

CS683 Project Assignment

FitnessTrack

Haohan Guo

Yi-Yang Lin

Overview	2
Related Work	2
Requirement Analysis and Testing	2
Design and Implementation	3
Project Structure	4
Timeline	4
Future Work (Optional)	4
Project Demo Links	4

1. Overview

FitnessTrack is a health record listing Android application, allows users to record food items in each meal and define the health level of each meal.

Our motivation for this App was when we workout, we want to have an app to record our food for each meal. Since there are several workout foods information that we would like to record all at once, we can use this app to record the date and healthy level about the food ingredients. In short, we made this app to manage our diet in terms of our personal workout exercise.

2. Related Work

A product with similar features on the market is a simple version of Google Fit: Activity Tracking, which has two main functions, "Journal" and "Profile" tabs. "Journal", for example, shows and tracks the route of running location information. The "Profile" will be able to update the user's information.

Some apps that are similar in the market -
Omo -

Lose weight and get fit with Omo, a fitness app for weight loss! It's like having a fitness coach, meal planner, calorie counter, and fasting tracker with you at all times. Discover healthy weight loss with Omo.

Strong Workout Tracker Gym Log -

Strong is the simplest and most intuitive workout tracker, designed to help you get better results from your workouts. Whether you want to gain strength or just stay healthy, join over 1.2 million people who have downloaded Strong to stay on track in the gym.

Keep -

The app allows users to view fitness videos and to buy fitness equipment. It contains a social networking service so that customers can share exercise routines with each other.

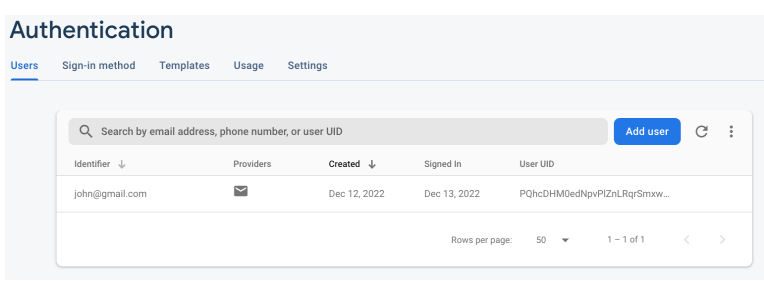
Difference -

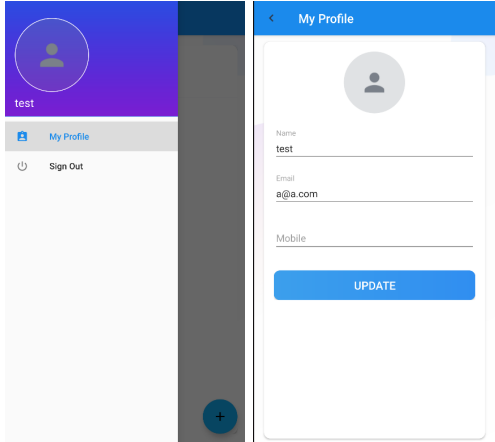
FitnessTrack's main function is to record the healthiness of each meal and categorize them for easy review by the user.

FitnessTrack offers a high degree of customization. Users can define what time they eat. For the meal log function, It also provides eye-catching color markings, so users will know if the meal they eat is healthy or not.

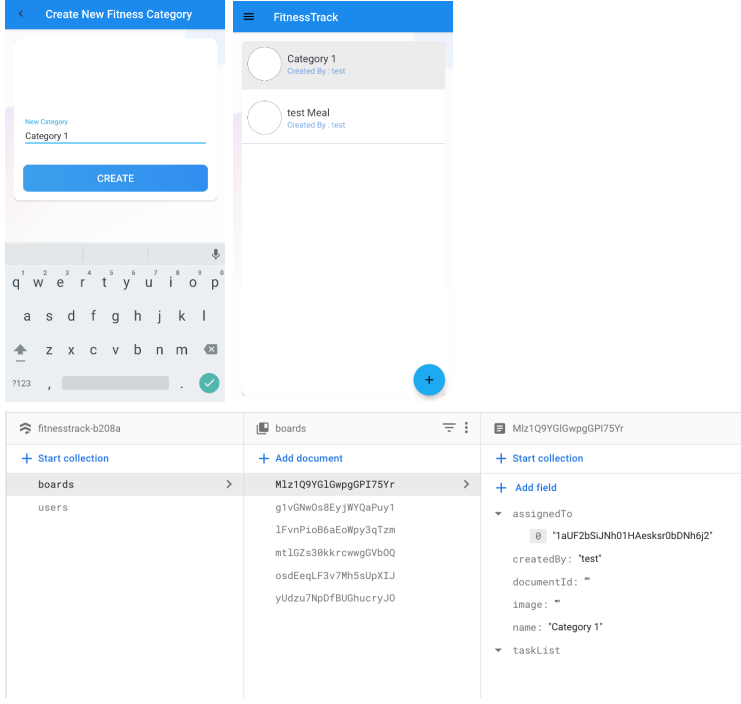
3. Requirement Analysis and Testing

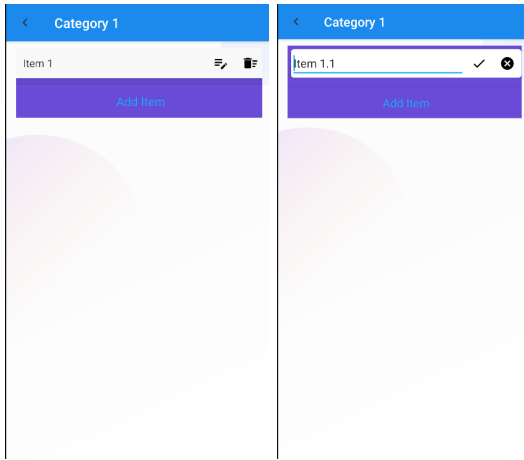
<i>Title</i>	<i>Login or signup user account (essential)</i>
<i>Description</i>	<i>Users are able to create their own account and store personal information on cloud Login page with queries: 1. Username 2. Password 3. Login button</i>
<i>Mockups</i>	<i>Checking if one user logs in, it will correctly gain the account corresponding data in Firebase.</i>
<i>Acceptance tests</i>	<i>The frontend account should match the same account in backend data</i>
<i>Test Results</i>	<i>If the user submitted username and password correspond to the data in the back-end part of the database.</i>

