UHN Overdose Response Network by Code The Change UHN Team

Team

1. Nicholas Kong (Lead)
2. Maxine Yee (Frontend)
3. Clarence Adrian (Frontend)
4. Wen Huang (Fullstack)
5. Andrew Choi (Fullstack)
6. Manu Koipallil (Backend)
7. Matthew Lee (Backend)

Disclaimer: Features described below are in brief and final product will only contain equal to or more than the features listed below for both Frontend and Backend.

Frontend

Overview: We are building a React Native Mobile Application to support drug users and responders to react to each other in times of need to reduce overdose deaths.

Vocabulary:

1. User: A person who is going to use the alarm prior to using drugs (e.g. opioid injection)
2. Responder: Also can be a User, but in this case he/she is available to receive any help requests notifications and can choose to accept/decline such requests
   Note* User and Responder can be the same person
3. Available: This is set to True when the User/Responder has logged in and indicated that they have Naloxone available
4. Help Request: A request initiated by a User to let nearby available Responders know that they need help, perhaps they are unresponsive and may be in danger of an overdose

Features:

1. Authentication
   - Users/Responders can sign up for an account using email, phone number, username, and password.
   - Users/Responders can login to their account using username and password.
2. Location GPS Checking
   - We ask the user/responder for their permission to use the device's GPS location.
   - Users/Responders can set/update their address using GPS location (latitude and longitude.
   - Users/Responders can change their address if the address conversion from latitude and longitude is not exactly correct (e.g. Include an apartment suite number).
   - Users/Responders can set a note for his/her Responders to see in times of need (e.g. Location of a hidden key).
3. Showing User information

- Users/Responders can view their own profile information (e.g. phone number, email address, Responders).

4. Maintaining Users and Responders Relationships
    - Users can add other Users/Responders as their Responders.
    - Users can remove other Users/Responders from their list of Responders.
    - Users can search for other Users/Responders.
5. Using the alarm
    - Users can change the duration of the alarm in 15 second intervals (+/-).
    - Users can start the alarm and it will count down.
    - Users will see a red ring when the alarm reaches 30 seconds or less.
    - Users can press Help Now to create a Help Request right away.
    - Users will see a flashing yellow and red background and the phone will vibrate and play ringtone (Only if Media volume is turned on) when the alarm reaches 15 seconds or less.
6. Alerting Responders
    - Responders that are in range, available, and is the respective User's Responder are the only Responders that will be notified when a Help Request is created.
    - Responders who are on the mobile app will see a popup notifying them when a Help Request is created.
    - Responders who are not on the app (either running in the background or closed) are notified via notifications. Once clicked, they will see the same popup.
    - Responders can choose to not accept a Help Request.
    - If responders accept the Help Request, the location of the User will be shown on a map and the address and note (if available) will be shown as well.
    - Responders can indicate that they have arrived at the scene
    - Responders can indicate that the Help Request has been resolved, either by calling 911 or administering Naloxone. Ultimately, this means the User is okay.

Architecture:

- Main frontend framework: **React Native**
    - React Native State management: **Redux**
    - Styling: Custom made components built off of **NativeBase**
    - Notification management: **Expo Push Notifications**
    - GPS location: **Expo Location**
    - Map rendering: **Leaflet.js**
    - HTTP client (for requests to the backend server): **Axios**
    - Token decode: **JWT**
- Development framework: **Expo**

Backend

Overview: We are creating a server and database that will handle a multitude of requests from hundreds to thousands of users at once. A metric database will be collecting data for analytical purposes. A database will maintain the users, help requests, authentication, and sends notifications.

Vocabulary:

1. Client: the mobile app illustrated above
2. Database: database where we store all the client data and allows features to work on the client side
3. Analytics Database: database where we store usage information for analytical purposes

Features:

1. Authentication
   - Given the correct username and password, the server will respond with a valid JWT token and refresh token. Clients can use this pair of tokens to make authorized requests
   - Given a correct refresh token and username combinations, the server will respond with a updated valid JWT token
2. Users
   - Maintain Available status for each User.
     - Recap: Logged in and indicated that they have Naloxone.
   - Signups will store the User into the Database (Username, password, email, phone number).
   - Logins will update User last login time into Analytics Database.
   - Maintain User's list of Responders.
   - Maintain User's latest logged in device for pushing notifications.
   - Maintain latitude and longitude last set by User.
   - Starting alarm will store the start and end times into Analytics Database.
3. Help Requests
   - Maintain all help requests ever created.
   - Allow Responders to be assigned to each help request, up to a maximum of 6.
   - Allow Responders to update the status of each help request (e.g. Taken, Arrived, Resolved)
4. Notifications
   - After each help request is created, server will send notifications to the User's Responders that are Available and in range.
5. Analytics
   - Relevant information is logged and stored in Analytics Database that can be retrieved on a daily basis via a CSV dump onto Google Drive.

Architecture:

- Main backend framework: **Node.js** with **Express.js**
  - Authentication: **JWT** with **refresh token, hashed** password on Database
    - Hashing: **bcrypt**
  - Available status management (Storing and Using): **Redis**
  - Notifications: **Expo Server**
  - Database: **Mongodb** with **Mongoose**
  - Analytics Database: **PostgresQL** with **Knex.js**
  - Logging: **morgan**
- Development and Test: **Mocha** and **Chai**

Deployment

Backend server is deployed to **AWS EC2 CentOS 7** instance hosted on **ca-central-1**
Only ports 22, 80, and 3000 are open along with SSH access with private key