



**UNIVERSITI  
MALAYA**

**WIA 1002**

**DATA STRUCTURE AND ALGORITHM**

**ASSIGNMENT TECHNICAL REPORT**

**GROUP MEMBERS : CHOO XIANG LUN (S2132552)**  
**CHONG JIA XUAN (U2102725)**  
**JASON WONG JACK (U2102864)**  
**LIM JIAJUN (S2124035)**

**LECTURER NAME : DR. HOO WAI LAM**

**GROUP NO : GROUP 1**

**TITLE : CONFESS TIME!**

## TABLE OF CONTENTS

<b>No.</b>	<b>Content</b>	<b>Page</b>
<b>1.</b>	<b>Table of Contents</b>	<b>1</b>
<b>2.</b>	<b>Introduction</b>	<b>2</b>
<b>3.</b>	<b>Basic Requirements</b>	<b>2-16</b>
<b>4.</b>	<b>Extra Features</b>	<b>17-23</b>
<b>5.</b>	<b>Project Flow Chart</b>	<b>24-28</b>

## Introduction to Confess-Ba:

People have to play many different roles and wear many different hats in life. Having all these duelling identities can be a stressful ordeal and finding a place to voice out bottled inner thoughts is definitely a good way to release all the stress. Here is the function of Confess-Ba. The confession page is a good place to voice out what needs to be said. People ask or talk about anything on confession pages, whether life-related questions, homework, or even career suggestions.

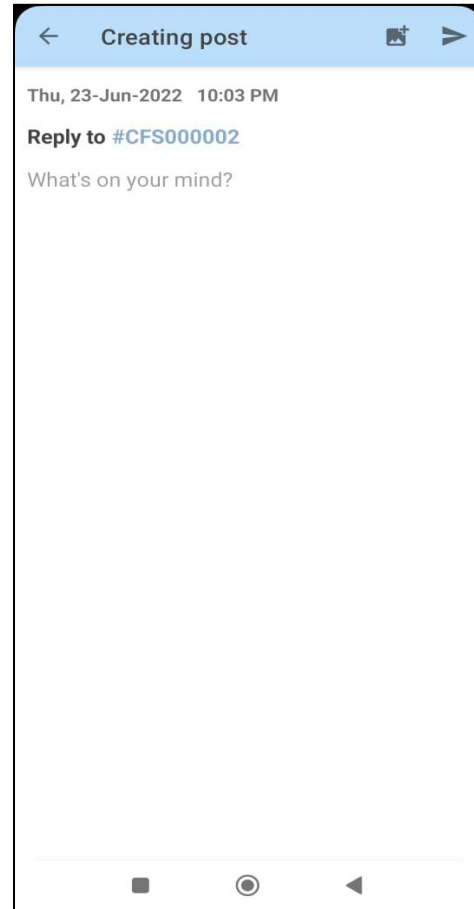
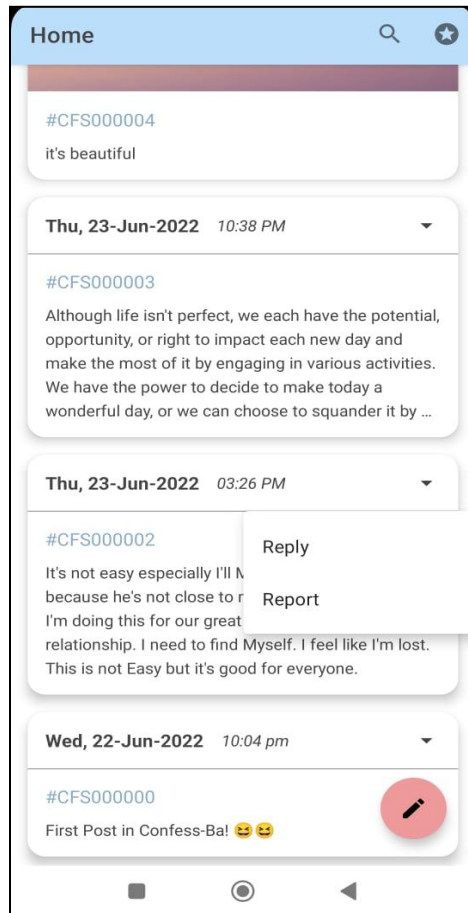
## Basic Requirements of Confess-Ba:

### A) Main Application Features

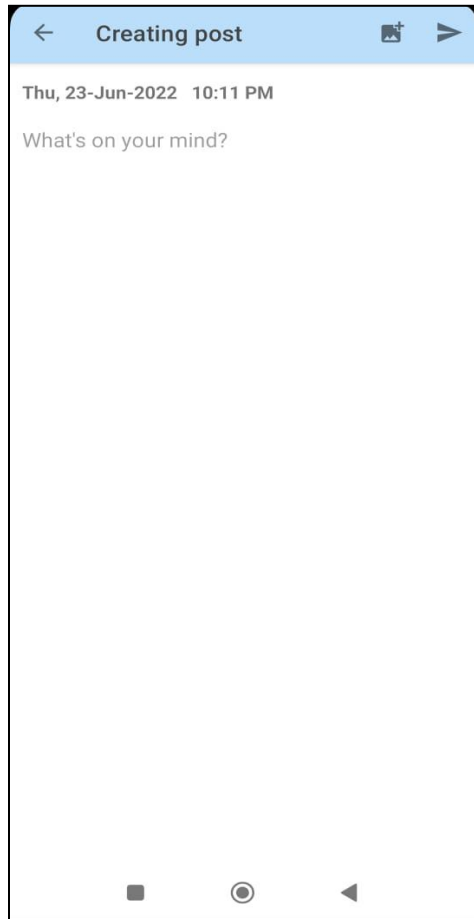
#### 1. Submit Confession Post

Users can submit their confessions. There will be two input fields for the user, users can either choose to reply to a post by entering an existing confession post ID or create a new confession post by leaving it blank. The second input field is the confession content. This field is mandatory for the user to be filled in upon creating a new post. Once the users have submitted their confession page, they will be notified by a message which contains the confession post ID and the submission date and time that their post will be published soon.

a) Replying to a confession post.



