Android Diary Application

Diaryly: A Good-Bad Diary Book application

PROJECT BY

Krittin	Chatrinan	6088022
Anon	Kangpanich	6088053
Tanawin	Wichit	6088221

A Report Submitted in Partial Fulfillment

of the Requirements for

ITCS424 - Wireless and Mobile Computing

Faculty of Information and Communication Technology

Mahidol University

Semester 2/2019

Source Code Link

Project_-_Dailyly.zip (99.51 MB)

[Putting the PDF report in the ZIP will make it exceed the 100 MB limit.]

https://drive.google.com/file/d/10_LkgTtMukZxZR8DqGgazVz9kNeTSxhK/view?usp=sharing

Move to the next page to continue to the report content.

Contents

Application Overview

Nowadays, smartphones or other mobile devices have tremendously gained popularity in every country. They have become part of our daily life. Some people focus their minds on external distractions such as social media or entertainment. Our team believes that to help people improve their life, creating an application, that help people keep track of their life in terms of the righteousness of their actions throughout a day, will certainly help remind people of their actions and project how good or bad they live their life.

Project Objectives

- To help users to keep track of their actions in everyday using a text editing tool.
- To help users remind and keep track of their daily good-bad goals.
- To remind users of their actions in the past.
- To create an application that is ready for real-world uses.
- Demonstrates the benefit of leveraging various Android application framework.
- Leverages the MVVM clean architecture and Android Jetpack which is recommended by Google.
- Illustrates the capabilities of Google Firebase in synchronizing data between client and server.

What does the app supposed to do?

With our app, users can register and login into the app to create new diary entries. In each diary entries, users can enter texts, location, and images as well as the type of their action (Good or Bad). Moreover, users can also search their diary and create a goal of doing good and get reminded depends on their preferences.

Planned Feature Check List

This section shows the list of planned features. Please note that during the development process, there were changes of feature and implementation plan made along the way. Each green-colored row indicate that a feature that is done.

Name	Description
Adding a diary entry	A user can add diary entries for each day.
Adding Images to each diary	A user can add new images to a diary entry.
Rich Text editor for a diary entry	In each diary entry, a user can input and stylize text based on their liking.
Adding Tagging system	A user can give each dairy entry a tag which help classifying each diary entry.
Adding Reminders	A user can add reminders which will be notified by the app based on specified time.
Favorite and Custom Lists	A user can add diary entries into the favorite list or custom lists.
Geotagging entries	A user can also specify locations and embed into a diary entry.
Search and Filtering	A user can specify search or filter diary entries based on a given criteria within the app.
Good-Bad Tracking	For each diary entry, users can specify the type of the action that they did to quantify the amount that can be used to determine a process to achieve user-customized goals.

UI Customization	Due to the fact that this app is supposed to be used daily, providing a customization capability will certainly prevent users from getting bored. UI will be customizable in terms of accent colors, light/dark theme, header images, notification sound, typefaces and some UI layout in the diary entry list or the dashboard.
Localization Thai-Eng.	To extend the coverage of user demographics, Thai localization is required.
Registration and User Authentication	To secure sensitive information of a user, user registration and authentication is required.
~~Room database~~ Replaced with Firebase	To demonstrate the capability of the room database, we planned to use RxJava and room database as well as LiveData to enable an MVVM clean architecture within the app with reactive programming.
Multi-local Accounts support	On top of registration and authentication, supporting multi-local account will be useful for people who share their devices with others. Accounts can be swiftly switched.
Good and Bad Goal	To help users improve the way they live, we also planned to introduce a goal which can help motivate users doing good more.
Good-bad statistics	On top of the goal, statistics such as the total or average of good-bad can prove useful for some users.

Overall Application Architecture

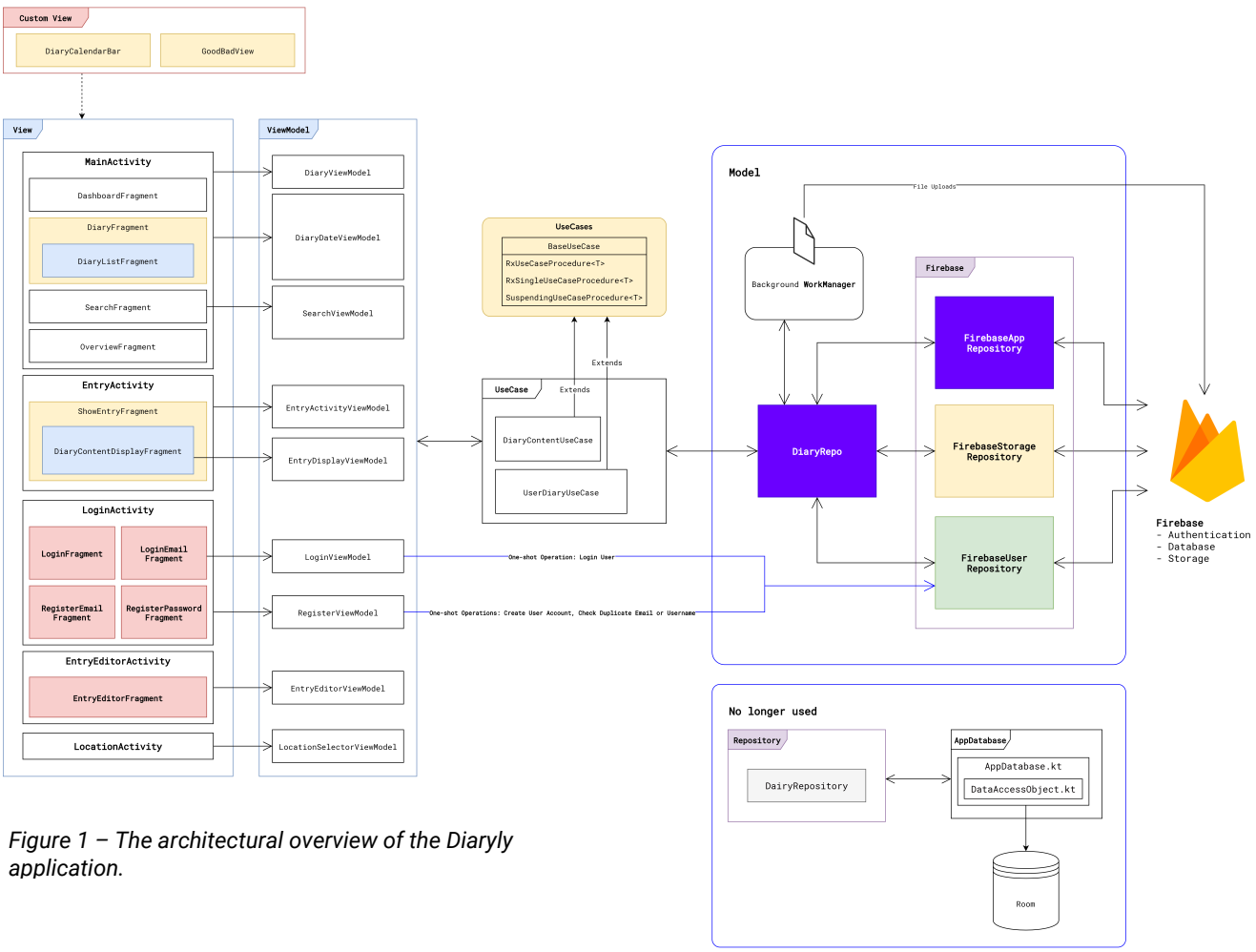


Figure 1 – The architectural overview of the Diaryly application.

