Final Report

Team: sddec19-03 **Client:** General Public **Advisor:** Dr. Goce Trajcevski

GoMe

A Life Improvement App

Team Members:

Michael Arnold- Chief Engineer

Jacob Montgomery - Lead UI

Jaclyn Ralfs - Data Analytics/Scribe

Akaash Suresh - Engineer/ML Tech

Mark Marrano - Systems Engineer/Requirements Analysis

Bailey Jensen - Lead Back End/ML Tech

Team Website: http://sddec19-03.sd.ece.iastate.edu

Created: 11/29/2019

Table of Contents

1. Introductory Material

1.1 Acknowledgement

We would like to thank Iowa State’s Dr. Goce Trajcevski for giving technical advice and resources in our weekly meetings. We would also like to thank the Iowa State University Department of Electrical and Computer Engineering for the opportunity to work on this project and gain professional experience before graduation.

1.2 Problem Statement

Today’s world is filled with productivity-assisting and enhancing applications. These digital organization methods offer many benefits including organizing your busy life, staying on top of your events or tasks, and improving your time management. Popular calendar apps like Google Calendar and Apple Calendar are free to use, allowing millions of people to take advantage of their services. In fact, these digital calendar applications are so popular that more than 70% of people rely on them in their daily lives (Ecal). In addition to that, the software world offers nearly 17 million applications to help a person create and manage lists of tasks they need to complete (Kashyap, Vartika). Obviously, there is a high demand for applications that assist in managing your hectic life, but in reality, how much are these apps are actually adding to your life? How many benefits do these applications provide over using pen and paper? Most importantly, how are these applications coaching you to improve your productivity or help improve your quality of life? When using a classic productivity app, it is your job to input and manage your schedule and tasks, but what if the technology was able to do that for you? What If there was a technology that already knew what you did throughout the day and used that to help guide you to spend your time in a more effective way?

To solve this problem we have developed GoMe, a social media application that combines the successes of existing productivity systems and goes multiple steps further by adding features to help you stay on top of your life. GoMe keeps a record of what you’ve done throughout your day like sleep, work, and social activities. By using device location, the application recognizes how a user is spending their time and will make adjustments if it acknowledges the user is not sticking to their schedule. Going further, GoMe takes the user’s list of created to-do items and integrates them into their schedule while making sure to prioritize the user’s time towards the most important tasks. By analyzing a user’s time consumption, GoMe captures and categorizes your life into simple containers like work, sleep, social time and wellness. This data is used to suggest activities on your calendar that could make you a more balanced person, ensuring that you spend time in the different areas essential to living a healthy life. On top of all of this, GoMe is also a robust social media application, allowing for user profile customization, group tasks and activities, recommended social activities, and even a feature to find and schedule an activity with a group of people based on availability. Overall, GoMe aims to be one’s very own personal assistant that helps you optimize your daily life without having to manage it yourself. In the end, GoMe will pave the way for you to increase your potential in all major areas of life.

1.3 Use Case Diagram

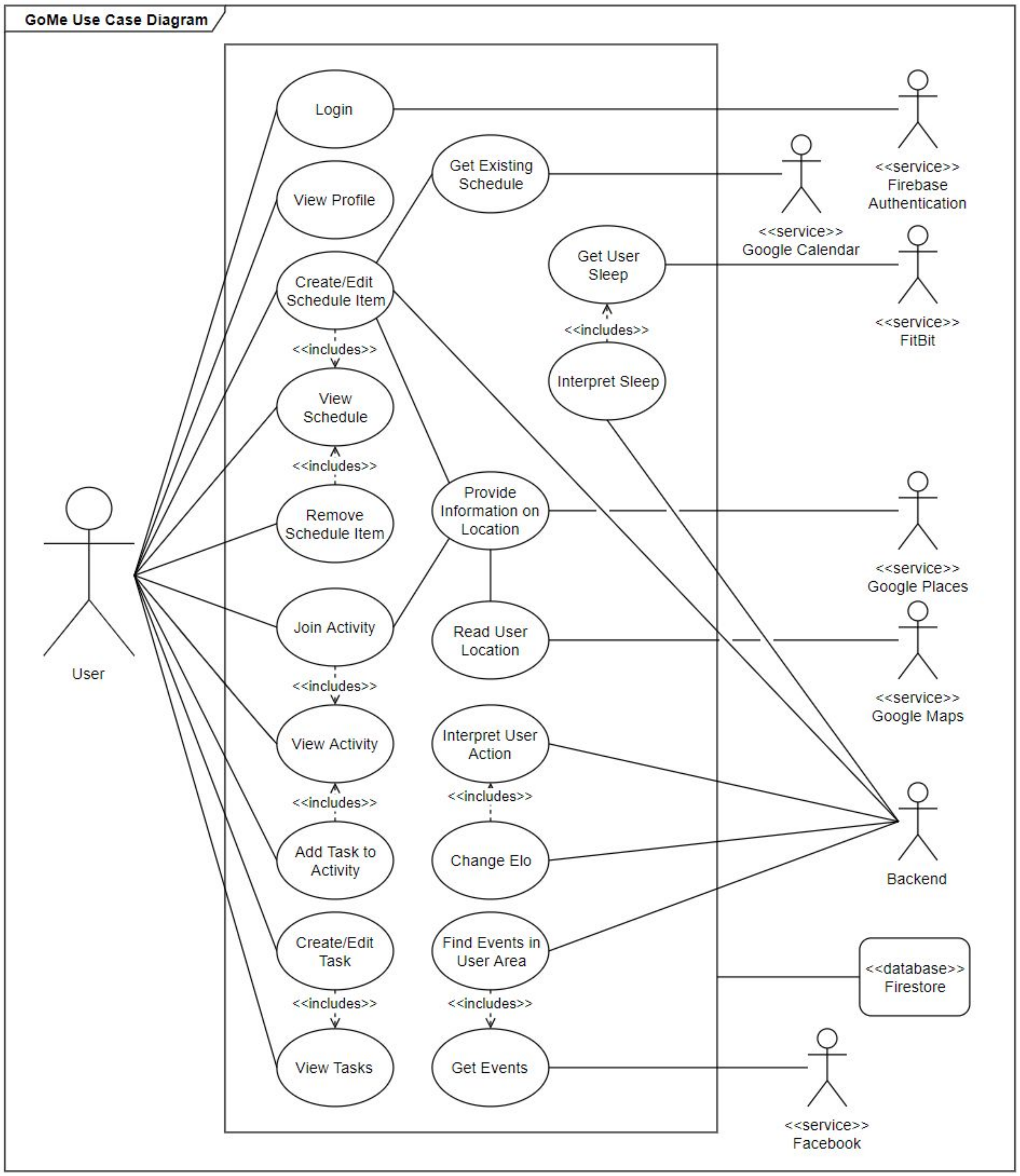


Figure 1: Use Case Diagram

Our updated Use Case Diagram fully demonstrates the use case actors, services and cases for our final application. For graphic simplicity, Firestore connections are black-boxed for the use case diagram.

1.4 Intended Users and Intended Uses

The intended users for our application are people who want to optimize their schedule to increase their productivity and balance their time. User's will typically be schedule-oriented and have the desire to track their activities in order to reach optimum hours of sleep and/or work goals while still making time for social events and/or other activities.

We intend for users to use our application on a daily basis to track their schedule. According to the use case diagram above (*Figure 1*), they will be able to use the application to view and update their schedules, activities, and tasks. A user will also be able to create and update a viewable profile page. Examples of more specific use cases would be recommending activities the user could complete with their friends, suggesting an optimal time to go to sleep, or proposing ways to spend free time.

1.5 Assumptions and Limitations

Assumptions

- Users will carry their powered-on cell phones to/from all activities
- Users will have the app enabled consistently in order to track location data
- The technologies used within the application will secure user information as indicated

Limitations

- Application is only available on Android devices
- Application must not use an unreasonable amount of battery life in order to ease problems for the user

2. Specifications and Analysis

In this section, we discuss the functional and non-functional requirements for our application. We expand on specific technologies used for development, and the rationale behind why we chose these technologies.

2.1 Functional Requirements

- Location recognition
  - The system shall recognize when the user's device has moved location
  - The system shall recognize when the user's device has arrived at or left an activity location
  - The system shall recognize when the user is not where they are supposed to be according to their scheduled items
- Dynamic Scheduling
  - The system shall make start/end time adjustments corresponding to their location
  - The system shall order activities by start time
  - The system shall display tasks and activities in the schedule
  - The system shall prioritize tasks that are due in the schedule
  - The system shall not have overlapping schedule activities
  - The system shall allow the user to add or delete an activity from their schedule
  - The system shall recognize the user's time management strengths and weaknesses
- Collaboration
  - The system shall allow an activity to be joinable while it is public
  - The system shall trigger activity and schedule updates for all users in an activity
  - The system shall delete a task for all users included when it is completed
- Notifications
  - The system shall notify the user when an activity has changed
  - The system shall notify the user when a member in an activity is late
  - The system shall notify the user when a different user completes a task for them
  - The system shall notify the user when they receive a direct message from another user.
- Profile & Social Media
  - The system shall allow the user to create an activity for the public
  - The system shall allow the user to upload pictures
  - The system shall allow the user to post a status update
  - The system shall allow the user to reset their password
  - The system shall allow the user to change their information
  - The system shall allow a user to see another user's profile page
  - The system shall allow a user to share their schedule
  - The system shall allow a user to be friends with another user

  - The system shall allow a user to see their friends and profile page
  - The system shall allow a user to filter and search for activities
- Recommendations
  - The system shall find activities for the user based on location and user interest
  - The system shall find activities for the user based on open space in their schedule
  - The system shall find activities for a group of people based on schedule availability and location
  - The system shall recommend future activities for the user based on their day so far
- Tasks
  - The system shall allow a user to create a task
  - The system shall allow a user to label a task
  - The system shall allow a user to edit a task
  - The system shall allow a user to complete a task
- Activities
  - The system shall allow a user to create an activity
  - The system shall allow the user to make an activity public or private
  - The system shall allow the user to upload a picture for an activity
  - The system shall allow the user to mark an activity with a tag
  - The system shall allow the user to share an activity
  - The system shall allow the user to view their activities for work (meetings, phone calls, etc.)
- Progress Logging
  - The system shall log each location change in the user’s day and display it in the progress page
  - The system shall display charts to the user showing their time usage in different categories
  - The system shall supply a brief summary of each user’s day in the past (recap)
  - The system shall update the user’s life score (Elo) after logging events

2.2 Non-Functional Requirements

- Performance
  - The application shall update data in realtime
  - The application shall be able to support 100,000 users
  - The application shall not buffer or lag while on strong internet connection
  - The application shall not log the user out of the app while it is installed
- Scalability
  - The application shall be able to scale to 1 million users
  - The application shall be operational on any Android device above API 14
- Testing
  - The application shall have a process to make testing the dynamic schedule accurate and efficient
  - The application shall be connected to a continuous integration testing pipeline
  - The application shall be UI tested by a crawler at least 100 times
- Security
  - The application shall not allow users to have the same account email as each other
  - The application shall use Firebase for secure authentication
  - The application shall have input sterilization methods for every user input component
- Privacy
  - The application shall not allow a user's friend to see their schedule if it has not been shared
  - The application shall not track the user's location without permission

2.3 Interface Requirements & Specifications

With the exponentially growing amount of applications available to users, it has never been more difficult to create a product that stands out among the crowd. This is why it is vital for our application to have an attractive user interface as well as a user experience that draws new users to the application. Our goal is to provide the user with a system that encompasses many parts of their life that they will enjoy experiencing. Instead of being a hassle to use, we want GoMe users to be extremely satisfied by their time using the application. Including a strong UI/UX into our application is easy to say, but measuring the interface success among a set of users is another story. This is why we have come up with a set of requirements with matching fit criterion to test the interface of the application.

