University of Arizona

ECE 413 Final Project

Tim Le, Ashton Rowe, and Christopher Westerhoff

ECE 413

Dr. Salehi

7 December 2022

Table of Contents

1.0 Project Description

1.1 Frontend

The frontend of this project is implemented using html and css. There are currently 8 html files for each page the user can go to. It first starts index.html which is the home page with a navigation menu with 4 buttons that lead to sign-up, log-in, team & project information, and references. Upon successful login the user is redirected to display.html where they can view their device data. Additionally, on each webpage there is a navigation bar at the top to access different pages on the website. From display.html the user can sign out, manage devices, and reset their password. To add a device the user will have to type into 3 text fields for device ID, device name, and device token then click a button to submit. In terms of css we only have 2 css files for styling our html files which are site.css and style.css. Site.css styles the index.html setting padding, margins, text color, and nav-wrapper. Style.css is used on the rest of the html files to add a border around divs and padding/margins.

1.2 Backend

The backend of this project is implemented using node.js, express, and mongoDB. The routes are handled using express to allow the webpage to communicate with the server. Inside a router.post the command will access the database by using the functions findOne and save. Saving to the database is done by a model file customer.js which creates a schema to be saved onto the database. Additionally, the route commands handle passwords with bcrypt hashSync and compareSync. X-authentication is also handled within the routes by creating an authentication token and saving it locally. These routes are then used in the web pages javascript files to enable the buttons to have functionality. The two things saved to the database are accounts and registered devices. For graphing the data from the IoT device we used chart.js.

1.3 Embedded Device

The core functionality of the embedded device came from the spo2 example for the MAX30102 heart rate sensor. This was simplified to remove bloat as well as modified to include functions necessary for web use. Particle Cloud API was utilized to send heart rate and spo2 data from the particle device to the Particle Cloud, and then webhooks integration transferred the data to the server "hook" endpoint. The Arduino wire library was used to control external LEDs that signaled various aspects of the test (blue when the test is running and green when data is sent to the server). A state machine and logic statements were used to implement the delay between tests, with the delay() and millis() functions being especially useful.

2.0 Files Description

- *Note: Here we will go through each section in our code with a brief description for each file*

2.1 Javascript

Device.js - This file deals with adding and deleting devices to a user's account. It will get the values from the text boxes found in device.html, clearing them afterwards. In a request it will ask the database to either add/remove the device in the user's device list.

Display.js - This file deals with displaying the sensor data to the user. This includes displaying the two charts for the daily summary, as well as the maximum, minimum, and averages for the weekly summary.

Login.js - This file deals with the login functionality for the user. This includes getting the text from the text boxes, and comparing them to what is stored in the database.

Reset.js - This file deals with the user functionality of changing their password. This checks if the new password is valid and replaces what is stored in the database (old password) and replaces it with their new password.

Signup.js - This file deals with users creating a new account. This file checks if the username and password are valid. If they are valid it creates a new entry into the database with their username (email) and stores a hashed password into the database.

2.2 Html

Device.html - This page is where the user will be able to add devices, remove devices and view their registered devices. The html consists of a navigation bar at the top with text fields and buttons to use the functions.

Display.html - This page is shown after the user logs in which allows the user to view the data collected from the device. The user chooses a device and can view weekly or daily data with the aid of text fields and buttons. The daily data is shown as a graph while the weekly view just displays the data in text form (max/min/avg heart rate, max/min/avg oxygen).

Index.html - This is the homepage with a navigation menu made of buttons that leads to sign-up, log-in, team/project information, and references.

Login.html - This is the login page where the user can log into an account. There are 2 text fields where the user can input a username and password then click a button to login. Upon a successful login the user will automatically be redirected to display.html, otherwise an error will appear of incorrect email/password.

Password.html - This page allows the user to reset their password only if they are logged in. There are 2 text fields, one for the new password with the other to confirm the new password and a button to reset the password.

References.html - This page can be reached by a button on index.html and it displays all the references used in the project

Signup.html - This page allows the user to sign-up using an email address and a password. The password must be strong as if there are any issues an unordered list will be populated with error messages to ensure the password is strong. Upon successful creation of an account the user will be redirected to the login.html.

Team.html - This page can be reached by a button on index.html and it displays a basic description of the project and the team behind it.

2.3 CSS

Site.css - This file has some styling for our webpage. Some of the styling that is included in this file is setting background colors for the index.html page. We also configure our font and padding for the body tag in this file. As well as text color and setting our nav-wrapper (height and margin-bottom).

Style.css - This file has some more styling for our webpage. The styling in this is setting up our "card" which puts a box around a div with class card as well as our chart-container which sets the width, height and margin of our charts.

2.4 Routes

Account.js - This file contains all the route information related to our user's account. This includes the login/signup functionality of our program, as well as changing the password for the user. We can validate the password to make sure that it contains a lowercase, uppercase, number, and special character, as well is greater than the length 6. We salt the text password in this file then store the hashed password into the database for greater security.

