routes/devices.js

/devices/summary

This endpoint receives a deviceID from the user. It then calculates how long is 7 days before the current date. It gets all of the device data that's been taken for the device and finds the data that has been taken within the past 7 days. If there has been no data for the past 7 days, then it returns a 400 status and "No data for past 7 days." If there is data, it iterates through the data points and adds up the duration and UV exposure. It then returns a status 200 and the data points, the total duration, and the total UV exposure.

/devices/sunRun

This endpoint is called by the webhook. The firmware sends the deviceID, the API key, longitude, latitude, speed, UV, time, and status. If any one of these or all of these are not sent, then it returns a 400 status and indicates what parameter is missing from the request. If it's the status is "start", indicating the start of an activity, deviceData is made. It will initialize the required points for the model, making the startTime and endTime the current Date and time as the activity has not stopped yet. If the user is doing an activity, the status will be "activity." This will append the speed and UV data to the deviceData speed and UV arrays. The end time will also be updated with the current time. If the user doesn't do anything, the status will be "paused" and do nothing but return a status of 200 and message "Paused." If the activity is stopped, the status is "stop." The longitude and latitude will be updated to the final location, the endTime will be updated to the current date and time, the duration will be calculated using the startTime and endTime, and the final speed and UV will be pushed to their respective arrays. The activityType will be determined by the average of the speeds. Finally, a call to openweather.com's third party API, using a hardcoded API key, will get the temperature and humidity based on the latitude and longitude. If the UV index in any of these states is beyond the threshold, a status of 201 and a message "Alert" will be sent back. If the deviceData is saved properly and there is no alert for the UV threshold, a status of 201 and a message of "No Alert" will be sent back. If an error occurs, a 400 error will be returned with an error message explaining the situation.

/devices/weather

This endpoint gets the 5 day weather forecast from a third party API, openweather's forecast API, using a hardcoded API key and the last latitude and longitude recorded in the database. If there are no errors, the endpoint will return a 200 status and JSON of the "list" of weather information from the requested API. If there is an error, it will return a 400 status and an error message.

/devices/myDevices

This endpoint receives a token from the user. It decodes the token to receive the user's email. It looks for devices that have an email matching the user email registered for a device. If the email matches the user email registered for a device, it returns all of the device(s) information and a 200 status. If there is an error, it will return a 400 status and an error message.

/devices/setThreshold

This endpoint receives a threshold and a deviceID from the user. If a device with the deviceID does not exist, it returns a 400 status and "Device does not exist." If the device with the deviceID does exist, it updates the UV threshold for that device and returns a 201 status and "Threshold updated." Otherwise, it returns a 400 status and "Could not update threshold."

/devices/activityDetail

This endpoint returns the details of an activity for the user. It will query the database for the activity. It will return a status 200 and the data for that activity.

/devices/status/:devid

This endpoint takes in a query, devid, and returns the deviceId and the last time of contact. If the query is all, it will return all deviceIDs and the last contact, with a response status of 200. This was primarily used for testing purposes.

/devices/changeActivity

This endpoint takes in an activity type and updates the deviceData with the activity type requested. If the user does not send an activity type, it returns a 400 and an error message. If a type exists, then it updates the deviceData with the new activity type. If the save doesn't cause an error, it returns a 201 status and "Activity type changed."

/devices/sensorData

This endpoint takes in a deviceID from the user. Searching based on the deviceID for the device data, it will return all the data for that device. If there is an error, it will return status 400 and an error message. Otherwise, it will return 200 and all of the device data for that device with the respective deviceID.

/devices/register

This endpoint takes in a token and deviceID from the user. If the token or the deviceID are invalid, it returns a status 400 and an error message. Otherwise, it finds the user with the email, decrypted from the token. It will create a new device registered with the deviceID and the user's email. It will make a new API key and use the threshold that the user currently has. Once it saves the device, it will add this device to the user's list of devices. It will return a 201 status with message "Device ID <deviceID> was registered."

/devices/ping

This endpoint takes in a token and a deviceID from the user. If the token or the deviceID are invalid, it returns a 400 status and an error message. Otherwise, it will ping the device and return a status 200 and "Device ID <deviceID> was pinged." This was primarily used for testing purposes.

/devices/deleteDevice

This endpoint takes in a token and a deviceID from the user. If the token or the deviceID are invalid, it returns a 400 status and an error message. Otherwise, it will find a device with the deviceID and check that its email matches the one decoded from the token. If it doesn't match it returns a 401 status and an error message. If it matches, it removes the device from the database and removes the device from the user's array of devices. If any errors occur in this process, it will return a 400 status and an error message. Otherwise, it will return a 200 status and a message saying "Successfully deleted device."

routes/users.js

/users/signin

This endpoint receives an email and a password from a user inputted on the login html page. It then searched the User database to see if the email is registered to a user and if it is, the endpoint then uses bcrypt to compare the hashed password in the User database collection to the plaintext password from the user. If the passwords match, the endpoint will generate a token and return it to the user with a 200 status code.

/users/register

This endpoint receives an email, password, and name from the user which is inputted on the register front end page. The endpoint then hashes the password and creates the newUser object, which is then stored in the User database collection and returns a 201 and a redirect to the login page.

/users/account

This endpoint gets the account information for the user, given a token from the user. If the token is not valid, a 401 status is returned and an error message is given. Otherwise, it decodes the token and finds the user with the matching email address. If one is not found, a 400 status is returned with an error message. If one is found, then it returns the users email, full name, devices, and last access. If there are no errors, then it returns a 200 status and the information.

/users/setThreshold

This endpoint receives a threshold, an email, and a deviceID from the user. If a user with the email does not exist, it returns a 400 status and “User does not exist.” If the user with the deviceID does exist, it updates the UV threshold for their devices and the user and then returns a 201 status and “Threshold updated.” Otherwise, it returns a 400 status and “Could not update threshold.”