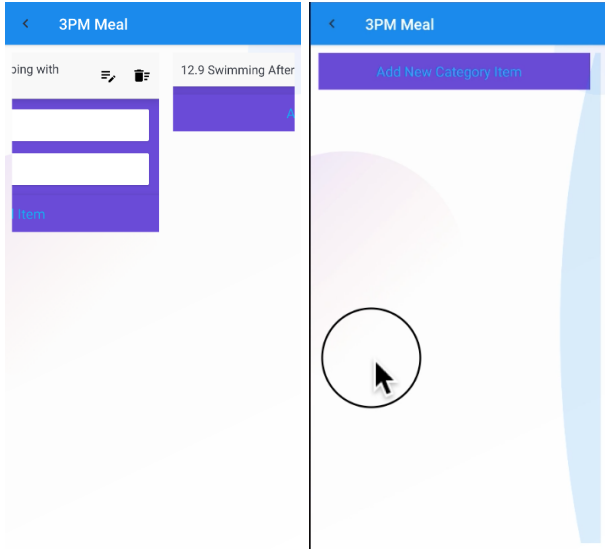
	
Status	<i>If a user registered an existing email account, it will show the “Registration Failed”; if a user typed not existing user account will login, it will show “Authentication failed” correctly.</i>

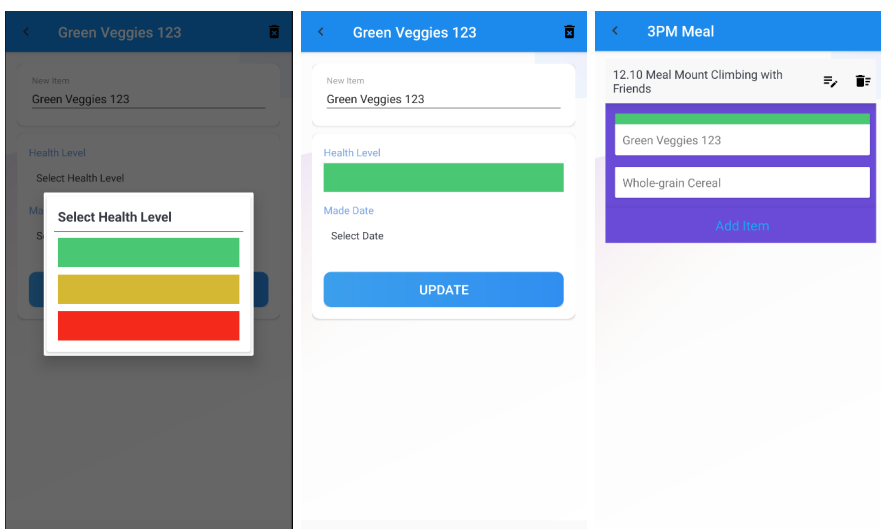
Title (Essential/Desirable/Optional)	<i>Switching views by tapping the My Profile view tab, FitnessTrack view, analysis view button (essential)</i>
Description	<i>At the bottom of layout of the app it appears tabs that can switching views to be redirecting to different functions</i>
Mockups	<i>The view can transition in correct order by following user’s clicks</i>
Acceptance tests	<i>User clicks on each view and the screen should show home, exercise, meal view correspondingly what user clicks</i>
Test Results	
Status	<i>Layouts of each was created correctly</i>

Title	<i>Add New Fitness Category function (Essential)</i>
Description	<i>Allows the user to enter various data about the consumption of each meal, such as calories, protein. The app will count the data and display it.</i>
Mockups	<i>Ability to correctly add the top level of the data entered by the user into the database.</i>
Acceptance tests	<i>The meal should be added after inputting data</i>

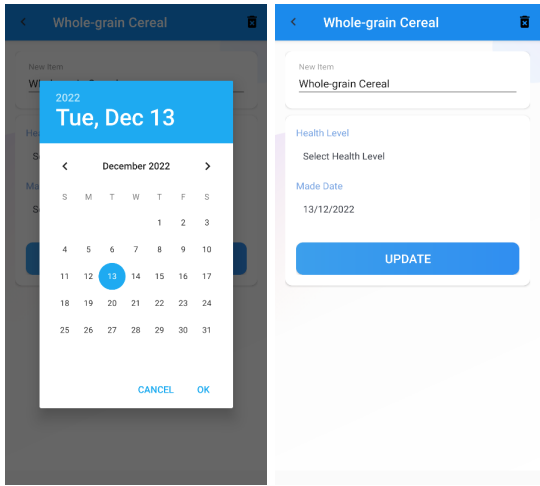
Test Results	<p>After inputting Meal, UI display the added Meal title The data should be found in cloud-based database</p> 
Status	Correctly added the New category on UI and stored on Cloud database

