Software Design Specification

Live+

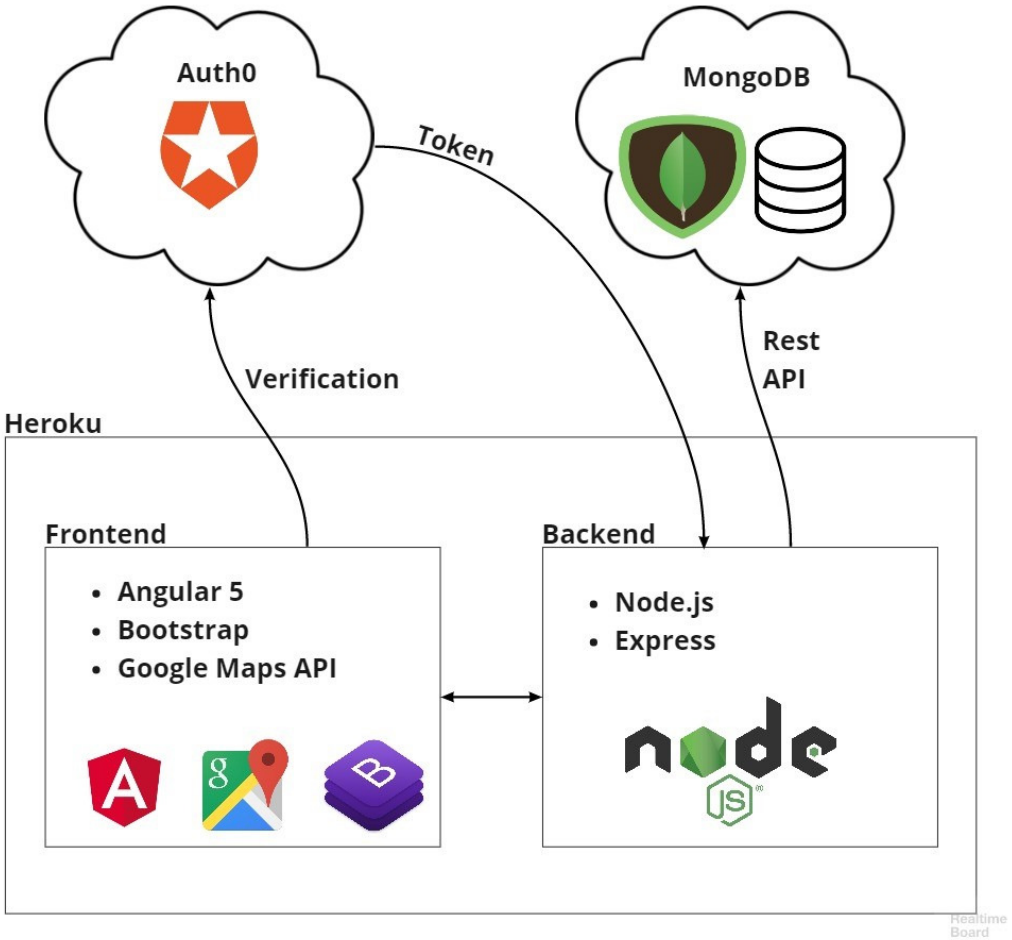
1. Project Description 3
2. System Analysis and Decomposition 4
2.1 Frontend 4
2.1.1 Angular 5 4
2.1.2 Bootstrap 4
2.1.3 Google Maps API 5
2.2 Authentication 5
2.2.1 Auth0 5
2.3 Backend 5
2.3.1 Express + Node.js 5
2.3.2 MongoDB 5
3. UML Diagrams 6
3.1 Class Diagram 6
3.2 Use Case Diagrams 7
3.2.1 Sign Up - Log in/out Model 7
3.3.1 Sign Up 9
3.3.2 Create Event 10
3.3.3 Messaging Service 11
4. Implementation Plan 12
4.1 User Interface 12
4.1.1 Main View 13
4.1.2 Create Event View 13
4.1.3 Event Info View 14
4.1.4 Profile View 15

1. Project Description

Live+ is a web application created so users can find/create generally local events to attend/host in a relatively short amount of time. This application uses a map that updates in real time with events. The application is centered around its events. Live+ is meant for ages thirteen and up and is accessible on a variety of web browsers.

User profiles must be created before using this application. The credentials for a user's account will be put into a third party authentication system. Once a profile is created a user may create, update, read, and delete events on the map. The user may also search for already created events by name or look at a map filtered by various categories.

2. System Analysis and Decomposition



2.1 Frontend

2.1.1 Angular 5

The frontend framework that we will be using for the application will be Angular 5. It provides a very nice way to separate logic from styling as well as provides tools to update our HTML dynamically. With Angular we will be able to separate our concerns and organize our application into different components nicely allowing for readability, reusability, and maintainability.

2.1.2 Bootstrap

In addition, for the front end we will be using Bootstrap for our CSS framework to allow for fluid design. Bootstrap will take care of a lot of the overhead required to design to

multiple devices. It comes as a sort of addition to raw CSS and removes a lot of the tedious styling that we would be dealing with just using CSS.

2.1.3 Google Maps API

In addition, we will be using the Google Maps API for displaying a map with our events drawn on top of it. This will serve as our means of displaying the current events to users so that they may visualize where events are taking place. The events will be displayed as markers on the embedded Google map placed within our application.

