B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institute, Affiliated to VTU)

Course – Mobile Application Development
Course Code – 20IS5PWMAD

Twitter Clone

Submitted By

Rahul T G-1BM20IS112
Rushil M S-1BM20IS124
Sudhanva G V-1BM20IS159

Under the guidance of

Sindhu K
Assistant Professor

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING
AY 2022-2023

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institute, Affiliated to VTU)
Bull Temple Road, Basavanagudi,
Bengaluru – 560019

Department of Information Science and Engineering

C E R T I F I C A T E

This is to certify that the project entitled “Fitness Tracker App” is a bona-fide work carried out by **Rahul TG (1BM20IS112), Rushil MS (1BM20IS124), Sudhanva GV (1BM20IS159)** in partial fulfillment for the award of degree of Bachelor of Engineering in **Information Science and Engineering** from **Visvesvaraya Technological University, Belgaum** during the year **2022-2023.**

Guide name-Sindhu K

Signature-_______________

bilegt sdf @bilegtwe2 · 3 Jan
very majestic

Rajdeep Gh... @rajdeepgh... · 3 Jan
@ccrsxx From where did you learn frontend? Did you follow any course, as your way of writing code is really nice a I would love to learn that.

Muhammad Saad @messi · 3 Jan
im boss
im messi
lolol

bilegt sdf @bilegtwe2 · 2 Jan
damn this thing is fire omg

That Random Du... @ronal... · 2 Jan
this is actully insane, great job!

RAKIMON @raki · 2 Jan
This looks pretty good.. Damn!

Twitter Clone

See what's happening in the world right now.

Sign up with Google

Sign up with Apple

or

Sign up with phone or email

By signing up, you agree to the Terms of Service and Privacy Policy, including Cookie Use.

Already have an account?

Sign in

What should we call you?

Your @username is unique. You can always change it later.

Username: Oyelabs001

Set username

Skip

Twitter Clone

Flutter Android App

Contents

Abstract

Twitter is an online social media and social networking service on which users send and respond publicly or privately texts, images and videos known as "tweets". Registered users can tweet, like, 'retweet' and direct message (DM), while unregistered users only have the ability to view public tweets. Users interact with Twitter through mobile**.** This project will build a communications application similar to the popular application Twitter." This will be called "Twitter Clone" or Twic for short. Twic will allow interested persons to subscribe to download the application to their PC and/or mobile device and subscribe to the service.The service allows a user to post short updates and subscribe to updates by specific persons.

Introduction

Social networks and community websites became more and more important over the last years. Big players on the market like Facebook and Twitter got so many active members, that these platforms even have an effect on daily life of modern societies. The youth of today tweets, pokes, follows and posts on walls instead of writing emails, letters or calling their friends. Tons of private messages, knowledge and feelings get published every second in the data universe.

These interactions between millions of people are a huge technical challenge for the companies. The users wouldn’t use these services if they can’t rely on their real time performance. Though in an environment with a smaller amount of users this real time aspect is simple to handle, it becomes a hardly satisfiable requirement while the amount of users grows keenly.

Aims

The purpose of our application is to develop an application that is valuable to gym goers and people who exercise in general who would like to track their workouts and accomplish their fitness goals.

The graphical user interface of the app should look appealing   ----to the user so as to entice them. The app should provide a pleasant experience and provide a feeling of accomplishment after being used to encourage recurrent usage. It should be highly accessible regardless of the user’s familiarity with applications. Whether the user is a novice or is experienced, the app will be good for both.

The key to this app is simplicity and this app will provide a few features popular in this market, through a simple and straight to the point application. The app should also provide the user with a fun experience.

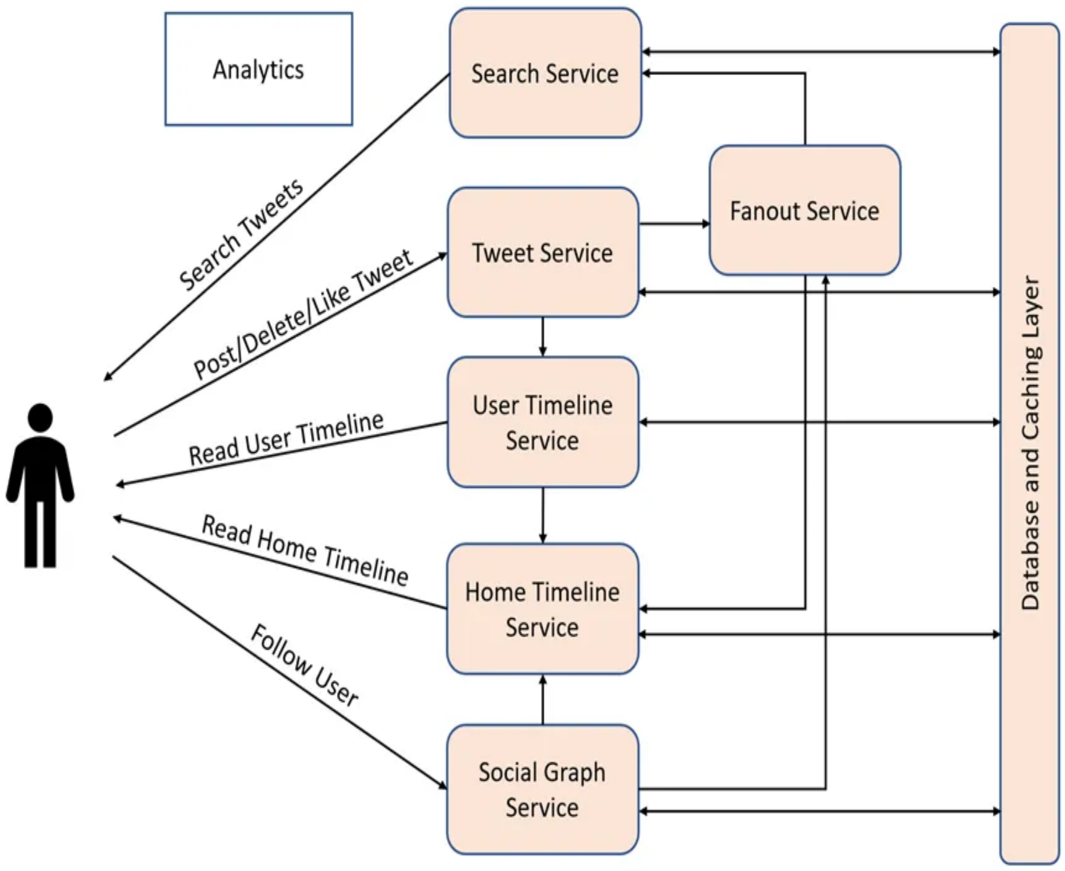
Technologies

In order to complete this project, we have utilized android studio as it is an application that we have become familiar with during our coursework. During our coursework, we have used android studio in the creation of numerous projects, and thus, have used this knowledge to create the classes and GUI -elements of this one. We also used two different android devices with different operating systems, one being a Samsung Galaxy s4 and the other being an Nokia 5.3. These would prove useful during the testing of the application.

Google Firebase was used for user authentication in the app. This was used to implement the login and registration sections. Choosing Firebase was an easy decision as it is easy to implement and provides a variety of authentication options through the Firebase console. The console makes it easy to manage users and includes an option for users to have a password reset sent to their email address.