1. **`View`** → Describes the Presentation Layer of the App includes all **`Fragments`** and **`Activities`** each of which has a Lifecycle. Please note that each fragment can be nested.
2. **`ViewModel`** → Describes a data holder for each view. **`ViewModel`**s can persist data that is usually garbage collected by the garbage collector when a view changes to a certain state. For example, if there is a text field filled

with texts. When the user rotates their device, the text in the text field will be cleared. Moreover, **`ViewModel`** can be used to communicate or share data between fragments; therefore, **`ViewModel`** is one of the main tools we used in this project.

3. **`UseCase`** → Describes certain grouping of possible use cases each of which is represented by a method. Each method wraps a function call that enables robust error handling and concurrency powered by RxJava and Koltin's coroutine.
4. **`Model`** → It acts as a single source of truth (data) for the entire app. Describes the layer that integrates multiple data sources such as room database, Google Firebase. This layer usually labelled as **`Repo`** or **`Repository`**. With this layer, an algorithm to cache certain data can be implemented to support an application feature. This layer can be broken down into components as follow:
   a. **Room database** → A app-private device-local database. We have implemented most of our room database, but later throw it away since there are complication, limitations and requirements such as the need to authenticate users or the need to synchronize the data.
   b. **Firebase** → We have migrated the app from Local Room database into Firebase. Firebase Authentication, Realtime Database and Storage are used to enable robust authentication, database and file storage.

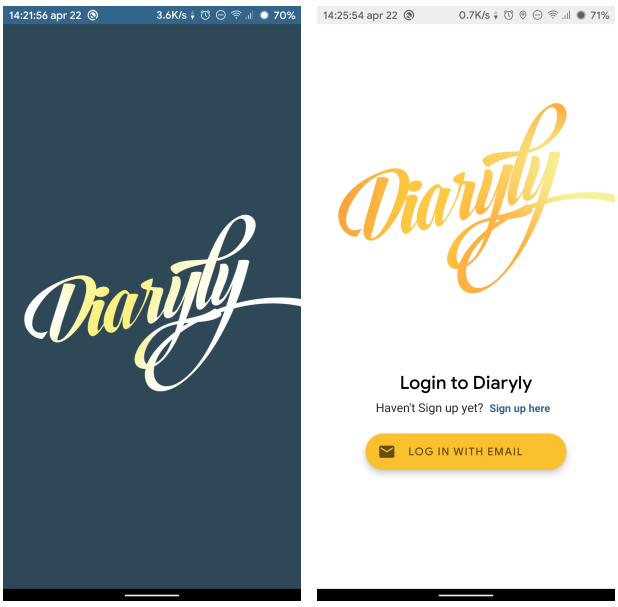
Overview

Our application can be segmented into 3 parts as follows:

1. **Out-of-box experience (OOBE)** part includes all sign in and registration flow of the application.
2. **Diary Book** (Main) part includes navigation and the process of browsing diary entries.
3. **Diary Editor** part includes the process of creating or editing a diary entry.

Out-of-box experience (OOBE)

In this section, the activity and fragments that related to user login or registration will be explained in details…



Overview

When a user launches the app for the first time, they will be greeted by a splash screen and the main landing page as illustrated on the **`LoginActivity.`** The User can select to either login with an existing user account or register a new account.

Figure 2 – The splash screen and the landing page of the Diaryly application.

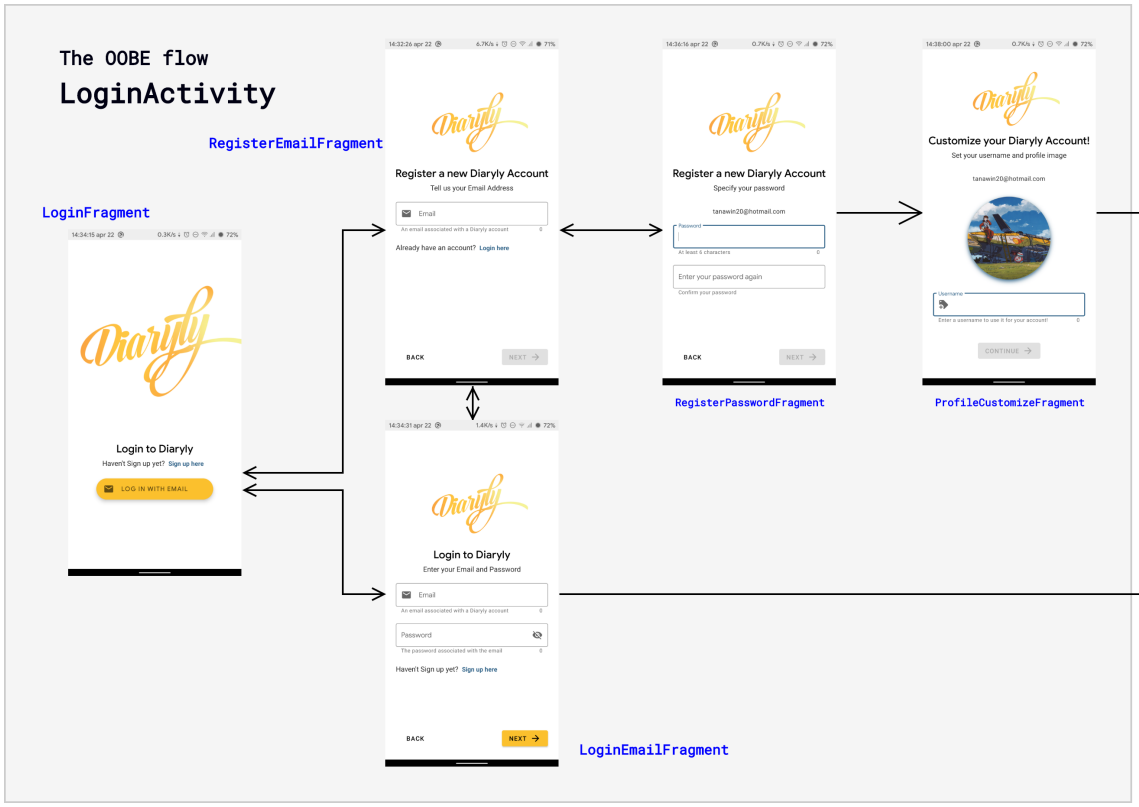


Figure 3 – The flow of the fragments with the `LoginActivity`.

From figure 3, it can be observed that the `LoginActivity` has 4 fragments that responsible for each step of logging in or registration. This activity actually has only `FragmentContainerView` that can only host a fragment at a time. We decided to use it to create a seamless experience in login and register. To use the `FragmentContainerView`, we must specify a navigation XML as in the figure 3.

When a user launches the app for the first time

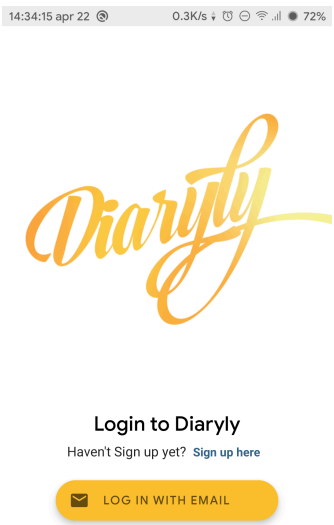


Figure 4 - The landing page of the app

Figure 3 shows how the organization and navigation flow of fragments within the `LoginActivity`. When a user enters the app for the first time, the user will be greeted by the `LoginFragment`, as illustrated in figure 4. In the `LoginFragment`, there are 2 choices for the user to make: either choose to **create a new account** or **log in to an existing account**. If the user chooses to register, the user has to enter the first register page (figure 5) which contains a form that will ask the user to enter their email. When an email is entered, the app will automatically verify the validity of the entered email address with the server. If the email is valid, the Next button will be enabled, allowing the user to continue (figure 11). Otherwise, if the email is not valid or already exists in the system, the app will not allow the user to continue and will notify the user (figure 10). When continues to the password page (figure 9), the user will be asked to enter the password associated with the account 2 times to confirm. The password entered must have a length greater than 6 characters. If the user entered both passwords correctly, they will be able to press the next button. Otherwise, they will be notified to change passwords (figure 14). After the user entered valid passwords and pressed the next button, they will be redirected to the customization page (figure 12) that allows the user to select an image for their profile and to specify a unique username for their account.

