GoMe

A Life Improvement App

Design Document

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

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: 03/26/2019
Updated: 04/23/2019

Table of Contents

List of Figures

List of Tables

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 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

Scheduling applications currently available to users around the world have static layouts which require them to input their information and edit it as needed. Users manage their own time and make adjustments where they see fit. GoMe is a mobile application with the primary goal of making scheduling easier for users. GoMe adjusts a user's schedule dynamically while helping them identify areas where improvements are needed in order to reach their goals. The application monitors what a user has done throughout the day by asking them to input a tentative schedule of activities for their week, then passively reading user information to determine if they are on schedule or not. This input comes from various activities and tasks the user has done throughout the day such as sleeping, going to work, and/or attending social events.

Using data gathered from the user's everyday activities, GoMe adjusts the user's schedule to react to changes in events. Whether this be a work meeting running long, waking up later than scheduled, or staying late at the office, GoMe adjusts scheduled times accordingly. The application provides an adjustable and flexible schedule for the user, provides feedback on how they spent their time, and builds an ideal schedule for them to meet their goals. Our application's objective is to motivate users with this information so they can develop a balanced lifestyle by optimizing their time between tasks, resting, and pleasure.

1.3 Use Case Diagram

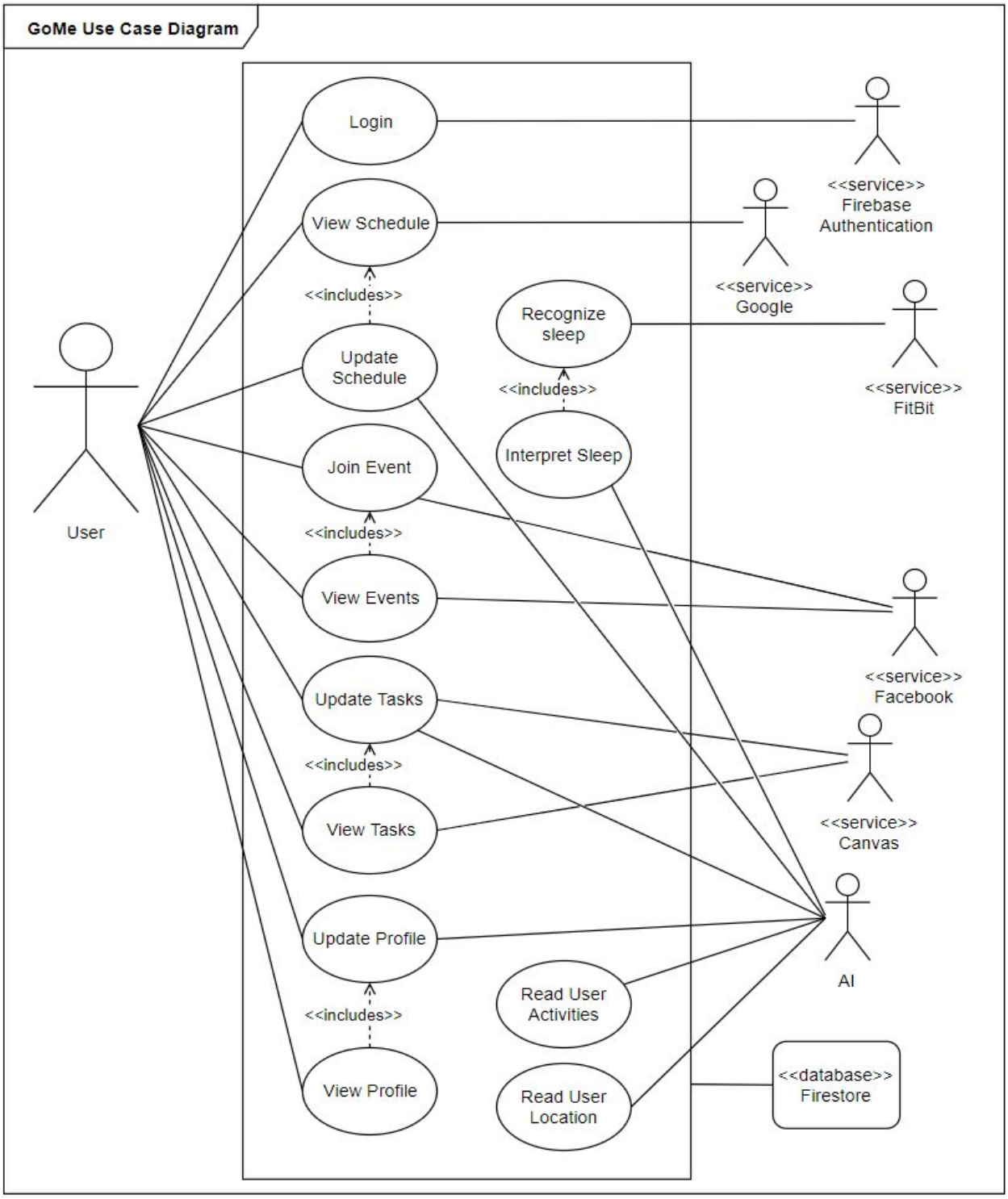

Figure 1: Use Case Diagram

This is our use case diagram for our application. It fully demonstrates the use case ideas, services and actors for the application. For graphic simplicity, Firestore connections are are grouped into one for the use case diagram.

