential personal details.
* The system should secure that it should not show any cookies regarding the password or the username of the user so that no one rather than the user can access the system.
* **Reliability**
* The system provides database for storage for all kinds of android devices whether it mobile or something else.
* The reliability of the whole system depends on the reliability of the separate components.
* The system should be such reliable that it should not crash or hang during the use by the user.
* **Maintainability**
* The application will be maintained and updated on a timely basis
* New modules can be easily added to the application
* **Portability**
* It should run on any android device or in any operating system android.

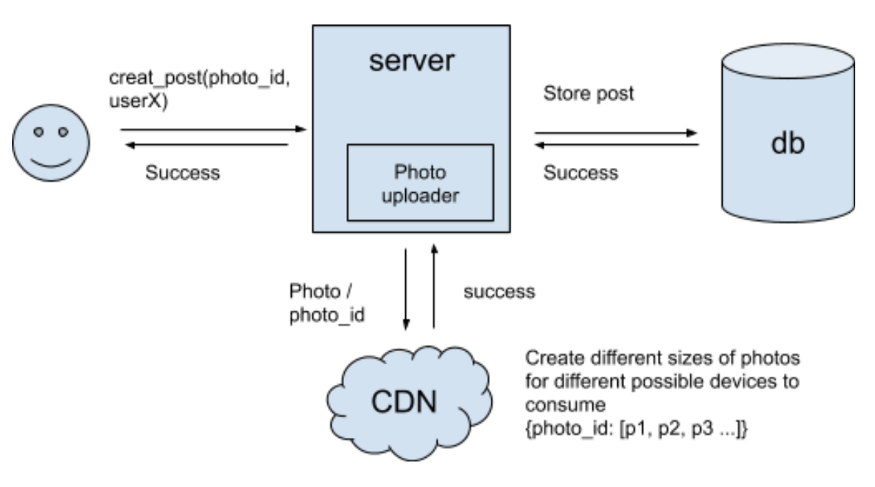
# Chapter 4: System Design

**4.1 Architectural Design**

**4.1.1 Problem Specification**

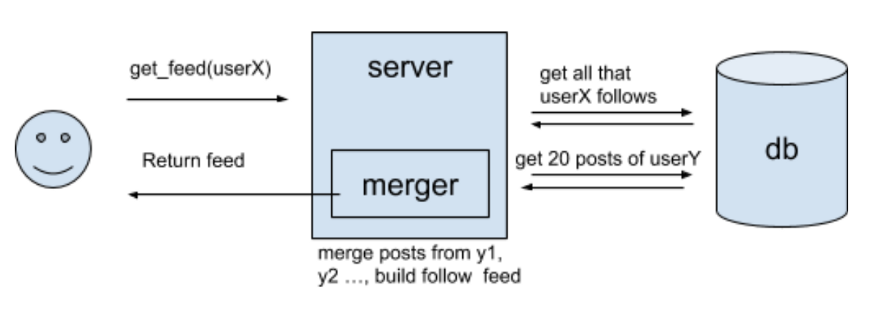
* + - Limitations on the usage if internet is not available
    - No way to know which user’s account is active currently
    - Not able to make calls to the other user

**4.1.2 Activity Diagram**



4.1.2.1**Activity Diagram**

A content delivery network (CDN) is a group of geographically distributed servers that speed up the delivery of web content by bringing it closer to where users are. ... CDNs cache content like web pages, images, and video in proxy servers near to your physical location.

**4.1.2.2ACTIVITY DIAGRAM IMPLEMENTING MERGER**

The MERGE statement in Firebase is a very popular clause that can handle inserts, updates, and deletes all in a single transaction without having to write separate logic for each of these.



**4.2 Module Specification**

* **Module 1:**

**CREATE ACCOUNT**

* Input- sign in using Google account or create a new account
* Process-authenticate the user’s mail id
* Output- display account created
* **Module 2**

**Like, Comment, Share**

* Input-clicks like or share buttons
* Process-counts number of likes, comments and share
* Output-displays total likes and shares below the post itself
* **Module 3**

**Uploading Status**

* Input- sequence of text or photo
* Process-uploads status in dashboard and deletes after 24 hour of uploading
* Output- status visible in the dashboard
* **Module 4**

**Uploading Photos, Videos**

* Input- access photos through gallery
* Process-uploads the photo in the application
* Output- photos can be viewed in user profile

# Chapter 5: Detailed Design

**5.1 SYSTEM DESIGN**

* + 1. **Use Case Diagram**

A use case diagram is a dynamic or behaviour diagram in UML.

User needs to register to the application if they are a first-time user.

Sign Up

Once the user registers to the application they need to login to the application.

Sign In



User is allowed to edit their profile such as their profile picture, cover photo and their bio as well.

Edit Profile

User can search for other users from the list of users who are currently using the application.

Search

User

User can add posts which is visible to other users using the application.



Add Post



User

User is allowed to comment on other people’s posts.

Comment

Post

User can follow other users and vice versa.

Follow

Users

Users are allowed to send direct messages to other users with whom they want to connect to.

Send

Messages



User can log out the application whenever they wish too.

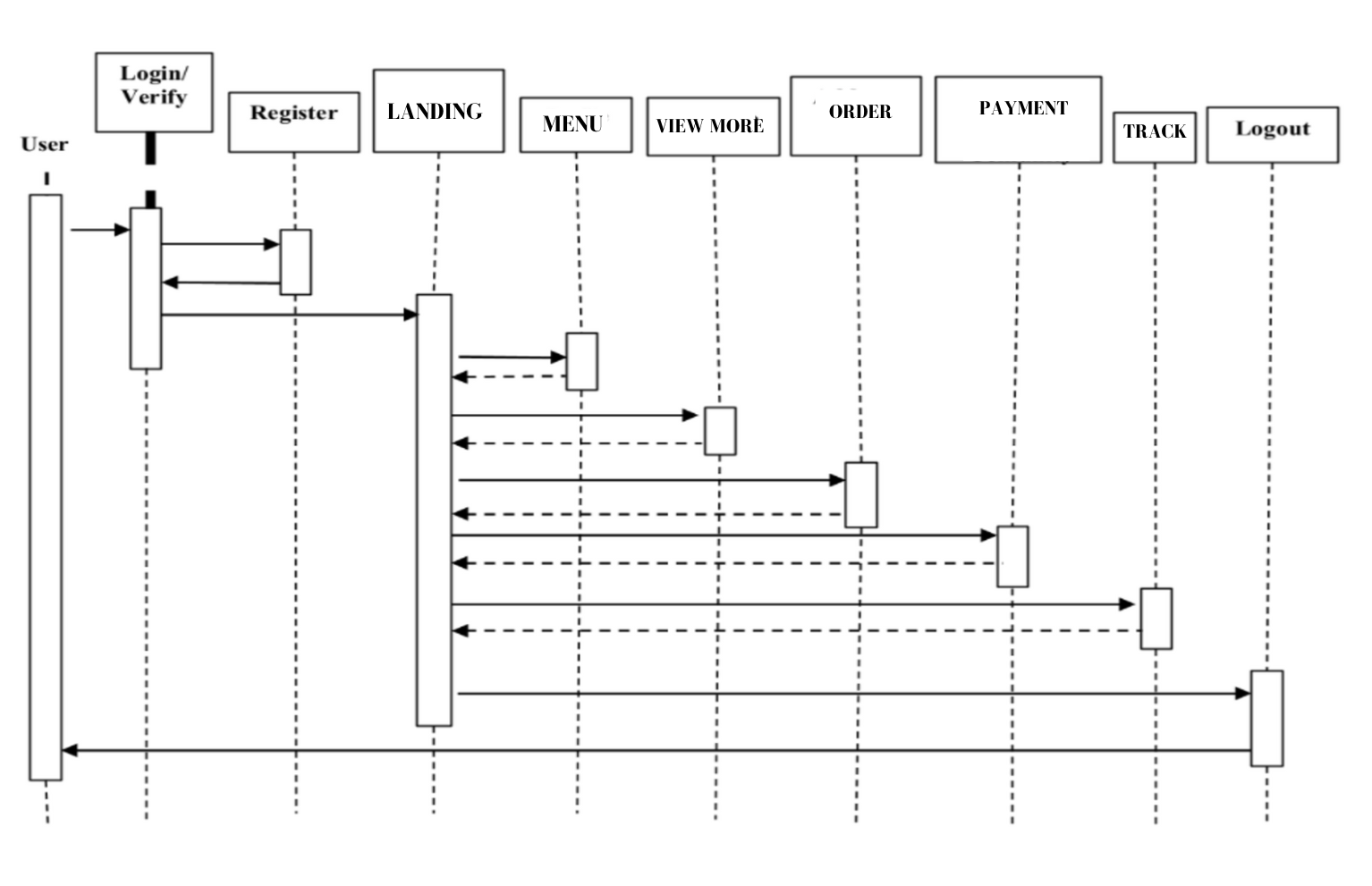
Log Out



5.1.1 Use Case Diagram

**5.1.2. Sequence Diagram**

A sequence diagram shows object interactions arranged in time sequence.

