Pacman

MentAlly

Android App

Syeda Fareeha Fatima
Navya Gupta
Kavya Mehta
Jasmine Tang
Alexangre Gagne
Syeda Malaika Zaidi
Viju Hiremath

Table of Contents

System Design

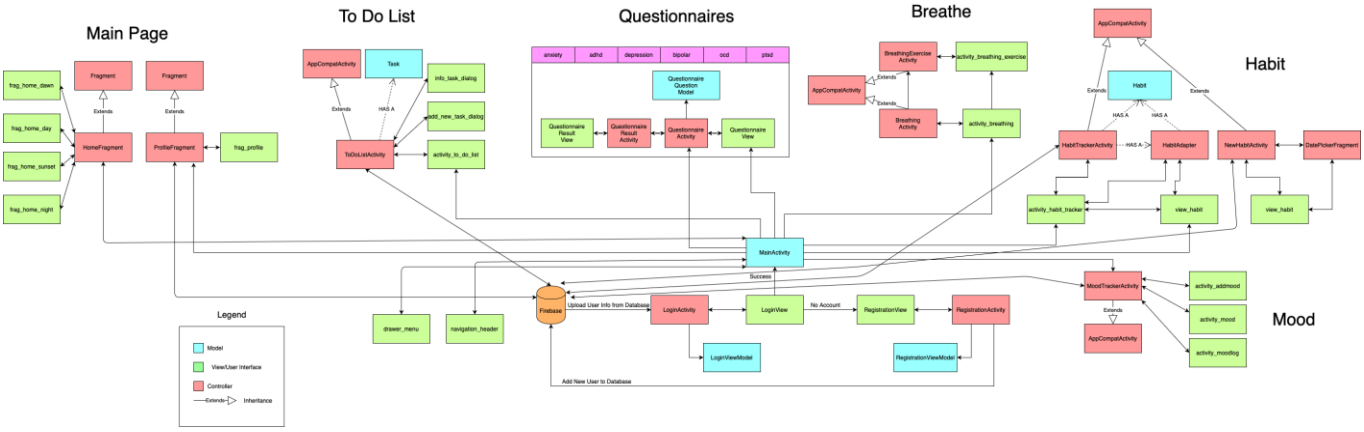
We will be building an Android app in Java. The app will be built using Android Studio and we use Firebase as the database to store user information. Firebase will be used to add and retrieve user data and authenticate new and returning users. Since the app is created for android users the operating system must be android. Our app requires a connection to the internet as some features of our app will not work without the internet including access to music and logging in via the database. The app is built to work with a minimum of API 28, i.e. Android 9.0 Pie. Support for earlier systems is not guaranteed.

Our app is divided into modules and we will be applying MVC during development. The view will correspond to the user interface. The controller will correspond to Android Activities which collect user data and input from the view and request action or send the information to the model. The model will call required methods then inform the controller of the result. The controller will then decide and make the appropriate changes to the view. Since we have multiple modules that work independently, each module will likely work with its own model, view and controller. The separate views will interact through the controllers which will switch to different views based on user input. The Activities will be responsible for interacting with the database when needed to authenticate user info, update it or delete it. Please see the System Architecture Diagram for a high-level diagram of the architecture.

As we are dealing with a user application, there are many ways that errors may occur in the program. To handle these errors there are various checks put in place. First, all user input must be validated before it is inputted into the database to prevent any issues in the database. Secondly, users must register an account and login to use all app features. Users cannot make multiple accounts with the same email address. These features will ensure that only verified users are able to access information inside the app and it will protect user data. Additionally, errors may occur if the database is down or there is network failure. In these cases, the app should notify users that they must connect to the internet or give them a message that an error has occurred, and users will be notified as soon as the problem is resolved. This will keep users from feeling irritated as they will be informed about the problem.

System Architecture Diagram

**NOTE TO TA: As below picture isn't clearly visible, the system architecture diagram will be in the sprint 2 directory as well. **



CRC Cards

LoginViewModel	
Parent: ViewModel	
• Responsible for validating user input and setting error state in LoginFormState	LoginFormState

RegistrationViewModel	
Parent: ViewModel	
• Responsible for validating user input and setting error state in RegistrationFormState	RegistrationFormState

LoginActivity	
Parent: AppCompatActivity	
• Responsible for gathering information from view • Use information from view to sign in user • Send information to LoginViewModel to validate input • Reads error/validation state from LoginFormState and makes appropriate changes to user view • Sends user to registration view on request to register	LoginFormState LoginViewModel LoginViewModelFactory RegistrationActivity