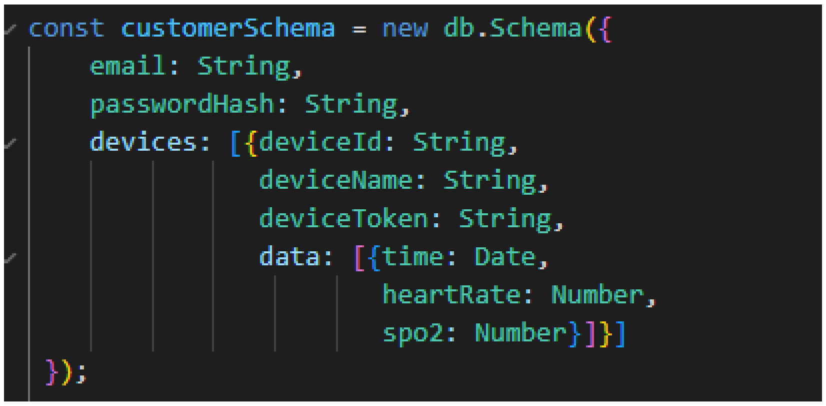
Devices.js - This file contains all the route information related to our device. Which includes adding/removing devices to a user's list, as well as getting the data that is stored in a device's data array.

Hook.js - This file is in charge of receiving the data from the particle board. In this file we receive the information from the particle board and store it into the database. When storing the data it goes through the database storing the information with devices that have the same Id.

Index.js - This file gets the home page, such that when the web page is first loaded it loads the index.html file for the user.

2.5 Model

Customer.js -This file sets up our database schema to hold all important information which can be seen Figure 1. The database schema stores the email (username), the hashed password, and an array of devices. Each device in the array has an Id, name, token, and data. The data for each device is an array which stores information received from the particle board which contains the fields: time, heart rate, and oxygen saturation.



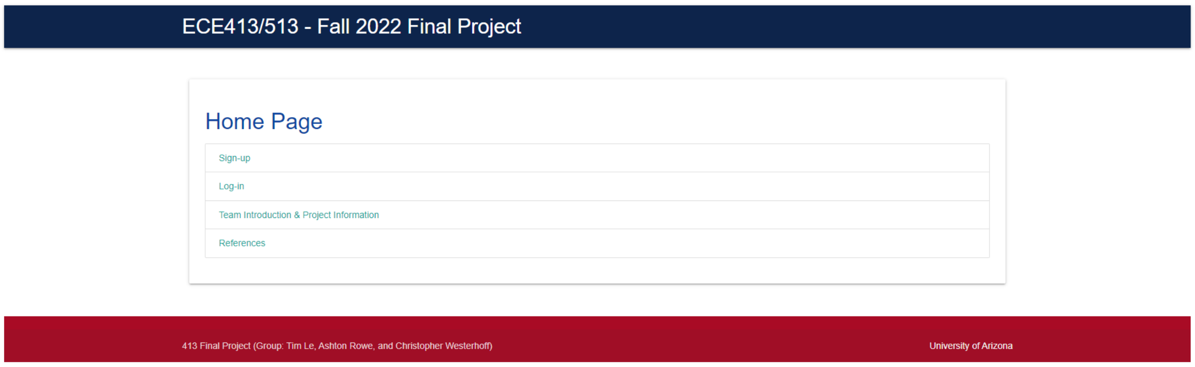
```
const customerSchema = new db.Schema({
    email: String,
    passwordHash: String,
    devices: [{deviceId: String,
               deviceName: String,
               deviceToken: String,
               data: [{time: Date,
                       heartRate: Number,
                       spo2: Number}]}]
 });
```

Figure 1: Code setting up the customerSchema

3.0 Results

- This section will include screenshots of the project with a description of what each screenshot is showing.

3.1 index.html


ECE413/513 - Fall 2022 Final Project

Home Page

Sign-up

Log-in

Team Introduction & Project Information

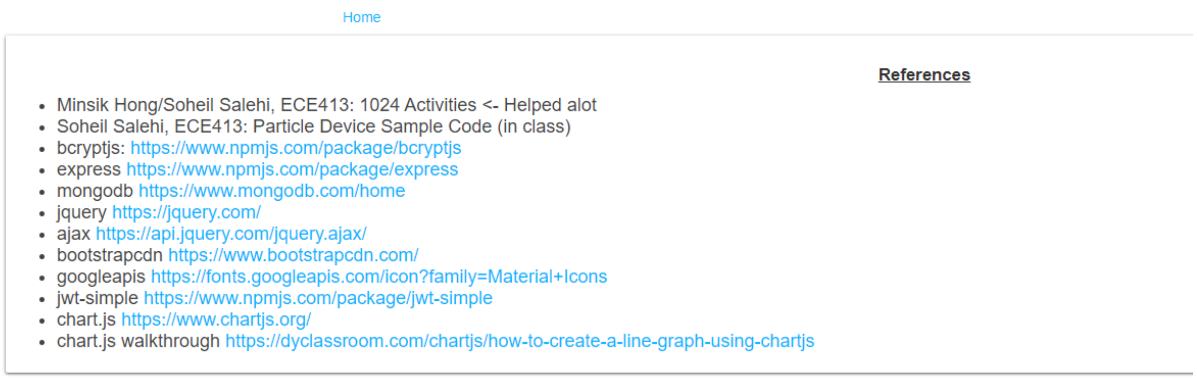
References

413 Final Project (Group: Tim Le, Ashton Rowe, and Christopher Westerhoff)

University of Arizona

Figure 3.1: *In this screenshot we see the index.html page, which acts as the home page for this project, and is the first page loaded to the user. From this page the user can navigate to where they wish to go: Sign-up, login, Team Introduct & Project Information Page, or the references page.*