- Look and Feel Requirements
  - The application shall have an attractive UI according to 80% of people
  - The application shall have both a light and dark mode feature
  - The application shall use colors that are not painful for the human eye to look at for more than 30 minutes
- User Experience
  - The application shall be easy enough to use that a 25 year old can figure out all of the functional requirements on their own
  - The application shall allow a user to personalize their profile
  - The application shall have a robust onboarding process that clearly explains the app’s purpose and main features
  - The application shall use simple language so that someone under 18 can understand
  - The application shall use appropriate symbols for buttons according to the material design library

2.4 Technologies Used and Rationale

The main technologies used for our application are Android Studio for mobile development, Firebase Cloud Firestore for the database, and Azure machine learning platform. In this section, we touch on how we used these technologies and our reasoning behind these choices.

2.4.1 Mobile Development Framework: Android Studio

Choosing a development framework, we had to balance different factors in order to properly make our decision. After reviewing our options, we ultimately decided between programming native applications (using iOS Swift and Android Studio) or using a hybrid development framework like React Native. The most important factor when making our decision was application complexity.

Rationale - Application Complexity

As you read further, it will be clear that our main idea revolves around a wide set of complex features that depend on very specific device capabilities. From prior experience using hybrid development frameworks, it can sometimes be difficult to create high-complexity features without taking advantage of the native development environment. For this reason, we decided that using native development would be our best option. In the end, we also decided that we likely did not have time to develop both a separate Android and iOS application. Since Android offers all free development services and is easier to work with and test on a real device, we chose to start with Android over iOS, hence making our mobile development framework of choice Android Studio.

2.4.2 Machine Learning Framework

Our ML framework is based on the Azure machine learning platform, with an experiment that helps us predict sleep patterns based on existing data. We decided to transition to this from Keras because it had a much easier implementation, as well as being able to deploy to a web API at a press of a button.

2.4.3 Database: Firebase Cloud Firestore

GoMe will be handling a lot of user data in order to get a good picture of how the user is spending their time, what the user likes/dislikes, and what the user needs to do. It then needs to feed this information back to the user and the ML algorithm in an efficient and meaningful way. To do this we will be using a Google Firebase Database.

Google Firebase offers two types of real-time data storage– a real-time database named Firebase RTDB (Realtime Database) and a new flagship database called Google Cloud Firestore. According to the Google Firebase documentation, the RTDB is a low latency database that is often used for applications that need to update in real-time and get information quickly. We had originally thought that we would use the RTDB because of familiarity and its ability to handle a large request load in real-time, however, reading further into Google Firestore led us to change our minds. Google Firestore is an upgrade to Firebase RTDB according to Google's standards, as it is very similar to past platforms, but is also more technologically advanced and will be supported more frequently in the future. Firestore also has a modern and friendly user interface and offers a more intuitive data organization model compared to the RTDB. For these reasons and others, we decided to use Firestore instead of the RTDB. We are certain that Google Firestore is the best database service for GoMe for the following reasons listed below:

Realtime Updates

As mentioned, all of Google's database products update in realtime. This is a vital feature, saving us time as developers and being able to rely on Google for fast data updating performance. Using the realtime feature, we are able to make efficient read and write updates to our database. This adds a great amount of positive user experience to the application.

Authentication

Another benefit of using Google Firebase is the secure authentication feature. Once again, this saves us the cost of having to integrate this ourselves in a reliable way using another method. In addition to that, Firebase's authentication feature provides extensive capabilities that maximize the user's sign up or log in experience. Firebase allows many different sign in methods such as email, phone number, username and more. Overall, Firebase provides the best option to know a user's identity.

Push Notifications

An important functionality needed for GoMe is the capability to send push notifications to different user's devices. With Google Firebase Cloud Messaging, we can easily integrate push notifications to certain groups of people subscribed to a specific topic. On top of that, it's also free to use.

Integration

You may be noticing a pattern. We are choosing to stick with Google products because they all integrate very nicely together. We have already made the decision to use Android Studio for development and continuing to use products backed by Google helps us save both time and stress. Google is a reliable technology company and gives us the capability to perform features that could take too much time to develop in the duration of this course. Conveniently, directly inside of Android Studio is the documentation to set up and use all of Firebase's offered features. In addition to this, Firebase includes a Test Lab that is already integrated as a feature in Android Studio.

Security

Google explains that while RTDB allows you to set security rules, the read and write rules require separate validation. This can become a tedious process and can be confusing for beginners to get the hang of. On the other hand, Cloud Firestore offers simpler, non-cascading security rules that automatically validate. Cloud Firestore also provides the option to implement other security services like IAM (Identity and Access Management). Overall, both databases provide strong security options, but Firestore provides a simpler, yet more advanced set of security features, making our lives easier and our app more secure.

Analytics

Firebase offers a simple analytics platform to help us understand who is using our application and how they are doing it.

Querying

While both RTDB and Cloud Firestore allow you to query, filter, and sort data, RTDB lacks in a lot of ways, while Cloud Firestore excels in providing intuitive query statements that give you plenty of options to get data no matter how you organize it. RTDB stores data in a nicely organized JSON tree, but a single query will return the entire structure, no matter how deep. Cloud Firestore organizes data in collections and documents, comparable to a folder with documents inside of it. This allows you to perform shallow queries and filter by a single attribute or property. Overall, Cloud Firestore allows us to be more creative with how we organize data, knowing that we will be able to easily access it in any situation.

Scalability

Google documents the RTDB as having good scalability options, but at some point it starts to get complicated. Unfortunately, once you reach 100,000 simultaneous connections, RTDB will require database sharding, which means you need to use a second database to keep scaling up. This might not be an issue for us in this class, since we probably will not reach over 100,000 connections at the same time, but thinking long-term, the RTDB does not provide a great scaling solution for large applications. On the flip side, Cloud Firestore offers automatic scaling with a limit of 1 million concurrent connections. Google also plans to keep increasing this in the future. Because of this, Firestore offers a much more scalable database service for a large application.

As listed above, Google Cloud Firestore offers great features that saves us the tedious hassle of implementing ourselves. This includes authentication, general data security, and scalability, all the features that are vital to the long term success of our application. Although we originally planned to build the app around Firebase RTDB, we decided that Cloud Firestore provided a more advanced, updated, and overall better service for our data.

3. Revised Design

As expected, the design of our application has changed a lot from Spring semester to Fall semester. While things were changing in our design, we had to come up with a lot of creative solutions to accomplish different functionalities throughout the application. This section contains all of the major design patterns, data models, services and components that allowed us to create GoMe.

3.1 System Components

Below is a detailed description of all the components and services within our application and what contributions they bring to create the different functionalities found within GoMe. These different components were what allowed us to bring our idea to life.

3.1.1 Component Diagram

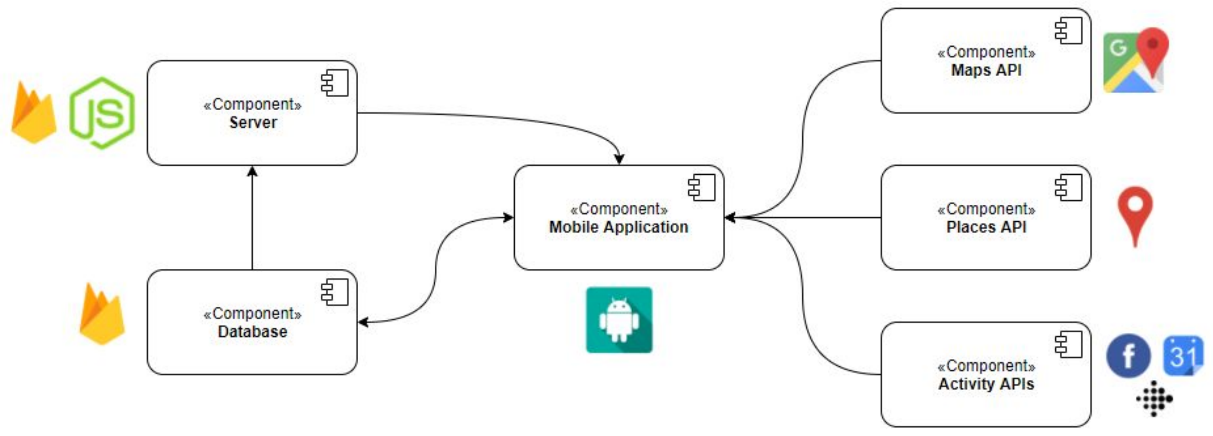


Figure 2: Component Diagram

Above is a high-level component diagram of how our mobile application interacts with different components. As seen, the center of the system is the mobile application, but it communicates with several APIs as well as a simple server and Firebase Database in order to maximize functionality. Expanding on the listed APIs, the application uses Google Maps for location tracking and verification, Google Places to help us link activities to locations, Facebook for authentication and social activities, Google Calendar for syncing calendar events, and Fitbit for sleep data.

3.1.2 Models

In order to provide all of the required functionality, GoMe needs to possess various different objects that play a specific role inside the application. In this section, we will run through all of the important objects that hold the information displayed in the apps interface and held in the Firebase database.

Schedule

Arguably the most important feature of our application is the dynamic schedule that is generated for the user everyday. In order to store all of the data for this schedule, we created a Schedule object. This object holds many data lists, for example, the user's schedule items (a list of Schedule Item objects) that is displayed on the schedule fragment screen. It also holds an item called "Schedule Score" which is a simple breakdown of how the user has spent their time during the day. This score is used in the schedule assessment service explained later on. This object also contains the date of the schedule so we can easily query the user's schedule by date. Once a day has been passed, we designed a method to crunch down all of the schedule's data into a small daily recap. This allows us to save storage space and provide the user with a short recap about what they spent their time on during that day.

Schedule Item

The user's schedule object is extremely important, but it serves no purpose without being populated by scheduled events. This is why we designed the polymorphic object Schedule Item. These objects contain a lot of useful data that allows us to show the user details about the things that they will be doing throughout the day. It also contains a place object so we can tell when the user has left or arrived at that schedule item. With Schedule Items, we can easily translate any information into a new object to put into the user's schedule.

Activity

An Activity is a very important application object that serves the purpose of a social event. Similar to something that can be seen in popular social media systems, a public Activity can be created or joined by any user and allows an unlimited amount of users to be involved. You can easily invite users to a new activity while it is being created. Activities allow for an easy method of user collaboration in life events which contributes to the applications several collaborative, multi-user features.

User

Every user on the application has a data model created for them. Inside of this object is all of the data that the user sets in the registration and on-boarding process (user's name, work address, home address, expected work and sleep times, interests and others). All of this data gives us an understanding of the user so we can build them a schedule template and a profile page. The user object also contains location tracking data for the user (places they have been), the user's schedule objects, and all of the activities/tasks that the user is involved in. The User object is present and heavily utilized in nearly every feature across the app. GoMe is centered around improving the person using the app, so it makes sense that the User object is important in every aspect across the application's various functionalities.

Task

A Task is an object that represents something a user must do, just like a normal task in life. A Task can be created by any user and gives the option to also invite other users to take part in the task with them. The Task object has a title, short description, estimated duration, users included, and other optional attributes. While designing Tasks, we came up with a couple different ways that we could classify the items that users need to accomplish during their day. First, not every task has a set start and end time, but some do. Similarly, some required tasks possess a location, while most do not. Lastly, not every Task in a person's life has a deadline and as a result is not a huge priority if they have other activities going on. Taking these considerations into account, we decided to divide Tasks into two primary child objects. These are Dated Task and Checklist Task. The former, Dated Task, is an object that has both a set time and location. For example, a Dated Task could be a meeting for coffee with one of your co-workers. This Task is something you need to do, but when it is over the application will do the work of checking it off as complete. Dated Tasks are easily displayed in a user's schedule since they provide a set start and end time. The latter, Checklist Tasks, are a more traditional approach to a checklist item, lacking a specific time and place, but allowing a user to set a deadline if they see fit.

For example, a Checklist Task could be completing your homework assignment that is due at midnight. This is slightly more complicated to schedule since it lacks a time, but we have created methods to properly put Checklist Tasks into a user's schedule using a couple background services. Once you have completed your Checklist Task and don't need to think about it anymore, simply check it off of your list and the task will be deleted for all users involved.

