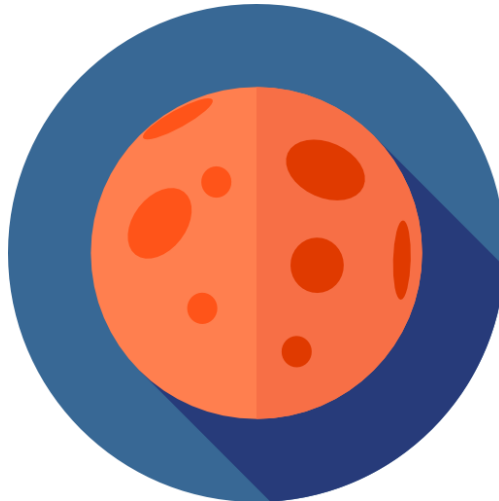


MARS MOBILE APP



JANUARY 9, 2021

AIN SHAMS UNIVERSITY – FACULTY OF ENGINEERING
1 Al-Sarayyat St, Abbassiya, Cairo 11517, Egypt

FACULTY OF ENGINEERING

AIN SHAMS UNIVERSITY

CSE430: Advanced Mobile Computing



Mars Mobile App

Submitted By:

Abdelrahman Ibrahim ELGhamry

16P3043

ghamry1998@gmail.com

Submitted to:

Prof. Dr. Haytham Azmi

Eng. Fatema Mahmoud

JANUARY 9, 2021

**A REPORT FOR ADVANCED MOBILE COMPUTING COURSE CODDED CSE430 WITH THE
REQUIREMENTS OF AIN SHAMS UNIVERSITY**

TABLE OF CONTENTS

1.0	INTRODUCTION.....	3
2.0	APP FEATURES	3
3.0	TECHNOLOGIES USED.....	3
4.0	APPROACH	4
4.1	System Architecture.....	4
4.2	Code Organization	5
4.3	Third Parties.....	6
5.0	APP DESIGN	8
5.1	Portrait	8
5.2	Landscape	9
6.0	APP FLOW AND CORNER CASES	10
6.1	Authentication Scenario.....	10
6.2	View Feeds Scenario	10
6.3	Corner Cases	10
7.0	IMPORTANT LINKS	11
7.1	Application (APK)	11
7.2	Demo Video (MP4).....	11
7.3	Source Control (GIT).....	11
8.0	CODE SNIPPETS	11
8.1	Views and XML.....	11
8.2	Adapters	13
8.3	View Models	15
8.4	Repositories	15
8.5	Services.....	16
8.6	Entities.....	18
8.7	Utils	18

1.0 INTRODUCTION

In the recent decades, there has been a persistent increase in the number of researches, projects and smart thoughts about the planet Mars. All these amazing information “data” are unfortunately located in different places, and hence requires a lot of time to search for and find.

This had driven me to implement this app (Mars) which is a platform that provides researchers, techies, and enthusiastic people a simple, cost efficient and reliable way to communicate and share their thoughts about space exploration and Mars.

And for this matter the app features and architecture, technologies used, and app design will be discussed in this report.

2.0 APP FEATURES

1. Social login using users’ Google and Facebook accounts.
2. The app maintains the user session, and hence keeps the user logged in every time he opens the app.
3. Fetch and display all posts in a chronological order based on the post creation date.
4. The ability to refresh the feeds and get the latest updates.
5. The ability to add post by either typing its title and description or through using the Speech-To-Text feature.
6. The ability to listen to the feed posts through the Text-To-Speech feature.
7. Push and in-app notifications driven by the admin through the firebase console for important announcements.
8. Store all app data as users and posts in Firebase Cloud Firestore database.
9. The app is structured using MVVM architecture.
10. Responsive design that fits on any android screen size.

3.0 TECHNOLOGIES USED

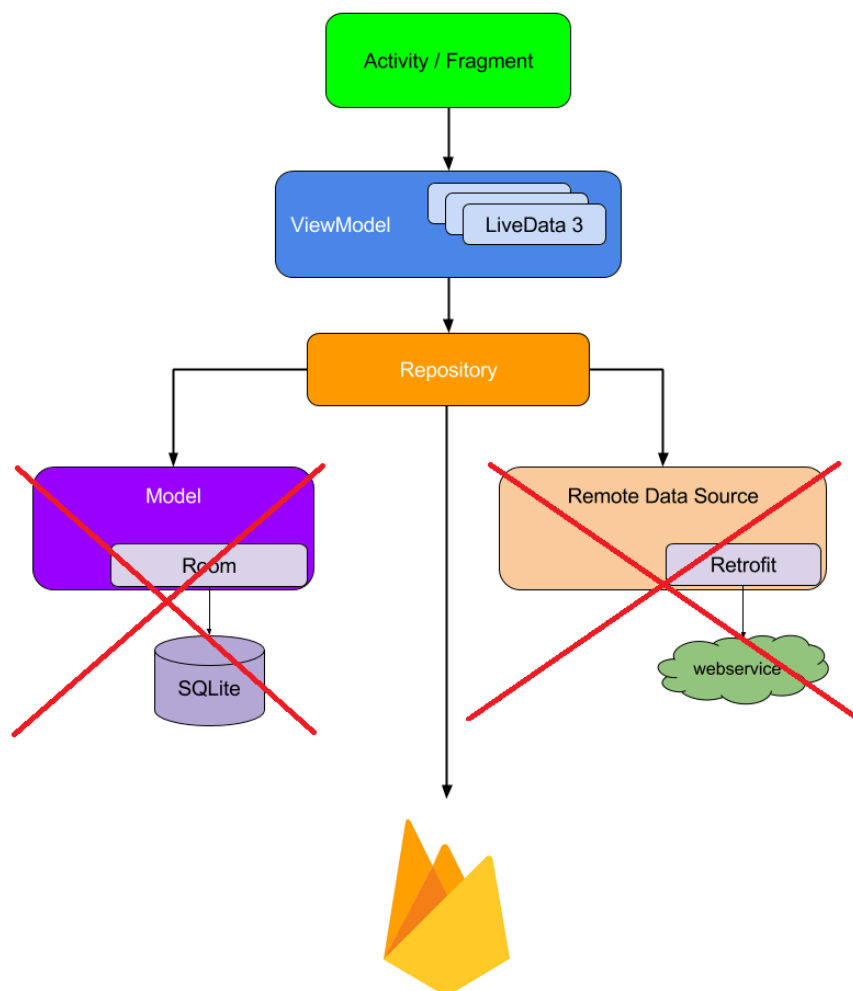
1. Android using Java.
2. Firebase cloud firestore.
3. Firebase authentication service.
4. Firebase push notifications service.
5. Facebook console for developers.