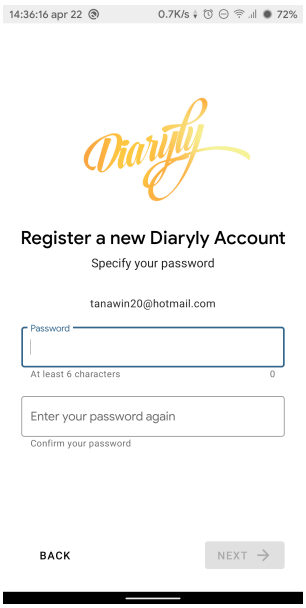


Figure 5 - Password Registration

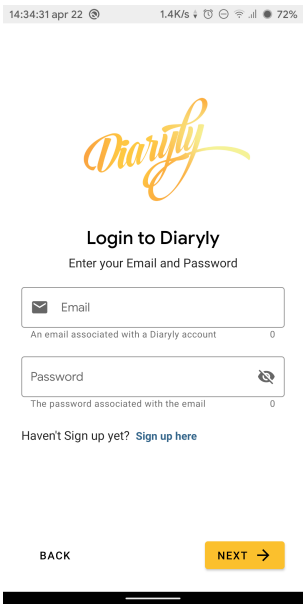


Figure 6 - Login Page

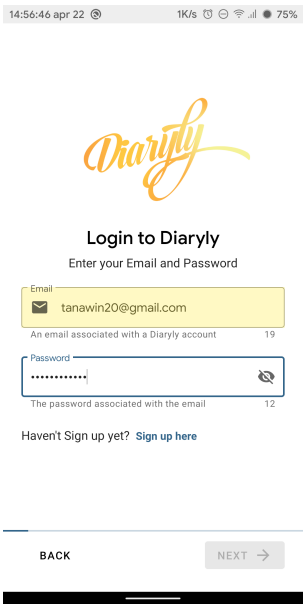


Figure 7 - Progress indicator bar on the bottom

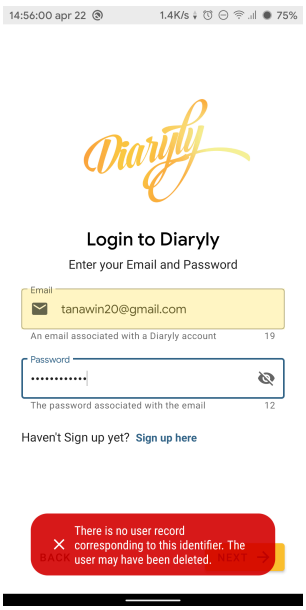


Figure 8 - Error Handling

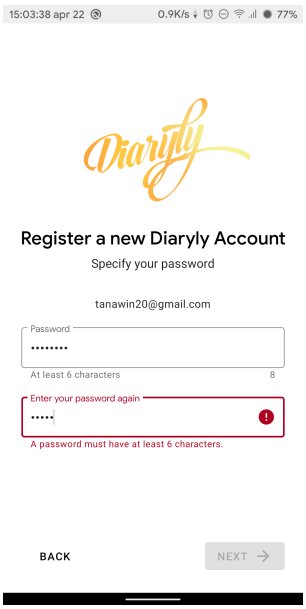


Figure 9 – A Password Error

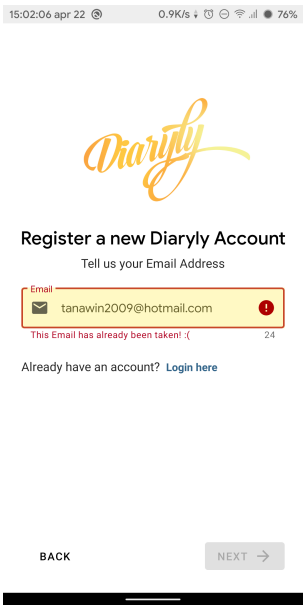


Figure 10 - Invalid Email

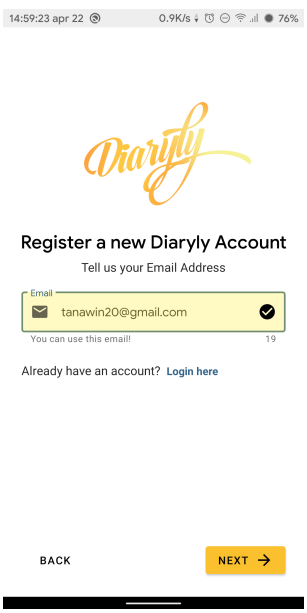


Figure 11 - Valid Email entered

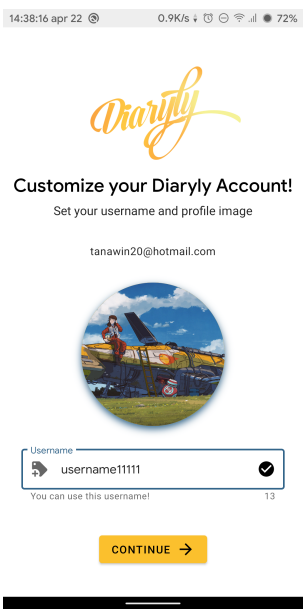


Figure 12 – Profile customization page

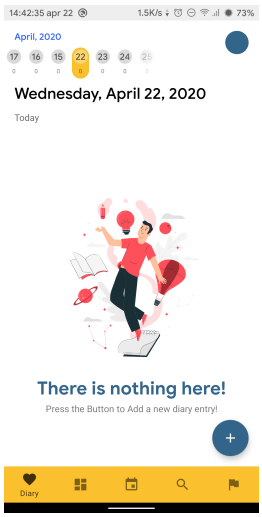


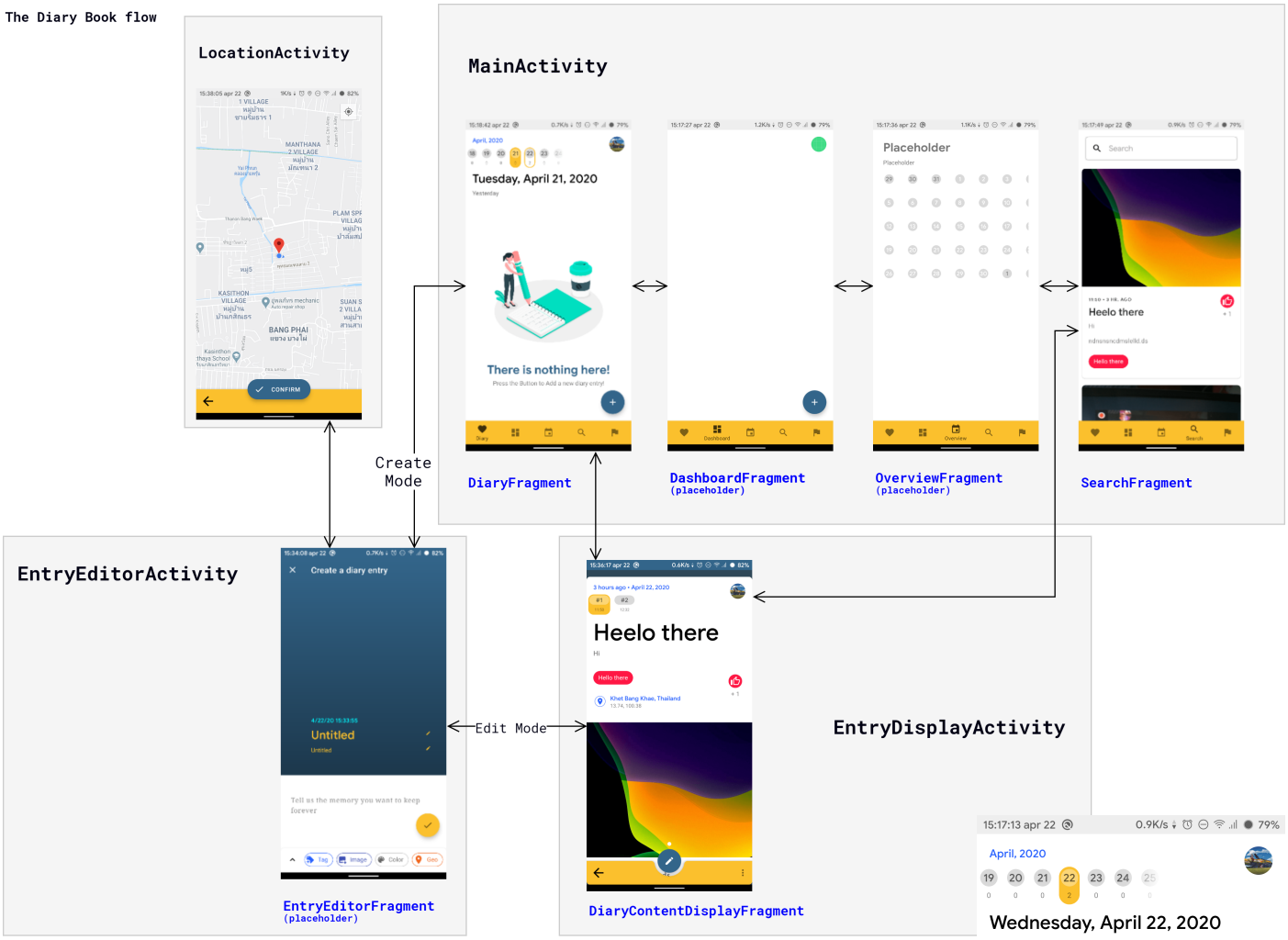
Figure 13 – Main Page

On the other hand, if the user already has an account and wants to log in, they can do so by directly entering the login page (figure 6). Similar to both register pages, the login page also has a progress indication (figure 7) and a rich error handling (Figure 8). After the user entered a valid email address and a valid password and pressed the next button, they will be redirected to the main page of the app (figure 13).

Diary Book

Overview