Priority

A Priority is an object that serves as an easy way to recognize what the user should do with their available time. We developed Priorities with the purpose of storing what the user has either been lacking or what they must spend time on due to deadlines, and routing that directly into prime consideration while building the user's schedule. The Priority object holds attributes of importance value, an optional reference id linking to a task or activity and an estimated duration of the time they should spend doing this priority. As an example of how Priorities are used in GoMe, consider the situation where you have an important feature to program for work by the end of the day. You probably have a busy day as is, but it is essential that you spend enough time today to finish the feature. You create the task in GoMe and it generates a matching Priority object. This Priority has a high importance value and therefore will take precedence over other activities that could be put on your schedule. Having Priority objects ensures that you will spend your time on the most important tasks throughout the day.

Place

The Place objects within our application serve a very important purpose. They tie a real world place to our user's location, activities, and schedule items within the app. This allows us to understand where and when the user will be going throughout the day. The objects contain things like addresses and latitude/longitude which we can use as a cross reference with their actual location. When a user goes through our onboarding process, they give us information such as their home and work addresses. We then assume that they will most often be sleeping at their home and working at their work address. In addition, we ask that the users enter the location of their events over the course of the day. From there, we can build out a good picture of where the user intends to be throughout the day. Using that information, we look at their current location to see if the user is where they should be in relation to their schedule each day, and react accordingly. For example, if a user gets to work on time, it may increase their Elo score (explained below), but if they are late to work, it may decrease their Elo score and give them some tips on how they could adjust their schedule to get there on time in the future.

Elo

We wanted to implement a way the user could recognize their personal progress for each of the major subjects (sleep, work, etc.) - what we created is our Elo object. Elo is a system of ranking typically used in video games today to measure a user's ability/skills against their peers. What we have created is an offset version of this. Our equation scores the user in each of the subjects on a scale from 0 to 100, 0 being rock-bottom and 100 being perfect. Therefore, a user with a score of 80 in the sleep subject is considered to be strongly on-top of their sleep schedule and sleep wellness. The user is also able to see what actions are increasing/decreasing their scores to further understand their personal progress.

3.1.3 Services

Creating a complex application requires work from many different software components inside the application. Here, we will outline the services that were designed to handle specific responsibilities in order to run our application's various features.

Device Tracker (Tracking Service)

The tracking service is used to monitor user location and make calls to the location verification service. It works via a location listener which recognizes when the user location has changed. When it does so, it starts the "on location changed" function and passes it the address that the user has moved to. Then, using the address, we make a call to Google Maps API in order to gain more information about the address, such as latitude and longitude. After that, we parse the information returned from the API call, and log it under the user profile in our database. Lastly, this service makes a call to the location verification service (described below) which will use the information that it acquired. For example, if the user moves from their apartment to campus, the location listener will see the location change and make a Google Maps API call on the new location. Then we will parse and log the information returned and pass it off to the location verification service to do the rest.

Location Verifier (Location Verification Service)

The location verification service is called by the tracking service as mentioned above. When it is called, this service checks the user's current location versus the location of their current or next event and handles it accordingly.

Some of the cases include:

- If the new location is their next event, we note that the user has arrived at the event.
- If the new location is their current event and they have already arrived, we recognize this as a margin for error in the location tracking.
- If the new location is not their current event, but they have not arrived yet, we check to see how far the user is from the event. If the user is going to miss the entire event, we can cancel the event. If the user is going to be late to the event, we check to see how late they will be (using travel time) and push the event back in their schedule.
- If the new location is not their current event, but they had already arrived at their current event, we note that the user has left the event.

Event Handler (Event Service)

The event service stores a list of the user's events for the day which changes dynamically as the user makes adjustments to that schedule throughout the day. It takes the user schedule and always keeps track of the user's current schedule event, as well as their next upcoming schedule event. Other services within the app make calls to the event service in order to make changes to the current and upcoming events, then update those events in the database so that they can be reflected in the user's schedule. For example, when a user arrives to an event, the location verification service will call the event service in order to record the arrival time to the event and reflect it in the database. Another example would be if a user is late to an event, the location verification service will make a call to the event service in order to move the start time of the event. Once it is changed to a new time that the user will be able to attend, it will reflect that in the database and user's schedule.

Schedule Assessment

In order for us to get an understanding of how the user has performed throughout the day, we designed a schedule assessment process. This process takes in the user's schedule for the last day as a parameter. The service then takes a look at how the user has spent their time since they last went to bed, in each of the 4 categories of sleep, work, wellness and social. Once we know how the user has spent their time, we take a look at how the user plans to spend their time the rest of the day. By looking at how the user has spent their time and plans to spend their time, we are able to assess how the user has been performing in those 4 categories. From that assessment, we can generate priorities for the user, which are scheduled into the user's day to make up for poor performance.

Schedule Builder

The schedule builder creates and updates the user's schedules throughout the day. It takes in all of the user's activities, tasks and priorities and organizes them into a schedule that the user can follow throughout the day.

Messaging Service

This service allows the app to call a Firebase function, a simple web function that allows messages to be sent to different topics, which is the basis of how our app will update other users on changes in their own schedule based on changes in other's schedule. This will be the main way for schedule changes to be propagated to all event participants.

Priority Selector (0/1 Knapsack)

The Priority Selection Service is a Java class that takes in a list of priorities for a user and a number representing the duration of a free time schedule item. Taking into account these parameters, the service runs a 0/1 Knapsack type algorithm in order to maximize the number of priorities that we can fit into a block of free time. The Priority Scheduler then interacts with the Priority Scheduler to populate the new items into the user's daily activities. The priorities chosen by the service are then marked as handled and will no longer be considered by a future instance of the algorithm. This process plays a vital role in our dynamic scheduling feature, allowing the app to determine how the user should utilize their free time and send that directly to the Priority Scheduler for immediate placement.

Priority Scheduler

The Priority Scheduling Service is a Java class that accepts a user's schedule object and a list of priorities that need to be inserted into the user's schedule. This is done by taking the priorities and translating them into schedule items that can be placed in the user's schedule. Based on the importance value attribute of the priority, the new schedule item either replaces a part of a block of free time (determined by the Priority Selector) or is force scheduled somewhere into the user's list of schedule items (usually tasks with a deadline coming up soon). This process plays an extremely important role in our dynamic scheduling feature, allowing the user to input tasks and see them show up on their schedule before they are due.

3.2 System Design

Designing the primary functionalities of the application took plenty of thought, hours of whiteboarding, and even more time for final implementation. Fortunately, our team felt that with a strong design going into implementation, we would be saving ourselves from later hassle caused by a poor initial strategy. In the following paragraphs, we outline a brief design explanation for each of our major functional features.

3.2.1 Location Tracking and Verification

Our application relies heavily on monitoring the location of the user for the dynamic portion of the schedule. Location tracking works via a location listener which recognizes when the user location has changed. When it does so, it starts a location change function and passes it the address that the user has just moved to. Then, using the address, we make a call to the Google Maps API in order to gain more information about the address, such as latitude and longitude. After that, we parse the information returned from the API call and log it under the user's profile in our database. Lastly, this service makes a call to the location verification service which uses the information that was acquired.

The location verifier then checks the user's current location (that was passed to it by the location tracker) versus the location of their current or next event. As a part of this process we use a travel time algorithm that finds how long it will take for the user to get from their current location to their desired location. Next, based on a series of possible cases, the location verifier makes calls to the event service that will alter events and make changes to the users schedule as necessary. These changes all take place behind the scenes with very minimal input from the user.

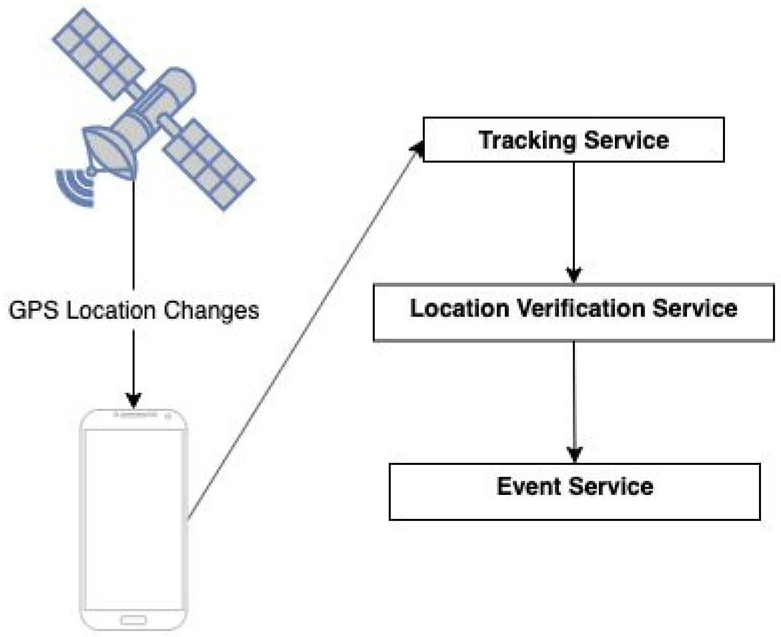


Figure 3: Location Tracking and Verification Diagram

3.2.2 External API Data Collection

For our application to function, we need a number of data points to assist our scheduling function. The APIs we are using (and what we are using them for) are as follows:

- Fitbit to obtain user sleep data
- Google Places to obtain information about addresses the user interacts with
- Google Maps to obtain location information on the user
- Google Calendar to see already existing schedule items/obligations/tasks
- Facebook Events to see what events are going on in the user's area

When implementing these APIs we used the adapter design pattern. Utilizing this pattern ensured that if an API changed how it works functionally, all we need to do is change the adapter class so that our code would maintain functionality throughout the application. By utilizing this pattern, coupling between our code and the external APIs is decreased, thus making our code easier to manage.

3.2.3 Data Design

Firebase Cloud Firestore provides a powerful, yet intuitive and unique way to structure data. Information is organized into 3 main structures. These are (1) collections, which are the highest level structure. Collections are the tables of Firestore, each of which contains multiple documents. Next, there are (2) documents, which are specific, uniquely identified objects holding information. Documents can easily be compared to a sheet of paper or files that would be stored in a folder. Lastly, Firestore includes (3) data, which is simply the attributes describing the document. The following diagram shows how we have chosen to structure our large set of data according to Cloud Firestore's organization model.