1.4 Operating Environment

The operating environment of GoMe will be from a mobile device. Our application will be built on Firebase which will provide authentication, databases, and file storage. Due to the necessity of location tracking to determine users' activity, it is intended that this application is active on the user's mobile device even when not in use.

1.5 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, view and join events, view and update tasks, and create or update their profile. Examples of more specific use cases would be determining when to leave for work, choosing an optimal time to go to sleep, or determining if the user has time after work to attend a certain social event.

1.6 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 team will be able to obtain sufficient test data

Limitations

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

1.7 Expected End Product and Deliverables

This project is open-ended, but using our defined use cases described in Figure 1, we expect our end product to be an Android application with the following features:

1) Schedule

The schedule resolves uncertainty a user may have about what they should do throughout the day. The data provided will come 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 user, 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, in order to compensate this it will suggest going to bed earlier on a future night.

The user's schedule will also change to reflect 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.

Use cases for schedule revolving around work/school, sleep, and events:

- When user works longer than usual, do not look to schedule a social event.
- If the user has been less social this week and has some free time, recommend to schedule an event for the user to attend.
- If the user works late and has a lot of tasks to do tomorrow, recommend an earlier bedtime.
- If it's the weekend and the user typically doesn't work the weekends, look for social events that the user might like to do.

2) Personalized Feed

The goal of the feed in GoMe is to present information that the user might be interested in. The ML algorithm will determine events going on that the user would like to know about, and put those at the top of the user's feed. The feed will also display important updates from around the user's location so they will know about events that might affect their day. The user's friend's achievements and accomplishments will be highlighted in this feed.

3) Time Breakdown

The user's profile will display a report of how the user spends their time. Categories will be the following: sleep, social, work/school. The user will have the ability to sort by day, week, month, or year.

4) Motivation

The app will work to motivate the user in several ways:

A. Experience Points (XP)
    a. Users will be rewarded for doing things like accomplishing tasks, going to events, helping friends, how productive they were throughout the day, etc. with points
    b. When the user obtains certain amounts of XP, they will "level up".
    c. Rewards, statuses (like professional or expert), posts celebrating achievement, leaderboards, and bragging rights are all incentives to leveling up.
B. Motivational Messages
    a. In the personalized feed or through the use of notifications, the user will receive fun, motivational messages to provide encouragement.
C. Goals
    a. The app provides the ability to create a list of goals to accomplish throughout the day (user would also gain XP for completed goals).

5) Recommendations

If GoMe notices something that the user could correct, it will notify the user so they can make the adjustment and help them reach a goal, schedule an item, or complete a task.

6) 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:

- When to leave for an event
- If user hasn't left for an event on time
- Changes to the schedule
- Suggest an event the user might like and ask if they should schedule it for them
- Time to go to bed
- 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.

7) Activity Page

A page listing available activities in the user's area with an easy option to see details about each event and add an event to the user's schedule.

8) Tasks Page

The user can create a list of tasks they are hoping to accomplish and associate those tasks with their schedule accordingly. The application will be used to optimize the user's schedule to complete these tasks on time.

9) Profile Page

The user will have their own page dedication to information about themselves. Here they will be able to see data about themselves as well as change their preferences (including security allowances).

Over the next year, we will also have many reports and presentations regarding GoMe. Each of these reports and presentations will be solely created by the 6 team members of our group likely with advice from our faculty advisor.

2. Specifications and Analysis

2.1 Functional Requirements

- A mobile application that does the following:
    - Creates a dynamic schedule for the user to follow
    - Allows user to easily add events to their schedule
    - Allows the user to share their schedule/events with friends
    - Allows users to make changes to their schedule easily
    - Allows users to see their profile and friend's profiles easily
    - Gives the users alerts based on their schedule
    - Provides a social media platform for the user
    - Allows user to "rank up" based on achievements and being productive/living a healthy lifestyle
- A machine learning algorithm that does the following:
    - Understands the users historical sleep data and understands when they will go to bed and wake up that day
    - Understands when the user should work each day
    - Recommends events
    - Adjusts schedule based on user actions
- A database that contains the following:
    - Sleep Data
    - Location Data
    - User Profile Data
    - Event Data
    - Messages

2.2 Non-Functional Requirements

- Visually appealing user interface (UI)
- The user should be able to navigate the UI in an intuitive way.
- The application must have fun and interactive features
- User should not be able to see other user's schedules, unless it is shared with them
- The user should be able to login through an existing account or using Facebook authentication in less than 15 seconds
- System should be scalable to support a growing number of users
- The ML algorithm should predict sleep & work patterns with an accuracy of 80%
- Location and any other private data stored can only be accessed by the user with the proper authentication token.

