Hourglass – Time management app

50.001 Team 2-A

Github: https://github.com/Samillynn/HourGlass

Members

Lin Yutian	1004881
Liu Renhang	1004873
Meng Fanyi	1004889
Lee Pei Xuan	1005513
Sun Zhengnan	1004882
Lim Hong Jun, Joshua	1005259
Lim Si Hui Brenda	1004578

Description

The advent of digital devices has helped increase many students' academic productivity, and also made various recreational apps available, which may serve as distractions. Although taking short breaks on such apps can be good for productivity, students are often tempted to procrastinate on their work due to the addictive nature of these recreational apps.

Hourglass is a focus app which aims to help individuals to manage their time better by limiting the amount of time they spend on such apps. Hourglass thus allows students to take short phone breaks without worrying about not having the discipline to get back to work.

Hourglass helps the user time their rest and focus activities and notifies the user once time is up. During the focus period, access to apps, other than those whitelisted, are denied. In Settings, the user can configure his default rest/focus time length, customize the whitelist apps, and write motivational messages to be sent to themselves at specified times. The app adopts a fresh and clear style, combining visual comfort and ease of usage.

Further improvements include implementing a user database for account systems, introducing a friend interaction feature, and a points and achievement system.

System Design and Implementation

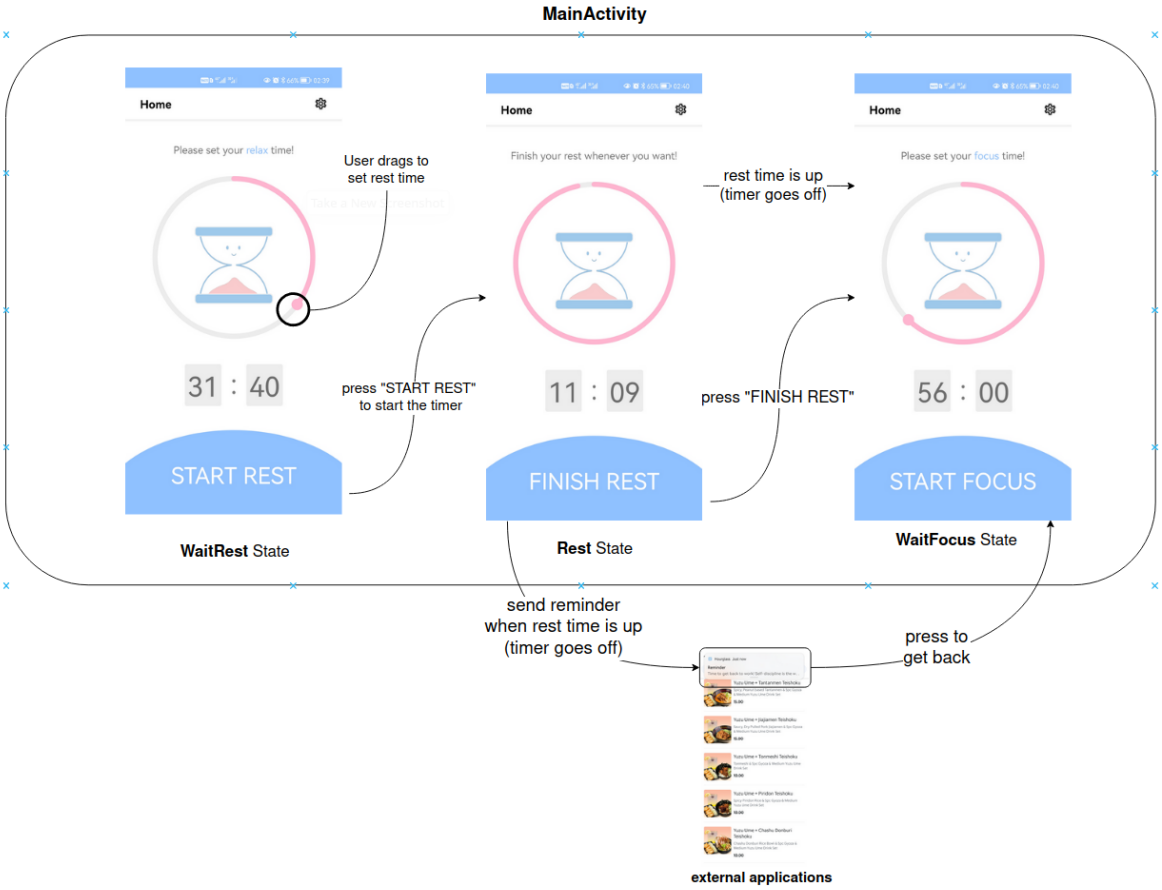
Start and Finish Rest

Once the user enters our app, the user is in the **WaitRest state**. The user can set a rest time by dragging the progress bar, and the user enters **Rest state** by pressing the “START REST” button. Then MainActivity enters **WaitFocus** mode in the following 2 situations:

1. Rest time is up (the timer goes off).
2. The user pressed “FINISH REST” to finish rest early.

If the user is playing other apps when the timer goes off, a notification will be sent to remind the user to go back.

The diagram demonstrates how the **three states** interact with each other.



Implementation Details:

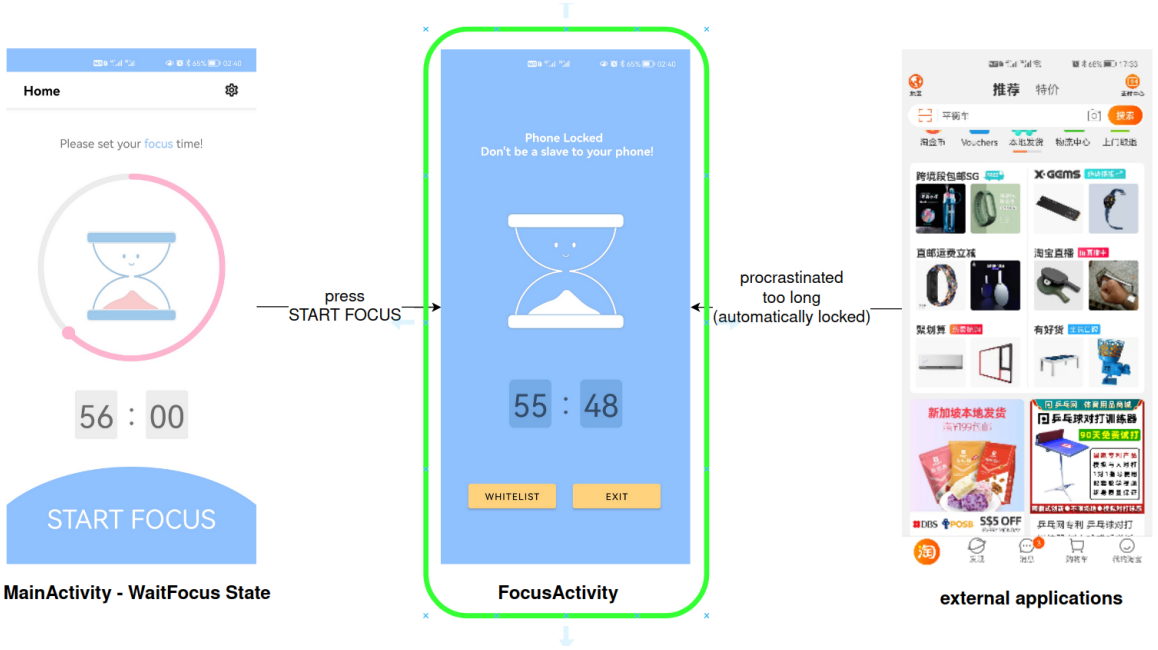
An abstract HomepageState class is used to implement the three modes: WaitRest state, Rest state, WaitFocus state. This makes it easy to switch between different states and allows for assets such as timeViewModel, timeSegment and circularSeekBar to be easily reused for both modes.