 5.1.2 SEQUENCE DIAGRAM

VERIFY / LOGIN



LOGOUT

CHAT

NOTIFICATIONS

POSTS

FRIENDS

PROFILE



HOMEPAGE



REGISTER

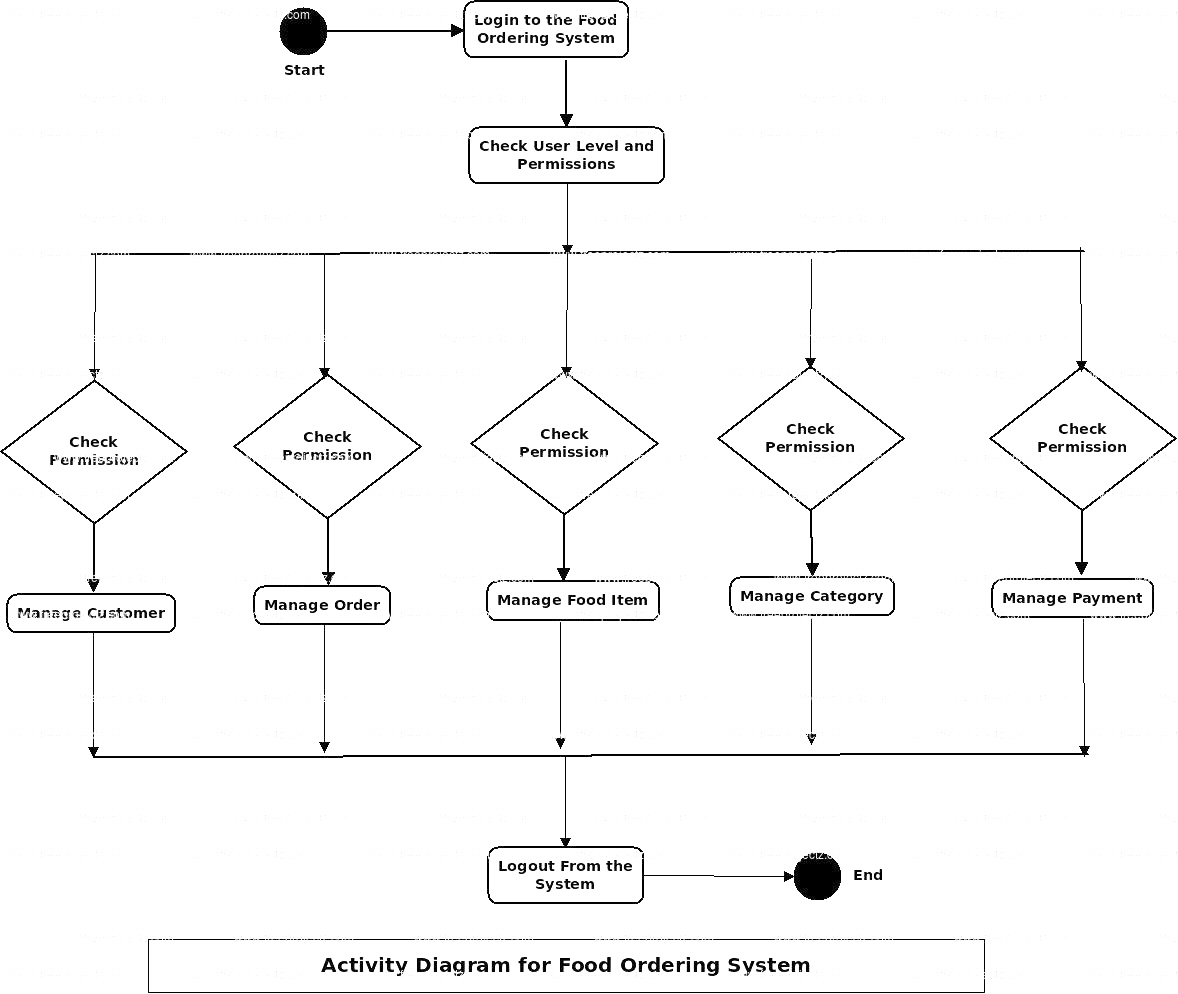


## **Activity Diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

LOGIN TO

ALUMINATE



LOGOUT FROM

ALUMINATE

MANAGE STATUS

MANAGE FRIENDS

MANAGE PROFILE

MANAGE CHATS

MANAGE POSTS

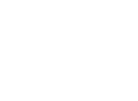
CHECK

PERMISSION

VERIFY CREDENTIALS

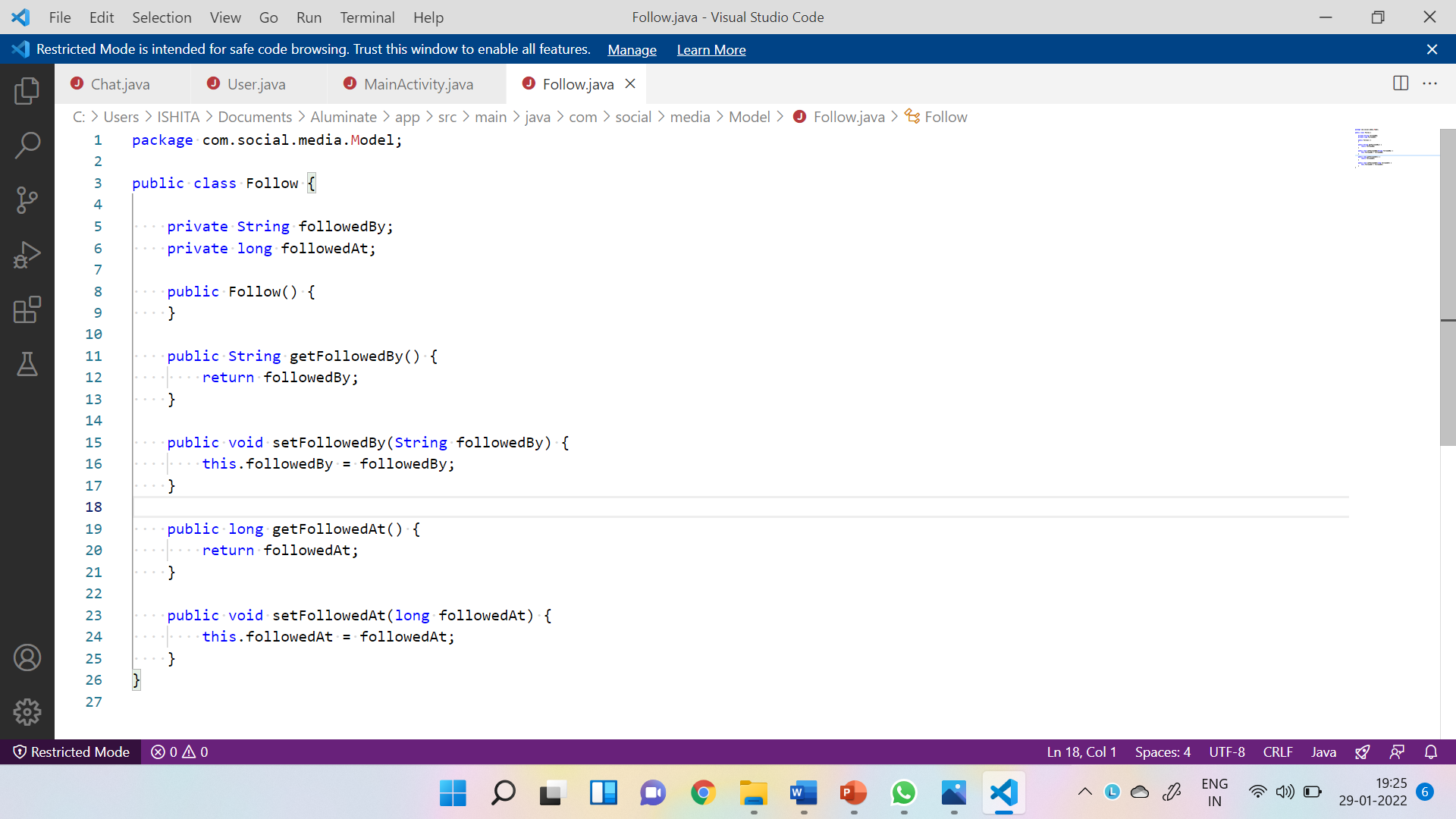
END

START



5.1.3 ACTIVITY DIAGRAM

## **Data Definition**



5.1.4 DATA DEFINITION OF FOLLOWING USER

# Chapter 6: Implementation

**6.1 CODE SNIPPETS**

**6.1.1 HOME FRAGMENT JAVA**

package com.social.media.Fragment;

import android.app.ProgressDialog;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import androidx.activity.result.ActivityResultCallback;

import androidx.activity.result.ActivityResultLauncher;

import androidx.activity.result.contract.ActivityResultContracts;

import androidx.annotation.NonNull;

import androidx.constraintlayout.widget.ConstraintLayout;

import androidx.core.widget.NestedScrollView;

import androidx.fragment.app.Fragment;

import androidx.recyclerview.widget.DividerItemDecoration;

import androidx.recyclerview.widget.LinearLayoutManager;

import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.ImageView;

import com.cooltechworks.views.shimmer.ShimmerRecyclerView;