3.2 references.html

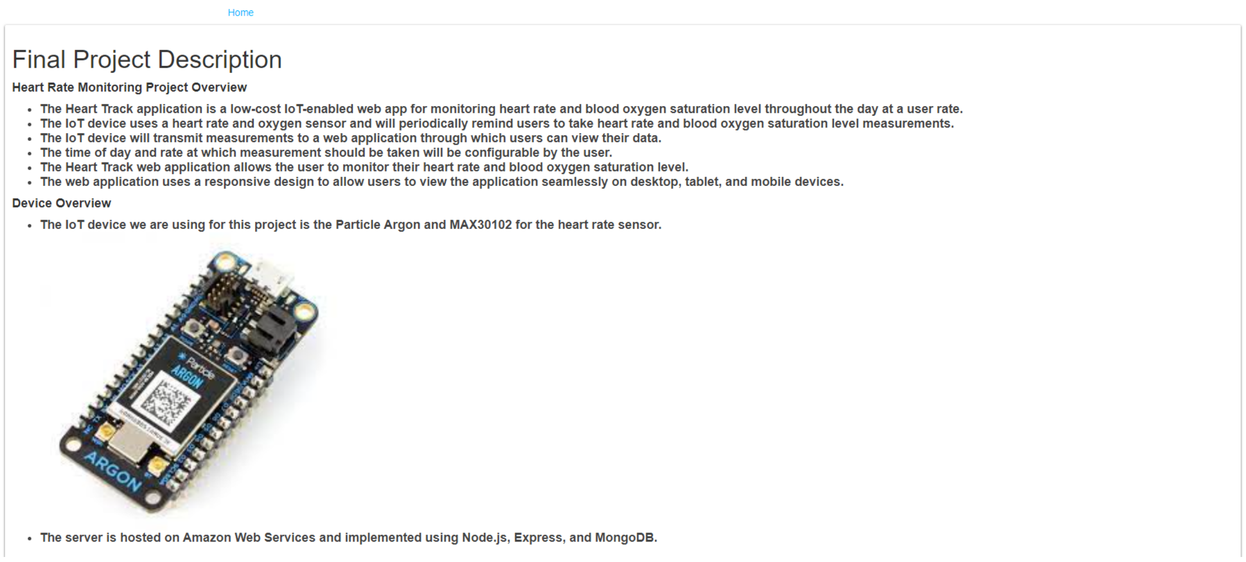

Home

References

- Minsik Hong/Soheil Salehi, ECE413: 1024 Activities <- Helped alot
- Soheil Salehi, ECE413: Particle Device Sample Code (in class)
- bcryptjs: https://www.npmjs.com/package/bcryptjs
- express https://www.npmjs.com/package/express
- mongodb https://www.mongodb.com/home
- jquery https://jquery.com/
- ajax https://api.jquery.com/jquery.ajax/
- bootstrapcdn https://www.bootstrapcdn.com/
- googleapis https://fonts.googleapis.com/icon?family=Material+Icons
- jwt-simple https://www.npmjs.com/package/jwt-simple
- chart.js https://www.chartjs.org/
- chart.js walkthrough https://dyclassroom.com/chartjs/how-to-create-a-line-graph-using-chartjs

Figure 3.2*: In this screenshot we can see the references that were used in the project. The user can go back to the home page by clicking on the home link at the top of the page. The User can also click on the links for our references if they wish to look into them.*

3.3 team.html


Home

Final Project Description

Heart Rate Monitoring Project Overview

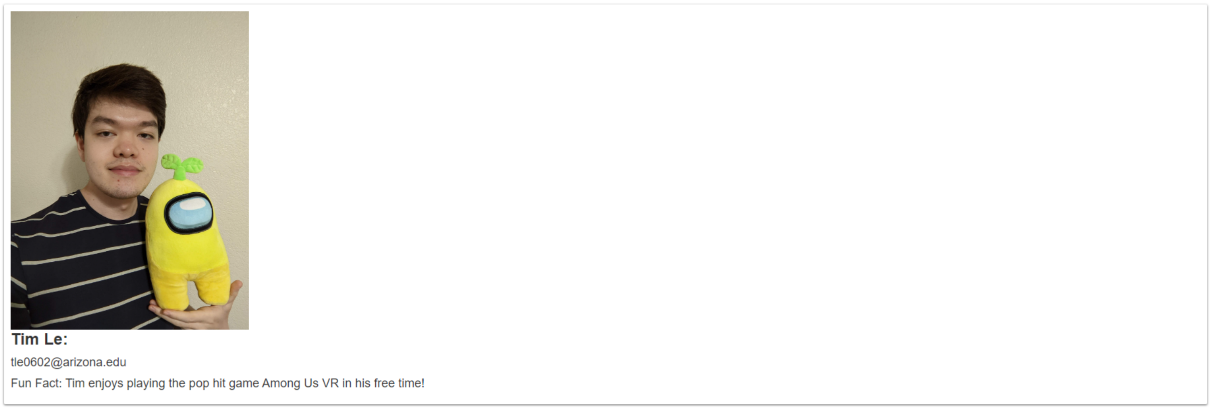
- The Heart Track application is a low-cost IoT-enabled web app for monitoring heart rate and blood oxygen saturation level throughout the day at a user rate.
- The IoT device uses a heart rate and oxygen sensor and will periodically remind users to take heart rate and blood oxygen saturation level measurements.
- The IoT device will transmit measurements to a web application through which users can view their data.
- The time of day and rate at which measurement should be taken will be configurable by the user.
- The Heart Track web application allows the user to monitor their heart rate and blood oxygen saturation level.
- The web application uses a responsive design to allow users to view the application seamlessly on desktop, tablet, and mobile devices.

