



Instagram black book doc

Bsc. Information Technology (University of Mumbai)



Scan to open on Studocu

INSTAGRAM CLONE APPLICATION

A Project Report

Submitted in partial fulfilment of the

Requirement for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)

BY

Bhavana Singh

34

Under the esteemed guidance of

Mrs. Bimal Kumbhar

Designation



DEPARTMENT OF INFORMATION TECHNOLOGY

ABHINAV COLLEGE OF ARTS, COMMERCE AND SCIENCE

(Affiliated to University of Mumbai)

THANE – 401107

MAHARASHTRA

2022-2023

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL**PRN NO.:2020016401746312****Roll no.:34**

Name of the Student:

Bhavana Singh

Title of the Project:

Instagram Clone Application

Name of the Guide:

Mrs. Bimal Kumbhar

Teaching experience of the Guide: 10 Years

Is this your first submission?

Yes ☐No ☐

Signature of the Student

Date: _____

Signature of the Guide

Date: _____

Signature of the Coordinator:

Date:

ABHINAV COLLEGE ARTS, COMMERCE AND SCIENCE***(Affiliated to University of Mumbai)*****THANE-MAHARASHTRA -401107****CERTIFICATE**

This is to certify that the project entitled, **“Instagram Clone Application”**,
is bonafied work of **Bhavana Singh** bearing Seat.No:**34** submitted in partial fulfilment of the
requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION
TECHNOLOGY from University of Mumbai.

Internal Guide**Coordinator****External Examiner****Date:****College Seal**

ABSTRACT

A growing number of people are interacting on the web to express and share their views and knowledge of products and brands through social networks. For consumers, some social networks' profiles serve as a reference in their purchasing decision process, since these profiles are perceived as giving their personal unbiased opinion. The fitness industry has been heavily influenced by the growth of social media, generating billions of dollars in revenues worldwide, including sales of clothes, equipment, and services. Within this context, this research aims to investigate interactions at Instagram, identifying how followers consume content from the fitness profiles on this social network—focused on body cult and physical beauty—and how this content may influence them. This study is based on the view of consumption as a social activity, capable of producing meanings and identities, and uses the contributions of sociologist Bourdieu (1989, 1991) as a theoretical framework for the interpretation of the results.

ACKNOWLEDGEMENT

I am glad to present my project "Instagram clone". However, it would not have been possible without the kind support and help of my individual and organization I would like to extend sincer thanks to all them.

I take this opportunity to express my profound gratitude to management of "Abhinav College of arts commerce and science " for giving this opportunity accomplish the project work.

I'm highly indebted to Mrs Bimal Kumbhar for her guidance and constant supervision as well as providing necessary information regarding the project and also for the support in completing the project.

I, thanks to make Mr Vicky Patil head of department of our section in college for supporting us.

I'm very thankful for Mr Alvin Menzie the principal of Abhinav degree college for high Co-operative in the completion of a project.

i would like to express my gratitude towards our parents for their kind cooperation and encouragement which help me in completion of this project last but not the least we would like to thank all of her friend for support motivation and encouragement

Bhavana Singh

DECLARATION

I hereby declare that the project entitled, "Instagram Clone Application" done at place where the project is done, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Name and Signature of the Student

TABLE OF CONTENT

S.R no.	Topic Name	Page
1	Introduction	
1.1	Background	8
1.2	Objective	9
1.3	Purpose, Scope, and Applicability	9
1.3.1	Purpose	10
1.3.2	Scope	10
1.3.3	Applicability	10
2	System Analysis	
2.1	Existing System	11
2.2	Proposed System	11
2.3	Requirement Analysis	11
2.4	Hardware Requirement	12
2.5	Software Requirement	12
2.6	Justification of selection of Technology	12
3	System Design	
3.1	Modul Division	13-16
3.2	Data Dictionary	16-21
3.3	ER Diagrams	22-23
3.4	DFD/UML Diagrams	24-27
4	Implementation	
4.1	Code	27-55
4.2	Testing Approach	56
4.2.1	Unit Testing(Test cases and Test Results)	56-57
4.2.2	Integration System(Test cases and Test Result)	57
5	Result and Discussion	57-61
6	Conclusion and Future Work	61
7	Reference	61

Chapter 1

Introduction

Instagram Clone is the replica of the Instagram app, where users can share images and video content and that can visible to the audience across the globe. Instagram Clone is our pre-built social media app software that helps entrepreneurs to build an Instagram like app. Our Instagram Clone script has several unique features. Features can be added, deleted, and modified easily in the Instagram Clone Script. We provide a simple, easy & seamless user experience. And our clone script is optimized to support numerous operations in real-time, with minimal delay.

Instagram app provides various features like sharing photos, videos, branding, influencing, snap shots, shopping and even many more things are present but the very common thing is gaming, which is going to be present in the clone app. The Instagram Clone app will be running in multiple Operating System like Android, Mac, Web. And Instagram Clone app is one of the best ideas for establishing the business in which the customization will be done to make it different from the exact one.

BACKGROUOND

Instagram is all about for fun and business platform where one can start their start-up like selling cloth's, cosmetic's, jewellery's, study materials, some share their travel experience, food blogs, study blogs, some influencer motivate other user and to achieve success in life. Talking about fun part of Instagram, User upload their daily life event in Instagram Clone via sharing photos and videos and snapshots. When Users are free or bored they watch memes and Reels to relax themselves and to consume their time by that. But sometimes user get bored by watching reels and memes, so the newest feature is gaming in Instagram. Instagram gaming feature has many games which will build logic and concentration of user. It will be fun to playing games in Instagram by sharing their scores and competing with their friends or family member. Instagram also provide messaging and video call to connect people from far away. It provide may employability.

As Instagram help us to know what kind of interest person is having and in which field a person can build their carrier. Instagram is a platform where a person can have detail of anything from scratch, and get chance to connect with people from anywhere in the world, and get the perfect guidance form the person who is in the specific field.

Objective

The main purpose of Instagram clone is to share their creative ideas of games. Now a days gaming has become the biggest part of human life because gaming gave us platform to earn money by developing and playing game, it provides huge championship or battlefield to many players and begun their carrier.

In this Clone Application, we will get chance to challenge many players with whom we are connected or and many champions from all over the world. And we can our achievement through post it as a post or in our feeds. Which will help others to learn how to play the game or challenge someone.

- Enhances memory, brain's speed and concentration.
- Improved multi-tasking skills.
- Build skills for future careers.

PURPOSE, SCOPE AND APPLILCABILTIY:

PURPOSE:

The purpose of this Application is it will connect more people together, it will increase creativity power, and to improve strategical thinking. It can also help young players build perseverance to achieve goals, build resilience and improve their communications skills, especially in online multiplayer video games. When two people will connect with each other while playing games they might get some new ideas which will help them to develop new application.

SCOPES:

The following use-cases are in scope:

1. Users should be able to upload their photos and videos.
2. Users should be able to like a post posted by other users.
3. Users should be able to follow other users.
4. Users should be able to view NewsFeed from other users they are following.

Not in Scope

The following use-cases are out scope:

1. Push Notification to users for updates.
2. Users are able to search Post across the platform.
3. Photo Filters, Post Comments & Location-based tagging.
4. User stories automatically expiring at 24 hours.
5. User messaging to an individual user or a group.

Chapter 2

System Analysis

Flutter: Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, Google Fuchsia, and the web from a single codebase

The major components of Flutter include:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

Dart Language

Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

While writing and debugging an application, Flutter runs in the Dart virtual machine, which features a just-in-time execution engine. This allows for fast compilation times as well as "hot reload", with which modifications to source files can be injected into a running application. Flutter extends this further with support for stateful hot reload, where in most cases changes to source code are reflected immediately in the running app without requiring a restart or any loss of state.

For better performance, release versions of Flutter apps on all platforms use ahead-of-time (AOT) compilation,[19] except for on the Web where code is transpiled to JavaScript.

Flutter inherits Dart's Pub package manager and software repository, which allows users to publish and use custom packages as well as Flutter-specific plugins.

Flutter engine

Flutter's engine, written primarily in C++, provides low-level rendering support using either Google's Skia graphics library or the custom "Impeller" graphics layer.[16] Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS to implement accessibility, file and network I/O, native plugin support, and more.

Foundation library

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine.

Design-specific widgets

The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same name, and Cupertino widgets implement Apple's iOS Human interface guidelines

Visual Studio Code:

Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS.[10] Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

In the Stack Overflow 2021 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 82,000 respondents, with 70% reporting that they use it.

Firebase










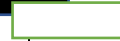


Firebase is a set of hosting services for any type of application (Android, iOS, Javascript, Node.js, Java, Unity, PHP, C++ ...). It offers NoSQL and real-time hosting of databases, content, social authentication (Google,

Facebook, Twitter and Github), and notifications, or services, such as a real-time communication server.

