CSCI 370 Final Report

Tori Geis

Ben Kamins

Blake Ripp

Eric Hayes

Jack Kohlsaat

Revised June 15th, 2022

CSCI 370 Summer 2022

Dr. Paone

Table 1: Revision history

Revision	Date	Comments
Rev - 1	5/18/22	Started document Completed first iteration of requirements section
Rev - 2	5/19/22	Strengthened the requirements section by clarifying the definition of done Separated *Stretch Goals* from *Functional Requirements*
Rev - 3	5/23/22	Continued revisions of *Requirements* Created and revised *Design*
Rev - 4	5/26/22	Modified definition of filtering Narrowed *Definition of Done* Added figure labels Made two additional figures (one for AWS server, one for interface)
Rev - 5	5/31/22	Added changes that were described in feedback Added *Ethics* and *QA*
Rev - 6	6/2/22	Updated *Definition of Done*, *Technical Issues*, *Non-functional Requirements*
Rev - 7	6/6/22	Updated test results Added *Completion Status*, *Future Work*, *Team Profile*
Rev - 8	6/13/22	Edited and revising entire document Incorporated feedback from previous section
Rev - 9	6/14/22	Edited early writing about code to reflect finished project
Rev - 10	6/15/22	Incorporated peer review feedback

Table of Contents

I. Introduction

Helio Home is a small startup that specializes in home energy efficiency. Their goal is to convert as many homes to an electricity-based system in order to utilize cleaner energy. They do this by subcontracting different services to install electric car amenities, solar panels, and electrical replacements for certain utilities. Helio Home's software is designed to take data on a specific home's assessment in order to run a National Renewable Energy Laboratory (NREL) energy load simulation. This simulation helps to determine if a home is able to be electrified and determine the extent that a home can be modified.

The goal of this project is to add to an existing web application owned by Helio Home that handles bulk requests of residential addresses to run through the NREL simulation. This involves creating a frontend web interface as well as the development of backend endpoints and the database to store all of the address and structural data. The proposed extension to the web app takes a zip code as an input and then identifies all of the residential addresses in that zip code by populating a table. It displays a map of the zip code along with a border around the zip code area. Potential for energy optimization is determined by using the NREL simulation software.

The basis of the existing software is an already-implemented web application for Helio's employees that handles a single address simulation. Currently, the website gives a home page, list of users, results of optimization, and a link to Estated: an Application Programming Interface (API) key system. Since the Helio Home team wants to market to specific homes that have a high potential for electrification, a bulk tool is necessary to run their defaults and simulation on multiple addresses at once. This bulk tool allows the team to identify more homes and grow at a much quicker pace.

Helio Home utilizes data provided by Estated and a simulation software provided by NREL. The databases and simulation are hosted on an AWS server that is accessed via the backend software. The primary users for the bulk tool are the Helio Home team, who is also in charge of maintaining the software and uploading new data to the database.

II. Functional Requirements

This program is a tool that allows for the gathering of data from multiple databases and programs into one convenient location, then displays the information visually on a map and in a table. The specific functionalities included are:

1. Add a web application onto the client's existing web application that is interactable and houses all other functionality.
2. Develop a database to house all of the address and structure data supplied through Estated CSV files.
3. Query our database for residential addresses given a zip code input.
4. Show map on webpage and zoom to zip code.
5. List the addresses for the inputted zip code in a sortable table along with year built and conditioned square footage.

6. Develop an endpoint to take in an address and format a Home Performance Extensible Markup Language (HPXML) file for simulation run.
7. Run an NREL simulation on selected addresses, utilizing the data from Estated.
8. Organize all data into a separate table with address, energy load in British Thermal Units (BTU), and the current heating fuel type.

III. Non-Functional Requirements

Since there is an implementation of a user interface for our project, user input and user error need to be considered. There also needs to be consideration for how the user will interact with the webpage and various functions on that webpage. The non-functional requirements that need to be included are:

1. This software is likely to be expanded nationally. Though we are given a set of default values, they vary from zip code to zip code as they are dependent on the nearest weather station. Therefore, we must not hardcode the default values so that they can be adjusted from place to place. This process will be referred to as defaulting.
2. This software needs to be user-friendly and intuitive.
3. The design of the software and databases must allow for continuous expansion and additions without changing the current structure.

IV. Risks

For this project, we are accessing many different systems that the company already has in place and are learning how to use them. Along with this usage of new systems comes certain risks of harming their code or infrastructure. The specific risks for our project include:

1. We are adding to the company's existing code.
    a. *Risk:* Breaking existing code
    b. *Likelihood:* Unlikely
    c. *Mitigation:* Create new branches
2. We are querying APIs that are not free to access.
    a. *Risk:* Overcharging Helio Home for our queries
    b. *Likelihood:* Likely
    c. *Mitigation:* Use dummy queries or free "sandbox" queries while testing to avoid excessive requests
3. This project is done in JavaScript, which our group holds little experience in.

a. *Risk:* Not having the technical skills to complete our project

b. *Likelihood:* Unlikely

c. *Mitigation:* Learn JavaScript syntax ahead of time, work in pseudocode if necessary, then translate to JavaScript

4. We are accessing the backend database that they already have.

    a. *Risk:* Changing their data or the structure of their schema

    b. *Likelihood:* Likely

    c. *Mitigation:* Create our own database, make sure the code is unable to access their database until final submission

V. Definition of Done

The minimal requirements for our project in order to satisfy the client's needs include a list of different features in the frontend and the backend. The following requirements must be completed in order for the project to be considered done.