RegistrationActivity	
Parent: AppCompatActivity	
• Responsible for gathering information from view • Responsible for registration of user and adding user information to database • Sends user input to RegistrationViewModel for validation • Reads error/validation state from RegistrationFormState and makes appropriate changes to user view	RegistrationFormState RegistrationViewModel RegistrationViewModelFactory

LoginViewModelFactory	
Implements: ViewModelProvider	
• Initializes a new LoginViewModel	

RegistrationViewModelFactory	
Implements: ViewModelProvider	
• Initializes a new RegistrationViewModel	

LoginFormState	
• Responsible for storing the error state of user login input	

RegistrationFormState	
• Responsible for storing the error state of user registration input	

BreathingActivity	
Parent: AppCompatActivity	
• Responsible for gathering information from view • Responsible for taking in values the user inputs to customize their breathing exercise. • Sends the user values to BreathingExerciseActivity so it can use them to start the exercise.	

BreathingExerciseActivity	
Parent: AppCompatActivity	
• Responsible for timing the exercise and displaying it on the view accurately.	

MoodActivity	
Parent: AppCompatActivity	
• Responsible for gathering information from view • Responsible for taking in values from the database to fill in the mood log. • When add mood button is clicked it is responsible to take the user to AddMoodActivity where the user can create add a new log to their mood log.	

AddMoodActivity	
Parent: AppCompatActivity	
• Responsible for taking in user input for date, mood type, and description from the view and store it in the database. • When the add button is clicked it stores the user information on the database and takes user back to MoodActivity home page to view the new mood log.	

HomeFragment	
Parent: Fragment	
• Responsible for displaying the home screen once the user has logged in	

ProfileFragment	
Parent: Fragment	
• Responsible for displaying user information • Responsible for allowing user to change their personal information	

MainActivity	
Parent: AppCompatActivity	
• Responsible for displaying the menu bar • Responsible for allowing user to navigate to different pages	HomeFragment ProfileFragment

ToDoListActivity	
Parent: AppCompatActivity	
• ToDoListActivity keeps track of the completed and incomplete tasks. It relies on the database to get this information for the user's to do list. • Tasks can be added, deleted, updated, and marked as complete. • When changes are made, it is responsible for updating the database taskLog for that user. • Also responsible for updating the view according to changes made.	Task

Task	
Parent: None	
• Task Class responsible for storing the taskname, taskId, startDate, finishDate, and completed (i.e. whether a task has been completed or not) for a task. • Responsible for allowing access to Task attributes • Responsible for updating Task attributes	None

AnxietyActivity	
Parent Class: Activity	
• Allows the user to take a GAD-7 (Diagnosis of Anxiety) questionnaire. • Based on user choices, calculate result and informs user about their level of anxiety	

DepressionActivity	
Parent Class: Activity	
• Allows the user to take a PHQ-9 (Diagnosis of Depression) questionnaire. • Based on user choices, calculate result and informs user about their level of depression	

PTSDActivity	
Parent Class: Activity	
• Allows the user to take a questionnaire which checks if the user could have probable PTSD. • Based on user choices, calculate result and informs user if they have probable PTSD	

adhdActivity	
Parent Class: Activity	
• Allows the user to take a questionnaire which checks if the user has symptoms of ADHD. • Based on user choices, calculate result and suggest user if they have ADHD	

biopolarActivity	
Parent Class: Activity	
• Allows the user to take a questionnaire which checks if the user has symptoms of being bipolar • Based on user choices, calculate result and suggest user if they have bipolar tendencies	

ocdActivity	
Parent Class: Activity	
• Allows the user to take a questionnaire which checks if the user has symptoms of OCD. • Based on user choices, calculate results and suggest users if they have OCD	

Habit	
Parent Class: None	
• Hold information about a single habit	

HabitTrackerActivity	
Parent Class: AppCompatActivity	
• Responsible for gathering information from view • Responsible for taking in values from the database to fill in the habit list. • When add habit button is clicked, it is responsible to take the user to NewHabitActivity where the user can add a new habit to track	Habit, HabitAdapter, NewHabitActivity

HabitAdapter	
Parent Class: RecyclerView.Adapter<HabitAdapter.HabitView>	
• Responsible for populating view of habit list based on information pulled from database • Notify listener when there is a change in data from user clicking button to increase progress	Habit, HabitTrackerActivity

NewHabitActivity	
Parent Class: AppCompatActivity	
• Responsible for gathering information from view • Responsible for updating database with new habit • When add habit button is clicked, it is responsible to take the user to HabitTrackerActivity where they can see the list of habits	HabitTrackerActivity, DatePickerFragment

DatePickerFragment	
Parent Class: DialogFragment	
• Responsible for setting up a datepicker dialog view • Update new habit view to reflect selected date	NewHabitActivity