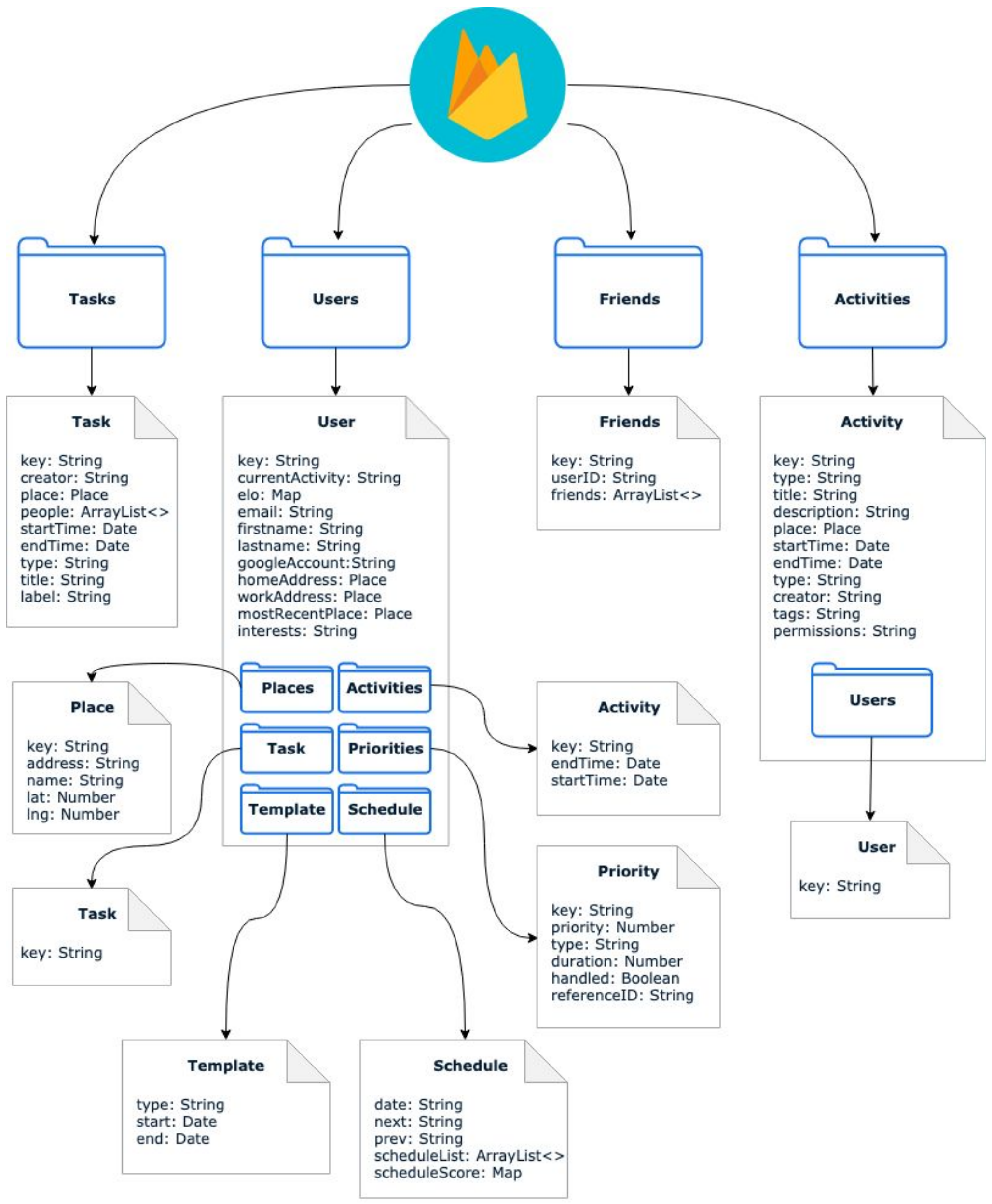
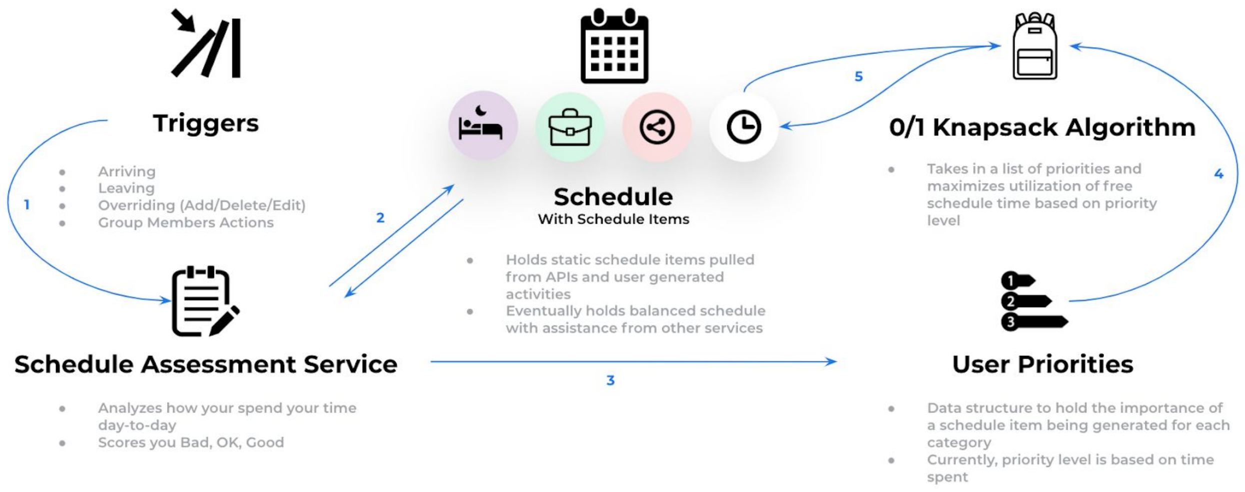


Figure 4: Database Hierarchy

3.2.4 Scheduling Algorithm

In order to schedule user's activities for them dynamically throughout the day, we needed to design an algorithm that could schedule activities efficiently and reflect the most important items for user. The algorithm also needed to understand which activities were most important for the user to do at any given time during the day and then place them into the user's schedule in a way for the user that is feasible for them to accomplish. In order to do that we came up with the following design for the GoMe scheduling algorithm:



Triggers
Arriving
Leaving
Overriding (Add/Delete/Edit)
Group Members Actions
1
2
Schedule
With Schedule Items
Holds static schedule items pulled from APIs and user generated activities
Eventually holds balanced schedule with assistance from other services
5
0/1 Knapsack Algorithm
Takes in a list of priorities and maximizes utilization of free schedule time based on priority level
4
Schedule Assessment Service
Analyzes how your spend your time day-to-day
Scores you Bad, OK, Good
3
User Priorities
Data structure to hold the importance of a schedule item being generated for each category
Currently, priority level is based on time spent

Figure 5: Data Flow Chart

As shown in the above diagram, the process begins with a certain trigger telling the application to update the user's schedule. This can be anything from arriving late to an activity, deleting or adding an activity, or logging into the app for the first time. After we receive signal that an update is needed, the Schedule Assessment Service begins the work on figuring out the user's most glaring needs. This typically analyzes how the user has spent their time and will generate priorities based on the area that the user is lacking. For example, if you only sleep for 4 hours one night, the app will ensure that sleeping is a high priority. Once accurate priorities are in place, the Priority Selection Service is created and selects priority items that can fit into the user's free time. If a priority reflects a task that is due tonight (priority level 10), a block of time is force scheduled for the user to ensure that they have time to finish the task. Finally, once all schedule items are in order, the user can view an updated list of activities for their day.

3.2.5 User Collaboration Design

A large part of the application functionality is allowing users to not only have an effect on their life, but also on the life of other users. To do this, we designed a feature that takes an activity with multiple users, and makes changes to the activity's start time if one of the included users is late to an activity, or will miss it altogether. This is an extremely rare functionality that allows anything you do to have an effect on someone else's time usage. For example, if you are participating in an activity with another user, your personal actions can cause a time change in the mutual activity. This change can have a cascading effect on all of the users included, ultimately changing the way the users will spend their time the rest of the day. This is briefly illustrated at a high level in diagram below. Here, we see one user's action triggering an impact on another user's following activities, very similar to a domino effect.

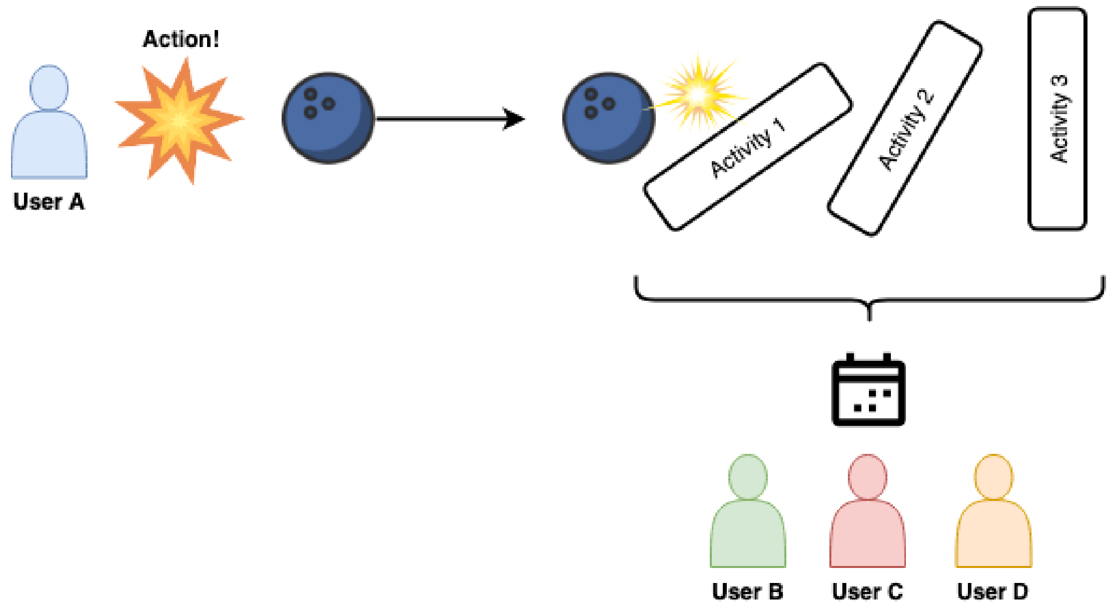


Figure 6: Multiple Plane Effect Diagram

Vital to this functionality, the location verification service (mentioned above) recognizes when a user is too far away from an event to arrive on time. Then, it calculates the travel time it will take for the user to arrive at that event and reschedules the event to a later time that the user will be able to make. If that event has multiple users, it will also reflect the time that the event has been rescheduled to in their calendars and notify them that it was rescheduled to that time because another user in that activity was going to be late. This is further illustrated in the diagram below.

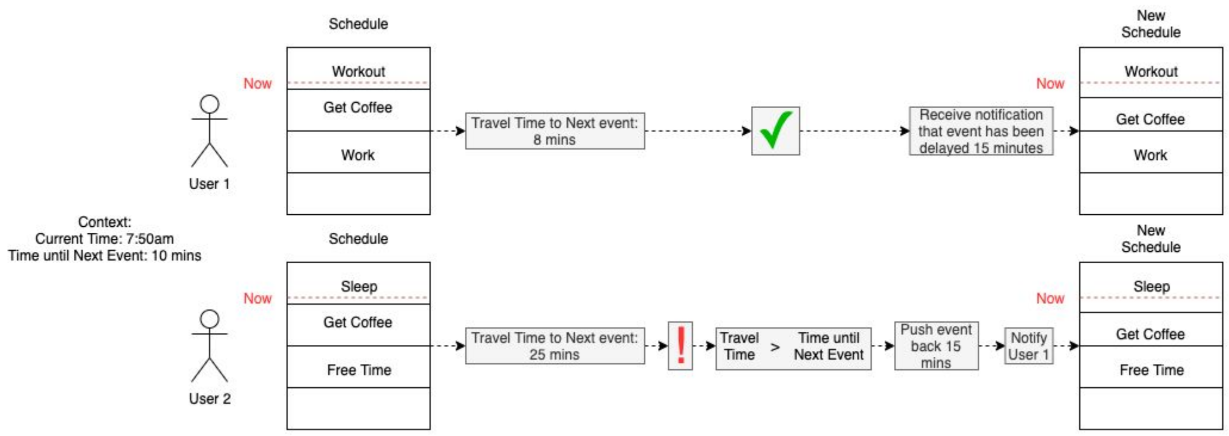


Figure 7: User Collaboration Diagram

3.2.6 Machine Learning Design

Designing the Machine Learning module was relatively easy by using Azure's intuitive drag and drop approach. The following graphic shows the basic layout and data flow of the two models, one for predicting sleep start time (left) & end time (right) based on the day of the week.

