Task GPS

By Matthew Miller and Landon Irwin

On behalf of the Nickoloff Faculty Cohort

18 June 2019

Introduction

The Nickoloff Faculty Cohort is a team of faculty at the Colorado School of Mines. The cohort operates under the Thomas R. Nickoloff Entrepreneurship and Innovation Fellowship Program. They aim to incorporate an entrepreneurial mindset into their teaching and research. Currently, they are working to develop tools to help students manage their time to increase productivity and reduce stress.

The cohort is designing a mobile and web-based application to improve time management for tasks called “Task GPS”. Task GPS should allow users to easily create and manage tasks with deadlines. Time is allocated for the user to complete these tasks in a reasonable time-frame. The user receives suggestions from the application on how much time tasks will take and when to complete them. The application receives feedback in the form of actual time use and adjusts accordingly. Task GPS is a personalized time management application designed for everyday use on the go.

Based on the vision for Task GPS, we focused on highlighting the minimal user interaction, personalization, and data science in the design of the application through the duration of this field session. In order to import, create, modify, and track tasks, there should be as much redundancy and defaults as possible to minimize user interaction. There should be adequate settings in order to provide personalization. Additionally, a major component that the Nickoloff faculty cohort wants to establish for Task GPS is a machine learning model that predicts properties of tasks based on individual user interactions with the application. Concise visualizations about task progress should also be provided to the user. The app should schedule sessions of work time for the user based on the machine learning and settings to help reduce procrastination. Finally, the application should run on most mobile devices and synchronize data between devices to maximize user outreach.

Requirements

The following set of requirements provide specific details about the overall components and specify features that needed to be implemented for Task GPS to be considered done. Additionally, there were constraints that we followed during the development of Task GPS that are detailed in these requirements. We categorized the requirements based on the principle goals for the application so that we could plan out sprints for our Agile process more easily.

Functional

This section of requirements details the features of Task GPS that should have been implemented. It contains only what should have been implemented and not how it was implemented.

- **Management** - It should be simple and quick to manage tasks and events within the application. Additionally, managing these tasks and events should allow the user to get at least all of the experience of a basic to-do list.

    - User must be able to create and modify tasks, which have a name, a due date and time, and an estimated duration, as well as events, which have a name and start and end dates and times.
    - There should be an option to import tasks and events from external applications or services. Namely, the user should be able to import tasks and events from Canvas and Google Calendar.

- **Time Tracking** - In order to incorporate time into the application, Task GPS should provide a way to track progress made on tasks.

    - The application should include a timer that counts progress made on a user-selected, incomplete task.
    - The timer should support a rounds system where the user works for a specified number of minutes and then the user breaks for a specified number of minutes. There should at least be support for rounds of 25 work minutes/5 break minutes and 52 work minutes/17 break minutes.
    - The user should be rewarded for spending time on tasks and using the application efficiently.

- **Scheduling** - To best reduce procrastination, the application should provide suggestions of when to work and what to work on.

    - For each task, the schedule should have allocated at least the amount of time that the user estimates the task will take.
    - Each day should have approximately the same amount of free time so that the user does not get bogged down with too many tasks.
    - There should not be periods of work in the schedule that are less than a certain threshold minimum work duration.
    - Working sessions should never be scheduled on top of an event timespan.
    - The user should be notified every time that one of the scheduled working sessions starts.

- **Visualizations** - There should be useful visualizations that summarize the progress on all tasks as well as visualizations that summarize progress on individual tasks. These visualizations should be concise and highlight user performance.

    - There should be a visualization that is a stacked line chart that shows the amount of time left on each task per day. Thus, the top line in this visualization describes how much time is left on all tasks.
    - There should be a visualization that is a stacked line chart that shows the amount of progress made on each task per day. Thus, the top line in this visualization describes how much time was spent on all tasks combined per day.
    - There should be a visualization that shows the percentage of a task is completed between the dates the user started on the task and when the task is due. There

should be a line on this chart that represents now so the user knows if they are behind schedule or not.

- **Data Synchronization** - The application should be as portable as possible meaning that if users have entered data before, they should never have to re-enter that data. This requires the user of data synchronization between devices.

    - Each user should be presented with an option to create an account for Task GPS with an email and a password.
    - Signed in users should have their tasks and events stored on a database that is always accessible and provides automatic data synchronization.
    - Users should be able to use the device without signing up for an account with some features limited.

- **Machine Learning** - Machine learning should be used to predict properties of the tasks that the user creates. For now, the machine learning should take the task name and the estimated time of a task and predict the actual amount of time that the task will take.

    - There should be a pre-trained version of this machine learning model that is hosted online that user devices will download when needed.
    - Each user device will re-train the machine learning model that is stored online in order to create a personalized version that is more accurate.

Non-Functional

This section of requirements details the constraints of Task GPS during development. These are either client-imposed or resource-imposed constraints. The constraints are either general requirements for the application or restrictions on our development process.

- **Interface** - The overall user interface and experience for the application is important since first-time users will only give Task GPS one chance to impress them.

    - The user interface should be easy to use and intuitive. It should be stylish and designed logically. Basic localization should be implemented such as icons for buttons and simple translations.
    - The application never stalls or delays the user tremendously. This applies particularly to machine learning parts of the application.
    - There should be settings to customize the user's experience that apply broadly to all users.

- **Development** - During development, we had a few restrictions in order to make sure that existing progress on the application did not go to waste and that future work on the application is easy.

    - The application should be developed using JavaScript on the Cordova mobile application framework. Cordova is a cross-platform framework so it is necessary for the deployment to multiple mobile device brands.

    - We use GitLab, which is similar to GitHub, in order to make the repository we are working on private, share code between team members and the client, and to track features and bugs that need to be implemented or fixed respectively.
    - The code that we create during field session should be easily extensible since there are many planned features to implement once we have stopped working on the project.

