TERENCE NG HAN HOW 28867491 FIT3178 ASSIGNMENT 1: MOBILE APPLICATION DESIGN SPECIFICATION

OVERVIEW

Procrastination is the action of delaying or postponing something and is something that we humans have experienced one time or another, at varying lengths. In the most extreme of cases, failure to address procrastination has resulted into huge loss of productivity and may create a sense of disappointment and failure to the user. Living in our app-based society, I propose my solution to this problem, the anti-procrastination app called SaveMyTime.

SaveMyTime is a mobile application aimed towards users struggling with procrastination and attempts at helping users overcome procrastination. To achieve this goal, the app first records all tasks that the user inputs. Afterwards, the app will aim to: detect, notify, prevent, and if needed, outright stop the user to stop from wasting time when their tasks are still at hand.

Long term usage of the app will allow users to better track their procrastination habits via a visually pleasing interface and provide useful statistics about the nature of how and when the user procrastinates. This in term will help users slowly understand and overcome their procrastination habit, and with time, eliminate it permanently.

The Objective of SaveMyTime

- #1: Eradicate procrastination behavior of the user whenever detected by the app
- #2: Be able to notify the user when they engage in procrastination
- #3: Detect when users have stopped working on a task via background activity tracking, motion-sensor tracking, microphone-activity tracking, and GPS tracking
- #4: Tracks and records the user's procrastination habits to provide as feedback to user

- #5: Help users become self-aware of how and when they procrastinate by providing useful personalized statistics
- #6: Help users manage their schedule and prioritize tasks more effectively
- #7: Work with the user's schedule and remind users when they are schedule to work on a task
- #8: Be able to categorize tasks via short-mid-long term and prioritize reminders and alerts according
- #9: Introduce achievement indicators and incentives which reward users for maintaining a procrastination-free habit
- #10: Be able to work against the user's actions. The app determines when the user is postponing activities and will attempt to "do what's good for the user".

Target Audience

The target audience for the app will be aimed towards 21st century young adults aged between 18-25. Alarming statistics show that 25 to 75 percent of college students procrastinate on academic work. Most college students nowadays are tech-savvy and have a great understanding of utilizing phones and phone apps, Hence, my app is a great solution to address many student's procrastination issues.

To further analyze our target demographic using, we make assumptions on the user:

A. The user is struggling with procrastination

B. The user may unaware when they engage in procrastination, only to discover in despair how much time the user has wasted.

C. The user is considerably mature and aware of his/her procrastination habit, hence why they have downloaded the app.

D. The user suffers from poor time-management skills and requires assistance for a well-planned schedule

E. the user (being a teenager) may lack self-responsibility and have a rebellious nature within them.

We also assume that the users have tried their own tactics to deal with procrastination:

I. Implementing a reward incentive to motivate the users to complete a task

II. Ask a close friend or relative to check up on them

III. Minimizing distractions in their environment

IV. Prioritizing tasks by importance and doing important tasks first via recording somewhere

V. Using task and time management apps to schedule their time

My app presents itself as a more efficient solution to procrastination than the ones listed above, while also accounting for the nature of the target demographic.

1. Independence – According to **C**, the user will prefer to handle the issue themselves. Solution **II** is inefficient and requires assistance of another user. the app will allow the user to manage their own complications.
2. Self-awareness – Problems **B** and **D** could be solved through **IV**, **III**, and **V**, but according to **E**, the solution requires the user to be responsible enough to schedule and prioritize their tasks themselves. My app solves this issue, by providing a calendar interface which can automatically prioritize tasks
3. Convenience – The user requires multiple help from **II**, **IV**, and **V**. What my app can provide, is an all in one access for keeping track of the user, prioritizing task automatically, and acting as a schedule manager for the user's tasks.
4. Distraction minimization and punishment-based reward – The user may use **I** and **III** to solve their issue. However, according to **E**, the user may not bother cleaning his environment, or may do a guilty pleasure of taking his incentive. The app solves this issue as it does not require **I** and **III**, instead acting as a punishment-based app which will punish the user if they ever get distracted. The punishment will make sure the user won't be distracted.