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

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.google.firebase.storage.FirebaseStorage;

import com.google.firebase.storage.StorageReference;

import com.google.firebase.storage.UploadTask;

import com.makeramen.roundedimageview.RoundedImageView;

import com.social.media.Adapter.PostAdapter;

import com.social.media.ChatListActivity;

import com.social.media.Model.Post;

import com.social.media.Model.UserStories;

import com.social.media.R;

import com.social.media.Adapter.StoryAdapter;

import com.social.media.Model.Story;

import java.util.ArrayList;

import java.util.Date;

public class HomeFragment extends Fragment {

    //Define your views here

    //We are not using Binding here thats why we need to define our views

    ShimmerRecyclerView dashboardRV,storyRV;

    ArrayList<Story> storyList;

    ArrayList<Post> postList;

    FirebaseDatabase database;

    FirebaseStorage storage;

    FirebaseAuth auth;

    RoundedImageView addStoryImage;

    ActivityResultLauncher<String> galleryLauncher;

    ProgressDialog dialog;

    ConstraintLayout group;

    NestedScrollView scrollView;

    ImageView imgChat;

    public HomeFragment() {

        // Required empty public constructor

    }

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        dialog = new ProgressDialog(getContext());

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        View view = inflater.inflate(R.layout.fragment\_home, container, false);

        //Initializing views

        scrollView = view.findViewById(R.id.scroll);

        storyRV = view.findViewById(R.id.storyRV);

        dashboardRV = view.findViewById(R.id.dashboardRV);

        dashboardRV.showShimmerAdapter();

        storyRV.showShimmerAdapter();

        group = view.findViewById(R.id.group);

        imgChat = view.findViewById(R.id.imgChat);

        database = FirebaseDatabase.getInstance();

        auth = FirebaseAuth.getInstance();

        storage = FirebaseStorage.getInstance();

        //Loading dialog

        dialog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);

        dialog.setTitle("Story Uploading");

        dialog.setMessage("Please wait...");

        dialog.setCancelable(false);

        imgChat = view.findViewById(R.id.imgChat);

        imgChat.setOnClickListener(view1 -> {

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

            startActivity(intent);

        });

        //Story Recycler View

        storyList = new ArrayList<>();

        StoryAdapter adapter = new StoryAdapter(storyList, getContext());

        LinearLayoutManager layoutManager = new LinearLayoutManager(getContext(), LinearLayoutManager.HORIZONTAL, true);

        storyRV.setLayoutManager(layoutManager);

        storyRV.setNestedScrollingEnabled(false);

        //Fetching Story data from database and set in story recyclerview

        database.getReference()

                .child("stories").addValueEventListener(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot snapshot) {

//                if (snapshot.exists()){

                if (snapshot != null){

                    storyList.clear();

                    for (DataSnapshot storySnapshot :snapshot.getChildren()){

                        Story story = new Story();

                        story.setStoryBy(storySnapshot.getKey());

                        story.setStoryAt(storySnapshot.child("postedBy").getValue(Long.class));

                        ArrayList<UserStories> stories = new ArrayList<>();

                        for (DataSnapshot snapshot1 : storySnapshot.child("userStories").getChildren()){

                            UserStories userStories = snapshot1.getValue(UserStories.class);

                            stories.add(userStories);

                        }

                        story.setStories(stories);

                        storyList.add(story);

                    }

                    storyRV.setAdapter(adapter);

                    adapter.notifyDataSetChanged();

                    //Hide shimmer adapter when data load

                    storyRV.hideShimmerAdapter();

                    group.setVisibility(View.VISIBLE);

                }

            }

            @Override

            public void onCancelled(@NonNull DatabaseError error) {

            }

            });//End of story Recycler View

        //Post Recycler View

        postList = new ArrayList<>();

        PostAdapter postAdapter = new PostAdapter(postList, getContext(),getActivity());

        LinearLayoutManager linearLayoutManager = new LinearLayoutManager(getContext(), RecyclerView.VERTICAL, true);

        dashboardRV.setLayoutManager(linearLayoutManager);

        dashboardRV.addItemDecoration(new DividerItemDecoration(dashboardRV.getContext(), DividerItemDecoration.VERTICAL));

        dashboardRV.setNestedScrollingEnabled(false);

        //Fetching Post data from database and set in post recyclerview

        database.getReference().child("posts").addValueEventListener(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot snapshot) {

                postList.clear();

                for (DataSnapshot dataSnapshot : snapshot.getChildren()) {

                    Post post = dataSnapshot.getValue(Post.class);

                    post.setPostId(dataSnapshot.getKey());

                    postList.add(post);

                }

                dashboardRV.setAdapter(postAdapter);

                dashboardRV.hideShimmerAdapter();

                postAdapter.notifyDataSetChanged();

            }

            @Override

            public void onCancelled(@NonNull DatabaseError error) {

            }

        });//End of post Recycler View

        //Upload Story

        addStoryImage = view.findViewById(R.id.storyImg);

        addStoryImage.setOnClickListener(v -> {

            //Call gallery launcher and set input

            galleryLauncher.launch("image/\*");

        });

        //Open gallery for images

        galleryLauncher = registerForActivityResult(new ActivityResultContracts.GetContent()

                , new ActivityResultCallback<Uri>() {

                    @Override

                    public void onActivityResult(Uri result) {

                        addStoryImage.setImageURI(result);

                        //Show loading dialog as user select image

                        dialog.show();

                        //Define storage reference for story image in Firebase Storage

                        final StorageReference reference = storage.getReference()

                                .child("stories")

                                .child(FirebaseAuth.getInstance().getUid())

                                .child(new Date().getTime() + "");

                        //Store image in reference you define above

                        reference.putFile(result).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {

                            @Override

                            public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

                                //Get image URL from Storage and store a copy of image in Firebase database

                                reference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {

                                    @Override

                                    public void onSuccess(Uri uri) {

                                        Story story = new Story();

                                        story.setStoryAt(new Date().getTime());

                                        //Create a stories child in database

                                        database.getReference()

                                                .child("stories")

                                                .child(FirebaseAuth.getInstance().getUid())

                                                .child("postedBy")

                                                .setValue(story.getStoryAt()).addOnSuccessListener(new OnSuccessListener<Void>() {

                                            @Override

                                            public void onSuccess(Void unused) {

                                                //Set story image and time

                                                UserStories stories = new UserStories(uri.toString(), story.getStoryAt());

                                                //Save story data in database

                                                database.getReference()

                                                        .child("stories")

                                                        .child(FirebaseAuth.getInstance().getUid())

                                                        .child("userStories")

                                                        .push()

                                                        .setValue(stories).addOnSuccessListener(new OnSuccessListener<Void>() {

                                                    @Override

                                                    public void onSuccess(Void unused) {

                                                        dialog.dismiss();

                                                    }

                                                });

                                            }

                                        });

                                    }

                                });

                            }

                        });

                    }

                });

        return view;

    }

}

**6.1.2 LOGIN ACTIVITY**

package com.social.media;

import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;

import android.content.Intent;

import android.os.Bundle;

import android.util.Patterns;

import android.view.View;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.social.media.databinding.ActivityLoginBinding;

public class LoginActivity extends AppCompatActivity {

    ActivityLoginBinding binding;

    FirebaseAuth auth;

    FirebaseUser currentUser;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivityLoginBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        auth = FirebaseAuth.getInstance();

        currentUser = auth.getCurrentUser();

        binding.loginBtn.setOnClickListener(v -> {

            String email = binding.emailET.getText().toString();

            String password = binding.passwordET.getText().toString();

            //Sign in User with email and password

            if (email.isEmpty() || password.isEmpty()){

                Toast.makeText(LoginActivity.this, "All fields are required", Toast.LENGTH\_SHORT).show();

            }else if (!Patterns.EMAIL\_ADDRESS.matcher(email).matches()){

                Toast.makeText(LoginActivity.this, "Enter valid email address", Toast.LENGTH\_SHORT).show();

            }else

            {

                ProgressDialog progressDialog = new ProgressDialog(LoginActivity.this);

                progressDialog.setMessage("Logging..");

                progressDialog.setCancelable(false);

                progressDialog.show();

                auth.signInWithEmailAndPassword(email , password).