4.0 APPROACH

4.1 System Architecture

The app depends on the MVVM architecture pattern which gives provides a simple way to separate the business logic of your app from the GUI.

Also, the app uses Firebase Authentication Services (Google and Facebook) for user authentication and Firebase Cloud Firestore as a remote database.



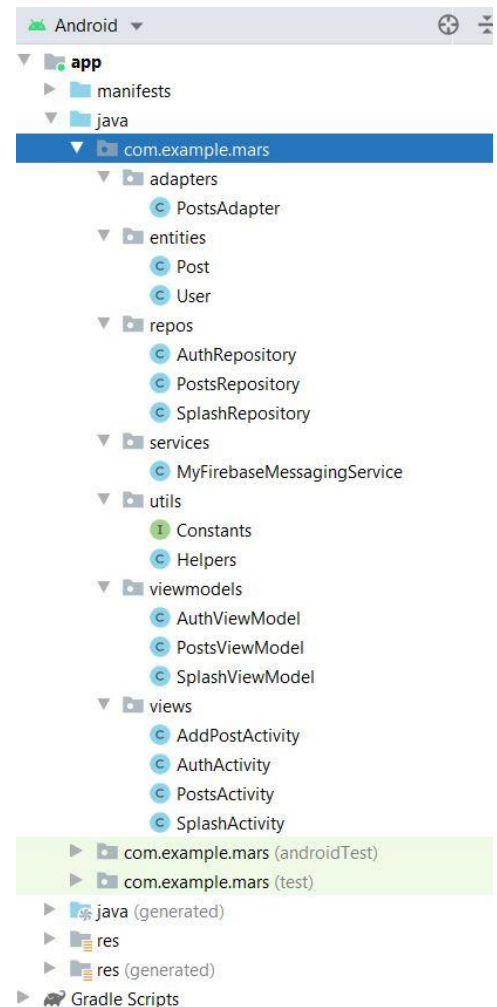
This diagram comes straight out from the Android Architecture Components, except the fact that we'll not be using Room for local persistence, since Firestore already has its own caching mechanism, nor or a web service as a remote data source, we'll be simply using Firebase.

As you can see, we have ViewModels and a LiveData and we need to make them work together using the Repository. This solution is very efficient, and it's actually recommended by the Android team.

4.2 Code Organization

The code is divided into seven main parts:

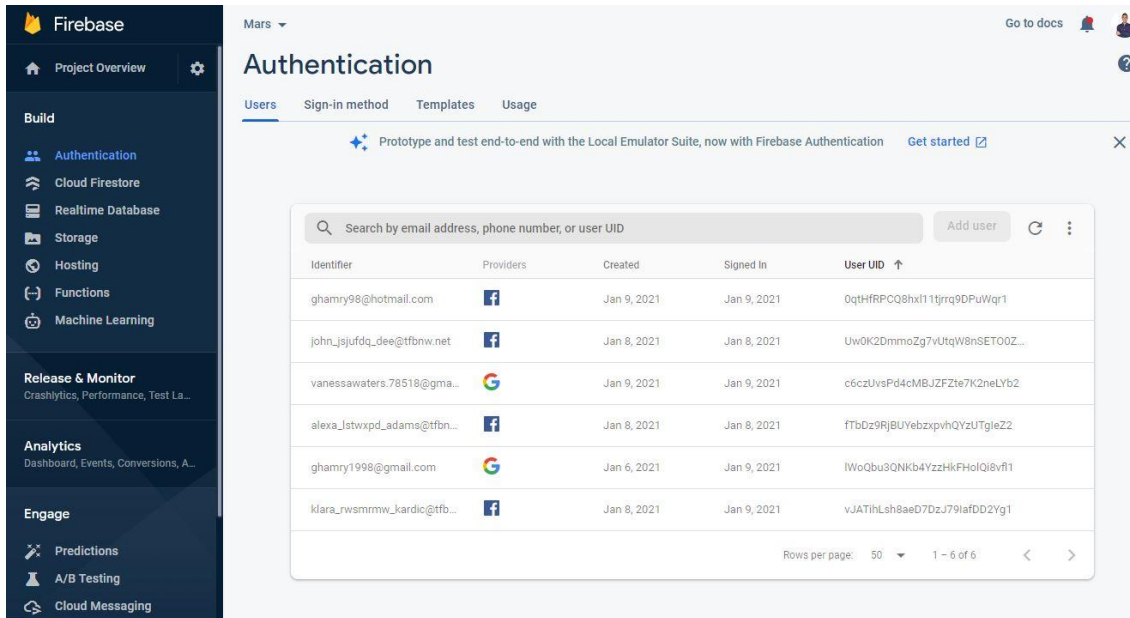
1. **Adapters:** responsible for binding the LiveData to the recycler view items.
2. **Entities:** defines the structure of the objects used within the system.
3. **Repositories:** holds the LiveData and responsible for the connection with the cloud firestore.
4. **Services:** represents the services used inside the system as the push notifications service.
5. **Utils:** contains all the utilities needed by the app, such as all the constant values and all the helper functions used in the app.
6. **ViewModels:** exposes the data to the Views and maintain some state for the View.
7. **Views:** the attached part with the XML files, which is aware of the activity lifecycle and takes actions based on it. It also communicates with and triggers the ViewModels in order to get the needed data.



4.3 Third Parties

1. Firebase Authentication Service

- Google Login.
- Facebook Login.