Application Features

- Provide local push notifications which can come in many forms:
    1. Location-based notifications – Detect when users are away from their scheduled work session and notifies the user.
    2. Reminder notifications – Will either remind users to record tasks down when none persist. Will also remind users of existing scheduled tasks. Notifications will be assigned based on the priority of tasks.
    3. Behavioral notifications – Notifications that pop up by app usage and background activity tracking – if the user procrastinates by browsing other phone apps, these notifications will pop up.
- Provide a calendar function to allows users to add, update, and track their tasks until completion. In addition, allow users to track their progress regarding their procrastination habit.
- Spotify integration allowing users to play whichever music suits them during their working session
- Punishment-based features which motivates users to stay in their work state such as (subject to changes):
    1. Alarms which cannot be cancelled until the app determines the user is back to working
    2. Payment feature which automatically takes a certain amount of money from the user to be donated to charity if the user is procrastinating.
- Utilize location-based tracking via GPS to determine user state
- Utilize phone activity tracking to determine user state
- Utilize phone microphone tracking to determine user state
- Utilize phone motion sensor tracking to determine user state
- Achievement feature which rewards users for performing certain tasks
- Cloud integration with Google Firebase to provide storage and user backup

- Provide useful procrastination statistics to the user which includes:
    1. Percent successful procrastination-free days
    2. Longest procrastination-free streak
    3. Most common time-of-day the user procrastinates
    4. Most productive time-of-day the user works
    5. Average procrastination time
    6. Longest procrastination time

iOS Features Implementation Outline

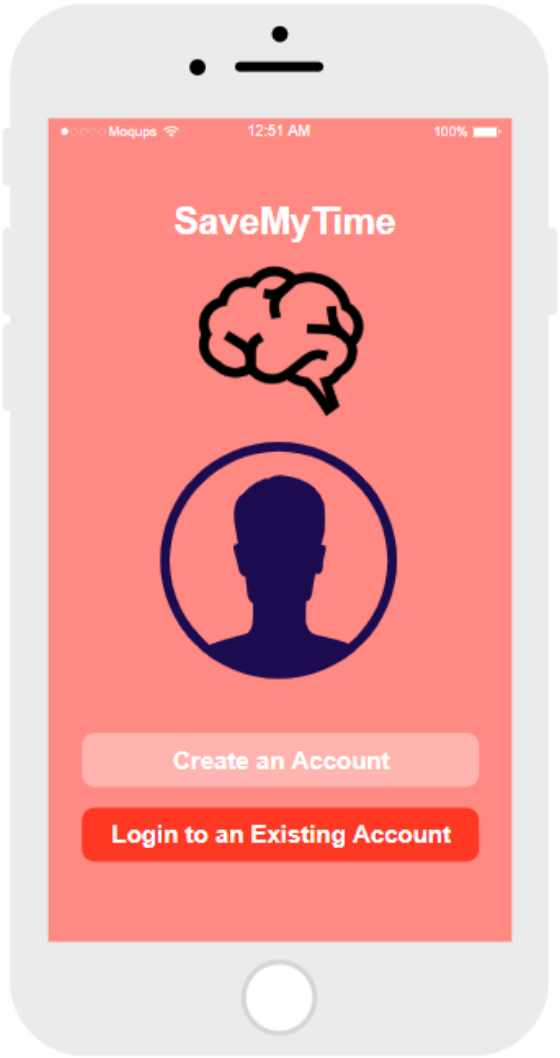
- #1: Calendar Functionality will be implemented via in-built custom **Swift classes**, storing tasks as objects inside Google's Persistence framework, **Firebase** Firestore. The bulk of tasks will be stored using **Firestore**. Personal user login details will be stored via the **Keychain Services API** framework. This allows data encryption of the important data.
- #2: Procrastination statistics will be implemented similarly via custom built-in **Swift classes**. I will be creating mathematical algorithms utilizing data generated by the app through user usage, converting data into a readable format to the user, and storing this info into **Firebase FireStore**. Statistical information generated will also act as trigger points for achievements in the app such as "You have not procrastinated for over a week! Good Job!".
- #3: Location-based tracking will be implemented through extensive use of Apple's **Core Location** framework. Data retrieved from the utilizing classes like **CLLocation**, **CLGeocoder** will be used when determining a user's working state.