2.3 Standards

We will be using the following standards for our project:

- Version Control
    - Git will be used as the primary means of version control for all project code.
    - The Google software suite (Docs, Sheets, etc.) will be used for all formatted documentation, such as planning and design documents.
- Unit Test Coverage
    - We will ensure all code that is requested to merge into master will have, at the minimum, 85% code coverage of all lines and all branches.
- Code Review
    - The master branch will have no push access to anyone, and development will be done on branches.
    - When a branch is ready to be merged, the author will assign one or more of other project members as reviewers.
        - Reviewers must ensure the code:
            - Correctly implements the desired functionality
            - Contains enough tests and branch coverage (85%)
            - Tests pass and are free of errors
            - No conflicts when merging with existing software
    - Reviewers will communicate code issues on the merge request, and the author is responsible for addressing all issues.
    - Once code has no issues, it can be merged to master.
- Collaboration Tool/Task Management
    - Trello will be our tool of choice for organizing tasks into different backlogs.
    - Trello will help us understand who is working on different assignments, what needs to be worked on, and what is already completed.

2.4 Proposed Design

There are many things to consider when thinking about all of the technologies required to create an app like GoMe. The main things we will be focusing on are the technologies that will make the user experience the best it can possibly be along with the technologies that will help us gather the most accurate and important information. With the data that GoMe collects, we will be able to create a data model to feed our ML model and help it learn more about the user and relay useful information to the user. We will be using the following technologies to do this:

2.4.1 Android Application

We will be building a mobile application on the Android Platform. Android is the best choice for the initial release due to the fact that using Android will allow us to target 54.2% of all smartphone users (statista.com). It also works well with Tensorflow, Google Firebase and other Google APIs that we will be using throughout the development of our application.

2.4.2 Machine Learning Framework

Our ML framework will be built using TensorFlow via the Keras API to model the user's schedule. We chose TensorFlow because it has the best compatibility with google services, as well as good integration with FireStore. In addition, tensorflow has a lot of libraries to support development on android, and to set up the proper models for our app to use.

2.4.3 Database

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, what the user needs to do and then feed this information to the user and the ML algorithm in efficient and meaningful ways. To do this we will be using a Google Firebase Database.

Google Firebase offers two types of storage, a real-time database named Firebase RTDB (Realtime Database) and a new Google Cloud Firestore. According to the Google Firebase documentation, the RTDB, 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 exactly what we needed in a database, however then we read more into Google Firestore. Google Firestore is an upgrade to Firebase RTDB by Google's standards, as it is very similar but is also more technologically advanced and will be supported more frequently in the future. For these reasons, we decided to use Firestore instead of the RTDB. We are certain that Google Firestore is the best service for us and GoMe.

Why Use Firebase Cloud Firestore as a Database?

- Realtime Database
- Easy Authentication function
- Firebase Cloud Messaging allows for easy push notifications
- Integrated in Android Studio for easy use
    - Well written documentation
    - Both are Google services
        - Allows us to have a DB and server hosted by Google (high reliability)
- No SQL
- Security maintained by Google
- Good analytics features to help us understand how users are interacting with our app better
- More scalable than other database services on the market

As listed above, Google Cloud Firestore offers great features that saves us the tedious hassle of doing ourselves. This includes authentication, general data security, and scalability, all features that are vital to the long term success of our application. Like stated earlier we originally planned to build the app around Firebase RTDB, but decided that Cloud Firestore provided a more advanced, updated, and overall better service for our data. The next few paragraphs will describe some differences between the two Firebase database services, and why we favored Cloud Firestore for multiple reasons.

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.

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.

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 nearly any situation.

Data Model
First, the most important data we will need to be collecting is location. This will allow the app to monitor where a user is at any time, and from this, allow other aspects of the app to learn from this. It will allow our model to start learning the anticipated location of a person at specific times throughout the day. This will also allow our other APIs, such as our Facebook integration, to utilize this data. Secondly, gathering sleep data is important, as this would be another core functionality of our app in adaptive scheduling. Our goal is to allow for the integration of the FitBit activity tracker to gather sleep data. As a backup, we will allow for manual user input of sleep times as well. Next, we need to keep track of a user's personal preferences when it comes to certain events and tastes. To give the most relevant events possible for a user, we will need to know the likes and dislikes. After that, We will need to be able to get and store

pre-existing calendar events for a user. Doing so will give our model the events that have to be completed, so that it can accommodate and work around these events.
With all this data, the model should be able to understand these key points:

1. Understand where a person works, and their typical working hours for certain days.
2. Understand what the user needs to get done from pre-existing calendars, and what time is left free for other activities.
3. From free time calculated after all other events, find the most relevant event that lines up with the user's preferences, and then suggest a couple that would fit in their schedule. In addition, it should be able to prioritize certain tasks that are more important than others and update/remove existing low-priority events as well.
4. Understand the typical sleep schedule of the user, and from that, inform the user whether they are getting too little or just enough sleep.