Diary Book is the main section of the application because it contains various elements that can be used to navigate to other parts of the application. The main container of this section is **`MainActivity`** which contains various **`Fragments`** inside. We achieve so by using the new navigation library from AndroidX.



After logging in or reopen the app

The user will be directed into the `DiaryFragment` inside the `MainActivity` as illustrated in figure 15. In the `MainActivity`, there are 5 navigation buttons at the bottom of the screen most of which can be pressed to navigate. There are 5 buttons from left to right as follows: Diary, Dashboard, Overview, Search and Goal. Please note that some of the features are not implemented.

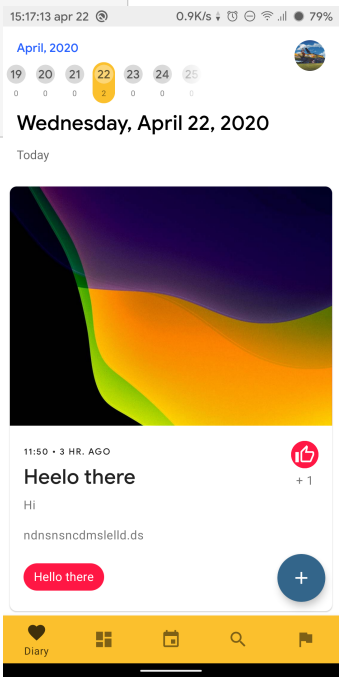


Figure 15 – DiaryFragment with items

DiaryFragment

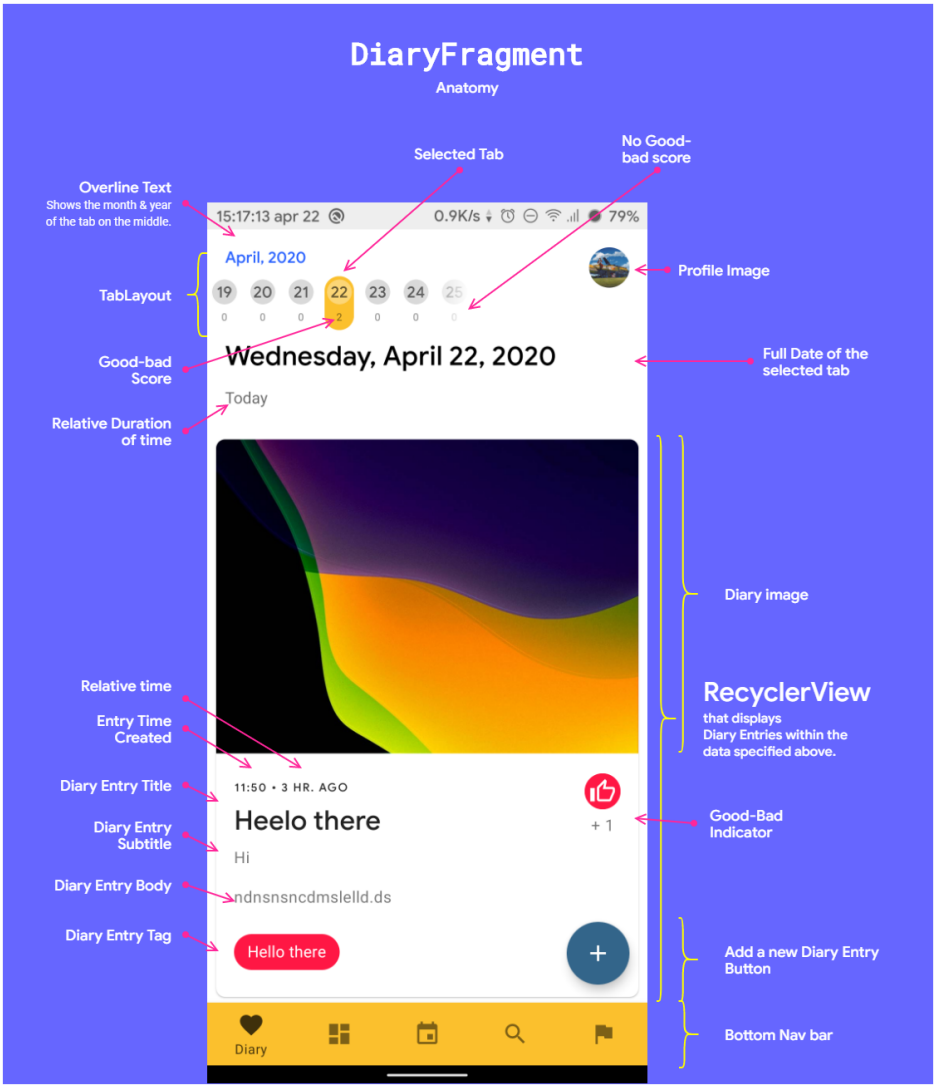


Figure 16 – Anatomy of the DiaryFragment

We considered `DiaryFragment` to be the main screen because it contains the main interaction part of the app. As displayed in figure 16, the screen allows user to create a new diary entry by pressing the button on the bottom right corner. The user is also allowed to navigate to each date by clicking at the numbered tab on the top. Moreover, use can also scroll the tab or click at the overline text to select a time window to display as in the figure 17.