2.4 Planning and Scheduling :

Sr.no	Activity Phases	Planned		Acutal		Sign
		Start Date	End Date	Start Date	End Date	
1	Preliminary Investigation	28/6/2022	20/7/2022	5/7/2022	10/8/2022	
2	System Analysis	15//8/2022	5/9/2022	22/8/2022	12/9/2022	
3	System Design	17/9/2022	20/10/2022	28/9/2022	15/10/2022	
4	System Coding	27/10/2022	18/12/2022	20/10/2022	12/1/2023	
5	System Testing	16/1/2023	12/2/2023	22/1/2023	24/2/2023	
6	System Implementation	1/3/2023	20/3/2023	1/3/2023	18/3/2023	

Gantt Chart -Planned -Actual

Activity	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	
Preliminary Investigation											
System Analysis											
System Design											
System Coding											
System Testing											
System Implementation											

Chapter 3

System Design

Problem of Definition

The main problem people facing through this project is not every one has IOS and Android Software. The problem this project is most spend most time in Instagram just to time pass. The popular photo sharing app negatively impacts body image and sleep, increases bullying and “FOMO”(Fear of Missing out), and leads to greater loneliness. People are more to show things which are not needed.

Requirement Specification

The Specific Requirement are classified into two groups:

- Functional
- Non-Functional

a. Functional Requirement

A functional requirement defines the function of a system or it's components.

Create Account:

The user can simply sign in using Facebook account or create a new account by providing required personal information about himself/herself. After creating the account it will display account create successfully.

Searching Friend:

Search for those people whom the user wants to follow.

Sending Follow Request:

After finding the people the user can send him or her the follow request and others can also send follow request to the user.

Accepting Follow Request:

After sending follow request if they accept the follow request and other able to see, like, share, comment on their posts.

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 any other text posts.

Like, Comment, Share:

The users can like each others posts, they can write something in the comment section, they can even share others posts.

b Non-Functional Requirement

Security:

The system use SSL(secured socket layer) in all transactions that includes any other confidential passenger information. The system should such 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:

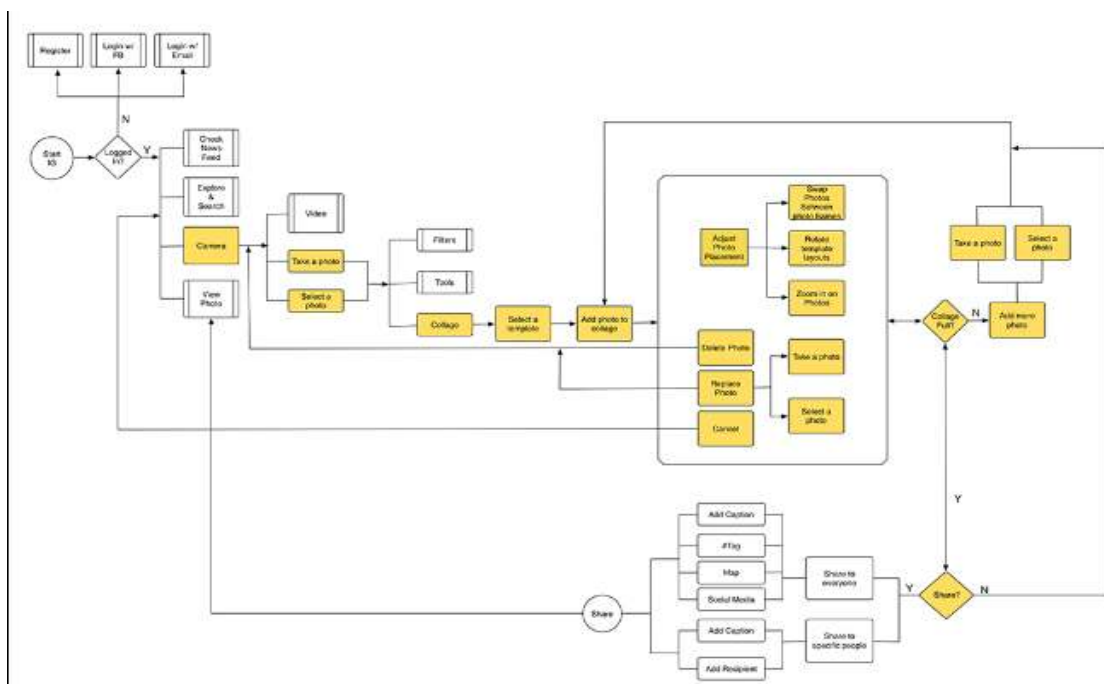
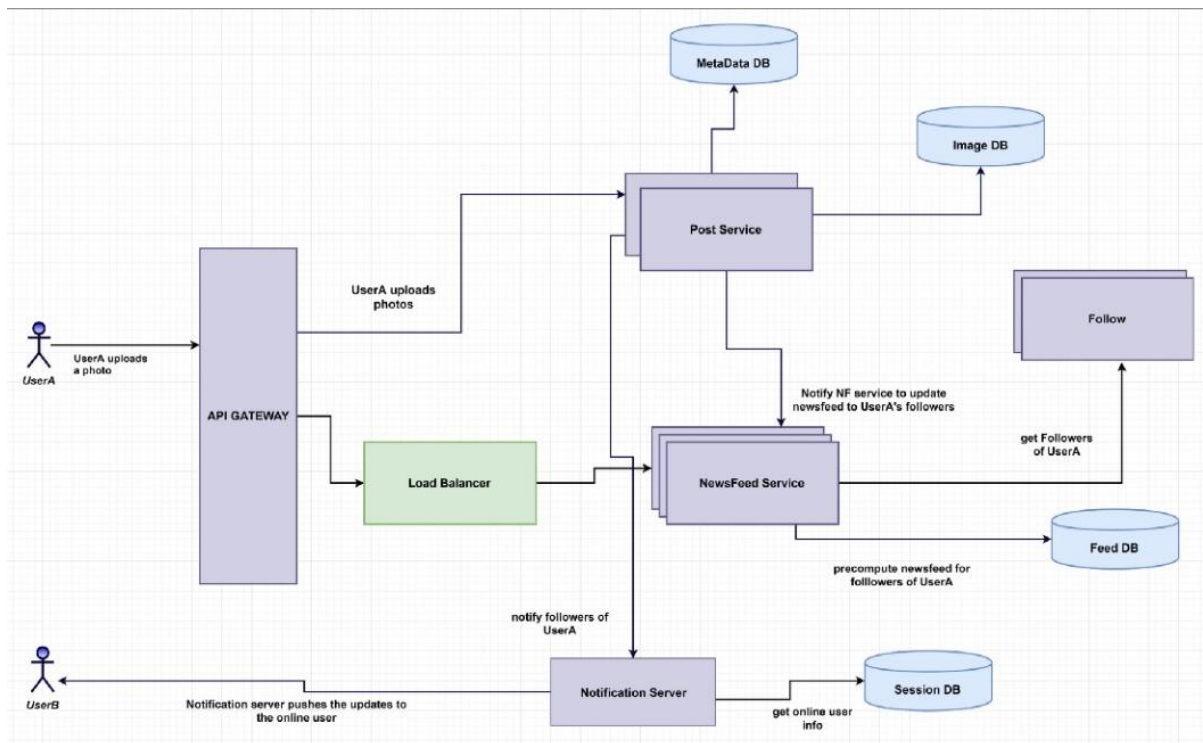
The system provides database for storage for all kinds of device whether it is a computer or mobile or something else. The reliability of the whole system depends on the reliable that it should not crash or hang during the use by the user.

Maintainability:

There are many people those are there for the maintenance purpose of the system. They can be software engineers or team of hackers. They are their to take care of that if there is some problems regarding the system or not.

Portability:

The system consist of scripting languages such as PHP,HTML etc. It should run on any device and any platform or in any operating system whether it is windows, IOs or android.



Hardware and Software Requirement:

Hardware:

1. 3 GB RAM but **8GB** recommended.

2. **2GB** HDD space but **4GB** recommended.
3. Windows, Linux or Mac all platforms.
4. Monitor resolution needs to be **1280x800** minimum.

Software:

- 1.Flutter
- 2.Visual Studio
- 3.Android Studio

Language:

Front-End Languages:

Html, CSS, JavaScript

Back-End Languages:

Dart, NativeJS, Dart ,Python

Server:

Firebase, Firebase Storage, Firebase Authentication, Firebase Framework.

Preliminary Product Description:

The aim of developing this clone project is to adding new feature to the Instagram application and test how it is working and how much people will love it and use it, before adding any new feature to real Instagram application, the developer who brings new such ideas like this first test this by making its clone application.

We can play games will using this social media, and we can share our scores by friends by posting it on stories where more friends can play together and share new creative ideas. We can also get to from where the person is locating.

