SWEN FORUMS

SWEN3002 – ANDROID APPLICATION DEVELOPMENT II

Abstract

This report describes the development of a Discussion Forum application. The application is targeted at the students enrolled in the BSc. Software Engineering programme offered by The University of the West Indies-Chinese Institute of Technology (UWI-CIIT).

CodeMetric Technologies

codemetricswen@gmail.com

Table of Content

ANDROID APPLICATION DEVELOPMENT II: Report

Lecturer:
Pro. Thomas Canhao Xu

Group Name:
CodeMetric Technologies

Group Leader:
Scott Allen

Other Members:
Khadeja Clarke
Mischka Neill

Table of Tasks/Responsibilities

Student ID No.		Student Name	Tasks
UWI	GIST		
620099245	180923	Allen, Scott Nicholas	Post Comments
			Database Setup
620098443	180908	Clarke, Khadeja Renee	User Authentication
			Post Categories
			Viewing a Specific Post
620100135	180913	Neill, Mischka Zoila	Viewing All Posts
			Creating a New Post

Introduction

Purpose of this Document

The main purpose of this document is to provide a detailed explanation of this project from start to finish.

Intended Audience

This document is intended for the lecturers of The University of the West Indies (UWI) and Global Institute of Software Technology (GIST).

Scope

This document should provide the intended audience with details about the project's lifecycle.

Background Information

Purpose of this Project

To fulfil the requirements of the course SWEN3002 – Android Application Development II.

Design Description

Overview

A native Android application built using Java and Kotlin.

Detailed Description

A discussion forum application which allows for the creation, viewing, edition, reporting, sharing and deletion of forum posts stored in a Firebase Realtime Database.

Goals

User Authentication

Register new users and allowing existing users to login.

Creating a new post

Create a new forum post entry in the Firebase Realtime Database.

Viewing all posts in a specific category
Retrieve all records from the Firebase Realtime Database that belong to the specified category.

Viewing a specific post
Retrieve a single specified record from the Firebase Realtime Database.

Editing a post
Make changes to the post being currently viewed, in the Firebase Realtime Database.

Sharing a post
Share the post being viewed with other applications.

Reporting a post
Remove the bulletin currently being viewed, from the Firebase Realtime Database.

Deleting a post
Remove the bulletin currently being viewed, from the Firebase Realtime Database.

Implementation

User Authentication

Registration

Registration is a two-step process. The two steps involved (in chronological order) are:

1. SEED Authentication
2. Create Firebase User with Email and Password

SEED Authentication

The purpose of this is to validate that the user falls within the target users of our application i.e. the students enrolled at UWI-CIIT. The application takes the values extracted from the EditText widgets and sends them to SEED for authentication. This is enabled by Jsoup, an open-source Java HTML parser. On a successful login to SEED, the user's GIST email address is generated. We also sign out of SEED because we no longer have need of it.

Create Firebase User with Email and Password

Using the email address that was generated, and the password supplied via the EditText, the application attempts to create a new user on Firebase. On a successful creation, the user is redirected to the Login screen.

Field validation takes place on all input fields.

On click of either the Continue Button or the Register Button—determined by what step is currently being executed—a circular ProgressBar appears to indicate that there is some processing taking place in the background.

A Snackbar is used to indicate any success or failure messages.