System

This section will concentrate on a number of areas. We will provide a detailed description of the various types of requirements for the app, the design and architecture of the system, the technologies used and implementation of the system, the graphical user interface and finally the testing methods. We will try to go into as much detail as possible whilst describing each of these topics.



Requirements

The requirements specification part of any project is a very important one. It allows us to gather information on what an audience actually wants from the app itself. This can give us an insight into the perspective of what would be required from the app in order for it to be successful.

In order to gather the requirements for this project, we asked three friends, one of whom is a personal trainer, whilst the other two frequent the gym what they would like to get from a fitness app. Between these three individuals they use a variety of different apps for fitness so they are familiar with what they like and don’t like. Also between them, they use a mixture of both iPhones and android devices. This gave us a solid mixture of requirements that users might come to expect.

Functional Requirements

Register:

The app will have a registration feature where the user will be required to sign up or sign in with his respective valid Gmail Account . These details will then be saved with the google firebase console.

Login:

Users are identified as already registered users when the sign up process is done . These details are then authenticated against the database and the user is directed to the homepage.

Tweets:

Tweets are publicly visible by default, but senders can restrict message delivery to only their followers.The users can post or share new tweets. The size of a tweet is 140 characters at most.

Delete:

The user can delete his tweets but not update/edit his posted tweets (this is a write operation)

Mark Favorite:

The users can mark favorite tweets (write operation)

Follow & Unfollow:

The users can follow or unfollow another user (write operation) — following a user means that the users can see the other user’s tweets on his timeline

Timeline:

2 types of timelines can be generated (read operation) — a user timeline consisting of the last N number of his tweets. A home timeline that is generated from the popular tweets of the users he is following in the descending order of time

Search:

The users can search the tweets based on keywords (read operation) The users need an account in the service to post or read tweets (we will be using an external identity provider for the time being.

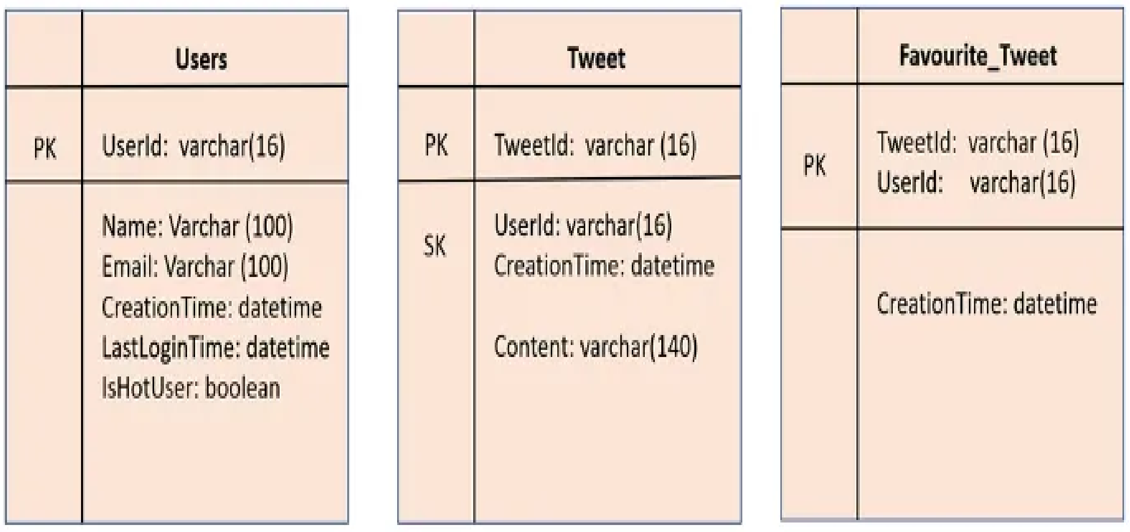
Non-functional requirements

1. The important requirement is that the service should be highly available. It means that the users can read tweets on their home timeline without any downtime
2. Generate the timeline must be within half a second at most
3. The system doesn’t need very strong consistency — eventual consistency is applicable. Keyword database is for searching tweets based on keywords.
4. The system should be scalable with the increasing load for increasing users and increasing tweets
5. User data should be durable

Data Requirements

In this section, I will give a detailed account of the data requirements essential in implementing the above features.

- **Google Firebase:** The app will also make use of google firebase to host user credentials. The firebase authentication functionality is used for the login and registration method



	Users
PK	UserId: varchar(16)
	Name: Varchar (100) Email: Varchar (100) CreationTime: datetime LastLoginTime: datetime IsHotUser: boolean

	Tweet
PK	TweetId: varchar (16)
SK	UserId: varchar(16) CreationTime: datetime Content: varchar(140)

	Favourite_Tweet
PK	TweetId: varchar (16) UserId: varchar(16)
	CreationTime: datetime

PK: Primary Key
SK: Secondary Index Key

User Requirements

This section will describe the user requirements needed so that they will be able to use the application efficiently.

- **Android phone/tablet:** The user must possess an android phone or tablet as the app is not compatible with other devices.

- **API level:** The minimum sdk level for the app is level 15 and the target sdk is 33.

- **Internet access:** The user will also need internet access to sign in/register.

Usability Requirements

- **Simple:** The app will be simple and easy to tweet, yet so simple as to cause users to lose interest. Whilst the app will be easy to use, it will also have a good balance for novice and experienced users alike, so as to appeal to levels.

- **Engaging GUI:** The layout should be simple and make use of a concurrent theme with a good color scheme, in order to grab the attention of users.

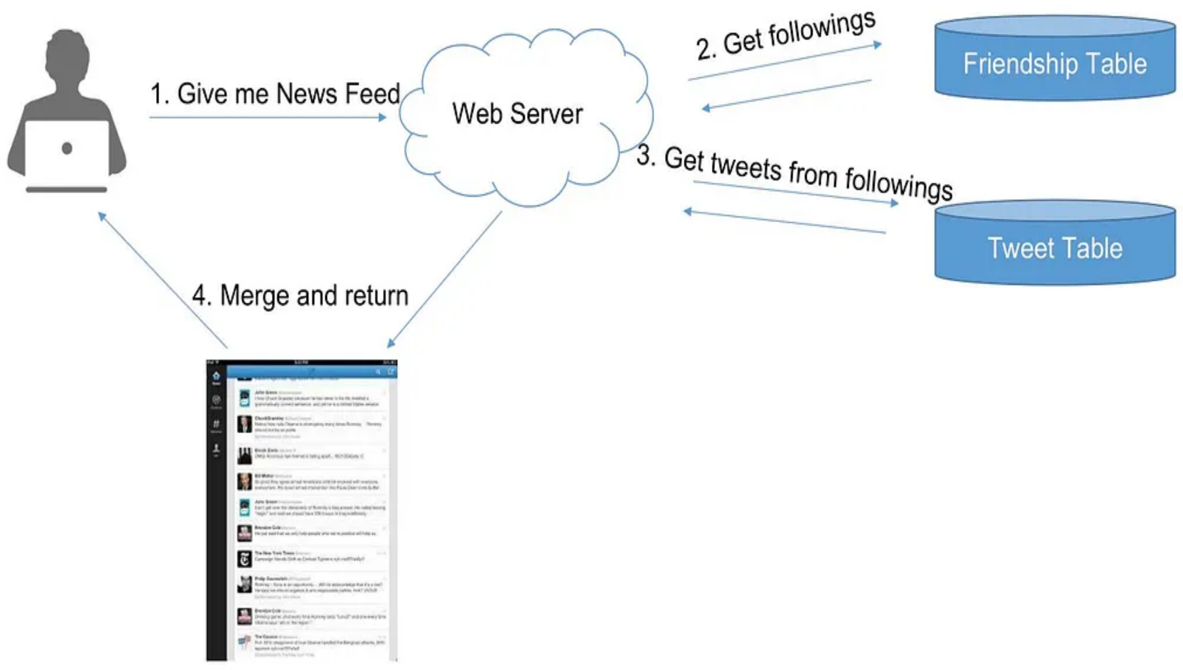
- **Response times/operability:** Error messages should explain issues as they occur. The app won’t leave users on a loading screen wondering when, if ever something is going to happen, as this can lead to them not returning to use the app. Toast messages make up the basis of this. The app's response time should be instantaneous so that the user does not lose interest.

