**CS551 Resit Assignment**

**Submitted by:**

**Joel Joseph**

**202381605**

**1. Introduction**

“The Calorie Quest” app, an Android application that allows daily monitoring and control of the number of calories consumed, helps users to live a healthy lifestyle. The app is developed to guide users how to be mindful while eating and capable of making their own decisions through critical thinking to secure their food consumption with their health objectives.

**2. Functionalities implemented**

The application was developed using Kotlin as the programming language and Jetpack Compose toolkit. MVVM architecture was employed to assure the application’s maintainability and smooth operation. The app’s main features are listed below, and they were implemented to engage users in the process of dieting:

**Main screen features:**

* User can manually enter the calorie intake.
* The main screen displays the total calorie intake of the
* It also calculates and display the percentage of the daily calorie goal consumed.
* The app can automatically reset the calorie counts on a new day.
* The app persistently stores daily calorie intake using Room.
* The settings screen is accessible from the main screen through an App Bar overflow menu option.

**Settings screen features:**

* User can set their daily calorie goal.
* The setting screen provides user the ability to set their motivation and ability level as high or low.
* Use can input the GPS coordinates of their favourite food store.
* Using Appbar the user can navigate to main screen.
* Users’ preference can be stored in Datastore preference.

**Contextual triggers:**

* Location-based trigger is implemented using geofence. The trigger will be activated near the vicinity of their favourite food store.
* Time-based trigger is implemented using Work Manager worker. It will be activated every 2 hours during the day between 7:00 and 22:00 and when the device is not low on battery.
* There are three major types of notifications based on user's motivation and ability(Spark, Facilitator and Signal).
* The time-based notifications will also show the current percentage of the users’ daily calorie goal that has been consumed.

### **2. Implementation Details**

Below are the specifics on how the Main and Settings Screens were implemented. It also highlights how Room was used for data persistence, Preference data store for saving user preferences and the utilization of Viewmodels and its factories.

#### **2.1 Jetpack Compose for UI Design:**

Jetpack Compose fundamentally changes the way Android developers design UI thanks to its declarative nature. Main Screen has been adapted to update UI elements as the state changes. An excellent example can be found in the calorie data and its display progress. The obvious illustration of the above is the @Composable functions MainScreen and MainContent . Enhanced interactivity is evident in the form of the Settings Screen which uses fields responsible for entering calorie goal, motivation level, ability level, and coordinates of favourite food store locations.

**2.2 MVVM Architecture:**

The app's architecture was implemented using MVVM principles. This architecture was beneficial in dividing the app's logic into manageable components:

* **Model**: It represents the app's data layer. It includes the database Room and Data entities. CalorieEntry is the primary data entity, encapsulating the attributes for each calorie record.
* **View**: It comprises the UI components built using Jetpack Compose. The MainActivity is the container for the UI.
* **ViewModel**: MainViewModel and SettingsViewModel compute UI data processing from the View. They deal with updating and retrieving data, communicating with the Model, and informing the View of any modifications. SharedViewModel shares the calorie goal across different screens, ensuring consistency in the user's goal settings.

#### **2.3 Data Persistence with Room:**

Room was used for storing daily calorie intake data. The “AppDatabase” class sets up how the database looks and works, including the CalorieDao for accessing the data. This DAO (Data Access Object) was used for inserting calorie entries and retrieving the total calories for the current date. The MainViewModel interacts with CalorieDao, so it can fetch and update calorie data.

#### **2.4 Preference Data Store for Settings:**

For persisting user settings, Preference Data Store(Preference manager) was used. These settings include the daily calorie goal, toggles for motivation and ability levels, and the GPS coordinates of the user's preferred food store.

#### **2.5 ViewModel and Factory Implementation:**

There are 3 types of viewmodel factories in the application, namely MainViewModelFactory, SettingsViewModelFactory, and SharedViewModelFactory. They are used to initialise ViewModels with specific dependency like data access objects (e.g., CalorieDao, PreferencesManager) for their operation. SharedViewModel is used to maintain a consistent state across the Main and Settings Screens, particularly the calorie goal. This approach ensures that changes to the calorie goal in the Settings Screen are immediately reflected and accessible in the Main Screen.

**2.6 User Interface & Navigation**

**Main Screen**: It serves as the app’s entry, enabling calorie input through OutlinedTextField and a submit button, with Text composables showing live updates of calorie counts and goal progress.

**Settings Screen**: Users set daily calorie goals, adjust motivation and ability level using Switch for toggles, and enter preferred food store GPS coordinates, using OutlinedTextField for input. A save button at the screen’s end ensures easy preference storage.

Both screens use a Scaffold for the top bar. The main screen can access the settings screen through App Bar overflow menu option. And there is an App Bar back arrow in the settings screen, by which the user can navigate back to the main screen. This is done through Jetpack Navigation. The Jetpack Navigation provides smooth transitions between the Main and Settings Screens. The navigation is done in the MainActivity with a NavController and a NavHost with two routes: "main" for the Main Screen and "settings" for the Settings Screen.