After all this data is collected and showed to the user in a way that they can understand it, inform the user on how well they did in meeting their goals for the day, and any improvements they can do in order to improve their day-to-day living.

Input:
   a. Location Data
   b. Sleep Data
   c. Event scraping
Parsing:
   a. Real-time updates
      i. Events lasting longer/shorter than planned
Output:
   a. Optimized Dynamic Schedule

What will we need to store?

- Profile data
  - General user information
  - Auth credentials
  - Achievements and Goals
  - Experience Points
  - Social Posts
  - Time Breakdown
  - Past Events
  - Messages
  - Task Completion Rate
  - Where the user works
- Tasks
  - Canvas Tasks
  - Google Calendar/Outlook

        - User Created
        - When I Work Application or Similar Scheduling Apps
- Calendar Event
    - Google Calendar
    - Outlook
    - User
    - Facebook
    - Messages/Chat stream
        - Pictures/Gifs
        - Posts
        - Comments
    - Users
    - Time
    - Description
    - Title
    - Picture
    - Location
- Productive Events
    - Sleep
        - Time/Length
        - Quality
    - Work
        - Chat
        - Tasks
        - Time
        - Location
    - Class/School
        - Time
        - Location

How we will structure the data:
Having a properly structured data model is one of the most important things a neural net needs to be order to learn properly. Thus, one the main focuses pertaining to our data model, will be structuring it in a way that our machine learning algorithm can understand and derive accurate predictions from. The process we will follow is listed below (altexsoft.com):

1. Articulate the problem early
2. Establish data collection mechanisms
3. Format data to make it consistent
4. Reduce data
5. Complete data cleaning
6. Decompose data
7. Rescale data

8. Discretize data

Full details are at this link:
https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-better/