Conceptual Models:

Agile Model



Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In Agile, the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

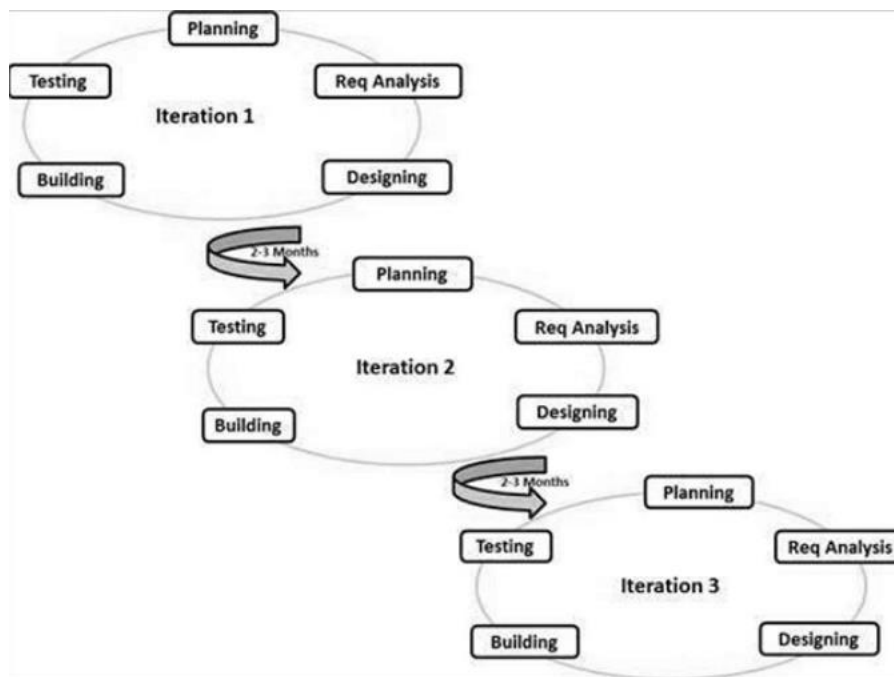
The Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

The most popular Agile methods include Rational Unified Process (1994), Scrum (1995), Crystal Clear, Extreme Programming (1996), Adaptive Software Development, Feature Driven Development, and Dynamic Systems Development Method (DSDM) (1995). These are now collectively referred to as **Agile Methodologies**, after the Agile Manifesto was published in 2001.

Following are the Agile Manifesto principles –

- **Individuals and interactions** – In Agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

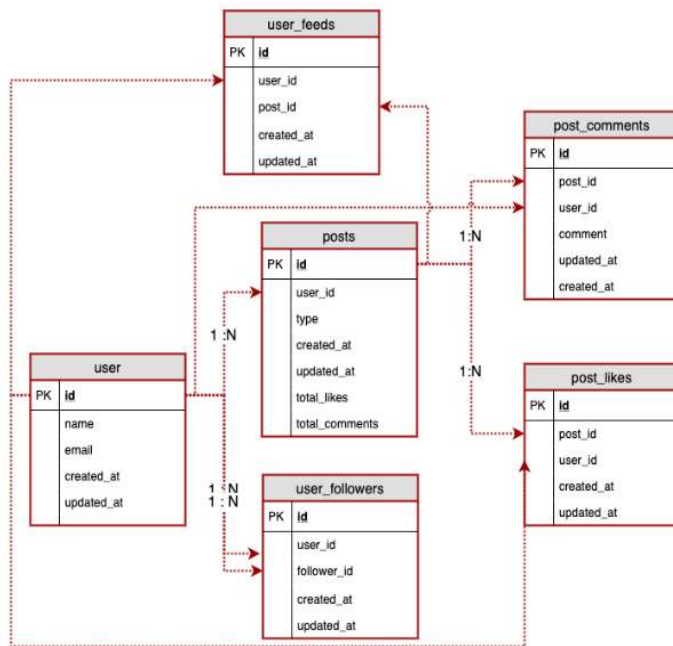
- **Working software** – Demo working software is considered the best means of communication with the customers to understand their requirements, instead of just depending on documentation.
- **Customer collaboration** – As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements.
- **Responding to change** – Agile Development is focused on quick responses to change and continuous development.



Basic Modules:

There are various modules in this project we are going to add such as

- Sign-Up Page
- Login Page
- Home Page
- Profile Page
- Search Page
- Update Page
- **Data Design:**



Schema Design:

Table Photo:

Field	Datatype	Key
photo_id:	(Integer)	Primary ID that preferably auto increments (if supported in chosen DB)
user_id:	(Integer)	ID of the user who owns this photo (Indexed field)
caption:	(String)	Photo caption
latitude:	(Float)	Latitude value for location
longitude:	(Float)	Longitude value location
image_path:	(String)	Path to image on server
image_size:	(Integer)	Image size on server
date_created:	(Unix Timestamp or DateTime)	When was this image created?
date_updated:	(Unix Timestamp)	Last time this image was updated?

	or DateTime)	
--	-----------------	--

Tabel Comment:

Field	Datatype	Key
comment_id:	(Integer)	Primary ID that preferably auto increments (if supported in chosen DB)
comment:	(Text)	a simple text field containing the comment

Tabel Photo_Comment:

Field	Datatype	Key
photo_id:	(Integer)	ID of the photo
comment_id:	(Integer)	ID of the comment being assigned to a photo

Table: hashtags

Field	Datatype	Key
hashtag_id:	(Integer)	Primary ID that preferably auto increments (if supported in chosen DB)
hashtag:	(Text)	a simple text field containing the hashtag

Table photos_hashtags

Field	Datatype	Key
photo_id:	(Integer)	ID of the photo
hashtag_id:	(Integer)	ID of the hashtag being assigned to a photo

Table: likes

Field	Datatype	Key
user_id:	(Integer)	ID of the user performing the like (Indexed field)
photo_id:	(Integer)	ID of the photo being liked (Indexed field)

	date_created: (Unix Timestamp or DateTime)	When was this image created?
	date_updated: (Unix Timestamp or DateTime)	Last time this image was updated?

Table: tags

Field	Datatype	Key
id:	(Integer)	Primary ID that preferably auto increments (if supported in chosen DB)
name:	(String)	Tag name

Table: photos_tags

Field	Data Type	Key
photo_id:	(Integer)	ID of the photo being tagged
tag_id:	(Integer)	ID of the tag being assigned to a photo

Table: users

Field	Data Type	Key
user_id:	(Integer)	Primary ID that preferably auto increments (if supported in chosen DB)
username:	(String)	Username (Unique Index)
email:	(String)	Email address (Unique Index)
salted_password:	(String)	Salted password digest
first_name:	(String)	First name of user
last_name:	(String)	Last name of user
last_ip:	(String)	Last known user IP address

date_created:	(Unix Timestamp or DateTime)	When did this user sign up?
date_updated:	(Unix Timestamp or DateTime)	Last time this user was updated?

Data Integrity and Constrain:

Data integrity is the overall accuracy, completeness, and consistency of data. Data integrity also refers to the safety of data in regard to regulatory compliance — such as GDPR compliance — and security. It is maintained by a collection of processes, rules, and standards implemented during the design phase.

Types of data integrity

Maintaining data integrity requires an understanding of the two types of data integrity: physical integrity and logical integrity. Both are collections of processes and methods that enforce data integrity in both hierarchical and relational databases.

Physical integrity

Physical integrity is the protection of the wholeness and accuracy of that data as it's stored and retrieved. When natural disasters strike, power goes out, or hackers disrupt database functions, physical integrity is compromised. Human error, storage erosion, and a host of other issues can also make it impossible for data processing managers, system programmers, applications programmers, and internal auditors to obtain accurate data.

Logical integrity

Logical integrity keeps data unchanged as it's used in different ways in a relational database. Logical integrity protects data from human error and hackers as well, but in a much different way than physical integrity does. There are four types of logical integrity:

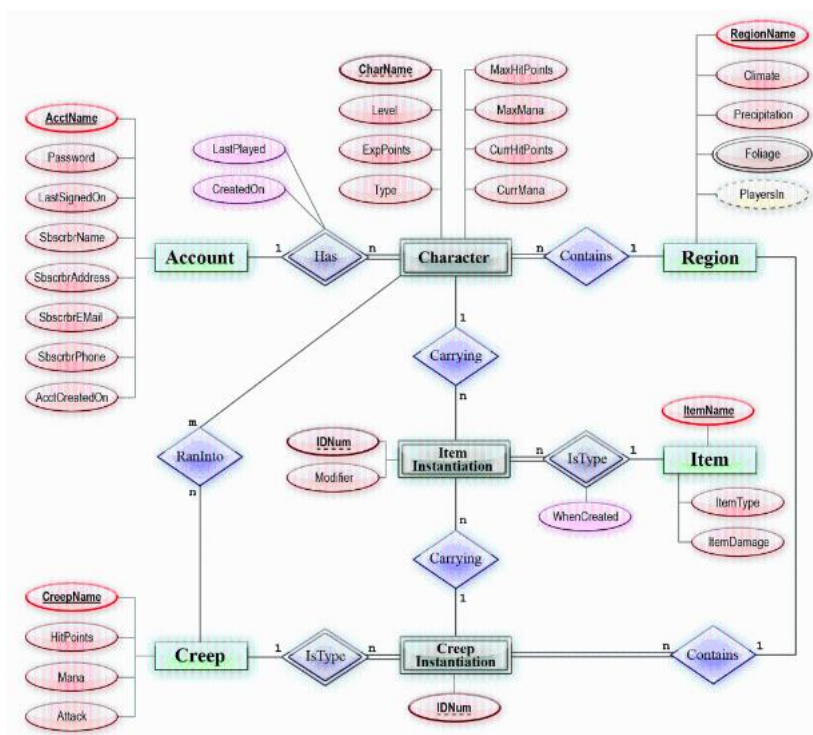
Entity integrity: Entity integrity relies on the creation of primary keys — the unique values that identify pieces of data — to ensure that data isn't listed more than once and that no field in a table is null. It's a feature of relational systems which store data in tables that can be linked and used in a variety of ways.