Title (Essential/Desirable/Optional)	Add or edit food in function under adding Meal function (Essential)
Description	A subset of adding meal, food data will be stored in cloud database and can be retrieved from API server
Mockups	The added data and retrieved data should always consistent and correctly stored
Acceptance tests	When click on the button of adding food, the data should be stored correctly
Test Results	<p>The data should be found in cloud-based database</p> 

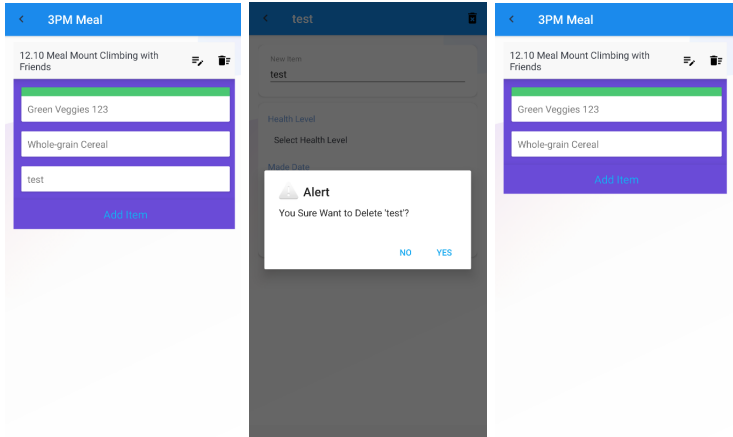
<i>Description</i>	<i>Under the same category of Meal, it should have a recycle view to list down all sub-meals</i>
<i>Mockups</i>	<i>Drop down window to enable user to select movement</i>
<i>Acceptance tests</i>	<i>The selected drop down exercise should be displayed after selected and also updated to cloud-based database</i>
<i>Test Results</i>	<p><i>The data should be found in cloud-based database</i></p> 
<i>Status</i>	<i>The sub-meals are listed and draggable.</i>

<i>Title</i>	<i>Popup meal view and healthy level (Essential)</i>
<i>Description</i>	<i>Allows users to set their own customized ingredient with the healthy level</i>
<i>Mockups</i>	<i>The clicked popup meal view should be correct and show the right data and editable</i>
<i>Acceptance tests</i>	<i>When click on the button of adding exercise, the data should be stored correctly</i>
<i>Test Results</i>	

<i>Status</i>	<i>The color is correctly updatable.</i>
---------------	--

<i>Title</i>	<i>Popup meal view and ingredient made date (Essential)</i>
<i>Description</i>	<i>Allows users to set their own customized ingredient with the ingredient made date</i>
<i>Mockups</i>	<i>The clicked popup meal view should be correct and show the right data and editable</i>
<i>Acceptance tests</i>	<i>The added food items should under meal view page</i>
<i>Test Results</i>	
<i>Status</i>	<i>The date is correctly updatable.</i>

<i>Title</i>	<i>Popup meal view can be deleted (Essential)</i>
<i>Description</i>	<i>In the popup meal view, user can click the button of top-right and get the current fitness results from the cloud base API database server</i>
<i>Mockups</i>	<i>A series of clicking actions</i>
<i>Acceptance tests</i>	<i>The clicked popup meal view should be correctly deleted.</i>

Test Results	
Status	<i>The ingredient in the meal can be deleted correctly</i>

4. System Requirements

Fitness requires a modern Android Phone (we will be testing primarily on Macbook based Android studio - *Intel* chip and *M1* chip with internet access.

- Back-end: Google Firebase
 - Authorization: Firebase Authorization
 - Database: Cloud Firestore
 - Image Database: Cloud Storage
- Front-end: Kotlin and Android's modern toolkit
- Host: Firebase (back-end + front-end)
- Development tools: *Android Studio*, *Google* accounts

5. Design and Implementation

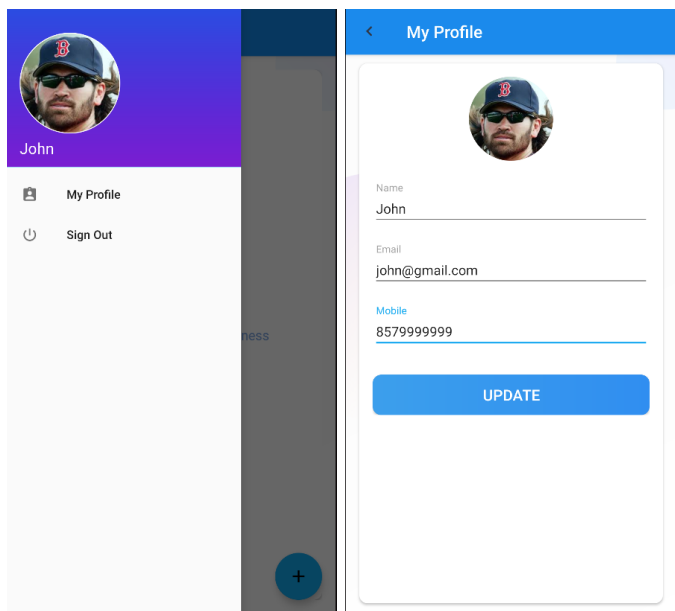
- *Basic architecture*
The project will use the MVC architecture, which is a method of organizing the code by separating business logic, data, and interface display, gathering business logic into a single component, and improving and personalizing the interface and user interaction without rewriting the business logic. The view layer is separated from the business layer, which allows changes to the view layer code without recompiling the model and controller code. Similarly, changes to an application's business processes or business rules only require changes to the model layer of MVC. Because the model is separated from the controller and view, it is easy to change the data layer and business rules of the application. For this project, the functionality to be implemented is not very complex, so the MVC model makes the business logic all separated into controllers, which is highly modular. When the business logic is changed, it is not necessary to change the view and model, but only the controller, which facilitates the development of our program.
- *UI design and implementation*

A person is running on a beach at sunset. The sky is filled with orange and yellow clouds, and the sun is low on the horizon. The person's reflection is visible in the wet sand. The text "FitnessTrack" is overlaid in white, bold, sans-serif font.