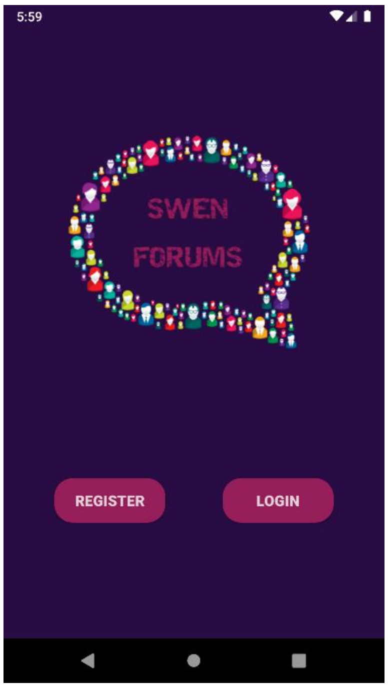


Figure 1
Home Screen

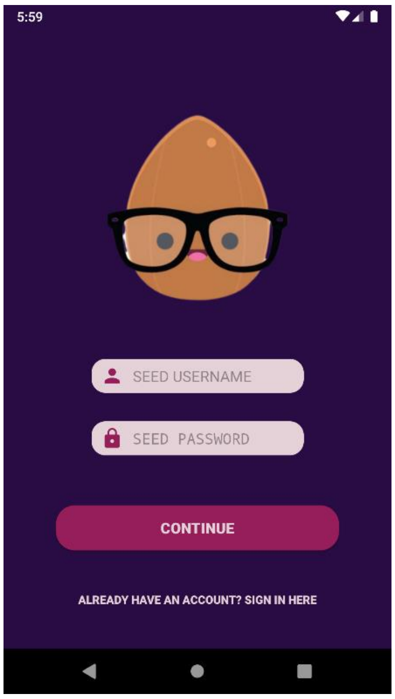


Figure 2
SEED User Validation Screen

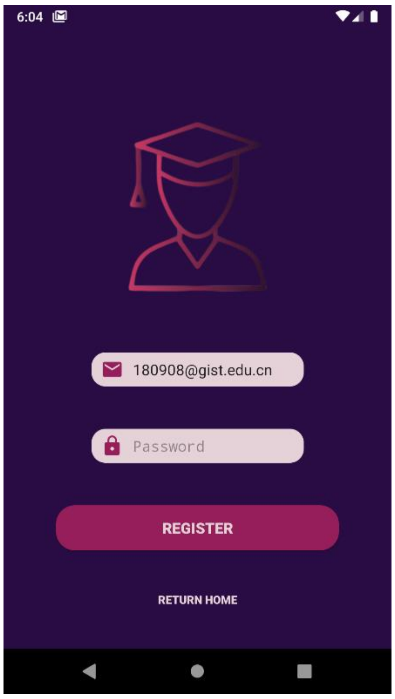


Figure 3
Firebase User Registration Screen

Login

The application takes the values extracted from the EditText widgets and sends them to Firebase for authentication. There is a CheckBox widget on the Login Form of the application that works with the application's SharedPreferences. If the CheckBox widget is checked, then the user's email address is stored in SharedPreferences. This allows for the user needing only to input his/her password the next time the application is launched. If left unchecked, nothing is stored.

Field validation takes place on all input fields.

While the application is signing in, a circular ProgressBar appears to indicate that there is some processing taking place in the background.

A Snackbar is used to indicate any success or failure messages.

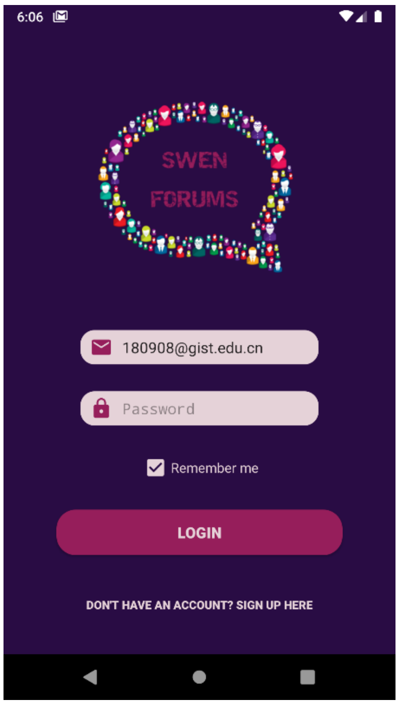


Figure 4
Login Screen

Logout

A user may logout from the application by selecting it from the options menu in the toolbar. An AlertDialog will appear to handle user decisions. If the user does indeed desire to logout, on click of the positive button, he/she will be redirected to the Login screen. On click of the negative button, the AlertDialog will dismiss.