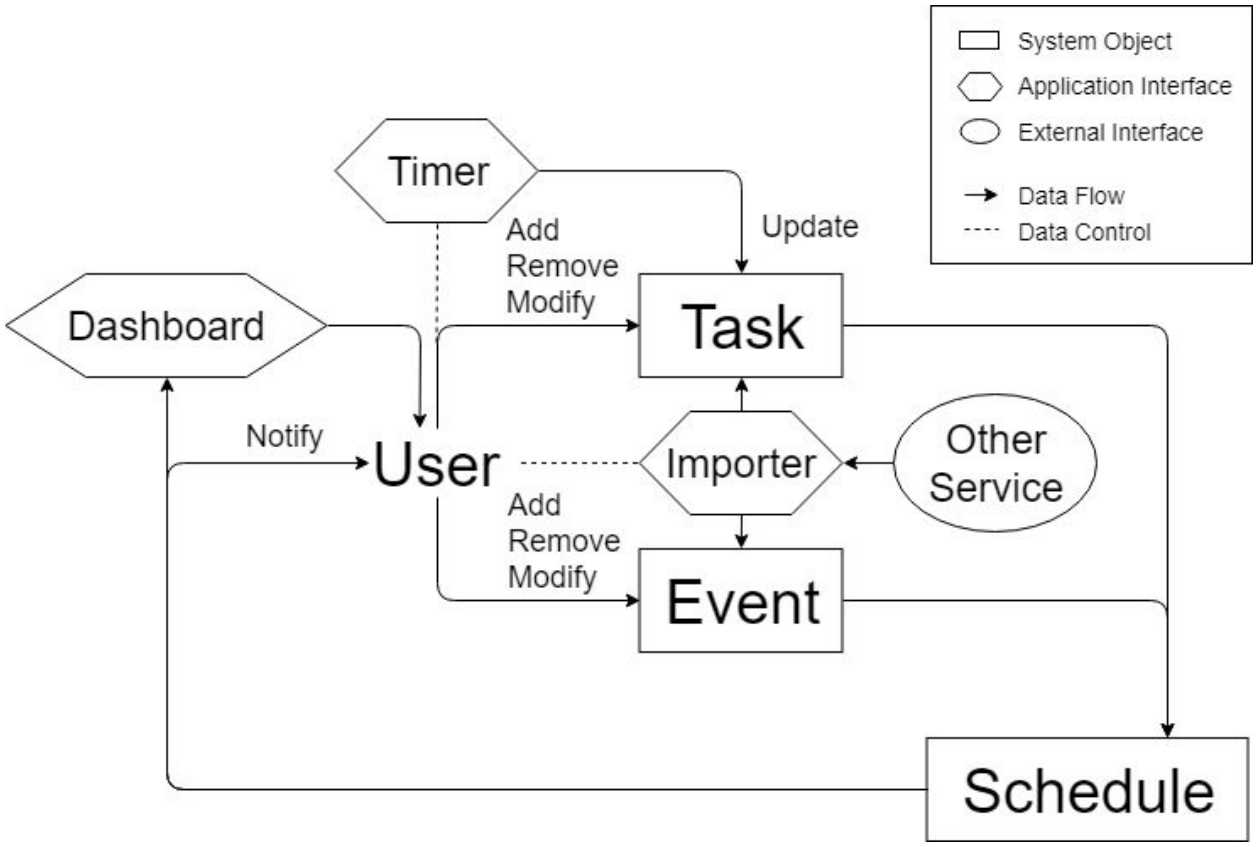
- **Deployment** - The application is planned to be released after the field session is over so there are a few deployment steps that we need to take to ensure a successful release.

    - The application will be deployed on the Apple App Store, Google Play Store, and online as a website. All prerequisite information to deploy to these platforms must be obtained.
    - The application must have enough of the required features to be useable in a course on time management by the following fall semester.

System Architecture

The Task GPS project is a full-stack development project since it incorporates both back-end resources—machine learning, a database, and a web server—and front-end resources—the user interface, notifications, and application assets. A full-stack project such as ours requires an elaborate system architecture. We describe the macro-scale view of our project design in the following section.