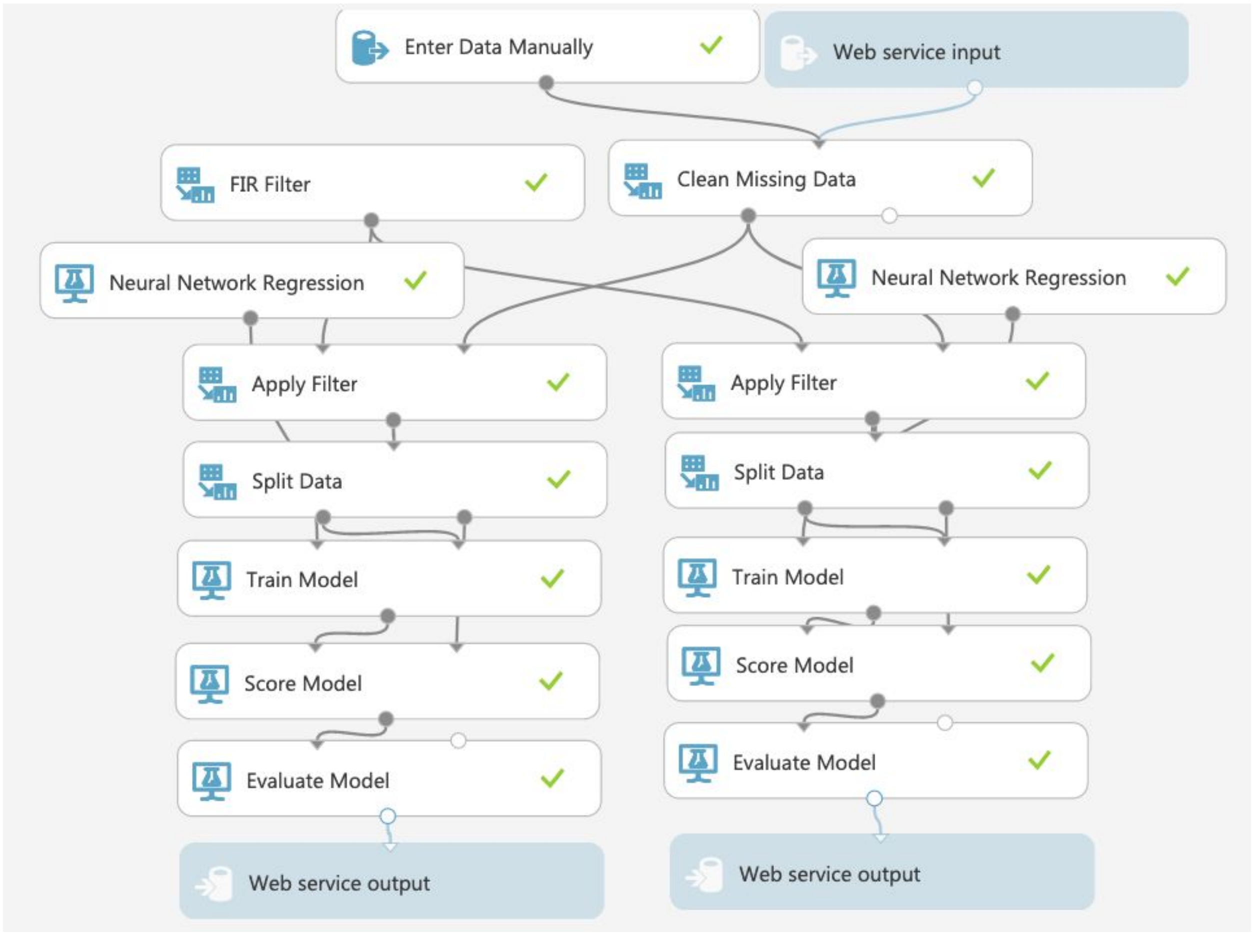


Figure 8: ML Data Flow

The model is initially trained after ~2 weeks of sleeping data, after which new data is fed in to retrain the model. These models are based on Azure's implementation of a neural network, with 100 hidden nodes, and a learning rate of 0.005. Each day, the day of the week is fed into the model and from that, the predicted start and end time is returned. Azure allows us to deploy these models as retraining and predictive endpoints, which can be called with the day of the week.

3.2.7 User Interface Design

Designing a professional level UI in Android Studio is not always convenient, but providing the user with a top-notch user experience was a high priority for the application. In order to create a professional looking product we went through a few different rounds of user interface research, balancing different references as ideas, until ultimately defining the look that we wanted for our application. This required coming up with a robust, bright color scheme and organizing the best way to layout our features. We designed an application interface organized into 5 main tabs, with a home tab fragment including 3 separate pages. This design allows for the user to easily navigate the application, but also looks clean and organized. The diagram below outlines the different application fragments and the major components contained inside.

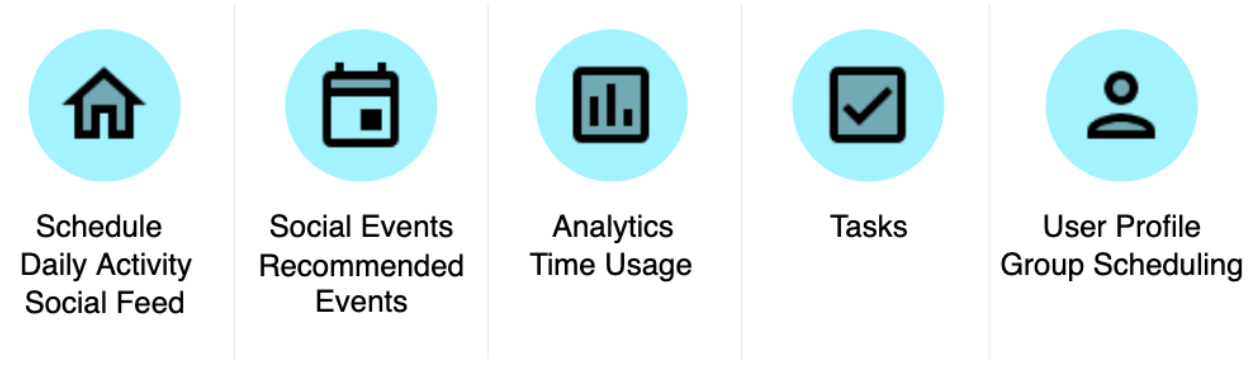


Figure 9: UI Pages

4. Implementation

Although we accomplished a small amount of our implementation goals during the first project semester, we left the most important features to be created in the second semester. To start, we devised a simple implementation protocol to assist us in staying organized and communicating well. We also needed to prioritize our main features to decide what was most important to work on. Although we had to deal with goal and design changes periodically, we still built a near professional-quality application.

4.1 Software Implementation Plan

Below is a simple process diagram outlining our implementation protocol process throughout our time working on this project. We started by gathering requirements mainly from internal team brainstorming. After knowing our requirements, we designed some simple screenshots to help us get a clear idea on what we needed to build from a user interface perspective. Next, we started the process of designing strategies to build specific features. After completion, the design was then explained to the full team. If anyone has a problem with the design or alternative ideas, they voiced their opinion and sent the designer back to the drawing board with new ideas. If the team approves the feature design, the designer continues to implement the feature. Following this, a quick code review is conducted. If this is approved by all reviewers, the feature is then merged and automatically integration tested through our Jenkins server. If something goes wrong, the team is notified by Jenkins and the code is manually reverted.

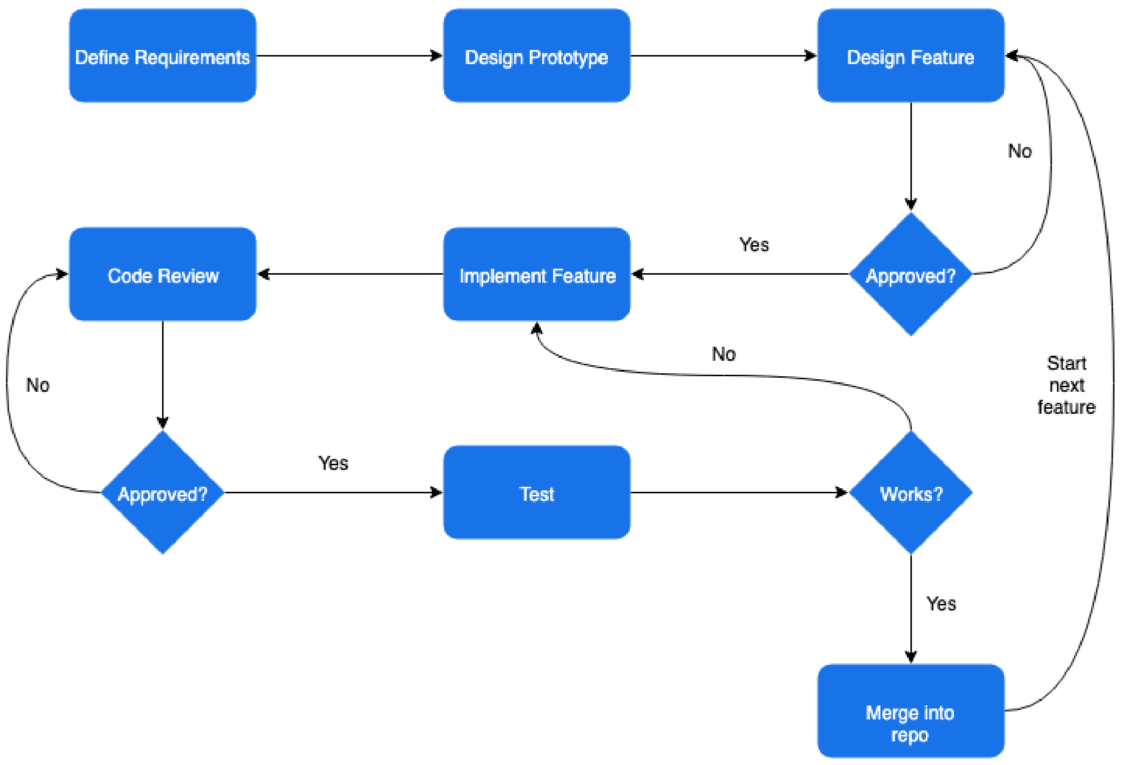


Figure 10: Implementation Plan

4.2 Major Features & Implementation Strategy

During the very first days working on GoMe, we had nearly unlimited ideas and no clear direction. We had determined that we wanted to create a productivity application that beat out existing methods, but suffered a difficult process of narrowing down our ideas into something feasible and effective. Ultimately, we decided to go for making a productivity centered social-media application. The main selling point was the capability of building a dynamic schedule. To build around that, we created common productivity features such as task management, time analytics and project tracking, but also includes social features like collaboration, push notifications and activity recommendations.

4.2.1 Dynamic Schedule

The schedule resolves uncertainty a user may have about what they should do throughout the day. The data provided comes from the individual's data profile as well as information from what is going on in their area (other users activity, traffic, resources available to users, etc.) to ensure they are as efficient as possible with their time. If something happens that impacts the user's ideal schedule, their schedule will update. For example, if the application notices that the user received less sleep than usual the previous night, it will suggest going to bed earlier on a future night in order to compensate for this. The user's schedule will also change based on how the user performed throughout the day. This schedule will be created by tracking the user's location and input where they've been or what they've been doing. It will display to the user what they actually did and adjust their future schedule as needed. In order to accomplish this dynamic schedule, we used our scheduling algorithm detailed in section 3.2.4.

4.2.2 Collaborative Schedule

Collaboration plays a huge role in productivity and well being, so it only made sense for us to incorporate collaborative features within GoMe. Within our application, every user has the potential to impact another person's schedule based on certain events that occur throughout the day. For example, if there are several users with an activity scheduled together and one of the people is running late to the activity, the app will recognize this. After it is recognized that someone is running late, the application then allows the user who created the activity to push the activity's start time back to the estimated arrival time of the person arriving late. This process of notifying everyone involved and rescheduling an event would usually take several text messages or phone calls and a lot of headaches. With GoMe, it is seamless.

Another feature within GoMe that allows users to collaborate is our task creation process. When a user is creating a task, they are given the option to add other people to the task. Doing so will add that task to both of the user's schedules so they can work on it together and see how it is progressing. Once one of the people working on the task is able to finish it, the other people are notified and it is removed from their schedule.

4.2.3 Social Media Features

GoMe not only creates a dynamic schedule for you, but it also has a social media platform built around that schedule. This platform strives to allow you to collaborate with your friends, watch each other succeed, and improve yourself. To do this, we allow users to send friend requests to one another. When the other accepts that friend request, they are now “GoMe friends.” Once users are friends on the application, they are able to see events that the other person has posted and schedule tasks together. Beyond that, the application is able to look at both of the user’s schedules and find activities that they both could do together during times that they are free. Once the users decide on an activity that they both want to do together, the app adds it to their schedules for them. Allowing users to easily find fun and new things to do together will lead to more people getting out and doing things with the people they like to spend time with. Another social media feature we have implemented in this application is allowing the users to create their own profile, which allows them to personalize how they are seen by others on the application.

4.2.4 Analytics, Progress Tracking and Recap

To allow users to continue improving themselves and their daily routine, it is important they understand how they’re performing in the different areas of their life. Within GoMe, users are able to stay updated on their performance via robust analytics and progress tracking in 4 different categories: sleep, work, social, and wellness. As described in Section 3.1.2, there is an Elo score for each user in these categories. The goal of “scoring” a user is to show them how they are spending their time, and what areas of their life that they could improve upon. A user will be notified of impending raises or drops in their Elo score. Another easy way the app demonstrates this information to the user is by using an Android graphing library to visually display the user’s scores. The user’s Progress Page displays a recap for each category as well, providing the user with a quick real time analysis of how they have been performing during the current day. With these functions in the app, a user can see their performance in long and short term time periods, making it easy to decide on any improvements that can be made to their routine.