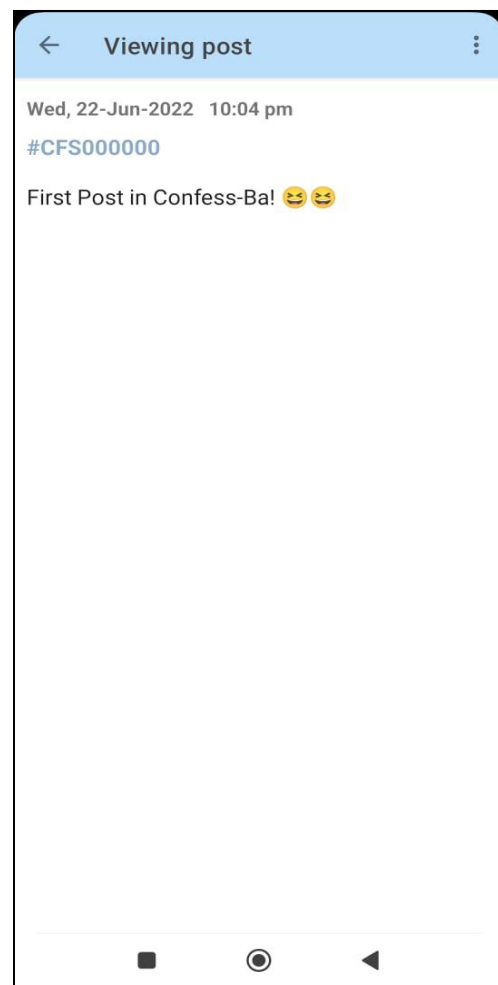
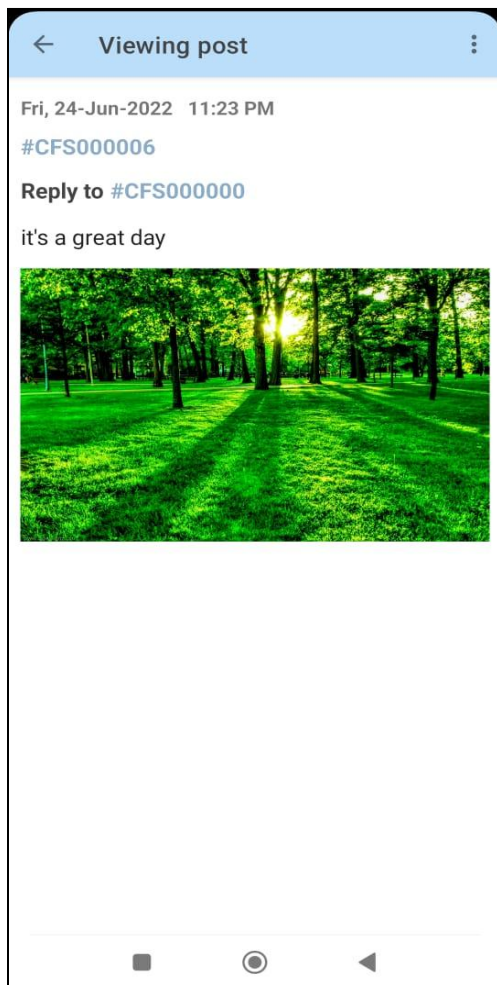
b) Creating a new confession post



## 2. View Published Confession Posts

Confess-Ba feed shows the confession page starting from the latest. The contents of a published confession post includes:

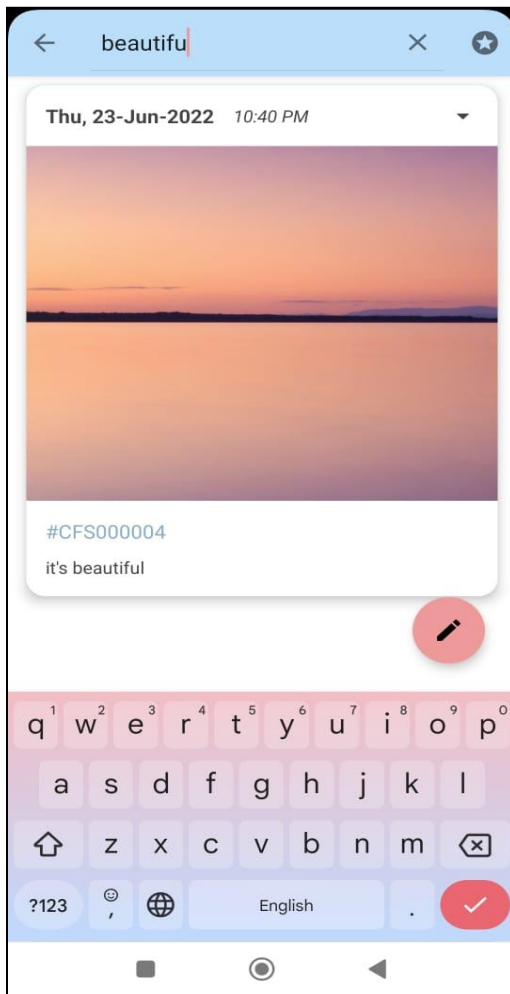
- a) Its confession post ID
- b) Published date & time
- c) Reply confession post ID (if exist)
- d) Confession content



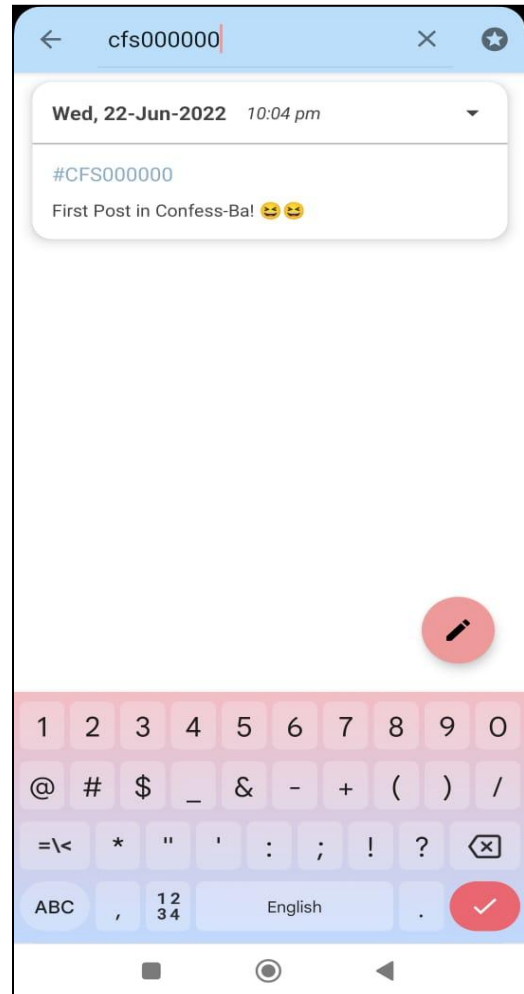
### 3. Search Published Confession Posts

Users can search the published confession post using confession post ID, published date or published date and time, keywords. The returned search result contains multiple confession posts.