- #4: phone activity tracking will be implemented via utilizing AppDelegate methods such as **UIApplicationDidGoToBackground** and **UIApplication** class methods such as **UIApplication.willResignActiveNotification**. By using such methods, the app will detect whenever the user exits the app, and is hence procrastinating.
- #5: To perform phone microphone tracking features such as audio-level tracking, I will be utilizing the **AVAudioRecorder** framework, this allows my app to detect if a user's environment is too loud for working sessions.
- #6: To perform motion-sensor tracking features to detect if the user should not be using the phone during work, I will be utilizing the **Core Motion** framework in the app. This allows the app to detect unnecessary physical actions on the phone and place an alarm/reminder.
- #7: Notifications will be implemented via the **UserNotifications** and **UserNotificationsUI** framework
    1. Location-based notifications – will utilize **Core Location** and be event-triggered via detecting the user's location.
    2. Time Reminder notifications – will be scheduled whenever the user has tasks which will be due soon, this will utilize both **UserNotifications** and **EventKit**.
    3. Behavioral notifications – will be event-driven when the app detects changes to external environment. I will be utilizing **AVAudioRecorder** for sound-based behavior, and **Core Motion** for motion-based behavior.
- #8: Spotify integration will be implemented via utilizing the **Spotify iOS SDK**. User authentication is required first, which is handled in the **AppDelegate** file of the app.
- #9: Punishment-based features which motivates users to stay in their work state such as (subject to changes):
    1. Alarms which cannot be cancelled will be implemented via the **EventKit** framework.
    2. Charity Payment feature will be decided based on payment integration e.g. **Apple Pay** and the decided Charity to donate to.

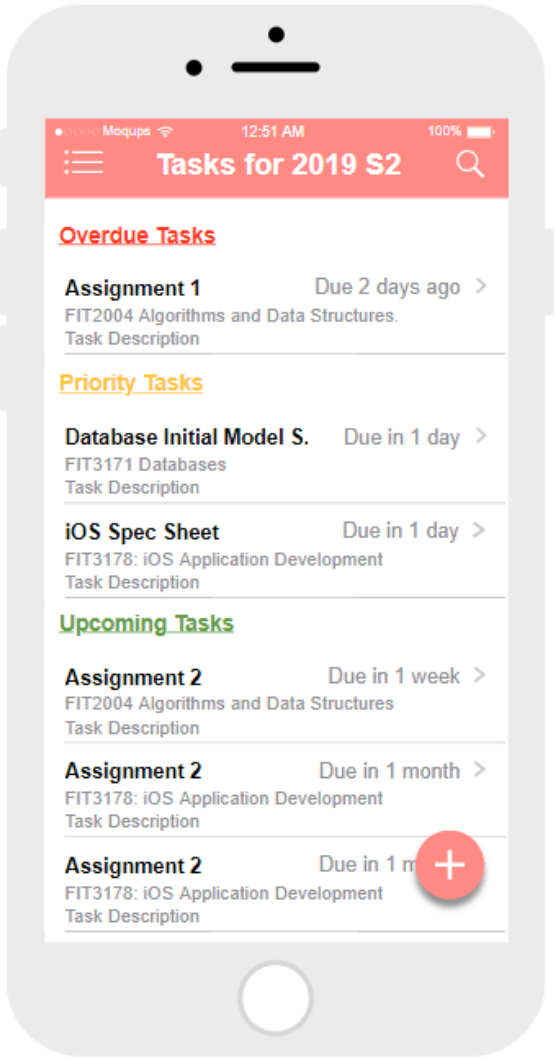
Application User Interface Design & Navigation

Based on the outlined features, the app will feature a minimalistic and visually pleasing design, with 4-5 Main UI Views.