                        addOnCompleteListener(task -> {

                            if (task.isSuccessful()){

                                progressDialog.dismiss();

                                //Open Main Activity when user login successfully

                                Intent intent= new Intent(LoginActivity.this, MainActivity.class);

                                intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK|Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);

                                startActivity(intent);

                            }else

                            {

                                progressDialog.dismiss();

                                Toast.makeText(LoginActivity.this, task.getException().getMessage(), Toast.LENGTH\_SHORT).show();

                            }

                        });

            }

        });

        binding.goToSignup.setOnClickListener(v -> {

            //Open sign up Activity

            Intent intent = new Intent(LoginActivity.this, SignUpActivity.class);

            startActivity(intent);

        });

    }

}

**6.1.3 SIGNUP ACTIVITY**

package com.social.media;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;

import android.os.Bundle;

import android.util.Patterns;

import android.widget.Toast;

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

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

import com.google.firebase.auth.AuthResult;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.social.media.Model.User;

import com.social.media.databinding.ActivitySignUpBinding;

import java.util.Objects;

public class SignUpActivity extends AppCompatActivity {

    ActivitySignUpBinding binding;

    FirebaseAuth auth;

    DatabaseReference reference;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivitySignUpBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        auth = FirebaseAuth.getInstance();

        reference = FirebaseDatabase.getInstance().getReference().child("Users");

        binding.signUpBtn.setOnClickListener(v -> {

            String email = binding.emailET.getText().toString();

            String password = binding.passwordET.getText().toString();

            if (email.isEmpty() || password.isEmpty()){

                Toast.makeText(this, "All fields are required", Toast.LENGTH\_SHORT).show();

            }else if (!Patterns.EMAIL\_ADDRESS.matcher(email).matches()){

                Toast.makeText(this, "Enter avlid email address", Toast.LENGTH\_SHORT).show();

            }else

            {

                ProgressDialog progressDialog = new ProgressDialog(SignUpActivity.this);

                progressDialog.setMessage("Creating New Account..");

                progressDialog.setCancelable(false);

                progressDialog.show();

                auth.createUserWithEmailAndPassword(email, password)

                        .addOnCompleteListener(new OnCompleteListener<AuthResult>() {

                            @Override

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

                                if (task.isSuccessful()) {

                                    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();

                                    assert user !=null;

                                    saveData(user.getUid(),progressDialog);

                                }else

                                {

                                    progressDialog.dismiss();

                                    Toast.makeText(SignUpActivity.this, "Error", Toast.LENGTH\_SHORT).show();

                                }

                            }

                        });

            }

        });

        binding.goToLogin.setOnClickListener(v -> {

            onBackPressed();

        });

    }

    private void saveData(String uid, ProgressDialog progressDialog) {

        User user = new User();

        user.setName(binding.nameET.getText().toString());

        user.setEmail(binding.emailET.getText().toString());

        user.setPassword(binding.passwordET.getText().toString());

        user.setProfession(binding.professionET.getText().toString());

        user.setUserID(uid);

        user.setProfile("");

        user.setStatus("offline");

        user.setBio("Hello user welcome to this app. Hope you enjoy well!!");

        reference.child(uid).setValue(user).addOnCompleteListener(task -> {

            if (task.isSuccessful()){

                progressDialog.dismiss();

                Toast.makeText(this, "Account created", Toast.LENGTH\_SHORT).show();

//                Intent intent = new Intent(SignUpActivity.this,LoginActivity.class);

//                intent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK|Intent.FLAG\_ACTIVITY\_CLEAR\_TAS K);

//                startActivity(intent);

            }else

            {

                progressDialog.dismiss();

                Toast.makeText(this, "Failed to create account: "+ Objects.requireNonNull(task.getException()).getMessage(), Toast.LENGTH\_SHORT).show();

            }

        });

    }

}

**6.1.4 POST FRAGMENT**

package com.social.media.Fragment;

import android.app.ProgressDialog;

import android.content.Context;

import android.content.Intent;

import android.net.Uri;

import android.os.Bundle;

import androidx.annotation.NonNull;

import androidx.annotation.Nullable;

import androidx.core.content.ContextCompat;

import androidx.fragment.app.Fragment;

import android.text.Editable;

import android.text.TextWatcher;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.view.inputmethod.InputMethodManager;

import android.widget.Toast;

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

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.google.firebase.storage.FirebaseStorage;

import com.google.firebase.storage.StorageReference;

import com.google.firebase.storage.UploadTask;

import com.social.media.Model.Post;

import com.social.media.Model.User;

import com.social.media.R;

import com.social.media.databinding.FragmentAddPostBinding;

import com.squareup.picasso.Picasso;

import java.util.Date;

public class AddPostFragment extends Fragment {

    FragmentAddPostBinding binding;

    Uri uri;

    FirebaseAuth auth;

    FirebaseDatabase database;

    FirebaseStorage storage;

    ProgressDialog dialog;

    public AddPostFragment() {

        // Required empty public constructor

    }

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        auth = FirebaseAuth.getInstance();

        database = FirebaseDatabase.getInstance();

        storage = FirebaseStorage.getInstance();

        dialog = new ProgressDialog(getContext());

    }

    @Override

    public View onCreateView(LayoutInflater inflater, ViewGroup container,

                             Bundle savedInstanceState) {

        // Inflate the layout for this fragment

        binding = FragmentAddPostBinding.inflate(inflater, container, false);

        //Loading dialog

        dialog.setProgressStyle(ProgressDialog.STYLE\_SPINNER);

        dialog.setTitle("Post Uploading");

        dialog.setMessage("Please Wait...");

        dialog.setCancelable(false);

        dialog.setCanceledOnTouchOutside(false);

        //Get user data from database

        database.getReference().child("Users")

                .child(FirebaseAuth.getInstance().getCurrentUser().getUid()).addListenerForSingleValueEvent(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot snapshot) {

                if (snapshot.exists()){

                    User user = snapshot.getValue(User.class);

                    assert user !=null;

                    //Set user profile using Picasso

                    if (user.getProfile().equals("")){

                        binding.profileImage.setImageResource(R.drawable.original);

                    }else

                    {

                        Picasso.get().load(user.getProfile())

                                .placeholder(R.drawable.placeholder)

                                .into(binding.profileImage);

                    }

                    //Set user name and profession

                    binding.name.setText(user.getName());

                    binding.profession.setText(user.getProfession());

                }

            }

            @Override

            public void onCancelled(@NonNull DatabaseError error) {

            }

        });

        //Check if User add description or not

        binding.postDescription.addTextChangedListener(new TextWatcher() {

            @Override

            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override

            public void onTextChanged(CharSequence s, int start, int before, int count) {

                String description = binding.postDescription.getText().toString();

                if (!description.isEmpty()){

                    //Enable post Button as user start typing

                    binding.postBtn.setBackgroundDrawable(ContextCompat.getDrawable(getContext(), R.drawable.follow\_btn\_bg));

                    binding.postBtn.setTextColor(getContext().getResources().getColor(R.color.white));

                    binding.postBtn.setEnabled(true);

                }else {

                    //Disable post Button if User remove description

                    binding.postBtn.setBackgroundDrawable(ContextCompat.getDrawable(getContext(), R.drawable.follow\_active\_btn));

                    binding.postBtn.setTextColor(getContext().getResources().getColor(R.color.gray));

                    binding.postBtn.setEnabled(false);

                }

            }

            @Override

            public void afterTextChanged(Editable s) {

            }

        });

        binding.addImg.setOnClickListener(v -> {

            //Open Gallery using intent

            Intent intent = new Intent();

            intent.setAction(Intent.ACTION\_GET\_CONTENT);

            intent.setType("image/\*");

            startActivityForResult(intent, 10);

        });

        binding.postBtn.setOnClickListener(v -> {

            //Call method we created bellow for keyboard popping up automatically

            hideKeyboard();

            //Show loading dialog

            dialog.show();

            //Define storage reference for post image in Firebase Storage

            final StorageReference reference = storage.getReference().child("posts")

                    .child(FirebaseAuth.getInstance().getUid())

                    .child(new Date().getTime()+"");

            //Store image in reference you define above

            reference.putFile(uri).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {

                @Override

                public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

                    //Get image URL from Storage and store a copy of image in Firebase database

                    reference.getDownloadUrl().addOnSuccessListener(new OnSuccessListener<Uri>() {

                        @Override

                        public void onSuccess(Uri uri) {

                            //Set post data to Post object

                            Post post = new Post();

                            post.setPostImage(uri.toString());

                            post.setPostedBy(FirebaseAuth.getInstance().getUid());

                            post.setPostDescription(binding.postDescription.getText().toString());

                            post.setPostedAt(new Date().getTime());

                            //Save post data in database

                            database.getReference().child("posts")

                                    .push()

                                    .setValue(post).addOnSuccessListener(new OnSuccessListener<Void>() {

                                @Override

                                public void onSuccess(Void aVoid) {

                                    //Hide loading dialog as post uploaded

                                    dialog.dismiss();

                                    //Hide post image as post uploaded

                                    binding.postImage.setVisibility(View.GONE);

                                    binding.postDescription.setText("");