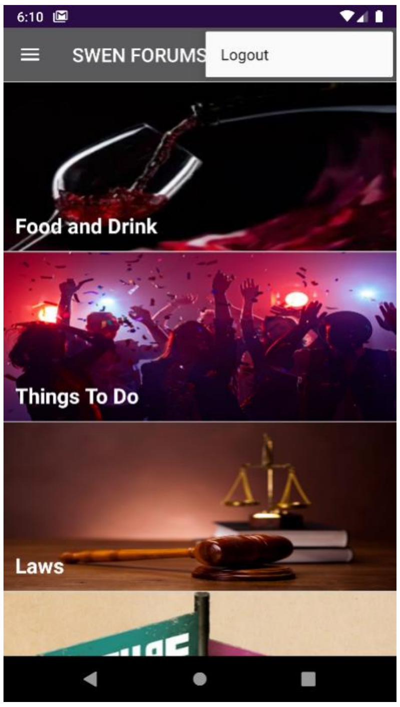


Figure 5
Options Menu

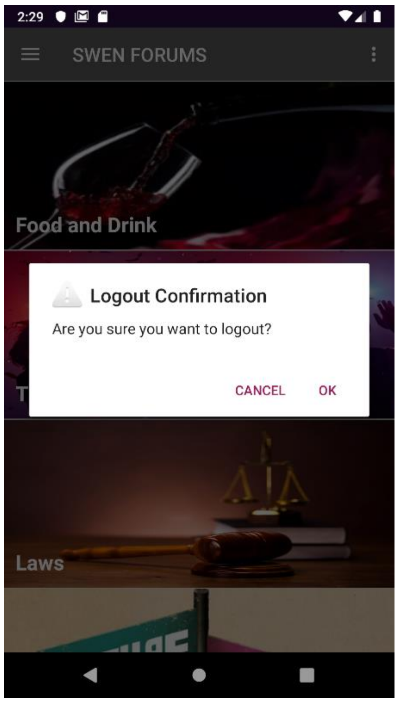


Figure 6
Logout Confirmation Alert Dialog

Category Selection

Once a user has successfully logged in, he/she will be allowed to choose one of six categories from which he or she can view posts or create new posts. A recycler view was used to display the categories. Each viewholder has the name of the category, a related background image, and an expandable sub-view which, by default, is not displayed. On click of either the category title or the background image, the expandable view will appear. From it, the user can view all posts tagged as the category of the expandable sub-view, or create a new post under that tag. Only one sub-view can be expanded at a time. If another is expanded without the user collapsing the previous, the application is designed the collapse the previous.

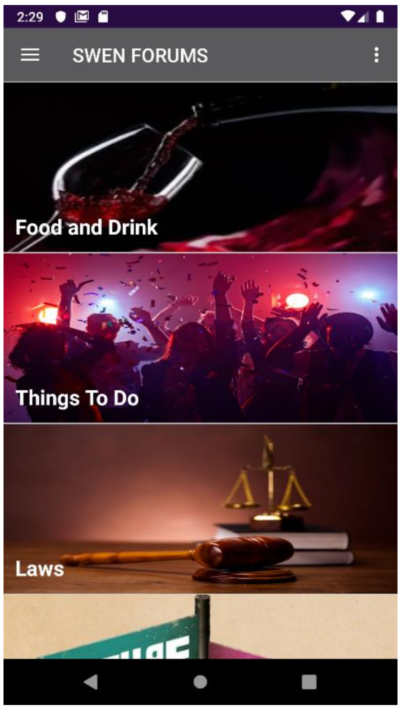


Figure 7
Options Menu

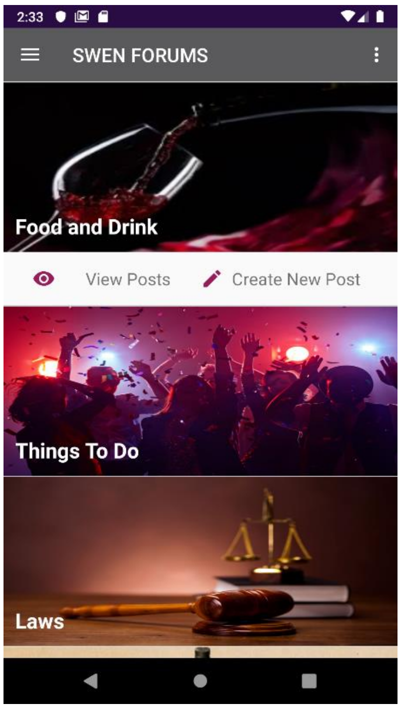


Figure 8
Expandable View

Creating A New Post

Each article in the database will store the following values: an id, the author's name, the post's title, the category the article falls under, the number of views the article has received and the date the post was created.