**2.7 Contextual Triggers:  
2.7.1 Location-based Trigger:**

The location-based trigger is designed in such a way that its alert users when they are near to their favourite food store, encouraging them to make healthier food choices. This is done through Geofencing.

* **Geofence Setup**: Using the GeofencingClient from Google Location Services, a geofence is constructed. The radius is specified to have 30 meters around the GPS coordinates of the user’s favourite food store. This geofence is set to never expire and to trigger upon entering this defined area.
* **Geofence Trigger and Notification**: When the geofence transition event is detected, a broadcast receiver (GeofenceBroadcastReceiver) handles this event. The receiver evaluates the user's motivation and ability settings and selects an appropriate notification type (signal, spark, facilitator or low) based on preference set by the user. Each notification type carries a tailored message to nudge the user towards healthier choices, incorporating NHS advice on “Managing Your Weight”.

For the Geofence to be working properly, Location permission should be set as “Allow all the time”. Provided the notification permission is allowed.

#### **2.8.2 Time-based Trigger:**

The time-based trigger is operated through WorkManager worker. It executes a background task every 2 hours from 7:00 AM to 10:00 PM, provided the device is not low on battery.

**WorkManager Setup**: A PeriodicWorkRequest for this worker (TimeWorker) is set up to only run only when the device has a sufficient battery level. The worker checks the current time to make sure it falls within the specified window (i.e., between 7 AM and 10 PM)and it also calculates the user's current calorie intake percentage against their daily goal.

* **Notification System**: Depending on what the user has set their preferences for motivation and ability, the TimeWorker class will decides which type of notification to display. This decision-making process is similar like that of the location-based trigger, aiming to provides a motivational notification(Signal, Spark, Facilitator). Each notification type carries a tailored message to nudge the user towards healthier choices, incorporating NHS advice on Managing Your Weight. This notification additionally shows the daily calorie goal of the user. The notification permission should be allowed in the app’s permission setting.

#### **2.9 Types of Notifications:**

The app uses BJ Fogg's behaviour model to issue timely notifications—Signal, Spark, Facilitator, and an additional Low—providing users with varying motivation and ability levels to foster healthier habits.

* **Signal Notification**: For highly motivated users, serving as prompts for healthy decisions.
* **Spark Notification**: Motivates users with high motivation but lower ability by suggesting straightforward actions.
* **Facilitator Notification**: Helps users with ability but lacking motivation through easy steps towards better habits.
* **Low Notification**: Targets users with low motivation and ability, encouraging any positive action.

The app intelligently selects the notification type by analysing stored user data on motivation and ability, ensuring tailored encouragement.

**3. Screenshot of Main Screen & Settings Screen**

A screenshot of a cell phone

Description automatically generated A screenshot of a calorie tracking app

Description automatically generated

**Main Screen Settings Screen**

**3.2 Location-based trigger**

A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated

**Low Ability & Motivation Spark Notification Facilitator Notification Signal Notification**

**3.3 Time-based trigger**

A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated A screenshot of a phone

Description automatically generated

**Low Ability & Motivation Spark Notification Facilitator Notification Signal Notification**

**4. Strengths and Weaknesses of the Implementation**

There are still several areas of strength and weakness in the current state of the application.

**4.1 Handling Uncertainty in Context Information**

**Strengths**: The ability of the app to trigger location-based aims to encourage healthy foods when it matters most, this uses the power of technology that understand user’s context and improve user experience.

**Weaknesses**: Even though geofence has certain advantages, it still heavily relies on precise GPS data, and this data is variable. These inaccurate location data could lead to false notifications when user is not near their favourite store or miss notifications even when the user is near the store. Furthermore, the app only relies on user-reported calorie consumption and goals, this introduces variability in the app's effectiveness. However, by implementing a food database for logging accuracy and using predictive analysis could provide a better service.

#### **4.2 Challenges of Mobile Computing**

**Strengths**: The WorkManager used for scheduling time-based notifications efficiently use resources, minimizing battery and data consumption.

**Weaknesses**: However, the background operation is continuous, especially location tracking, which still pose a threat to increased battery drainage. Future updates can be done by understanding user behaviour pattern to opimize background task scheduling based on usage, this includes adjusting geofence check. Furthermore, the app’s UI and functionality to adapt vast Android devices remains a challenge.

#### **4.3 Security and Privacy Concerns**

**Strengths**: The app gives importance for user privacy by asking permission to access location and sending notifications, following Android’s guidelines.

**Weaknesses**: The current implementation does not have clear features for ensuring encryption of the sensitive data either when on the device or transmitted over networks. Incorporating AES encryption for data at rest and SSL/TLS for data in transit would enhance the security of the application. To address the privacy concern in a more holistic way, other measures that would ensure user transparency with regards to data culled as well as clear privacy policies accompanying the agreement could be implemented. Deleting data for users and giving them a full rundown of data currently stored on them, along with a GDPR compliance segment, could also go a long way in making sure users are happy and their data privacy is in check.