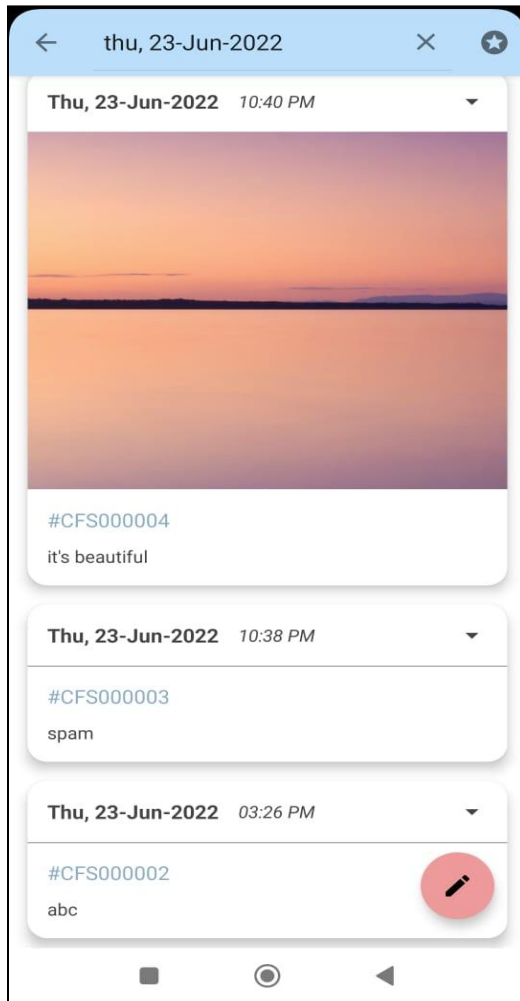
a) Search post using content



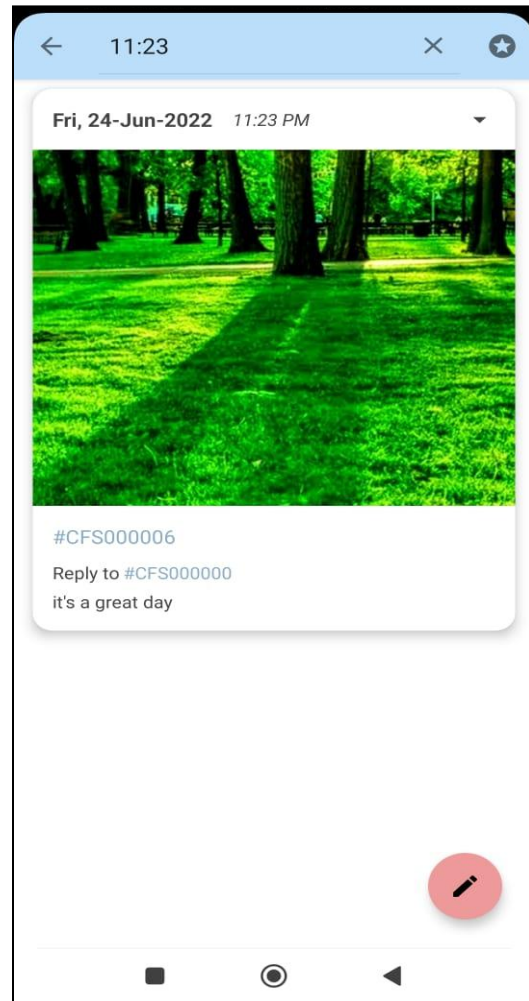
b) Search post using id



c) Search post using published date



d) Search post using published date and time



For the implementation, we iterate through every single post from our database and compare the information of the post with the inputs from the user.

```
for (PostModel post : postList) {  
    String input = str.toLowerCase().trim();  
    if ("#" + post.getId().toLowerCase().contains(input) || post.getContent().toLowerCase().contains(input)  
        || post.getTime().toLowerCase().contains(input) || post.getDate().toLowerCase().contains(input)) {  
        // add the post to the filtered list, to show it in the recycler view  
    }  
}
```



```

        filteredList.add(post);
    }
}

```

## B) Admin Panels and System

### 1. Waiting / Queueing list

Simulate and automate the admin approval time, where the waiting time is 15 minutes, 10 minutes and 5 minutes, when the number of elements in the list is equal or less than 5, equal of less than 10, or greater than 10 respectively.

Implementation wise, we will open up a background thread, once the user launch the application. In this background thread, we use the Java Timer and TimerTask utility class to schedule a Runnable task which will check the content of the waiting list in every second. The figure below shows the corresponding code.

```

@Override
public void onCreate() {
    // create background thread for waiting list when the application was launched
    BackgroundRunnable backgroundRunnable = new BackgroundRunnable();
    new Thread(backgroundRunnable).start();
}

// Implement the Runnable task for the background thread
class BackgroundRunnable implements Runnable {
    @Override
    public void run() {
        Timer timer = new Timer();
        TimerTask timerTask = new TimerTask() {
            long elapsedSec = 0;

            @Override
            public void run() {
                int waitingListSize = waitingList.size();
                // print out the information of the waiting list
                System.out.println("The waiting list size is " + waitingListSize + " and the elapsed time is " + elapsedSec);
                if (waitingListSize == 0) {
                    elapsedSec = 0;
                } else if (waitingListSize > 10) {
                    // wait for 5 min (300 sec)
                    if (elapsedSec >= 300) {
                        // if elapse time is 5 min or more
                        PostModel post = waitingList.poll();
                        waitingListContent.poll();
                    }
                }
            }
        };
        timer.schedule(timerTask, 0, 1000);
    }
}

```

```

        if (post != null) {
            savePostToDatabase(post);
        }
        elapsedSec = 0;
    }
} else if (waitingListSize > 5) {
    // wait for 10 min (600 sec)
    if (elapsedSec >= 600) {
        // if elapse time is 10 min or more
        PostModel post = waitingList.poll();
        waitingListContent.poll();
        if (post != null) {
            savePostToDatabase(post);
        }
        elapsedSec = 0;
    }
} else {
    // wait for 15 min (900 sec)
    if (elapsedSec >= 900) {
        // if elapse time is 15 min or more
        PostModel post = waitingList.poll();
        waitingListContent.poll();
        if (post != null) {
            savePostToDatabase(post);
        }
        elapsedSec = 0;
    }
}
// increase elapse time for each loop
elapsedSec++;
}
};
// call timerTask every 1 sec
timer.scheduleAtFixedRate(timerTask, 0, 1000);
}

```

Output:

- I. When the waiting list is empty, the elapsed time will keep reset to zero in every second.

```
I/System.out: The waiting list size is 0 and the elapsed time is 1
I/System.out: The waiting list size is 0 and the elapsed time is 1
I/System.out: The waiting list size is 0 and the elapsed time is 1
I/System.out: The waiting list size is 0 and the elapsed time is 1
I/System.out: The waiting list size is 0 and the elapsed time is 1
I/System.out: The waiting list size is 0 and the elapsed time is 1
```

- II. When the waiting list size is 5 or less, the data will be pop out from the waiting list every 15 minutes.

```
I/System.out: The waiting list size is 1 and the elapsed time is 10
I/System.out: The waiting list size is 1 and the elapsed time is 11
I/System.out: The waiting list size is 1 and the elapsed time is 12
I/System.out: The waiting list size is 1 and the elapsed time is 13
I/System.out: The waiting list size is 1 and the elapsed time is 14
I/System.out: The waiting list size is 1 and the elapsed time is 15
I/System.out: The data was popped out of the waiting list after 15 min
```

- III. When the waiting list size is 10 or less, the data will be pop out from the waiting list every 10 minutes.

```
I/System.out: The waiting list size is 10 and the elapsed time is 6
I/System.out: The waiting list size is 10 and the elapsed time is 7
I/System.out: The waiting list size is 10 and the elapsed time is 7
I/System.out: The waiting list size is 10 and the elapsed time is 8
I/System.out: The waiting list size is 10 and the elapsed time is 9
I/System.out: The waiting list size is 10 and the elapsed time is 10
I/System.out: The data was popped out of the waiting list after 10 min
```

- IV. When the waiting list size is more than 10, the data will be pop out from the waiting list every 5 minutes.

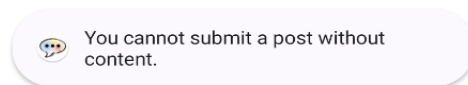
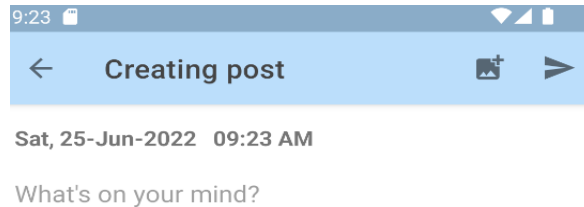
```
I/System.out: The waiting list size is 11 and the elapsed time is 4
I/System.out: The waiting list size is 11 and the elapsed time is 5
I/System.out: The data was popped out of the waiting list after 5 min
```

## **2. Spam Checking**

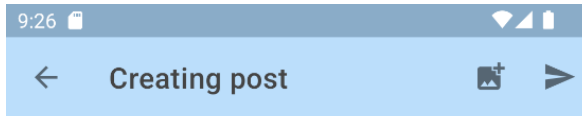
As a requirement from the question, we need to have some basic spam checking to avoid the confession page having repeatedly submitted posts. Therefore, we have implemented two checking methods: whenever users submit a post without content, there will be a pop-up message stopping the user from submitting an empty content, and another one if the system detects the user submitted a duplicated post that already exists in the waiting list, there will be a pop-up message indicating the user to not submit a repeated post. The rationale behind this is that sometimes users' gadget may have a poor connection, which causes the user to think that they have not submitted the post whereas the post has already been pushed to the waiting list

Output:

- I. User cannot submit a contentless post. A pop-up message will be shown if the user attempts to submit a contentless post. This is implemented by simple validation to check whether the input fields are empty.

