

# 1D Project Report - SnapTrack Time Management App (Team 1-1)

---

SUTD 50.001 Introduction to Information Systems & Programming (2020)

Name	Student ID
Huang He (Mark)	1004561
Daniel Low Yu Hian	1004372
Chan Jun Hern, Cawin	1004487
Sim Jia Ren	1004401
Ong Zhi Yi	1004664

---

## Problem Statements

---

Working adults are finding it harder to separate between work and life. Similarly, students are finding it harder to partition time for studies and leisure. We hope to empower our users to make smarter time management choices by providing an all-encompassing, no-nonsense companion, to encourage them to compartmentalize their life and re-find their centre.

## Our Solution

---

An NFC-enabled mobile application that tracks your time spent on different activities effortlessly with a simple tap. With our intuitive data analytics and visualization, users can easily review past time investments and make future time management goals.

## Main Features

---

- General
  - ✓ User sign up and sign in
  - ✓ NFC tag tapping to start / end `Event`
- Today Page
  - ✓ Today Timeline Overview
  - ✓ User manually start / end `Event`
- Activities Page
  - ✓ User create `UserActivity`
  - ✓ User link NFC tag to `UserActivity`
- Analytics Page
  - ✓ Daily Overview for the past week
- Me Page
  - ✓ User sign out

- ☒ User reset account
- For Development Testing and Demo
  - ☒ Dev Button: Add dummy UserActivities
  - ☒ Dev Button: Add dummy tracked Events

## System Design and Architecture

---

### Key Custom Java Classes

#### User

Attributes:

- (string) authID: generated and maintained by Firebase Authentication.
- (string) userID: App generated (specifically) 16-char unique ID.
- (string) userName: user's name for display
- (string) email: maintained by Firebase Authentication
- (string) password: maintained by Firebase Authentication

#### UserActivity

`UserActivity` is the user-defined activity (e.g. "Work out", "Study", "CompStruct", "Entertainment", etc) for time tracking.

Attributes:

- (string) AID: stands for Activity ID, also App generated (specifically) 16-char.
- (string) activityName: name of the user-defined activity, must be unique.
- (int) color: color that ties to this user-defined activity for UI display.
- (string) category: the category of this user-defined activity.

#### Category

`Category` is used to group `UserActivity`.

Attributes:

- (string) name: name of the category.
- (int) color: color that ties to this category for UI display.

#### Event

`Event` is the object being created when a 'User' starts doing a `UserActivity`, either using NFC tapping or manual creation.

Attributes:

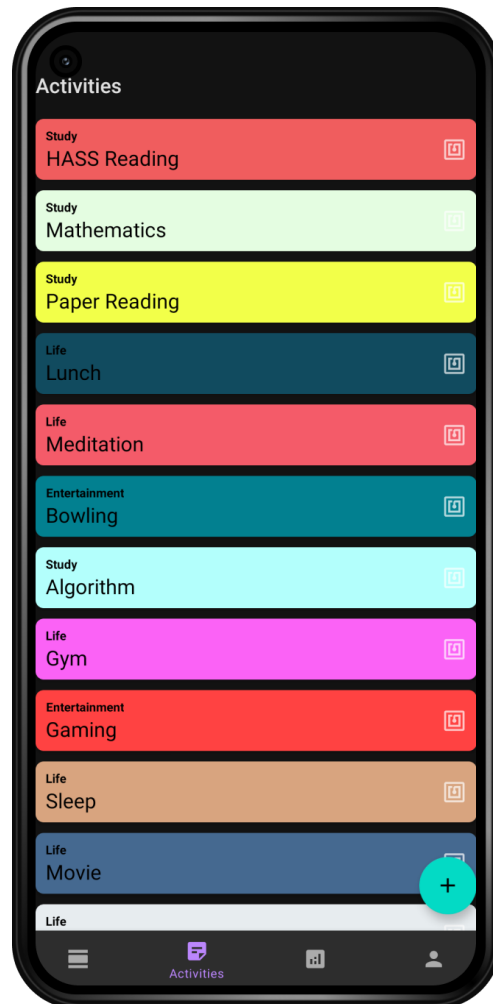
- (string) EID: stands for Event ID, randomly generated unique ID.
- (string) userID: current User's userID.
- (string) AID: current tracking `UserActivity` AID.
- (long) timeStart: start time of this event.
- (long) timeEnd: end time of this event.

# Design Patterns

## RecyclerView (Adapter Design Pattern)

*Objective:*

In our application, users can define their own list of `UserActivities`, so we need a dynamic list view to show different `UserActivities` to different users.