Login UI View



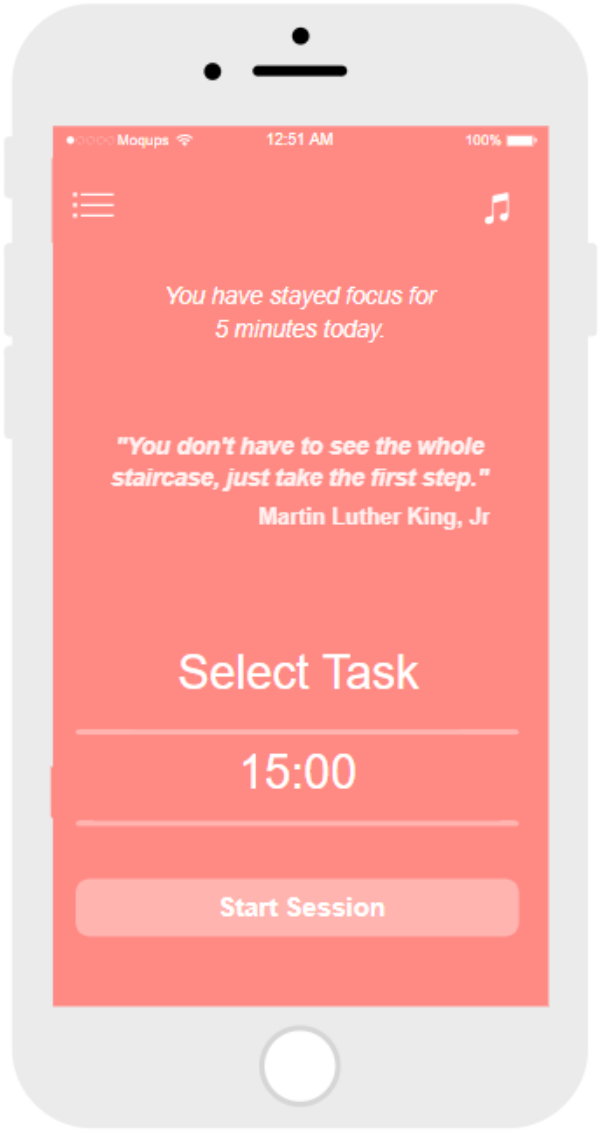
Task Dashboard UI View



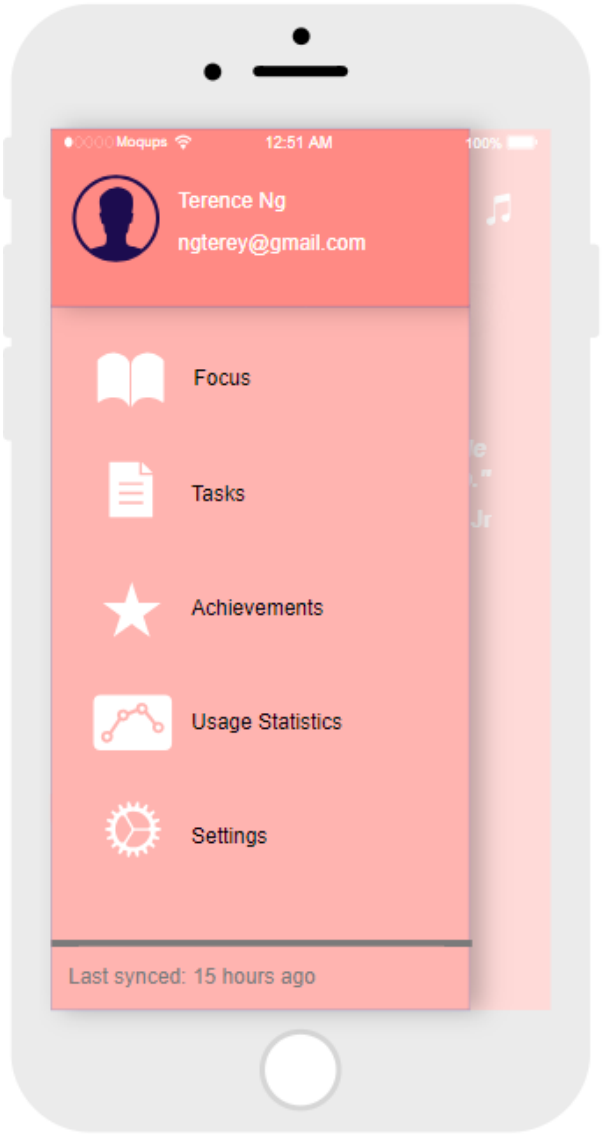
My Login and Task user interface design features a simple color scheme and utilizes simplistic UI elements. Metaphors are employed which follow **Apples Human Interface Guidelines** such as hamburger icons as navigation menu popups, and search icons for filtering tasks.

Users will be able to filter specific tasks, click on the task to modify existing details, and add new tasks via the **Floating Action Button**.