Referential integrity: Referential integrity refers to the series of processes that make sure data is stored and used uniformly. Rules embedded into the database's structure about how foreign keys are used ensure that only appropriate changes, additions, or deletions of data occur. Rules may include constraints that eliminate the entry of duplicate data, guarantee that data entry is accurate, and/or disallow the entry of data that doesn't apply.

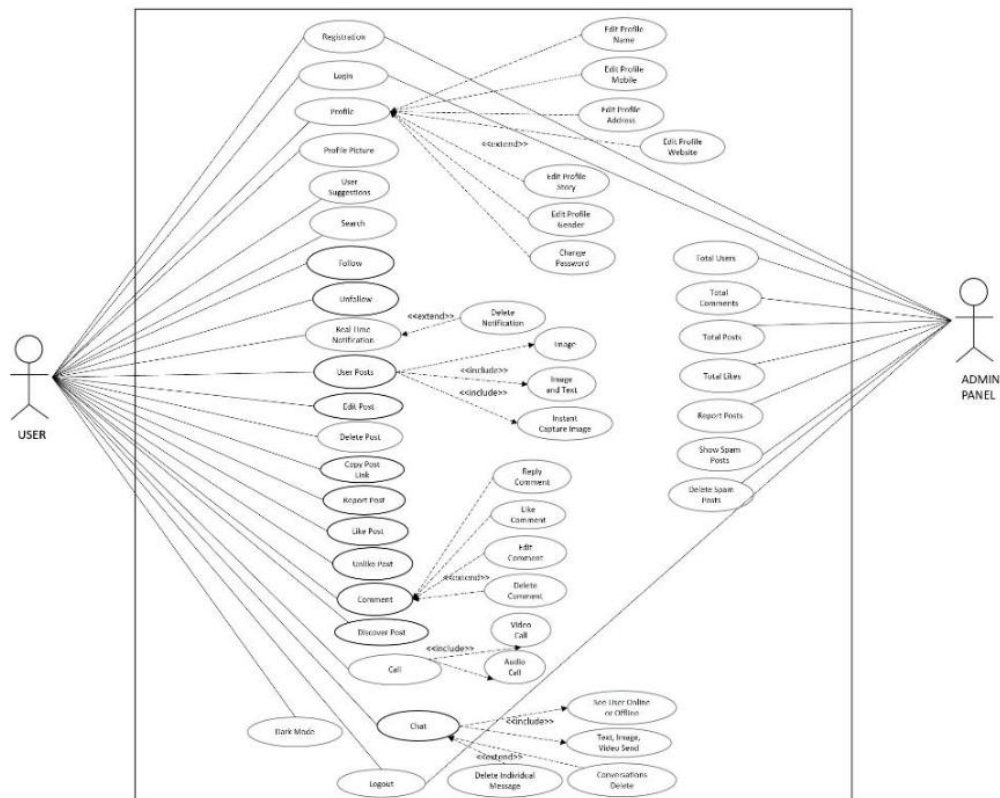
Domain integrity: Domain integrity is the collection of processes that ensure the accuracy of each piece of data in a domain. In this context, a domain is a set of acceptable values that a column is allowed to contain. It can include constraints and other measures that limit the format, type, and amount of data entered.

User-defined integrity: User-defined integrity involves the rules and constraints created by the user to fit their particular needs. Sometimes entity, referential, and domain integrity aren't enough to safeguard data. Often, specific business rules must be taken into account and incorporated into data integrity measures.

4.3 Procedural Design



ER Diagram:



Security Issues:

I have used commercially reasonable safeguards to help secure a user's information collected through the service. I have also state that Instagram

takes reasonable steps to verify a user's identity before granting them access to an account. However, the policies do not clearly indicate whether or not a user's data is encrypted while in transit or at rest. And also do not disclose how they would notify users in the event of a data breach.

Chapter 4

Implementation And Testing

Code:

Main.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:instagram_clone/providers/user_provider.dart';
import
'package:instagram_clone/responsive/mobile_screen_layout.dart'
;
import
'package:instagram_clone/responsive/responsive_layout.dart';
import
'package:instagram_clone/responsive/web_screen_layout.dart';
import 'package:instagram_clone/screens/login_screen.dart';
import 'package:instagram_clone/utils/colors.dart';
import 'package:provider/provider.dart';

void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  // initialise app based on platform- web or mobile
  if (kIsWeb) {
    await Firebase.initializeApp(
      options: const FirebaseOptions(
        apiKey: "AIzaSyCSQekh2a09pI_7-XEYaTOjIjbA4tRb8EI",

        projectId: "instagram-6cd6f",
        storageBucket: "instagram-6cd6f.appspot.com",
        messagingSenderId: "507327205725",
        appId: "1:507327205725:web:ca2c1b3e817695a03318b4"
      ),
    );
  } else {
    await Firebase.initializeApp();
  }
  runApp(const MyApp());
}
```

```

}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(create: (_) =>
UserProvider(),),
      ],
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
        title: 'Instagram Clone',
        theme: ThemeData.dark().copyWith(
          scaffoldBackgroundColor: mobileBackgroundColor,
        ),
        home: StreamBuilder(
          stream: FirebaseAuth.instance.authStateChanges(),
          builder: (context, snapshot) {
            if (snapshot.connectionState ==
ConnectionState.active) {
              // Checking if the snapshot has any data or not
              if (snapshot.hasData) {
                // if snapshot has data which means user is
logged in then we check the width of screen and accordingly
display the screen layout
                return const ResponsiveLayout(
                  mobileScreenLayout: MobileScreenLayout(),
                  webScreenLayout: WebScreenLayout(),
                );
              } else if (snapshot.hasError) {
                return Center(
                  child: Text('${snapshot.error}'),
                );
              }
            }

            // means connection to future hasnt been made yet
            if (snapshot.connectionState ==
ConnectionState.waiting) {

```

```

        return const Center(
          child: CircularProgressIndicator(),
        );
      },
    ),
  ),
);
}
}

```

mobile_screen.dart

```

import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:instagram_clone/utils/colors.dart';
import 'package:instagram_clone/utils/global_variable.dart';

class MobileScreenLayout extends StatefulWidget {
  const MobileScreenLayout({Key? key}) : super(key: key);

  @override
  State<MobileScreenLayout> createState() =>
    _MobileScreenLayoutState();
}

class _MobileScreenLayoutState extends
State<MobileScreenLayout> {
  int _page = 0;
  late PageController pageController; // for tabs animation

  @override
  void initState() {
    super.initState();
    pageController = PageController();
  }

  @override
  void dispose() {
    super.dispose();
  }
}

```

```

    pageController.dispose();
}

void onPageChanged(int page) {
  setState(() {
    _page = page;
  });
}

void navigationTapped(int page) {
  //Animating Page
  pageController.jumpToPage(page);
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    body: PageView(
      children: homeScreenItems,
      controller: pageController,
      onPageChanged: onPageChanged,
    ),
    bottomNavigationBar: CupertinoTabBar(
      backgroundColor: mobileBackgroundColor,
      items: <BottomNavigationBarItem>[
        BottomNavigationBarItem(
          icon: Icon(
            Icons.home,
            color: (_page == 0) ? primaryColor :
secondaryColor,
          ),
          label: '',
          backgroundColor: primaryColor,
        ),
        BottomNavigationBarItem(
          icon: Icon(
            Icons.search,
            color: (_page == 1) ? primaryColor :
secondaryColor,
          ),
          label: '',
          backgroundColor: primaryColor),

```

```

        BottomNavigationBarItem(
          icon: Icon(
            Icons.add_circle,
            color: (_page == 2) ? primaryColor :
secondaryColor,
          ),
          label: '',
          backgroundColor: primaryColor),
        BottomNavigationBarItem(
          icon: Icon(
            Icons.favorite,
            color: (_page == 3) ? primaryColor :
secondaryColor,
          ),
          label: '',
          backgroundColor: primaryColor,
        ),
        BottomNavigationBarItem(
          icon: Icon(
            Icons.person,
            color: (_page == 4) ? primaryColor :
secondaryColor,
          ),
          label: '',
          backgroundColor: primaryColor,
        ),
      ],
      onTap: navigationTapped,
      currentIndex: _page,
    ),
  );
}
}

```

web_screen_layout.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:instagram_clone/utils/colors.dart';
import 'package:instagram_clone/utils/global_variable.dart';

class WebScreenLayout extends StatefulWidget {

```



```

const WebScreenLayout({Key? key}) : super(key: key);

@override
State<WebScreenLayout> createState() =>
  _WebScreenLayoutState();
}

class _WebScreenLayoutState extends State<WebScreenLayout> {
  int _page = 0;
  late PageController pageController; // for tabs animation

  @override
  void initState() {
    super.initState();
    pageController = PageController();
  }

  @override
  void dispose() {
    super.dispose();
    pageController.dispose();
  }

  void onPageChanged(int page) {
    setState(() {
      _page = page;
    });
  }

  void navigationTapped(int page) {
    //Animating Page
    pageController.jumpToPage(page);
    setState(() {
      _page = page;
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        backgroundColor: mobileBackgroundColor,