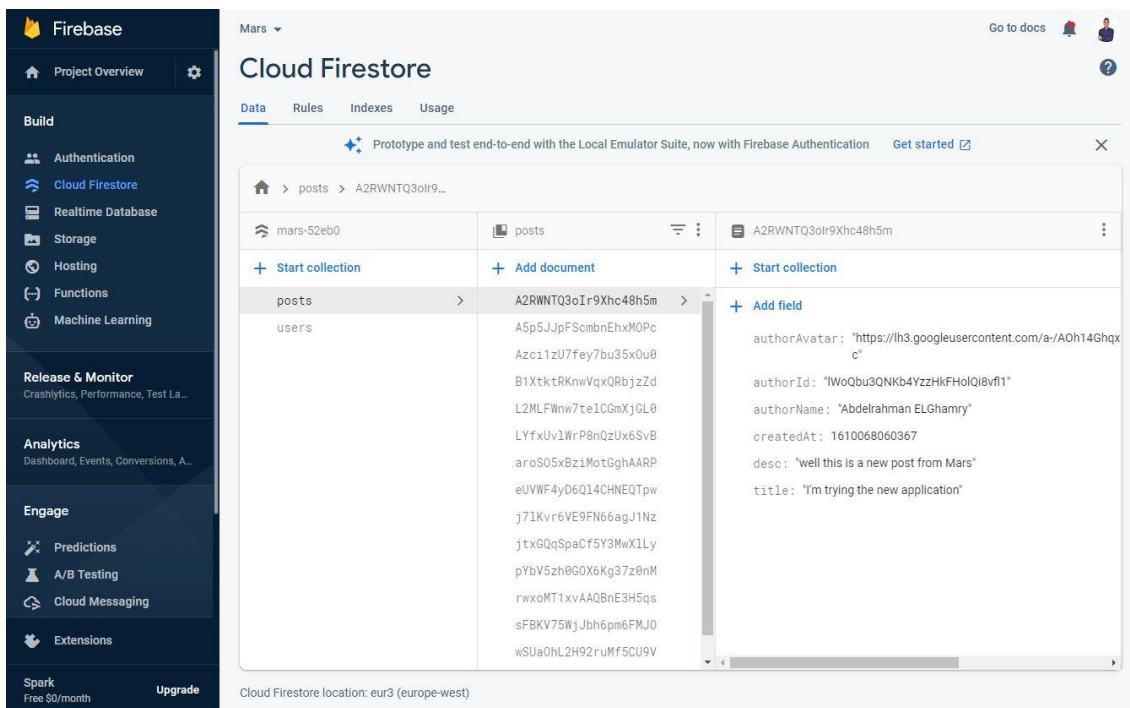


The screenshot shows the Firebase Authentication console. On the left is a sidebar with navigation options: Project Overview, Build (Authentication, Cloud Firestore, Realtime Database, Storage, Hosting, Functions, Machine Learning), Release & Monitor, Analytics, and Engage. The main panel is titled 'Authentication' and has tabs for Users, Sign-in method, Templates, and Usage. A search bar at the top allows searching by email address, phone number, or user UID. Below the search bar is a table of users.

Identifier	Providers	Created	Signed In	User UID
ghamry98@hotmail.com	Facebook	Jan 9, 2021	Jan 9, 2021	QqthFRPCQ8hx11tjrrq9DPuWqr1
john_jsjufdq_dee@tfbnw.net	Facebook	Jan 8, 2021	Jan 8, 2021	Uw0K2DmmoZg7vUtgW8nSET00Z...
vanessawaters.78518@gmail.com	Google	Jan 9, 2021	Jan 9, 2021	c6czUv9Pd4cMBJZFZt7K2neLYb2
alexa_lstwxdp_adama@tfbnw.net	Facebook	Jan 8, 2021	Jan 8, 2021	fTbDz9RjBUYebzpxphvQYzUTgleZ2
ghamry1998@gmail.com	Google	Jan 6, 2021	Jan 9, 2021	lWoQbu3QNKb4YzzHkFHoIQi8vfl1
klara_nwsmmw_kardic@tfbnw.net	Facebook	Jan 8, 2021	Jan 9, 2021	vJATihLsh8aeD7DzJ79lafDD2Yg1

2. Firebase Cloud Firestore (Database)

- Table for users.
- Table for posts.

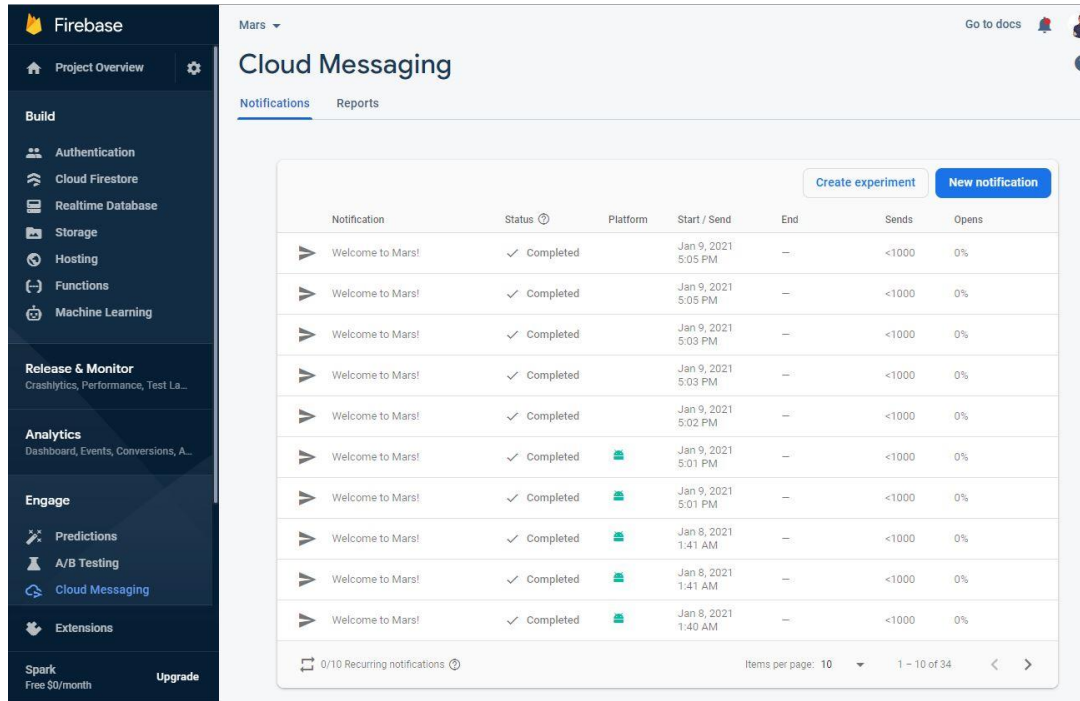


The screenshot shows the Firebase Cloud Firestore console. On the left is the same sidebar as the previous screenshot. The main panel is titled 'Cloud Firestore' and has tabs for Data, Rules, Indexes, and Usage. A breadcrumb trail shows the path: posts > A2RWNTQ3olr9... The main area displays a collection named 'posts' with a document 'A2RWNTQ3olr9Xhc48h5m'. The document contains fields for authorAvatar, authorId, authorName, createdAt, desc, and title.