Powered by `ViewPager2`, users can also swipe to navigate between tab as in figure 18.

In this page, we also have a state indicator for users. So far there are indicator for loading and empty states as illustrated in figure 19 and 20.

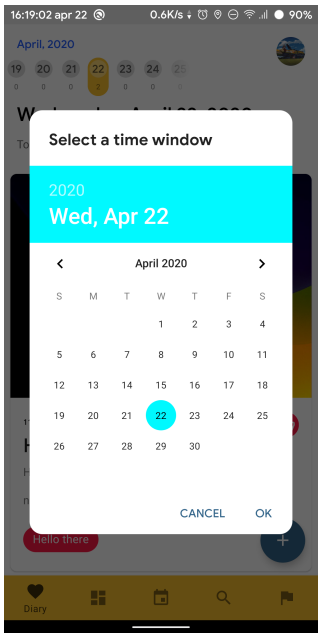


Figure 17 – Dialog to select Time window for tabs

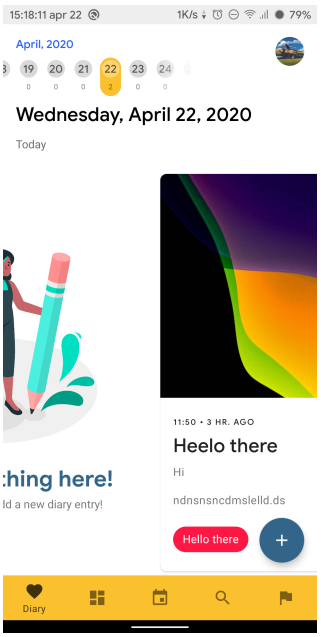


Figure 18 – Swiping between tabs

Figure 19 – Loading state of the `RecyclerView`

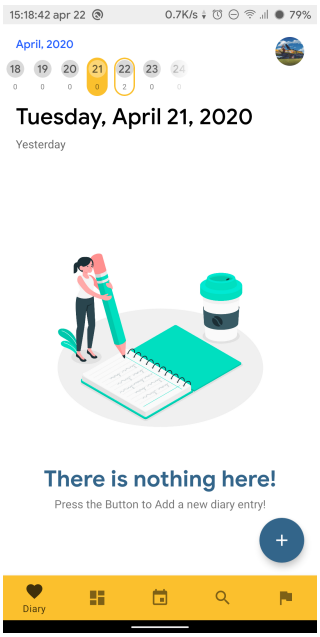
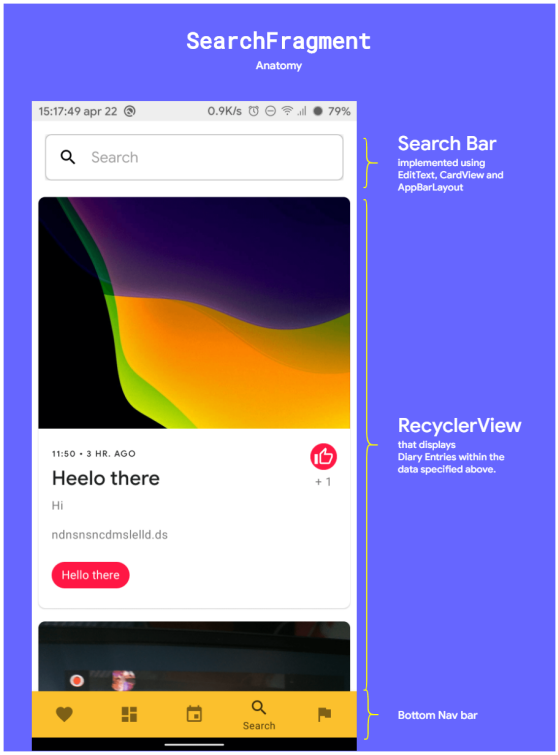


Figure 20 – Empty state of the `RecyclerView`

SearchFragment



After the user clicked on the search icon on the bottom navigation bar, the search screen will be displayed. With it the user is allowed to search their diary entry by the title name. This feature is using basic Firebase database commands as displayed in figure 21.

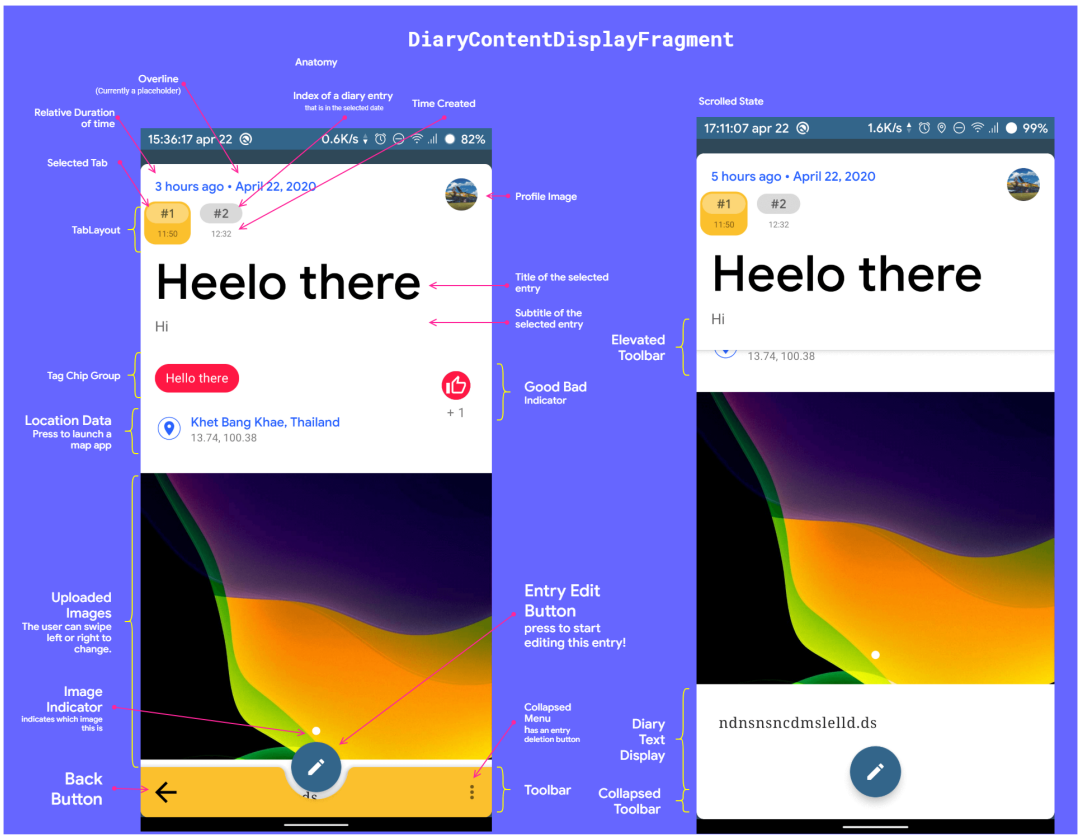
```
databaseRef
    .orderByChild("title")
    .startAt(title)
    .endAt(title + "\uf8ff")
```

Figure 21 – Snippet of Code that allows partial search for the entry title.