Device Overview

- The IoT device we are using for this project is the Particle Argon and MAX30102 for the heart rate sensor.

- The server is hosted on Amazon Web Services and implemented using Node.js, Express, and MongoDB.

Figure 3.3.1: *In this screenshot we can see the project description part of the team.html page. This part of the page gives the user a brief description of what the project encompasses as well as how we implemented it. At the top of the page the NAV bar includes a path to go back to the home page.*

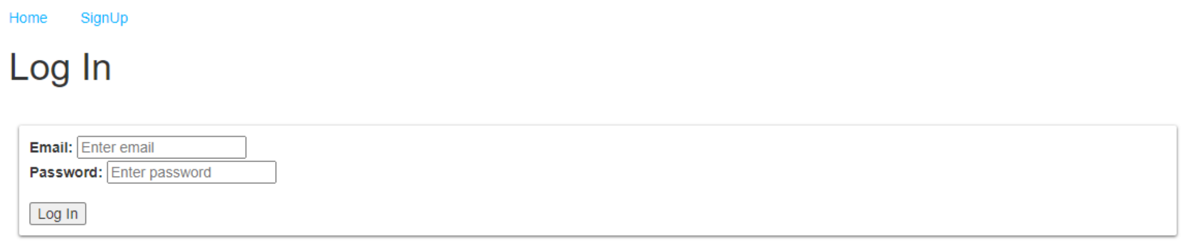

Tim Le:

tle0602@arizona.edu

Fun Fact: Tim enjoys playing the pop hit game Among Us VR in his free time!

Figure 3.3.2: *In this screenshot we can see a team introduction box for one of our members. Every member of the team has their own box which includes a picture of themselves, their name, email address, and a fun fact about themselves.*

3.4 login.html

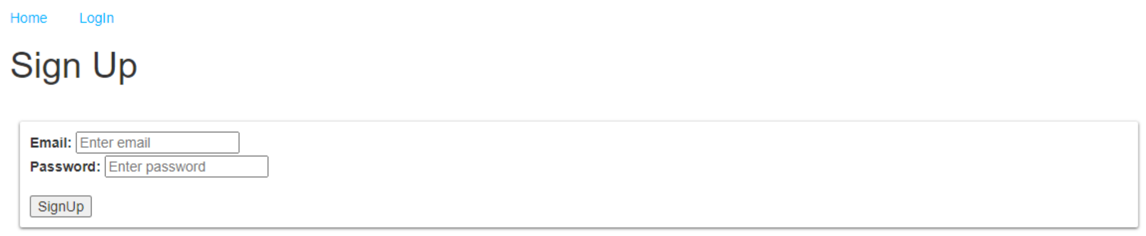


Log In

Email: Enter email
Password: Enter password

Figure 3.4: *In this screenshot we can see the log in page. The user has two navigation options, either go back to the home page (index.html) or to the signup page to create a new account. Once the user enters valid credentials it will bring the user to the account home page.*

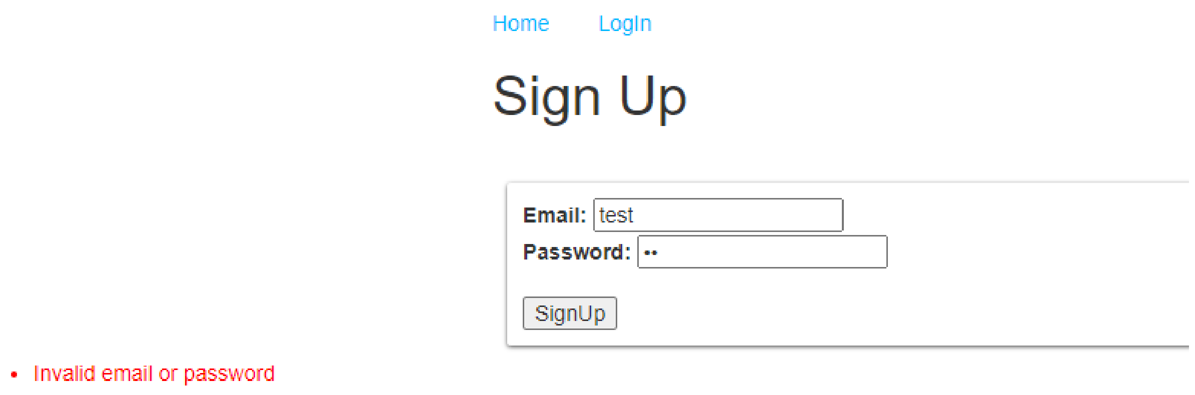
3.5 signup.html



Sign Up

Email: Enter email
Password: Enter password

Figure 3.5.1: *In this screenshot it shows the signup page where the user has to enter a valid email and password to create an account. If the user enters valid credentials it will bring them to the login page to login into their new account.*