Field	Value
authorAvatar	"https://lh3.googleusercontent.com/a-/AOh14Ghqxc"
authorId	"lWoQbu3QNKb4YzzHkFHoIQi8vfl1"
authorName	"Abdelrahman ELGhamry"
createdAt	1610068060367
desc	"well this is a new post from Mars"
title	"I'm trying the new application"

3. Firebase Cloud Messaging (FCM)

- Send to all users (Broadcast).
- Send on a certain topic (Initially all users are subscribed to “posts” topic).



The screenshot shows the Firebase Cloud Messaging console for a project named 'Mars'. The left sidebar contains navigation links for Build (Authentication, Cloud Firestore, Realtime Database, Storage, Hosting, Functions, Machine Learning), Release & Monitor, Analytics, Engage (Predictions, A/B Testing, Cloud Messaging), and Extensions. The main content area is titled 'Cloud Messaging' and has tabs for 'Notifications' and 'Reports'. Below the tabs is a table of notifications. The table has columns for Notification, Status, Platform, Start / Send, End, Sends, and Opens. The notifications are all 'Welcome to Mars!' and are marked as 'Completed'. The platform for the first three is 'Android' and for the last three is 'iOS'. The start times range from Jan 8, 2021 1:40 AM to Jan 9, 2021 5:03 PM. The number of sends is less than 1000 for all, and the open rate is 0%.

Notification	Status	Platform	Start / Send	End	Sends	Opens
Welcome to Mars!	✓ Completed	Android	Jan 9, 2021 5:05 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	Android	Jan 9, 2021 5:05 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	Android	Jan 9, 2021 5:03 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	Android	Jan 9, 2021 5:03 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	Android	Jan 9, 2021 5:02 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	iOS	Jan 9, 2021 5:01 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	iOS	Jan 9, 2021 5:01 PM	—	<1000	0%
Welcome to Mars!	✓ Completed	iOS	Jan 8, 2021 1:41 AM	—	<1000	0%
Welcome to Mars!	✓ Completed	iOS	Jan 8, 2021 1:41 AM	—	<1000	0%
Welcome to Mars!	✓ Completed	iOS	Jan 8, 2021 1:40 AM	—	<1000	0%

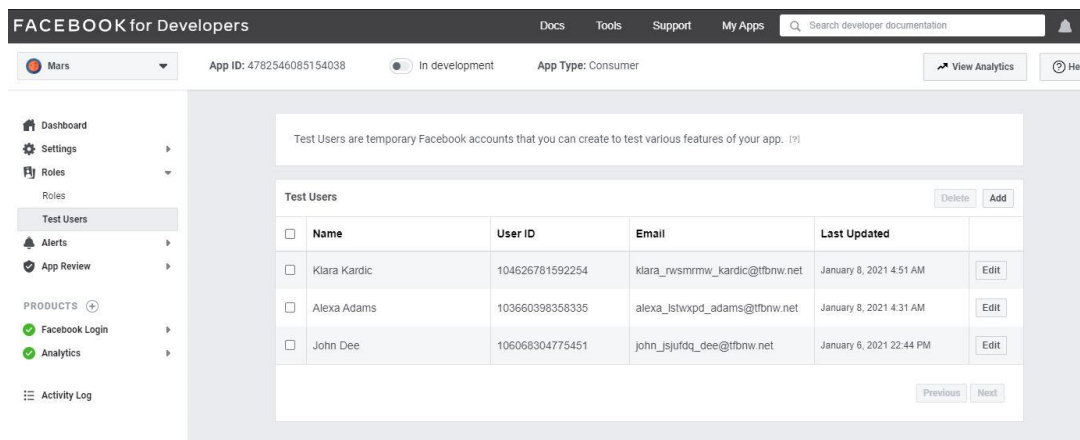
4. Facebook Developers Console

We've got 3 test users:

- mars_hxbmpvv_user@tfbnw.net
- alexa_1stwxpd_adams@tfbnw.net
- klara_rwsrmw_kardic@tfbnw.net

All have the same password which is “Vorx!123”.

(Kindly be noted that the app is in-development mode, so we can only use Facebook authentication through the registered devices on console).