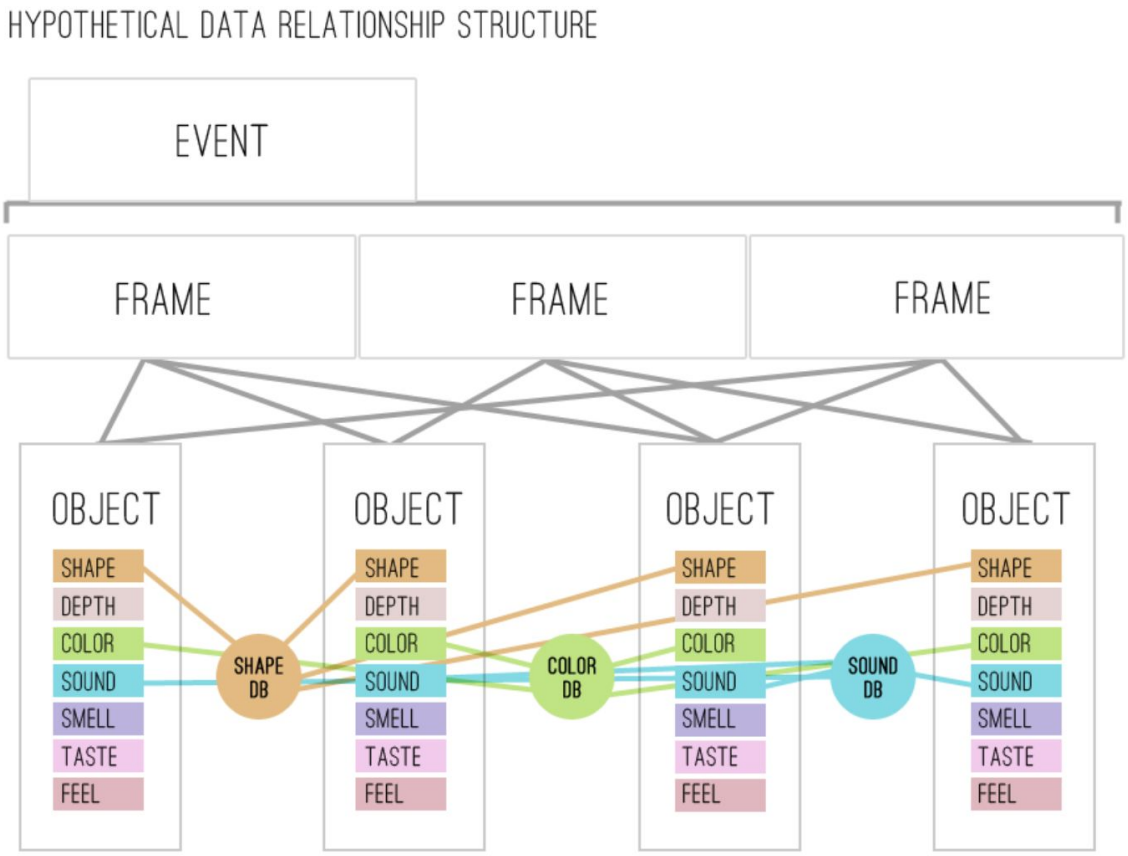

Figure 3: Example Data Relationship Structure- responsiblemachines.wordpress.com

2.4.4 APIs

To gather the information from our users necessary to build our data model, we will be implementing our data model using the following APIs:

- Fitbit - to capture sleep data
- Google Places - to get data about the locations the user is at
- Canvas - to get school tasks
- Google Calendar and Outlook - to get other pre existing user obligations
- Facebook Events - scrape facebook for events users can attend

When implementing these APIs we will be using the adapter interface to ensure that if the APIs change how they work, all we need to do is change the adapter and then the code will work the same way all throughout the application. It will also reduce coupling and make the app easier to manage.

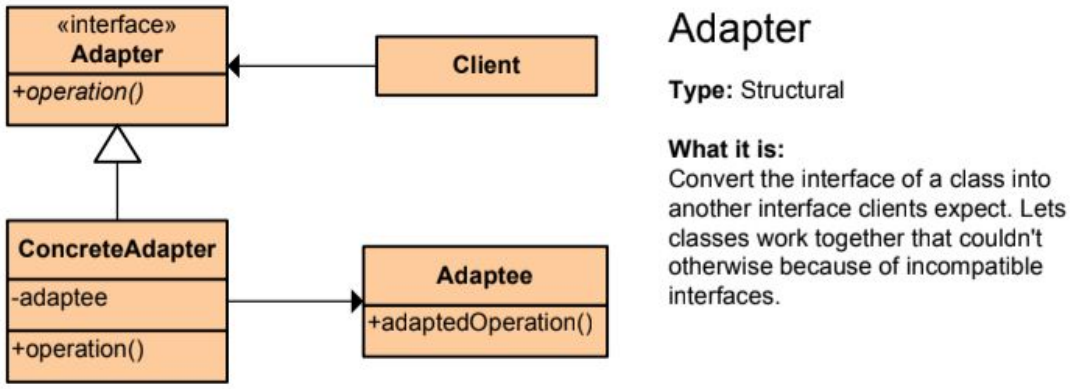

Figure 4: Adapter Interface Implementation

2.5 Design Analysis

So far we have began implementation on our Android Application with the following features:

- Login/Register
- Facebook Authentication
- Event Creation
- Task Creation
- Location Tracking -- and an understanding of the location the user is at through the use of Google Places API
- FItBit Sleep Tracking
- Google Calendar Integration
- Facebook Events

We have also began implementing our Machine learning algorithm, so far it has the ability to take in sleep data and understand the sleep patterns of the user.

So far our implementation has been going smoothly. We have split up into 3 groups of 2 (one for Machine Learning, Systems, and Data Analytics) and each group has been working to develop individual features that pertain to their subject. Our team has been able to implement the APIs necessary to create our intended data model for each user, create a neural net that understands sleep patterns and have designed a system that will be easily expanded and managed. While we have been developing, there has been a focus on building a project that will be scalable to both large amounts of data and features. We have also been ensuring that the code base is easy to read and has low coupling through the use of interfaces.

While our project continues to progress, we are starting to see a more coherent application starting to surface. However, we have not fully implemented coupling each of the individual

projects together. This is something that we need to put more emphasis on in the coming weeks as this project progresses.

Our decision to use technologies that work well together (Android, TensorFlow, Google APIs, Firebase) has seemed to paid off for us early on in the project. Not only does the cohesion of these technologies make things easier, but Google also provides very descriptive and easy to follow documentation. This has allowed us to progress quickly and allowed us to get a solid base of features created already.

One thing we may need to continuously modify as the project moves along is our expected end result. This is a difficult project and it has a lot of complex features, this coupled with the fact that we have a limited development period, means we may need to re-access our progress at the end of each of iteration.

Strengths:
One strength of our solution is the way nearly all of the technologies we have chosen to use interface well with one another. As mentioned above, we have chosen to implement this android app using tensorflow, firebase, and firestore. All of which are products maintained by Google that are meant to work well together and be extremely compatible. This will make the the implementation much easier and allow us to use the pre existing functionalities of each product to integrate them with one another.