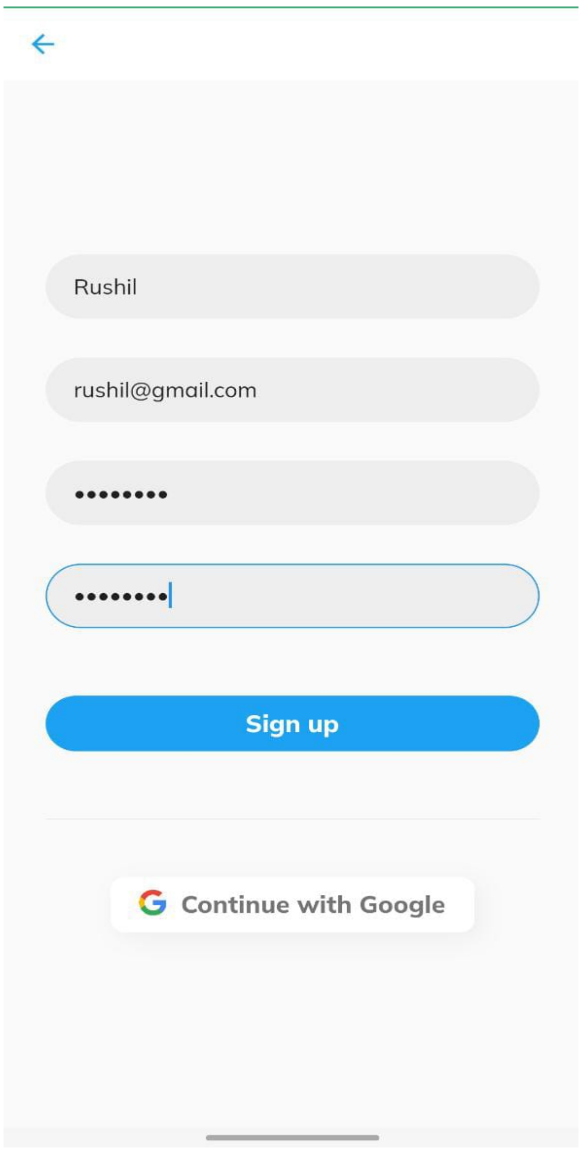
- **Well-structured GUI:** Sometimes apps can pack too much information into our phones tiny screens. This is definitely something the app will attempt to avoid by being well structured and only having a small portion of information visible at any time.

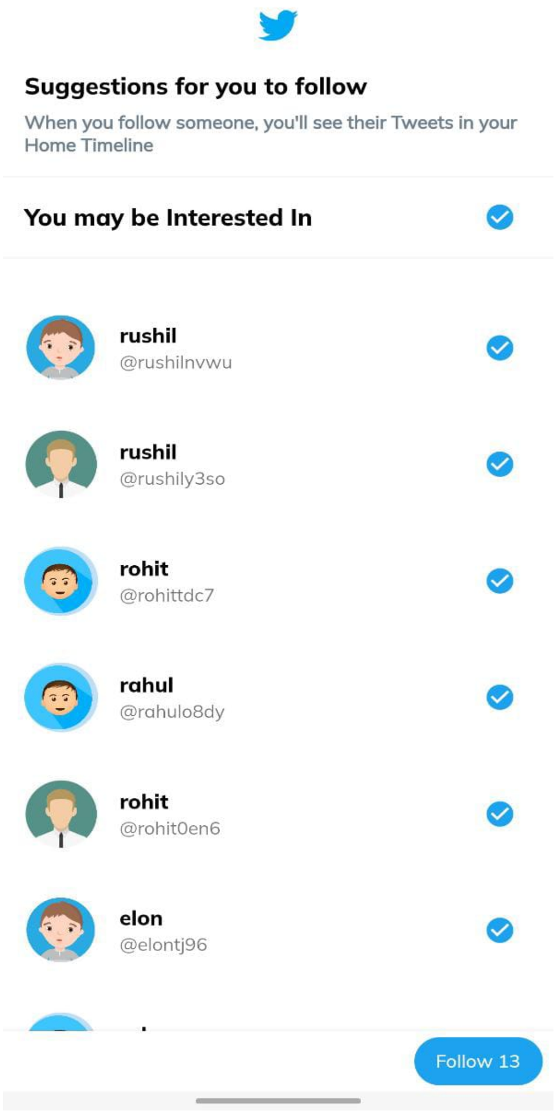
Design and architecture

The app was developed to be based solely around the user's ------health and fitness, so all of the features in it are based around this. The app was built in android studio and each page of it was designed following the same theme. Each feature has an individual class which can be accessed from the homepage. Through customizing my colors.xml file we were -------able to create a color scheme which is present throughout every page of the app. Using different layouts and layout components we were able to design each page as we saw fit.

We developed an ad hoc model to study interactions and different concepts during the design process. We looked at all of the different components and the relationships between them which ultimately laid the foundation for the overall design of our application. The app's design allows the user to traverse through the various pages easily.