Figure 22 – Anatomy of the `SearchFragment`

DiaryContentDisplayFragment (located inside `EntryDisplayActivity`)

After the user select a `RecyclerView` Entry in either `SearchFragment` or `DiaryFragment`, the user will be redirected to the `EntryDisplayActivity` that contains `DiaryContentDisplayFragment`.



This page allows users to view and interact with their diary entry. The user can also tap on tabs above to navigate between entries on the same date. Please note that the user can also scroll down to see all of the content.

Figure 23 – Anatomy of the `DiaryContentDisplayFragment`

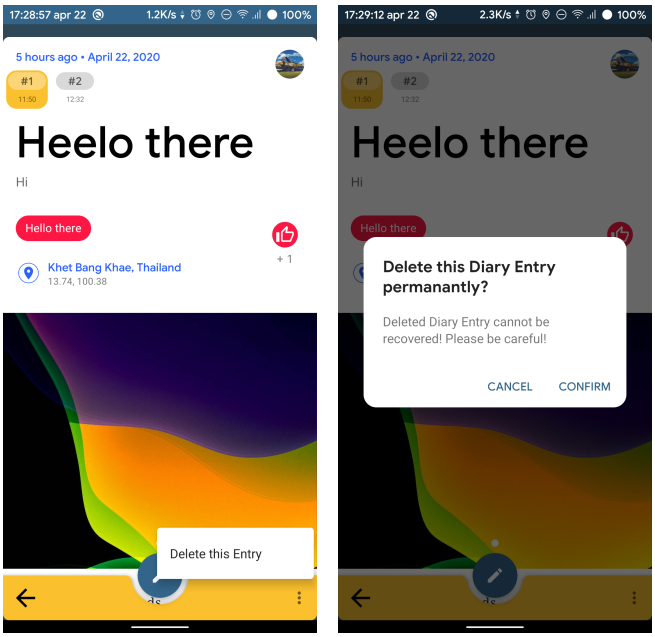


Figure 24 – Users can delete their diary entry from this screen.

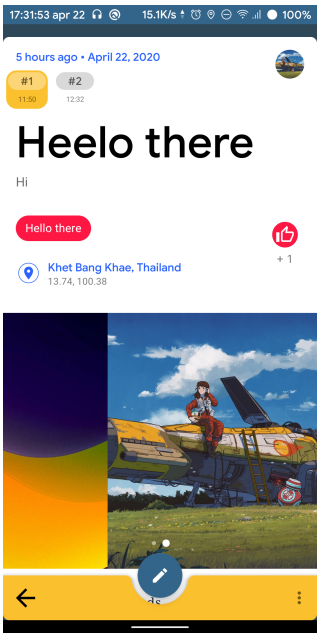


Figure 25 – Sometimes users added multiple images. To support that use case, we allow them to swipe between images.

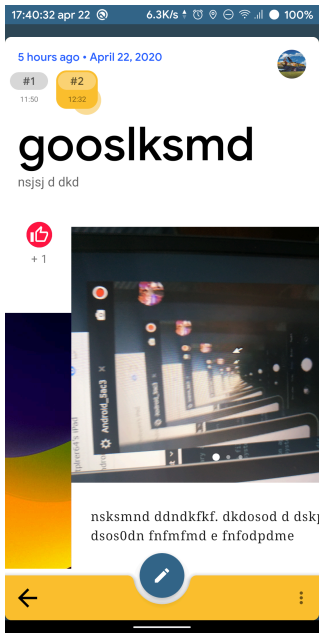


Figure 26 – Unlike the main page where the user can swipe to change pages, in here, users are allowed to change pages by tapping the tab only.

Miscellaneous

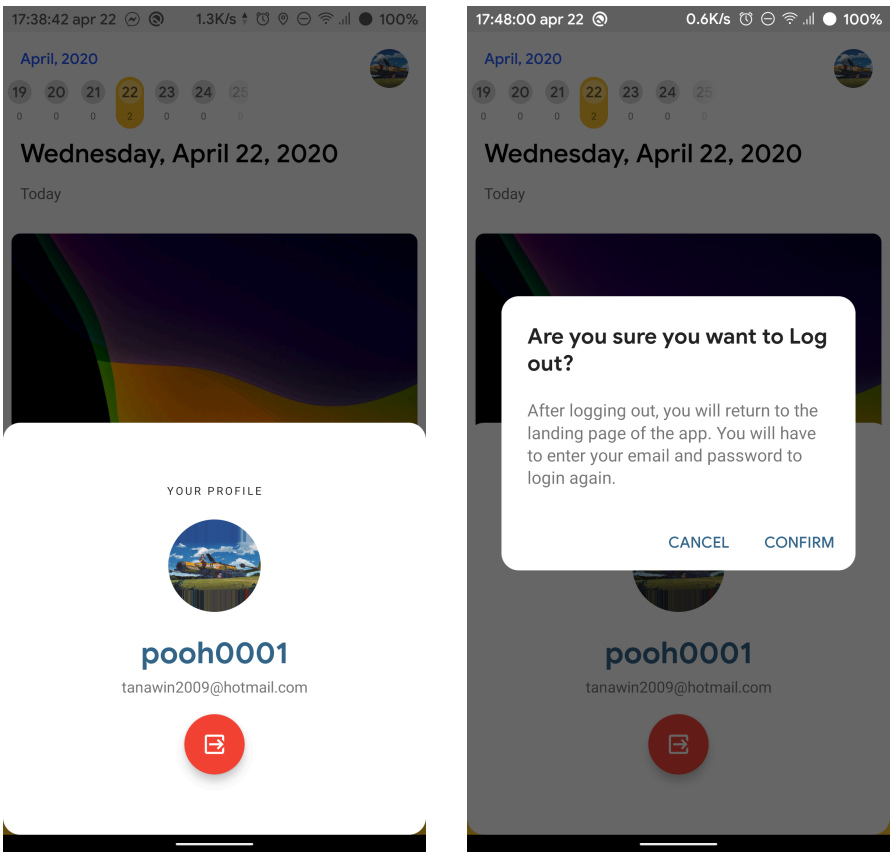


Figure 26 –Users is allowed to log out of the application by tapping at the profile image in the main screen. And tap the red button to log out.

Diary Editor

Overview

Diary Editor allows users to create or edit a diary entry. The main container of this section is **`EntryEditorActivity`** which contains **`EnttryEditorFragment`** that facilitate the logic and UI for the user to edit or create an entry. To enable reactive UI, similarly to other parts of this app, we heavily utilized **`LiveData`** that allows us to observe its values and reactively change UI upon any changes.