2.2 Authentication

2.2.1 Auth0

Auth0 is a third party authentication service that we will be using to handle user authentication. Users will be required to sign in via Auth0 and in turn Auth0 will provide a user a token which can be used to make calls to our backend (REST API). The use of Auth0 allows us to not have to worry about storing user credentials in our database and delegate this to a service specifically catered for this.

2.3 Backend

2.3.1 Express + Node.js

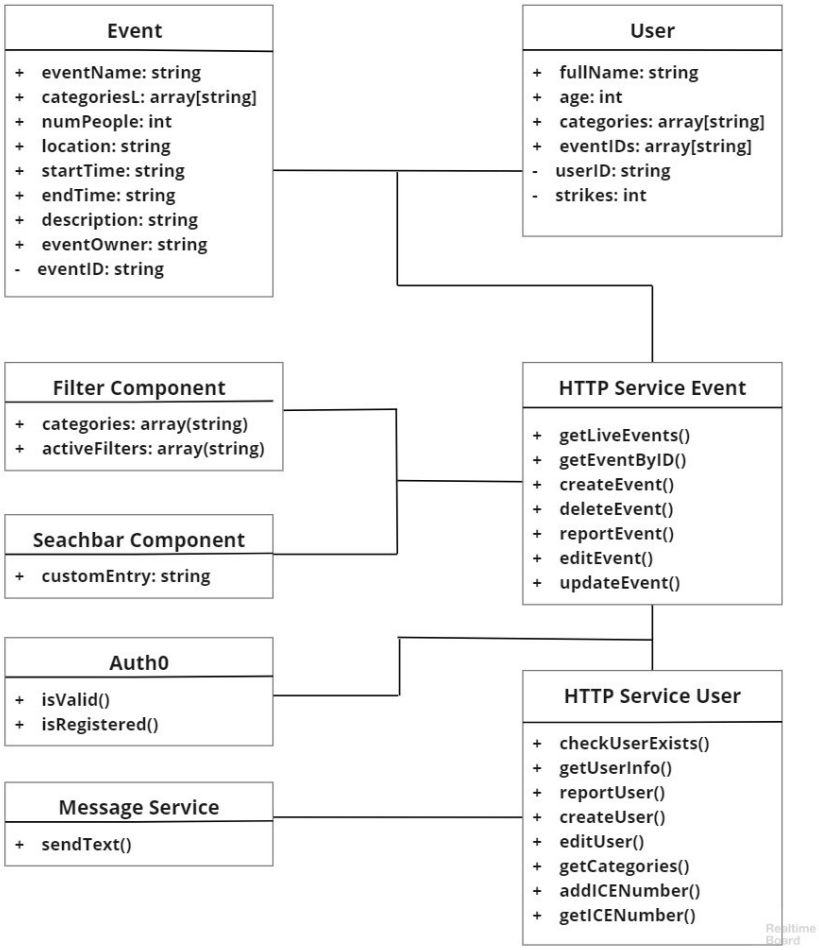
Express and Node.js will be used to serve our application. Express is a framework for Node.js and will allow us to quickly set up routes as well as serve content to those requesting it. This is where we will be setting up our REST API through the routes we define. This REST API must exist in the backend since anyone can change client-side code. The REST API will serve as a mediator for storing and retrieving information from our database. Only authenticated users (via Auth0) will be able to make requests to update or view information from our database.

2.3.2 MongoDB

MongoDB will be used as our database. One of the things it will be used for is to store information regarding our live events such as location, category tags, start time, end time, and more. In addition, we will be storing information regarding our users such as their name, categories related to their interests, as well as IDs related to the events that they have created.