Start Focus Mode

After entering **WaitFocus** state, the user is ready to enter the focus mode. The user can set how long he/she wants to be focused by dragging the progress bar. The FocusActivity will start in the following 2 situations:

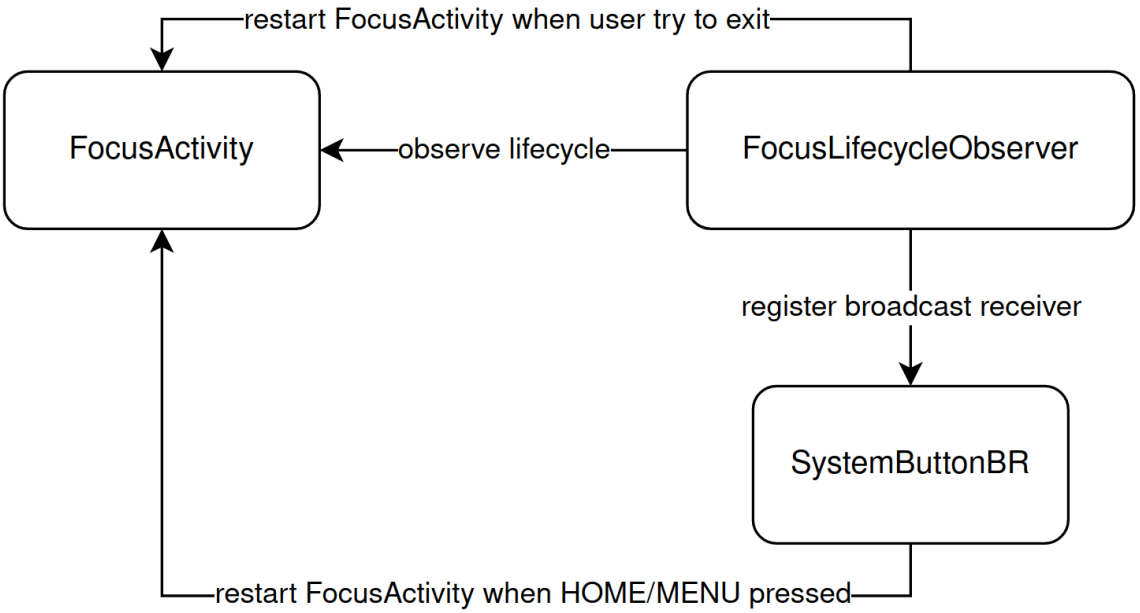
1. The user presses the “START FOCUS” button.
2. The time that the user has procrastinated in other apps reached the snooze time the user set in the settings page (or 2 minutes by default). In this case, the focus timer will be set to the default focus time the user set in the settings page (or 25 minutes by default).

The following diagram describes how the user enters focus mode.



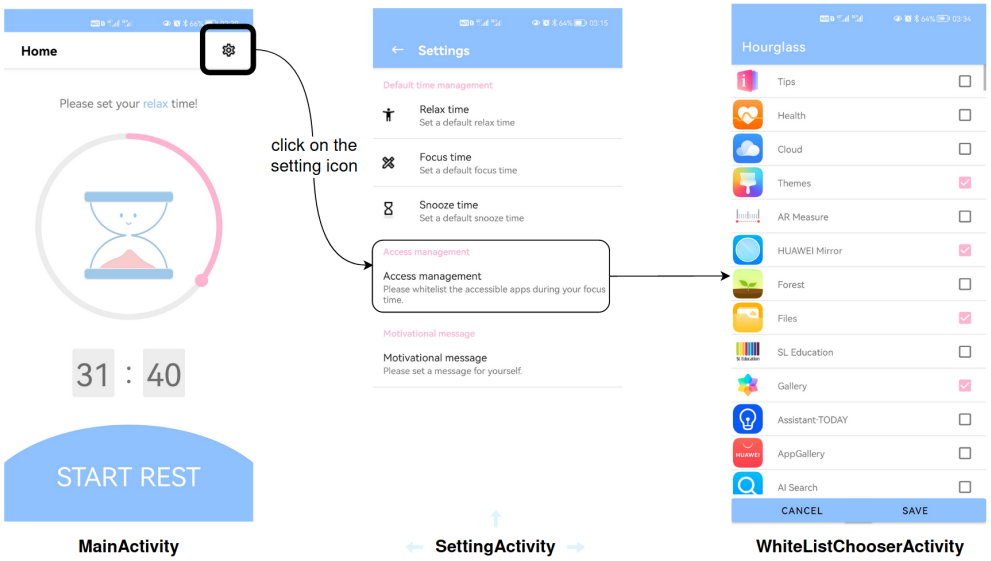
Stay in Focus Mode (Lockdown the Phone)