Our development process has also been an area of strength so far. Once a week, we have been meeting with our faculty advisor to get advice and feedback on our progress. Our whole group then meets to discuss how we want to implement features and what research we need to do. Finally, we break up into our three groups of 2 and research our topics and implement the features we were tasked with completing. This process has been working very well and has allowed us to progress quickly while keeping everyone informed on the status of the project.

Weaknesses:
One weakness of our model is how we use predictive algorithms to decide on the schedule. There will be users who are very unpredictable, and therefore we will be unable to anticipate for all discrepancies in their normal schedule.

3. Testing and Implementation

3.1 Interface Specifications

GoMe will be a fully software Android application, where our only communication will be through servers and databases owned by the GoMe team, as well as certain approved APIs. Discussed previously, we will be utilizing Google Firebase services for our server and database, thus limiting the amount of services GoMe will need to interact with.

It is our plan to interface as many classes as we can. With this, our goal is to simplify the tasks and responsibilities each class holds. By giving classes single responsibilities to lower complexity, we will be decreasing coupling and increasing cohesion throughout our codebase. This plays a huge part when implementing design patterns (like the Adapter pattern).

To utilize all the APIs we need, we will require multiple interfaces. A problem we have for-seen is that, if an API updates and potentially breaks some our calls to the API, we would need to find every situation where we call that API and change the lower level code. To ‘adapt’ to this problem, we will be utilizing the Adapter pattern. The Adapter pattern will help lower coupling between GoMe and the APIs. This works by implementing an ‘adapter’ interface and concrete class which will be used as kind of a gateway to the API. All of the functionality for connecting to and drawing information from the API will be placed in the adapter, thus lessening the complexity of our source code. Also, in the case that the API changes and possibly changes the name of function call, we would only need to change our functionality in the adapter class, instead of everywhere we use that API function in our source code.

3.2 Hardware and software

Since our app is completely software, our testing will be 100% software testing. We will be utilizing many functionalities in the Firebase Test Lab, specifically simulations. The Firebase Test Lab allows us to run simulated scenarios through GoMe, even allowing us to enact two simulated users to cover one test case. We will be heavily utilizing the Firebase Test Lab’s Robo Tests.

Firebase Test Lab Robo Testing: A Robo test recognizes the design of an application’s UI and then explores it strategically by simulating real user activity. A Robo Test always runs identical user activities in the same order with the same configurations. This lets us easily validate bug fixes and test for regressions in a way that isn't possible when testing with other similar test engines. A Robo Test saves log files, collects screenshots from navigation, and then creates a video from those screenshots to show the actions that the test performed. These logs, screenshots, and videos will help us determine the root cause of app crashes, bugs, and will be useful in finding issues with GoMe’s UI.

For other functional testing, we will use Android Studio test classes, writing our own tests for every important functional method or component.

3.3 Functional Testing

Unit Testing - We will unit test inside of Android Studio, writing tests to try to break individual components and functions. We will document expected outputs for several different data inputs and make sure that everything is correct.

Integration Testing - After testing individual components within the system, we will test how they interact with each other. This will include saving events to your calendar, social interactions, data storing, and other components interactions. Writing these tests will require brainstorming scenarios from many different people on how different inputs or buttons will trigger events to occur.

System Testing - Once we are confident in the working integration of the apps primary components, we can begin testing the large system functionalities. This includes ML functions, data streaming, schedule update triggers, etc..

Acceptance Testing - testing the main deliverable features and acceptance criteria before finalizing the deploy.

Table 1: Testing Plans for Functional Requirements

Functional Requirement	**Test Plan**	**Required Result**
System creates a dynamic schedule for the user to follow	Generate schedule in application for user, verify that ideal schedule is correct based on events like work school and sleep.	The system generates a sensible schedule for the user based on their previous habits.
Allows user to easily add events to their schedule	Test saving an event where free time is available and verify that it appears in the user's schedule for the day.	The saved event will appear in the user's schedule for the day.
Allows the user to share their schedule/events with friends	Test sharing an unchangeable schedule object with a user's friend. Then, try sharing an event link with another user.	The user's friend should be able to view the user's schedule and shared events. The events on click will successfully link to the application where the user's friend can view the event info.
Allows users to make changes to their schedule easily	Test the 'edit schedule' function allowing the user to adjust the amount of time that they will be doing something.	The schedule will reflect the updated event timespan and will adjust the user's level of time allocation and free time.