Focus Session UI View



Navigation Menu UI View

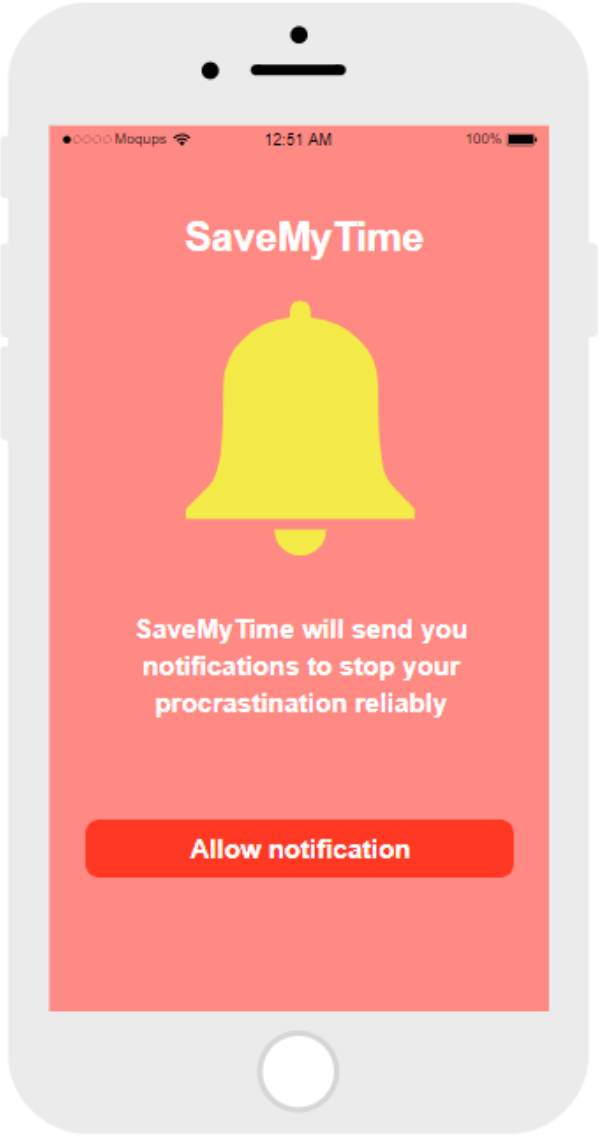


The following depicts my initial Focus Session UI View and navigation menu UI View. Color scheme is still followed, and the navigation menu provides easy navigation between different features of the app.

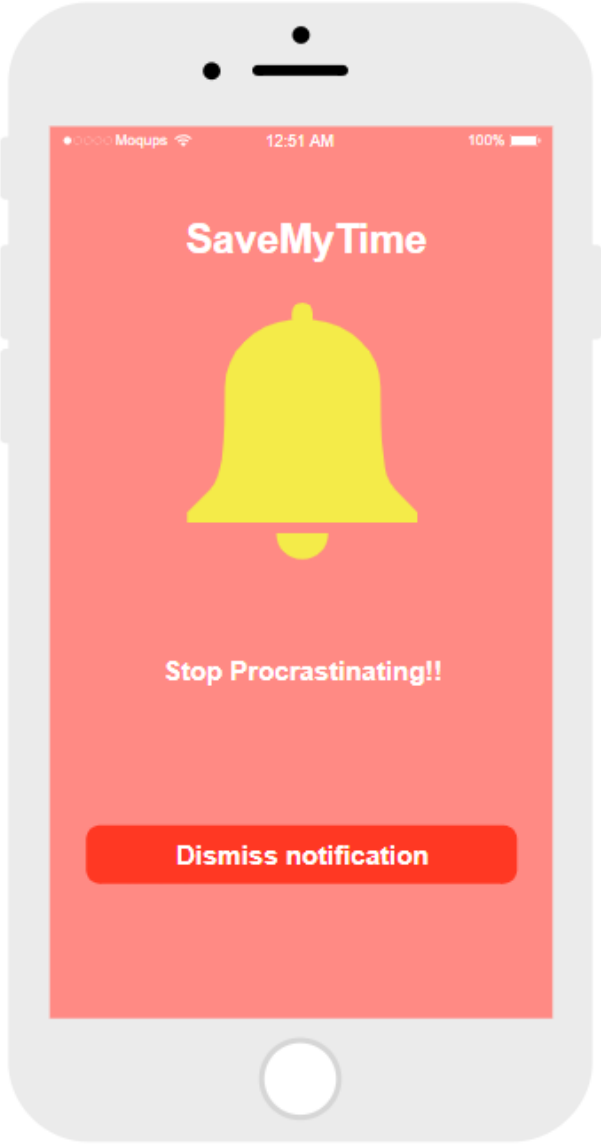
Users can listen to music via Spotify during the focus session by selecting the music icon. Users can also specify the task to work on and amount of time they want to spend for the focus session.