Suggestions for you to follow

When you follow someone, you'll see their Tweets in your Home Timeline

You may be Interested In

rushil @rushilnvwu

rushil @rushily3so

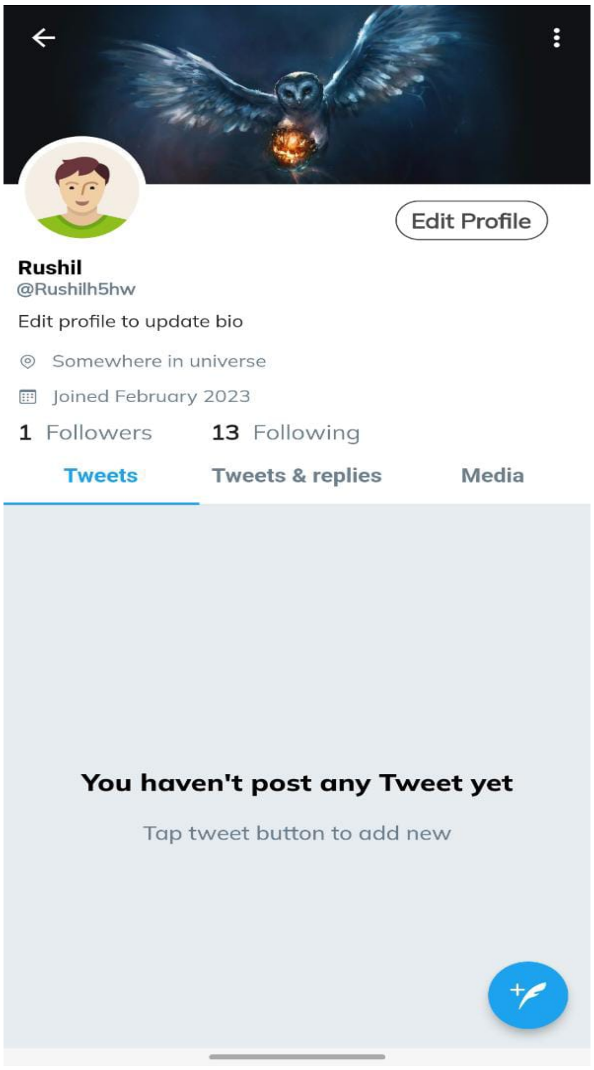
rohit @rohittdc7

rahul @rahulo8dy

rohit @rohit0en6

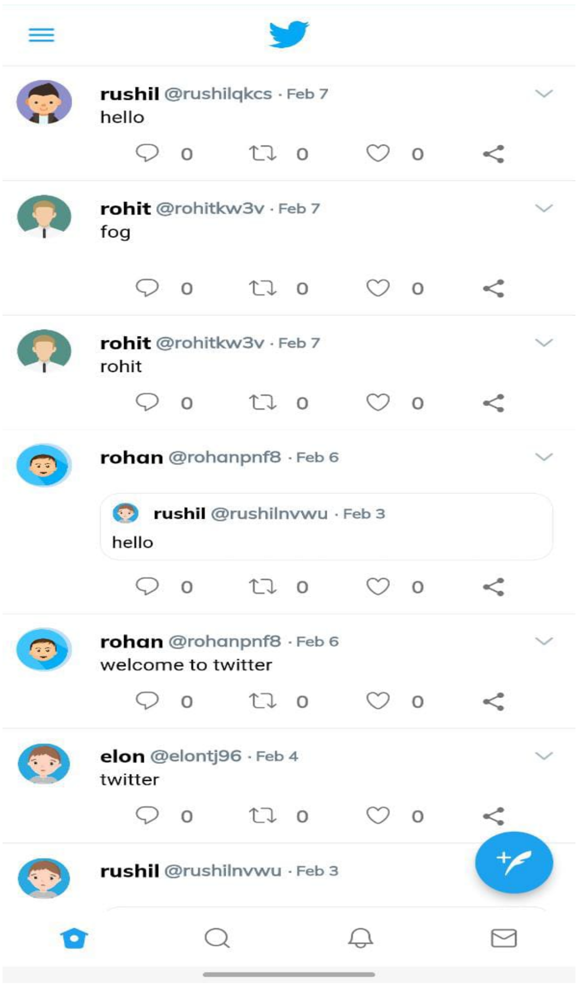
elon @elontj96

Follow 13


Rushil
@Rushilh5hw
Edit profile to update bio
Somewhere in universe
Joined February 2023
1 Followers 13 Following

You haven't post any Tweet yet

Tap tweet button to add new


rushil @rushilqkcs · Feb 7
hello
0 0 0

rohit @rohitkw3v · Feb 7
fog
0 0 0

rohit @rohitkw3v · Feb 7
rohit
0 0 0

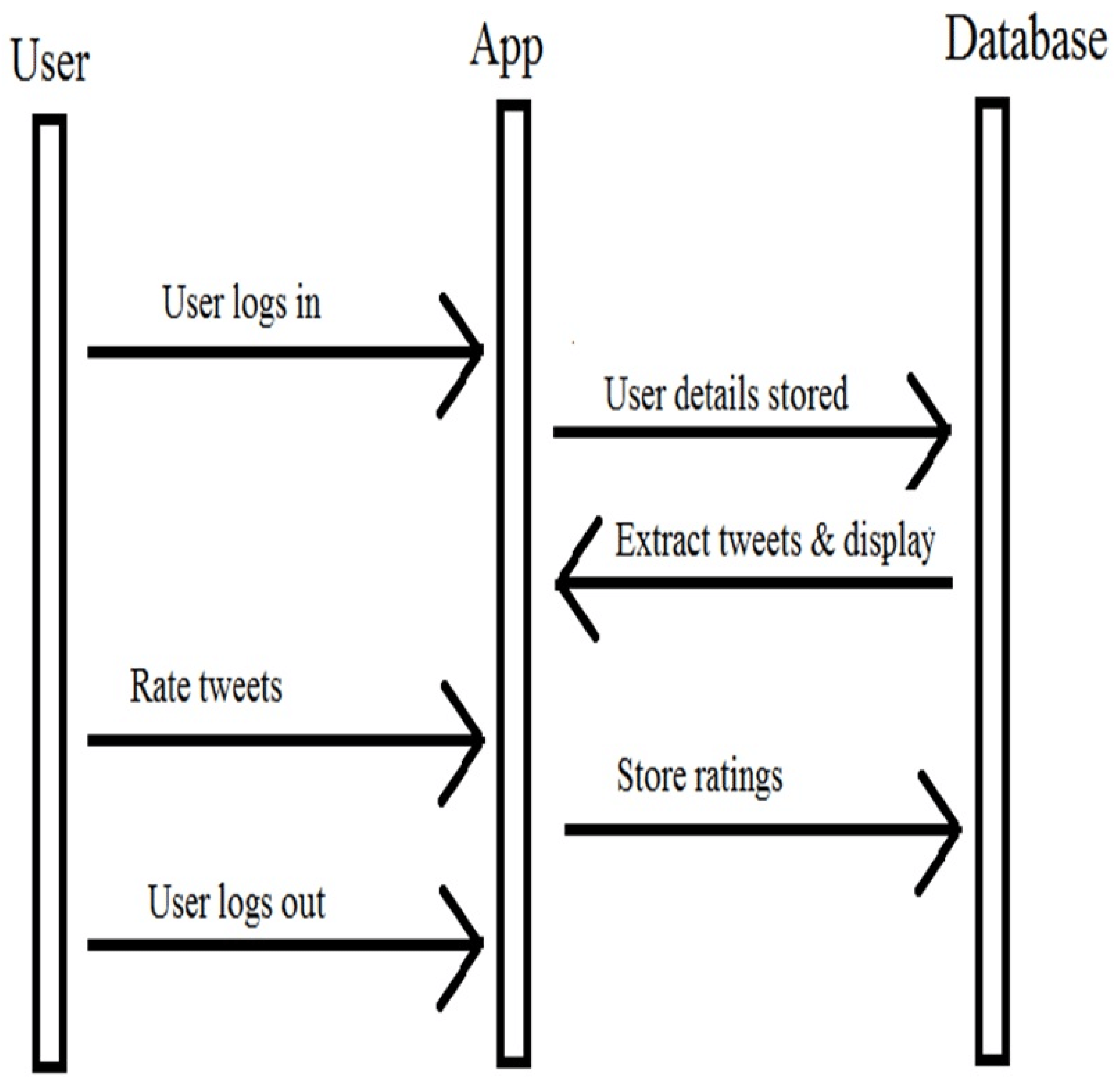
rohan @rohanpnf8 · Feb 6
rushil @rushilnvwu · Feb 3
hello
0 0 0