                                    Toast.makeText(getContext(), "Posted Successfully", Toast.LENGTH\_SHORT).show();

                                }

                            });

                        }

                    });

                }

            });

        });

        return binding.getRoot();

    }

    @Override

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

        super.onActivityResult(requestCode, resultCode, data);

        //Check User select any image

        if (data.getData() != null){

            uri = data.getData();

            //set post image user select from gallery

            binding.postImage.setImageURI(uri);

            binding.postImage.setVisibility(View.VISIBLE);

            //Enable post button and change background color

            binding.postBtn.setBackgroundDrawable(ContextCompat.getDrawable(getContext(), R.drawable.follow\_btn\_bg));

            binding.postBtn.setTextColor(getContext().getResources().getColor(R.color.white));

            binding.postBtn.setEnabled(true);

        }

    }

    //Stop keyboard popping up automatically

    public void hideKeyboard() {

        // Check if no view has focus:

        View view = getActivity().getCurrentFocus();

        if (view != null) {

            InputMethodManager inputManager = (InputMethodManager) getActivity().getSystemService(Context.INPUT\_METHOD\_SERVICE);

            inputManager.hideSoftInputFromWindow(view.getWindowToken(), InputMethodManager.HIDE\_NOT\_ALWAYS);

        }

    }

}

**6.1.5 CHAT ACTIVITY**

package com.social.media;

import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.recyclerview.widget.LinearLayoutManager;

import android.os.Bundle;

import android.view.View;

import android.widget.Toast;

import com.google.firebase.auth.FirebaseAuth;

import com.google.firebase.auth.FirebaseUser;

import com.google.firebase.database.DataSnapshot;

import com.google.firebase.database.DatabaseError;

import com.google.firebase.database.DatabaseReference;

import com.google.firebase.database.FirebaseDatabase;

import com.google.firebase.database.ValueEventListener;

import com.social.media.Adapter.MessageAdapter;

import com.social.media.Model.Chat;

import com.social.media.Model.User;

import com.social.media.databinding.ActivityChatBinding;

import com.squareup.picasso.Picasso;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

public class ChatActivity extends AppCompatActivity {

    ActivityChatBinding binding;

    DatabaseReference reference;

    FirebaseUser user;

    MessageAdapter messageAdapter;

    List<Chat> mchat;

    private String userid;

    ValueEventListener seenListener;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        binding = ActivityChatBinding.inflate(getLayoutInflater());

        setContentView(binding.getRoot());

        userid = getIntent().getStringExtra("userid");

        user = FirebaseAuth.getInstance().getCurrentUser();

        reference = FirebaseDatabase.getInstance().getReference();

        setSupportActionBar(binding.toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        getSupportActionBar().setTitle("");

        binding.toolbar.setNavigationOnClickListener(view -> {

            finish();

        });

        setReceiverData();

        binding.imgSend.setOnClickListener(view -> {

            String message = binding.textSend.getText().toString();

            if (message.isEmpty()){

                Toast.makeText(ChatActivity.this, "Empty can't be send!", Toast.LENGTH\_SHORT).show();

            }else

            {

                sendMessage(user.getUid(), userid, message);

            }

        });

        binding.recyclerView.setHasFixedSize(true);

        binding.recyclerView.setLayoutManager(new LinearLayoutManager(getApplicationContext()));

        seenMessage(userid);

    }

    private void seenMessage(final String userid){

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

        seenListener = databaseReference.addValueEventListener(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                for (DataSnapshot snapshot : dataSnapshot.getChildren()){

                    Chat chat = snapshot.getValue(Chat.class);

                    if (chat.getReceiver().equals(user.getUid()) && chat.getSender().equals(userid)){

                        HashMap<String, Object> hashMap = new HashMap<>();

                        hashMap.put("isseen", true);

                        snapshot.getRef().updateChildren(hashMap);

                    }

                }

            }

            @Override

            public void onCancelled(@NonNull DatabaseError databaseError) {

            }

        });

    }

    private void setReceiverData(){

        reference.child("Users").child(userid).addValueEventListener(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot snapshot) {

                if (snapshot.exists()){

                    User model = snapshot.getValue(User.class);

                    assert model !=null;

                    if (model.getProfile().equals("")){

                        binding.profileImage.setImageResource(R.drawable.original);

                    }else

                    {

                        Picasso.get().load(model.getProfile())

                                .placeholder(R.drawable.placeholder)

                                .into(binding.profileImage);

                    }

                    binding.username.setText(model.getName());

                    readMesagges(user.getUid(), userid, model.getProfile());

                }

            }

            @Override

            public void onCancelled(@NonNull DatabaseError error) {

                Toast.makeText(ChatActivity.this, error.getMessage(), Toast.LENGTH\_SHORT).show();

            }

        });

    }

    private void sendMessage(String sender, final String receiver, String message){

        String id = reference.push().getKey();

        HashMap<String, Object> hashMap = new HashMap<>();

        hashMap.put("sender", sender);

        hashMap.put("receiver", receiver);

        hashMap.put("message", message);

        hashMap.put("isseen", false);

        hashMap.put("messageId",id);

//

        reference.child("Chats").child(id).setValue(hashMap);

        binding.textSend.setText("");

//

//

        // add user to chat fragment

        final DatabaseReference chatRef = FirebaseDatabase.getInstance().getReference("Chatlist")

                .child(user.getUid())

                .child(receiver);

        chatRef.addListenerForSingleValueEvent(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                if (!dataSnapshot.exists()){

                    chatRef.child("id").setValue(userid);

                }

            }

            @Override

            public void onCancelled(@NonNull DatabaseError databaseError) {

                Toast.makeText(ChatActivity.this, databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

            }

        });

        final DatabaseReference chatRefReceiver = FirebaseDatabase.getInstance().getReference("Chatlist")

                .child(userid)

                .child(user.getUid());

        chatRefReceiver.child("id").setValue(user.getUid());

    }

    private void readMesagges(final String myid, final String userid, final String imageurl){

        mchat = new ArrayList<>();

//        reference = FirebaseDatabase.getInstance().getReference("Chats");

        reference.child("Chats").addValueEventListener(new ValueEventListener() {

            @Override

            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                mchat.clear();

                for (DataSnapshot snapshot : dataSnapshot.getChildren()){

                    Chat chat = snapshot.getValue(Chat.class);

                    assert chat !=null;

                    if (chat.getReceiver().equals(myid) && chat.getSender().equals(userid) ||

                            chat.getReceiver().equals(userid) && chat.getSender().equals(myid)){

                        mchat.add(chat);

                    }

                    messageAdapter = new MessageAdapter(ChatActivity.this, mchat, imageurl);

                    binding.recyclerView.setAdapter(messageAdapter);

                    messageAdapter.notifyDataSetChanged();

                }

            }

            @Override

            public void onCancelled(@NonNull DatabaseError databaseError) {

                Toast.makeText(ChatActivity.this, databaseError.getMessage(), Toast.LENGTH\_SHORT).show();

            }

        });

    }

    private void status(String status){

//        reference = FirebaseDatabase.getInstance().getReference("Users").child(fuser.getUid());

        HashMap<String, Object> hashMap = new HashMap<>();

        hashMap.put("status", status);

        reference.child("Users").child(user.getUid()).updateChildren(hashMap);

    }

    @Override

    protected void onResume() {

        super.onResume();

        status("online");

//        currentUser(userid);

    }

    @Override

    protected void onPause() {

        super.onPause();

        reference.removeEventListener(seenListener);

        status("offline");

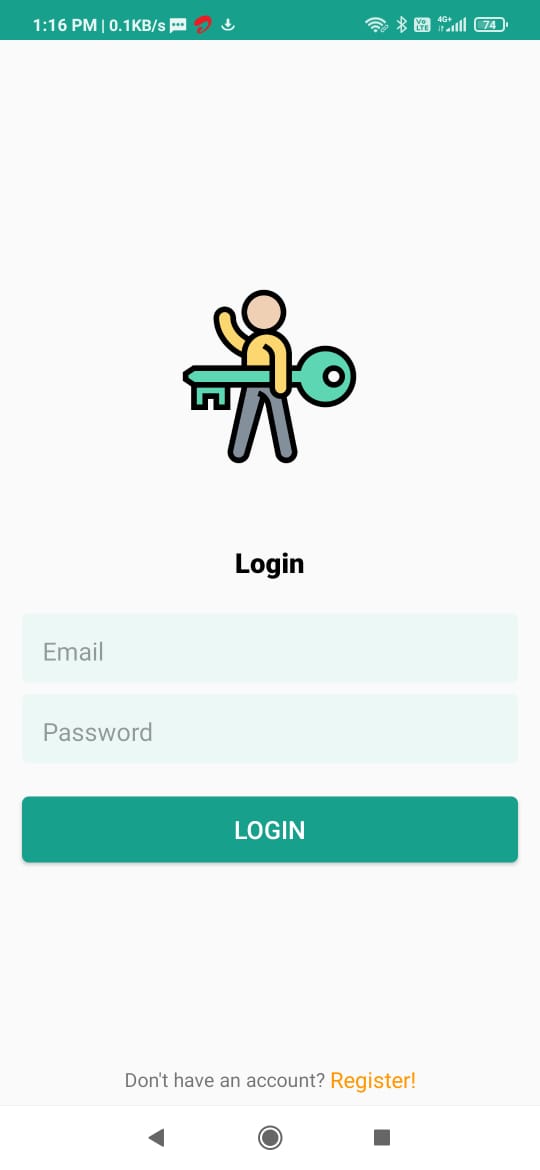
//        currentUser("none");