*Implementation Details:*

- Each `UserActivity` is presented to user as a `androidx.cardview.widget.CardView` as shown below:



- This `CardView` serves as a template for items in my `androidx.recyclerview.widget.RecyclerView`.
- Using Adapter, we can adapt an `ArrayList` of `UserActivities` into a list of `CardView` which populates the `RecyclerView`.

```
1 public class ActivitiesAdapter extends
  RecyclerView.Adapter<ActivitiesAdapter.ActivityViewHolder> {
2     ArrayList<UserActivityInfo> mActivityList;
```

```

3      // ...
4      @Override
5      public void onBindViewHolder(@NonNull ActivityViewHolder holder, int
position) {
6          UserActivityInfo currentItem = mActivityList.get(position);
7          // set name
8          holder.aTextView.setText(currentItem.getActivityName());
9          // set category
10         holder.cTextView.setText(currentItem.getCategory());
11         // set color
12         holder.mCardView.setCardBackgroundColor(currentItem.getColor());
13         // set onClick action
14         holder.mCardView.setOnClickListener(new View.OnClickListener() {
15             @Override
16             public void onClick(View v) {
17                 // ...
18             }
19         });
20     }
21 }

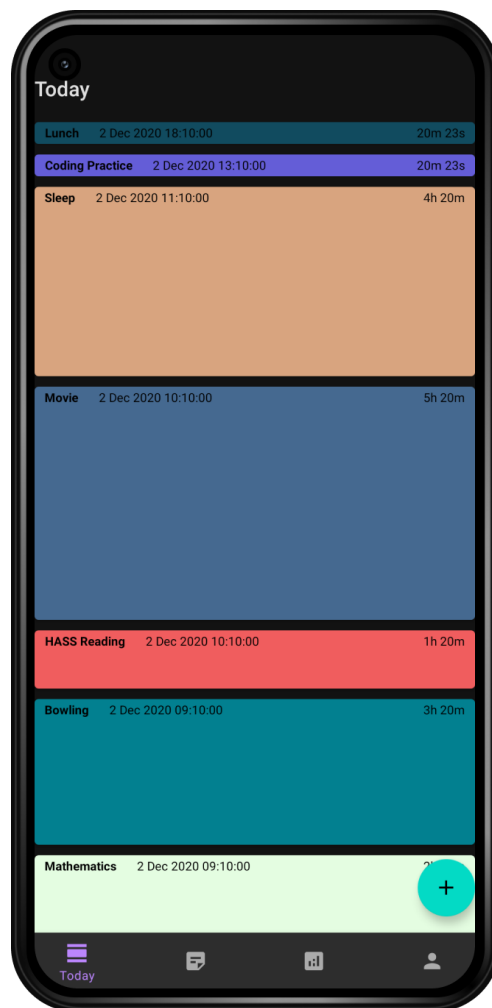
```

- Result Screenshot:

## Comparable Interface

*Objective:*

At the first page (Today page) of our application, users can view **Events** tracked for today in a timeline view, the height of each **Event** block should reflect the duration of the **Event**, and the list of events should be sorted in reverse chronological order.



#### Implementation Details:

- Firstly, the list of events cards is achieved using `RecyclerView` and `CardView` as well.
- We implemented the `Comparable` Interface for `Event` such that we can sort the list of events in an `ArrayList` before pass it to the `RecyclerView` Adapter.

```

1 public class EventInfo implements Comparable<EventInfo>{
2     // ...
3     @Override
4     public int compareTo(EventInfo o) {
5         if (this.start_time < o.start_time) return -1;
6         else if (this.start_time > o.start_time) return 1;
7         return 0;
8     }
9 }

```

```

1 ArrayList eventList = new ArrayList<>(events);
2 Collections.sort(eventList);
3 Collections.reverse(eventList);

```

- Dynamic height of `CardView` is achieved by modifying `CardView`'s `LayoutParams` based on Event duration. Minimum height is set to ensure proper display of information.

```

1 public class EventsAdapter extends
  RecyclerView.Adapter<EventsAdapter.EventViewHolder>{
2     ArrayList<EventInfo> eventList;
3     // ...