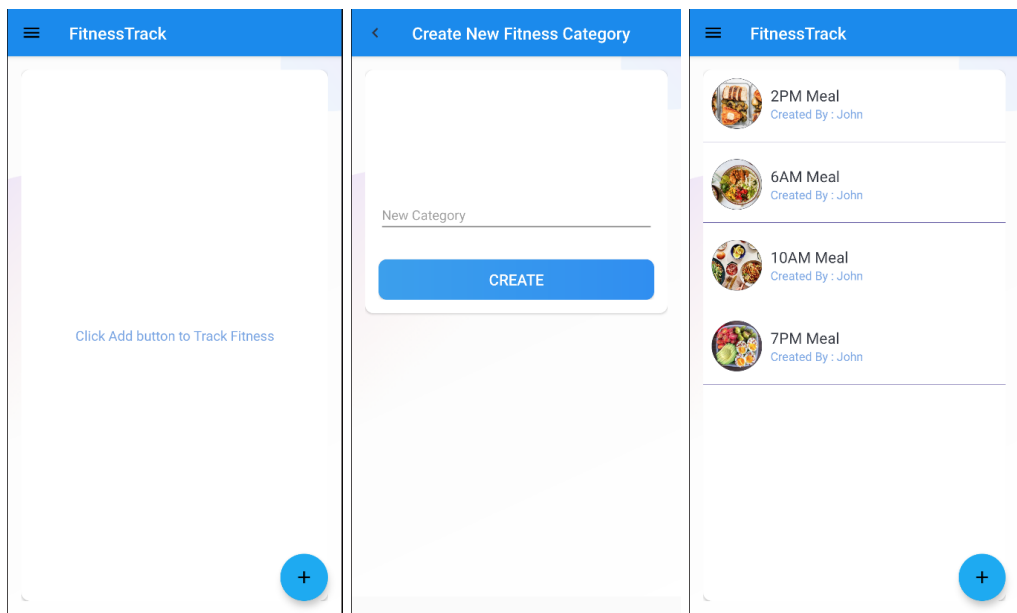
The image displays a wireframe design for a FitnessTrack app, consisting of three distinct screens arranged horizontally.

- Screen 1 (Left):** Features the "FitnessTrack" logo in blue at the top left. Below it is a photograph of a woman smiling while using a gym machine. Under the photo, the text "Let's Get started" is followed by "Nice Fitness app?". At the bottom, there are two blue buttons: "SIGN IN" and "SIGN UP".
- Screen 2 (Middle):** Titled "SIGN UP" at the top. It prompts the user with "Please Register with us." Below this, there are three input fields labeled "Name", "Email", and "Password". The "Name" field contains the text "John". The "Email" field contains "john@gmail.com". The "Password" field contains six dots. A blue "SIGN UP" button is positioned at the bottom of the form.
- Screen 3 (Right):** Titled "SIGN IN" at the top. It prompts the user with "Enter Email and Password to Login." Below this, there are two input fields labeled "Email" and "Password". The "Email" field contains "johnfail@gmail.com". The "Password" field contains six dots. A blue "SIGN IN" button is positioned at the bottom of the form. At the very bottom of this screen, there is a grey button labeled "Authentication failed."