rohan @rohanpnf8 · Feb 6
welcome to twitter
0 0 0

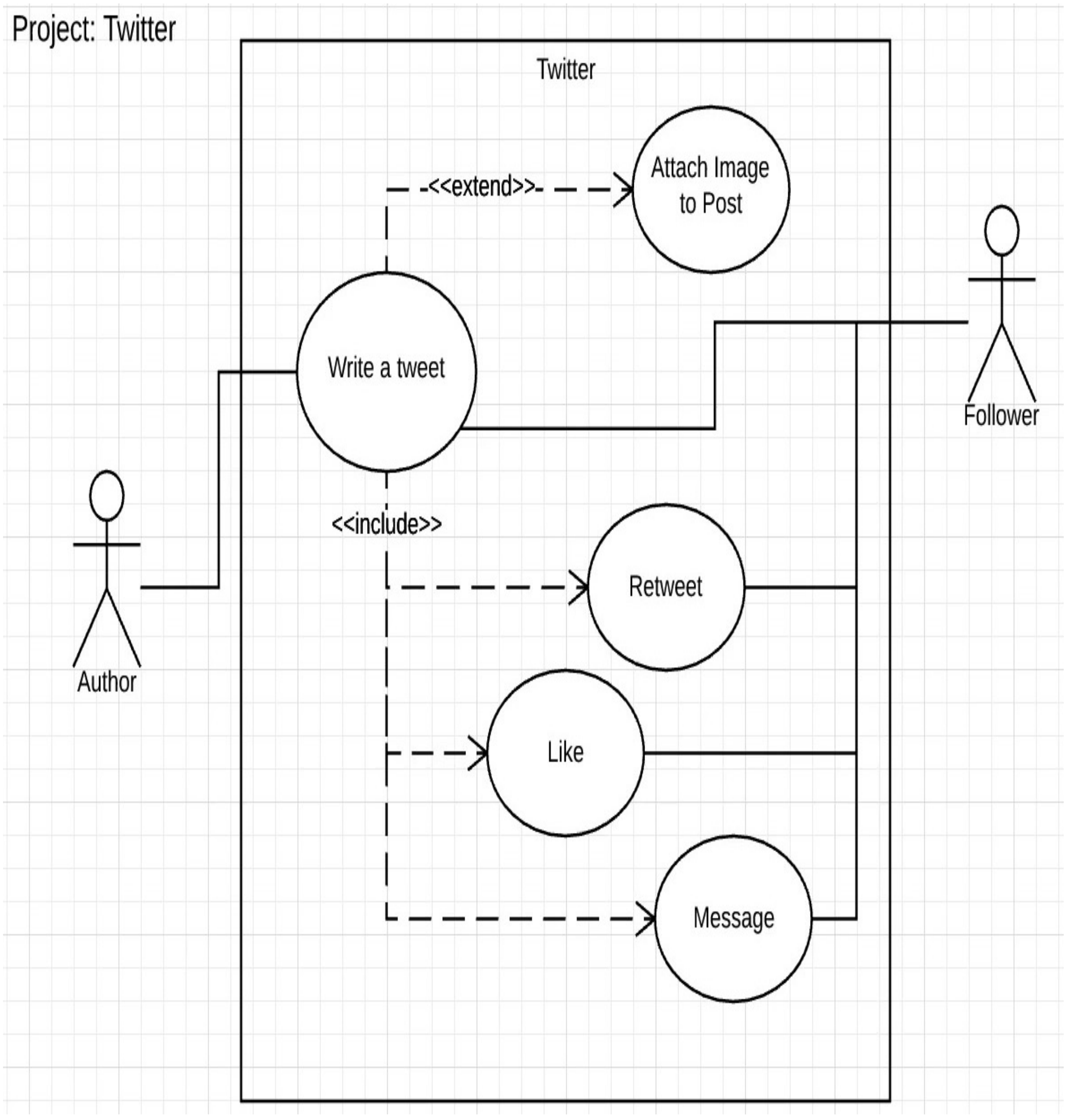
elon @elontj96 · Feb 4
twitter
0 0 0

rushil @rushilnvwu · Feb 3

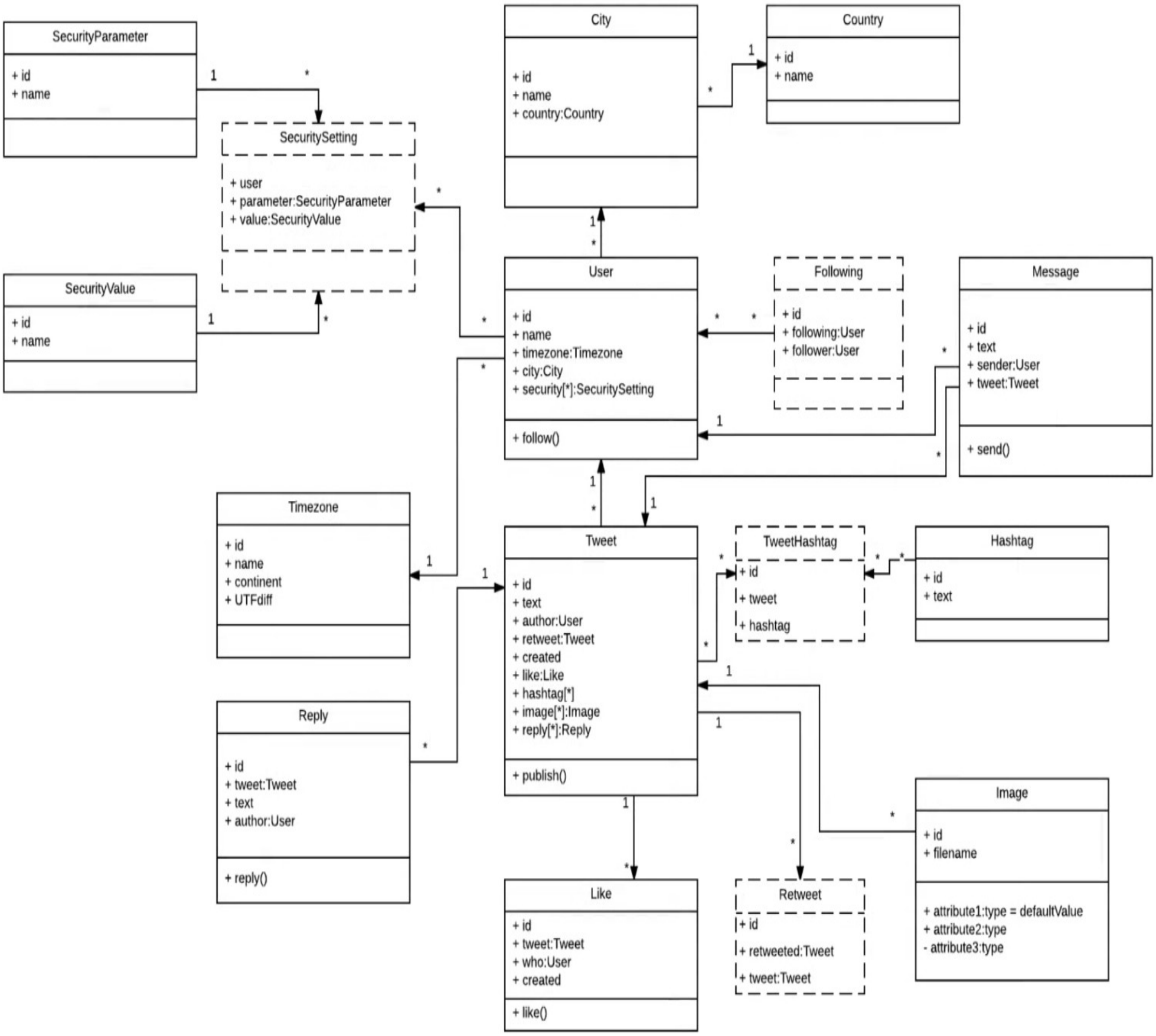
Sequence Diagram



Use Case Diagram



Class Diagram



Implementation

The purpose of this section is to cover the technologies that--were used in the development of this application.