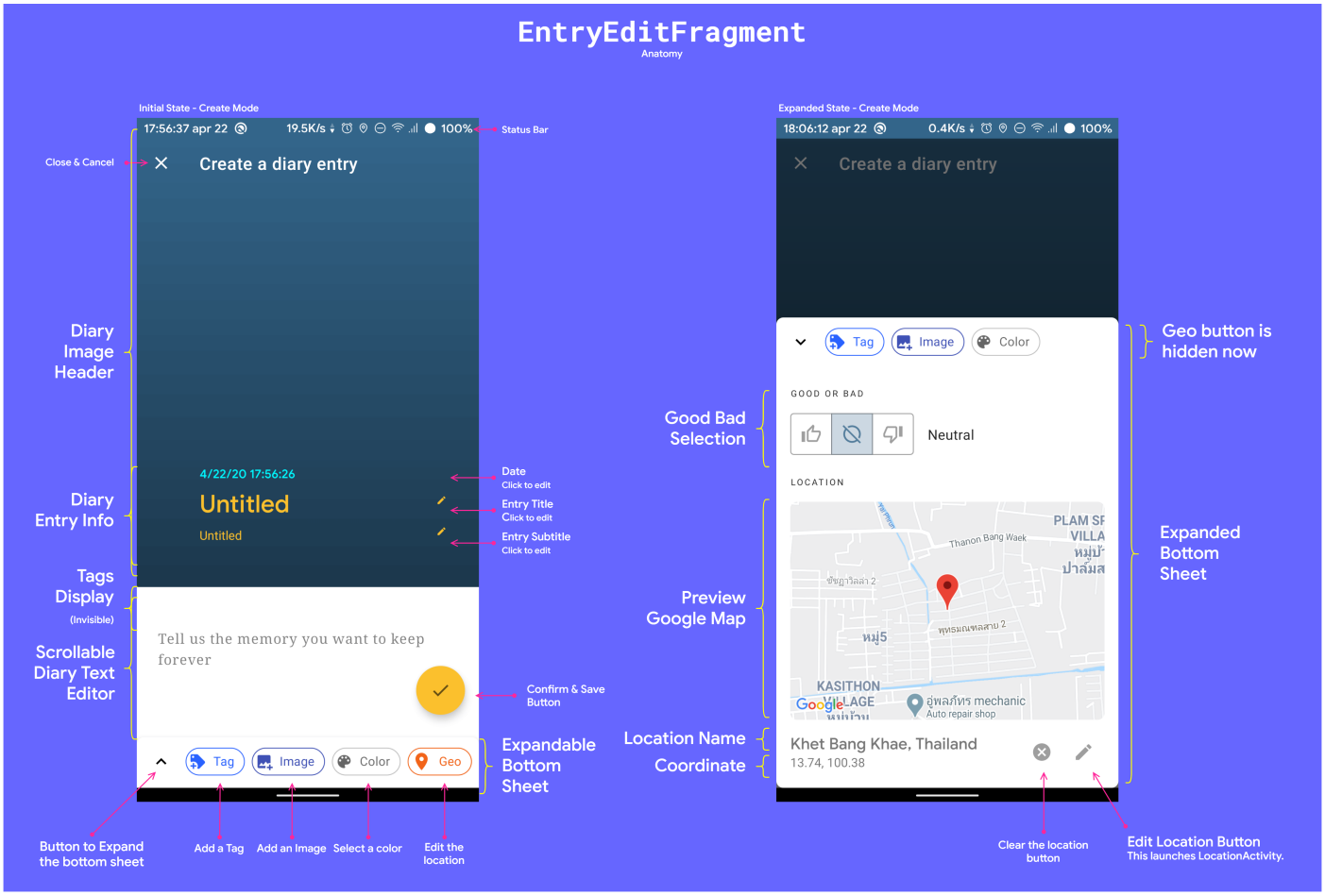


Figure 27 – Anatomy of `EntryEditorFragment`

`EntryEditorActivity` is an activity that hosts a fragment named `EntryEditorFragment`. `EntryEditorFragment` allows users to add a new diary entry that can consist of an image, text, geolocation coordinate and tags. `EntryEditActivity` can be accessed from the main page by using the add floating action button on the bottom right corner (figure 16). This screen also contains a bottom collapsible sheet that will expand when a user press the expand button.

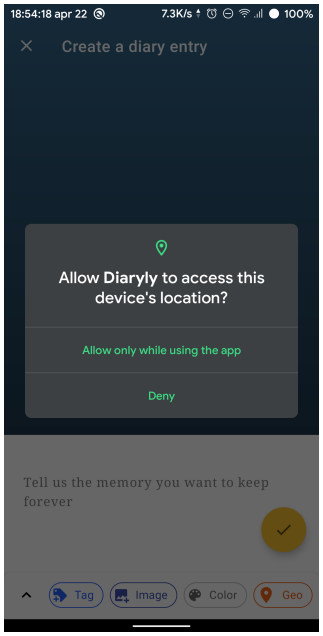


Figure 28 - When a user enters this screen and click on the Geo button. The location permission request dialog will be displayed.

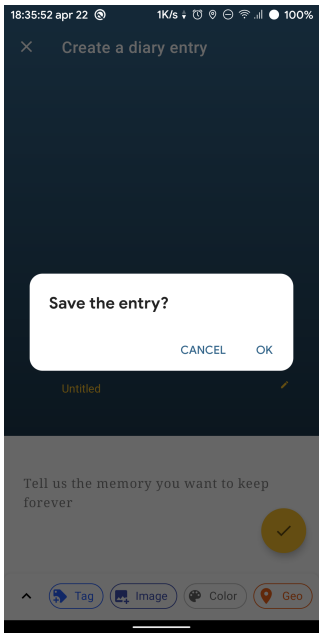


Figure 29 – When the user wants to confirm and save the diary entry, a dialog will be displayed to ask the user for confirmation.

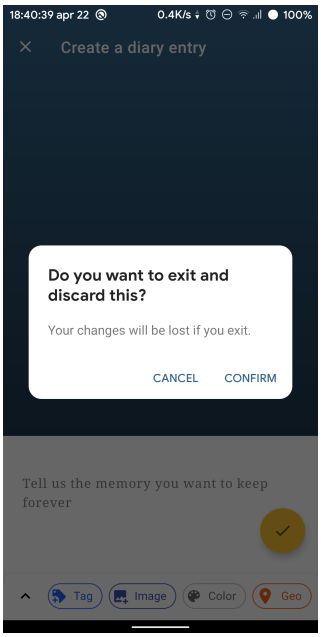
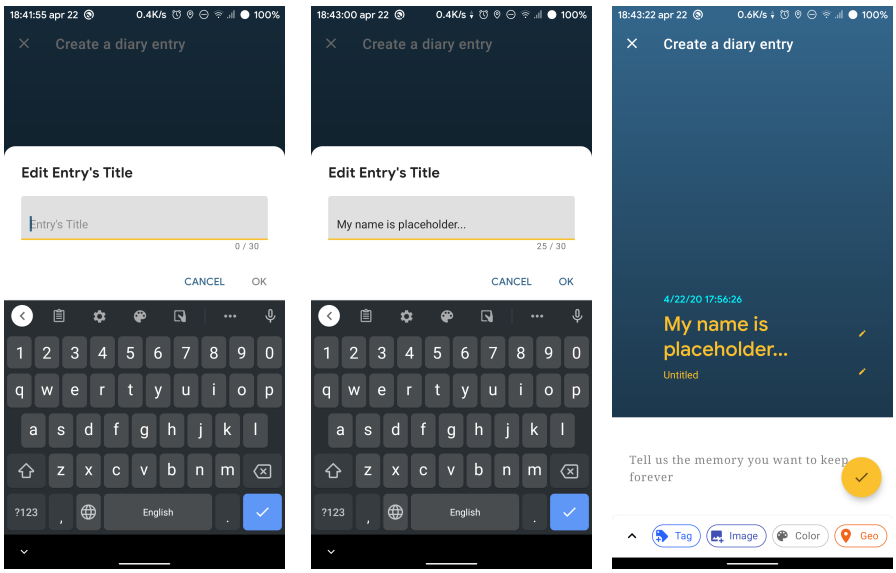
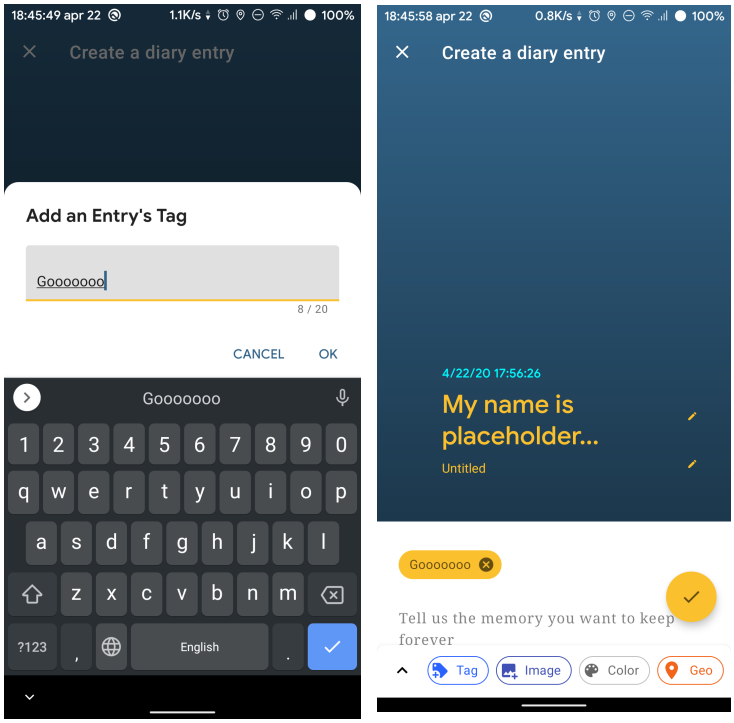