```

```

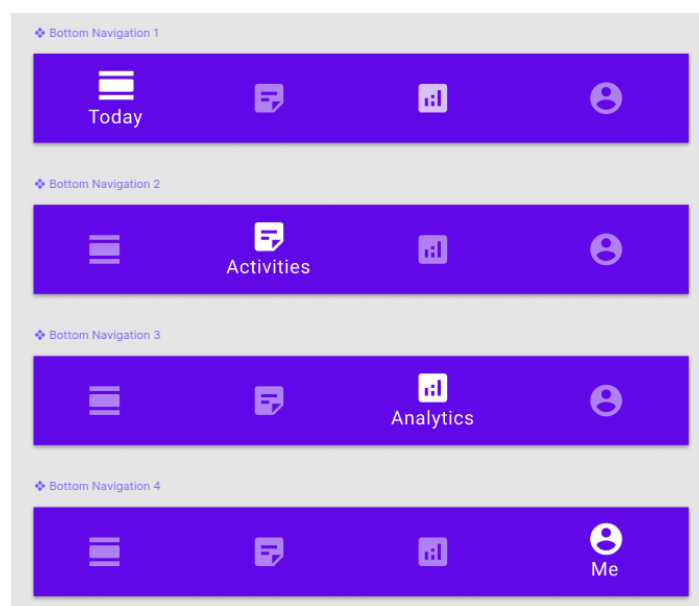
4      @Override
5      public void onBindViewHolder(@NonNull EventViewHolder holder, int
position) {
6          EventInfo currentItem = eventList.get(position);
7
8          holder.activityTextView.setText(currentItem.getUserActivityName());
9
10         holder.startTimeTextView.setText(currentItem.getStartTimeAsString());
11         holder.durationTextView.setText(
12             String.valueOf(currentItem.getDurationString())
13         );
14
15         // set card height according to Event duration
16         int heightValue = (int) currentItem.getDurationSeconds() / 36;
17         if (heightValue < 50) heightValue = 50;
18         ViewGroup.LayoutParams params =
holder.mCardView.getLayoutParams();
19         params.height = heightValue;
20         holder.mCardView.setLayoutParams(params);
21
22         // set card color according to activity color
23
24         holder.mCardView.setCardBackgroundColor(currentItem.getUserActivityColor());
25     }
26 }

```

## Fragments Navigation

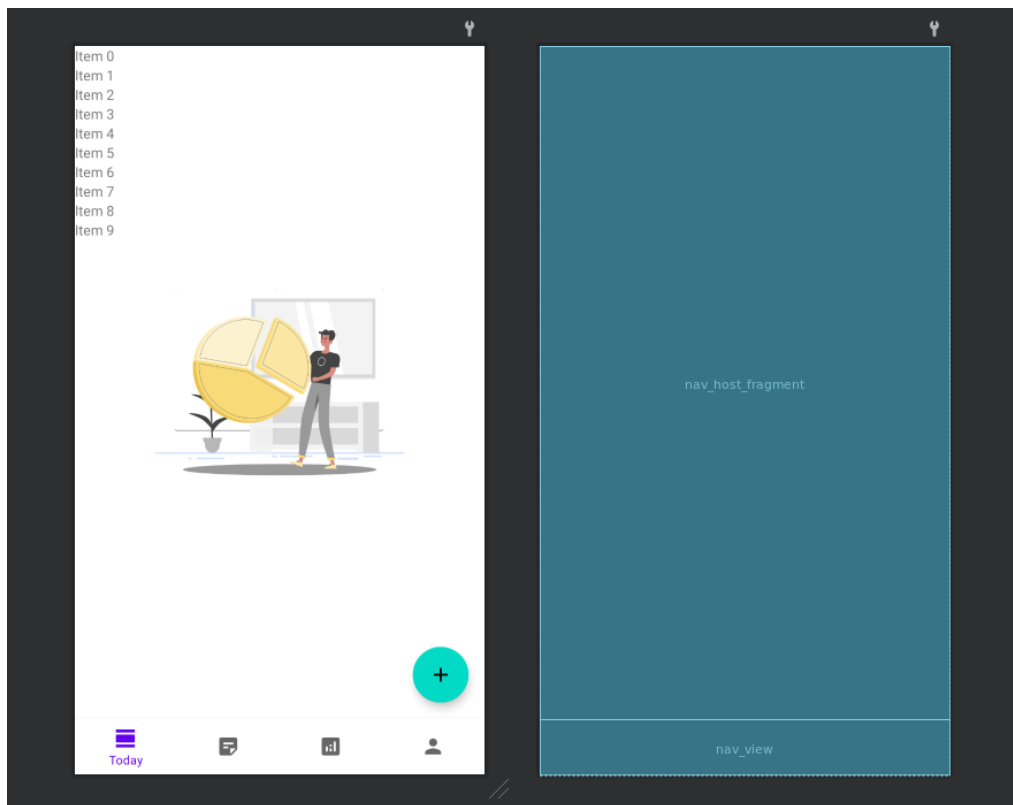
### Objective:

We have 4 top-level navigation destinations, namely **Today**, **Activities**, **Analytics**, and **Me**. According to [Material Design Principals](#), App with three to five top-level destinations should use bottom navigation bar instead of side navigation panel.