☐ Android Studio

The tool we used to create this application from start to finish was Android Studio. The version we used in making this was...2021.2.1, compile SDK Version “33”,target SDK version “33”.

Profile

```
void _getProfileUser(String? userProfileId) {
    assert(userProfileId != null);
    try {
      loading = true;
      kDatabase
          .child("profile")
          .child(userProfileId!)
          .once()
          .then((DatabaseEvent event) {
        final snapshot = event.snapshot;
        if (snapshot.value != null) {
          var map = snapshot.value as Map;
          // ignore: unnecessary_null_comparison
          if (map != null) {
            _profileUserModel = UserModel.fromJson(map);
            Utility.logEvent('get_profile', parameter: {});
          }
        }
        loading = false;
      });
    } catch (error) {
      loading = false;
      cprint(error, errorIn: 'getProfileUser');
    }
  }
```

Create User

```

void createUserFromGoogleSignIn(User user) {
    var diff = DateTime.now().difference(user.metadata.creationTime!);
    // Check if user is new or old
    // If user is new then add new user to firebase realtime kDatabase
    if (diff < const Duration(seconds: 15)) {
      UserModel model = UserModel(
        bio: 'Edit profile to update bio',
        dob: DateTime(1950, DateTime.now().month, DateTime.now().day + 3)
            .toString(),
        location: 'Somewhere in universe',
        profilePic: user.photoURL!,
        displayName: user.displayName!,
        email: user.email!,
        key: user.uid,
        userId: user.uid,
        contact: user.phoneNumber!,
        isVerified: user.emailVerified,
      );
      createUser(model, newUser: true);
    } else {
      cprint('Last login at: ${user.metadata.lastSignInTime}');
    }
  }
```

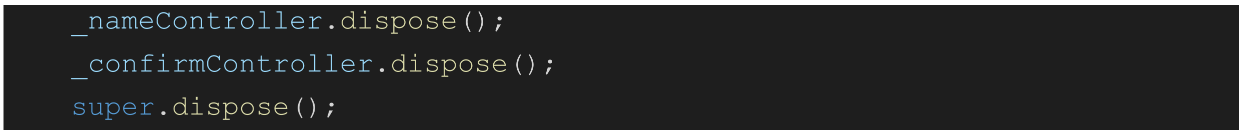
```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp();
  setupDependencies();
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MultiProvider(
      providers: [
        ChangeNotifierProvider<AppState>(create: (_) => AppState()),
        ChangeNotifierProvider<AuthState>(create: (_) => AuthState()),
        ChangeNotifierProvider<FeedState>(create: (_) => FeedState()),
        ChangeNotifierProvider<ChatState>(create: (_) => ChatState()),
        ChangeNotifierProvider<SearchState>(create: (_) =>
SearchState()),
        ChangeNotifierProvider<NotificationState>(
            create: (_) => NotificationState()),
        ChangeNotifierProvider<SuggestionsState>(
            create: (_) => SuggestionsState()),
      ],
      child: MaterialApp(
        title: 'Fwitter',
        theme: AppTheme.appTheme.copyWith(
          textTheme: GoogleFonts.mulishTextTheme(
            Theme.of(context).textTheme,
          ),
        ),
        debugShowCheckedModeBanner: false,
        routes: Routes.route(),
        onGenerateRoute: (settings) => Routes.onGenerateRoute(settings),
        onUnknownRoute: (settings) => Routes.onUnknownRoute(settings),
        initialRoute: "SplashPage",
      ),
    );
  }
}
```