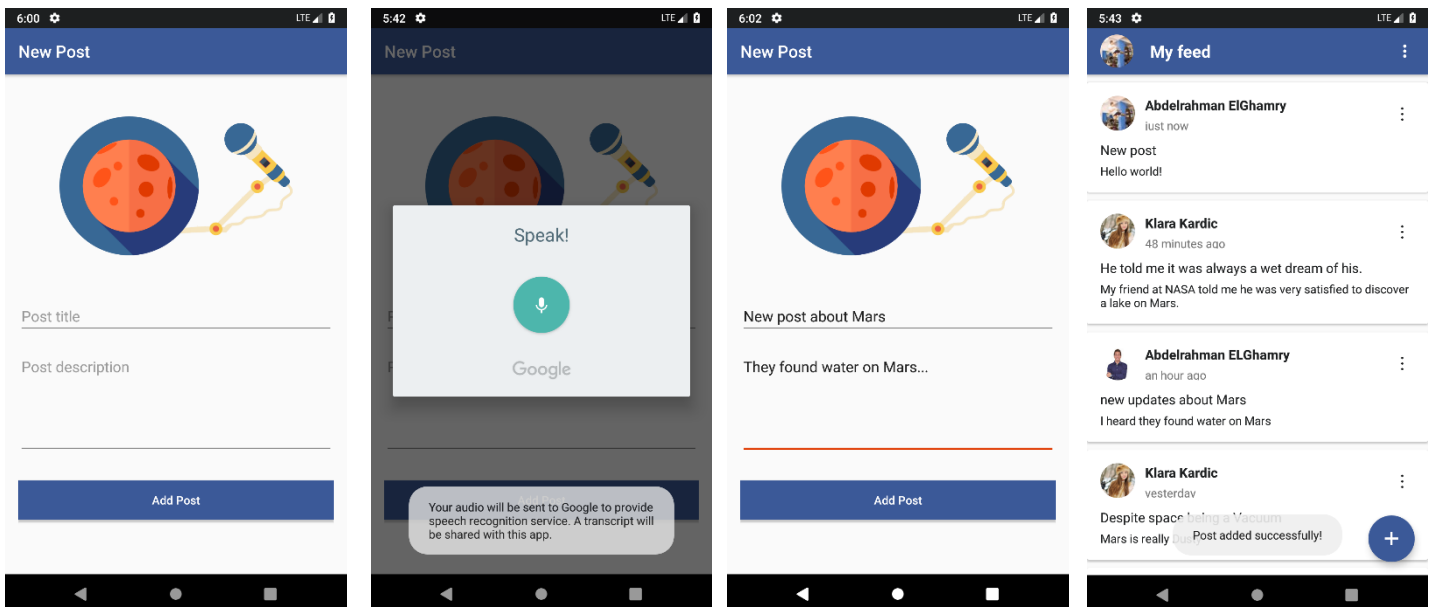
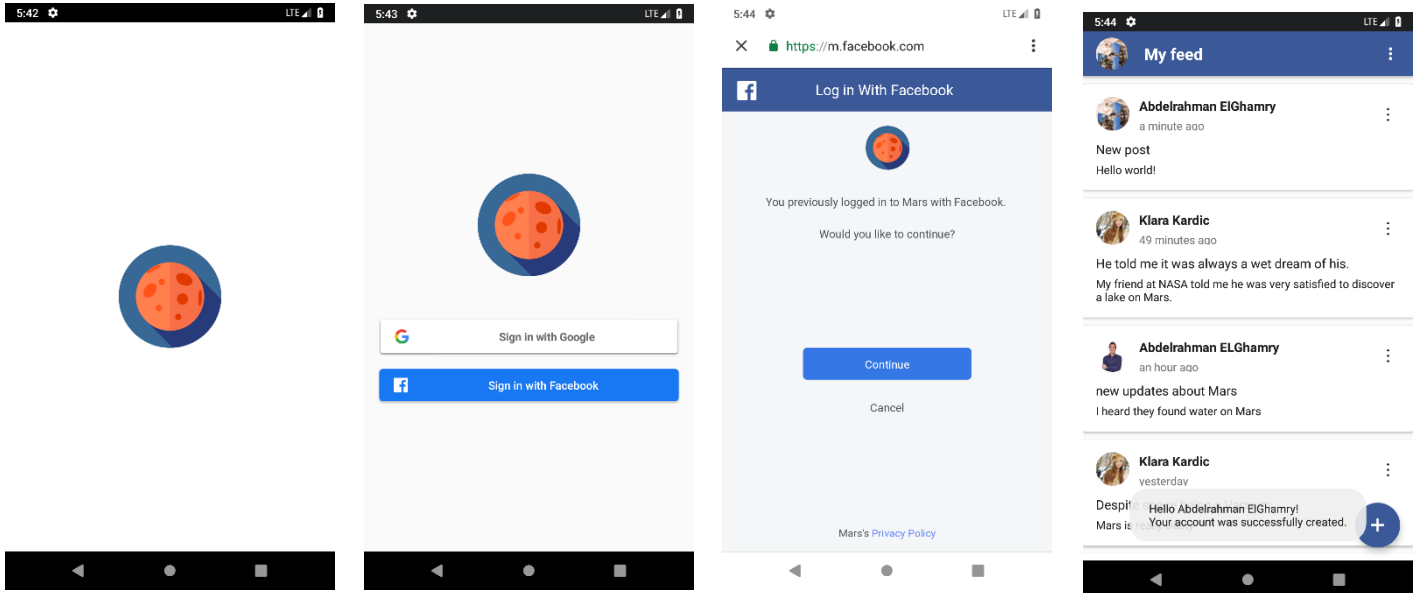