Sign Up

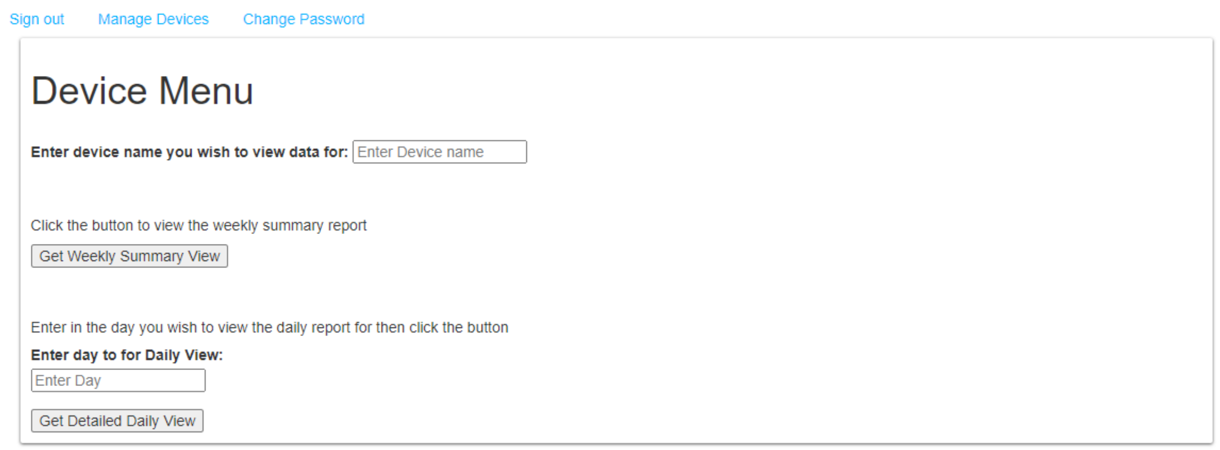
Email: test
Password: ••

- Invalid email or password

Figure 3.5.2: *In this screenshot it shows what happens when the user does not enter a valid email or password. The user has two navigation options, either go back to the home page (index.html) or to the login page. The system searches if the email has a @(something) and if the*

password contains the following: a lowercase letter, an uppercase letter, a special character, and has a length longer than 6 characters.

3.6 display.html



Device Menu

Enter device name you wish to view data for: Enter Device name

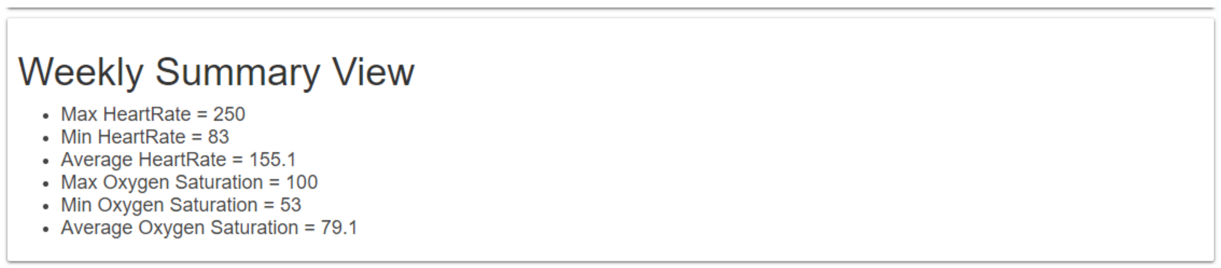
Click the button to view the weekly summary report

Enter in the day you wish to view the daily report for then click the button

Enter day to for Daily View:

Enter Day

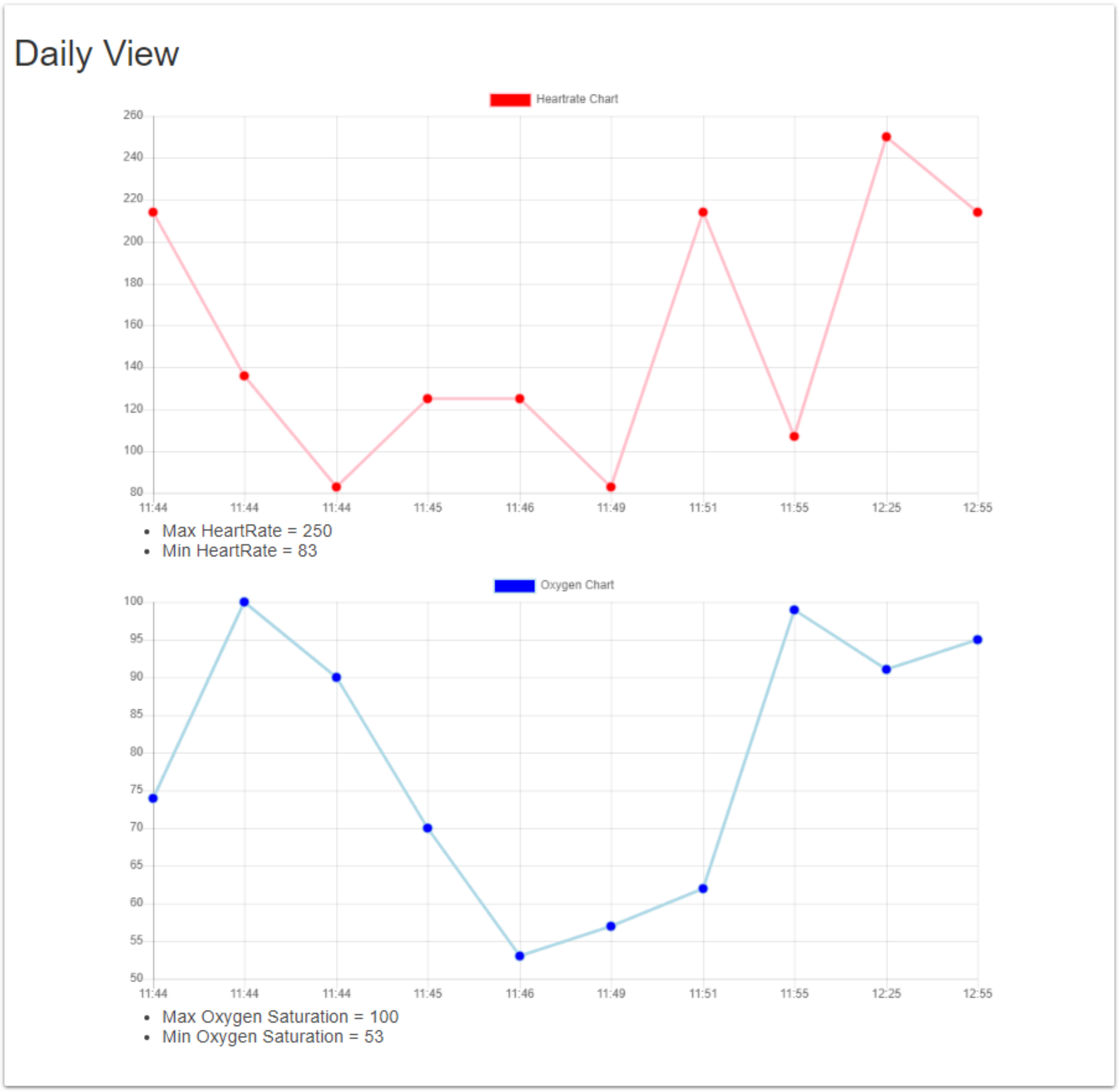
Figure 3.6.1: *In this screenshot the display.html page, the page that is shown to the user immediately after logging into their account. The user has a couple navigation options: sign out, which signs the user out of their account and brings them back to index.html, manage devices page which brings the user to device.html which allows the user to edit the devices on their account, as well as change password, which brings the user to password.html where they can change their password. The functionality of the page requires the user to enter what device they want to view data for, then gives them the option to get the weekly summary report or to enter a day to view that day's selected daily report.*



Weekly Summary View

- Max HeartRate = 250
- Min HeartRate = 83
- Average HeartRate = 155.1
- Max Oxygen Saturation = 100
- Min Oxygen Saturation = 53
- Average Oxygen Saturation = 79.1

Figure 3.6.2: *In this screenshot, we can see the weekly summary view. It will show the max, min, and averages for the heart rate and oxygen saturation over the course of the last 7 days.*