Once the user enters the focus mode, the user is banned from leaving the activity (FocusActivity). To implement the lockdown, the FocusLifecycleObserver class is used to observe the change in the state of the lifecycle of the FocusActivity. When the user tries to leave FocusActivity, the FocusLifecycleObserver will restart FocusActivity. The FocusLifecycleObserver also registers a broadcast receiver called SystemButtonBR. Whenever the user presses the HOME or RECENT_APPS buttons, SystemButtonBR restarts the target activity (FocusActivity in this case). By combining FocusLifecycleObserver and SystemButtonBR, our application is capable of preventing the user from leaving the focus mode. The system architecture follows.



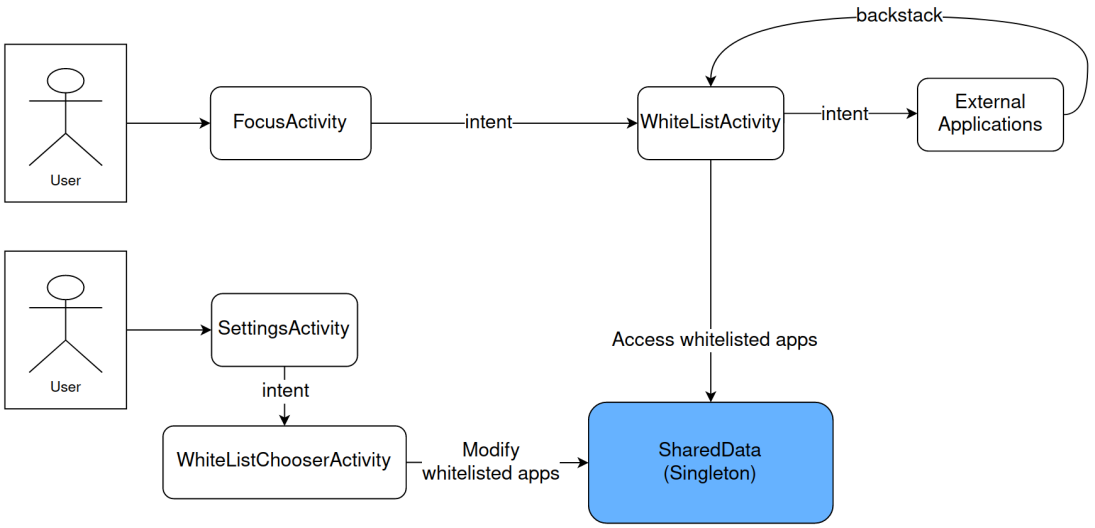
Open Whitelisted Apps in Focus Mode

WhiteListChooserActivity and WhiteListActivity are used in conjunction to implement the whitelist function for Focus mode. WhiteListChooserActivity scans the phone for installed apps to allow the user to choose their whitelisted apps. The diagram below shows how the user is able to choose the whitelisted apps.