### Implementation Details:

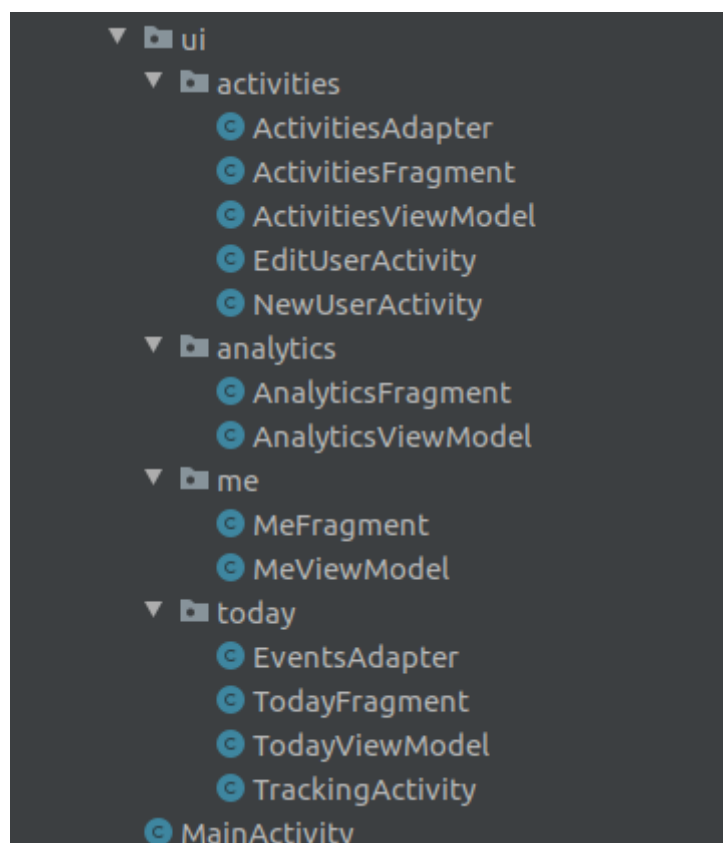
We decided to use fragments to achieve the navigation. So, at MainActivity Layout, we have a `BottomNavigationView` at the bottom to serve as the navigation bar. The rest of the screen space is filled with a `fragment` to serve as a fragment-holder / fragment-host. This fragment-host will be the parent of all fragments holding each different app pages.



## Model-View-ViewModel (MVVM) Design Pattern

*Objective:*

To separate the backend data handling and the frontend UI handling. Making the project much more structured and clean.



### Implementation Details:

- Each page has a `ViewModel` which handles data for this page. For example:

`MeViewModel.java`

```
1 public class MeViewModel extends ViewModel {
2     // ...
3 }
```

- Each page has a View Controller made of `fragment` with its corresponding `Layout` file in `XML`. For example: `MeFragment.java`

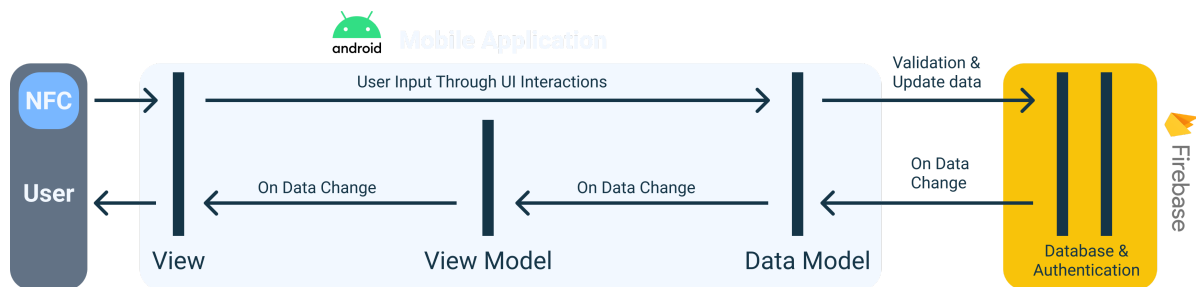
```
1 public class MeFragment extends Fragment {
2     // ...
3     @Override
4     public void onCreate(@Nullable Bundle savedInstanceState) {
5         super.onCreate(savedInstanceState);
6         // Get the View Model
7         meViewModel = new ViewModelProvider(this).get(MeViewModel.class);
8         // ...
9     }
10 }
11 }
```

## Realtime Data Binding using MutableLiveData

### Objective:

- To establish data linkage between View Controller and View Model.
- To establish realtime data synchronization between Firebase database and User-end UI.
- To optimize for performance, update UI only when data changes.