3. UML Diagrams

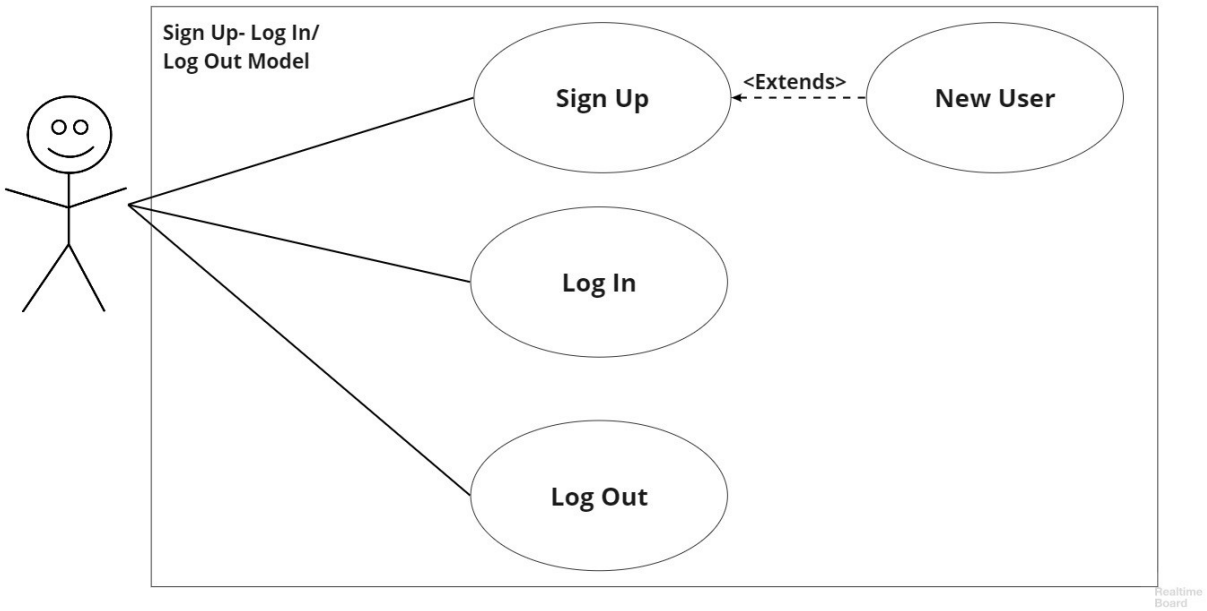
3.1 Class Diagram



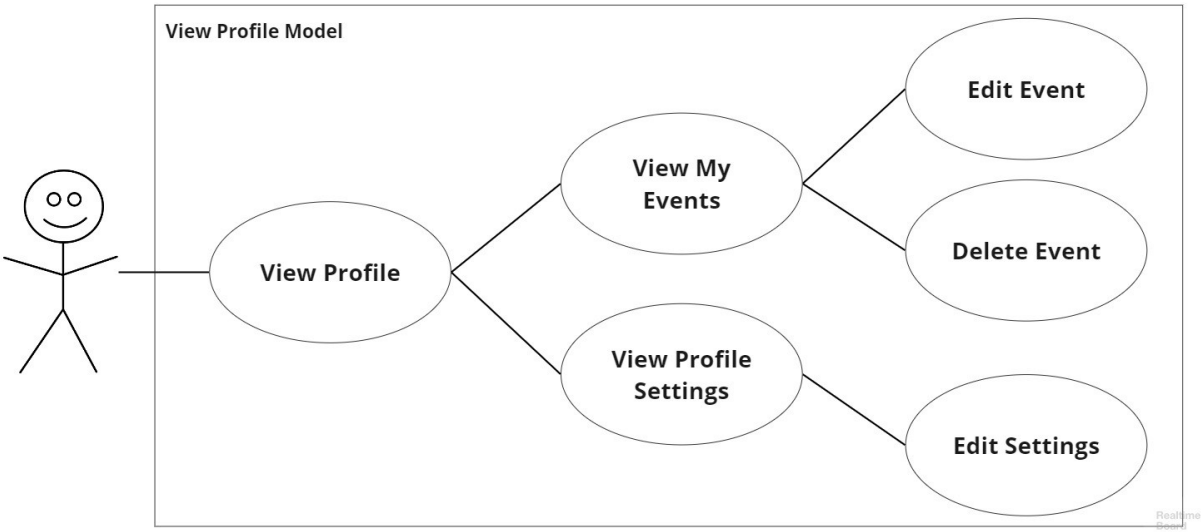
- Event: Collection of data entries for managing event marker information
- User: Collection of data entries for managing user profile information
- Filter: Tag method to filter out markers by an array of categories
- Search bar: Custom message to filter out markers by a string
- Auth0: Login in class that interacts with Auth0 login service
- Message Service: Service to send SMS message to custom emergency contact
- HTTP Service
    - Event: Methods to get/post/update information on live events
    - User: Methods to get/post/update information on a given user