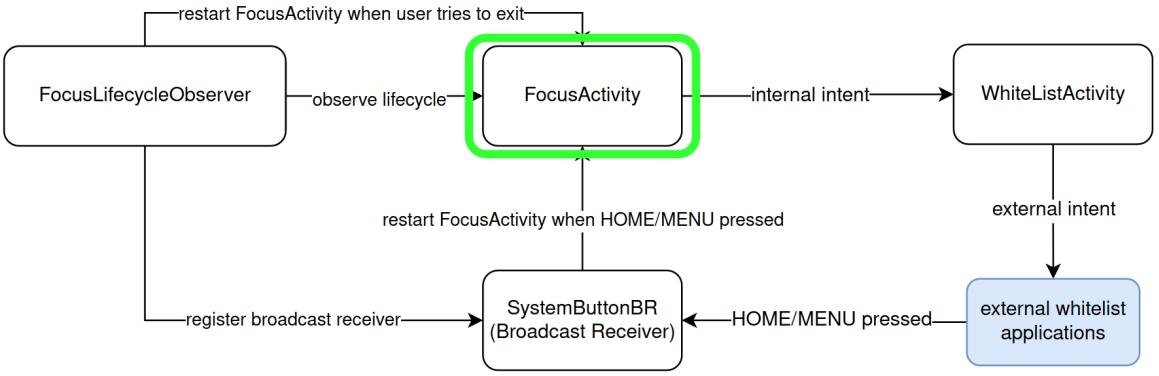


WhiteListActivity sets up the whitelist activity for users to access their whitelisted apps in the focus below. The diagram below shows the procedure.

The system architecture to modify and access the whitelisted apps follows.



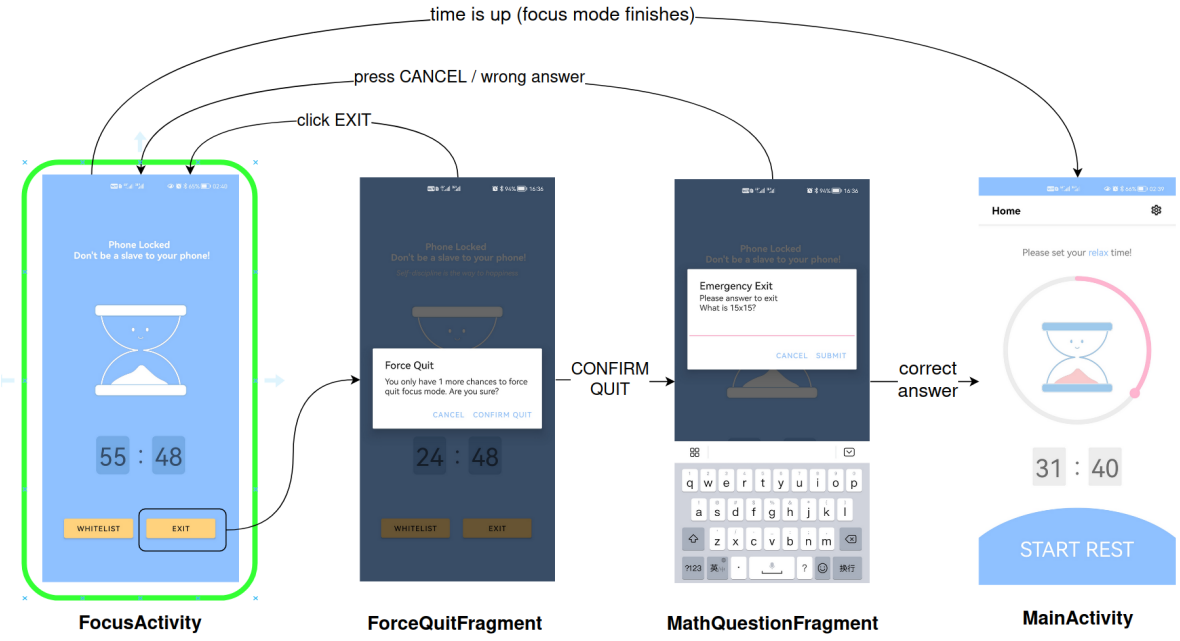
The following system diagram illustrates how FocusActivity integrates with WhitelistActivity, which ensures the user will not break the lockdown mode.