On the Navigation Menu, we see that the user has signed in via user/password managed through **Firebase**, we also see last data sync.

Notification Permission View



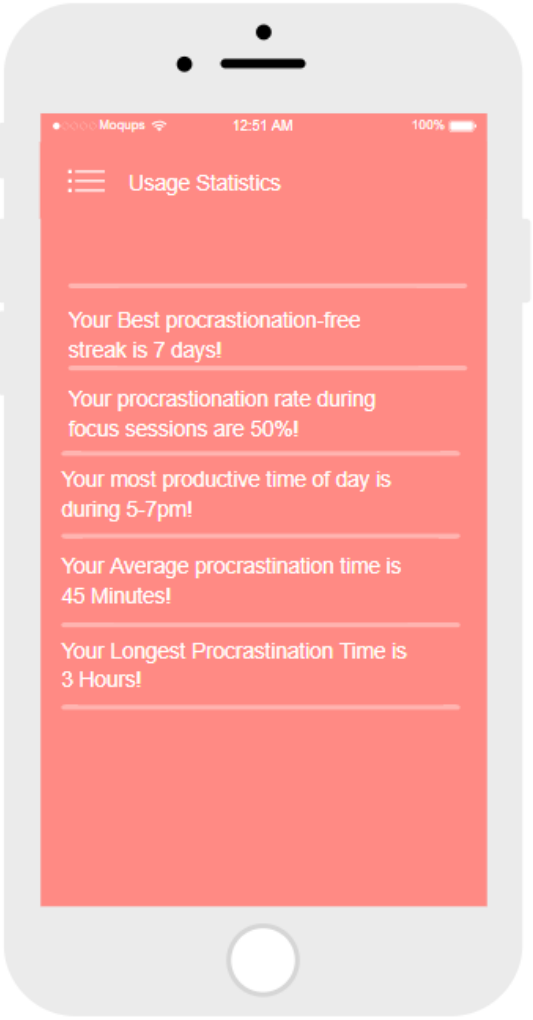
Notification Example View



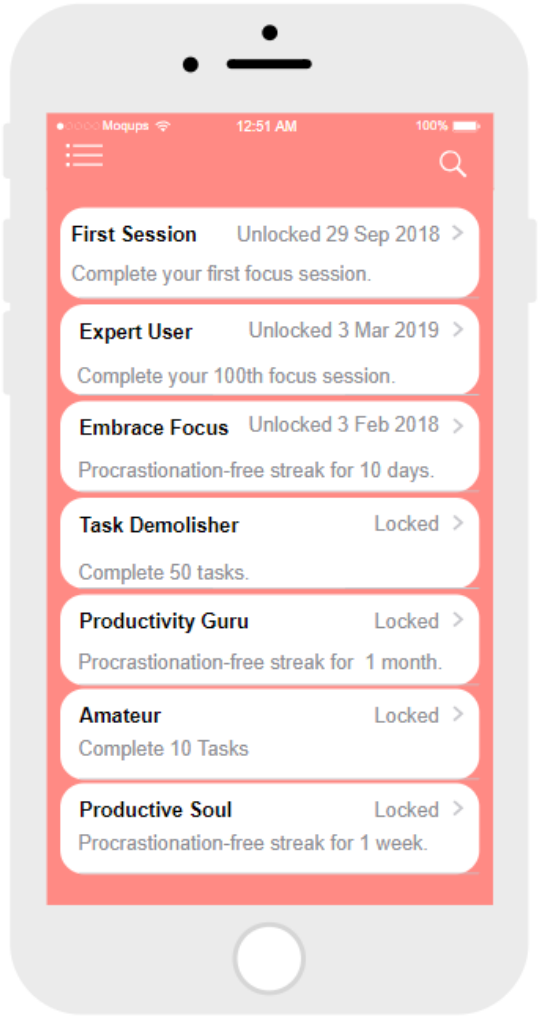
The following show how notifications are utilized in the app. First the user must allow them, afterwards the right reminder will pop up alongside a very annoying alarm which annoys the user to dismiss it.

Note: After dismissing if its still detected the user is procrastinating the alarm and notification will ring again.