Figure 1 - User Cycle

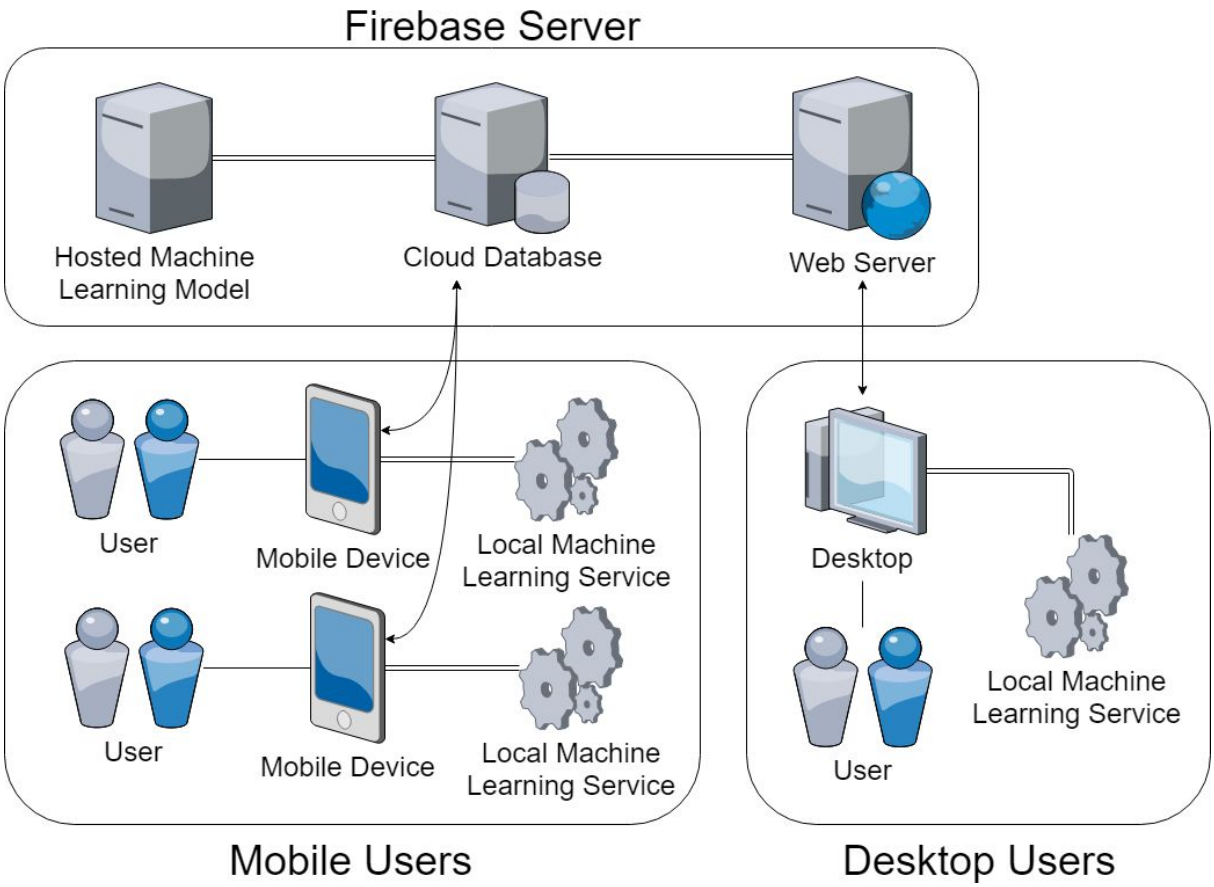


In Task GPS, the user is presented with numerous ways to interact with the application that produces responses later on. In *Figure 1*, the actions that a user can perform are visualized.

The user only interacts with tasks and events. These objects can be directly added, removed, or modified by the user or can be imported from other services such as Calendar apps. The timer is used to keep track of the progress of tasks. With it, the user modifies tasks indirectly. Both tasks and events are taken into consideration when making a work-time schedule.

The schedule is used to provide useful information to the user on the dashboard. Finally, the user is notified of work sessions in the schedule as they pass via local notifications in the device's notification center. Some other components of the application such as machine learning or data synchronization work in parallel to these operations. They are not included in *Figure 1* because they are not visible to the user.

