System Design Document
Application Name: My Plan Book
Team: The Brogrammers
Course: CSC301

Table of Contents

CRC Cards

Class Name: **LoginActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Allow the user to log in given an account and a correct username and password
- Give the user the option to sign up if no account exists for the user
- Check if the username and password are valid
    - If valid, proceed to the Planner Selector.
    - If not valid, prompt the user that the username and/or password is incorrect

Collaborators:

- Signup
- MainActivity

Class Name: **Signup**
Parent Class: AppCompactActivity
Responsibilities:

- Once the “Done” button is clicked, check if all the fields (first name, username, password, etc...) are filled and aren’t empty
    - If they are, go back to login page
    - If they aren’t filled, prompt the user to ensure all fields are filled

Collaborators:

- LoginActivity

Class Name: **CalendarEventsActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Show Calendar
- Know the current date
- Allow user to select a new date to see events for that date
- Add Calendar Events
- Remove Calendar events
- See Calendar Events
- Link to the main menu

Collaborators:

- CalendarEventsModel
- MainActivity

Class Name: **CalendarEventsModel**
Responsibilities:

- Add a calendar event
- Remove a calendar event
- Get calendar events for a specific date
- Save and alter calendar event data from a Firebase database

Collaborators:

- DatabaseHandler

Class Name: **EventNotifier**
Responsibilities:

- Notify the user of events that are due today as part of the calendar via Android phone notifications

Collaborators:

- NotificationCompat (Android API)

Class Name: **DatabaseHandler**
Responsibilities:

- Securely connect to the firebase database.
- Provide convenience functions for querying the database for other classes use.
- Cache recently collected data locally.

Collaborators:

- External Firebase Database

Class Name: **FinancialHubActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Graph monthly transactions.
- Provide an interface to navigate between the financial managing functions (“set financial goals”, “track expenditures”, “load transactions”, “set savings goals”, “wish list”, “log purchases”)
- Link to the main menu

Collaborators:

- MainActivity
- ExpenditureChart
- FinancialGoalsActivity, FinancialManagerActivity, ImportTransactionsActivity, SavingsGoalsActivity, WishListActivity, LogPurchasesActivity (need to collaborate when transitioning between activities; when instantiating new Intent objects).

Class Name: **FinancialGoalsActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Have ability to set monthly spending goals.
- Have ability to set yearly spending goals.
- Send notifications when spending thresholds are met
- Set spending goals by transaction category.
- Set spending goals by savings targets.
- Link to the financial hub

Collaborators:

- FinancialModel
- FinancialHubActivity

Class Name: **FinancialManagerActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Provide graphical representation of:
    - Yearly expenditure (line chart)
    - Monthly expenditure (line chart)
    - Expenditure by category (pie chart)
    - Projected savings.
- Link to the financial hub

Collaborators:

- ExpenditureChart
- FinancialModel
- FinancialHubActivity

Class Name: **ImportTransactionsActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Automate the upload of banking transaction:
    - Parse monthly bank .csv, .tsv transaction files.
- Provide interface for manual input of transactions
- Notify the user of each successful commit to the database.
- Link to the financial hub

Collaborators:

- FinancialModel
- DatabaseHandler
- FinancialHubActivity

Class Name: **FinancialModel**
Responsibilities:

- Store yearly expenditure numbers.
- Store specific monthly transactions.
- Provide fast convenience methods for extracting data based on an query.

Collaborators:

- DatabaseHandler

Class Name: **ExpenditureChart**
Parent Class: Fragment
Responsibilities:

- Graph monthly expenditure data
- Graph monthly projected expenditure based on yearly spending goals

Collaborators:

- FinancialModel

Class name: **SavingsGoalsActivity**
Parent Class: AppCompatActivity
Responsibilities:

- Add savings goals
- Remove savings goals
- Show savings for each goal as a progress bar
- Link to the financial hub

Collaborators:

- SavingsGoalsModel
- FinancialHubActivity

Class name: **SavingsGoalsModel**
Responsibilities:

- Add savings goals
- Remove savings goals
- Get savings goals
- Save and alter savings goals data from a Firebase database

Collaborators:

- DatabaseHandler

Class name: **WishListActivity**
Parent Class: AppCompatActivity
Responsibilities:

- Add new items to wishlist
- Remove items from wish list
- Show wishlist items on screen
- Link to the financial hub

Collaborators:

- WishListModel
- FinancialHubActivity

Class name: **WishListModel**
Responsibilities:

- Add new items to wish list
- Remove items from wish list
- Get wish list items
- Save and alter wish list data from a Firebase database

Collaborators:

- DatabaseHandler

Class name: **LogPurchasesActivity**
Parent Class: AppCompatActivity
Responsibilities:

- Add new items to purchase list
- Remove items from purchase list
- Show purchased items on screen
- Link to the financial hub

Collaborators:

- PurchasesModel
- FinancialHubActivity

Class name: **PurchasesModel**
Responsibilities:

- Add new items to purchases
- Remove items from purchases list
- Get purchased items
- Save and alter purchase data from a Firebase database

Collaborators:

- DatabaseHandler

Class name: **HealthFitnessHubActivity**
Responsibilities:

- Provide links to the 4 health and fitness tools: the body weight logger, the calorie logger, the body fat calculator and logger, and finally the workout logger.
- Link to the main menu

Collaborators:

- LogBodyWeightsActivity
- LogCaloriesActivity
- LogBodyFatPercentagesActivity
- LogWorkoutsActivity
- MainActivity

Log Body Weight Activity:

Class name: **LogBodyWeightsActivity**
Parent class: AppCompatActivity
Responsibilities:

- Know today's date
- Know the selected date
- Display calendar on screen
- Display selected date in text form on screen
- Display weight for the selected date
- Create and display a graph of change in body weights over time for the month of the selected date
- Add new weight for a selected date if no weight recorded
- Remove weight for a selected date
- Link to BodyWeightsGraphActivity screen
- Link to the health and fitness hub

Collaborators:

- BodyWeightsModel
- BodyWeightGraph
- BodyWeightsGraphActivity
- HealthFitnessHubActivity

Class name: **BodyWeightsModel**
Responsibilities:

- Add new weight for a selected date if no weight recorded
- Remove weight for a selected date
- Get a weight for a selected date if it exists
- Get recorded weights for a selected month and year
- Save and alter body weight data from a Firebase database

Collaborators:

- DatabaseHandler

Class name: **BodyWeightGraph**
Parent Class: LineChart from com.github.mikephil.charting.charts.LineChart Graph API
Responsibilities:

- Create a graph of change in body weight over time for a given month of a specific year
- Create a graph of change in body weight over time for a given year

Collaborators:

- BodyWeightsModel

Class name: **BodyWeightGraphsActivity**
Parent class: AppCompatActivity
Responsibilities:

- Create and display a graph of change in body weight over time for a given month of a specific year
- Create and display a graph of change in body weight over time for a given year
- Link to LogBodyWeightsActivity screen

Collaborators:

- BodyWeightsModel
- BodyWeightGraph
- LogBodyWeightsActivity

Log Calories Activity:

Class name: **LogCaloriesActivity**
Parent class: AppCompatActivity
Responsibilities:

- Know today's date
- Know the selected date
- Display calendar on screen
- Display selected date in text form on screen
- Display food and calorie count for the selected date
- Add new food with calorie count
- Remove entry for food with calorie count
- Show a total for caloric intake for the selected date
- Link to the health and fitness hub

Collaborators:

- CaloriesModel
- CaloricIntakeCalculator
- HealthFitnessHubActivity

Class name: **CaloriesModel**
Responsibilities:

- Add new food with calorie count for a selected date
- Remove food with calorie count for a selected date
- Get food and calorie counts for a selected date
- Save and alter food and calorie data from a Firebase database

Collaborators:

- DatabaseHandler

Class name: **CaloricIntakeCalculator**
Responsibilities:

- Calculate and get the total calories eaten for the day

Collaborators:

- CaloriesModel

Body Fat Percentage Activity:

Class name: **LogBodyFatPercentagesActivity**
Parent class: AppCompatActivity
Responsibilities:

- Know today's date
- Know the selected date
- Display calendar on screen
- Display selected date in text form on screen
- Display body fat percentage for the selected date
- Create and display a graph of change in body weights over time for the month of the selected date
- Add new body fat percentage for a selected date if no entry
- Remove entry for bodyfat percentage if one exists
- Link to BodyFatGraphsActivity screen
- Link to the health and fitness hub

Collaborators:

- BodyFatModel
- BodyFatCalculator
- BodyFatGraph
- BodyFatGraphsActivity
- HealthFitnessHubActivity

Class name: **BodyFatModel**
Responsibilities:

- Add new body fat percentage for a selected date
- Remove body fat percentage for a selected date
- Get body fat percentage for a selected date
- Save and alter body fat percentage data from a Firebase database

Collaborators:

- DatabaseHandler

Class name: **BodyFatCalculator**
Responsibilities:

- Calculate and get the body fat percentage

Collaborators:

- BodyFatModel

Class name: **BodyFatGraph**
Parent Class: LineChart from com.github.mikephil.charting.charts.LineChart Graph API
Responsibilities:

- Create a graph of change in body fat over time for a given month of a specific year
- Create a graph of change in body fat over time for a given year

Collaborators:

- BodyFatModel

Class name: **BodyFatGraphsActivity**
Parent class: AppCompatActivity
Responsibilities:

- Create and display a graph of change in body fat over time for a given month of a specific year
- Create and display a graph of change in body fat over time for a given year
- Link to LogBodyFatPercentagesActivity screen

Collaborators:

- BodyFatModel
- BodyFatGraph
- LogBodyFatPercentagesActivity

Log Workouts Activity:

Class name: **LogWorkoutsActivity**
Parent class: AppCompatActivity
Responsibilities:

- Know today's date
- Know the selected date
- Display calendar on screen and selected date in text form on screen
- Display exercises, sets, and reps which define a workout for the selected date
- Add a new exercise with sets and reps for a selected date
- Remove exercise entry for a selected date
- Link to the health and fitness hub

Collaborators:

- WorkoutsModel
- HealthFitnessHubActivity

Class name: **WorkoutsModel**
Responsibilities:

- Add new exercise with sets and reps for a selected date
- Remove exercise for a selected date
- Get exercises, sets, and reps for a selected date
- Save and alter workout data from a Firebase database

Collaborators:

- DatabaseHandler

Class Name: **MainActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Link to the Settings Page
- Link to the 3 planners (health and fitness planner, financial planner, and calendar events planner)
- Link to the Summary page

Collaborators:

- CalendarEventsActivity
- HealthFitnessHubActivity
- FinancialHubActivity
- SettingsActivity
- SummaryActivity

Class Name: **SettingsActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Link to the profile page, change password page, and include a logout button.
- Can load a picture at the top of the screen as the button/link to the profile page.
- Link back to the main page

Collaborators:

- ProfileActivity
- ChangePasswordActivity
- MainActivity
- LoginActivity (link to here when logging out)

Class Name: **ProfileActivity**
Parent Class: AppCompatActivity
Responsibilities:

- Allow the user to see their profile data
- Allow the user to change their profile data
- Link to the settings page

Collaborators:

- DatabaseHandler
- SettingsActivity

Class Name: **ChangePasswordActivity**
Parent Class: AppCompactActivity
Responsibilities:

- Link to the settings page.
- Can get access to the old password of the recent account and update to the new password

Collaborators:

- SettingsActivity

Class Name: **Feedback**
Parent Class: AppCompactActivity
Classname Subclasses: N/A
Responsibilities:

- Link to the setting page.
- Can get feedback from user

Collaborators:

- DatabaseHandler
- SettingsActivity

Class Name: **SummaryActivity**
Parent class: AppCompatActivity
Responsibilities:

- Summarize data for the past month from various planners: the health and fitness planner, the financial planner, and the calendar
- Link to the main menu

Collaborators:

- MainActivity
- CalendarEventsModel
- FinancialModel
- ExpenditureChart
- SavingsGoalsModel
- WishListModel
- Purchases Model
- BodyWeightGraph
- BodyWeightsModel
- CaloriesModel
- CaloricIntakeCalculator
- BodyFatModel
- BodyFatGraph
- WorkoutsModel

System Interaction

Here we outline how our application interacts with external API's used in the development, as well as which operating systems that it is designed to interface with and finally its proposed interactions with external services and dependencies (such as application servers and/or databases).

- External API's and Libraries
    - Gradle: as a result of the choice to use Android Studio as the Software Development Kit for the application. Gradle helps build our project and manage easily automatable tasks such as compilation.
    - MPAndroidChart: a free, open-source, native Android API for generating aesthetic charts and graphs, mostly to be used by the financial planning tool and the fitness planning tool.
- Operating Systems: this Android application is proposed to run on early Android operating systems. Specifically, the application supports operating systems providing the Android Icecream Sandwich - API 14 (released on October 18, 2011) and is compatible with the newest Android softwares/ APIs. This design decision was made to reach the most users with Android phones as possible (while also considering the tradeoffs with using older Android software API's).
- External Interactions and Services
    - Firebase: the key-value based server that we plan to integrate with this application as to support persistent storage of user data rather than relying on local storage (local serialization).
    - Notification System: the application uses the Android operating system's notification system to alert users of specific events.
    - Internet: since this application uses a database, it needs to communicate constantly with the Firebase database over an internet connection. Therefore, we the developers of the app require and assume users have a connection to the internet when using the application.

