Milestone 4

Team Name: Sweet Victory

Revised Feature List:

- Task prediction calculator: a view of some sort that can show how long one would need to work on each task every day to get it done before the due date
- Be able to schedule meeting with others based on available times from other group members
- An alert system that notifies a person if they have a schedule conflict
- Be able to read, update, edit and delete events
- Different themes to change to look of the application
- Two-step Verification: two factors of authentication to verify that the user is who they claim to be

Milestone 4
March 21, 2019
CSCI 3308

Architecture Diagram:

Our application stack consists of React frontend, Node backend with Postgre for our database. We are serving up the application via node locally. This is the front end and how the state and components relate to each other. Node will have to be used for all forms via GET request. The login page will have a POST request to keep the user's login information private. The edit, delete, add, and view features of the calendar will all have to make a GET request to the database to show the user's events.

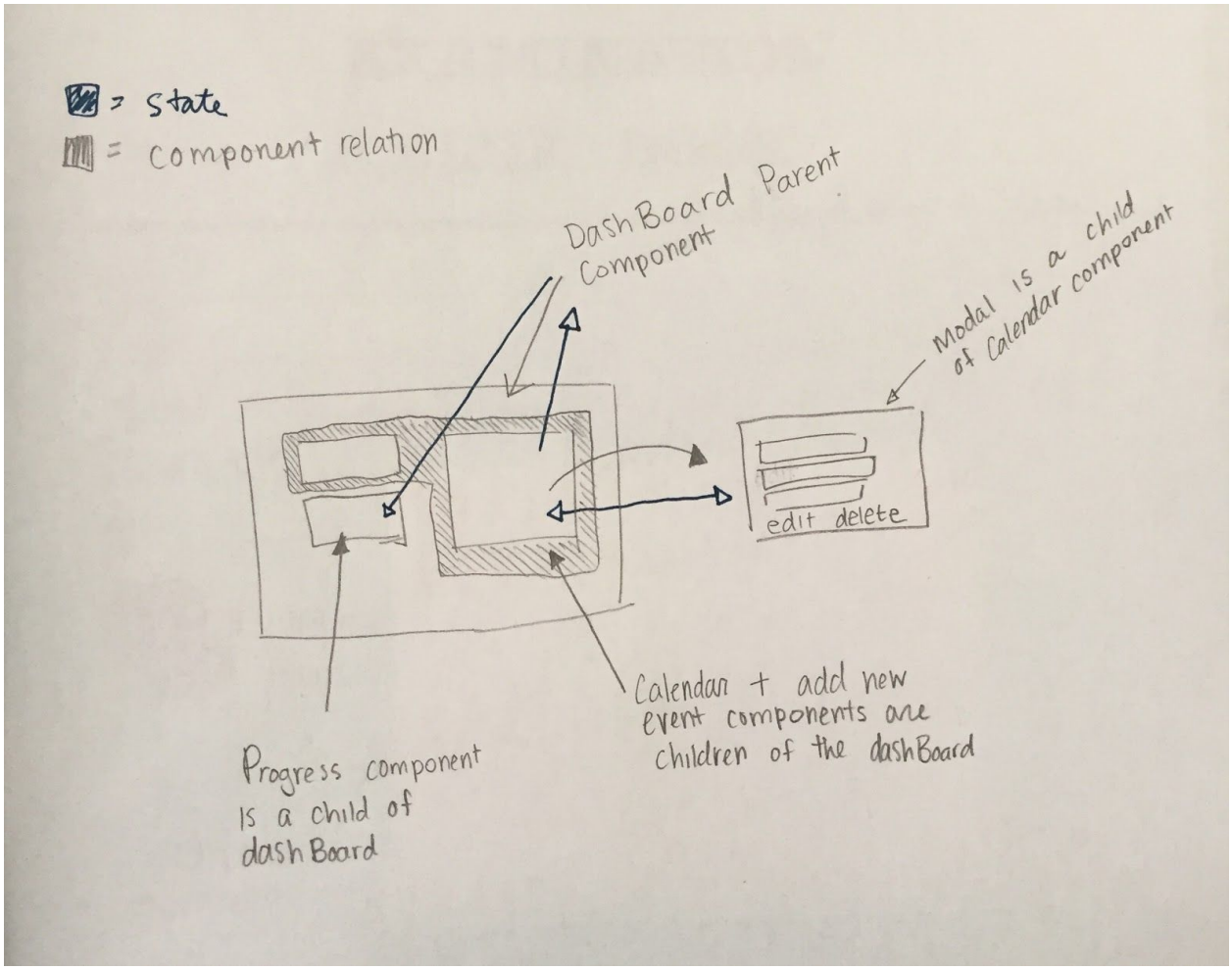

Figure 1: Architecture Diagram

Front End Design:

We are using a calendar API that was made for React, however the documentation is difficult to understand. We are using a modal that was designed for React for forms. These dependencies were all downloaded with npm. We may be switching to another Calendar API.

