Assignment 3

Sicheng Chen csc0007@bu.edu (Thursday Section)
Siyi Du dsharon@bu.edu (Tuesday Section)
Team Name: Tuesday And Thursday

2. Sensor

a. What design questions do you need to ask before beginning?

What threshold value should be set for "significant" movement initially?

Start with a threshold of 1/5 the maximum accelerometer value, adjusting based on experimentation.

How should the SeekBar look and function to allow easy threshold adjustments?

Horizontal SeekBar on the top, with values ranging from the least to most sensitivity.

b. What are the functions that need to be written?

onPause(): Called when the system is about to put the activity into the background or when the activity becomes partially obscured.

onSensorChanged(event: SensorEvent?): Invoke this function when there's a new sensor reading. It's used to detect and act on significant movements.

c. What exceptions do you need to handle and how should you handle them?

Sensor not available: Display a message that the device doesn't support this feature.

Sensor malfunctioning: Display an error message.

d. What types of feedback do you need to provide the user as they interact with your App?

Display a toast when there's a significant movement.

Update the log in real-time to show significant movements.

Visually indicate the current threshold level on the SeekBar.

3. FlashApp

a. What design questions do you need to ask before beginning?

How should the "significant" fling gesture be defined and distinguished from regular touch?

Define fling based on velocity and distance to prevent accidental triggers.

What visual feedback should be given to indicate flashlight status?

Use a visual indicator that changes between ON/OFF states.

b. What are the functions that need to be written?

turnOnFlashlight(), turnOffFlashlight(): Turns flashlight ON or OFF based on the given status.

addTextChangedListener: Reads input from the Action Text Box and acts accordingly.

onFling: Determines if the user performed a fling gesture.

c. What exceptions do you need to handle and how should you handle them?

Flashlight not available: Display an error message.

Camera permission denied: Ask the user to grant the necessary permissions.

d. What types of feedback do you need to provide the user as they interact with your App?

Visually change the toggle switch state (ON/OFF) based on the flashlight status.

Display a toast or a brief message for certain actions (like display an error message).

Pop out the Action Text Box when click on the enter space.

4. FiveActivities

a. What design questions do you need to ask before beginning?

Does the app require the use of a ViewModel to retain states during screen rotation?

For this app, given that it must support screen rotation, using a ViewModel to preserve states is essential.

How many layouts in the APP?

The app will feature five activities, therefore, five distinct layouts are required.

b. What are the functions that need to be written?

In the MainActivity, the onCreate() function will be overridden. Additionally, the following methods have been overridden: onFling (to execute specific actions upon detecting a fling), onSensorChanged, onUpwardFling, onDownwardFling, onRightFling, onLeftFling, and onTouchEvent.

c. What exceptions do you need to handle and how should you handle them?

The app has minimal exceptions, primarily focusing on saving the current state to the ViewModel during screen rotations.

d. What types of feedback do you need to provide the user as they interact with your App?

If users shake the phone, a TextView will shake for 2 seconds as feedback. Upon detecting a fling, a new activity will launch, informing the user of the fling direction.

5.Hangman

a. What design questions do you need to ask before beginning?

Will the app support both portrait and landscape displays?

The app is designed to function in both portrait and landscape orientations.

Does the app need a ViewModel to maintain states during screen rotation?

Yes, to ensure consistent user experience during screen rotations, the app will utilize a ViewModel.

b. What are the functions that need to be written?

For MainActivity.kt, potential functions are:

Setting click listeners for all buttons (setOnClickListener()).

Enabling all buttons (enableAllButtons()).

Checking the current status (checkStatus()).

Updating answers (updateAnswer()).

For HangmanViewModel, potential functions are:

Generating questions (generateQuestion()).

Checking answers (checkAnswer()).

c. What exceptions do you need to handle and how should you handle them?

During the design and testing phases, several challenges emerged.

List overflows.

Random variable changes.

State alterations post-screen rotation.

To address these, I employed the following strategies:

Utilized the ViewModel to retain the app's state, ensuring continuity.

Engaged debugging tools to pinpoint when and where the list overflows occurred.

Kept a close watch on variable values to monitor and rectify any inconsistencies.

d. What types of feedback do you need to provide the user as they interact with your App?

User will be notified how many chance they have left by the hangman image change. Also, the button will be disable after used. After win or lose, there will be text displayed.

Team GitHub Repo: csc0007/CS501_GroupWork: Group Work Only (github.com)

Backup: https://github.com/csc0007/CS501_GroupWork

For homework code and details, please refer to the GitHub repo.