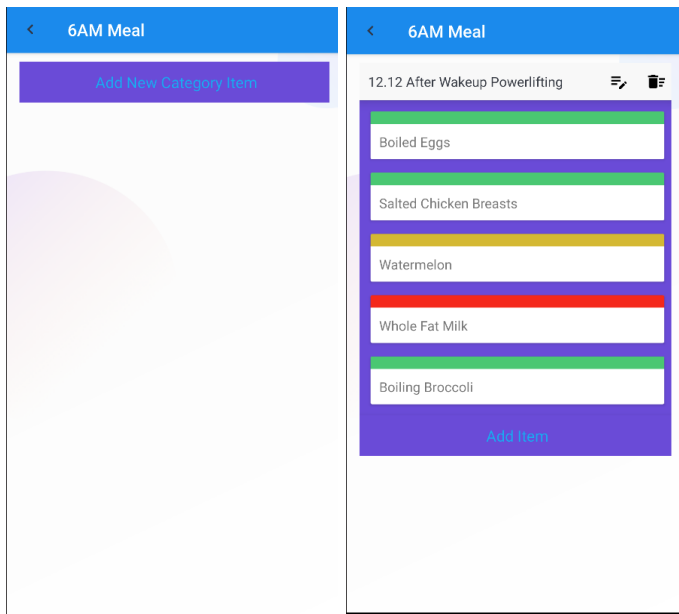
Profile



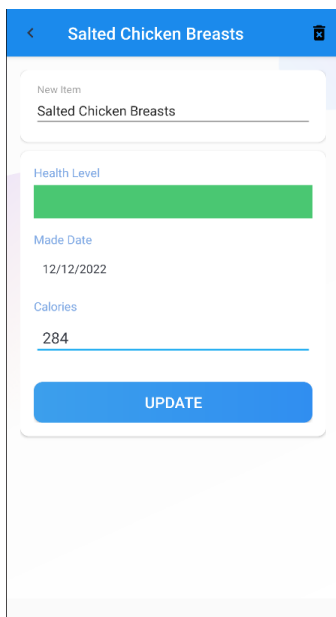
Create Board



Create TaskList/ Create Card



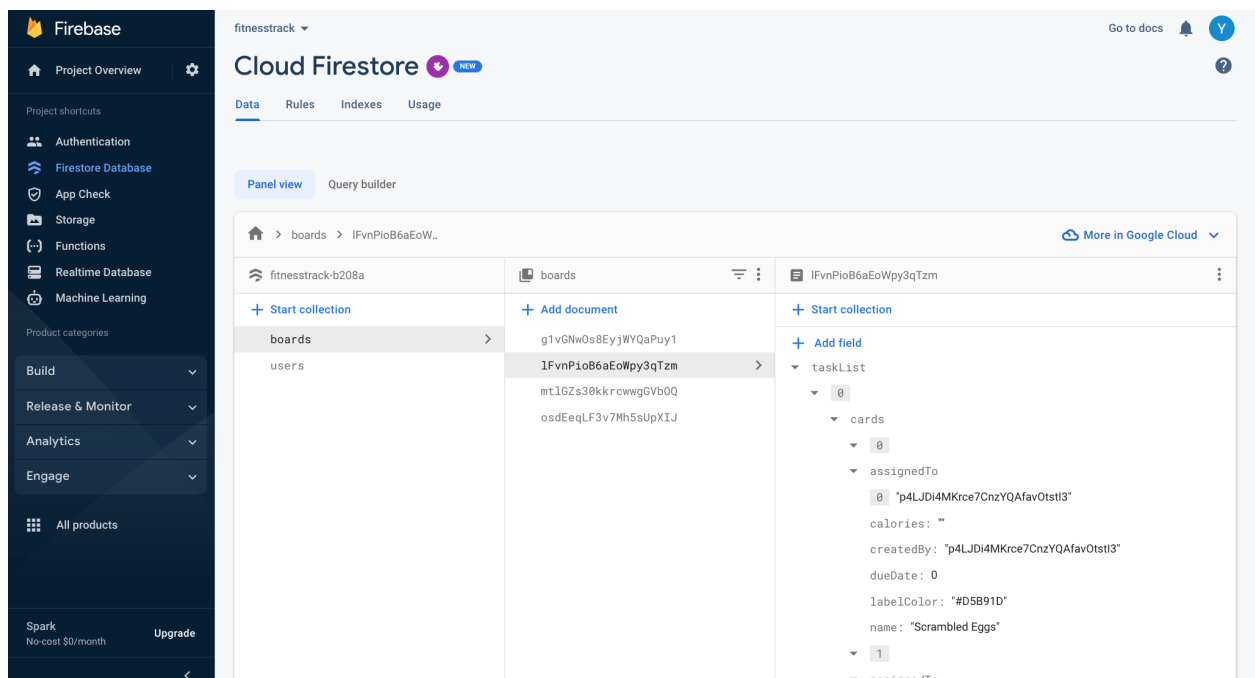
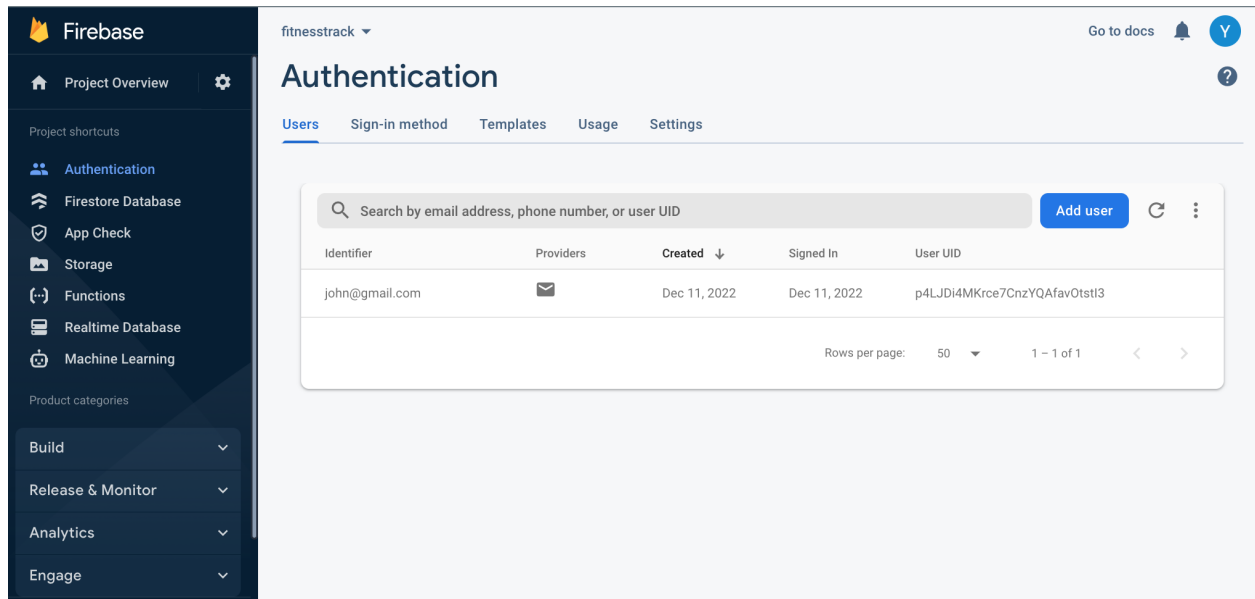
Update Card