Private Message

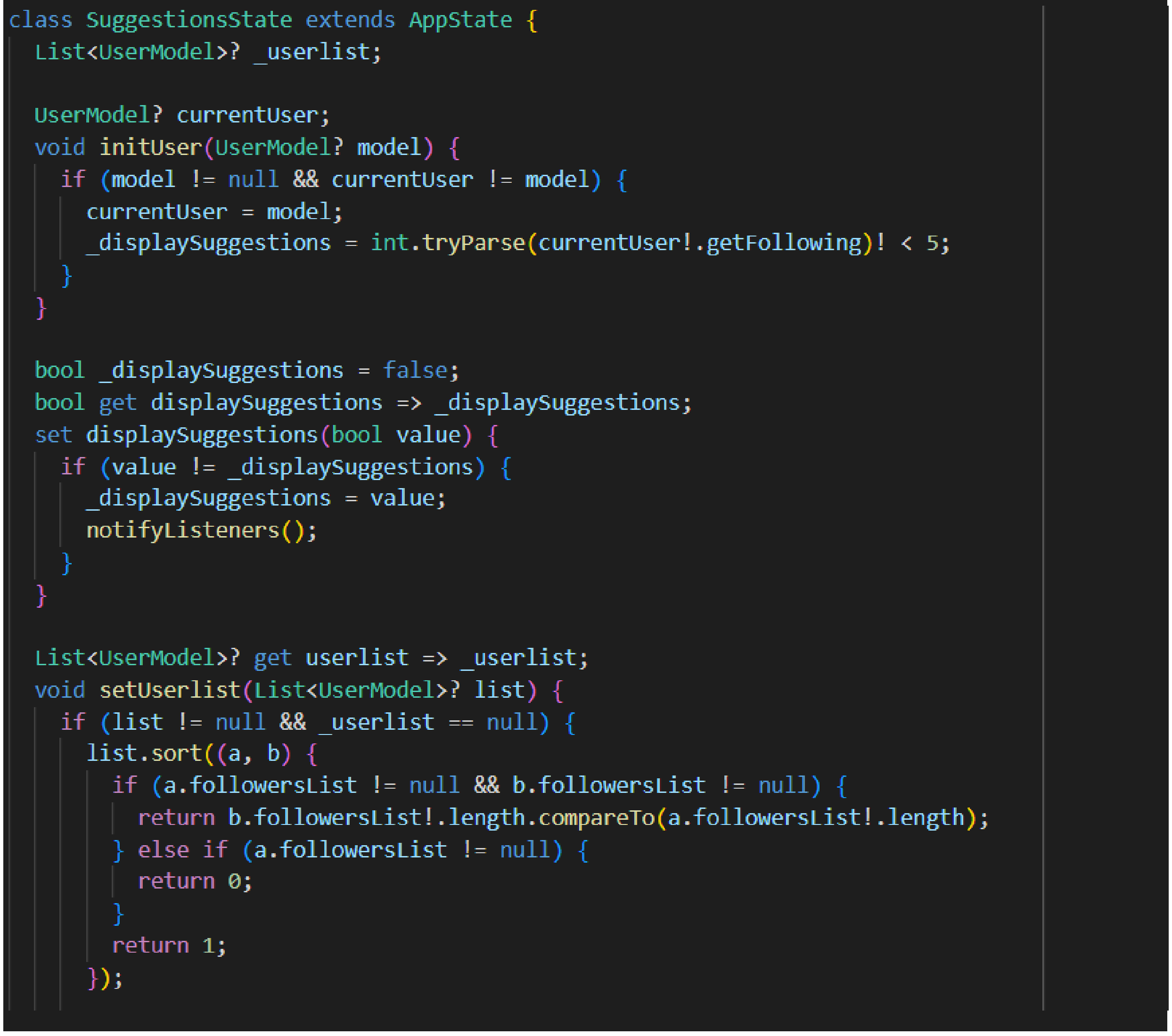
```
class NewMessagePage extends StatefulWidget {
  const NewMessagePage({Key? key, this.scaffoldKey}) : super(key: key);

  final GlobalKey<ScaffoldState>? scaffoldKey;
  @override
  State<StatefulWidget> createState() => _NewMessagePageState();
}


class _NewMessagePageState extends State<NewMessagePage> {
  late TextEditingController textController;

  @override
  void initState() {
    textController = TextEditingController();
    super.initState();
  }

  Widget _userTile(UserModel user) {
    return ListTile(
      onTap: () {
        final chatState = Provider.of<ChatState>(context, listen: false);
        chatState.setChatUser = user;
        Navigator.pushNamed(context, '/ChatScreenPage');
      },
      leading: CircularImage(path: user.profilePic, height: 40),
      title: Row(
        children: <Widget>[
          ConstrainedBox(
            constraints:
                BoxConstraints(minWidth: 0, maxWidth: context.width -
104),
            child: TitleText(user.displayName!,
                fontSize: 16,
                fontWeight: FontWeight.w800,
                overflow: TextOverflow.ellipsis),
          ),
          const SizedBox(width: 3),
          user.isVerified!
              ? customIcon(context,
                  icon: AppIcon.blueTick,
                  isTwitterIcon: true,
                  iconColor: AppColor.primary,
                  size: 13,
                  paddingIcon: 3)
              : const SizedBox(width: 0),
        ],
```

```dart
      ),
      subtitle: Text(user.userName!),
    );
  }
```

```dart
class Signup extends StatefulWidget {
  final VoidCallback? loginCallback;

  const Signup({Key? key, this.loginCallback}) : super(key: key);
  @override
  State<StatefulWidget> createState() => _SignupState();
}

class _SignupState extends State<Signup> {
  late TextEditingController _nameController;
  late TextEditingController _emailController;
  late TextEditingController _passwordController;
  late TextEditingController _confirmController;
  late CustomLoader loader;
  final _formKey = GlobalKey<FormState>();
  @override
  void initState() {
    loader = CustomLoader();
    _nameController = TextEditingController();
    _emailController = TextEditingController();
    _passwordController = TextEditingController();
    _confirmController = TextEditingController();
    super.initState();
  }

  @override
  void dispose() {
    _emailController.dispose();
    _passwordController.dispose();
```



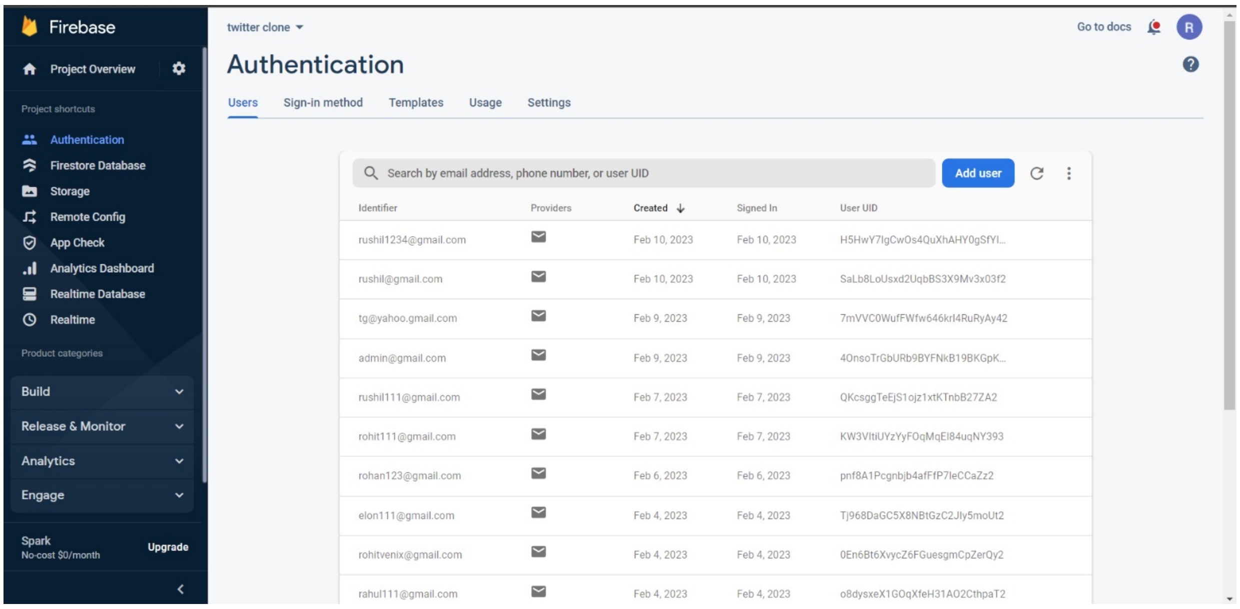
```
    _nameController.dispose();
    _confirmController.dispose();
    super.dispose();
```



```
class SuggestionsState extends AppState {
  List<UserModel>? _userlist;

  UserModel? currentUser;
  void initUser(UserModel? model) {
    if (model != null && currentUser != model) {
      currentUser = model;
      _displaySuggestions = int.tryParse(currentUser!.getFollowing)! < 5;
    }
  }

  bool _displaySuggestions = false;
  bool get displaySuggestions => _displaySuggestions;
  set displaySuggestions(bool value) {
    if (value != _displaySuggestions) {
      _displaySuggestions = value;
      notifyListeners();
    }
  }

  List<UserModel>? get userlist => _userlist;
  void setUserlist(List<UserModel>? list) {
    if (list != null && _userlist == null) {
      list.sort((a, b) {
        if (a.followersList != null && b.followersList != null) {
          return b.followersList!.length.compareTo(a.followersList!.length);
        } else if (a.followersList != null) {
          return 0;
        }
        return 1;
      });
```

