Mobile Apps 2

Assignment Two

Submission Date 18/10/2024

Percent 10%

Include all workings and references

Section A.

“The passing of data from screen to screen can be difficult, especially if the data is complex”

1. Discuss, with 2-3 proposed means of achieving such a data transfer. (*3 marks*)

Proposed Means of Achieving Data Transfer

- Using Intent Extras (Android)

In Android apps, one common way to share information between screens is through Intents. You can think of an Intent to send a package from one screen to another. Inside this package, you can include extras, which are key-value pairs. For example, if you want to pass a user's name from a login screen to a welcome screen, you can pack it as an extra. It’s straightforward for simple data transfers!

- ViewModels and LiveData

Another method is to use ViewModels and LiveData. A ViewModel acts like a helper that keeps your data safe, even if the screen changes, such as when you rotate your device. LiveData notifies your UI components whenever the data changes, much like a group chat where everyone gets updates. This is especially useful for more complex data that needs to be shared across different parts of your app, like user profiles or lists.

- Lambda Functions for Data Transfer

I used lambda functions to pass data between screens. For example, in the Screen1 composable, I defined a lambda parameter onWordEntered: (String) -> Unit. This lets Screen1 send the entered word back to Screen2 when the user clicks the button. It’s a clean and efficient way to transfer data in modern Jetpack Compose apps.

2. Give the advantages and disadvantages of two of the above. (*2 marks*)

Using Intent Extras

Advantages:

- Easy to Use: Intent Extras are super simple. You can quickly send small pieces of data (like a user's name) from one screen to another. Just pack it in an Intent, and you're good to go!
- Lightweight: This method doesn't add much extra weight to your app. It's perfect for quick and simple data transfers without any fuss.

Disadvantages:

- Size Limits: Intent Extras can't hold too much data. If you need to pass large items or complicated information, this won't work very well.
- Scattered Data: Since each activity has its own Intent, it can get messy keeping track of everything as your app grows. It can feel a bit like having a bunch of scattered notes instead of one organized file.

Lambda Functions for Data Transfer

Advantages:

- Flexible: Lambda functions let you pass data between screens in a very adaptable way. You can customize how the data is handled, making it fit your needs perfectly.
- Clean Code: Using lambda functions helps keep your code neat and tidy. They fit nicely into your composables, which helps you manage everything in one place.

Disadvantages:

- Limited Use: Lambda functions work best within a single composable. If you have multiple screens, you might end up passing a lot of lambdas around, which can get tricky.
- Can Get Complicated: As your app grows, keeping track of multiple lambda functions can be hard. If you're not careful, your code can become tangled and hard to follow.

Section B.

Build a three-screen app with the following behaviour

1. **Landing Page** – A textview field allowing the user to input a word. There is a button underneath that that transfers the word to Screen 2
2. **Screen 2** – A text field showing the word transferred from the Landing Page in uppercase. There are two buttons that either
    a. Navigates to the Landing page. The textview field should be empty, allowing another value to be inputted.
    b. Goes to Screen 3
3. **Screen 3** – A text field showing the number of vowels in the word. There are two buttons that either
    a. Navigates to the Landing Page. The textview field should be empty, allowing another value to be inputted.
    b. Goes to Screen 2