3.2 Use Case Diagrams

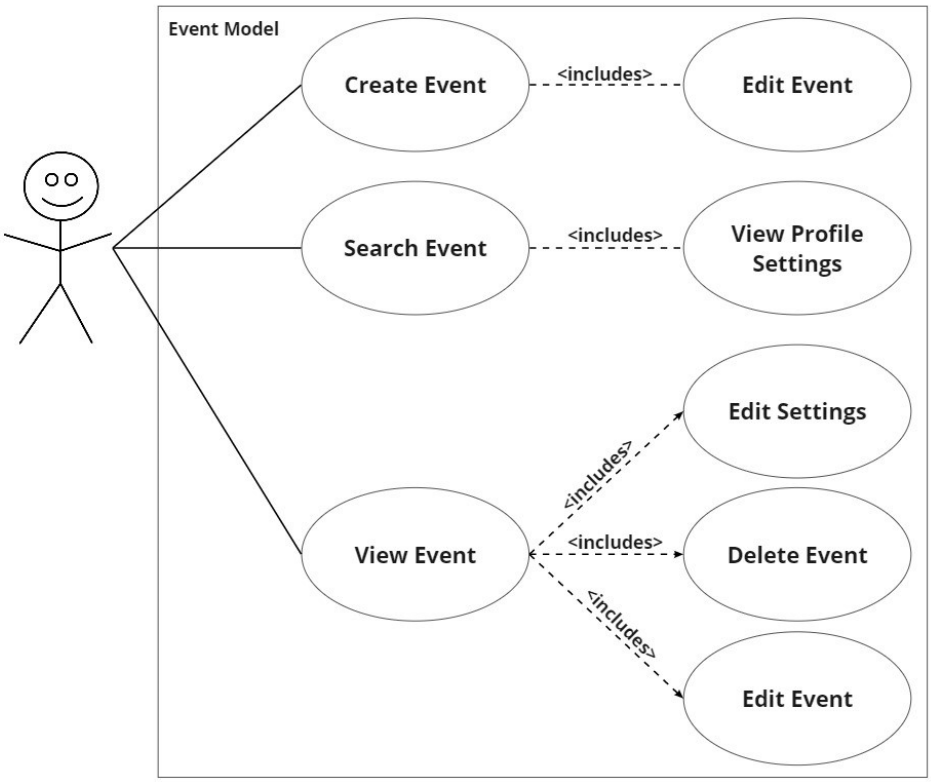
3.2.1 Sign Up - Log in/out Model



3.2.2 View Profile Model

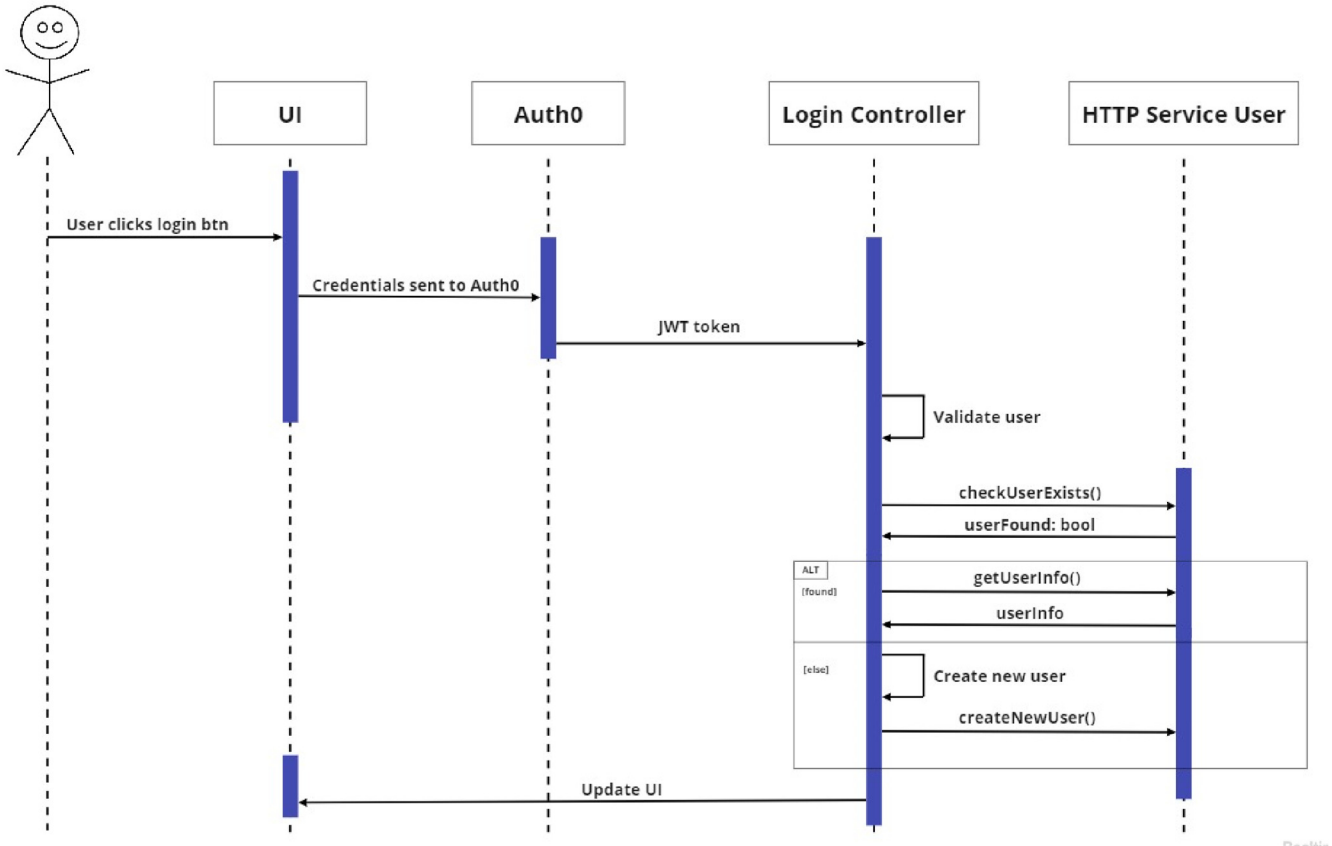


3.2.3 Event Model



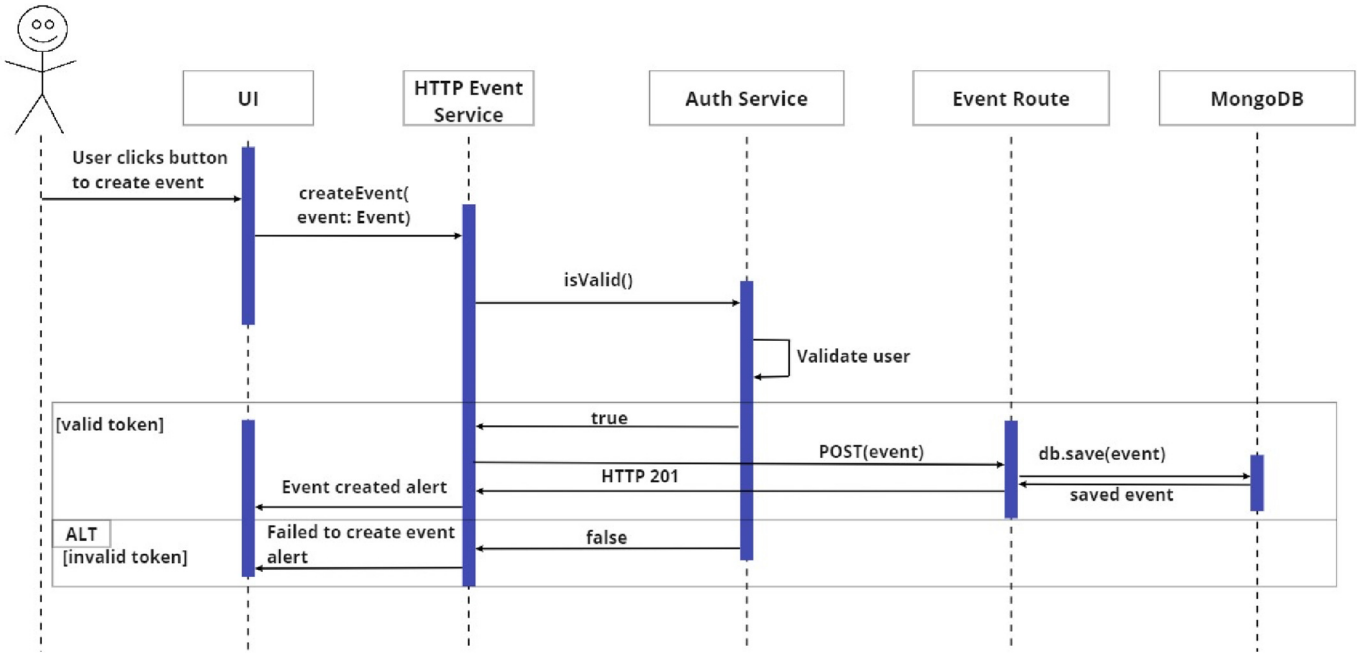
3.3 Sequence Diagrams

3.3.1 Sign Up



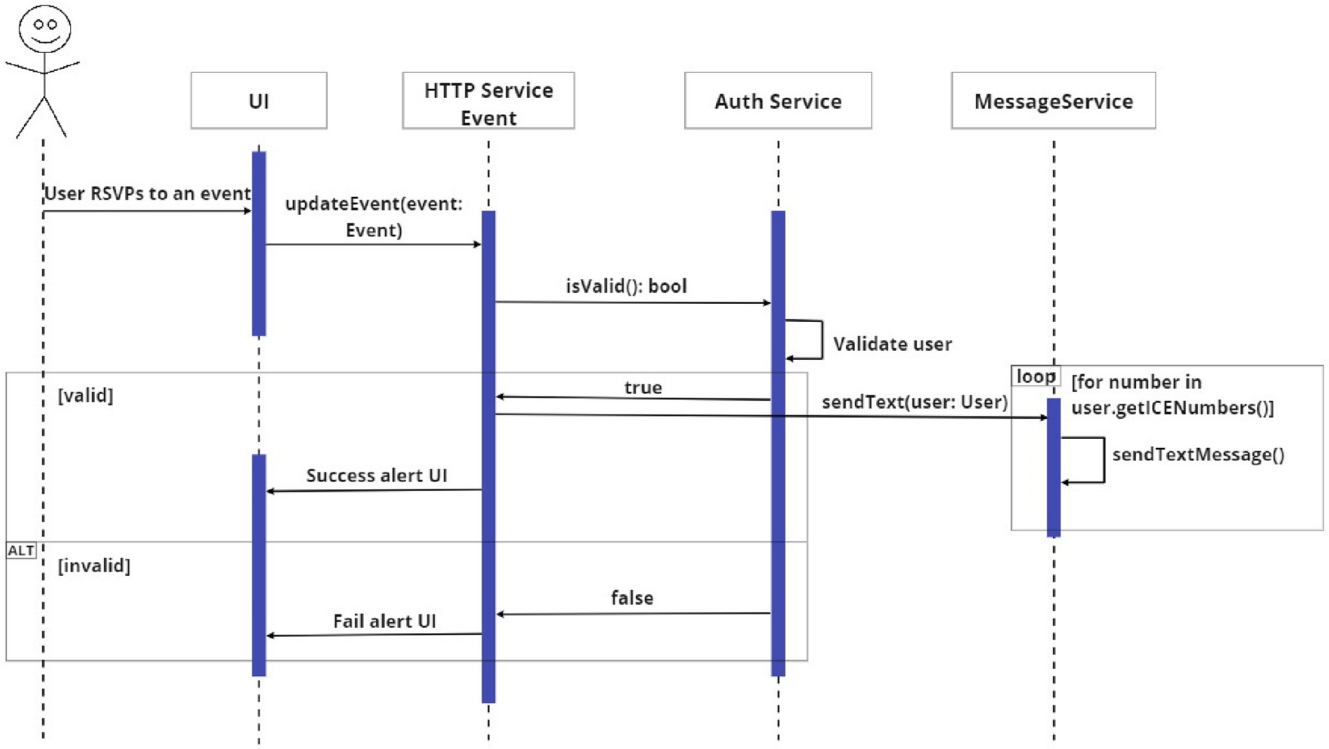
Description : When a user logs in they will be entering in their credentials through Auth0. This service will take care of all the necessary security concerns that come with storing/accessing login credentials. Once a user logs in, if successful, we immediately check the user's id against our database to see if he had already signed on to our application. If not, then the user will go through our account setup phase for the first. After completing the setup, or retrieving existing account information, that account information will be sent back to the UI accordingly.