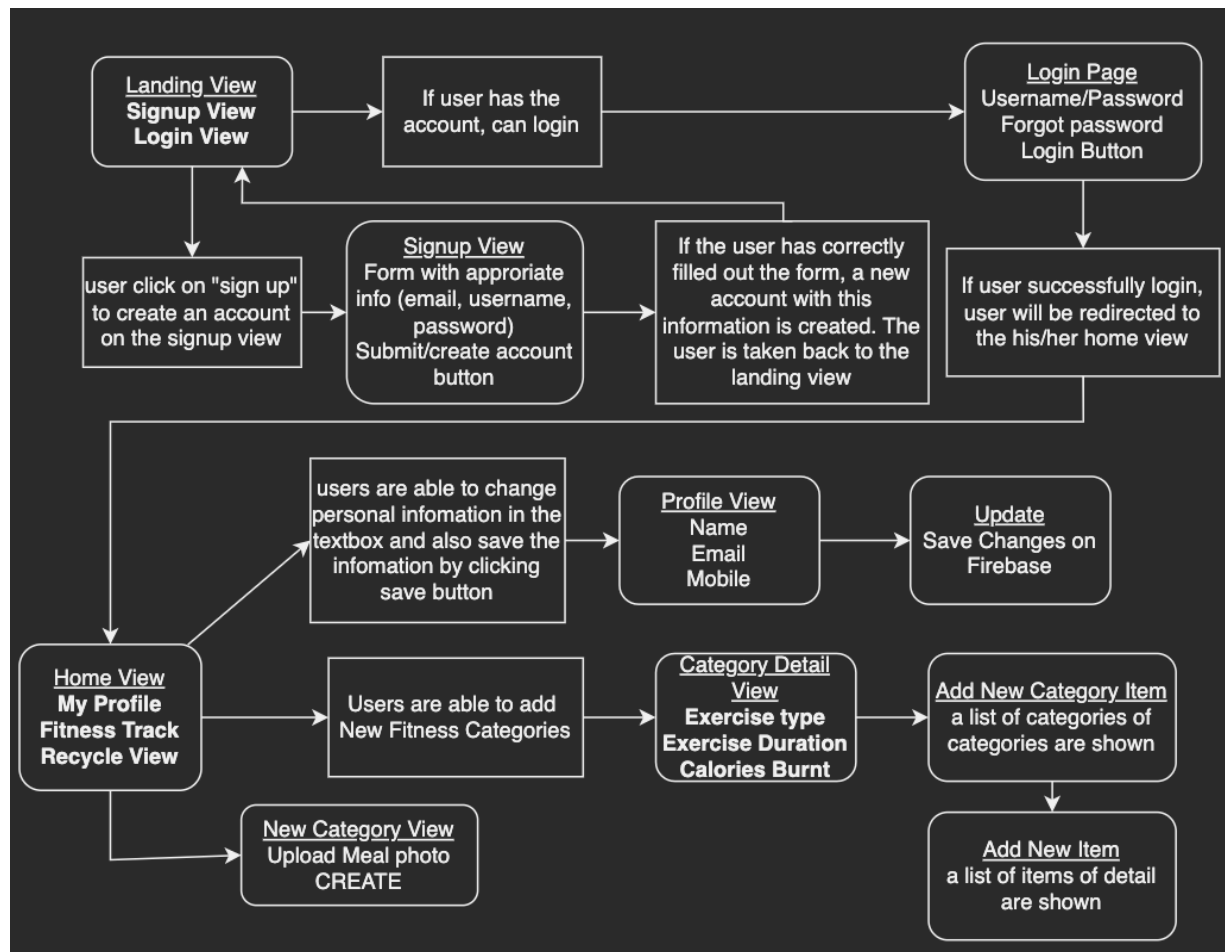


OBJ*OBJ*OBJ*OBJ*OBJ*OBJ*OBJ

- Other android features
 - Service, sensors, animations, etc (Optional)

- **Third party APIs**
 - *Glide - open source media management and image loading framework for using uploading images to Firestore*
- **Data Design and implementation**
 - *Database schema, data storage*
 - *Firebase NoSQL data structure will be utilized*
 - *Each user has each object of variables*
 - *Whenever a user uploads image file data, it stores the file in “storage” and attached url in metadata, and in the database the collection, the url can be stored and be referenced.*





6. Project Structure

- *AndroidManifest.xml*
 - An overview of the views of entire application, structures the application view flow redirecting to different views
- *CardDetailsActivity*
 - It is a view after the user clicks a card from the Board, user can input or edit item name and change the color of the health bar, select the made date of the food, and input the calories of the meal detail.
- *CreateBoardActivity*
 - It is a view after the user clicks a button from MainActivity, user can input the initial meal creating recyclerview and make a new row of category to firebase.
- *SplashActivity*
 - The first view that pop up a few seconds before the the login/signup view appear
- *IntroActivity*
 - The view that shown to let user to login or sign up by redirecting in two button "SIGN UP" and "LOGIN"
- *MainActivity*
 - A drawer function based view that user will swipe the tabs from left side of the app, showing the downloaded User Image from Firebase, Username, and 2 tabs - "My Profile", "Sign out"

- *LabelColorListDialog*
 - A pop up selecting view after the user clicks the text field of Health Level bar.
- *MyProfileActivity*
 - User can update his/her personal details in those text fields, and there is a 'UPDATE' button to save it to Firebase database
- *SignInActivity*
 - An activity to retrieve user data and validate the username and password from Firebase's authentication system
- *SignUpActivity*
 - An activity to retrieve user data and validate the username and password from Firebase's authentication system to create an account on the Firebase.
- *TaskListActivity*
 - An activity to manage the list after the list created inside the board.