When a user decides to cancel the creation of a post, the application carries the user back to the Categories Fragment.

When a user presses "create", the addArticle() function is called which deals with adding the article to the database.

In the add article function, the text was retrieved from their respective widgets, the time at which the button was clicked was stored, and the values were assigned to a new article object, which was then added to the database.

After the post is added to the database, an intent is created to facilitate the change from the Create Article Fragment to the View Post Fragment. An intent was then used to allow a change on the application's views from create an article to view the article once it is created in the database.

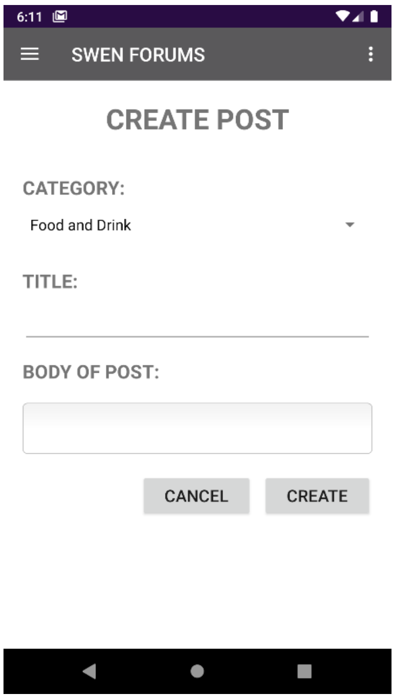


Figure 9
Create Post Screen

Viewing All Posts in a Category

Firstly, the YouTube API was imported into the android project and added to the dependencies of the app gradle.

To facilitate the YouTube player, the activity was set to extend “YoutubeBaseActivity”, which allowed for the implementation of the functions needed to successfully launch the YouTube player in the app.

The firebase database reference was also created at this point.

When the view is created, an initializer listener is used to get the url of the video to be played, assign it to the player, and then initialize the playing of the video.

So as to make it easier to update videos in the future, a separate class was created where the urls specific to the respective categories are assigned.

When an article has been added to the database, we used this value event listener that will automatically update the recycler view with the article added.

A check was implemented to ensure the articles shown in the view are specific to the particular category at hand.

The custom aspect of the adapter was found in the read more button's on-click listener. It was here that an intent had to be used to switch between the button in the card of the recycler view to the view post fragment.

To facilitate changes between fragments and activities that can't be done locally within where they are needed, the main activity was used to facilitate this change. The activity or fragment to be changed will send an intent to main activity where an instance of the view post fragment is created and a fragment manager is used to switch between the fragment and the activity.

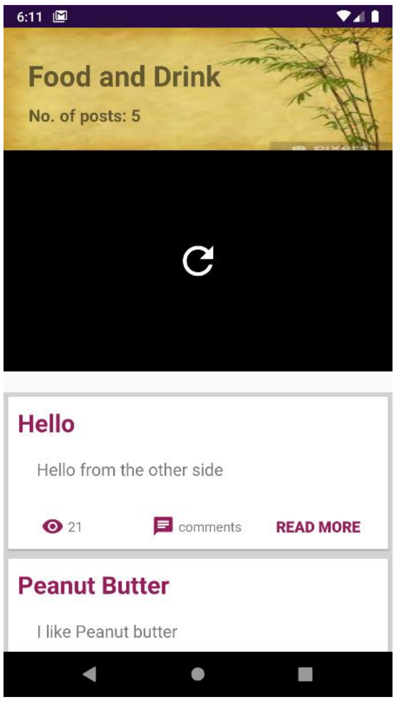


Figure 10
Viewing all posts belonging to the Food and Drink Category

Viewing A Specific Post

On click of the "Read More" in the view holder of the relevant post, the user is redirected to the full post. A post consists of (from top to bottom):