With either API, the interface will be continuously modified depending on what feels aesthetically and physically comfortable.

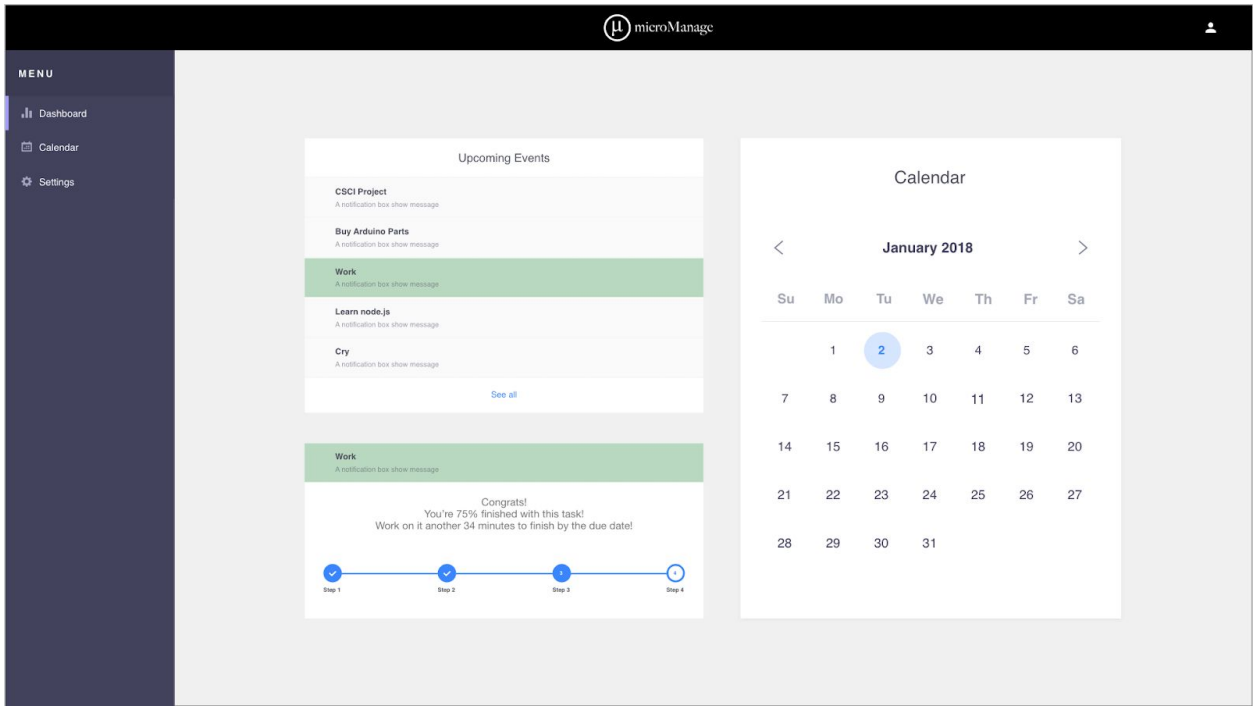

Figure 2: High fidelity screens made with Adobe XD

Web Service Design:

We are using Node.js for our backend. We don't think we will have to use Express or another node framework since React can work well with node. Our Web service will be a REST API because we are using JSON and HTTP requests. We need the JSON files for project initialization and managing dependencies. We will be using HTML GET/POST requests to communicate with our database.

Back End Design:

We are using PostgreSQL for our database. We have one table that will have the usernames, passwords, and the ids. We then have another tables that will hold the events. The event table is able to reference the user table via foreign keys. To organize more efficiently, we may add more tables and separating columns.

Users Table

Columns	Column Values/Description
user_name	varchar | Name of the user
user_password	varchar | stores user password for entry and authentication
user_email	varchar | stores user email for reminders and two factor authentication
user_group	varchar | Stores the group the user is in for scheduling purposes
user_group_password	varchar | stores password to enter group

Appointments Table

Columns	Column Values/Description
name_of_event	varchar | Name of event
date_of_event	timestamp without timezone | Date of the event
length_of_event	interval | Length of the event
urgency_of_event	integer | Integer value that determines how important the event is, 0-100 with 100 being of the utmost importance
color_of_event	varchar | Stores values in a "#xxxxx" format so users can customize their calender

Figure 3: Database Tables