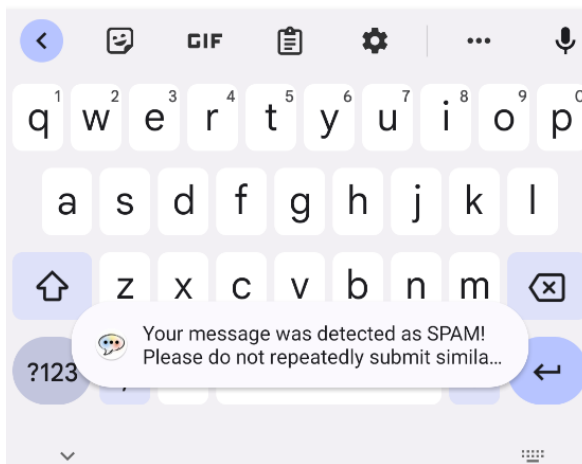


- II. Users cannot submit a repeated content post. A pop-up message will be shown to ask the user to not repeatedly submit a similar content. This is done by looping through each Post element in the waiting list and checking whether the content is the same as the input.



Sat, 25-Jun-2022 09:26 AM

I am so happy!



### 3. Batch Removal

Allow admin to batch remove a post that some posts that violate the terms and conditions of the platform. All the children of the selected post should be removed simultaneously.

For this, we use the Depth First Search (DFS) algorithm that consists of the following steps:

1. Get the information about the current post's children list from the database.
2. Delete the current post.
3. Delete all its children post, using DFS (recursion)
4. Remove the presence of the current post from its parent's children list

```
/**
 * Batch removing the targeted post and its children using DFS
 */
private void batchRemoval(String id, String replyId) {
    // access database & storage reference
    FirebaseDatabase firebaseDatabase =
    FirebaseDatabase.getInstance("https://confession-android-app-default-rtdb.asia-southeast1.firebaseio.com");
    ;
    DatabaseReference postNode = firebaseDatabase.getReference().child("submitted_posts");
    DatabaseReference reportedPostNode = firebaseDatabase.getReference().child("reported_posts");
    FirebaseStorage firebaseStorage = FirebaseStorage.getInstance();

    // perform deletion using dfs to delete itself and all of its children
    dfsDelete(id, postNode, reportedPostNode, firebaseStorage);

    // remove its presence in its parent's repliedBy list
    if (replyId != null) {
        postNode.child(replyId).get().addOnCompleteListener(task -> {
            DataSnapshot snapshot = task.getResult();
            if (snapshot.exists()) {
                PostModel postModel = snapshot.getValue(PostModel.class);
                List<String> newRepliedByList = Objects.requireNonNull(postModel).getRepliedBy();
                newRepliedByList.remove(id);
                HashMap<String, Object> hashMap = new HashMap<>();
                hashMap.put("repliedBy", newRepliedByList);
                postNode.child(replyId).updateChildren(hashMap);
            }
        });
    }
    Toast.makeText(context, "Batch remove successfully!", Toast.LENGTH_SHORT).show();
}

/**
 * The helper method to remove the current post and all of its children post
 */
private void dfsDelete(String id, DatabaseReference postNode, DatabaseReference reportedPostNode,
    FirebaseStorage firebaseStorage) {
    postNode.child(id).get().addOnCompleteListener(task -> {
```

```

DataSnapshot snapshot = task.getResult();
if (snapshot.exists()) {
    PostModel currPost = snapshot.getValue(PostModel.class);

    // delete current post from post collection in the database
    postNode.child(Objects.requireNonNull(currPost).getId()).removeValue();

    // delete current post from reported post collection in the database
    reportedPostNode.child(currPost.getId()).removeValue();

    // remove all current post's images from firebase storage
    List<String> imagePaths = Objects.requireNonNull(currPost).getImagePaths();
    if (imagePaths != null) {
        for (String path : imagePaths) {
            firebaseStorage.getReferenceFromUrl(path).delete();
        }
    }

    // get access to current post's children list and call dfsDelete on each child posts
    List<String> repliedByList = Objects.requireNonNull(currPost).getRepliedBy();
    if (repliedByList != null) {
        for (String replyingId : repliedByList) {
            dfsDelete(replyingId, postNode, reportedPostNode, firebaseStorage);
        }
    }
}
});
}

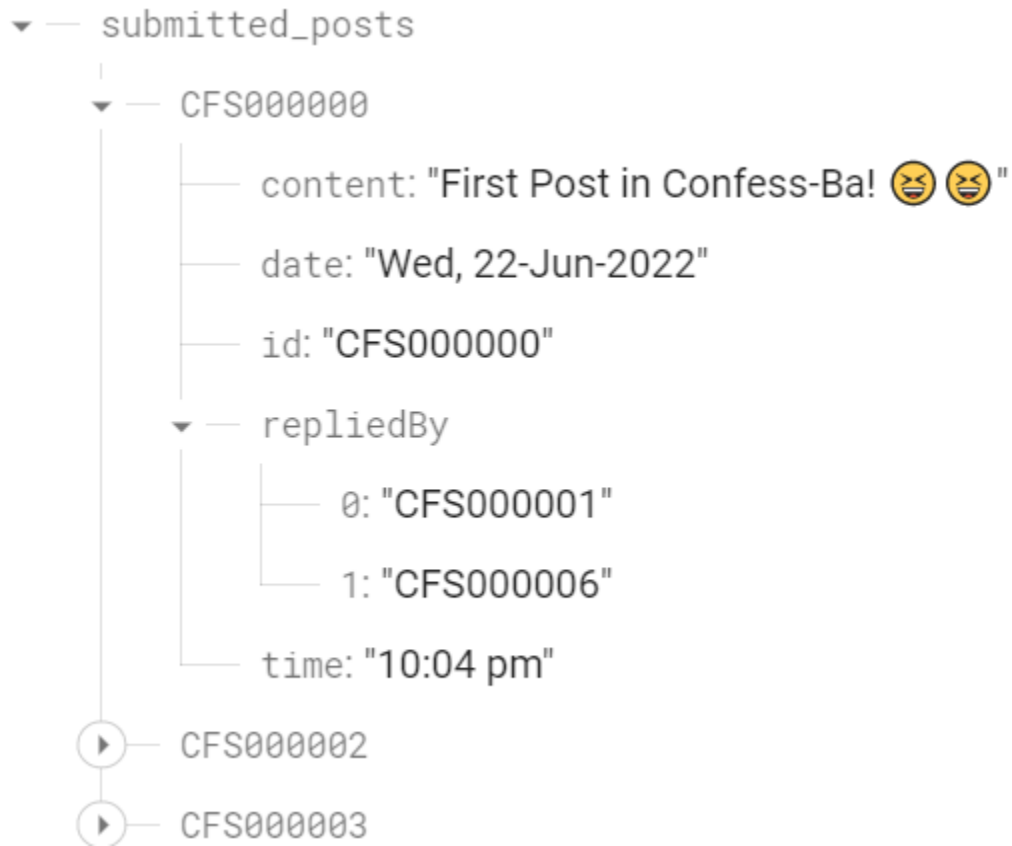
```

#### 4. Save and Load Files

All of the application's information is stored online by using the Firebase Realtime database, and Firebase Storage for storing images.