    }

}

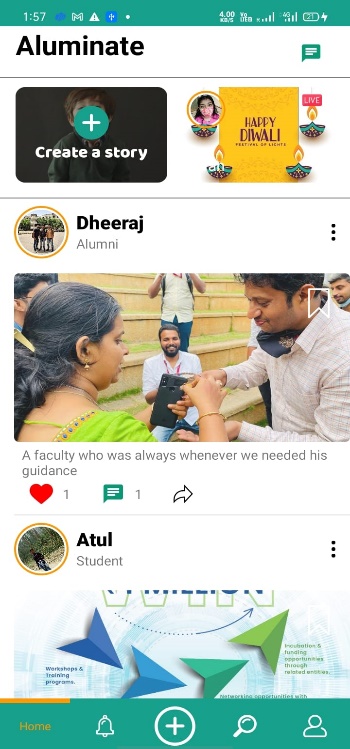
**6.2 SCREEN SHOTS**

**6.2.1 APPLICATION UI**

****

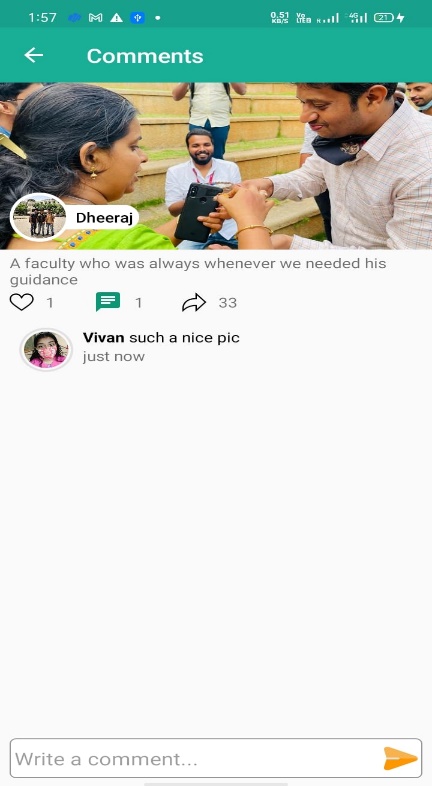
6.2.2 LOGIN PAGE with LOGIN ID

6.2.1 LOGIN PAGE



6.2.4 HOMEPAGE

6.2.3 POST

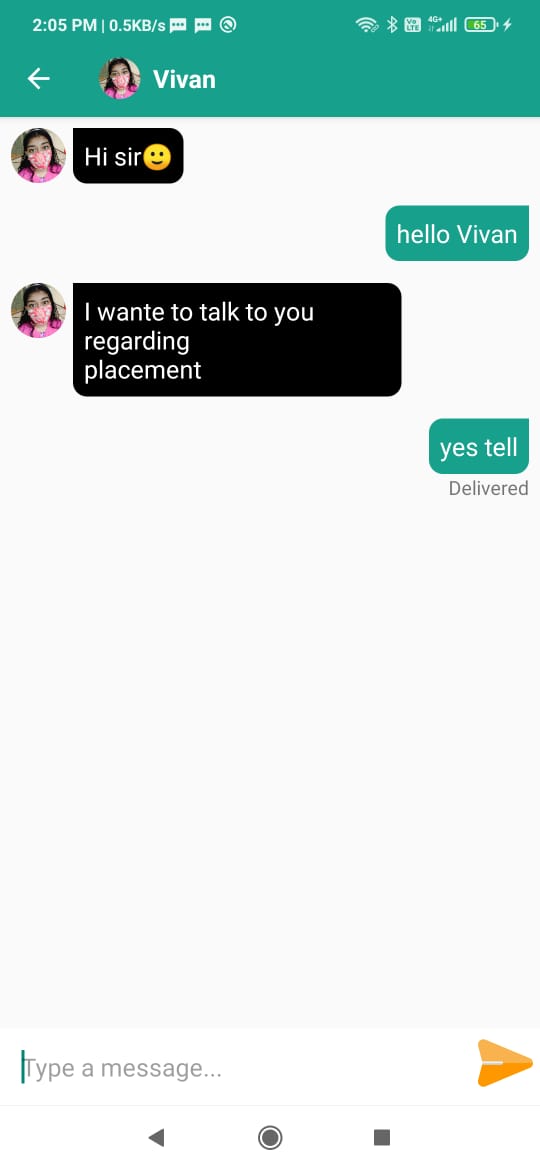
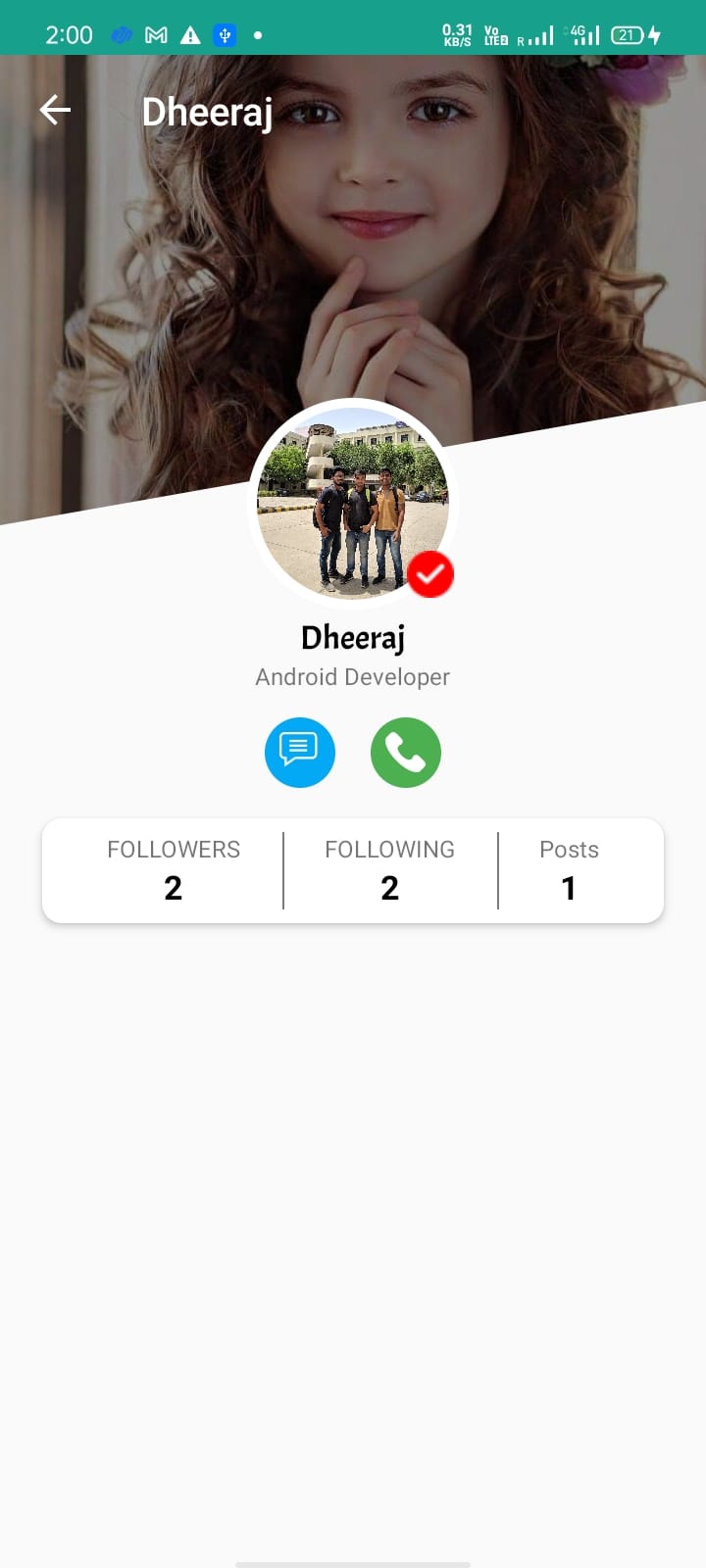