Figure 30 – When the user wants to cancel and discard, a dialog will be displayed to ask the user for confirmation.



Editing Title or Subtitle

Tap on the “untitled” text or the pencil icon of either title or subtitle. A text dialog will be displayed.

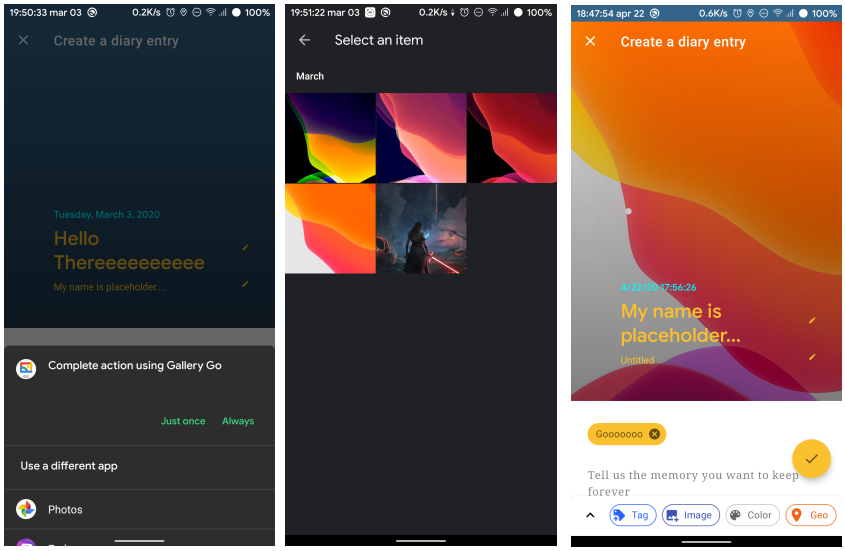


Adding a Tag

Tap on the “Add a new tag” icon. A dialog will ask a user to enter a tag name. If the user confirms, a new section called “Entry Tag” and a new Tag Chip will be visible on the screen.

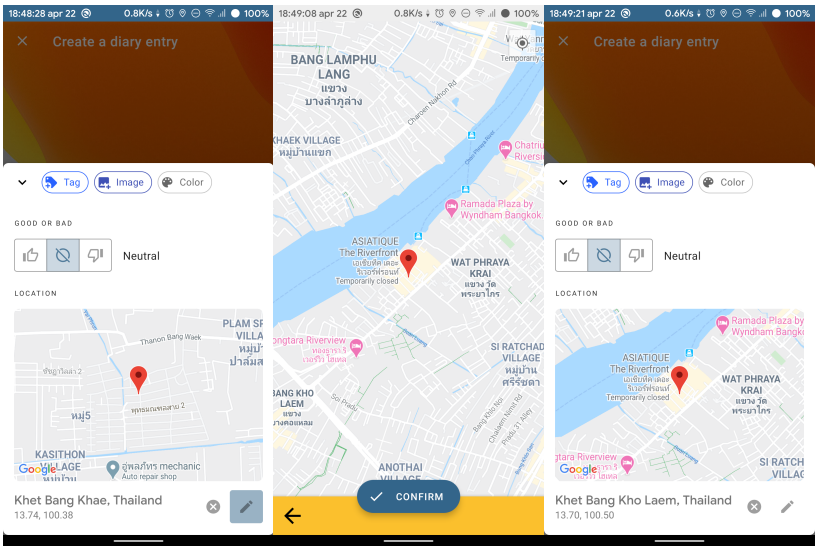
Removing a Tag

To remove a tag, a user can tap on the cross icon on each Tag Chip.



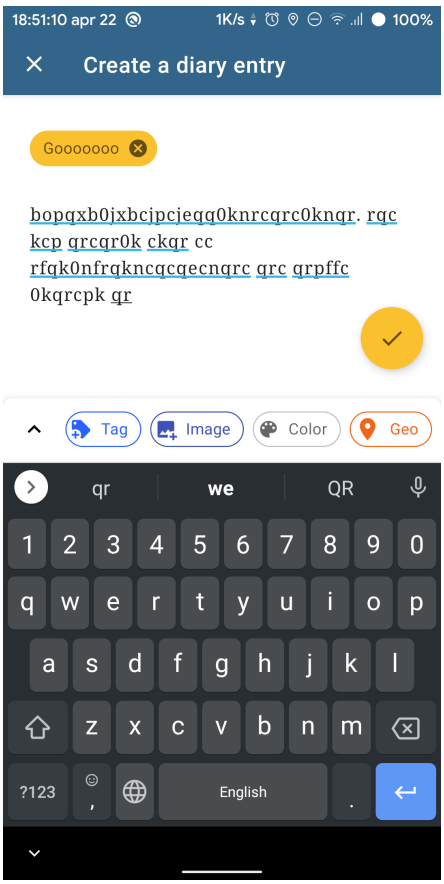
Adding an Image

To add a new image, a user can tap on the colored space beneath the title or the “Add an image” button. The app will allow the user to select an image or photo from other gallery apps. When an image or photo is selected, the image will be displayed.



Adding a location

Tap on the Location Preview will redirect user to the **`LocationActivity`** which contains a map that allow user to tap on to select a location. When confirmed, the location will be return back to **`EntryEditorActivity`**.



Edit an entry text content

A user can freely edit their text using the **`TextEdit`** we provided below the tag view.

Project Requirement Checklist

Feature	Status
Support both Thai and English menu	Yes 
Provide registration and user authentication	Yes
The app must store data in a local database using Room or Firebase.	In this project we initially used Room, but later changed to Firebase. We used Firebase Authentication, Realtime Database and storage.
Use at least three of the following features:	
• Tabs	Yes
• Linking to camera	
• Linking to network	Yes
• Animated images	
• Location services and maps	Yes, inside the `EntryEditorFragment`
• Preferences	
• Dialogs	Yes
• Google Map	Yes, we use both **maps embedding** and **intent to the Google Map app.**
• `RecyclerView` or `CardView`	Yes, for the list of entries. We used `ListAdapter` which is a more sophisticated version of typical `RecyclerView` Adapter
• Other advanced Android features	Yes 1. Background Image Upload – `WorkManager` 2. Reactive View + Lifecycle – `LiveData` & `ViewModel` 3. Advanced Reactive data stream – `RxJava` 4. Custom View – Overriding API-provided view classes. 5. Kotlin Coroutine – We use it to grab `Bitmap` from URI.