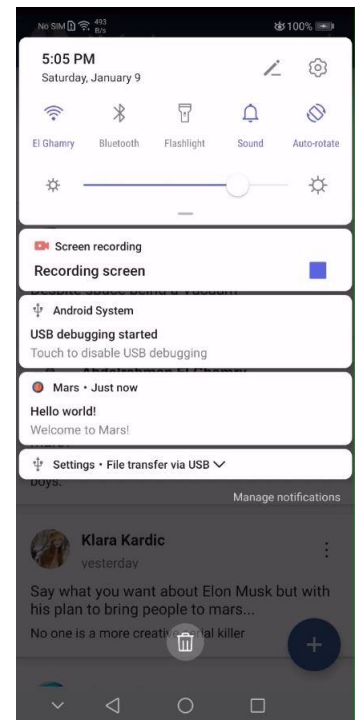
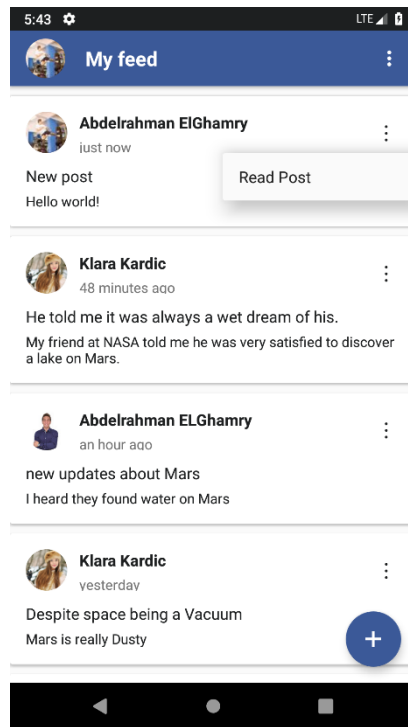
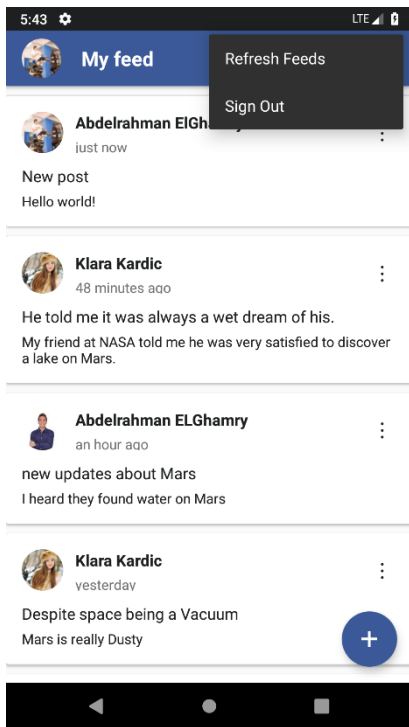
The screenshot shows the Facebook Developers Console for an app named 'Mars'. The top bar includes links for Docs, Tools, Support, and My Apps, along with a search bar. The left sidebar contains navigation links for Dashboard, Settings, Roles, Test Users, Alerts, App Review, PRODUCTS (Facebook Login, Analytics), and Activity Log. The main content area is titled 'Test Users' and contains a table of test users. The table has columns for Name, User ID, Email, and Last Updated. There are three test users listed: Klara Kardic, Alexa Adams, and John Dee. Each user has an 'Edit' button next to their name. The table also has 'Delete' and 'Add' buttons at the top right. The last updated times are January 8, 2021 4:51 AM for Klara Kardic, January 8, 2021 4:31 AM for Alexa Adams, and January 6, 2021 22:44 PM for John Dee.

Name	User ID	Email	Last Updated
Klara Kardic	104626781592254	klara_rwsrmw_kardic@tfbnw.net	January 8, 2021 4:51 AM
Alexa Adams	103660398358335	alexa_1stwxpd_adams@tfbnw.net	January 8, 2021 4:31 AM
John Dee	106068304775451	john_jsufddg_dee@tfbnw.net	January 6, 2021 22:44 PM

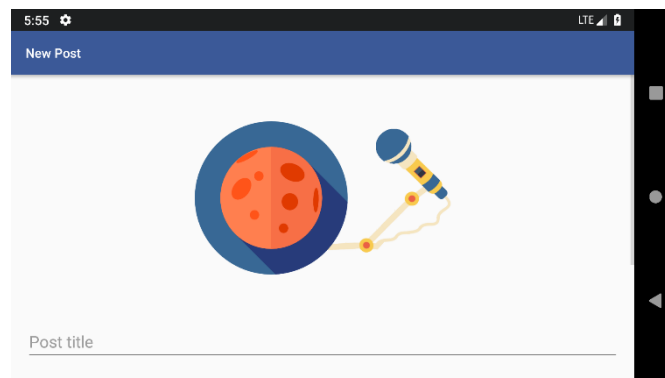
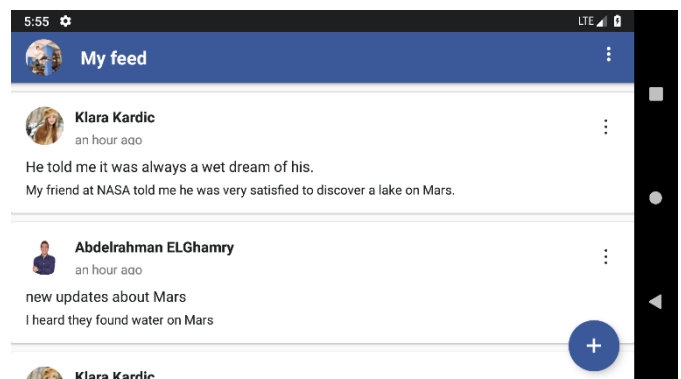
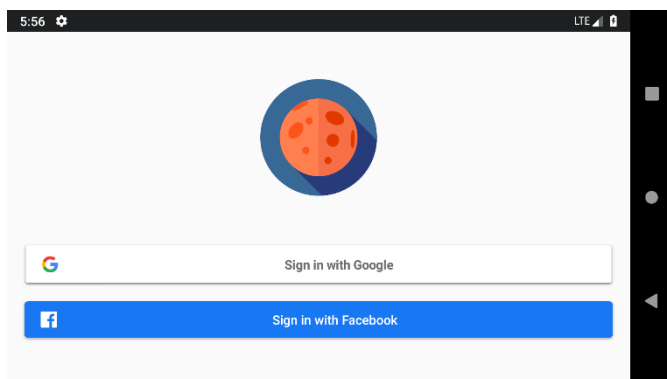
5.0 APP DESIGN

5.1 Portrait





5.2 Landscape



6.0 APP FLOW AND CORNER CASES

6.1 Authentication Scenario

- The app starts by checking if this user is authenticated or not.
- If the user is authenticated, the app automatically navigates to the Posts Activity, otherwise, the app navigates to the Authentication Activity, and then user authenticates using any of the available authentication options.

6.2 View Feeds Scenario

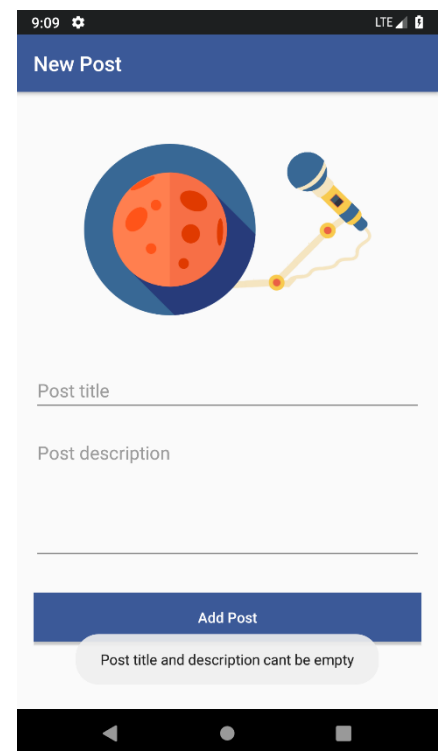
- The app requests all the existing feeds from the cloud firestore.
- If there's available data, the Posts Activity will show cards containing the posts, otherwise, it will display empty feed.

