Design Justification - G.T.H.C

Game Tenting Help Center (GTHC)

This document serves to describe how the team justified the design of the software the way that this project is laid out.

Deciding Design

Website Design

In the beginning we were not certain whether we should create our own designs or use a pre-existing template. And if we were going to choose a pre-existing library, which library should we choose and for what reasons. We decided to choose a design that best fit these criterias

1. Flexible features that can be used on both React and React Native
2. Members of the team have experience with the design
3. Library most user friendly with the WIX calendar library we are using for the MVP of the product

Backend

We were unsure of whether to use a SQL or a NoSQL database. Our team members have experience building databases using PostgreSql and writing functions that could query data. However, we had heard about useful third party Databases like Firebase, Parse, Kinvey, etc.

Choice Ultimately Made

Material UI → Makes up the Icons, Containers, Nav, Organization

Material UI was our choice we ultimately made for the library that we intended to import into our design considerations.

Justification:

1. Flexible features that can be extended to many applications of use
2. Created by Google; therefore, open source code is reliable on multiple platforms
3. Teammate has experience working with Material UI before. Therefore, decreasing learning curve
4. Material UI easily compatible with other frameworks as well
5. Easy to work with → simple imports into our script
6. Clean designs in the library

7. Importing icons were key in our project

Alternatives/Tradeoffs:

1. React Toolbox
    a. React Toolbox Advantage
        i. React Toolbox library is written with style-components which is consistent with React Code
        ii. Most amount of documentation on React and ReactToolbox
    b. React Toolbox DisAdvantage
        i. It is not coupled with React, can be used with any 3rd party JS platform...important if we use Angular, Vue, etc. for further editions of the product
        ii. JSS is easier to mount, which is important with sharing and manipulation of data in our application
        iii. Team Members don't have experience with this platform
2. Materialize/ React -
    a. Materialize Advantage
        i. Support IE 10 and more browsers than Material UI
        ii. Teammate has experience with this design
    b. Disadvantages
        i. Less powerful and has less design options than Material UI

Assumptions:

1. Material UI is built by Google so it has performed multiple tests on Internet based platforms.
2. Material UI is the most popular React platform being used right now, so it probably is the best and encompassing what we are working on

Dependencies:

1. Material UI is compatible with few of the calendars we were deciding from. One of the Calendars had been built on Material UIs design, while the other was built on React Bootstrap. The functionalities of both calendars were the same. The Material Ui dependant one won!!

Firebase API →

The API for our backend was something we have been contemplating on. We ultimately came upon the decision to utilize Firebase our main database.

Justification:

1. Real time data synchronization with app and backend
2. NoSQl and data already managed by application
3. Commonly utilized database that has a lot of documentation on its use
4. Created by Google; therefore, open source code is reliable on multiple platforms
5. Secure database which was necessary for storing personal information and demanded by our clients
6. Consist of notification, messaging, and other api endpoints

1. Parse-Server
    a. Advantages
        i. Completely open source, not dependant on price model
        ii. Provides you a very complete range of prebuilt SDKs to plug into your app project
        iii. Comes with a plethora of features that offer a better user experience
    b. Disadvtanges
        i. It's not multi-tenancy App compliant
        ii. Does not have messages as a feature -> future edition of the ap
2. Postgre
    a. Advantages
        i. All of us have built a PostgreSQL database
        ii. Aman had already built out database for previous project
        iii. Data organized
    b. Disadvantage
        i. No Realtime database synchronization
        ii. Data is not visual organized for developers
        iii. Does not have API endpoints already implemented
3. Horizon
    a. Advantage
        i. You can build and deploy cross-platform, real-time web and mobile apps at lightning speed;
        ii. Unlike Firebase, you can build complex enterprise apps with Horizon, not just basic ones with limited functionality;

iii. eliminates repetitive boilerplate and tedious steps such as hand-writing CRUD endpoints, authentication, and session management