6.2.8 CHECK IF USER ONLINE

6.2.7 CHAT

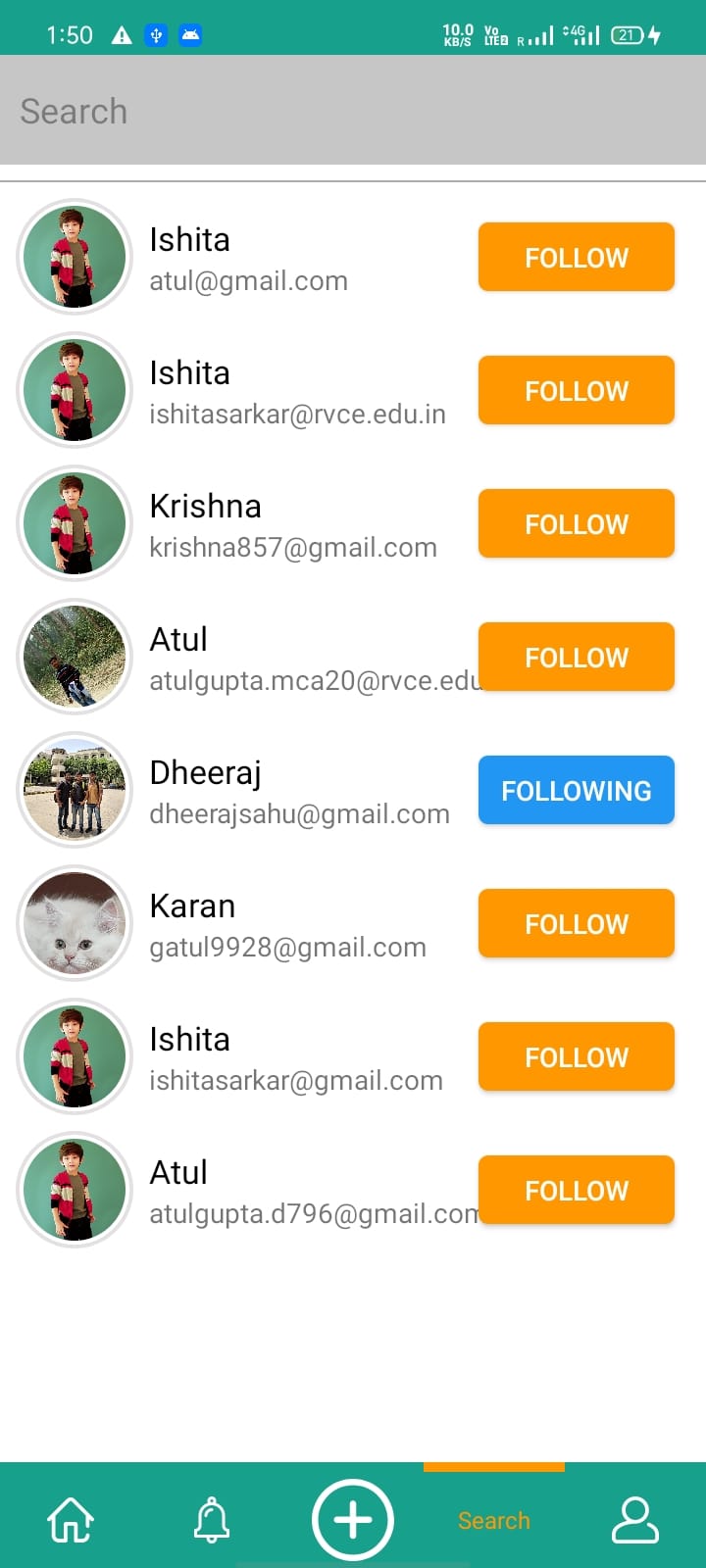
6.2.5 COMMENT

6.2.6 NOTIFICATION



6.2.9 CHAT BOX

6.2.10 PROFILE



6.2.11 FRIENDS PAGE

# Chapter 7: Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Feature Tested** | **Sample Input** | **Sample Input** | **Actual Output** | **Remarks**  **( Pass/Fail)** |
| **1** | **LOGIN MODULE** | **Email: ishitasarkar@gmail.com**  **Password : 123456789I@** |  | **If credentials are incorrect** | **PASS**  **FAIL** |

7.1 TEST CASE 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Feature Tested** | **Sample Input** | **Sample Input** | **Actual Output** | **Remarks**  **( Pass/Fail)** |
| **2** | **Post** | **THE POST ADDED BY USERS ARE SHOWING OR NOT** |  | **If text field and image field both are null.** | **PASS**  **FAIL** |

7.2 TEST CASE 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Feature Tested** | **Sample Input** | **Sample Input** | **Actual Output** | **Remarks**  **( Pass/Fail)** |
| **3** | **Session Maintenance** | **The same user who logs on the same machine will not have to log onto the same app if opened again** |  | **If the user clear application’s cache or reinstall the application.** | **PASS**  **FAIL** |

7.3 TEST CASE 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Feature Tested** | **Sample Input** | **Sample Input** | **Actual Output** | **Remarks**  **( Pass/Fail)** |
| **4** | **Sending messages** | **Whether the user is able to send other users messages or not** |  | **If the recepient has unregistered from the application.** | **PASS**  **FAIL** |

7.4 TEST CASE 4

# Chapter 8: Conclusion

The conclusion comes to be that having a secure and hassle free way to connect to our seniors or Alumni can make interactions quicker and easier where we need not to worry to search for them online on other platforms which may result in either not able to find them or else not able to connect to them due to various app policies, in such case Aluminate can be a saviour when it comes to interacting with our Alumni.

# Chapter 9: Future Enhancement

* Adding a payment gateway where students and Alumni can make contributions to the college
* To make it more dynamic by adding ML Kit to help with detecting inappropriate images being posted
* Add other dynamic actions such as audio and video calling option, sharing a post.

# Chapter 10: References

* **Alumni Database Management System** [**[1]**](https://www.ijert.org/alumni-database-management-system)

Tarun Kumar, Yeeshu Prateek, Prajwal Atharga, Dr. Rajashekarappa, Prof. V. K. Parvati,

**Paper ID :** IJERTCONV7IS10070

**Volume & Issue :**[NCRACES – 2019 (Volume 7, Issue 10)](https://www.ijert.org/ncraces-2019-volume-7-issue-10)

**Published (First Online):**22-06-2019

**ISSN (Online) :** 2278-0181

**Publisher Name :** IJERT

* **Design and Implementation of Student and Alumni Web Portal** [**[2]**](https://www.researchgate.net/publication/320119228_Design_and_Implementation_of_Student_and_Alumni_Web_Portal)

**Authors:** Shaimaa Q. Sabri,Akeela M. Ahmad,Maiwan B. Abdulrazzaq,

Science Journal of University of Zakho 5(2):272

**DOI**:10.25271/2017.5.3.395

**Received:** June 2017,

**Accepted:** August 2017,

**Published:** September 2017

* **CAs Based Student-Alumni Management System** [**[3]**](https://ieeexplore.ieee.org/document/9485017)

D.V. Nishanth; Satish; Niteesh S. Narasimha

**Published in:**[2021 International Conference on Communication, Control and Information Sciences (ICCISc)](https://ieeexplore.ieee.org/xpl/conhome/9484856/proceeding)

**Date of Conference:**16-18 June 2021,

**Date Added to IEEE *Xplore*:**20 July 2021, **DOI:**10.1109/ICCISc52257.2021.9485017,

**Publisher:**IEEE

* **Centralized Alumni Management System (CAMS) - A Prototype Proposal** [**[4]**](https://ieeexplore.ieee.org/document/8701383)

Aritra Mukherjee; Adrita Roy,

**Publisher:**IEEE,

**Date of Conference:**4-6 Feb. 2019,

**Date Added to IEEE *Xplore*:**29 April 2019,

**INSPEC Accession Number:**18635678,

**DOI:**[10.1109/AICAI.2019.8701383](https://doi.org/10.1109/AICAI.2019.8701383)

* **Alumni Info-Com with Distinct Classification of Data using Support Vector Machine Algorithm** [**[5]**](https://www.ijrte.org/wp-content/uploads/papers/v8i6/F8757038620.pdf)

**Author:** R. Sasikumar, B. Haritha, T. Borshiya Vincy, M.Kamali, S. Deva Priya,

**Published By**: Blue Eyes Intelligence Engineering & Sciences Publication,

International Journal of Recent Technology and Engineering (IJRTE)

**ISSN:** 2277-3878,

**Volume**-8 Issue-6, March 2020

* **Alumni Interaction System** [**[6]**](http://www.ijcstjournal.org/volume-5/issue-2/IJCST-V5I2P58.pdf)

**Authors:** Subashini.S , Sowndarya.A

International Journal of Computer Science Trends and Technology (IJCST) –

**Volume** 5 Issue 2, Mar – Apr 2017,

**ISSN:** 2347-8578 [www.ijcstjournal.org](http://www.ijcstjournal.org)

* **BVDUCOE College Alumni** [**[7]**](http://www.ijcstjournal.org/volume-5/issue-2/IJCST-V5I2P70.pdf)