4.2.5 Push Notifications

Notifications will be sent to the user whenever something important and/or urgent comes up. Notifications can be turned on and off as desired. The type of notifications consist of the following:

- Friend requests from another user
- Activity updates, telling the event owner that a user included in the activity will be late or miss the event entirely
- Activity invites from another user
- Task completion by another user (collaborative features)
- If the user is late to an activity by an hour, ask if they are still going to noted activity. If not, update their schedule

To facilitate the pushing of notifications to those who are in the same activity, we implemented a Firebase Function API. This is a simple node express API that allows notifications to be sent to users attending a specific event. When a user joins an event, they automatically get subscribed to notifications from that event. As an example of how this is used, consider the case that another user is going to be late to that event. In this case, a push notification will be generated by that user, and then pushed to all other participants.

4.2.6 Recommendations

GoMe recognizes user activity and makes recommendations based on patterns and user interests. The “Events” page recommends activities in your area that align with the interests you selected during the onboarding process. These interests are customizable on your profile page as well. As public activities are created by other users, they will appear on your page with the option to join the activity, which will add it to your schedule and allow you to receive its notifications. GoMe can also recommend activities for you and a friend to attend by looking at each user’s schedule and finding open free time where both users are available. With remaining free time in your schedule, the app analyzes your scores and suggests what else you should be doing during your free time in order to improve your scores and accomplish tasks.

5. Testing

As with any software system, testing is the greatest way to verify that your software is of professional quality. Being slightly short on time, we did not accomplish 100% complete test coverage of GoMe, although we believe that we came close. Instead of our original plan, leaning heavily on unit testing with fake user data, we decided to go a different route. Here, we outline our testing methods by using Jenkins continuous integration as well as our own personally built simulation environment.

5.1 Functional Testing

Throughout the development process, we utilized an agile-driven design while demonstrating new features to our advisors/clients as they are developed and tested. Each iteration addressed different goals and requirements that are listed in this document.

When a new functionality was developed and added to the code repository, it contained detailed scenarios on how and when the new feature is used. These generated scenarios are used frequently throughout testing. For testing, we consistently used the tried-and-true 'Given-When-Then' scenario model. An example of a possible scenario could be:

Given an authenticated user on the Events page

When user selects an event displayed to them

Then the event description and information is displayed for the user

This structured scenario process allowed us to brainstorm and user test possible scenarios for a new feature with ease. Before each push of a new feature, developers would consider scenarios in this model to test and debug on their local machine.

Below is a table of our carried out test plans for many functional requirements:

Table 1: Testing Plans for Functional Requirements

Functional Requirement	**Test Plan**	**Results**
System shall create a dynamic schedule for the user to follow.	Generated a schedule for the user and verified that the ideal schedule is correct based on events (like work school and sleep).	The system generated a sensible schedule for the user based on their previous habits.
The user shall be able to easily add events to their schedule.	Tested saving an event where free time is available and verify that it appears in the user's schedule for the day.	The saved event appears in the user's schedule for the day which the event was created for.

The user shall be able to share their schedule/events with friends inside the application.	Tested share functionality using an unchangeable schedule object with a user's friend.	The user's friend was able to view the user's schedule and shared events using the share feature.
The user shall be able to easily make changes to their daily schedule.	Tested the 'edit schedule' function allowing the user to adjust details about the events on their schedule.	The schedule reflected the updated event details (like timespan) and adjust the user's level of time allocation and free time around the scheduled event.
The user shall be able to see their profile and their friends profiles easily.	Tested navigation of the application for such cases.	Clicked on a profile name or picture whom is not the current user and were linked to the intended user's profile page.
The user shall be able to see changes to their 'Elo' score(s).	Tested Elo change due to user action as well as visual displaying of changed scores.	The user's Elo score was changed due to poor sleep. This change was reflected on the Go page where the user can see his/her scores.

Integration Testing - Continuous Integration

Continuous Integration (CI) is a huge topic in the world of software engineering today, so we wanted to implement it into our project. We decided to use Jenkins for our integration testing needs. We were able to download, configure, and run a Jenkins instance locally on a team members desktop. Using plugins available for Jenkins, the job pulls the current master branch code from the project's repository, performs a number of tasks on the code, performs a debugged installation onto an emulated Android device, and then emails a team member the status of the build (success or failure) along with the build output. The types of tasks (78 total) that are completed during the build include but are not limited to:

- Debugging the Manifest, config files, resources, Google services, etc.
- Compiling the code base
- Debugging Android .apk installation

The job is configured to run once daily to monitor and maintain the health of the repository. It has been proven that CI makes developer's lives easier by integrating building and testing together, helps write higher quality software by checking on the health of the code frequently, and saves time in the development lifecycle by catching bugs they may or may not have gotten through without the integration testing.

Simulation Testing

Our simulation testing creates a fake schedule for as many users and days as we want. For each day generated, the simulation creates fake arrival times, deletions, and other objects or actions in the user's schedule and then runs the simulated day using that schedule to see how our application reacts to scenarios. This allows us to test infinite scenarios on our application's scheduling algorithm, Elo creation process, user time analysis, trend correction, and any other major feature of our application. This is extremely helpful because we now know that our application works in thousands of scenarios, and it would have been impossible to do this by manually testing this aspect of GoMe.

5.2 Non-Functional Testing

Our original plan was to hit all the major non-functional testing areas (performance, security, usability, and compatibility) mainly using the Firebase Test Lab. However, as discussed above, we decided to focus more on the integration testing side of things.

Performance Testing/Compatibility Testing

Because of our focus on integration testing, we moved away from the Firebase Test Lab to Jenkins, but still hold onto our wishes to do performance and compatibility testing in the future. We strongly believe that continuous integration had a longer lasting effect on our project lifecycle.

Security Testing

With this being a completely software project, we hold a lot of reliability in the services we are using. Much of our security testing was making sure our database configurations were set correctly, as well as ensuring each user only had access to their own information. We made sure to strongly test that last point once we built features that allowed users to see other users and their created events.

Usability Testing

Because our only actor will be the general population, we asked some of our colleagues and friends to quickly use GoMe and tell us what they thought we could improve on design wise (disclaimer: we did not allow anyone to create profiles within the application, as we wanted to avoid data recording for anyone other than us). We were able to do this usability testing earlier in the second semester when our application design was still raw. Much of the information we received was based on app navigation. After listening to feedback, we moved some of the page fragments around to make navigation of GoMe easier and more intuitive. Major UI upgrades were also made during the second semester to beautify the application based on the results of usability testing.

6. Closing Material

In conclusion, we have completely designed and developed a mobile application that seeks to improve the user’s life by creating a dynamic schedule and providing feedback. Initially, we had many ideas and features that we hoped to implement for our application, but realized we needed to narrow down the scope and focus on the major aspects that could separate us from existing productivity systems. By combining the benefits of existing applications with additional functionality like collaborative tasks, activity recommendations, dynamic updates, and social media capabilities, we believe we have successfully fulfilled the vision of the GoMe application. Going forward, we are excited to explore how GoMe can grow and become a staple in a user’s real life.

7. References

"Calendar Survey: 70% of Adults Rely on Digital Calendars." *ECAL*, 23 May 2018, ecal.com/70-percent-of-adults-rely-on-digital-calendar/.

"Choose a database: Cloud Firestore or Realtime Database | Firebase," *Google*. [Online]. Available: https://firebase.google.com/docs/firestore/rtdb-vs-firestore. [Accessed: 26-Mar-2019].

Kashyap, Vartika. "Top 19 Online To-Do List Apps To Stay Ahead in 2019." *ProofHub*, 27 Nov. 2019, www.proofhub.com/articles/to-do-list-apps.

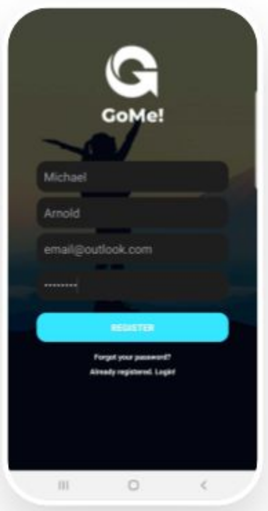
Appendix I: Operation Manual

Refer to this section to get started and learn more about what you can do with GoMe.

1. Getting started

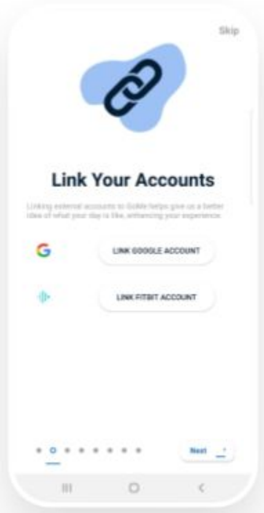
Step 1: Register

Simply enter your name, email and password to get started



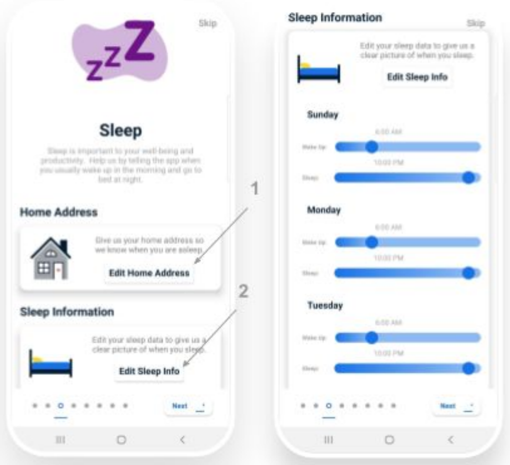
Step 2: Link Accounts (Optional)

Connect to your Google and Fitbit account to help us learn more about you and pull in your Google Calendar events.



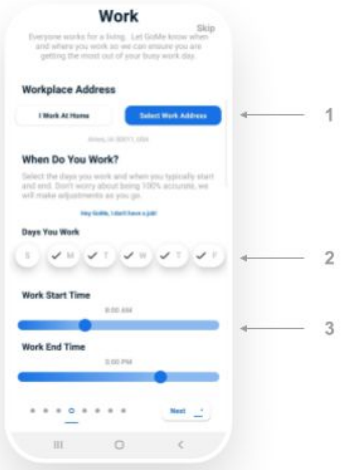
Step 3: Enter Home Address & Sleep Info

Start by choosing your home address with button **1**. This helps us understand when you are at home. Next, edit your sleep info with button **2** then adjust the time sliders.



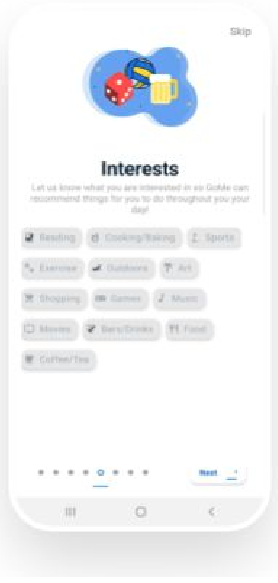
Step 4: Enter Work Information

Select your work address with button **1**. This helps us understand when you are working. Next, select the days you typically work in section **2**. Finally, adjust the sliders in section **3** to reflect when you start and end your day at work.



Step 5: Select Your Interests

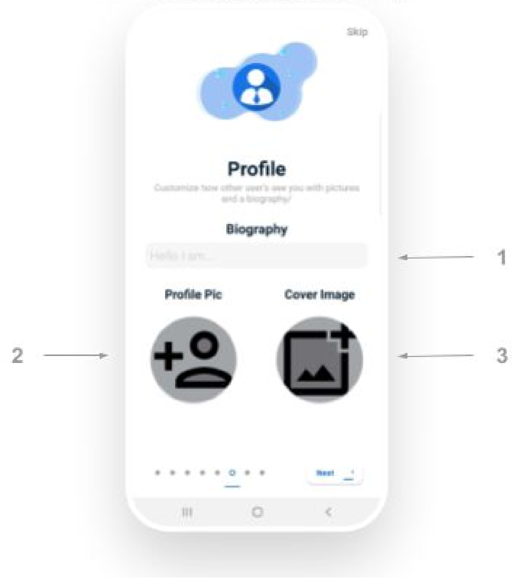
Select your hobbies and interests so we can accurately recommend social activities to you. Simply tap on each chip that you are interested in seeing events about.