b. DisAdvantage

i. Horizon code base is not as stable or secure as Firebase

ii. Does not have notification functionality similar to Firebase

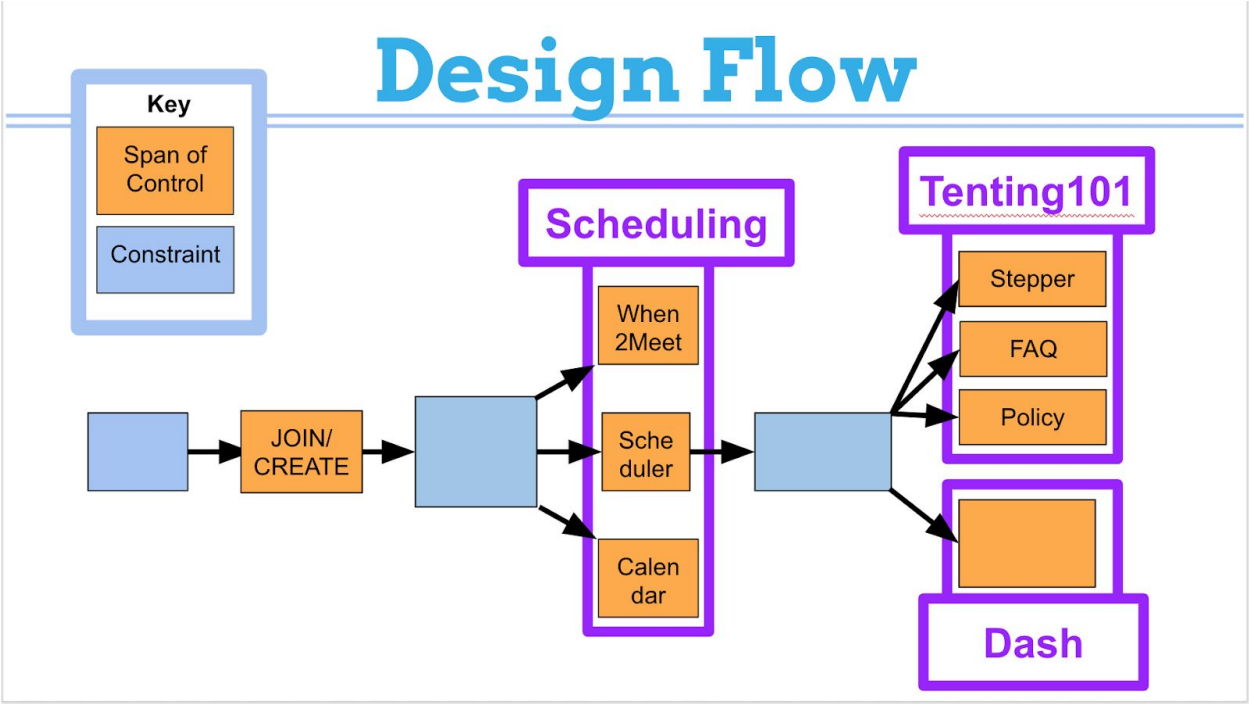
Assumptions:

1. People have experience with Firebase for later future development
2. Can build out notifications, messaging within app
3. Most secure platform

Dependencies

1. We had to integrate authentication and API endpoints to connect web app to the application.
2. Much more complex to add Notifications and messaging that Notifications may not make it in this product