### Implementation Details:



- At View Controller, asynchronous listener is set to listen to changes on `MutableLiveData` object from `ViewModel`. For example, we use `MutableLiveData` to contain the list of Events:

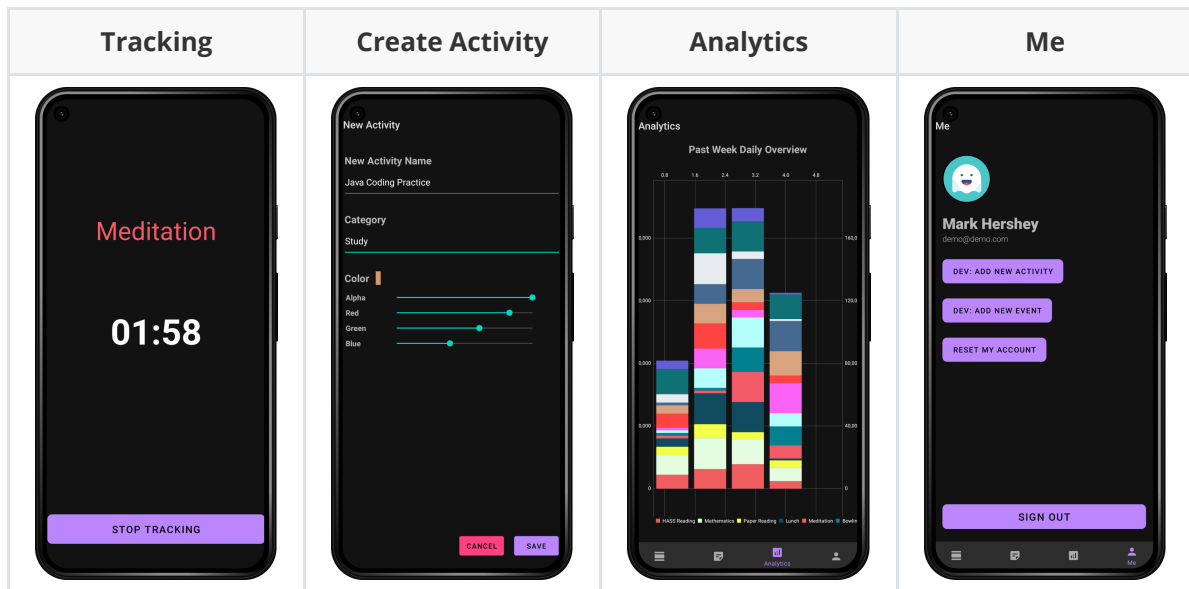
```
1 MutableLiveData<ArrayList<EventInfo>> eventListLive;
```

- At `ViewModel`, we set asynchronous database listeners.
- Data is retrieved from database using our Data Model
- Data is then put to `MutableLiveData`
- At View Controller, UI elements get updated due to `onchange` listener being triggered.

## User Flow



# App Screenshots



## User Journey

1. Create Personal Activities
2. Pair NFC to activity
  - Activities screen → Create activity → Select Activity → Pair NFC
3. Start activity tracking
  - Manual
    - Today Screen → Press Floating button → Choose created activity → Start activity tracking
  - NFC Tapping
    - Tap NFC → NFC starts timer for paired Activity
4. Stop activity tracking
  - Manual
    - Press STOP TIMER
  - Automatic
    - Scan NFC Again

## Source Code

Source code is available at [GitHub - MarkHershey / SnapTrack](#).

## Responsibilities

Name	Student ID	Responsibilities
Huang He (Mark)	1004561	Development lead, feature ideation, system architecture. Maintained the app repo and managed the inclusion of new features.
Daniel Low Yu Hian	1004372	Software Developer, Data Model Implementation, database schematic design, data utilities and more.
Chan Jun Hern, Cawin	1004487	Software Developer, Implementation of NFC functionalities (reading / writing) and more.
Sim Jia Ren	1004401	Software Developer, UI Implementations, Today Page and more.
Ong Zhi Yi	1004664	Software Developer, UI Implementations, Analytics Page and more.

## Future Work

---

- This app is applicable in many different scenarios and sectors, such as cyber-security through recording door activation timings or food delivery knowing precisely when orders are completed.
- Goals and app restriction functions would be a great function for the user to better make time management plans.

## Conclusion

---

Our project is catered to anyone that needs an easier way to keep track of time as such any professional and student can use this app as a companion to their work.