Figure 2 - Database Synchronization

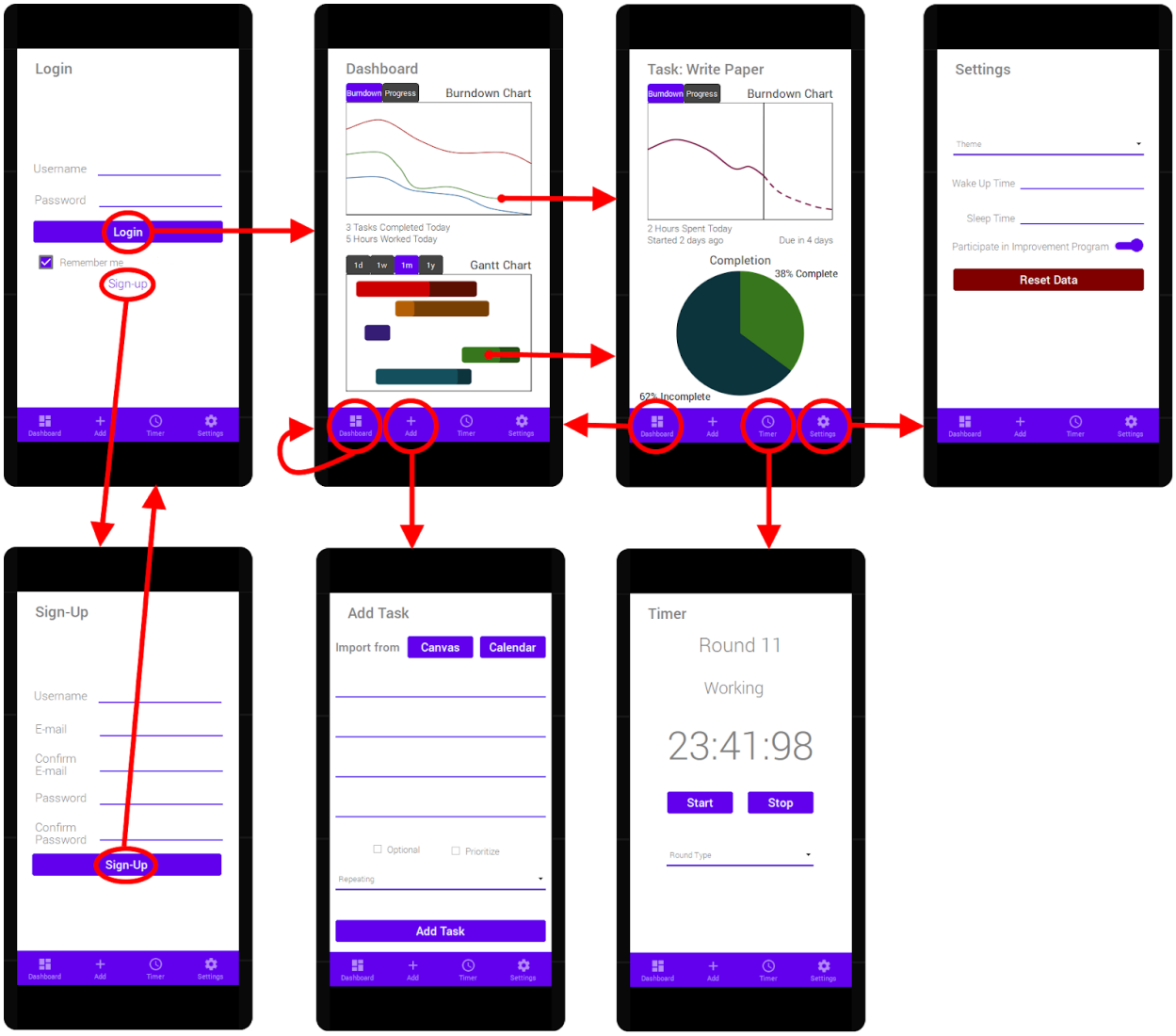


The users of the Task GPS app fall into two categories. Most users will be mobile users with the Task GPS app downloaded onto their phones. Other users may use a desktop instead and access the web hosted version of the application. The data flow between user devices and the Task GPS components is demonstrated in *Figure 2*.

We host our application back-end components on a Google Firebase server. This was the simplest and most reliable way of distributing services for Task GPS. Contained on the Firebase server is a hosted machine learning model. This is used as a starting point for personalized machine learning models. Then, the database on the server synchronizes information between devices as long as they are logged in to the same account. Both mobile and desktop devices access the database and machine learning model. Finally, a web server is hosted for non-mobile users.

Users on both types of platforms should have the same type of experience even if data is distributed differently. We achieve this easily since we are using JavaScript to develop the mobile application which means it can easily be deployed as a web page. However, we do lose some functionality such as notifications and service integration on the web version. Otherwise, both types of users have the same experience.

Figure 3 - Application Wireframes



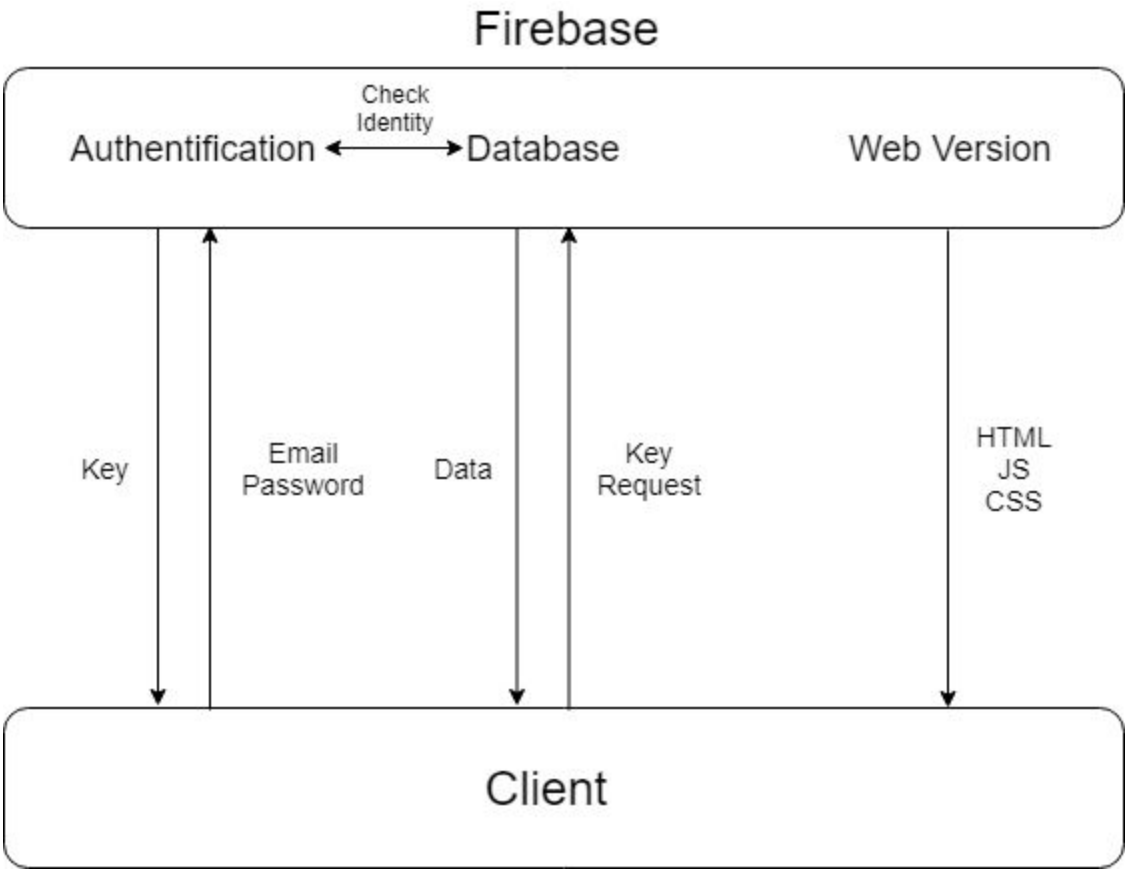
There are five main application pages that the user will be able to use in Task GPS. The pages are shown in *Figure 3*. The pages will be accessible via a navigation bar at the bottom of the application. First, there is the dashboard which will display the visualizations described previously. Next, there is an add and manage page that allows users to add, import, and edit tasks and events. There is a timer page which presents a timer with the features described earlier. Finally, there is a settings page which will allow the user to customize options such as the theme, scheduling parameters, and their involvement in the machine learning improvement program.

Besides these pages, there are additional sign-up, log-in, and privacy policy pages. The sign-up and log-in pages are self-explanatory. The privacy policy page describes the privacy policy which must be agreed to in order to sign-up for an account.

Technical Design

The Task GPS project contains many intricate components to function. Many of these parts required lots of research and brainstorming in order to produce a design that works. Primarily, back-end components of the project such as the machine learning or database required the most details. We describe a micro-scale technical design overview of these components in the following section.