6.3 Corner Cases

The user navigates to the Add Post Activity and tries to add:

- An empty post.
- Post with no title.
- Post with no description.
- Post with either the title or description is white-spaces.

The app doesn't add the post and displays a toast message to the user that clarifies the problem.



7.0 IMPORTANT LINKS

7.1 Application (APK)

<https://drive.google.com/file/d/1oJ8FxFYamsLE7NGybfx--n7cS57CZ3US/view?usp=sharing>

7.2 Demo Video (MP4)

<https://drive.google.com/file/d/1E7sKpc2sgPFysSbdE3pLJRoO4thzkyv2/view?usp=sharing>

7.3 Source Control (GIT)

<https://github.com/Ghamry98/Mars-social-app-mvvm>

8.0 CODE SNIPPETS

8.1 Views and XML

Custom_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/refresh_feeds_menu"
        android:title="@string/refresh_feeds"
        app:showAsAction="never"/>
    <item
        android:id="@+id/sign_out_menu"
        android:title="@string/sign_out"
        app:showAsAction="never"/>
</menu>
```

PostsActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_posts);

    initViewModels();
    getUserFromIntent();
    initAppBar();
    initFab();
    initRecyclerView();

    loadPosts();
    configureFCM();
}

private void configureFCM() {
    FirebaseMessaging.getInstance().subscribeToTopic("posts");
}

private void initRecyclerView() {
    recyclerView = findViewById(R.id.postsList);
    LinearLayoutManager layoutManager = new LinearLayoutManager(this);
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
}

private void loadPosts() {
    postsViewModel.getAllPosts();
    postsViewModel.postsLiveData.observe(this, data -> {
        postsAdapter = new PostsAdapter(this, data);
        recyclerView.setAdapter(postsAdapter);
    });
}

private void getUserFromIntent() {
    user = (User) getIntent().getSerializableExtra(USER);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RC_ADD_POST && data != null) {
        Post post = (Post) data.getSerializableExtra(POST);
        post.setAuthor(user);
        postsViewModel.createPost(post);
        Toast.makeText(PostsActivity.this, R.string.add_post_success,
            Toast.LENGTH_SHORT).show();
    }
}
```

8.2 Adapters

PostsAdapter.java

```
public class PostsAdapter extends RecyclerView.Adapter<PostsAdapter.ViewHolder>
{
    private List<Post> posts;
    private LayoutInflater mInflater;
    private Context context;

    TextToSpeech tts;

    final int VOICE_RECOGNITION = 3;
    String mostRecentUtteranceID;
    int count = 0;

    public PostsAdapter(Context context, List<Post> data) {
        this.mInflater = LayoutInflater.from(context);
        this.posts = data;
        this.context = context;

        initTTS();
    }

    // inflates the row layout from xml when needed
    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = mInflater.inflate(R.layout.item_post, parent, false);
        return new ViewHolder(view);
    }

    // total number of rows
    @Override
    public int getItemCount() {
        return posts.size();
    }

    // stores and recycles views as they are scrolled off screen
    public class ViewHolder extends RecyclerView.ViewHolder {
        TextView authorName;
        ImageView authorAvatar;
        TextView title;
        TextView desc;
        TextView createdAt;
        TextView menuBtn;

        ViewHolder(View itemView) {
            super(itemView);
            authorName = itemView.findViewById(R.id.authorName);
            authorAvatar = itemView.findViewById(R.id.authorAvatar);
            title = itemView.findViewById(R.id.title);
            desc = itemView.findViewById(R.id.desc);
            createdAt = itemView.findViewById(R.id.createdAt);
            menuBtn = itemView.findViewById(R.id.menuBtn);
        }
    }

    // binds the data to the TextView in each row
    @Override
```

```

public void onBindViewHolder(ViewHolder holder, int position) {
    Post post = posts.get(position);

    holder.authorName.setText(post.authorName);
    holder.title.setText(post.title);
    holder.desc.setText(post.desc);
    holder.createdAt.setText(post.formattedCreationDate());

    Glide.with(context)
        .load(post.authorAvatar)
        .centerCrop()
        .circleCrop()
        .into(holder.authorAvatar);

    holder.menuBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Log.d("MarsTag", "onClick: ");
            PopupMenu popupMenu = new PopupMenu(context, holder.menuBtn);
            popupMenu.inflate(R.menu.post_options_menu);
            popupMenu.setOnMenuItemClickListener(new
PopupMenu.OnMenuItemClickListener() {
                @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
                @Override
                public boolean onMenuItemClick(MenuItem menuItem) {
                    switch (menuItem.getItemId()) {
                        case R.id.menu_item_read:
                            speakText(post);
                            break;
                    }

                    return false;
                }
            });
            popupMenu.show();
        }
    });

    @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
    public void speakText(Post post) {
        try {
            String TTS = post.formattedTTS();

            count++;
            mostRecentUtteranceID = Integer.toString(count) + " ID";
            tts.speak(TTS, TextToSpeech.QUEUE_ADD, null, mostRecentUtteranceID);
        } catch (Exception e) {
        }
    }

    private void initTTS() {
        tts = new TextToSpeech(context, new TextToSpeech.OnInitListener() {
            @RequiresApi(api = Build.VERSION_CODES.LOLLIPOP)
            @Override
            public void onInit(int status) {

                if (status == TextToSpeech.SUCCESS) {

```

```

        tts.setLanguage(Locale.ENGLISH);
        tts.setPitch((float) 1);
        tts.setSpeechRate((float) 1);
    }
}
});
}
}
}