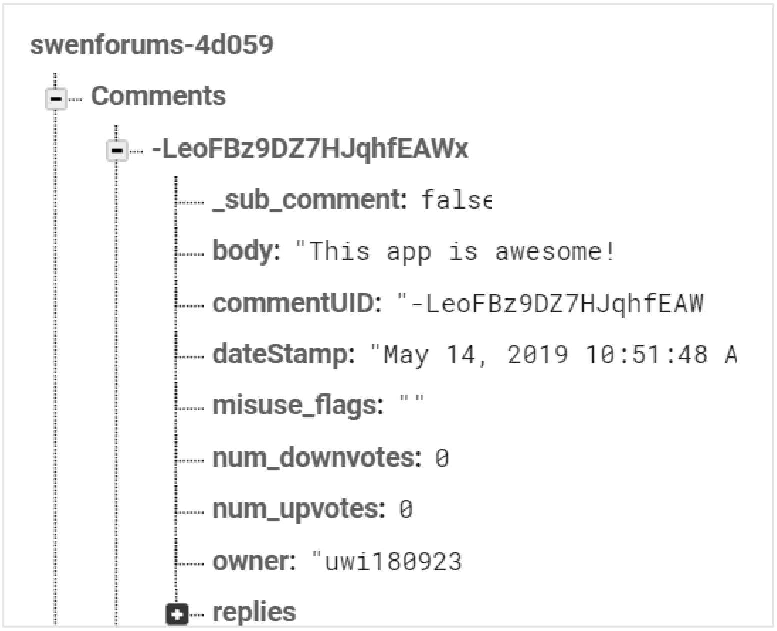
1. A title
2. An author
3. A date
4. A category
5. A body
6. Views
7. Comments
8. Actions

COMMENTS

Comments is a clickable TextView. On click of this view, a bottom sheet is expanded that contains the comment section. A comment is defined by the following attributes:

1. A unique ID
2. An owner
3. A body
4. An upvote count
5. A downvote count
6. A list of replies
7. A flag (optional)
8. A type (sub- comment or comment)

The user can scroll through the list of comments inside the bottomsheet. Each comment is displayed within a custom expandable MaterialCardView. Google's Firebase Realtime Database is used to store information on comments. The display is updated using listeners that trigger when a new comment is added to the database.



```
swenforums-4d059
  Comments
    -LeoFBz9DZ7HJqhfEAWx
      _sub_comment: false
      body: "This app is awesome!
      commentUID: "-LeoFBz9DZ7HJqhfEAW
      dateStamp: "May 14, 2019 10:51:48 A
      misuse_flags: ""
      num_downvotes: 0
      num_upvotes: 0
      owner: "uwi180923
      replies
```

Figure 11
Firebase Database Structure

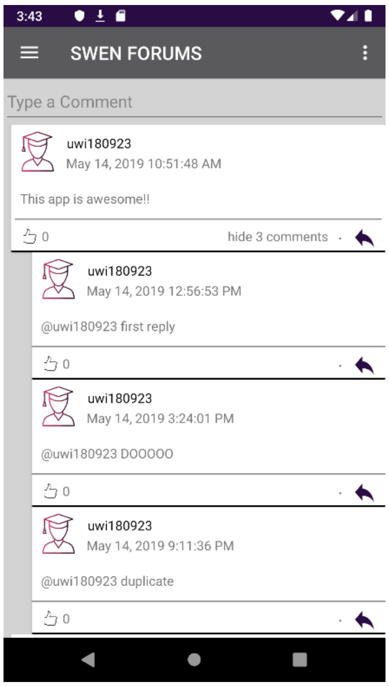
Note: Reply is simply a comment of type 'sub-comment'.

Comment

The user may post a new comment by clicking into the text box at the top of the bottomsheet. Once focus is on the textbox, the cancel and post buttons become visible. If the cancel button is pressed, then focus is taken away and the buttons become invisible. Post will push the relevant post to firebase database.

Sub-Comment

The user can reply to any comment (or sub-comment) by pressing the reply icon on a specific comment card. On click of the reply icon makes a textbox visible, along with the reply and cancel buttons. On click of the cancel button will reverse this. On click of reply will create a comment with type set to 'subcomment' and add that comment to the replies section of the relevant parent comment in the database.



uwi180923
May 14, 2019 10:51:48 AM
This app is awesome!!
0

uwi180923
May 14, 2019 12:56:53 PM
@uwi180923 first reply
0

uwi180923
May 14, 2019 3:24:01 PM
@uwi180923 DOOOOO
0

uwi180923
May 14, 2019 9:11:36 PM
@uwi180923 duplicate
0

Figure 12
Comments Section

ACTIONS

Actions is a clickable TextView. On click of this view, a PopupMenu will appear. Its menu items are determined by whether the current user is the owner of the post.