Figure 4 - Firebase Interactions



As stated previously, we decided to use Google's Firebase service. The service provides many features to help deploy applications on Android, Apple, and web browser. Thus, this service was exactly what was needed for our project. Specifically, we used the Firebase Authentication, Firestore, and Hosting services as shown in *Figure 4*.

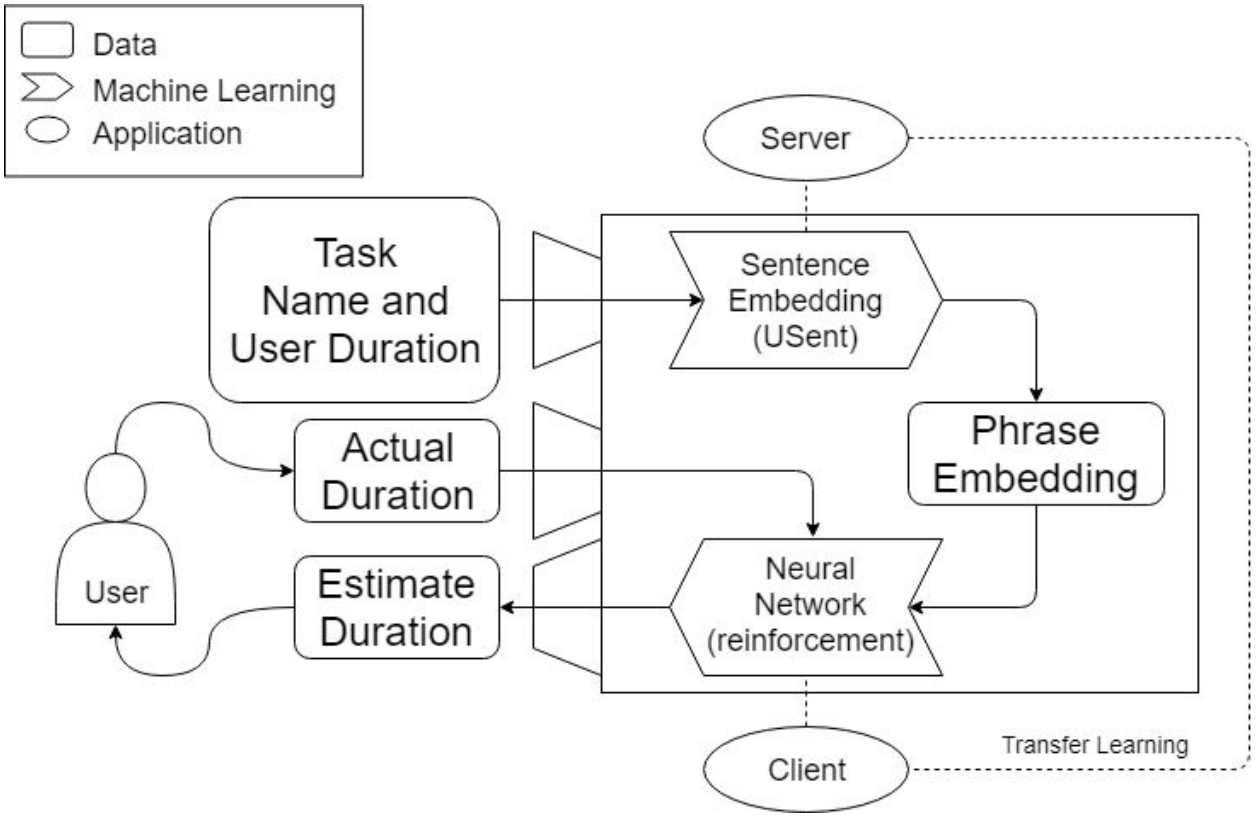
First, since we are using JavaScript and Cordova is a web-based framework, we decided to use the web-based Firebase libraries for Task GPS. We needed a way to implement a user sign-in and log-in system that was secure and interacted with the database. Next, we needed a database itself that was able to store each user's data in a separate and secure collection. Firebase allows us to have a web page hosted for users that are not using mobile devices as a fallback mechanism. Finally, we needed to host the entire application in the cloud so that all devices can access the machine learning model, database, and web page.

The web hosting for Firebase is relatively simple to work with. The entirety of the Cordova project is essentially a web page with multiple HTML, JavaScript, and CSS files. Cordova takes these files and converts them into a format that can run on mobile devices. Since the project was set up like this, we could simply export the files to the Firebase web hosting service and deploy the application. As mentioned previously, this results in the downside of not being able to use notifications and calendar services like in the mobile version of the application.

The authentication from Firebase allows us to create user accounts. In the Firebase configuration, there are multiple ways to allow user sign-up and log-in. The most convenient way for us is to allow the user to simply enter an email address and a password to create a new account or access an existing one. Associated with this account is a unique identifier. This identifier is considered public and is okay to transmit freely online. The email and password information is sensitive information, so we have to use Google's cryptography module 'scrpyt'. In return for the user information, the Firebase authentication server sends back a key that can be used for other services.

The key for the authentication is used to set up a directory for each user in the Firestore database. The Firestore database design deviates from traditional relational databases. In Firestore, there are a series of collections which contain at least one document. Each document can reference more collections. Thus, we designed the following data structure in order to contain our information. There is a collection of users with corresponding user documents. Each user document points to collections of tasks, events, and work sessions. The information associated with each of these objects is detailed below.