Database structure for storing posts:







Images stored using Firebase Storage:

gs://confession-android-app.appspot.com > postImages

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	 image:39144	789.25 KB	image/jpeg	23 Jun 2022
<input type="checkbox"/>	 image:39360	635.82 KB	image/jpeg	24 Jun 2022

 image:39144



Name  
[image:39144](#)

Size  
808,192 bytes

Type  
image/jpeg

Created  
23 Jun 2022, 22:41:27

Updated  
23 Jun 2022, 22:41:27

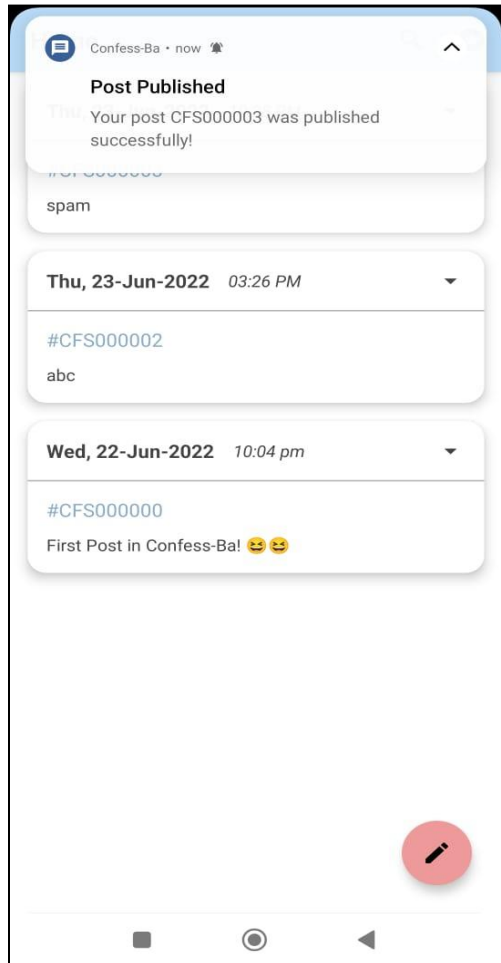
File location

Other metadata

## Extra features:

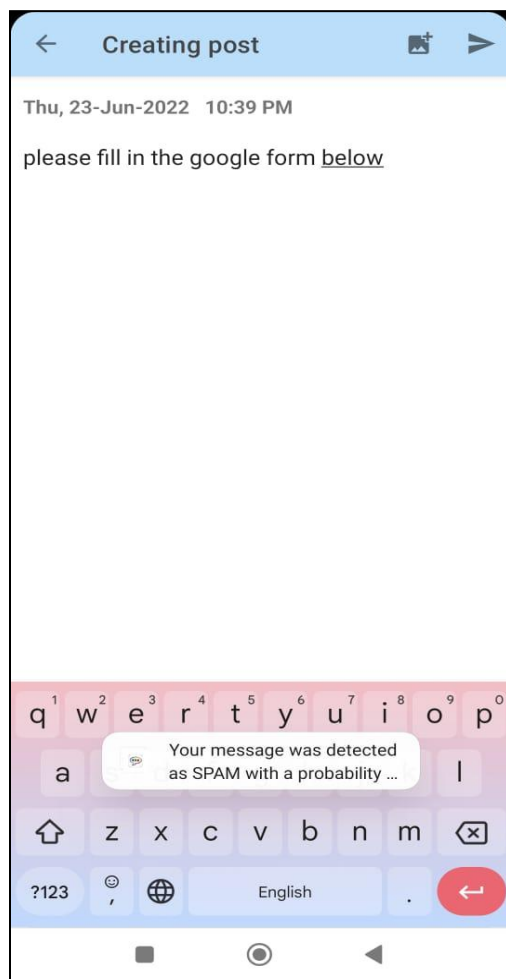
### 1. Notification System.

As we are implementing a Waiting List system where the confession will be posted after a certain time (eg. 15 minutes), the notification system plays a role to alert the user that the post has been successfully published. Sometimes some post may violate the guideline and may be taken down, so this notification system can notify the user if their post has been uploaded.



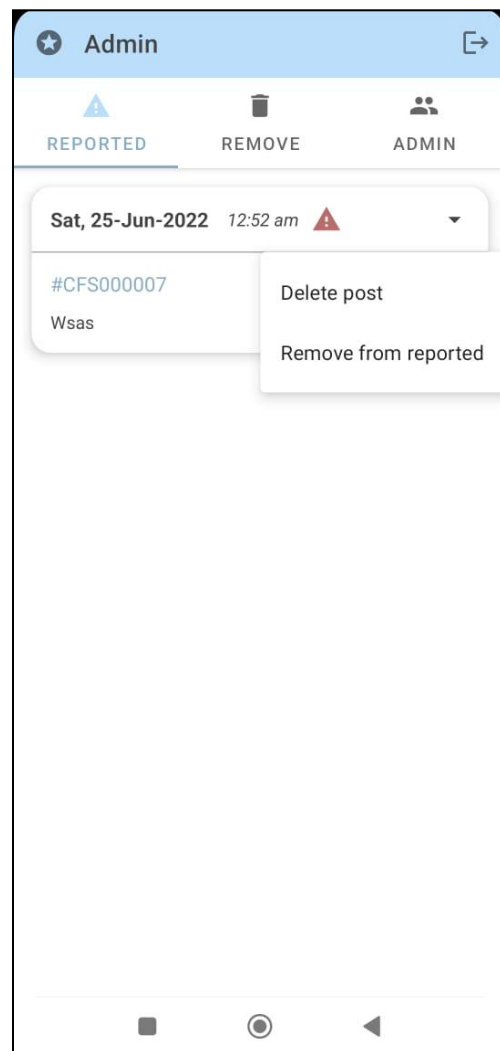
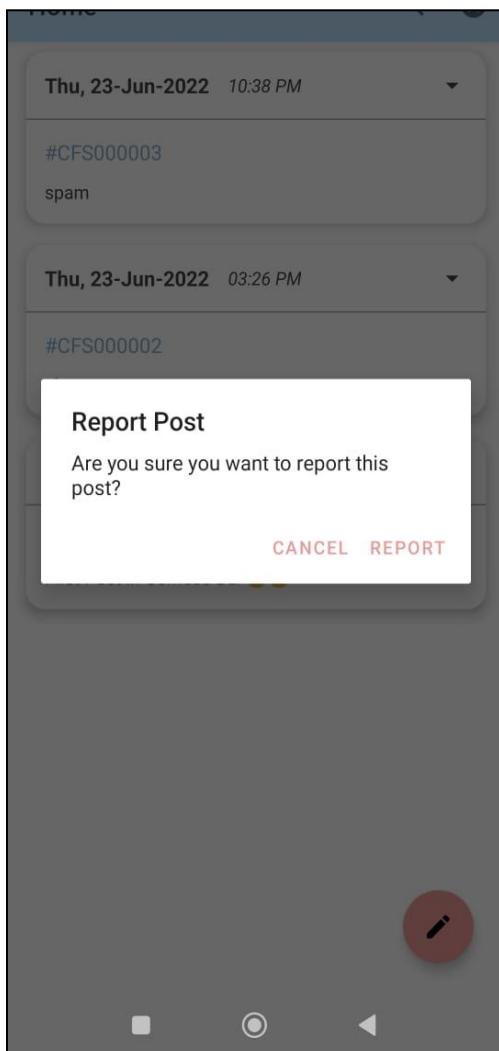
## 2. Spam detection for fraudulent or promotions content