- Actions exclusive to the owner:
  - Editing,
  - Deleting
- Actions exclusive to non-owners:
  - Reporting
- Non-exclusive actions:
  - Sharing

Editing

This feature allows the current user to make changes to the post. If any changes are made, the database is updated.

Deleting

This feature allows the current user to delete a post. On click of this menu item, an AlertDialog will appear. On click of the positive button, the post is removed from the database. On click of the negative button, the AlertDialog is dismissed.

Sharing

This feature allows the current user to share a post. It is enabled by a ShareActionProvider.

Reporting

This feature allows the user to report a post. On click of this menu item, a BottomSheetDialog will appear. A BottomSheetDialog is a Dialog styled as a BottomSheet. According to Material Design, a BottomSheet is an element displayed only as a result of a user-initiated action, used to reveal more content. The BottomSheetDialog was further supported by a BottomSheetListView i.e. a BottomSheet with a ListView implementation.

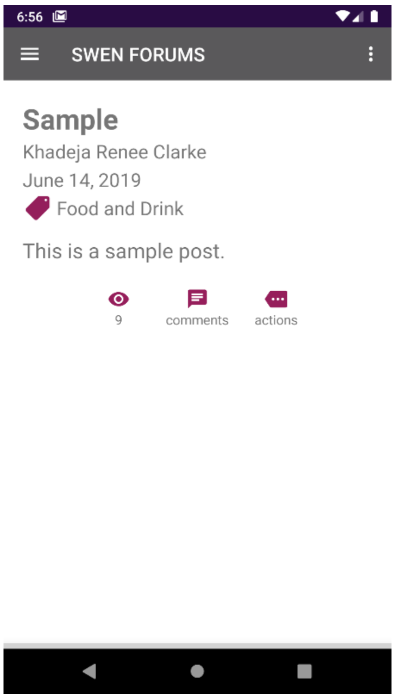


Figure 13
View Post: Owner View

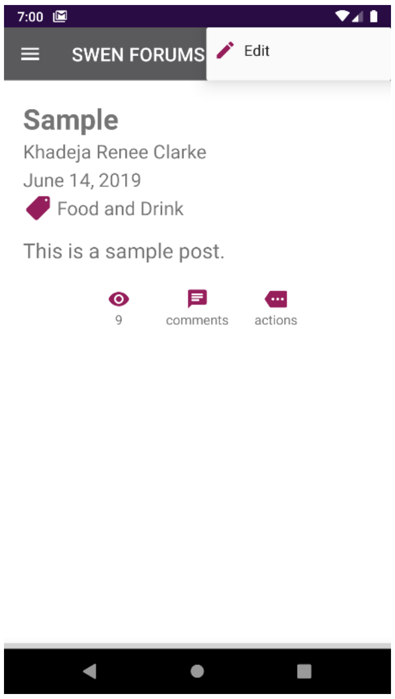
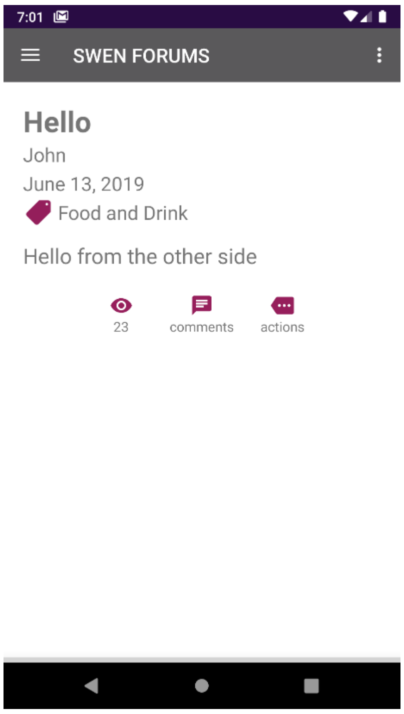