Figure 5 - Machine Learning Process



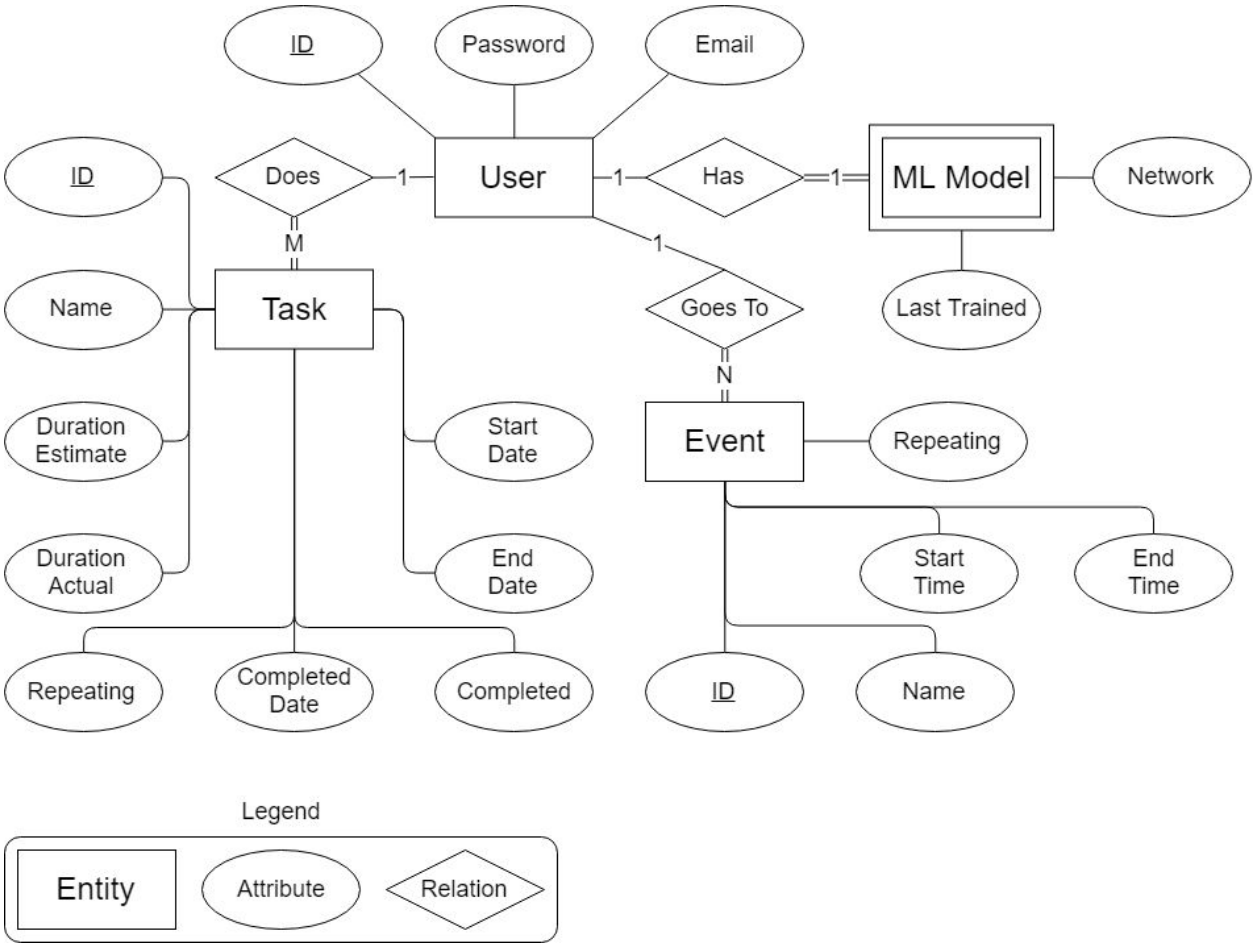
In order to extract useful information from the tasks that users create, we will be using machine learning. Since we are provided with very little information from the user, we must create rather complex algorithms to predict useful information. Currently, we are attempting to predict the amount of time that a task will take given its name and estimate duration. A demonstration of how this machine learning will take place is given in *Figure 5*. We hypothesize that there is some correlation between the name of a task and the amount of time it will take.

The input to our machine learning model is a task name and the output is the estimated duration of that task. First, the task name is converted into a phrase embedding by the open-source *Universal Sentence Encoder Lite* model. The phrase embedding is simply a numerical representation of the task name. The phrase embedding is then passed into a neural network in order to predict the task duration.

The neural network is initially stored on the Firebase server. This is a model that has been trained on data from all agreeing users. Then, the neural network is transferred to the client device to be reinforced on individual task data. As the user uses the app, they generate information about the actual time required to complete a task which is used to perform reinforcement training on the neural network.

The machine learning is all performed using Tensorflow.js which is version of Tensorflow which is more portable and can run on JavaScript. The Universal Sentence Encoder is retrieved from its source whenever the application is loaded and the user is entering information. The time prediction neural network is hosted on the Firebase server as a TensorFlow lite file. The model is downloaded to the client when necessary and then stored in the Firebase database when personalized.

Figure 6 - Database Entity-Relationship Diagram



The task and event data that users generate must be in a consistent format that can be stored on the Firebase database. Luckily, the Firebase database is very lenient in how it stores data so data types and data capacity were not a concern. However, we still needed consistency so that the application does not fail when retrieving data. Thus, we used the model given in *Figure* 6 to store data.

In our model, there is a user entity which represents a single account with an email and password pair which has multiple tasks and multiple events. Every user has a machine learning model which is composed of a neural network and a timestamp determining when the model

was trained last. The timestamp is used to determine when to retrain the model. Each task and each event has the information required to describe its name, dates and times, and additional metadata about the task or event.

Quality Assurance

There are many components of the Task GPS project that needed to have high quality in order for the application to be successful. This is different than the implementing features that we discussed in the requirements section. For instance, the application may implement machine learning but the predictions may be so terrible that it causes more harm than good. For this reason, we have developed a quality assurance plan for Task GPS. We first introduce the following key quality aspects and concerns that must be resolved before being the Task GPS project is considered complete.