System Architecture

- Our application will follow the 3-Tiered Architecture: XML files will present the view as part of the Presentation Layer, Java code files will be the bridge between the view and the database as part of the Application Layer, and a Firebase database will store user data as part of the Data Layer.
- The Application Layer will follow a mock MVC structure by including an activity class which inherits from AppCompatActivity (Android's class for making screens on a display screen which are compatible with older Android APIs) which will act as the controller: one controller per activity. There will also be a model class for most activities: one model per activity. If the XML files which are rendered may be considered the View in MVC, then our Application Layer follows the MVC structure. But otherwise, our application has a 3-Tiered Architecture with an Application Layer that follows loosely the MVC architecture.
- In general, from a high level perspective, model classes from the Application layer will connect to the Data Layer to retrieve and store data and the Activity classes which inherit from AppCompatActivity will connect to the Presentation Layer which consists of the XML files that render the screen views.
- Errors and exceptions such as invalid input, network failures, and general system failures will be handled in various ways. For invalid input and network failures, a message will pop up notifying the user of the correct input to use if the input was incorrect and a "Network Unavailable" message or similar in case of a network failure. As for system failures, Android automatically tells the user that the application "has stopped" if there is an error loading activities or other data in the program. However, we will also include a message indicating that the application did not shut down properly if the application shuts down due to a system error.
- System Decomposition and Component Connection to Larger Architectural Design:
- The Model Classes connect to the Data Layer via the DatabaseHandler
    - [**CalendarEventsModel, FinancialModel, WishListModel, SavingsGoalsModel, PurchasesModel, BodyWeightsModel, CaloriesModel, BodyFatModel, WorkoutsModel**] -connect to-> **DatabaseHandler** -connect to-> Data Layer (Firebase database) to store and retrieve data from the database
- Various Activity Screens link to each other so that the user can access the screens
    - **LoginActivity** <-connects to-> [**Signup**] to link to each other
    - **LoginActivity** -connects to-> [**MainActivity**] to link to each other
    - **MainActivity** <-connects to->[**CalendarEventsActivity, FinancialHubActivity, HealthFitnessHubActivity, SettingsActivity, SummaryActivity**] to link to each other

    - [**SettingsActivity**]<-connects to->[**ProfileActivity, ChangePasswordActivity, Feedback, MainActivity**] to link to each other
    - [**FinancialHubActivity**]<-connects to->[**FinancialGoalsActivity, FinancialManagerActivity, ImportTransactionsActivity, SavingsGoalsActivity, WishListActivity, LogPurchasesActivity**] to link to each other
    - [**HealthFitnessHubActivity**]-connects to->[**LogBodyWeightsActivity, LogCaloriesActivity, LogBodyFatPercentagesActivity, LogWorkoutsActivity**]
    - [**LogBodyWeightsActivity**]-connects to->[**BodyWeightGraphsActivity**] to link to each other
    - [**LogBodyFatPercentagesActivity**]-connects to->[**BodyFatGraphsActivity**] to link to each other
- The Activity Classes connect to various xml files (the view and presentation layer) as well as the Model Classes which provide an interface to access activity specific data from the Data Layer
    - [**CalendarEventsActivity**]-connect to->[**CalendarEventsModel**] to retrieve data from the database via the model interface
    - [**FinancialGoalsActivity**]-connect to->[**FinancialModel**] to retrieve data from the database via the model interface
    - [**FinancialManagerActivity**]-connect to->[**FinancialModel**] to retrieve data from the database via the model interface
    - [**ImportTransactionsActivity**]-connect to->[**FinancialModel**] to retrieve data from the database via the model interface
    - [**SavingsGoalsActivity**]-connect to->[**SavingsGoalsModel**] to retrieve data from the database via the model interface
    - [**WishListActivity**]-connect to->[**WishListModel**] to retrieve data from the database via the model interface
    - [**LogPurchasesActivity**]-connect to->[**PurchasesModel**] to retrieve data from the database via the model interface
    - [**LogBodyWeightsActivity**]-connect to->[**BodyWeightsModel**] to retrieve data from the database via the model interface
    - [**LogCaloriesActivity**]-connect to->[**CaloriesModel**] to retrieve data from the database via the model interface
    - [**LogBodyFatPercentagesActivity**]-connect to->[**BodyFatModel**] to retrieve data from the database via the model interface
    - [**LogWorkoutsActivity**]-connect to->[**WorkoutsModel**] to retrieve data from the database via the model interface

As for the other classes: **EventNotifier, ExpenditureChart, BodyWeightGraph, CaloricIntakeCalculator, BodyFatCalculator, and BodyFatGraph,** some of these classes interact with the corresponding model (for example, **BodyWeightGraph** depends on **BodyWeightsModel**) to retrieve appropriate data, and some others perform class specific functions such as calculating the body fat percentage based on an input: what **BodyFatCalculator** does. These are usually intermediary classes that interact between the corresponding model and activity classes.