```

```

centerTitle: false,
title: SvgPicture.asset(
  'assets/ic_instagram.svg',
  color: primaryColor,
  height: 32,
),
actions: [
  IconButton(
    icon: Icon(
      Icons.home,
      color: _page == 0 ? primaryColor :
secondaryColor,
    ),
    onPressed: () => navigationTapped(0),
  ),
  IconButton(
    icon: Icon(
      Icons.search,
      color: _page == 1 ? primaryColor :
secondaryColor,
    ),
    onPressed: () => navigationTapped(1),
  ),
  IconButton(
    icon: Icon(
      Icons.add_a_photo,
      color: _page == 2 ? primaryColor :
secondaryColor,
    ),
    onPressed: () => navigationTapped(2),
  ),
  IconButton(
    icon: Icon(
      Icons.favorite,
      color: _page == 3 ? primaryColor :
secondaryColor,
    ),
    onPressed: () => navigationTapped(3),
  ),
  IconButton(
    icon: Icon(
      Icons.person,

```

```

        color: _page == 4 ? primaryColor :
secondaryColor,
      ),
      onPressed: () => navigationTapped(4),
    ),
  ],
),
body: PageView(
  physics: const NeverScrollableScrollPhysics(),
  children: homeScreenItems,
  controller: pageController,
  onPageChanged: onPageChanged,
),
);
}
}

```

login_screen.dart

```

import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:instagram_clone/resources/auth_methods.dart';
import
'package:instagram_clone/responsive/mobile_screen_layout.dart'
;
import
'package:instagram_clone/responsive/responsive_layout.dart';
import
'package:instagram_clone/responsive/web_screen_layout.dart';
import 'package:instagram_clone/screens/signup_screen.dart';
import 'package:instagram_clone/utils/colors.dart';
import 'package:instagram_clone/utils/global_variable.dart';
import 'package:instagram_clone/utils/utils.dart';
import
'package:instagram_clone/widgets/text_filed_input.dart';

class LoginScreen extends StatefulWidget {
  const LoginScreen({Key? key}) : super(key: key);

  @override
  _LoginScreenState createState() => _LoginScreenState();
}

```

```

class _LoginScreenState extends State<LoginScreen> {
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  bool _isLoading = false;

  @override
  void dispose() {
    super.dispose();
    _emailController.dispose();
    _passwordController.dispose();
  }

  void loginUser() async {
    setState(() {
      _isLoading = true;
    });
    String res = await AuthMethods().loginUser(
      email: _emailController.text, password:
_passwordController.text);
    if (res == 'success') {
      Navigator.of(context).pushAndRemoveUntil(
        MaterialPageRoute(
          builder: (context) => const ResponsiveLayout(
            mobileScreenLayout: MobileScreenLayout(),
            webScreenLayout: WebScreenLayout(),
          ),
        ),
        (route) => false);

      setState(() {
        _isLoading = false;
      });
    } else {
      setState(() {
        _isLoading = false;
      });
      showSnackBar(context, res);
    }
  }
}

```

```

@override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    body: SafeArea(
      child: Container(
        padding: MediaQuery.of(context).size.width >
webScreenSize
          ? EdgeInsets.symmetric(
              horizontal:
MediaQuery.of(context).size.width / 3)
            : const EdgeInsets.symmetric(horizontal: 32),
        width: double.infinity,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          children: [
            Flexible(
              child: Container(),
              flex: 2,
            ),
            SvgPicture.asset(
              'assets/ic_instagram.svg',
              color: primaryColor,
              height: 64,
            ),
            const SizedBox(
              height: 64,
            ),
            TextFieldInput(
              hintText: 'Enter your email',
              textInputType: TextInputType.emailAddress,
              textEditingController: _emailController,
            ),
            const SizedBox(
              height: 24,
            ),
            TextFieldInput(
              hintText: 'Enter your password',
              textInputType: TextInputType.text,
              textEditingController: _passwordController,
              isPass: true,
            ),
          ],
        ),
      ),
    ),
  );
}

```

```

const SizedBox(
  height: 24,
),
InkWell(
  child: Container(
    child: !_isLoading
      ? const Text(
          'Log in',
        )
      : const CircularProgressIndicator(
          color: primaryColor,
        ),
    width: double.infinity,
    alignment: Alignment.center,
    padding: const
EdgeInsets.symmetric(vertical: 12),
    decoration: const ShapeDecoration(
      shape: RoundedRectangleBorder(
        borderRadius:
BorderRadius.all(Radius.circular(4)),
      ),
      color: blueColor,
    ),
  ),
  onTap: loginUser,
),
const SizedBox(
  height: 12,
),
Flexible(
  child: Container(),
  flex: 2,
),
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
    Container(
      child: const Text(
        'Dont have an account?',
      ),
      padding: const
EdgeInsets.symmetric(vertical: 8),

```

```

    ),
    GestureDetector(
      onTap: () => Navigator.of(context).push(
        MaterialPageRoute(
          builder: (context) => const
SignupScreen(),
        ),
      ),
    child: Container(
      child: const Text(
        ' Signup.',
        style: TextStyle(
          fontWeight: FontWeight.bold,
        ),
      ),
      padding: const
EdgeInsets.symmetric(vertical: 8),
    ),
  ],
),
],
),
),
),
);
}
}

```

signup_screen.dart

```

import 'dart:typed_data';

import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:image_picker/image_picker.dart';
import 'package:instagram_clone/resources/auth_methods.dart';
import
'package:instagram_clone/responsive/mobile_screen_layout.dart'
;
import
'package:instagram_clone/responsive/responsive_layout.dart';

```

```

import
'package:instagram_clone/responsive/web_screen_layout.dart';
import 'package:instagram_clone/screens/login_screen.dart';
import 'package:instagram_clone/utils/colors.dart';
import 'package:instagram_clone/utils/utils.dart';
import
'package:instagram_clone/widgets/text_file_input.dart';

class SignupScreen extends StatefulWidget {
  const SignupScreen({Key? key}) : super(key: key);

  @override
  _SignupScreenState createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  final TextEditingController _usernameController =
TextEditingController();
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  final TextEditingController _bioController =
TextEditingController();
  bool _isLoading = false;
  Uint8List? _image;

  @override
  void dispose() {
    super.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    _usernameController.dispose();
  }

  void signUpUser() async {
    // set loading to true
    setState(() {
      _isLoading = true;
    });

    // signup user using our authmethodds

```



```

String res = await AuthMethods().signUpUser(
  email: _emailController.text,
  password: _passwordController.text,
  username: _usernameController.text,
  bio: _bioController.text,
  file: _image!);
// if string returned is sucess, user has been created
if (res == "success") {
  setState(() {
    _isLoading = false;
  });
  // navigate to the home screen
  Navigator.of(context).pushReplacement(
    MaterialPageRoute(
      builder: (context) => const ResponsiveLayout(
        mobileScreenLayout: MobileScreenLayout(),
        webScreenLayout: WebScreenLayout(),
      ),
    ),
  );
} else {
  setState(() {
    _isLoading = false;
  });
  // show the error
  showSnackBar(context, res);
}
}

selectImage() async {
  Uint8List im = await pickImage(ImageSource.gallery);
  // set state because we need to display the image we
  selected on the circle avatar
  setState(() {
    _image = im;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,

```

```

body: SafeArea(
  child: Container(
    padding: const EdgeInsets.symmetric(horizontal: 32),
    width: double.infinity,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        Flexible(
          child: Container(),
          flex: 2,
        ),
        SvgPicture.asset(
          'assets/ic_instagram.svg',
          color: primaryColor,
          height: 50,
        ),
        const SizedBox(
          height: 10,
        ),
        Stack(
          children: [
            _image != null
              ? CircleAvatar(
                  radius: 64,
                  backgroundImage:
MemoryImage(_image!),
                  backgroundColor: Colors.red,
                )
              : const CircleAvatar(
                  radius: 50,
                  backgroundImage: NetworkImage(
'https://i.stack.imgur.com/l60Hf
.png'),
                  backgroundColor: Colors.red,
                ),
            Positioned(
              bottom: -10,
              left: 80,
              child: IconButton(
                onPressed: selectImage,
                icon: const Icon(Icons.add_a_photo),
              ),
            ),
          ],
        ),
      ],
    ),
  ),
)

```

```

    )
  ],
),
const SizedBox(
  height: 24,
),
TextFieldInput(
  hintText: 'Enter your username',
  textInputType: TextInputType.text,
  textEditingController: _usernameController,
),
const SizedBox(
  height: 10,
),
TextFieldInput(
  hintText: 'Enter your email',
  textInputType: TextInputType.emailAddress,
  textEditingController: _emailController,
),
const SizedBox(
  height: 10,
),
TextFieldInput(
  hintText: 'Enter your password',
  textInputType: TextInputType.text,
  textEditingController: _passwordController,
  isPass: true,
),
const SizedBox(
  height: 10,
),
TextFieldInput(
  hintText: 'Enter your bio',
  textInputType: TextInputType.text,
  textEditingController: _bioController,
),
const SizedBox(
  height: 10,
),
InkWell(
  child: Container(
    child: !_isLoading