3.3.2 Create Event



Description : Upon user submission to create a new event our HTTP event service will first communicate with our authentication service to check whether the user is valid. If the user is valid our HTTP event service will make a POST call to our event route to create the event that the user provided. Our route will be in charge of notifying MongoDB to update our events database to save the new event and will return a HTTP 201 response code if the event is saved. If our user is not validated then no event will be saved in our MongoDB. In both cases the user will be notified about the status of their event request.

3.3.3 Messaging Service



Description: When a user RSVPs to an event the HTTP event service will first check with the authentication service to see if the user is validated. similar to our previous sequence diagram. In the case that the user is valid then the HTTP event service will call upon the messaging service to send a notification text to all of a given user's in-case-of-emergency (ICE) numbers notifying them about where they will be and from what time. If a user is not validated then the request to RSVP will not go through and the messaging service will not send out a message to a user's emergency contacts. In both cases the user will be notified if their request has gone through.

4. Implementation Plan

During development, we will run the backend server locally for testing. The developmental database, however, will be hosted on Chris' computer and the other members will connect to it. For actual deployment, the instance of MongoDB will be hosted via Amazon Web Services (AWS). In regard to the programming languages, we will only need JavaScript and TypeScript, which is a syntactical superset of JavaScript developed and maintained by Microsoft. This forms one of the reasons for selecting the MEAN (MongoDB, Express, Angular, Node.js) stack. We do not have to grapple with a number of programming languages (such as JavaScript, PHP, Python, etc.) for frontend and backend development.

Our team members will collaborate through git for version control and GitHub for storing the remote repository. Chris has already created the repository for development (https://github.com/cvasqu09/Live) . In terms of branching, we have a master branch and a develop branch. The team members will work on individual features by cloning the development branch locally and creating a new sub-branch for that feature, such as "feature/GoogleMapsAPI." Each team member will make commits to his feature branch several times a week. Then, we can merge these changes into the develop branch after review once a week or once every two weeks.

Most of us will be using Windows 10, and Josh will be switching between Windows and Mac operating systems. However, this should not be a factor in development because the browsers we will use for testing will be identical. Other than that, we only need to have Angular, Node.js and npm installed on our machines (Express is installed as an npm package). All of these are used through command line.

At present, the developmental responsibilities are divided between the members as listed below:

- Shashi - Google Maps Backend, Component Data structures
- Dhruuv - User Interface
- Nathan - Component Data structures
- Josh - Auth0, Google Maps Frontend
- Chris - Services (messaging, initial Angular structure), (work with Josh for Auth0)

4.1 User Interfaces

Our application consists of four views the user can use to interact with the app.

4.1.1 Main View

- Navigation is intended to primarily be located on the top of the screen.
- Dropdown Categories section that allows users to easily filter out activities.
- Tags display below the search bar in order to visible see and remove tags that have been selected.
- Markers display live on the maps according to the user's selections
- Orange marker located on the bottom right allow users to add an new event