- *BoardItemsAdapter*
 - Listing down a list of boards that are also clickable on each row.
- *CardListItemsAdapter*
 - Showing a list of Cards that are inside of the Board
- *LabelColorListItemsAdapter*
 - A list of Colors that are provided to select the health levels.
- *TaskListItemsAdapter*
 - A list of items that provides new item under the cards

- *Firestore*
 - Several Firebase functions that other view activities will utilize including collections with ID and storing data to Firebase server and retrieving data from Firebase
- *Board*
 - A object that bypassing from inputting and uploading to Firebase server, also able to download from NoSQL database from Firebase
- *Card*
 - A data class object that pass the Card details into each Task
- *Task*
 - A task listed in the first page
- *User*
 - A object that stores information about personal user

- *Constants*
 - General helper constants and functions

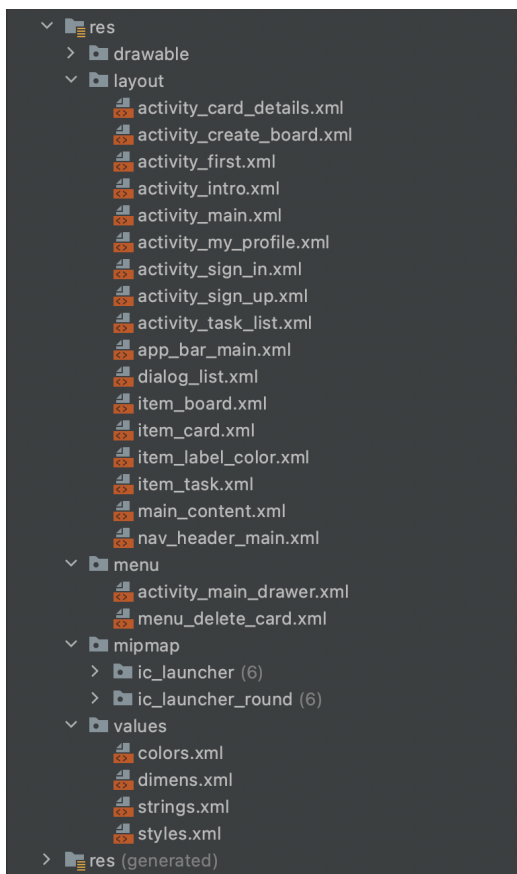
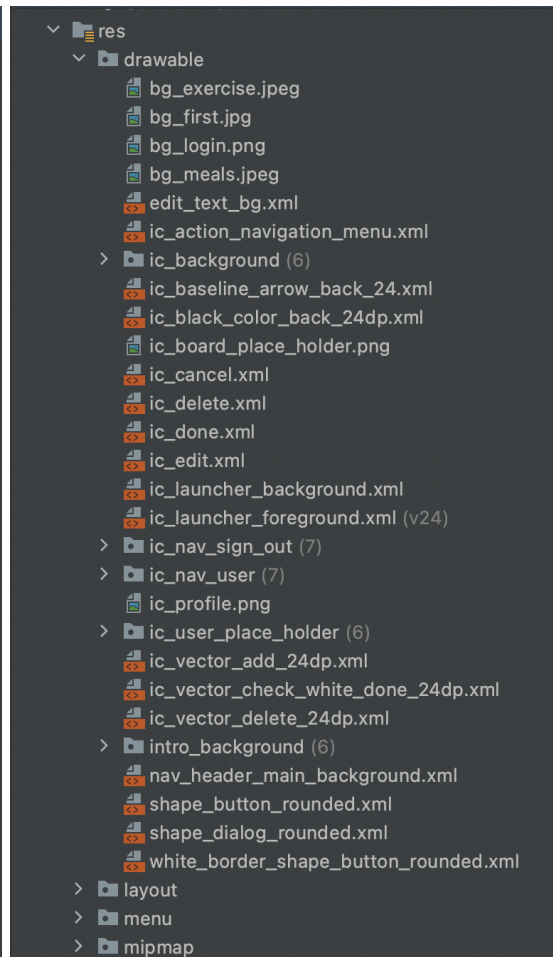
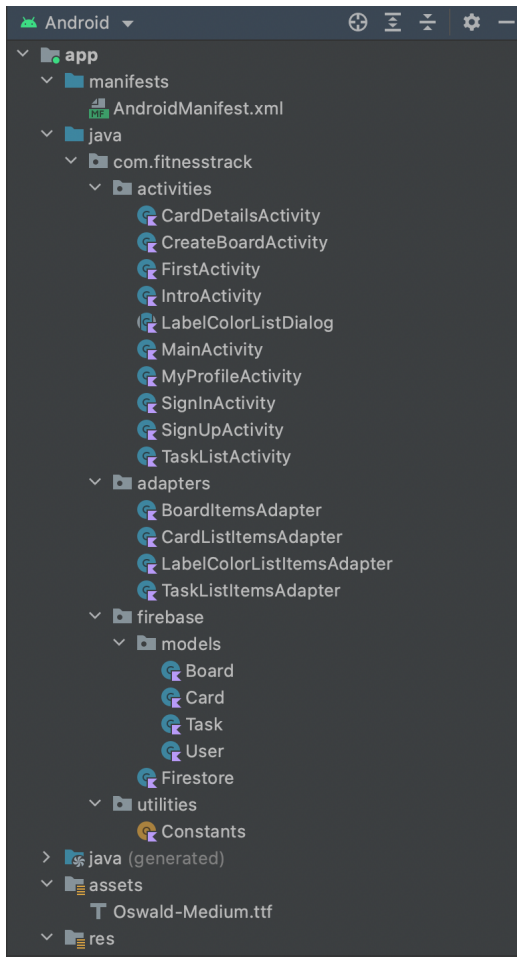
```
dependencies {

    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.2.0'
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'

    implementation platform('com.google.firebase:firebase-bom:31.0.2')
    // implementation 'com.google.firebase:firebase-auth'
    implementation 'com.google.firebase:firebase-firestore-ktx'
    implementation 'com.google.firebase:firebase-auth-ktx:19.2.0'
    implementation 'de.hdodenhof:circleimageview:3.1.0'
    implementation 'com.google.android.material:material:1.1.0'
    implementation 'com.google.firebase:firebase-storage-ktx'

    implementation 'com.github.bumptech.glide:glide:4.11.0'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.11.0'

    testImplementation 'junit:junit:4.13'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
}
```