Allows users to see their profile and friends profiles easily	Test if a user can view their profile and other profiles and their information.	A click on a profile name or picture will link the user's screen to the profile page of the intended account
Gives the users alerts based on their schedule	We will need to view the generated schedule for the user and write down when events are scheduled to start. We will then wait and verify that the device receives a notification.	The device will receive a notification 15 minutes prior to a scheduled event.
Provides a social media platform for the user	Scroll through the feed page of the application, making sure that we are able to see every post made by friends in chronological order. This will require noting down all of the user's friend's social posts.	The user will be able to see all of their friends status updates and activity.
Allows user to "rank up" based on achievements and being productive/living a healthy lifestyle	We will complete events and achievements based on later decided criteria.	The user will be rewarded with an accurate amount of XP points which will be reflected on their profile page.
Database Requirements	Firebase Console makes it easy to view our database tables and verify that our data is being stored correctly.	All data will reflect accurate results as stored.
ML Requirements	Use static data to make a "control model", and then re-train "slave models" with the same static data. Then test the predictions from each of these slave models to check if it is within a certain tolerance of the control model	This will allow us to verify that any trained model will be accurate to within a certain variance.

3.4 Non-Functional Testing

The majority of our non-functional testing will be done with the help of Firebase Test Lab, which will save massive amounts of time when developing new features.

Performance Testing - Firebase Test Lab

We will do all of our application's performance testing with Firebase Test Lab. Running a test through firebase will launch our app on many devices at once and will provide us with analytics that help us understand how our app performs as the application's workload increases. It will test both the speed and stability and will generate a report that informs us where the application starts to slow down and if/when any crashes occur.

Security Testing

Seeing as we are using external technologies for our application (such as Firebase), we will have to rely on the security of these technologies to some extent out of our control. However, we will test the different possibilities to create a user– email registration and Facebook authentication—and attempt to breach this data from an outside source in order to verify no unqualified user can access the data being collected. We will also cover needed tests to assure that GoMe is safe from security vulnerabilities such as XSS.

Usability Testing - Field User Testing

We want GoMe to give every user a great experience, so we need to work hard on the UX portion of the app. In order to maximize this experience, we will utilize real people and let them use the app in real life! While using the app, we will make sure to survey them throughout the process to make sure we get their feedback on points we could improve on, noting where they may have been confused or unhappy while navigating the application.

Compatibility Testing - Firebase Test Lab

We will do all our of compatibility testing through Firebase Test Lab. By utilizing Firebase robo tests, we can launch our application on many different devices and OS versions. The test will generate a report of how the application performed on each device and if/when any crashes occurred. Test Lab will let you view screenshots of how the test navigated through the app allowing us to view where things may go wrong if they do. We will also be able to script starting points for the test flow in order to specify specific features or functions to test.