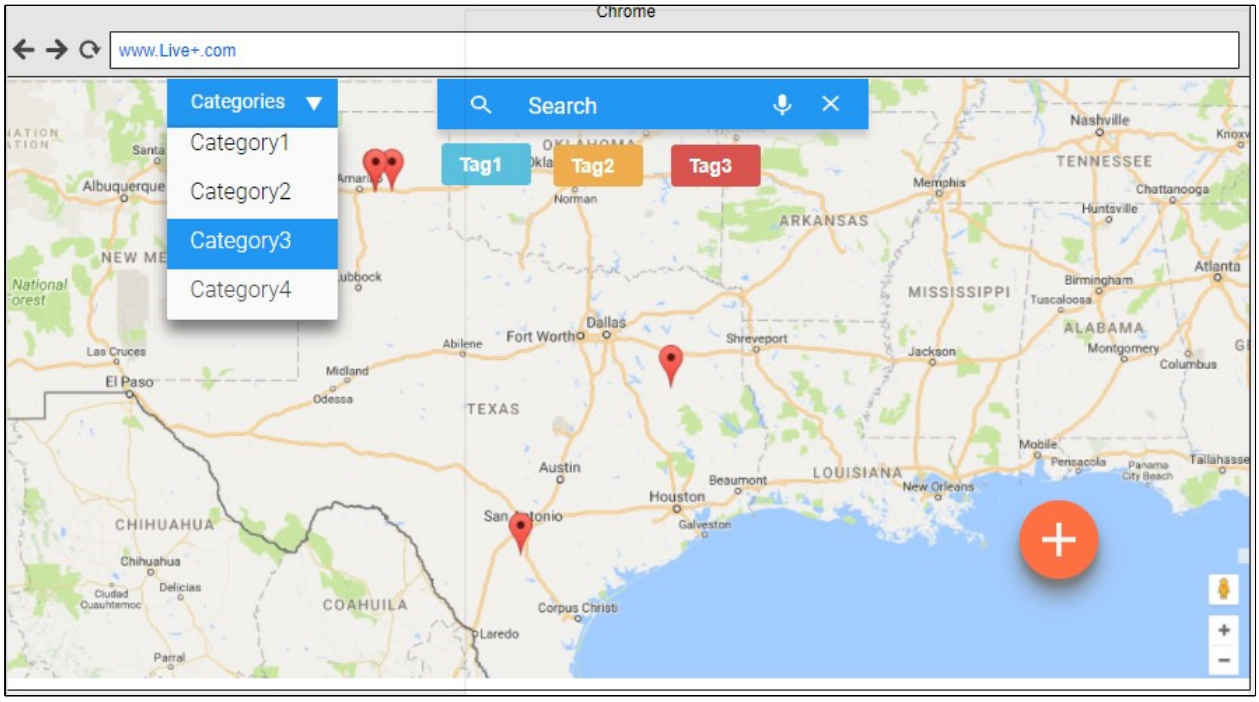


Figure 1: Main View

4.1.2 Create Event View

- This view is triggered by selecting the orange button on the bottom right
- Various options the user can select in order to post an event
    - Event Name, custom name by user
    - Tag s, selected from preset list
    - People Needed, max amount of people that can join
    - Start Time, time indicating when the event will start
    - End Time, time indicating when the event will end
    - Location, location of event
    - Description , additional event information can be set by user

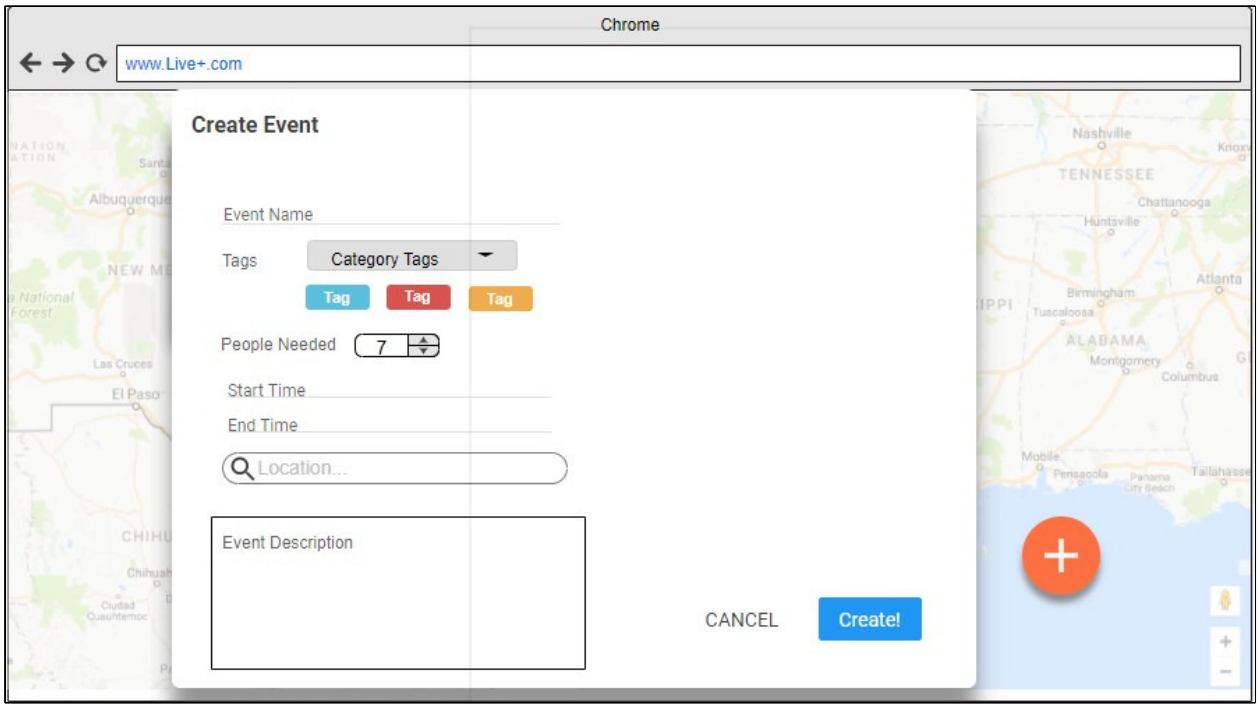


Figure 2: Create Event View

4.1.3 Event Info View

- Triggered by selecting a marker displayed on the map
- View event details that were set in 2.1.2.2
- Live updates on how many people committed to going to the event
- RSVP button to indicate that the user is interest in going to the event