Let us know what you are interested in so GoMe can recommend things for you to do throughout you your day!

Step 6: Customize Your Profile (Optional)

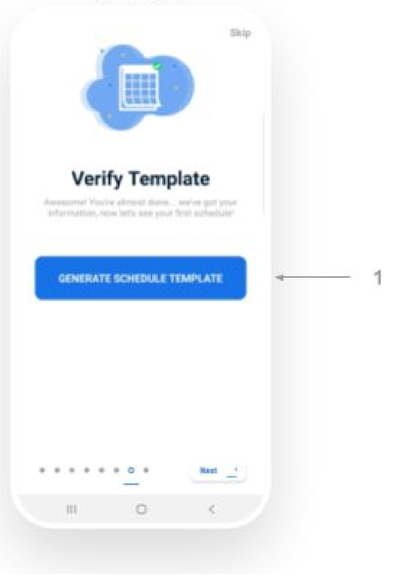
Start customizing your profile by adding a description about yourself in section **1**. After that, upload a profile and cover picture for you page to help your recognition. Simply tap in section **2** to upload a profile picture and in section **3** to upload a cover image.



Customize how other user's see you with pictures and a biography!

Step 7: Create First Schedule

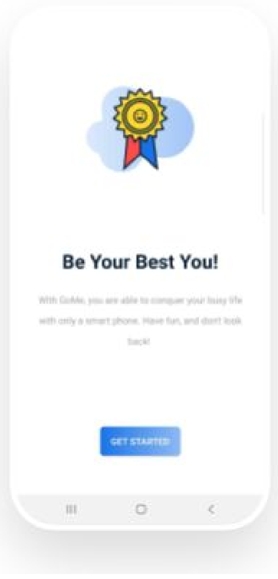
In this step, you will be generating your first schedule in GoMe. By pressing the 'Generate Schedule Template' button in section **1**, the app will create a simple schedule for you, taking into account the sleep and work times you input.



Awesome! You're almost done... we've got your information, now let's see your first schedule!

Step 8: Done!

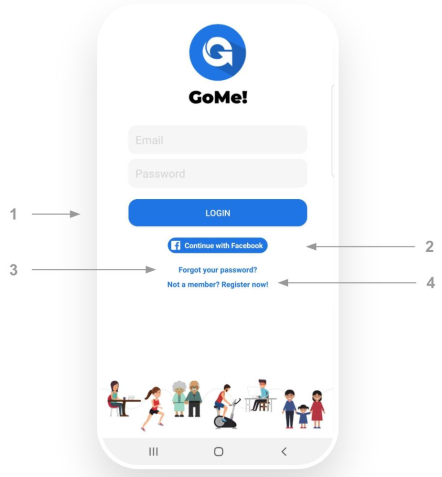
Congrats! You have officially completed the onboarding process. You are now ready to get started with GoMe! Simply tap the 'Get Started' button to continue.



With GoMe, you are able to conquer your busy life with only a smart phone. Have fun, and don't look back!

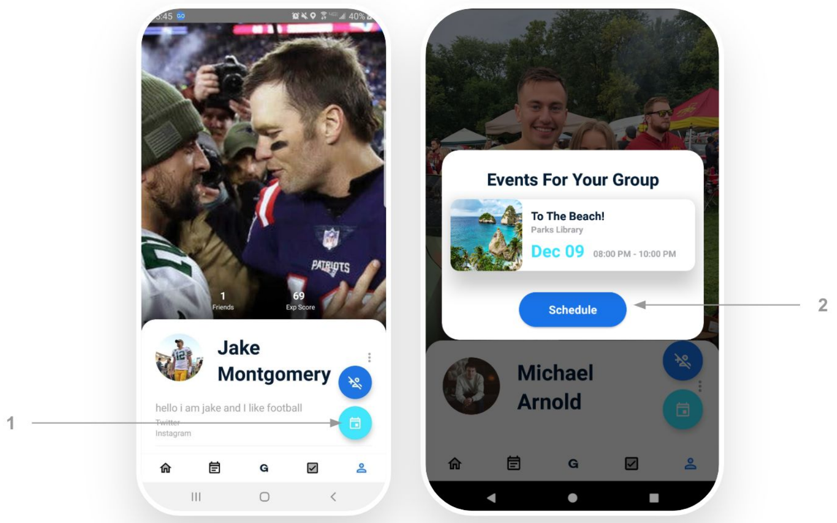
2. Signing In

Logging in to GoMe is extremely simple! All that is required is for you to enter your email and password, then tap button **1**. Note that this will only work if you have created an account before on GoMe. If you have not created an account before, press button **4** to continue to the registration page. If you happen to forget your password, simply tap button **3** to receive an email regarding password reset. Lastly, GoMe also provides the option to sign in with your Facebook account. This can be performed by pressing button **2**.

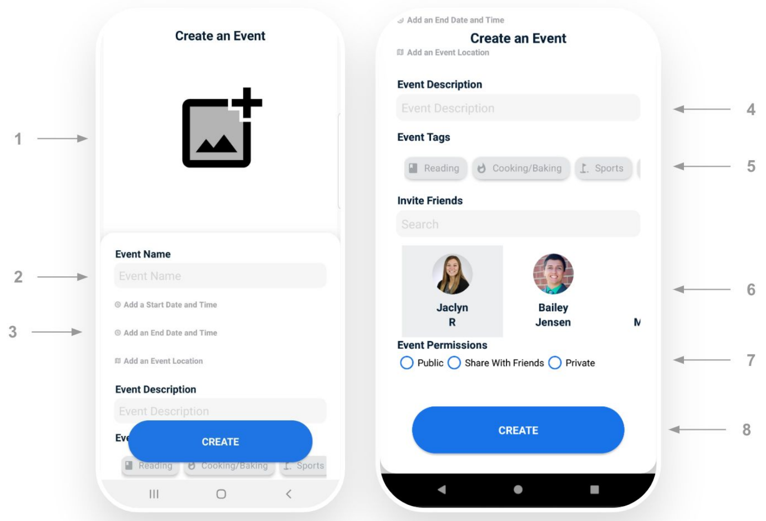


3. Scheduling Activities With Other Users

GoMe includes a unique feature of allowing you to quickly schedule an event with another user. This is so simple that it can be done in the push of a button! To start, navigate to another user's profile page. From there, click the calendar icon button **1**. This will automatically find your availability and match it to events. You will shortly see a dialog with a list of events that you can scroll through horizontally. When you decide which activity you want to schedule, simply tap the blue button (**2**) and you will be added to the activity, while an invite will be sent to your friend.

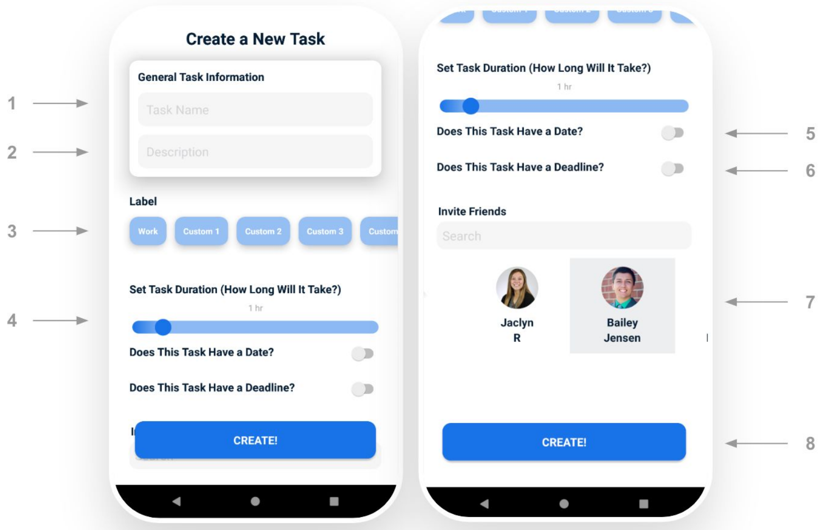


4. Creating an Activity



Creating an activity is a quick yet robust process allowing any user to create a detailed Activity. Doing this is very simple and takes about 1 minute. To start, tap on the image placeholder **1** to upload an image for your event. Next, give your activity a name by typing into input **2**. Third, provide a start and end time as well as a location by using the provided time and place pickers in section **3**. Moving forward, type out a short description of your event in input **4**. Next, select any relevant tags from the options in section **5**. Continuing, select any users that you want to invite to your event in section **6**. Any user selected will receive a notification once the event is created. Almost finishing, set the permission value of your activity by selecting an option in section **7**. Selecting ‘public’ will allow any user to see your event. Selecting ‘private’ will only allow you to see the event. Lastly, selecting ‘Share with friends’ will only let users that you are friends with view your activity. Finally, you are done! Tap the create button (**8**) to finish officially create your new event.

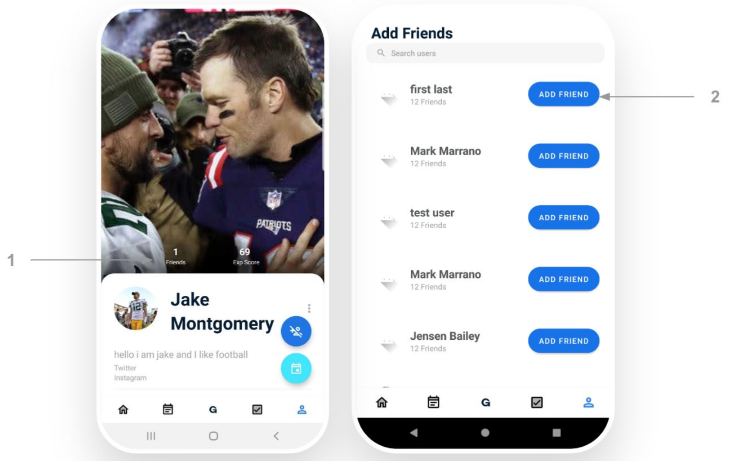
5. Creating a Task



Creating a task is an easy process allowing any user to create a detailed task. Doing this is very simple and takes about 1 minute. To begin, give you task a name in input **1**. Following that, write a short description about your task in input **2**. Next, choose a label in section **3**. This label is intended to categorize your tasks. For example, if the tasks relates to something at work, label the task as 'Work'. Next, adjust the slider in section **4** to give an estimate of how long the task will take. This is important, as it helps the app determine how much time to schedule to allow you to complete your task. Moving forward, switch on the date option in section **5**. If you want to give your task a start time and location. This is optional. In section **6**, turn the switch on to give your task a deadline. This is also optional. In section **7**, select any friends that you want to invite to work on the task with you. Once created, invites will be sent to all users selected. Finally, press the create button (**8**) to officially create your new task!

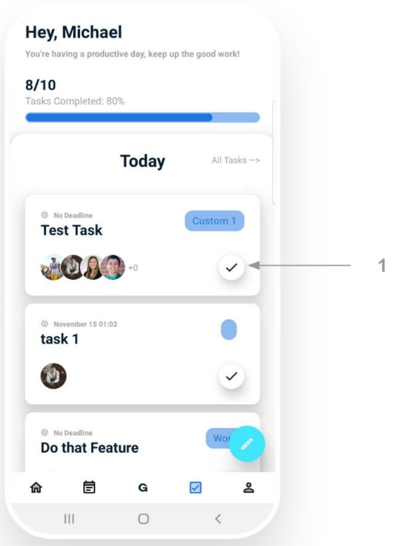
6. Add Friends

Adding friends is an easy, minimal-step process. From your profile page, click on the textbox showing the amount of friends you have (**1**). You will now be directed to a new screen showing all of the users that you can add as a friend. Simply click the blue button (**2**) next to the user that you want to add as a friend to send them a request. When they choose to accept, you will then be friends.