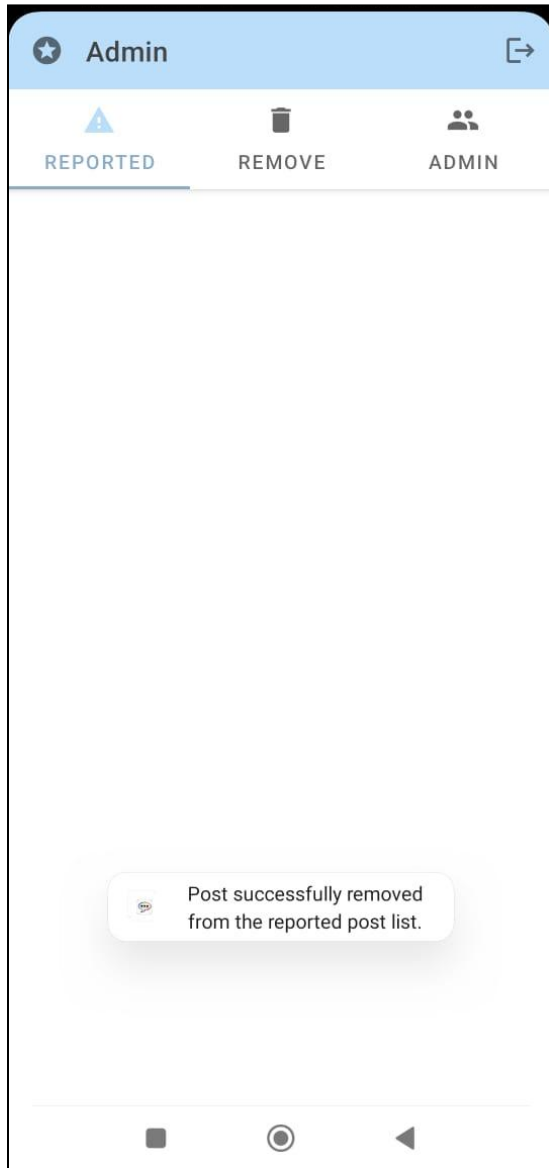
Feeling tired of promotions or survey forms that are being published inappropriately on the existing Confession page on Facebook? In Confess-Ba, we incorporated a text classification SPAM machine learning model into our application that we trained using TensorFlow and TensorFlow Lite Model Maker. This model is trained by using the dataset which is analysed from thousands of Youtube videos' comments. This model is able to accurately classify fraudulent or promotional content, thus preventing posts with this content to be published on our platform.



### 3. Report post feature.

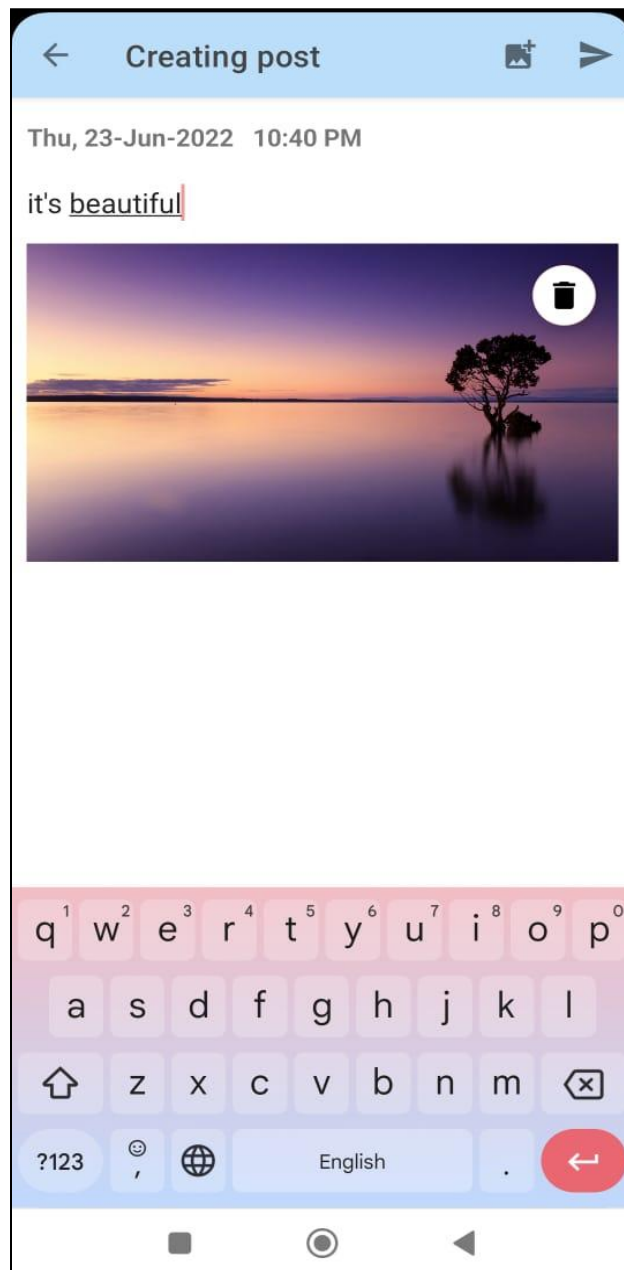
Users can easily report any abusive content by just clicking on the drop-down button on the top right of each post. Besides, the admin can view the list of reported posts. Here, the admin can choose to batch remove the post permanently, or to just remove the reported post from the reported list. This provides a way for the user to report any inappropriate and abusive posts on our platform.





#### 4. Add image feature,

We would not want our confession page to be filled with text only as this will cause boredom. So we implemented this feature so that users can attach one or more images for each post. They can delete any selected image by clicking on the corresponding trash-can button on the top right corner.

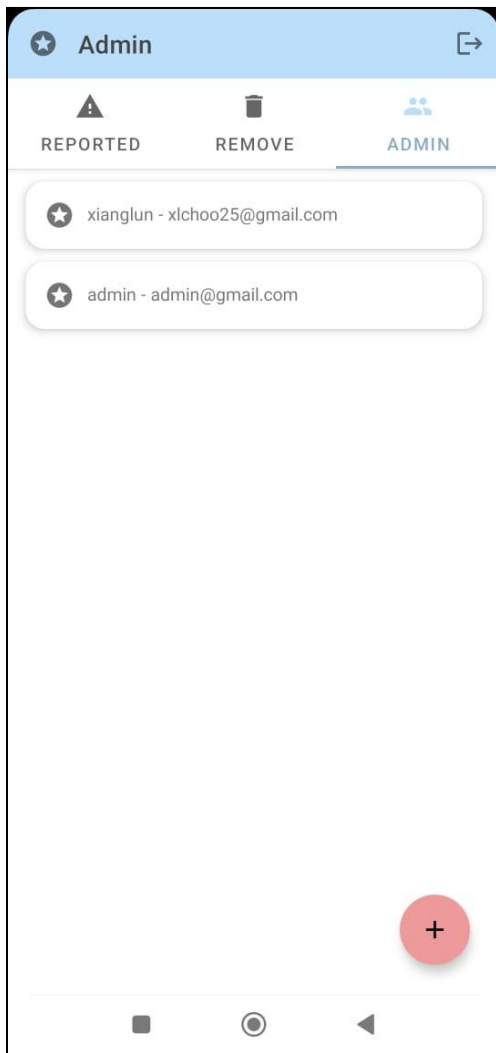


5. Implements Graphical User Interface (GUI) by making an Android app

In this 21st century, mobile phones are a must-have gadget where almost everyone has access to at least one mobile phone. In regard to our assignment topic, we need to create one confession page suitable for all levels of society no matter if you are a student, a working adult or a teacher. So here we have come up with the idea to develop a mobile app named Confess Ba where everyone has the ease of accessing this mobile app. With just a click button, people eventually can enter our interface. Unlike web pages, they might need to enter the URL or log in to their Facebook to enter the confession page.