Usage Statistics UI View



Achievements UI View



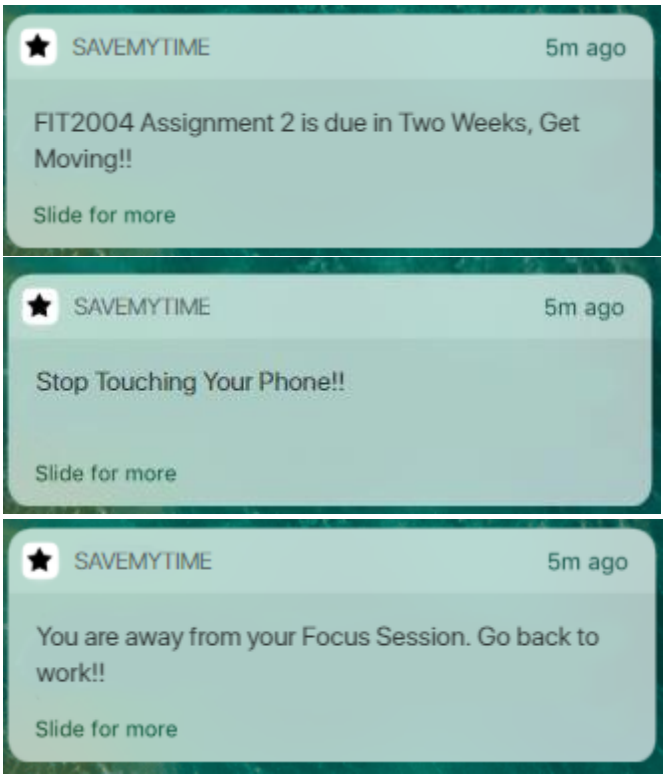
The following shows my prototype design for displaying usage statistics to the user. It is a very basic conceptual model and will be suspected to improvements including displaying visual features such as graphs, and more attractive pictures representing the statistic.

Furthermore, the following is my prototype UI for my achievements page.

Users can search for specific tasks or select the tableView task to view additional details.

The achievement View may undergo rework to allow for better visual design and convenience (such as being able to select unlocked vs locked achievements).

Types of Lock Screen Notifications



Following are some examples of notifications which results from various cases during usage of the app. These notifications are further equipped with alarm features to annoy the user.

Scope and Limitations

The minimum functionality necessary for SaveMyTime to be a useful product will be outlined below:

1. Able to accurately record, edit, and manage tasks created by users.
2. Able to provide timely notifications and alarms during focus sessions to prevent a user from procrastinating.
3. Able to provide push notifications for users when task deadlines are approaching.
4. Able to provide valuable statistics to allow users to understand their procrastination habit more.
5. Able to successfully detect when users are engaging in procrastination via several methods:
    a. Phone app activity tracking
    b. Phone motion sensor tracking
    c. Phone GPS tracking
6. Able to successfully issue alarms and alerts whenever detected procrastination and stop users at the minute.
7. Able to provide persistent storage and offline support via utilizing Google's database framework Firebase Firestore.
8. Able to provide seamless navigation in the app via the use of metaphors including but not limited to:
    a. hamburger icons
    b. Navigation menus
    c. Understandable buttons

Project Timeline

Description	Start Date	End Date	Duration
Set up Project Architecture and Initial View Controllers	Sat 13th April	Sun 14th April	2 Days
Implement Project User Interface and Connect all Segues + Labels	Mon 15th April	Sun 21th April	6 Days
Implement Firebase Firestore and Firebase Authentication for data persistence	Mon 22nd April	Wed 24th April	2 Days
Implement TableView Data Structure and connect data with Firebase	Wed 24th April	Fri 26th April	2 Days
<Milestone 1> Prototype 1 – Testing for basic implementation of task management app	Sat 27th April	Sun 28th April	1 Day
Implement Apple API's for notifications and alarms	Mon 29th April	Sun 5th May	6 Days
Implement Apple API's for motion-sensor, Location Services. Also perform Spotify Integration	Mon 6th May	Sun 12th May	6 Days
<Milestone 2> Implement algorithms for statistics and achievements – Prototype 2 – App with most functionality applied	Fri 17th May	Sun 19th May	2 Days
App Modifications for bugs, improvements, and testing	Mon 20th May	Sun 31st May	11 Days
<Milestone 3> Final testing - Final Product	Sun 31st May	Sun 2nd June	3 Days