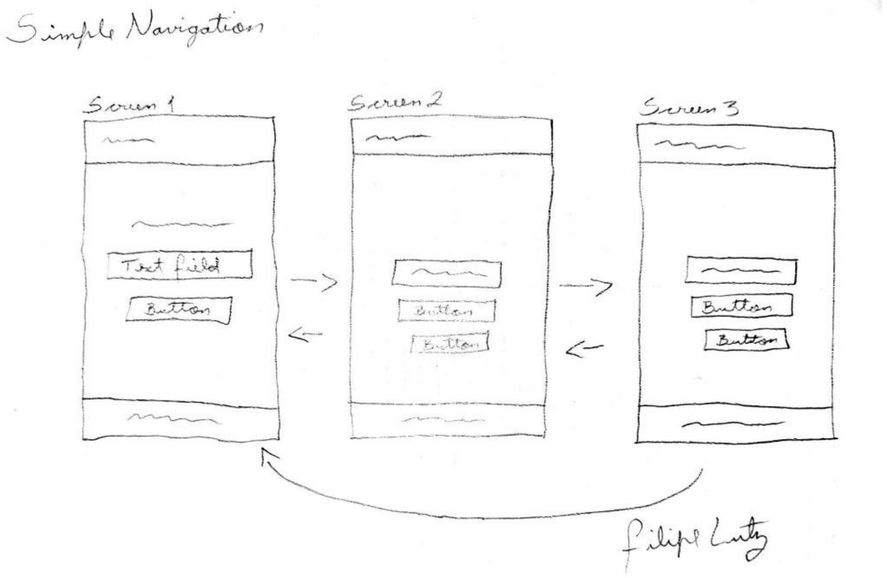
Marks for Section B

a. Wireframe and documented design considerations – *1.5 marks*

b. Code – *3 marks*

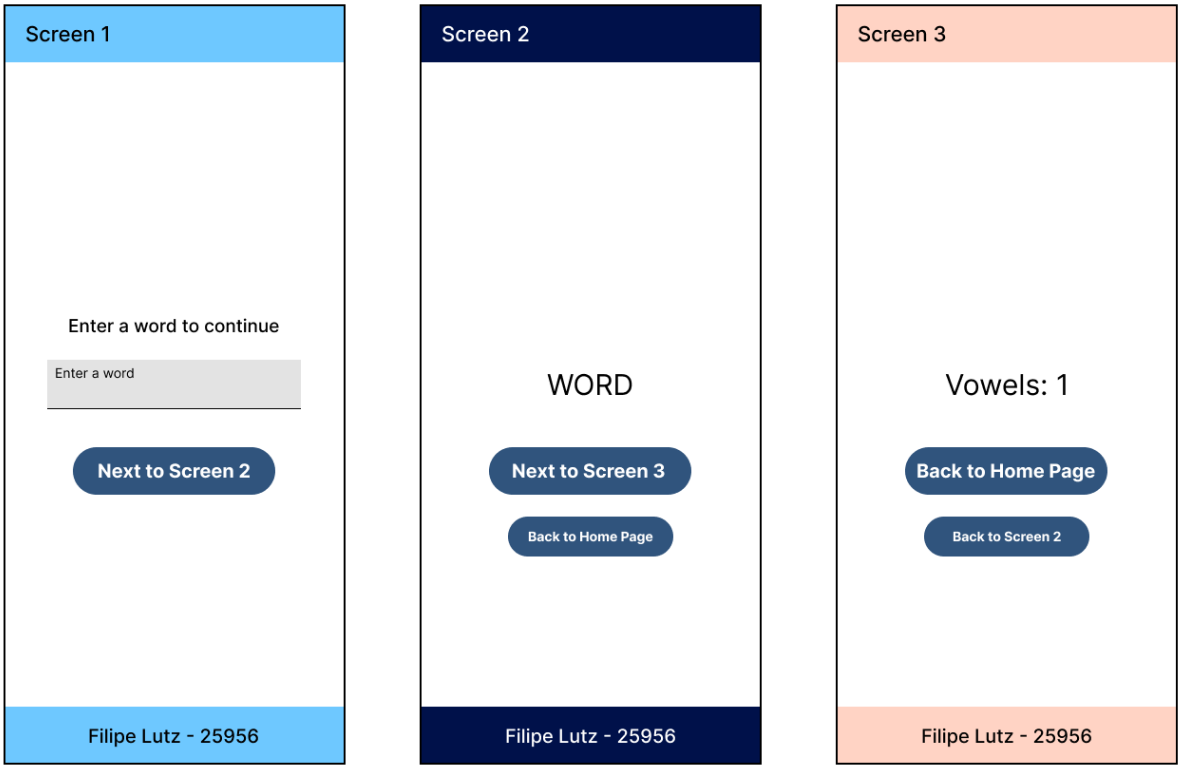
c. Unique formatting or functionality features - *.5 marks*

a. Low-Fidelity Wireframe:



a. High-Fidelity Wireframe:

- Figma Simple Navigation 3 Screens



b. Code:

- GitHub Simple Navigation

c. Unique formatting or functionality features

As unique formatting and functionality, I implemented

- Scaffold
- topBar and bottomBar
- Title on the tobBar
- SnackbarHost(snackBarHostState)

References:

1. Intent Extras in Android:

   - Intents and Intent Filters
   - Using Intent Extras

2. ViewModels and LiveData:

   - ViewModel Overview
   - LiveData Overview
   - App Architecture Guide

3. State Management Libraries:

   - State Management in Android

4. Lambda Functions in Kotlin:

   - Kotlin Lambda Expressions
   - Jetpack Compose Basics

5. Jetpack Compose Navigation for Beginners

   - Android Studio Tutorial YouTube
   - Manage State in Jetpack Compose YouTube

6. Counter user input word

   - Handle user input
   - Counter Text with Jetpack Compose