## 6. Admin member list.

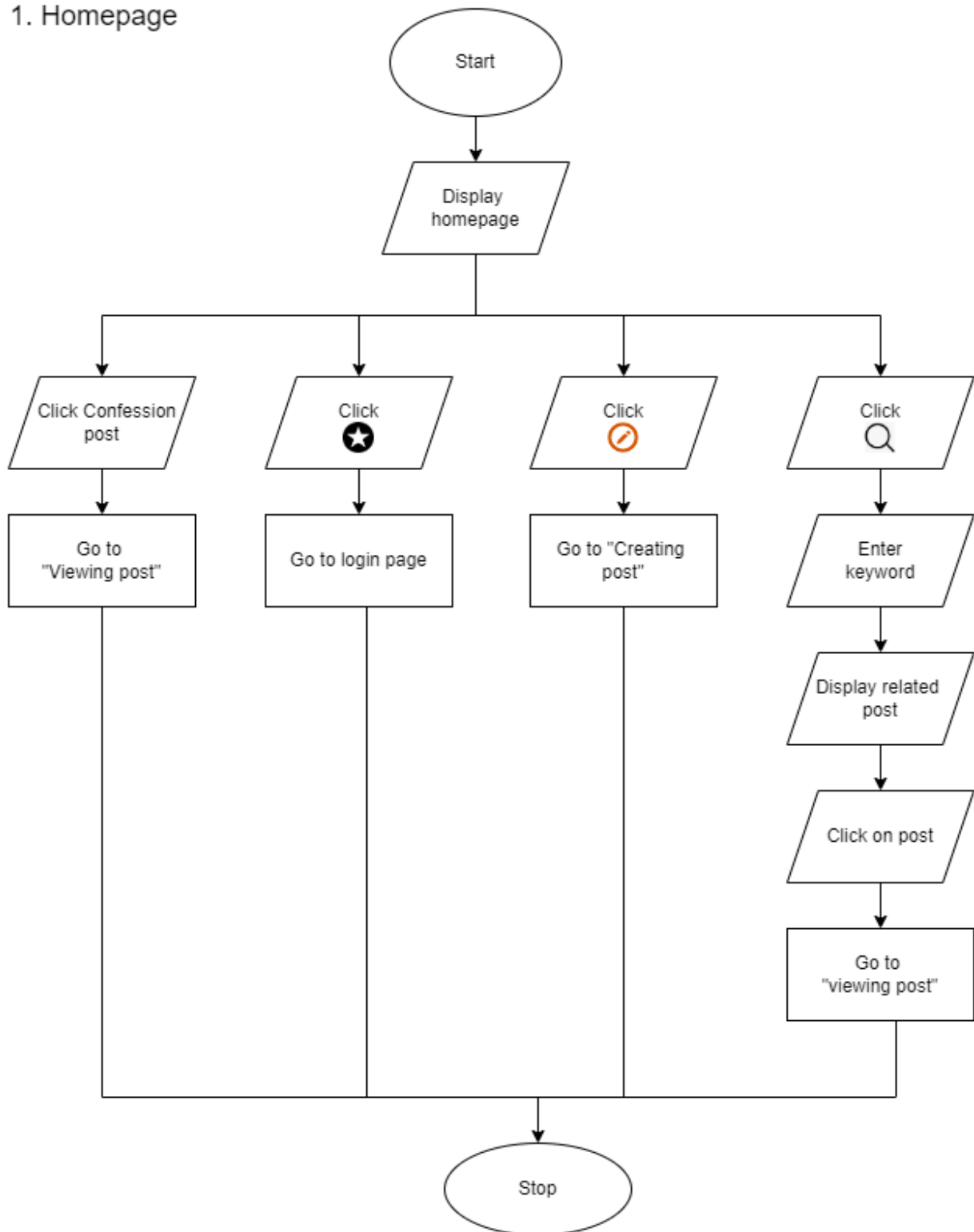
The confession is managed by several admins. The admin member list allows the admin to easily view other existing admins. The existing admins can also add other admins by entering their username, email, and password. The admin panels function to ensure the community guidelines are being followed properly. They have the authority to remove any posts that contain unsuitable content.