- **Machine Learning** - As per the example, the machine learning model must be accurate enough that its prediction for actual time spent is better than the user estimate. In addition, the model should be optimized to be computationally efficient. If the application stalls when machine learning tasks place, nobody will want the machine learning feature. Finally, the machine learning model should be easily modifiable since it is currently experimental and will likely experience many changes.

- **User Interface** - The user interface must be functional and appealing. No user should have difficulty looking at the application. Additionally, every part of the application should be styled consistently. Further, if the user interface does not promote useability, then it is too complicated and should be simplified. Finally, the settings for the application should contain styling options such that every user is satisfied with the look and feel of the application.

- **Data Synchronization** - As previously noted, the application uses Google Firebase to store and synchronize data. Since we are using an external service to provide data storage capabilities, we need to make sure we are completely familiar with how secure the service is. We need to make sure that user data cannot be leaked and that a user's identity cannot be deduced solely by the information they provide. Additionally, we need to make sure that each user is aware of the usage of their data.

- **Source Code** - This project will continue to be developed thoroughly after we have finished this field session. Thus, future developers need to be guaranteed that the code that we produce is easily maintainable and extensible. Further, the codebase needs to follow good software engineering principles.

Plan

In order to make sure that the qualities above are ensured, we have devised the following quality assurance plan. This plan details actions we have taken in order to meet our quality goals stated in the previous section.

- **Code Reviews** - Each member of the team reviewed major components of the code that we create on a regular basis. Additionally, a software engineer in the client group helped review the code.

    - This makes sure that code is understandable to everybody that is part of development or may be part of future development. The interfaces and modules in the code should make sense.
    - The codebase will be architectured well enough that any common programming task on the application should only take a few lines of code by using well-researched modules and services and actively refactoring code to follow the SOLID software engineering principles.

- **Unit Testing** - We tested the basic functionality of key components of the application by writing unit tests for primarily used modules. We performed the unit testing by using the QUnit JavaScript testing library. Both members of this team and a software engineer in the Nickoloff Cohort wrote unit tests. This helps guarantee that the modules work correctly under all conditions for future developers. We implemented QUnit tests such that it will be easy to add them to our GitLab continuous integration. This ensures that no code will be committed if it is faulty. The following modules were tested with the specified tests.

    - *Schedule* - The scheduling module provides utilities to create a schedule given specified tasks and events and scheduling parameters such as wake time and sleep time.

        - The amount of time allocated for each task should be at least equal to the remaining time for that task and should be no greater than twice the remaining time for the task
        - Task sessions should be spaced across days such that the free time on each day between task due dates is approximately equal. The margin for error on this equality is the minimum session duration.
        - Task sessions should always be at least as long as the minimum session duration.
        - Task sessions should never be planned in a way that they intersect with existing events at the time of scheduling.
        - Task sessions should not be planned before the user-specified wake time or after the user-specified sleep time.
        - Repeating tasks and events should be accounted for when scheduling such that the schedule acts as though multiple individual tasks and events are planned.
        - Events that span over multiple days should act identically to multiple events over single days.

    - *Storage* - The storage module allows data to be stored locally on devices as well as in the cloud database easily.

        - Data passed to the storage module should be stored on the user device between multiple sessions of opening and closing the application.

        - Data stored in a JSON format should be correctly stored and retrieved such that none of the original data is lost or corrupted.
        - Data should be stored locally when user not logged in and should be stored online when user logged in. Data should be transferred upon logging in.

- **User Interface Testing** - We made sure that the user interface works according to design and that using the user interface produces no bugs or unintended effects. Additionally, we manually tested that each feature within the application is accessible in an intuitive way in a small number of actions.

    - We tested that user is made abundantly clear about the usage of their data in the application by making sure that every pop-up and notification appears correctly when appropriate. Specifically, the user should be notified:

        - When they are signing up about the terms of data usage by the app.
        - When the user has added a task, event, or recorded a session that is transferred to the database about not releasing any sensitive data and the use of the data.

    - We ensured everything in the app in accessible in ten clicks or less without confusion by following instructions in user cases. Specifically,

        - The user should be able to find any task or event in less than ten clicks.
        - The user should be able to start timing a task in less than ten clicks.
        - When a notification to start working is sent to the user, the user should be able to start timing a task in less than ten clicks.
        - Any setting should be able to be changed in less than ten clicks.
        - Information about how much progress is made on tasks during the week should be able to be accessed in less than ten clicks.

    - We showed that the user interface promotes recurring user population by making sure that the UI follows the material design principles as specified in the Material Design Guidelines and by using Material Design Components for the Web.

- **User Acceptance Testing** - It is most important to the future success of the application to make sure that potential users of the application are willing to keep using the application. We will present the application to multiple potential users before deployment and observe how they interact with the application.

    - This will help make certain that everything the user would like to know and every originally intended feature is available in the application by having client use the app and confirm that every feature is implemented. Specifically, the following should be available and working:

        - Dashboard and visualizations of task progress
        - Adding a task or event
        - Modifying a task or event

        - Timing a task
        - Being notified of a task
        - Machine learning for task duration prediction
        - Data synchronization
        - Settings and customization

    - We will guarantee customization settings work for all users by handing the application off to multiple users and making sure that the user can customize the settings enough for the app to always work as the user intends.
    - The application will be deployed and deployment of the application will be tested by making sure that application is downloadable correctly by multiple people from the Google Play Store.

- **Mockup Server Attacks** - The user is submitted their information to an external service. Thus, we must make sure that the data that the user submits is secure and that the user is aware of the usage of their data. We will execute a series of tests attempting to access the database in unintentional ways.

    - This will protect against future security threats by making sure that they cannot cause damage to the current system by using a well-tested system such as Google Firebase.
    - We will ensure no useful confidential user information can be leaked from the system by restricting permissions of the user and performing mock break-ins to test security. We will do this specifically by using Google Firebase's security simulator to:
        - Make sure users that aren't signed in can't access the database.
        - Users can't access other users' information in the database.