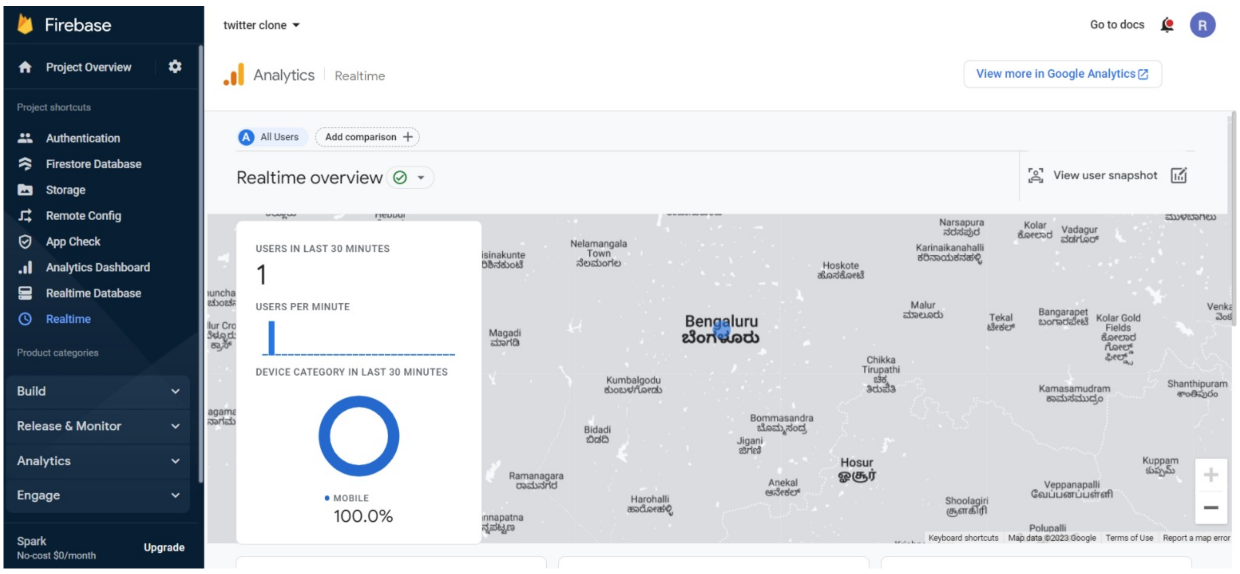
Firebase:

For both login and registration classes Firebase auth allows us to authenticate users. For our login page , Firebase saves input from Google sign in account and runs it against the database. If successful the app starts an intent to the homepage of the app & sending a toast to say registration was successful, the user is then brought to the homepage directly if he is a registered user.



Authentication

Identifier	Providers	Created	Signed In	User UID
rushil1234@gmail.com	✉	Feb 10, 2023	Feb 10, 2023	H5HwY7IgCwOs4QuXhAHY0gSfYI...
rushil@gmail.com	✉	Feb 10, 2023	Feb 10, 2023	SaLb8LoUsxd2UqbBS3X9Mv3x03f2
tg@yahoo.gmail.com	✉	Feb 9, 2023	Feb 9, 2023	7mVVC0WufFWfw646krI4RuRyAy42
admin@gmail.com	✉	Feb 9, 2023	Feb 9, 2023	4OnsoTrGbURb9BYFNkB19BKGpK...
rushil111@gmail.com	✉	Feb 7, 2023	Feb 7, 2023	QKcsggTeEjS1ojz1xtKTnbB27ZA2
rohit111@gmail.com	✉	Feb 7, 2023	Feb 7, 2023	KW3VItiUYzYyFOqMqEI84uqNY393
rohan123@gmail.com	✉	Feb 6, 2023	Feb 6, 2023	pnf8A1Pcgnbjb4afFfP7IeCCaZz2
elon111@gmail.com	✉	Feb 4, 2023	Feb 4, 2023	Tj968DaGC5X8NBtGzC2JIy5moUt2
rohitvenix@gmail.com	✉	Feb 4, 2023	Feb 4, 2023	0En6Bt6XvycZ6FGuesgmCpZerQy2
rahul111@gmail.com	✉	Feb 4, 2023	Feb 4, 2023	o8dysxeX1GOqXfeH31AO2CthpaT2



Realtime overview

USERS IN LAST 30 MINUTES
1

USERS PER MINUTE

DEVICE CATEGORY IN LAST 30 MINUTES

MOBILE
100.0%

Unit Testing

Throughout the course of developing this application, after each new feature was added or changes were made we performed unit testing. Whenever we added something, we ran the app through an emulator on our laptops as well as on both of the android devices we have. We made use of both the android monitor and logcat when testing the app and this was key in helping us identify and eradicate errors. The “Gradle” console also helped with error detection. In regard to Firebase, any issues we encountered were logged via crash reporting and I was able to view the log via Firebase console.

Security Testing

- Used to determine if the social media app data is safeguarded from hackers, spyware, etc.
- Data must be encrypted well.
- Data must be completely deleted if it is not used by the client.
- Many APIs will be connected with the social media app. Hence data privacy and security need to be ensured.

Functionality Testing

- This parameter has to be checked to maintain consistent functionality for a good user experience.
- Should integrate correctly with other applications.
- Determine if the social media app works well in multiple environments and different platforms.
- Ensure each part of the application works as expected.

Usability

- Should provide a user-friendly user interface for the users.
- The developed design should meet the intended workflow.

- This is a good way to test the separate function and the application as a whole.

Performance

- The social media app should behave the same at any peak loads.
- Failure for one user on the cloud for any reason should not affect other users.
- Manual and automatic scaling should not produce any disturbance.
- Perform availability testing for a zero downtime recovery.

Load and stress

- Should work seamlessly when a high number of users are using it concurrently.
- By providing realistic and unrealistic load scenarios, the failure point of the social media app can be found.

Interoperability and compatibility

- The app should work well on different browsers and different platforms on the cloud.
- Any incomplete data should not be transferred to the cloud.

Network

- The network connectivity and data integrity should be consistent while transferring data.

Database

- Front-end fields should be correctly mapped with the database tables.
- The data should be the same on all screens.
- There will be a need for making complex queries using components like relational constraints, triggers, and stored procedures to ascertain the functionalities of the social media app.

Backwards Compatibility Testing

- The app was developed on compile sdk version 33 with the minimum sdk version set at 21. The app has been tested on a number of different versions and multiple devices.
- Most of the features can be used from low versions and upwards. The only concern I would have is if the user's device does not have the hardware required for the step counter to work.

Scenario Testing

- Ensure that on clicking the twitter button, the user gets directed to the twitter's sign-in page.
- Ensure that on signing in, the user is able to share that page (tweet).

Evaluation

Upon completing the testing of this app, we really had our eyes opened at how important testing is in terms of delivering a product. We have our own biased ideas towards a project and testing allows us to view it through other people's eyes highlighting problems you probably wouldn't have noticed which ultimately could have a massive impact on whether someone will come back and use the app again. I took both testing people's advice on board and made changes where necessary.

Conclusion

This report gives an insight into the background as to why we chose to develop this application. We believed that there was a problem with apps of this type and that if done a certain way this could be solved. The idea and our motivations have been stated clearly. The requirements are described in detail and the technologies used have also been outlined in this document. We have touched on the Systems design and architecture and provided a class diagram to explain it further.

We believe that we achieved most of what we set out to do when this idea was conceived. However, there are multiple different areas which could be improved and lots of room for expansion.

References:

- https://www.youtube.com/watch?v=GvIoBgmNgQw

- https://youtu.be/Pye5uM8t7OE

- https://flutter.dev/learn

- https://flutterawesome.com/twitter-using-firebase-auth-realtime-firestore-database-and-storage/