CSE 535: Individual Report – Project 4

Group 51

Vishrut Rajendra Kumar Harsoor
Email: vharsoor@asu.edu
ASU ID: 1230036628

Alignment with Guardian Angel

For our Project 5, we aim to create an application that assists people in staying focused and managing their daily routines more effectively. My contribution to the app involves helping users stay focused while working on important tasks by blocking notifications from distracting apps. Additionally, it facilitates the organization of their events and tasks through calendar integration.

Specifications

Our app is primarily divided into two modes:

1. Focus Mode: Designed for working on important tasks.
2. Leisure Mode: Created for unwinding and relaxation. My contribution falls under the Focus Mode and is divided into two components: Notification Manager and Calendar Integration.

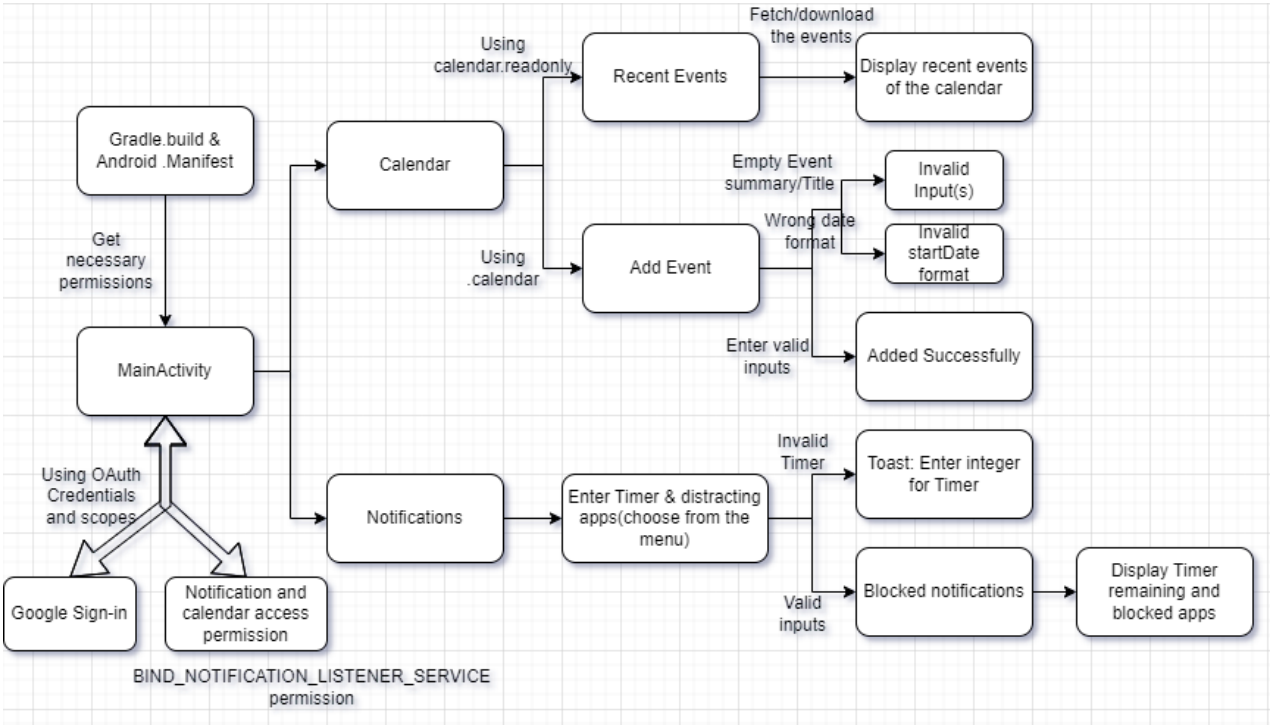
The key components utilized for this purpose are outlined below:

1. Notification Manager:
   1.1 NotificationListenerService: This service receives a call when a new notification is posted or removed.
   1.2 StatusBarNotification: This component allows us to obtain a notification as an object, enabling further operations.
   1.3 BIND_NOTIFICATION_LISTENER_SERVICE permission: This permission is required to access the aforementioned classes.

2. Calendar Integration:
   2.1 Google Calendar API: This RESTful API, provided by Google, was instrumental in seamlessly integrating Google Calendar with our application.
   2.2 Oauth 2.0 credentials: These credentials, designed for a web application (not Android), are utilized to access the Google Calendar account, utilizing the web client ID for authentication.
   2.3 Oauth Consent screen: This feature proves valuable for obtaining user consent for specified scopes (e.g., **readonly** for viewing or downloading the calendar, **/auth/calendar** for editing and sharing the calendar) within our project.
   2.4 Event Model: This represents the structure of an event, facilitating either the display or addition of new events to the calendar.

Design

Tech Stack Used: Android Studio, Kotlin (API 33 version), Emulator and Google Calendar API.



Testing Strategies

1. Add/show events from the Google Calendar:
   1.1. Create/Sign-in into Google Cloud console and enable Google Calendar API.
   1.2. Generate OAuth credentials (Web Application type, not Android). During the configuration of the OAuth consent screen, ensure to enable the required scopes auth/calendar.readonly & auth/calendar) and specify the project name exactly as it is stated in the Android manifest. (Ex: package="com.example.proj5")
   1.3. Utilize the Web Client ID obtained from OAuth and add/replace it in the "web_client_id" field in the strings.xml file.
   1.4. Enable all necessary permissions when the app is launched for the first time and sign into the corresponding Google account (following the prompt).
   1.5. Navigate to the calendar page and click on "Recent Events" to display your recent events from Google Calendar.
   1.6. On the page, to add an event, input the event summary/title along with the start date of the event (in the displayed format).
   1.7. Entering invalid inputs will trigger an error, while entering valid inputs will successfully add the new event.

2. To cancel/block the notifications from the distracting apps based on a timer:
   2.1. Enter the timer duration (in hours) and start typing the names of the apps from which you want to block notifications; a menu option will pop up for selection.

2.2. If a non-integer value is entered in the timer, a toast error message will be displayed.
2.3. Entering valid values will initiate the timer, blocking notifications from the selected apps until the timer expires.
2.4. The UI displays the remaining timer duration and the list of blocked apps.

Navigating Challenges

1) UserRecoverableAuthIOException Error:

a) Despite granting access to my Google account for the project and using the correct web client ID, I encountered this error.

b) I attempted to create a new OAuth credential using various Google accounts, but none of them proved successful.

c) After a day of troubleshooting, I discovered the necessity of providing an intent to prompt the app to display an "Allow access" window from Google.

2) "Something went wrong. Error 400." Error:

a) Following the selection of the Google account, handling the UserRecoverableAuthIOException exception, and providing an intent, a Google prompt threw an error: "Something went wrong. Error 400."

b) Apparently, this issue is a known bug on Google's side. A Stack Overflow solution suggested that to address this error, it's recommended to sign in with a single Google account instead of choosing from multiple signed-in Google or Gmail accounts. In simpler terms, using a new emulator should resolve the issue. We need to do this only the first time we launch the app.

c) I learnt that this occurs because Google encounters difficulty in determining the chosen account. Using a single account helps resolve this issue with Google's authentication process.