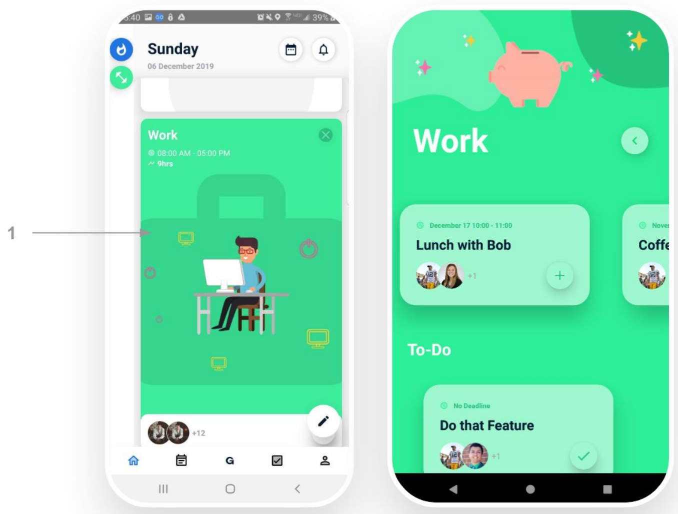
7. Completing Tasks

Completing tasks may be difficult in real life, but in GoMe it is extremely easy! From the task page under the checkmark tab, you can view any tasks that you have created in an organized list. If you don't see any tasks, try creating one first (see section 5). When you have finished a task and want to check it off on GoMe, simply tap the checkmark button (**1**) on the completed task and watch it disappear forever.



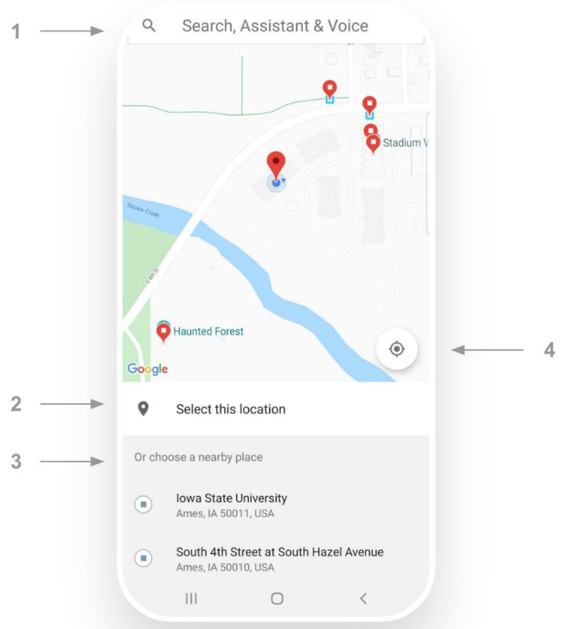
8. Viewing Categorized Tasks

GoMe provides a convenient system of labeling tasks, allowing you to see your tasks according to their label. As a useful example, imagine that you wanted to see all of the meetings and tasks that were on your plate for your day at work? This is possible with GoMe, and it's very simple. First, navigate to your schedule under the home tab. Find your event today and tap on it (**1**). Once that is done, that's it, you will be swiftly redirected to a page matching your event that lists all of the tasks that you need to complete.



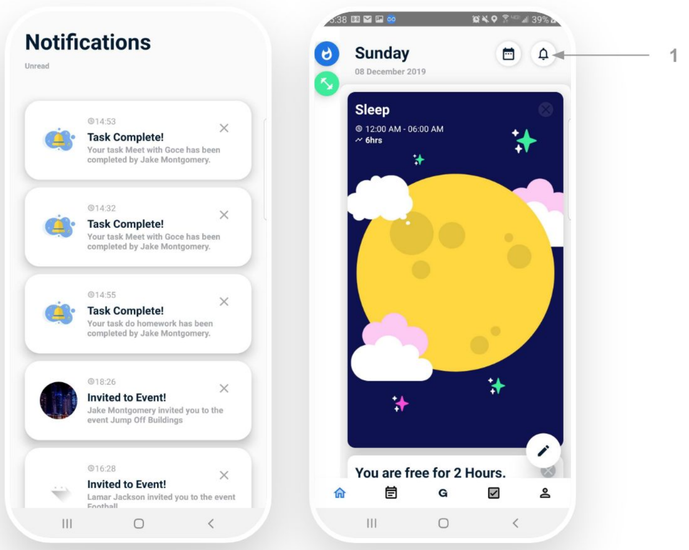
9. Using the Location Picker

Across the app, you will have the option to choose a location for different tasks and activities. This requires using Google's location picker. This widget is extremely helpful and easy to use. Getting started, search for the location that you want in input **1**, or select a commonly chosen place in section **3**. You can also select your current location with the pinpoint button (**4**). Once you have selected your location of choice, click the button in section **2** to complete the location pick.



10. Viewing Notifications

GoMe includes an extensive feature that notifies you about different events. To view these notifications, first navigate to the schedule page under the home tab. From here, click on the notification icon button in the top right corner (**1**). This will automatically redirect you to the notification list where you can view and interact with your notifications.



14:53
Task Complete!
Your task Meet with Goce has been completed by Jake Montgomery.

14:32
Task Complete!
Your task Meet with Goce has been completed by Jake Montgomery.

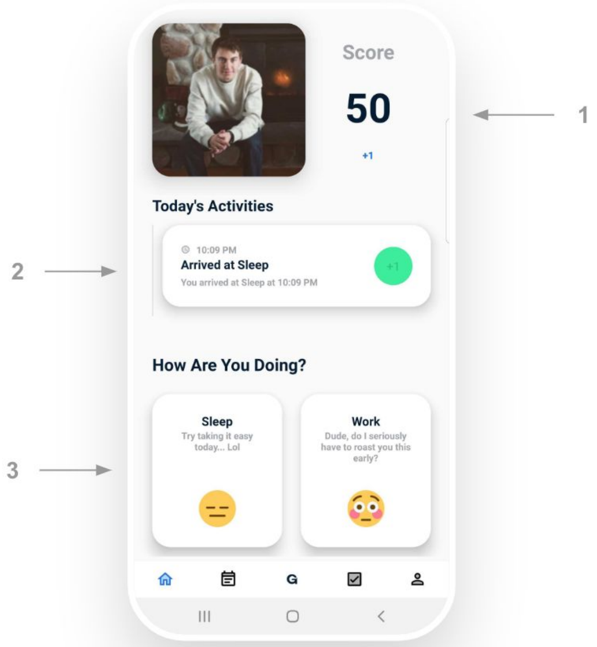
14:55
Task Complete!
Your task do homework has been completed by Jake Montgomery.

18:26
Invited to Event!
Jake Montgomery invited you to the event Jump Off Buildings

16:28
Invited to Event!
Lamar Jackson invited you to the event Football

11. Viewing Your Daily Activity

GoMe keeps track of your daily activities and scores you based on punctuality and time spent. This is displayed to the user on the progress page under the home tab. On this page, you can view your Elo score and recent changes in the top right (**1**). Below your picture in section **2**, you can see a list of your daily activity. This outlines when you leave places, when you arrive at places and other events. Finally, in section **3** you can view how you are doing in the categories of sleep, work, social, and wellness.



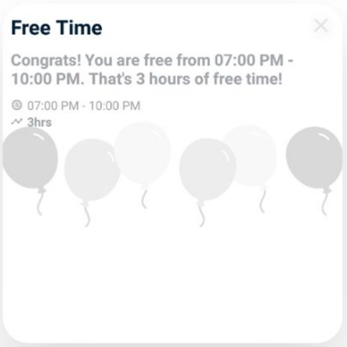
10:09 PM
Arrived at Sleep
You arrived at Sleep at 10:09 PM

Sleep
Try taking it easy today... Lol

Work
Dude, do I seriously have to roast you this early?

12. Interacting With Your Schedule

In case you didn't already know, GoMe includes the powerful feature of a dynamic schedule (!). Although this is an operation manual, the best benefit of a dynamic schedule is that you don't need to know how to use it. Instead, all the work is done for you! The schedule works by taking into account when you usually sleep and work, your scheduled activities for the day and also gives you live recommendations to help you make balanced time decisions. If this still doesn't make sense, open the app and tap on the home tab. This will reveal a list of different items that are on your calendar for the day. There are many different types of cards that can show up in your schedule list. They are as follows…



Free Time

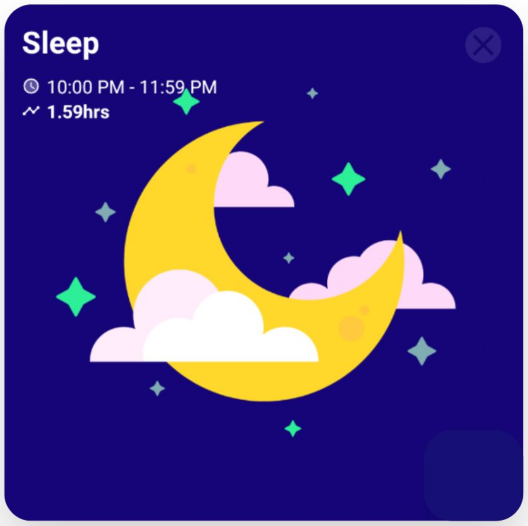
Congrats! You are free from 07:00 PM - 10:00 PM. That's 3 hours of free time!

07:00 PM - 10:00 PM

3hrs

Free Time

A Free Time card represents time that you don't have any responsibilities. You can't really interact with your free, but instead, make sure to take advantage of it. Free Time will likely show in your schedule more than once per day, depending on how busy you are.



Sleep

10:00 PM - 11:59 PM

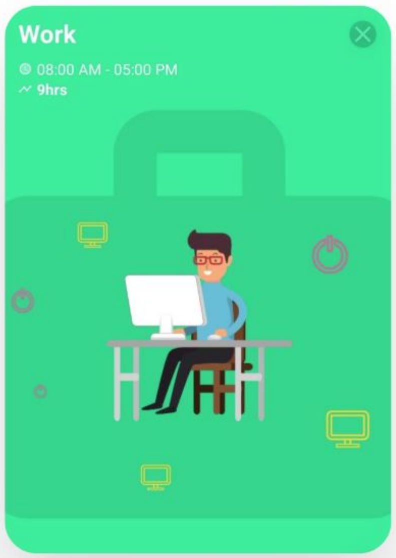
1.59hrs

Sleep

A Sleep card shows up on your schedule when the app thinks that you should be sleeping. This is mostly determined by your input during the onboarding process, but over time GoMe will try to reflect more accurate sleep times as you go throughout your life. You will typically see 2 sleep cards in your schedule, the first representing you morning sleep after midnight, and the second holding a place for you night sleep prior to midnight.

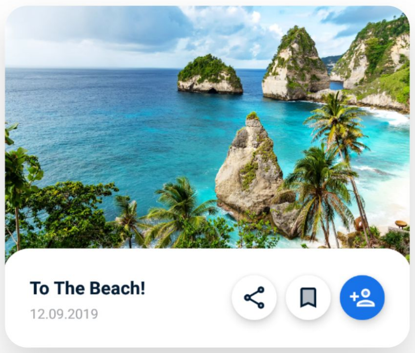
Work

A work card will show up on your schedule during a time that GoMe thinks you will be at work. Currently, this card is actually quite interactive. Try tapping the work card in your schedule to reveal a new page that shows all of your activities and tasks related to work. You will typically only see 1 card representing work in your schedule, since you hopefully only work for one extended period a day.



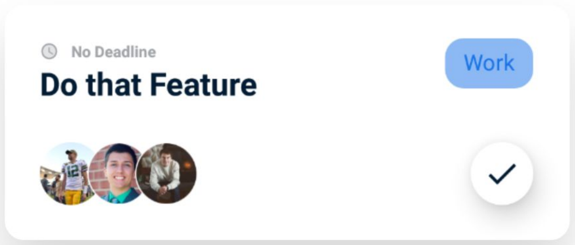
Social Activities

Every once in a while, you will likely find a fancy looking event card with a nice picture and a title. This card represents a social activity that you joined. If you for some reason run out of time in your day to attend this activity, simply take it out of your schedule manually, or, since this is a dynamic schedule, don't show up at all and your schedule automatically update your schedule based on your actions.



Tasks

Life is always keeping you busy. Because of this, you will often find yourself needing to allocate daily time towards working on certain tasks. When you create tasks in GoMe, they are automatically loaded into your schedule before they are due. In your schedule, you will see task card to reflect this. Task cards are also interactive.



When you have finished your task, simply check it off and watch it disappear forever.