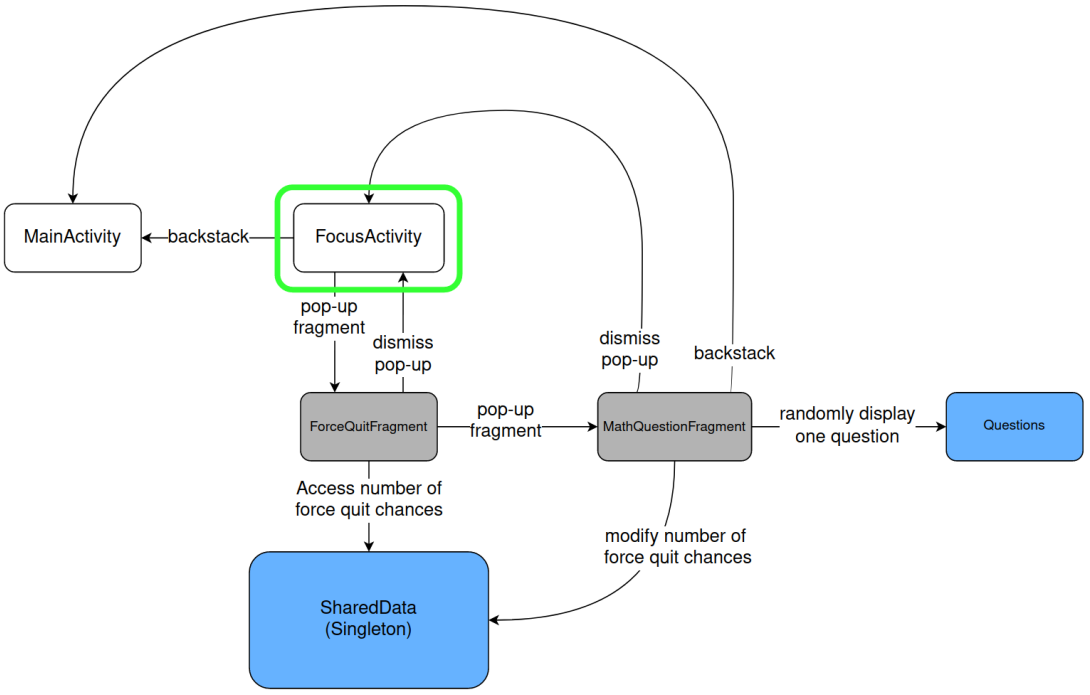
The timer is implemented using the Timer and TimerViewModel class. The former uses CountDownTimer to create the countdown sequence while the latter is used to display the timer in the app.

Exit the Focus Mode

In our design, every user only has 3 chances to forcibly exit the focus mode. If the chances are used up, the user has to wait until the timer goes off. To forcibly exit, the ForceExitFragment makes use of the dialog to cover the FocusActivity, and prompt the user with a confirmation message, if the user still has enough chances to exit. An easy math question will also pop up when they confirm to exit. If the user answers correctly, the FocusActivity exits. If not, the dialog closes and the user remains in Focus mode. The math question design aims to give our users a time buffer, so that they can think twice whether they really need to forcibly exit the focus mode. The diagram below shows how the user can exit the focus mode.



The system architecture follows.



Concepts and their applications from Information Systems and Programming

1. Design Patterns
    a. Singletons:
    We implement a singleton SharedData class, which is a centralized and unique data manager for all other classes to access and modify.
    b. Observers
    LifecycleObservers: we implement a FocusLifecycleObserver class which observes the life state of FocusActivity, and behaves accordingly.
    c. States
    In the implementation of MainActivity, we used State pattern to switch between different modes to reuse components in MainActivity
    d. View-Model-Viewmodel
    In MainActivity and FocusActivity, to synchronize the UI time related component and the background timer, we use View-Model-Viewmodel pattern to split the process.
2. Abstract Class
    We implement an abstract class called HomepageState, so that the different states in MainActivity can share the same codes and interface.
3. Fragments
    We used fragments in SettingActivity and FocusActivity
4. ListViewAdapter
    In the setting page, to allow the user to choose their whitelisted applications, we implemented a ListViewAdapter to display all the user applications.
5. LifeCycle
6. Explicit/Implicit intents
7. OnActivityResult

Contributions

Name	Contributions
Lin Yutian	Code, Report
Liu Renhang	Code, Report
Meng Fanyi	Code, Video
Lee Pei Xuan	Code, Report
Sun Zhengnan	Code, Report
Lim Hong Jun, Joshua	Code, Report
Lim Si Hui Brenda	Code, Poster