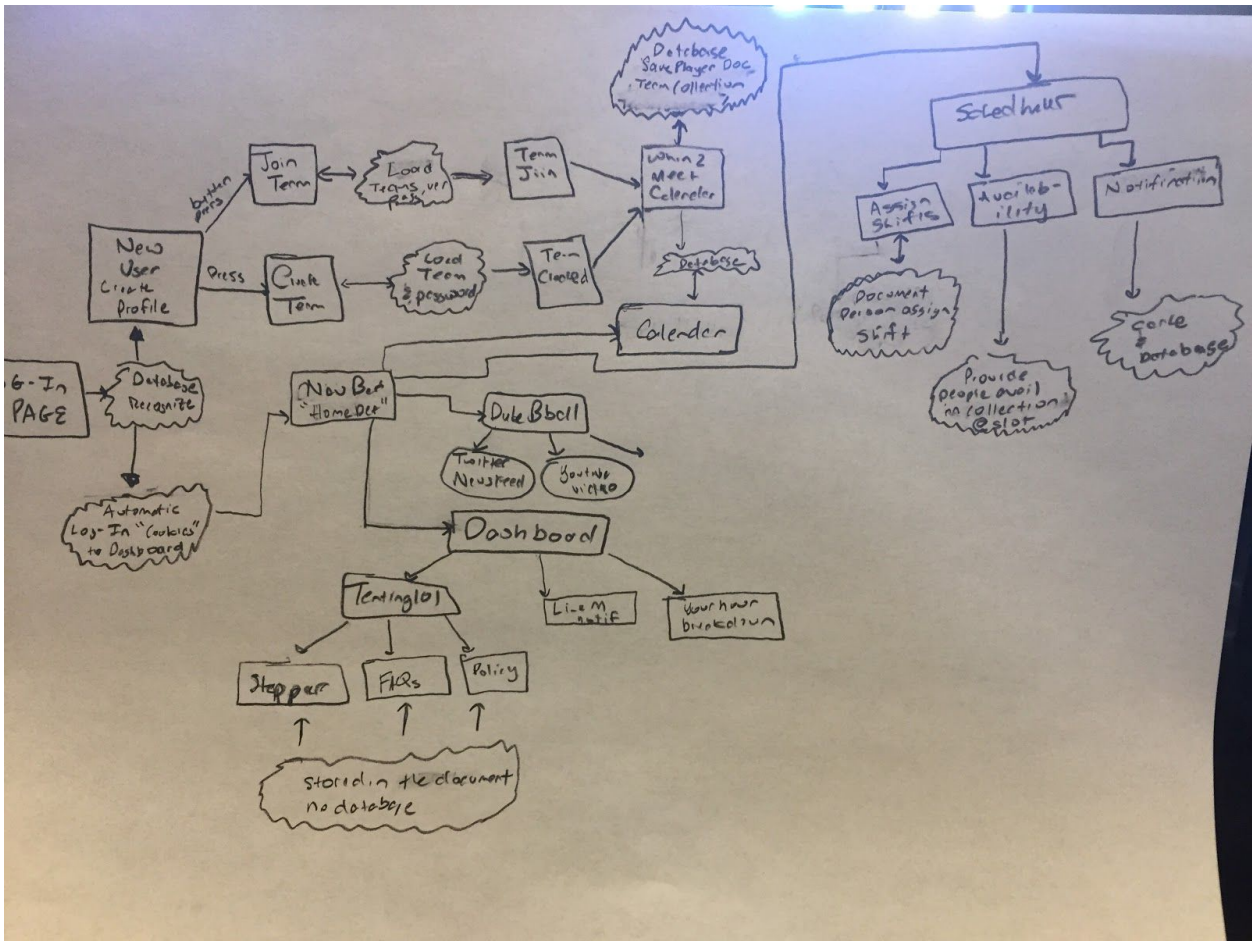
High Level Overview of Architecture



The Design Flow shows the application's breakdown into four key components. The first component is the "Log In" section where an individual can join or create a team. The second Component is called "Scheduling". Scheduling occurs once the user is done Logging In. The user inputs their Schedule into a When2Meet Calendar display. He/she has an option to edit that schedule in the scheduler portion of the application. In order to see the general tenting schedule for Most Users, the Calendar section displays this information. Also, users can change the shift and see who is available and not available at any given time. The third Component is the "Tenting 101" slide webpage, which provides a breakdown with hints/tips/ and information on what to do during different phases of tenting. The fourth component is the Dashboard. The dashboard will have info on upcoming shifts, line monitor notifications, and the Tenting 101 stepper feature.

Since we are behind schedule, we will primarily focus on building out the Scheduling and Tenting 101 feature, since we find these to be the most helpful and beneficial features for the user. Our goal is to primarily make the tenting process easier for new tenters. Therefore, providing tenters with a stepper with key information regarding black, blue, and white tenting is imperative. Also, we want to make scheduling easier for users. Therefore, we will be adding a feature that allows users to see who is available at any given time period. This will allow members to have a clear understanding of who to contact during last minute changes in their schedule. Also we will add features that allow the Captain to easily assign shifts to the rest of the team.

Lower Level Overview of Architecture



This diagram shows each step of the application and how it speak to the database. Steps in the application are represented in squares and database is represented in a jagged circle.

The Goals of this Lower Level Overview is to describe which components of this product will be directly speaking to the database. Our priority will be implementing the scheduler (calendar) feature and re-displaying the When2Meet portion of the application to the user. The Calendar's two main features are member availability and availability suggestions.The reason we are prioritizing these two features is to address Captain's difficulty with scheduling peoples shifts and to help address the issue people face in finding a replacement when they need to drop a shift last minute.