7. Timeline

Iteration	Application Requirements (Essential/Desirable/O ptional)	Android Components and Features to be used	Haohan contribution/ planned tasks	Yiyang contribution/ planned tasks
0	Listed app overview and potential features Functions are listed and defined requirements Listed app overview and potential features Functions are listed and defined requirements	none	Communicate and discuss to determine the topic, write iteration 0, and summarize	Communicate and discuss to determine the topic, write iteration 0, and summarize
1	Complete the essential part of the function, including user login, fitness detail section, food detail section, analysis section	We defined activity_main and four main fragment.xml files in which Spinner, Button, EditText, TextView are used	1. Completed the registration interface and login interface of the application 2. Completed the fragment.xml file for the two main functions of the project, meal and exercise 3. used navigation to create the bottom navigation bar of the application	1. Created Home view and final generate analysis view 2. Draw the app diagram to show each step with function details 3. Adjusted Overview description 4. Managed source code and pushed to Github
2	Implemented User click event.Improvements to the UI in Iteration0 based on subsequent requirements. Consider extending the application functionality depending on time.	We make a user account system in UI and connect to the backend Firebase system. Successfully uploading data to Firestore in NoSQL type data in MyProfileActivity	1. Set up the XML UIs and setup the view flow such as after clicking each button 2. Created Adding Exercise and Meal views letting the Exercise list and Meal list to retrieve data from Firestore	1. Checking the syncing function in the Gradles files and create Firebase account and make cloud features work 2. Created the LOGIN and SIGNUP system on Firebase and able to store validation information on Firebase Authentication and Database Server

3	<p>Created Board, TaskList, Activity,</p> <p>Prepared Recycleviews - Board, CardList, LabelColorList, TaskList adapters</p> <p>Implemented Editing and Deleting functions with firebase</p> <p>Implemented color selection, Date selections, and Calories input features</p>	<p>Special Widgets, RecyclerView, Toolbar, AppCompat, CircleView, NavigationView</p> <p>Server Side - Authentication, Firestore Database, FireStorage</p> <p>Permission INTERNET ACCESS_NETWORK_STATE READ_EXTERNAL_STORAGE</p> <p>API - Glide</p>	<p>1.Responsible for CardDetailsActivity and TaskListActivity, implementing the creation of TaskList and card, and completing the layout of the corresponding xml file.</p> <p>2.Implemented the TaskListItemsAdapter and CardListItemsAdapter, allowing users to add custom meals and add various food items to the meal. Complete the xml layout associated with them and the activity_card_details xml file.</p> <p>3.Created Card, Task models in firebase. Implemented methods related to Task and Card in the Firestore class to complete the storage and retrieval of application data in firebase.</p> <p>4.Created slides.</p>	<p>1.CreateBoardActivity for first top layer of creating New Category Meal related to activity_card_details</p> <p>2.BoardItemsAdapter for recursively get onClickable item from Main View</p> <p>3.LabelColorListItemsAdapter for selecting color in the detail of ingredient view related to items' xml</p> <p>4.Object Class that is used with Board model</p> <p>5.Functions in Firestore class used in for creating Board and passed to cloud</p>
---	--	--	---	--

8. Future Work (Optional)

Finally, the Firebase Cloud function can be utilized on Google Cloud Platform. After gathering data from the user, the cloud function can be triggered and send an analysis image to the front end.

As time permits, we may expand on the current functionality of the app. For example, we will allow users to record their own fitness time schedules and have alarms or pop-up reminders for users when it's time to get on the fitness schedule with better meal plans. We would also like to expand the functionality of the analytics interface to allow users to customize the time period for analysis, etc.

9. Project Demo Links

(For on campus students, we will have project presentations in class. For online students, you are required to submit a video of your project presentation which includes a demo of your app. You can use Kaltura to make the video and then submit it on blackboard. Please check the following link for the details of using Kaltura to make and submit videos on blackboard. You can also use other video tools and upload your video to youtube if you like:

https://onlinecampus.bu.edu/bbcswebdav/courses/00cwr_odeelements/metcs/cs_Kaltura.htm)

10. References

[Google Fit: Activity Tracking](#)

[Omo: Fitness & Weight Loss](#)

[Strong Workout Tracker Gym Log](#)

[The Complete Android 12 & Kotlin Development Masterclass](#)

[CS 193a: Android App Development Winter 2019](#)

[Dartmouth CS 65/165 Smartphone Programming Professor Andrew T. Campbell](#)

[Glide](#)

[Google Firebase](#)

[Learn Android App Development from Zero to Hero - Build 60+ Apps from scratch - Become a real developer](#)

[Android Development Fundamental \(Chinese\)](#)