Figure 14
View Post: Owner View Actions


7:01

SWEN FORUMS

Hello

John

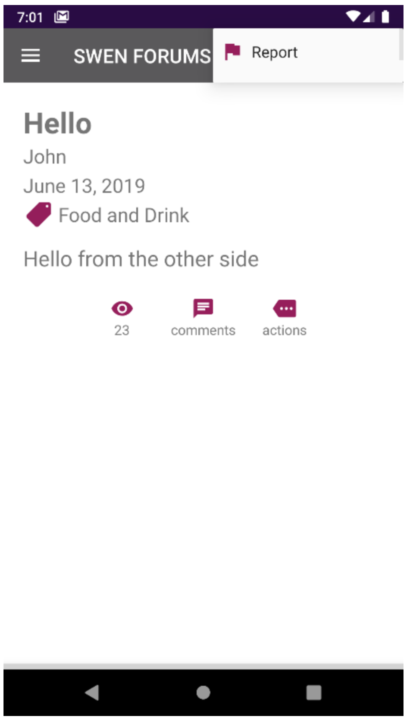
June 13, 2019

Food and Drink

Hello from the other side

23 comments actions

Figure 15
View Post: Non-Owner View


7:01

SWEN FORUMS

Report

Hello

John

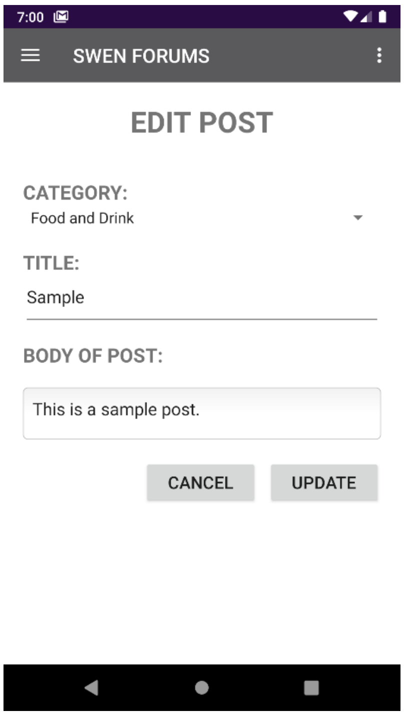
June 13, 2019

Food and Drink

Hello from the other side

23 comments actions

Figure 16
View Post: Non-Owner View Actions


7:00

SWEN FORUMS

EDIT POST

CATEGORY:

Food and Drink

TITLE:

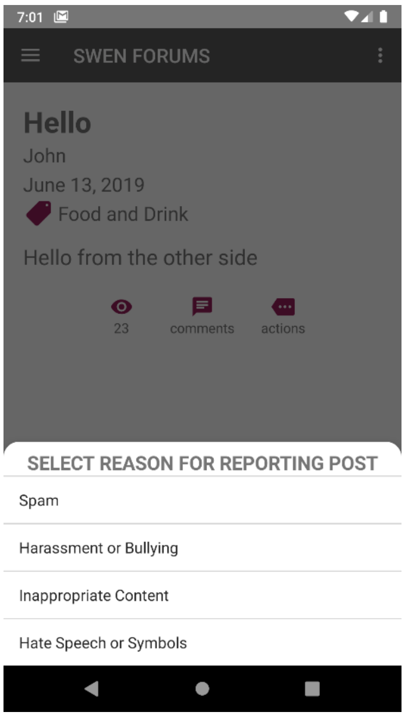
Sample

BODY OF POST:

This is a sample post.

CANCEL UPDATE

Figure 17
Owner Action: Editing


7:01

SWEN FORUMS

Hello

John

June 13, 2019

Food and Drink

Hello from the other side

23 comments actions

SELECT REASON FOR REPORTING POST

Spam

Harassment or Bullying

Inappropriate Content

Hate Speech or Symbols

Figure 18
Non-Owner Action: Reporting

Bugs

1. The RecyclerView for All Posts does not update immediately. Usually on the second or third attempt that the data set is updated.
2. On back press from viewing a post does not update the RecyclerView either, though the database has been updated.
3. Only one menu item is displayed when the actions button in the View Post Fragment is clicked.
4. On back press from the Comments Fragment to the View Post fragment clears the view. Only when the bottom sheet is dragged down that the view remains present.