3.5 Process

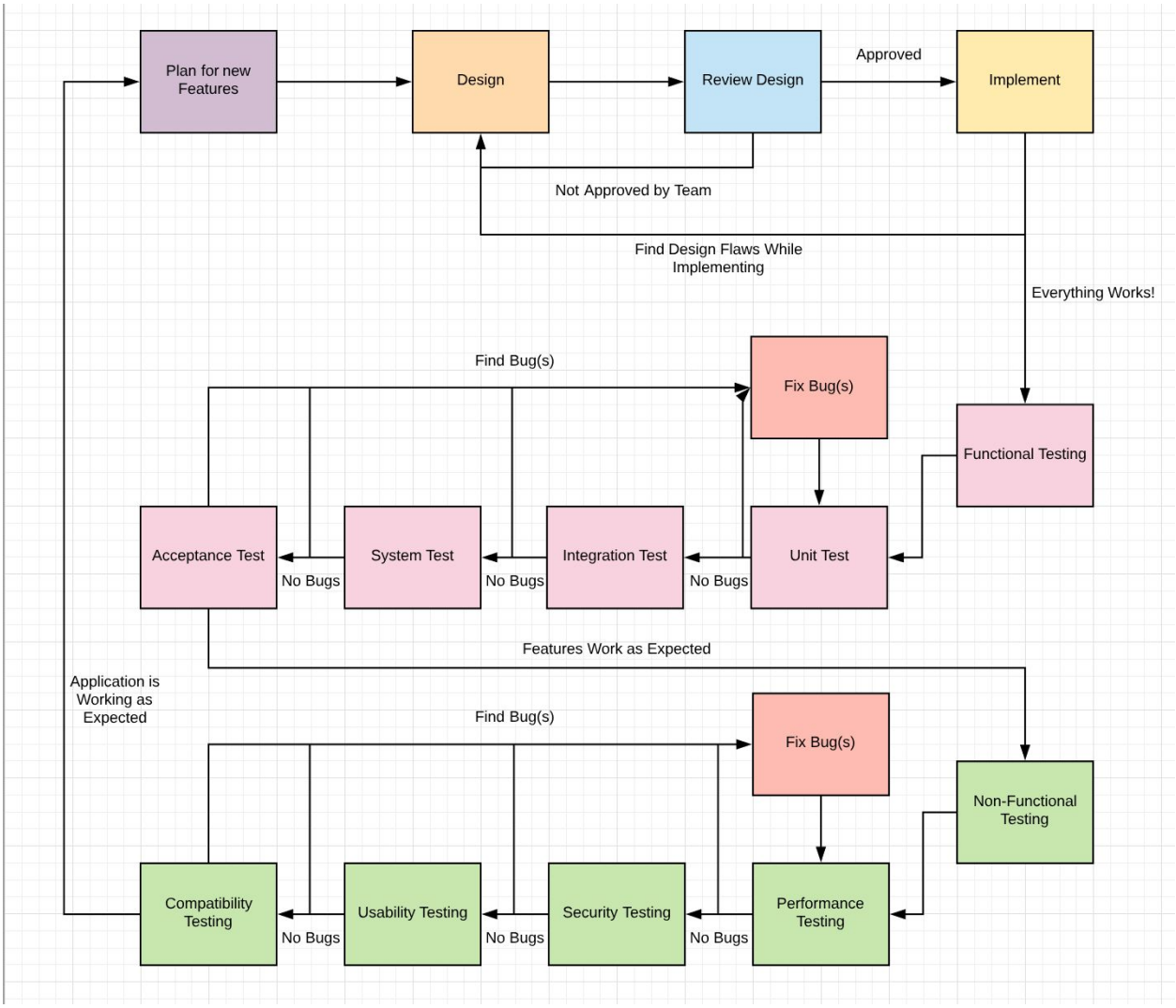

Figure 5: Testing Diagram

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

When a new functionality is added, it will have many detailed scenarios on how and when the new feature is used. These generated scenarios will be used frequently throughout testing. Using the Firebase Test Lab (detailed in section 3.2), we will be able to cover these scenarios by simulating them with the Test Lab. We will be utilizing 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 will allow us to brainstorm and test every possible scenario for a new feature with ease. Additionally, new features will be tested against our functional and non-functional requirements, as described in Section 3.3 and 3.4. Our end goal is to have 100% test coverage of our code.

3.6 Results

Our team has implemented the necessary APIs to create our data model for each user, created a neural net that understands sleep patterns, and designed a system that will be easily expanded and managed. We have outlined the testing of our system, and are beginning to test what has been developed thus far. We have spent time researching Firebase Test Lab to see how we can use this to test our application. We expect that as our application is developed, we will have to update our testing of functional and non-functional requirements accordingly.

4. Closing Material

4.1 Conclusion

GoMe will be an application that seeks to better the user's life by creating an automated schedule able to give the user feedback and help them live a more balanced life. This scheduler will be exceptionally user friendly and much easier to use, understand, and navigate than other scheduling apps. GoMe will also understand the user and give them analysis based on what the user did throughout the day. It will attempt to motivate the user to do the things they need to do by building a social media platform that revolves around the user's schedule and other motivational features.

Using an agile process, we have been able to communicate with our client/advisor on the design plans for this project, received feedback, and made adjustments accordingly. This has helped when creating design documents and starting on the initial development of our application. Our documents focus on the functional and non-functional requirements of our application in relation to the use cases described in the project plan. We have specifically outlined how these requirements will be tested and when. We have broken down tasks and goals into separate iterations of our project, with the flexibility to adjust these iterations as we go. The team has split up our members into sub-groups that focus on the data analysis, machine learning, and database/system aspects of our application. By having a few people work on these parts, we have been able to research and develop more in-depth. Overall, our team has defined roles, developed a plan to fully implement the requirements for our application, and started implementation to create a product that will aim to make the user's life easier and more enjoyable.

4.2 References

“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].

“Figure 2f from: Irimia R, Gottschling M (2016) Taxonomic revision of Rochefortia Sw. (Ehretiaceae, Boraginales). Biodiversity Data Journal 4: e7720. https://doi.org/10.3897/BDJ.4.e7720.”

“Firebase Test Lab Robo Test | Firebase,” *Google*. [Online]. Available: https://firebase.google.com/docs/test-lab/android/robo-ux-test. [Accessed: 26-Mar-2019].

Microgreen, “Data Modeling for Artificial Intelligence,” *Responsible Machines*, 04-Oct-2017. [Online]. Available: https://responsiblemachines.wordpress.com/2017/01/17/data-modeling-for-artificial-intelligence/. [Accessed: 26-Mar-2019].

“Preparing Your Dataset for Machine Learning: 8 Basic Techniques That Make Your Data Better,” *AltexSoft*. [Online]. Available: https://www.altexsoft.com/blog/datascience/preparing-your-dataset-for-machine-learning-8-basic-techniques-that-make-your-data-better/. [Accessed: 26-Mar-2019].