smart_meter

The remote server is broken up into 2 different parts:
Express.js Backend Server
- database_setup.py: Creates table for meters and users.
- index.js: This file was given from past group and I did not have to touch it too much for my purpose. This mostly goes through the different functions the express.js is responsible for (ex: adding users to the database). Print statements are put through to ensure accurate testing.
- meter.py: Meter class. This is legacy code that works in the remote server. When a meter is created, a corresponding database is made that stores history about that meter. The goal is to have the general meter.db have a value that corresponds to each meters own database for identification.
- receiver.py: This does not work fully - the goal is to have this file be used to get the data from ngrok.
- sim_meter.py: Legacy code

React.js Frontend
- serviceWorker.js: Legacy code
- App.js
- /components/: These are the most important files. Some are just for details like the Header.js, NavBar.js, some are the most important files. ex. The database pulls data and it's displayed through the MeterTable.js file -> gets displayed on the website.

RPI Files.
- Only 2 files here are important: flask_server.py and sender.py
- Eventually the local_database.py and power_store.db will be replaced with the code from the Local Operations System. This did not happen in Spring 2022 since our demo was done online. This is ONLY kept for legacy. Future implementation will only (I believe) need the flask_server.py/main.py/sender.py files.
- sender.py: addr needs to be changed each time ngrok is rerun. This sends the data to the remote server.
- flask_server.py: needed to create a HTML Page to pull the data from the website and display it.
- Full functionality of the data being pulled from the database and displayed through the flask server was not implemented. Local Operations System Spring 2022 displayed the data separetely.

Remote Web Server Installation/Setup:
- 1. Clone repo (https://github.com/jennythakkar/smart_meter)
  - Or downloading existing files from Project_Data folders.
- 2. cd into the expressjs-backend directory
  - Delete node_modules folder and package-lock.json file
  - Run "npm install" to get project dependencies
- 3. Run node index.js - This begins running the Web Server backend
  - " Listening on port 3001" should appear in terminal
- 4. In a separate terminal window in the root smart_meter directory, run ngrok
- cmd: 'ngrok http https://localhost:3000'

- 5. In another separate terminal window, cd into the reactjs-frontend directory
  - run npm install for front-end dependencies
- 6. Run npm start - launches front end interface
  - Check first terminal window for console log statements to verify connection
  - In ngrok, you may see GET requests for the favicon icon (react.js icon). This also verifies ngrok working properly.

Additional Setup - Google API Key
- Get an API Key from a Google account and make sure MAps is turned on for the API key settings
- In testapp -> src -> components -> GoogleMapContainer.js -> line 117. There should be a comment that said /* INSERT ACTUAL API KEY */ - insert here in the double quotes to active Google Maps heatmap in interface.
- Spring2022: This did not work for me. Don't know why.

Interface Guide: - Legacy Description.
- To create an account, go to Create Account tab and enter username, password
  - 0 for consumer, 1 for utility, 2 for admin (not functioning)
Utility interface:
- Use information on the right side of the top of the page to operate Google Maps heatmap. Use Start Playback and Stop Playback buttons to control heatmap animation.
- Below Google Maps heatmap, The four buttons can be used to Add a Simulated Meter, Add a Physical Meter (smart meter is hard coded to be represented by Meter ID 6), Update the Data, or Delete all the meters in the system.
- In the chart below the buttons in (b), clicking the buttons in the Meter ID column change which meter is displayed in the measurement display chart at the very bottom of the Utility Interface. The buttons in the State column change Meter between running and stopped for data collection.
- Under the “Meter Assignments” title, the left-hand Update Meter Assignment tab allows for assigning a meter to a specific user in the system as well as giving the meter a geographical location for the heatmap.
- At the very bottom of the page, meter information with each timestamp, real power, reactive power, … is displayed in pages of 25 values. Step © mentions how to change which meter’s information is displayed.

Consumer Interface:
- The interface simply displays power and energy consumption for the customer to see from their own meter
- Data displayed on pages of 25 entries

Raspberry Pi Flask Server Setup
- In main.py and flask_server.py, change the IP address and port value to the values shown on the wifi menu (wlan0)

- Run sudo python3 with flask_server.py file

- Click on IP address URL to display data

External Notes:
- Not all the files uploaded to the Project_Data folder correctly. My assumption is its just some of the node_modules. It gets deleted and reinstalled anyway so it's fine.