```

```

        ? const Text(
            'Sign up',
        )
        : const CircularProgressIndicator(
            color: primaryColor,
        ),
        height: 10,
        width: double.infinity,
        alignment: Alignment.center,
        padding: const
EdgeInsets.symmetric(vertical: 12),
        decoration: const ShapeDecoration(
            shape: RoundedRectangleBorder(
                borderRadius:
BorderRadius.all(Radius.circular(4)),
            ),
            color: blueColor,
        ),
    ),
    onTap: signUpUser,
),
const SizedBox(
    height: 12,
),
Flexible(
    child: Container(),
    flex: 2,
),
Row(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
        Container(
            child: const Text(
                'Already have an account?',
            ),
            padding: const
EdgeInsets.symmetric(vertical: 8),
        ),
        GestureDetector(
            onTap: () => Navigator.of(context).push(
                MaterialPageRoute(

```

```

        builder: (context) => const
LoginScreen(),
        ),
    ),
    child: Container(
      child: const Text(
        ' Login.',
        style: TextStyle(
          fontWeight: FontWeight.bold,
        ),
      ),
      padding: const
EdgeInsets.symmetric(vertical: 8),
    ),
  ),
],
),
],
),
),
),
);
}
}

```

profile_screen.dart

```

import 'dart:typed_data';

import 'package:flutter/material.dart';
import 'package:flutter_svg/svg.dart';
import 'package:image_picker/image_picker.dart';
import 'package:instagram_clone/resources/auth_methods.dart';
import
'package:instagram_clone/responsive/mobile_screen_layout.dart'
;
import
'package:instagram_clone/responsive/responsive_layout.dart';
import
'package:instagram_clone/responsive/web_screen_layout.dart';
import 'package:instagram_clone/screens/login_screen.dart';
import 'package:instagram_clone/utils/colors.dart';
import 'package:instagram_clone/utils/utils.dart';

```

```

import
'package:instagram_clone/widgets/text_filed_input.dart';

class SignupScreen extends StatefulWidget {
  const SignupScreen({Key? key}) : super(key: key);

  @override
  _SignupScreenState createState() => _SignupScreenState();
}

class _SignupScreenState extends State<SignupScreen> {
  final TextEditingController _usernameController =
TextEditingController();
  final TextEditingController _emailController =
TextEditingController();
  final TextEditingController _passwordController =
TextEditingController();
  final TextEditingController _bioController =
TextEditingController();
  bool _isLoading = false;
  Uint8List? _image;

  @override
  void dispose() {
    super.dispose();
    _emailController.dispose();
    _passwordController.dispose();
    _usernameController.dispose();
  }

  void signUpUser() async {
    // set loading to true
    setState(() {
      _isLoading = true;
    });

    // signup user using our authmethodds
    String res = await AuthMethods().signUpUser(
      email: _emailController.text,
      password: _passwordController.text,
      username: _usernameController.text,
      bio: _bioController.text,

```

```

        file: _image!);
// if string returned is sucess, user has been created
if (res == "success") {
  setState(() {
    _isLoading = false;
  });
  // navigate to the home screen
  Navigator.of(context).pushReplacement(
    MaterialPageRoute(
      builder: (context) => const ResponsiveLayout(
        mobileScreenLayout: MobileScreenLayout(),
        webScreenLayout: WebScreenLayout(),
      ),
    ),
  );
} else {
  setState(() {
    _isLoading = false;
  });
  // show the error
  showSnackBar(context, res);
}
}

selectImage() async {
  Uint8List im = await pickImage(ImageSource.gallery);
  // set state because we need to display the image we
  selected on the circle avatar
  setState(() {
    _image = im;
  });
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    resizeToAvoidBottomInset: false,
    body: SafeArea(
      child: Container(
        padding: const EdgeInsets.symmetric(horizontal: 32),
        width: double.infinity,
        child: Column(

```

```

crossAxisAlignment: CrossAxisAlignment.center,
children: [
  Flexible(
    child: Container(),
    flex: 2,
  ),
  SvgPicture.asset(
    'assets/ic_instagram.svg',
    color: primaryColor,
    height: 50,
  ),
  const SizedBox(
    height: 10,
  ),
  Stack(
    children: [
      _image != null
        ? CircleAvatar(
            radius: 64,
            backgroundImage:
MemoryImage(_image!),
            backgroundColor: Colors.red,
          )
        : const CircleAvatar(
            radius: 50,
            backgroundImage: NetworkImage(
              'https://i.stack.imgur.com/l60Hf
.png'),
            backgroundColor: Colors.red,
          ),
      Positioned(
        bottom: -10,
        left: 80,
        child: IconButton(
          onPressed: selectImage,
          icon: const Icon(Icons.add_a_photo),
        ),
      )
    ],
  ),
  const SizedBox(
    height: 24,

```



```

),
TextFieldInput(
  hintText: 'Enter your username',
  TextInputType: TextInputType.text,
  textEditingController: _usernameController,
),
const SizedBox(
  height: 10,
),
TextFieldInput(
  hintText: 'Enter your email',
  TextInputType: TextInputType.emailAddress,
  textEditingController: _emailController,
),
const SizedBox(
  height: 10,
),
TextFieldInput(
  hintText: 'Enter your password',
  TextInputType: TextInputType.text,
  textEditingController: _passwordController,
  isPass: true,
),
const SizedBox(
  height: 10,
),
TextFieldInput(
  hintText: 'Enter your bio',
  TextInputType: TextInputType.text,
  textEditingController: _bioController,
),
const SizedBox(
  height: 10,
),
InkWell(
  child: Container(
    child: !_isLoading
      ? const Text(
          'Sign up',
        )
      : const CircularProgressIndicator(
          color: primaryColor,

```

```

        ),
        height: 10,
        width: double.infinity,
        alignment: Alignment.center,
        padding: const
EdgeInsets.symmetric(vertical: 12),
        decoration: const ShapeDecoration(
          shape: RoundedRectangleBorder(
            borderRadius:
BorderRadius.all(Radius.circular(4)),
          ),
          color: blueColor,
        ),
      ),
      onTap: signUpUser,
    ),
    const SizedBox(
      height: 12,
    ),
    Flexible(
      child: Container(),
      flex: 2,
    ),
    Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Container(
          child: const Text(
            'Already have an account?',
          ),
          padding: const
EdgeInsets.symmetric(vertical: 8),
        ),
        GestureDetector(
          onTap: () => Navigator.of(context).push(
            MaterialPageRoute(
              builder: (context) => const
LoginScreen(),
            ),
          ),
          child: Container(
            child: const Text(

```



```

// Signing Up User

Future<String> signUpUser({
  required String email,
  required String password,
  required String username,
  required String bio,
  required Uint8List file,
}) async {
  String res = "Some error Occurred";
  try {
    if (email.isNotEmpty ||
        password.isNotEmpty ||
        username.isNotEmpty ||
        bio.isNotEmpty ||
        file != null) {
      // registering user in auth with email and password
      UserCredential cred = await
_auth.createUserWithEmailAndPassword(
        email: email,
        password: password,
      );

      String photoUrl =
        await
StorageMethods().uploadImageToStorage('profilePics', file,
false);

      model.User _user = model.User(
        username: username,
        uid: cred.user!.uid,
        photoUrl: photoUrl,
        email: email,
        bio: bio,
        followers: [],
        following: [],
      );

      // adding user in our database
      await _firestore
        .collection("users")
        .doc(cred.user!.uid)

```

```

        .set(_user.toJson());

        res = "success";
    } else {
        res = "Please enter all the fields";
    }
} catch (err) {
    return err.toString();
}
return res;
}

// logging in user
Future<String> loginUser({
    required String email,
    required String password,
}) async {
    String res = "Some error Occurred";
    try {
        if (email.isNotEmpty || password.isNotEmpty) {
            // logging in user with email and password
            await _auth.signInWithEmailAndPassword(
                email: email,
                password: password,
            );
            res = "success";
        } else {
            res = "Please enter all the fields";
        }
    } catch (err) {
        return err.toString();
    }
    return res;
}

Future<void> signOut() async {
    await _auth.signOut();
}
}

```

firestore_methods.dart

```
import 'dart:typed_data';
```

```

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:instagram_clone/models/post.dart';
import
'package:instagram_clone/resources/storage_methods.dart';
import 'package:uuid/uuid.dart';

class FireStoreMethods {
  final FirebaseFirestore _firestore =
  FirebaseFirestore.instance;

  Future<String> uploadPost(String description, Uint8List
file, String uid,
    String username, String profImage) async {
    // asking uid here because we dont want to make extra
calls to firebase auth when we can just get from our state
management
    String res = "Some error occurred";
    try {
      String photoUrl =
        await StorageMethods().uploadImageToStorage('posts',
file, true);
      String postId = const Uuid().v1(); // creates unique id
based on time
      Post post = Post(
        description: description,
        uid: uid,
        username: username,
        likes: [],
        postId: postId,
        datePublished: DateTime.now(),
        postUrl: photoUrl,
        profImage: profImage,
      );
      _firestore.collection('posts').doc(postId).set(post.toJson());
      res = "success";
    } catch (err) {
      res = err.toString();
    }
    return res;
  }
}