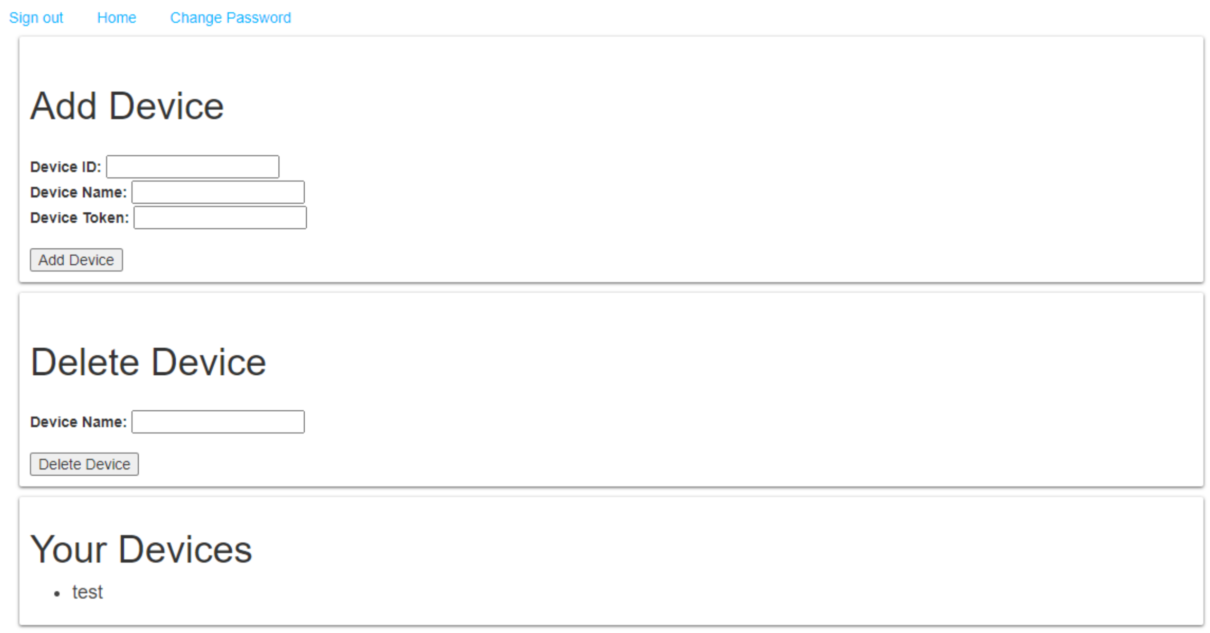

Daily View

- Max HeartRate = 250
- Min HeartRate = 83

- Max Oxygen Saturation = 100
- Min Oxygen Saturation = 53

Figure 3.6.3: *In this screenshot, we can see the daily view. It will plot the data points given in the device data array for the given day entered by the user. The y access represents heart rate in (beats per minutes) and oxygen saturation is in a percentage. The x axis is the time at which the recording was registered (hh:mm) in 24 hour time.*

3.7 device.html


Sign out Home Change Password

Add Device

Device ID:
Device Name:
Device Token:

Add Device

Delete Device

Device Name:

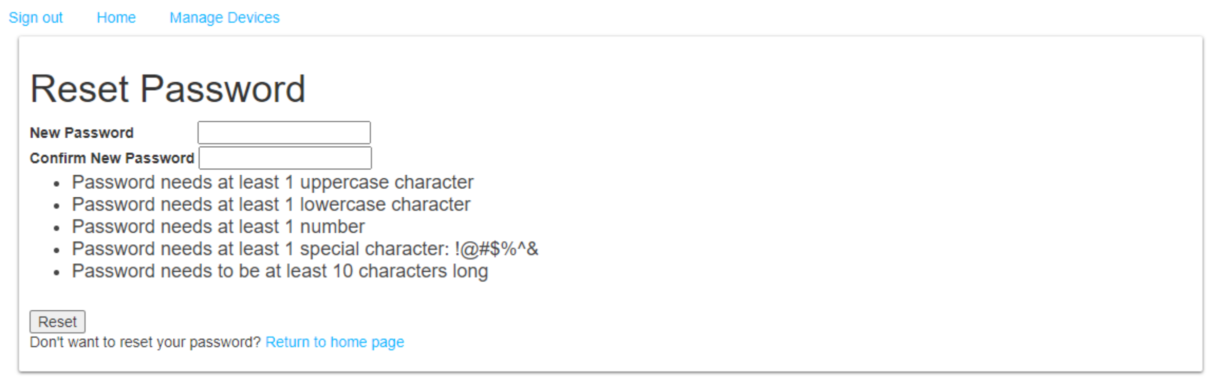
Delete Device

Your Devices

- test

Figure 3.7: *In this screenshot it shows the device page. The user has a couple navigation options, sign out which brings them back to index.html (and logs them out of the account), go back to the home page (now display.html) or go to the change password page.In this page a user can add/delete a device attached to their account. The user can also view their devices that are currently added to their account by device name (in this case the device "test" is added for demonstration).*

3.8 password.html


Sign out Home Manage Devices

Reset Password

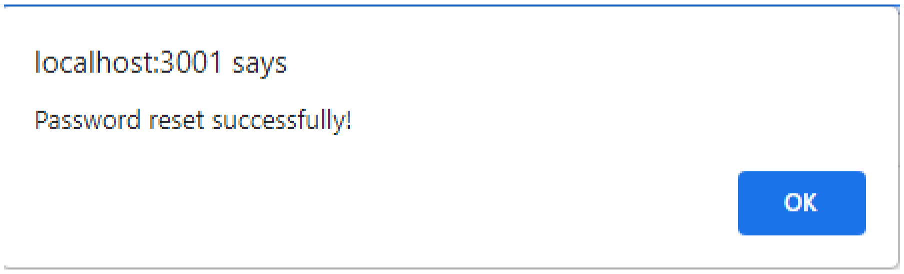
New Password
Confirm New Password

- Password needs at least 1 uppercase character
- Password needs at least 1 lowercase character
- Password needs at least 1 number
- Password needs at least 1 special character: !@#$%^&
- Password needs to be at least 10 characters long

Reset

Don't want to reset your password? Return to home page

Figure 3.8.1: *In this screenshot it shows the change password page. The user has a couple navigation options, sign out which brings them back to index.html (and logs them out of the account), go back to the home page (now display.html) or manage devices (device.html). The change password has the user enter their new password, and confirm it by typing it again. The conditions are listed below the two text boxes and disappear as the user meets the conditions. After meeting all conditions this user can reset their password.*



localhost:3001 says

Password reset successfully!