Results

For the Task GPS project, we were able to implement most of the initially required functionality and even implement some stretch goals such as adding themes to the user interface and allowing customization of the scheduler through the settings menu. We have designed what we hope to be a successful product for the Nickoloff Faculty Cohort. Concluding this field session, we reflect on the accomplishments that we have made and the work that we have done,

Unable to Complete

Unfortunately, there were some features and goals that we were unable to achieve. Either there was a lack of sufficient development time to actually implement a feature or we had insufficient resources to accomplish what we planned. The following section is a summary of the features we were unable to complete.

- We could not host a server version of the machine learning model which trains on all user data and a client version of the model which is only trained on individual user data.

- We did not have the resources to implement localization for different languages and locations.
- Service integration with Canvas for importing tasks and events was infeasible due to requiring some amount of administrative status on the service.
- Gamification of completing tasks such as rewarding experience and prizes could not be completed in the given time frame but will be recommended to the client.

Quality Assurance Results

We were able to execute all parts of the quality assurance plan to some degree. Most of the results were satisfactory and provided good insight on the project. The following is a summary of the results that we found while executing our quality assurance plan.

- The code reviews were successful. We were able to refactor code into highly extensible units that follow the SOLID principles.
- The tests for the scheduling and storage modules are passing. Some of the schedule tests alerted us of some bugs with the even distribution of tasks. We were not able to implement the continuous integration for GitLab.
- The user interface and app has been demonstrated twice a week to the client and to a few other potential users of the app. They have approved the user interface design and app functionality. Many possible origins of bugs in the user interface have been investigated and resolved.
- The ten clicks or less test on the app revealed the following results. These results correctly reflect that what we are prioritizing in the application is available in the least number of clicks (such as progress and timing).
    - The user can find any task within at most 5 clicks if not signed in and 2 clicks if already signed in.
    - The user can start timing a task within at most 9 clicks if not signed in, at most 6 clicks if signed in, and 4 in the common case.
    - The user can start timing a task from a notification in at most 3 clicks.
    - All settings are able to be changed in at most 3 clicks.
    - Progress information is available in at most 3 clicks if not signed in and at most 1 click if already signed in.
- The machine learning was not implemented completely but the architecture is available for our client to extend. Data still needs to be collected in order to train the models that we have researched and designed. The current model does not cause the application to stall since it is running on a background thread that is hardware accelerated.
- The database has been tested against malicious requests in the Google Firebase simulator. None of the attempts to add, modify, or delete other users' data or central data were successful.
- The application has been successfully deployed to the Google Play Store and Apple App Store under the name "Task GPS", and to the web hosting server.

Extensions

We have come up with multiple additional features that the client might consider implementing in the future. The following are recommendations for the client to extend the application to be more successful.

- The machine learning in the application should be extended from only predicting the amount of time that a task will take to predicting what amount of time the user spends taking breaks, when and where the user is most productive, and how important tasks are.
- Adding gamification to the app in order to add rewards for using the app to track tasks and to not procrastinate. A leaderboard should be available for users and users should be able to join groups or organizations such as a particular class at the Colorado School of Mines in order to compare how they are doing on a particular task compared to others.
- Add subtasks and goals to the current task structure so that users can better organize tasks and the scheduler can better plan out specific sessions for a task.
- Add more synchronization between devices such as synchronizing when the timer is running, synchronizing settings, or synchronizing the screen where the user left off.

Lessons Learned

Throughout the duration of this project, we encountered many issues and had to adapt the trajectory of the project to meet both our client's and our own needs. The following section details a few of the lessons we learned in this field session.

Web applications can be nice to develop since the web-based design allows for inherent cross-platform capabilities. We were originally hesitant to use a web-based framework over traditional binary applications but we learned that there are a lot of benefits to going web-based. Many common libraries and visual frameworks already exist and are easy to include. Modifications to the app affect all platforms very similarly.

Actually deploying an application is a very complicated process separate from developing the application. Our application, although compatible with all types of devices, follows entirely different processes for deploying onto those devices. The Google Play Store is relatively easy to deploy to with a low initial cost while the iOS store has more restrictions and guidelines along with a steeper annual fee.

Setting up new services or integrating existing services is always nontrivial. Although there may be some great service out there that does exactly what is needed for a project, actually using the service in a project is difficult.

- We wanted to set up a web server quickly based on an existing web framework for Python called Django. However, there were complications with getting a server hosted on campus that broke our time constraints.

- We wanted to integrate with the Canvas service which was initially thought to be a simple API call. It turned out that it was impossible for us to set up the integration because the process required access to a Canvas administration account and a key that couldn't be deployed in our application due to security issues.

Having the right setup for development is important. We spent approximately an entire week attempting to set up development environments that suited our needs. Additionally, not having the ability to set up computers on campus without tech support was detrimental because it made us have to wait longer. Problem after problem arose with various conflicting softwares and packages and eventually was resolved by some crafty tricks.

With the client involved during the entire development process, we developed a higher quality vision than would ever be possible if we were just given a list of expectations. We were able to easily bounce ideas for the project back and forth to the client and refine and modify the end result to adapt to what is feasible.

This project was essentially the first full-stack experience that we have had and it required lots of time spent to learn new technologies. We had to work on all parts of the application from database management to app functionality to the user interface to the deployment. All of the different parts required vastly different knowledge bases. A lot of research was involved in the project and how to implement each part of the stack.

Concluding Remarks

We think that this field session was a complete success for us and the Nickoloff Faculty Cohort. Not only did we develop a successful application, but, we also learned lots about the software engineering process from architecture to deployment. We used the Agile development process which helps make projects like the Task GPS release successfully. Finally, we got to experience a realistic software development setting to experiment in before developing in industry. This field session will help us greatly in our future careers.