```

```

Future<String> likePost(String postId, String uid, List
likes) async {
  String res = "Some error occurred";
  try {
    if (likes.contains(uid)) {
      // if the likes list contains the user uid, we need to
remove it
      _firestore.collection('posts').doc(postId).update({
        'likes': FieldValue.arrayRemove([uid])
      });
    } else {
      // else we need to add uid to the likes array
      _firestore.collection('posts').doc(postId).update({
        'likes': FieldValue.arrayUnion([uid])
      });
    }
    res = 'success';
  } catch (err) {
    res = err.toString();
  }
  return res;
}

// Post comment
Future<String> postComment(String postId, String text,
String uid,
String name, String profilePic) async {
  String res = "Some error occurred";
  try {
    if (text.isNotEmpty) {
      // if the likes list contains the user uid, we need to
remove it
      String commentId = const Uuid().v1();
      _firestore
        .collection('posts')
        .doc(postId)
        .collection('comments')
        .doc(commentId)
        .set({
          'profilePic': profilePic,
          'name': name,

```

```

        'uid': uid,
        'text': text,
        'commentId': commentId,
        'datePublished': DateTime.now(),
    });
    res = 'success';
} else {
    res = "Please enter text";
}
} catch (err) {
    res = err.toString();
}
return res;
}

// Delete Post
Future<String> deletePost(String postId) async {
    String res = "Some error occurred";
    try {
        await
        _firestore.collection('posts').doc(postId).delete();
        res = 'success';
    } catch (err) {
        res = err.toString();
    }
    return res;
}

Future<void> followUser(
    String uid,
    String followId
) async {
    try {
        DocumentSnapshot snap = await
        _firestore.collection('users').doc(uid).get();
        List following = (snap.data()! as dynamic)['following'];

        if(following.contains(followId)) {
            await
            _firestore.collection('users').doc(followId).update({
                'followers': FieldValue.arrayRemove([uid])
            });
        }
    }
}

```



```
        await _firestore.collection('users').doc(uid).update({
            'following': FieldValue.arrayRemove([followId])
        });
    } else {
        await
        _firestore.collection('users').doc(followId).update({
            'followers': FieldValue.arrayUnion([uid])
        });

        await _firestore.collection('users').doc(uid).update({
            'following': FieldValue.arrayUnion([followId])
        });
    }

} catch(e) {
    print(e.toString());
}
}
```

5.2 Testing Approach

5.2.1 Unit Testing

- Each module is considered independently.
- It focuses on each unit of software as implemented in the source code.

5.2.2 Integration Testing

- Integration testing aims at constructing the program structure while at the same constructing the modules. Modules are integrated by using the top-down approach.
- For unit testing, the modules are tested for the correctness of logic applied and should detect errors in coding. Valid and Invalid data should be created and programs should be made to process this data to catch errors.
- In the registration module, while entering the data for the user, one should only go for a password that should only contain numbers, so one should ensure that it should result in an error message.
- All dates that are entered should be validated. No program should be accepted For system testing when unit tests are satisfactorily concluded, the system as a complete entity must be tested

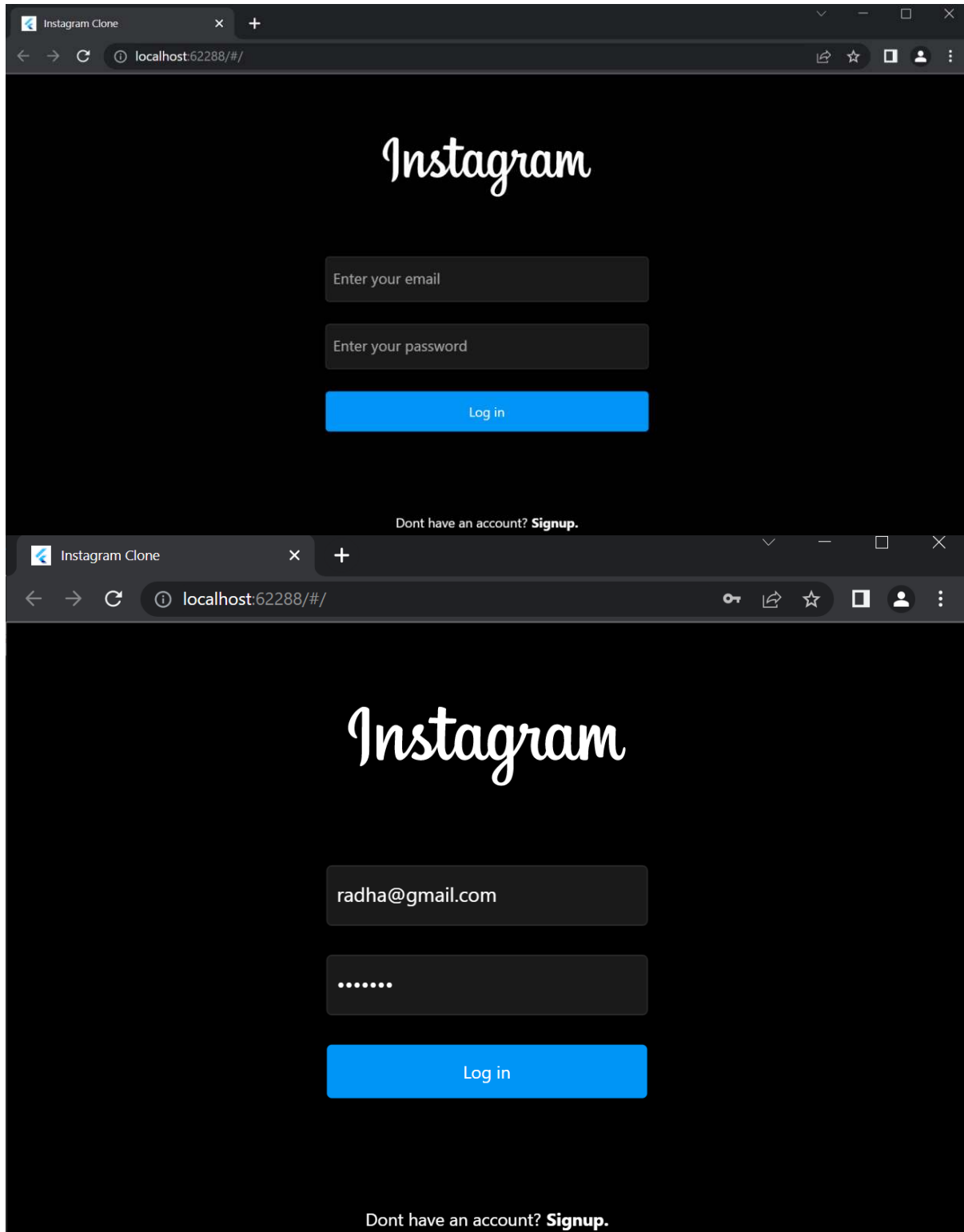
■ SOME COMMON TESTS ARE:

- Handling values in Primary Key fields.
- Handling empty records.
- Max length allowed in controls should match the database structure.
- Handling special characters like ' , #, %, \$, {, }, [,], etc.
- Handling of errors in modules as specified in the project specifications.
- Use global error handling functions if there are any in the project.
- Termination of activities while error should be monitored accordingly.
- Keeping control of the application to self while unexpected errors.
- Raise errors and handle them as and when necessary.
- Showing meaningful error messages as and when required.
- Updating error log file with specific information on errors.
- Use proper captions for error message dialog boxes.
- The format of data should be as specified in the project specifications.
- Showing default values in all subforms as and when required.
- Query handling within the form.
- Enabling and disabling of controls as and when required.
- Showing complete data as and when required