Figure 3.8.2: *In this screenshot it shows what happens when the user successfully resets their password. It brings up this notification box and logs the user out, bringing them to the login page.*

4.0 Lessons Learned

The lessons/topics that we learned by completing this project can be seen in the list below.

- Using schemas to store information into the database was a big lesson learned from this, we knew how schemas worked but configuring it to make sure the data you need is stored in the database correctly. Designing our schema to work with storing a user's account information as well as storing the sensor information was a big learning experience on working with schemas.
- Using MongoDB and databases, the lesson we took away from this was how and why using an external database is important and how to implement one. Before this project we all roughly knew a database stored data in it, but throughout the course of the project we began to understand why the database is so important and how we can implement a database into our design. A big lesson learned here was how to store and encorporate databases into a project design.
- Using Jquery to help pull information from the database, this was a lesson learned as we struggled getting the information into the database as previously mentioned. Part of the problem was figuring out if we stored the information correctly into the database. Part of the lesson learned here was learning how to stringify and parse JSON files correctly, to get the information we want from the database.
- Using third party APIs was a big part of this project, the lesson learned here was not only how to implement an external API to help in our design process. We worked with a couple of external APIs in our design and figuring out how to use them can extend in the future with other APIs that we may encounter.
- Developing particle firmware, a big part of the project was getting the particle device to send sensor data to the server, along with a bunch of other requirements for the device. A lesson learned from this was how to develop firmware for an IoT device which can be applied to future project/career path if we have to work with IoT devices.

5.0 Challenges

When developing this project we encountered a couple of challenges, in this section we will go over what the challenges were and how we overcame them. Below will be the challenges we encountered as well as the solutions we implemented.

1. Graphing the sensor data from the particle device using chart.js.
   - To help ourselves graph the data from the particle device we overcame this challenge by referring to the online guides on how to use chart.js. This helped us base our chart off of and then configuring it to our desired application.
2. Getting the sensor data from the particle device and storing it into our database.
   - To help ourselves get the sensor data from the webhook and then transfer it into our database, we did a lot of troubleshooting. This included going through and checking if the code had any errors, and following the mongodb guides that can be found from class lectures and zybooks.
3. Setting up the webhook on our AWS server.
   - To help ourselves set the webhook up on the AWS we referred to in-class activities, as well as search online any errors we encountered. To test to make sure the webhook was working correctly we would just print the data coming to the server to the console.
4. Getting all the routes to work how we wish them to work.
   - To get all the routes working we referred to the 1024 Activities posted on the d2l page. This was a great starting point for most of the routes that we used, when adding/creating new routes we could base the new routes off of the ones found in the activities.
5. Getting information out of the database using jquery, especially the sensor data.
   - To help get the information out of the database it was a lot of troubleshooting once again. The hardest part was figuring out how to use JSON files and correctly stringify and parse them such that we can access the arrays of data.
6. Setting up the AWS server such that our files would run on the service.
   - The help solve this challenge it was difficult to overcome as we had issues getting the AWS server to correctly install and use mongoDB, for whatever reason it worked on some of our group member's AWS server but not for everyone so after

each of us gave it a try we were able to successfully upload and utilize our web application on the AWS server.

6.0 Contribution Table

Team Member	Frontend (%)	Backend (%)	Embedded Device (%)
Chris Westerhoff	50	50	-
Tim Le	50	50	-
Ashton Rowe	-	-	100

In the table above it shows the breakdown of each of the team members' work. As can be demonstrated in the table Tim and Chris worked on getting the functionality of the AWS server up and running. The work on the website development was split between the two of them, both of them contributing to the project. Ashton took up the role of working on the particle device implementation, which includes all embedded firmware for the device as well as setting up the webhook for the AWS server. Overall, the three team members on this project contributed to the project with no one excessively lacking behind expectations.

7.0 References

- *Note: All references here will be the same in our references section on the web server.*

[1] Minsik Hong/Soheil Salehi, ECE 413: 1024 Activities

[2] Soheil Salehi, ECE 413: Particle Device Sample Code

[3] bcryptjs https://www.npmjs.com/package/bcryptjs

[4] express https://www.npmjs.com/package/express

[5] mongodb https://www.mongodb.com/home

[6] jquery https://jquery.com/

[7] ajax https://api.jquery.com/jquery.ajax/

[8] bootstrapcdn https://www.bootstrapcdn.com/

[9] googleapis https://fonts.googleapis.com/icon?family=Material+Icons

[10] jwt-simple https://www.npmjs.com/package/jwt-simple

[11] chart.js https://www.chartjs.org/

[12] Chart.js code: https://dyclassroom.com/chartjs/how-to-create-a-line-graph-using-chartjs

8.0 Video Submissions

Pitch Video:

- https://www.youtube.com/watch?v=V3TOyucUdJ0

Video Demo:

- https://www.youtube.com/watch?v=PCgK6rp18DI

Link To AWS Server:

- http://ec2-3-86-45-18.compute-1.amazonaws.com:3000/