```

8.3 View Models

Custom_menu.xml

```

public class PostsViewModel extends AndroidViewModel {
    private PostsRepository postsRepository;
    public LiveData<List<Post>> postsLiveData;

    public PostsViewModel(Application application) {
        super(application);
        postsRepository = new PostsRepository();
    }

    public void getAllPosts() {
        postsLiveData = postsRepository.getAllPosts();
    }

    public void createPost(Post post) {
        postsRepository.addPost(post);
    }
}

```

8.4 Repositories

PostsRepository.java

```

public class PostsRepository {
    private FirebaseFirestore rootRef = FirebaseFirestore.getInstance();
    private CollectionReference postsRef = rootRef.collection(POSTS);
    MutableLiveData<List<Post>> posts = new MutableLiveData<>();

    public MutableLiveData<List<Post>> getAllPosts() {

        postsRef.get().addOnCompleteListener(new
        OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    List<Post> list = new ArrayList<>();
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        list.add(document.toObject(Post.class));
                    }

                    Collections.sort(list, new Comparator<Post>() {
                        @Override
                        public int compare(Post p1, Post p2) {
                            return new
                            Long(p2.createdAt).compareTo(p1.createdAt);
                        }
                    });
                }
            }
        });
    }
}

```



```

        });
        posts.setValue(list);
    } else {
        LogErrorMessage(task.getException().getMessage());
    }
}

});
return posts;
}

public void addPost(Post post) {
    DocumentReference docRef = postsRef.document();
    docRef.set(post);
    List<Post> ps = posts.getValue();
    ps.add(0, post);
    posts.setValue(ps);
}
}

```

8.5 Services

```

MyFirebaseMessagingService.java
public class MyFirebaseMessagingService extends FirebaseMessagingService {

    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        Log.d(TAG, "From: " + remoteMessage.getFrom());

        if (remoteMessage.getData().size() > 0) {
            Log.d(TAG, "Message data payload: " + remoteMessage.getData());

            /* Check if data needs to be processed by Long running job */
            if (true) {
                // For Long-running tasks (10 seconds or more) use WorkManager.
                scheduleJob();
            } else {
                // Handle message within 10 seconds
                handleNow();
            }
        }

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            Log.d(TAG, "Message Notification Body: " +
remoteMessage.getNotification().getBody());
        }
        sendNotification(remoteMessage.getNotification().getTitle(),
remoteMessage.getNotification().getBody());
    }

    @Override
    public void onNewToken(String token) {
        Log.d(TAG, "Refreshed token: " + token);
        sendRegistrationToServer(token);
    }
}

```

```

    }

    private void sendRegistrationToServer(String token) {
        // Implement this method to send token to your app server.
    }

    private void scheduleJob() {
        //      OneTimeWorkRequest work = new
        OneTimeWorkRequest.Builder(MyWorker.class)
        //          .build();
        //      WorkManager.getInstance().beginWith(work).enqueue();
    }

    private void handleNow() {
        //      Log.d(TAG, "Short lived task is done.");
    }

    private void sendNotification(String messageTitle, String messageBody) {
        Intent intent = new Intent(this, SplashActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 ,
intent,
                PendingIntent.FLAG_ONE_SHOT);

        String channelId = getString(R.string.default_notification_channel_id);
        Uri defaultSoundUri =
        RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder notificationBuilder =
            new NotificationCompat.Builder(this, channelId)
                .setSmallIcon(R.mipmap.ic_launcher_foreground)
                .setContentTitle(messageTitle != null &&
messageTitle.length() > 0 ? messageTitle : getString(R.string.app_name))
                .setContentText(messageBody)
                .setAutoCancel(true)
                .setSound(defaultSoundUri)
                .setContentIntent(pendingIntent);

        NotificationManager notificationManager =
            (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

        // Since android Oreo notification channel is needed.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel(channelId,
                "Channel human readable title",
                NotificationManager.IMPORTANCE_DEFAULT);
            notificationManager.createNotificationChannel(channel);
        }

        notificationManager.notify(0 /* ID of notification */,
notificationBuilder.build());
    }
}

```

8.6 Entities

User.java

```
public class User implements Serializable {
    public String uid;
    public String name;
    @SuppressWarnings("WeakerAccess")
    public String email;
    public String avatar;
    @Exclude
    public boolean isAuthenticated;
    @Exclude
    public boolean isNew, isCreated;

    public User() {}

    public User(String uid, String name, String email, String avatar) {
        this.uid = uid;
        this.name = name;
        this.email = email;
        this.avatar = avatar;
    }

    public String toString() {
        return "id: " + uid
            + " name: " + name
            + " email: " + email
            + " avatar: " + avatar
            + " isAuthenticated: " + isAuthenticated
            + " isNew: " + isNew
            + " isCreated: " + isCreated;
    }
}
```

8.7 Utils

Helpers.java

```
public class Helpers {
    private static int SECOND_MILLIS = 1000;
    private static int MINUTE_MILLIS = 60 * SECOND_MILLIS;
    private static int HOUR_MILLIS = 60 * MINUTE_MILLIS;
    private static int DAY_MILLIS = 24 * HOUR_MILLIS;

    public static String getTimeAgo(Long time) {
        Long now = System.currentTimeMillis();
        if (time > now || time <= 0) {
            return "";
        }

        Long diff = now - time;
        if (diff < MINUTE_MILLIS) {
            return "just now";
        } else if (diff < 2 * MINUTE_MILLIS) {
            return "a minute ago";
        } else if (diff < 50 * MINUTE_MILLIS) {
            return (diff / MINUTE_MILLIS) + " minutes ago";
        } else if (diff < 90 * MINUTE_MILLIS) {
            return (diff / HOUR_MILLIS) + " hours ago";
        } else if (diff < 24 * HOUR_MILLIS) {
            return (diff / DAY_MILLIS) + " days ago";
        } else {
            return (diff / (24 * 60 * 60 * 1000)) + " days ago";
        }
    }
}
```

```

        return "an hour ago";
    } else if (diff < 24 * HOUR_MILLIS) {
        return (diff / HOUR_MILLIS) + " hours ago";
    } else if (diff < 48 * HOUR_MILLIS) {
        return "yesterday";
    } else {
        return (diff / DAY_MILLIS) + " days ago";
    }
}

public static void logErrorMessage(String errorMessage) {
    Log.d(TAG, errorMessage);
}
}

```