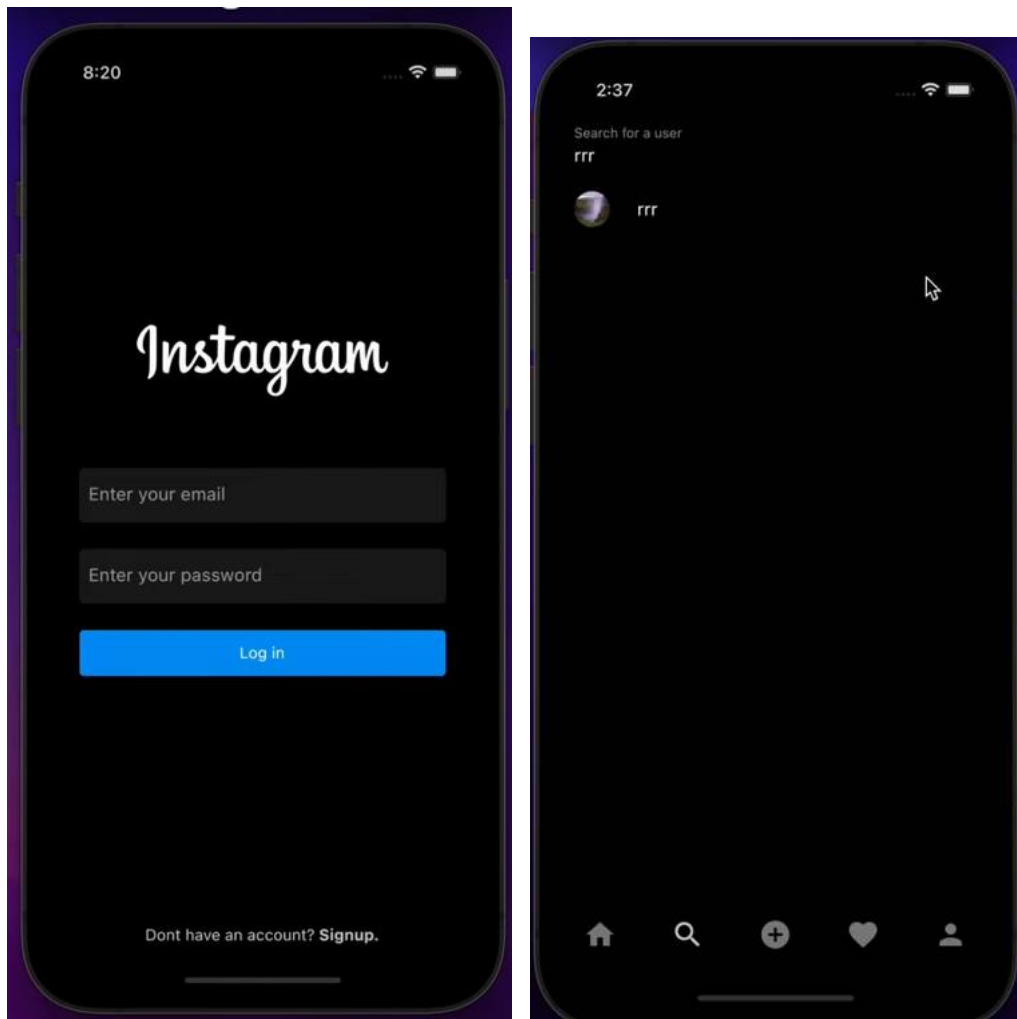
Chapter 5

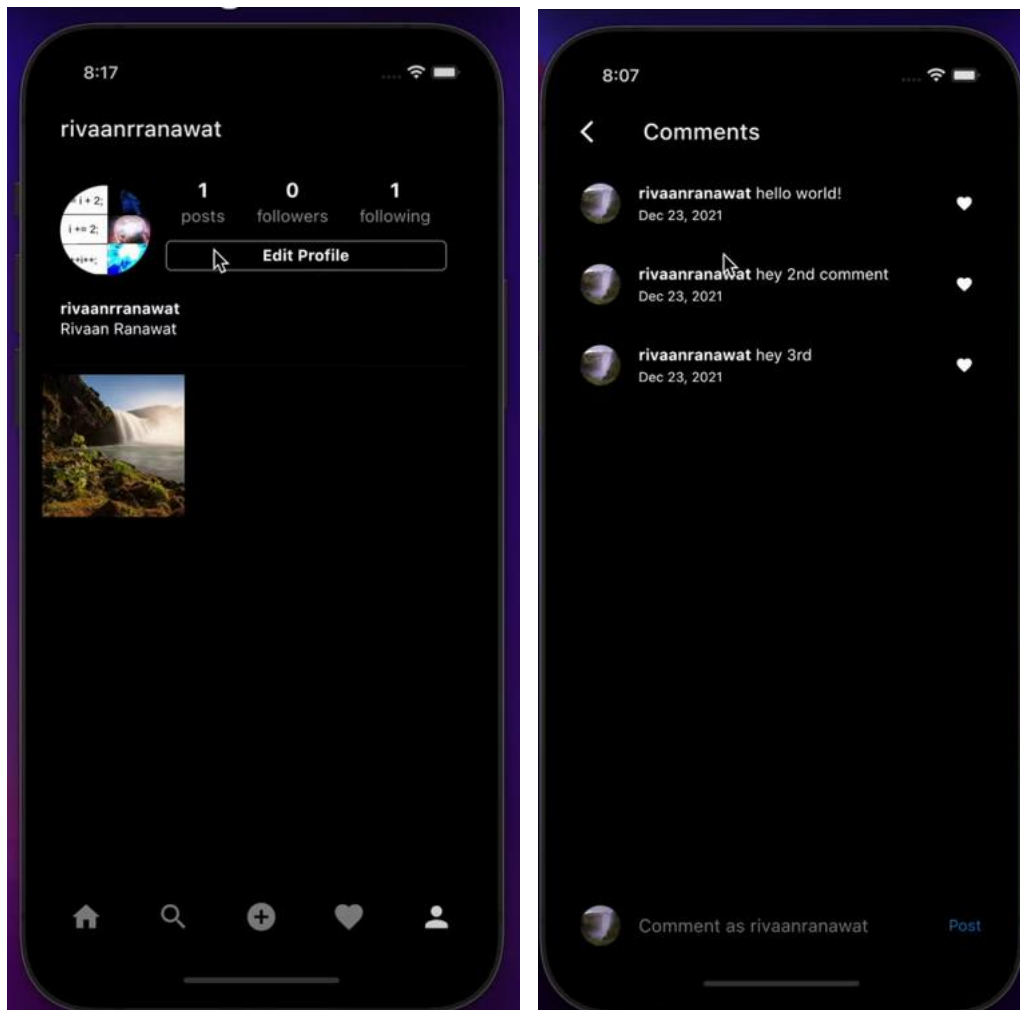
Result and Discussions

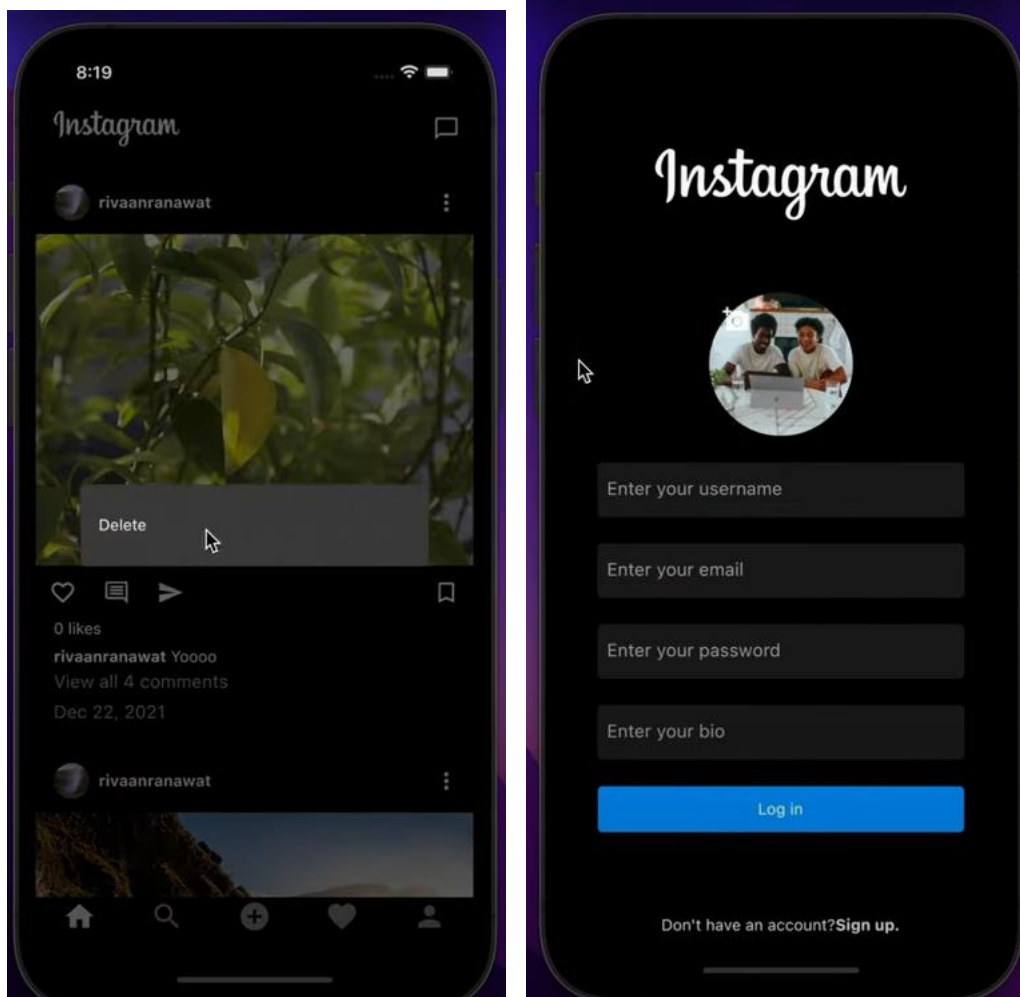
Web :



Android app:









Conclusion and Future Work

Conclusion:

I have developed a Android social media Application , in which we can share image and video , we can follow and unfollow option , theirs a profile page in which follows and following are visiable, in search page we search registered user .

Future Enhancement :

To connect with MircoSoft sql server for unlimated for data.

Reference:

Youtube.com

Google.com

Github.com

<https://github.com/RivaanRanawat/instagram-flutter-clone>