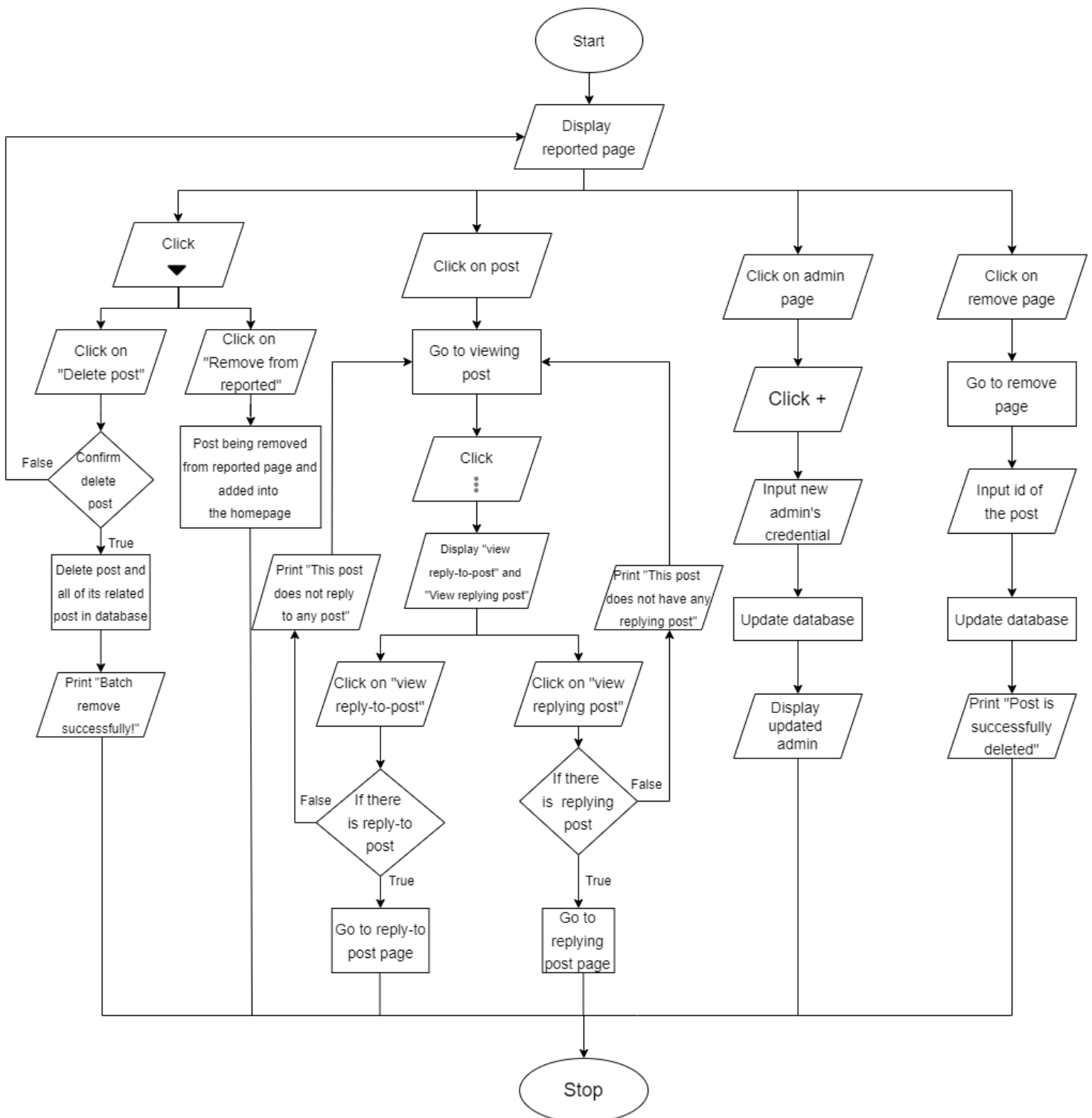


## Project Flow Chart:

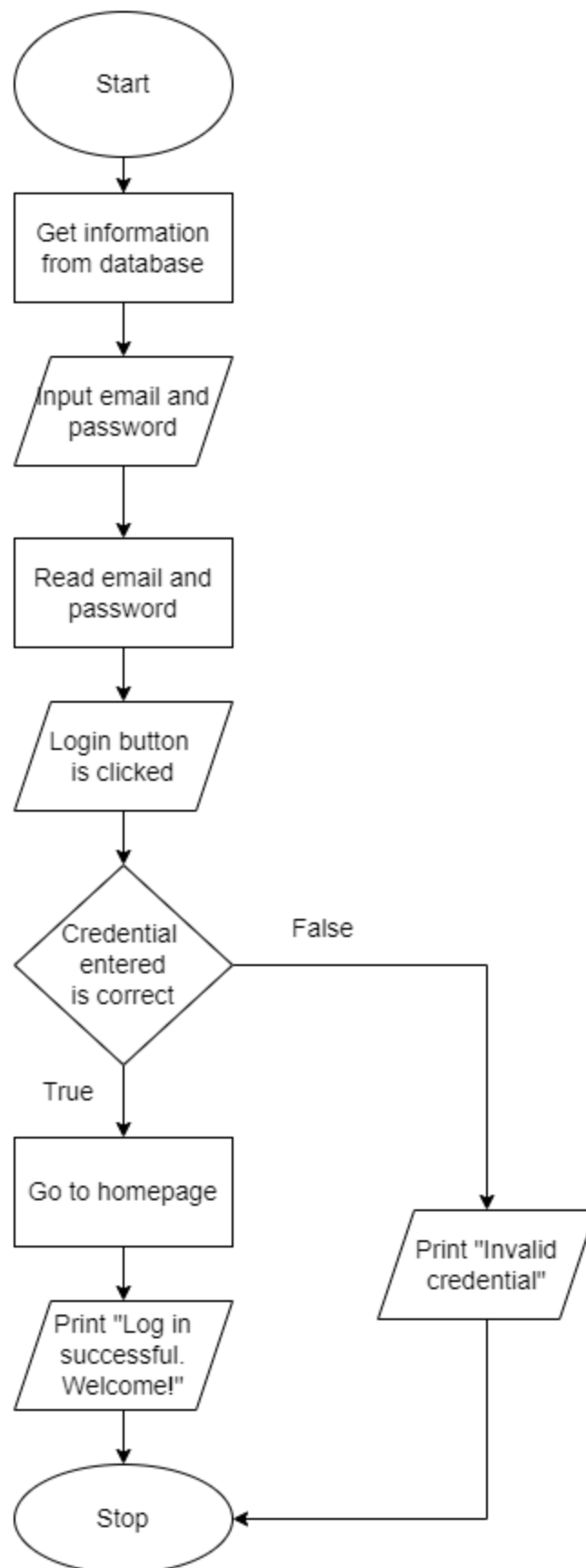
### 1. Homepage



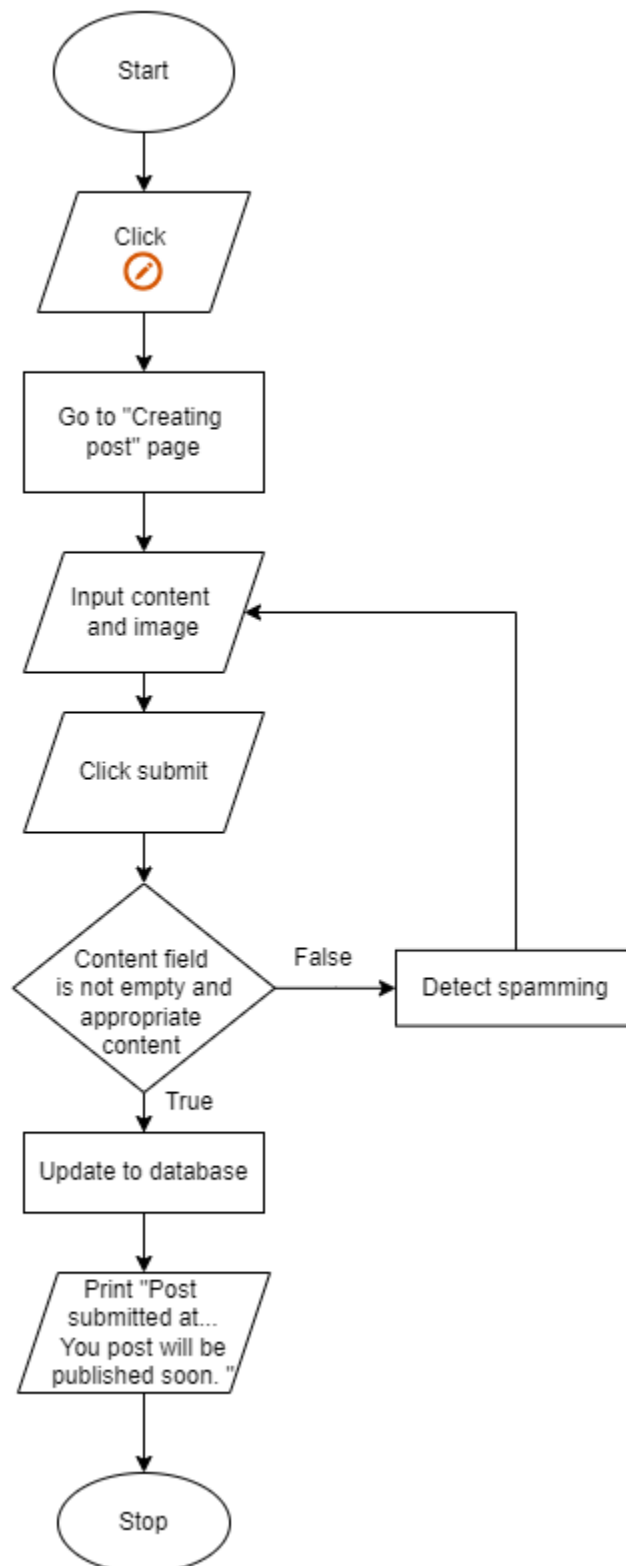
## 2. Admin page



### 3. Login page



#### 4. Creating post



## 5. Viewing post