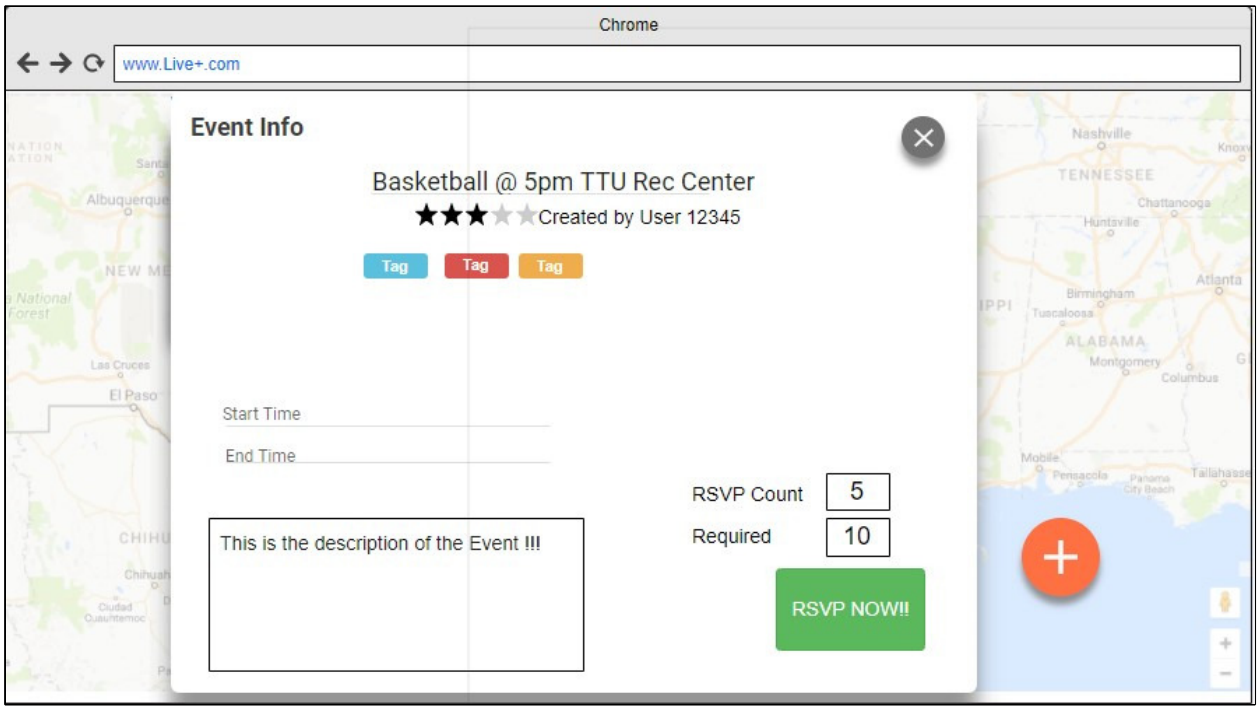


Figure 3: Event Info

4.1.4 My profile View

- Triggered by selecting my profile button in main view
- Shows user profile info
- User can update personal info and category tags
- List of user events with edit and delete functionality

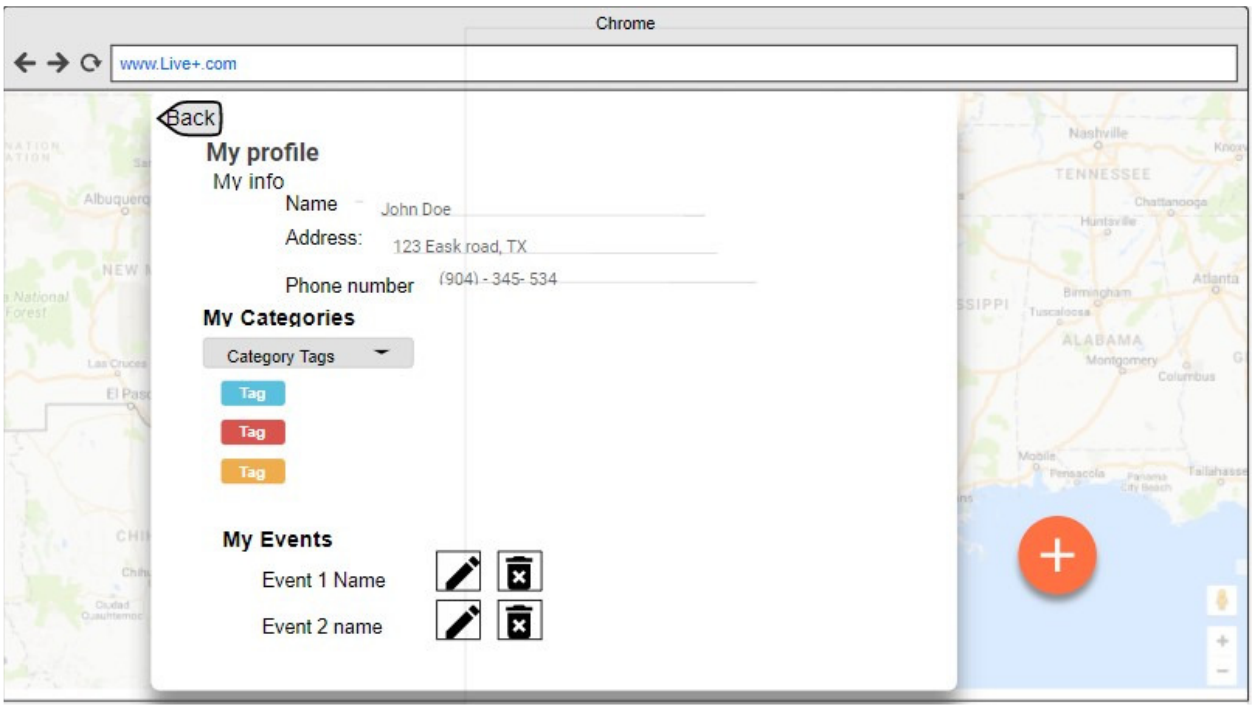


Figure 4: My profile view