**Authors:** Portal Rohit Singh, Rajat Singh Parmar, Saurabh Tripathi Prof. Rohini Khalkar, Prof. Sheetal Patil

International Journal of Computer Science Trends and Technology (IJCST) –

**Volume** 5 Issue 2, Mar – Apr 2017

* **Personalized expertise search at LinkedIn** [**[8]**](https://arxiv.org/abs/1602.04572)

**Authors:** Viet Ha-Thuc, [Ganesh Venkataraman](https://arxiv.org/search/cs?searchtype=author&query=Venkataraman%2C+G), [Mario Rodriguez](https://arxiv.org/search/cs?searchtype=author&query=Rodriguez%2C+M), [Shakti Sinha](https://arxiv.org/search/cs?searchtype=author&query=Sinha%2C+S), [Senthil Sundaram](https://arxiv.org/search/cs?searchtype=author&query=Sundaram%2C+S), [Lin Guo](https://arxiv.org/search/cs?searchtype=author&query=Guo%2C+L)

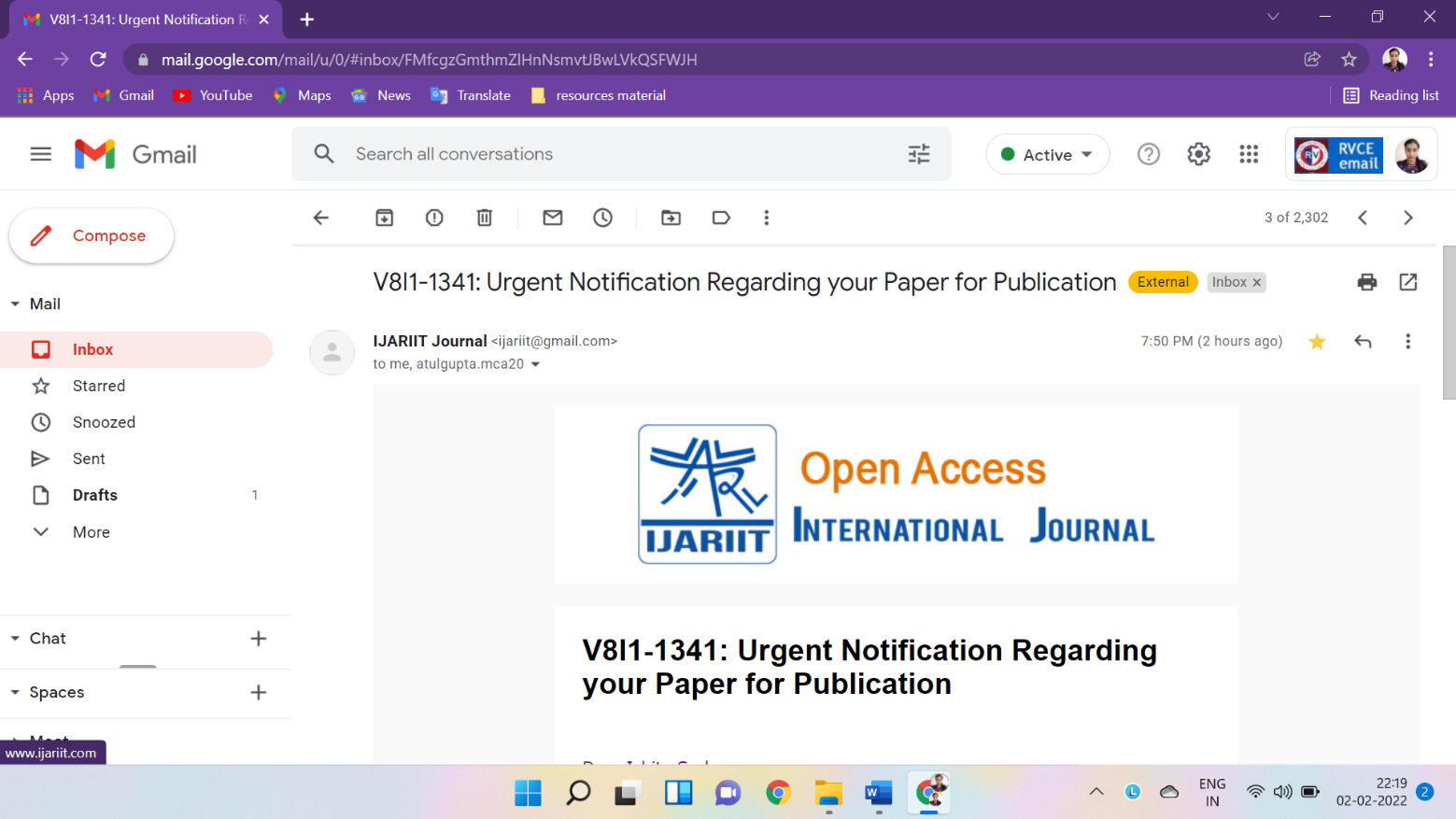
IEEE Explore,

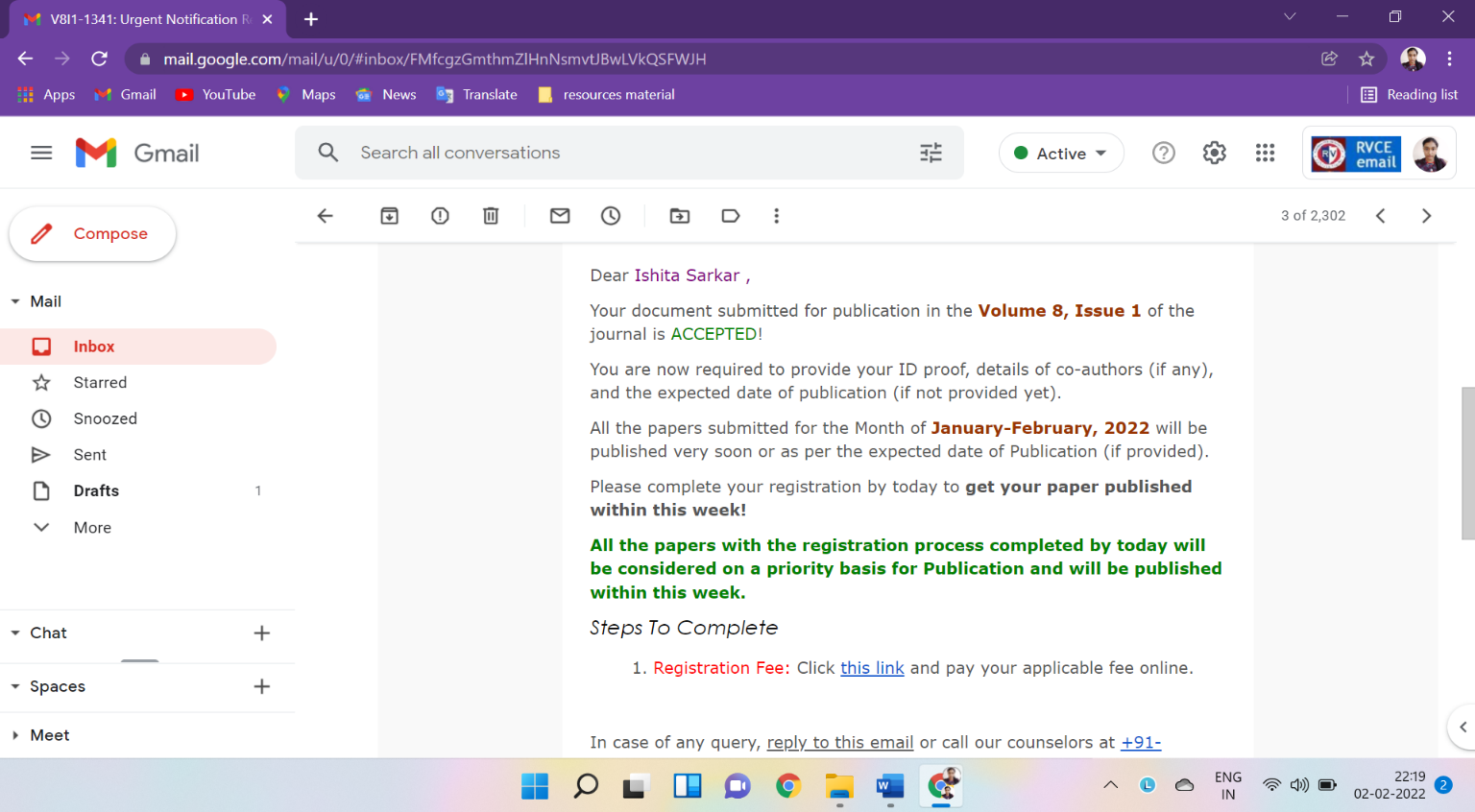
**DOI**: 10.1109/BigData.2015.7363878,

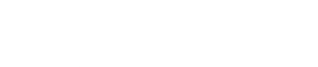
**Date of Conference:**29 Oct.-1 Nov. 2015,

**INSPEC Accession Number**: 15679815

**Paper Publication**

****

****



|