1. Create a web page that exists within the existing Helio Home domain.
2. Zip code request box: An entry box allows users to enter a zip code. The webapp then queries the database to retrieve a set of addresses. The entry box executes a query and the zip code remains in the entry box.
3. Address table 1: Display addresses in table format. Allow the user to sort the table using the year built. The table includes the address, year built, and conditioned square footage.
4. Map: Pan and zoom map to show the zip code entered into the zip code entry box.
5. Query Estated data: Using the list of addresses, query the Estated data hosted in our database to retrieve features about each address. Update address table with features.
6. Address table 2: Display addresses selected in a new table after pressing the 'Run Simulation' button. The user sees the addresses they selected as well as the BTU load for the address.
7. Run NREL Simulation: Given an input of a chosen address from the table, run the NREL simulation and populate the energy load column of address table 2.

With all of these updates in the company code, there is a list of different tests that need to be done before we deploy the software:

1. Appropriate unit testing frameworks are selected for each feature. Our client may run tests using the selected framework's API.
2. For each story, a minimum of two functional tests are written and passed.
3. For any Graphical User Interface (GUI) story, ad hoc testing is done as well as visual inspection.

Delivery of the project is done through Github after testing and proofreading. The code is managed on different branches throughout the team and is a branch of the company's git codebase. After the client confirms that the code is functional and well tested, it is merged with the main git branch.

For each story on Trello, the team updates the current progress of the story. There is a 'Doing', 'Ready for Testing,' and 'Done' tab on the Trello board. Before updating the status of the story, we must review the following guidelines:

1. The code meets the requirements set in the weekly sprint and passes all tests that have been created in the process.

2. Before merging to the main development branch, the code must be tested successfully and reviewed by the team.

3. The tests include functional requirements tests that test the capability of any new feature added to the web application.

The company has highlighted a long list of requirements, so in order to manage them, we have divided the requirements into functional requirements, non-functional requirements, and stretch goals. The stretch goals include goals that would be helpful to the company, but we may not get to them given the timeframe.

1. The map color codes individual houses within the zip code based on potential for energy optimization.

2. The map displays the boundaries of the zip code.

3. Clicking on a house within the zip code displays relevant information used to calculate energy load.

4. The code has an automatic data upload script.

VI. System Architecture

The system architecture for our project includes the development of a frontend web app, backend API, and database that are integrated together to run a series of NREL simulations. The workflow of the overall project as initially provided by the client is shown below in Figure 1. First, the Helio Home team makes a bulk query of the Estated API, which is then uploaded to the address database. This database is queried based on a zip code entered by the user, a table containing data about all addresses in this zip code, and a map of the zip code are displayed in a web application. Helio Home can then select specific addresses in the table to run their proprietary simulation software to see the best candidates for Helio Home's services.

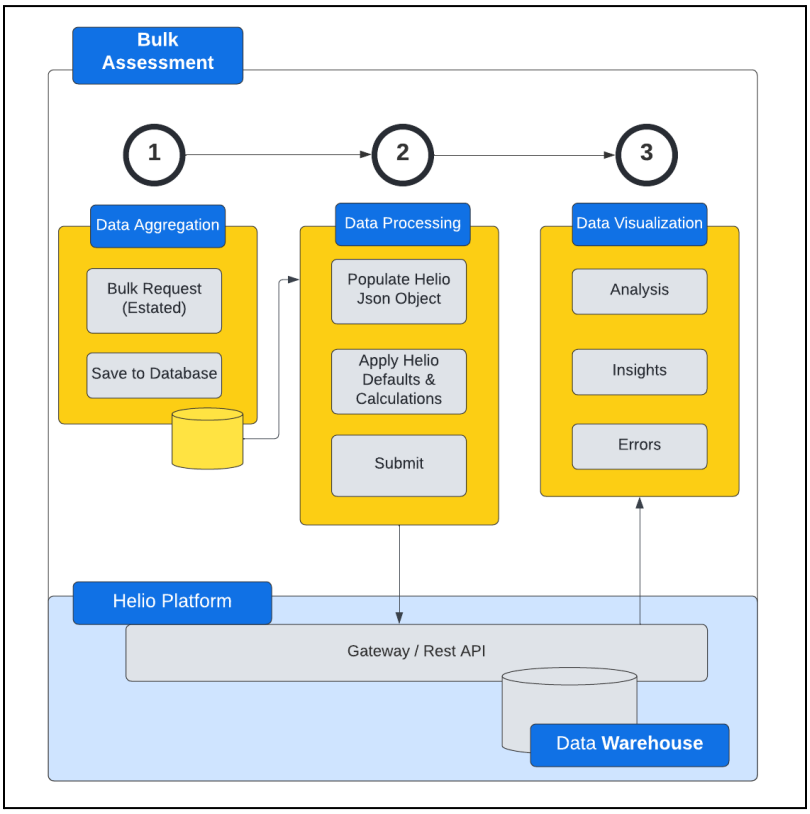

Figure 1: Initial Development Flow as Provided by Client

Over the course of the project, different functionality and architecture has been built in order to meet the client's requirements. This architecture is reflected in Figure 2, where a more detailed outline of the system workflow is shown along with the system architecture implemented. The frontend's (Figure 3) abstraction is in the top left corner. The frontend comprises all that is visible to the user and takes in the user input. This interacts with two backend elements: an API and a database. The database, which is shown on the left, returns every address in a zip code and has an upload script to make future data upload easier. This database has been made by the team and the results of this query are displayed on a table in the frontend. The API, which is boxed in the center, has endpoints added which are shown in Figure 2. This is the element responsible for allowing the user to call the database and run the NREL simulation Helio Home has designed. If a simulation has already been run on that address, the API pulls the data from a database and returns the energy load. If the simulation has not been previously run, the API fills in default values, creates a simulation input HPXML file, and runs the simulation.

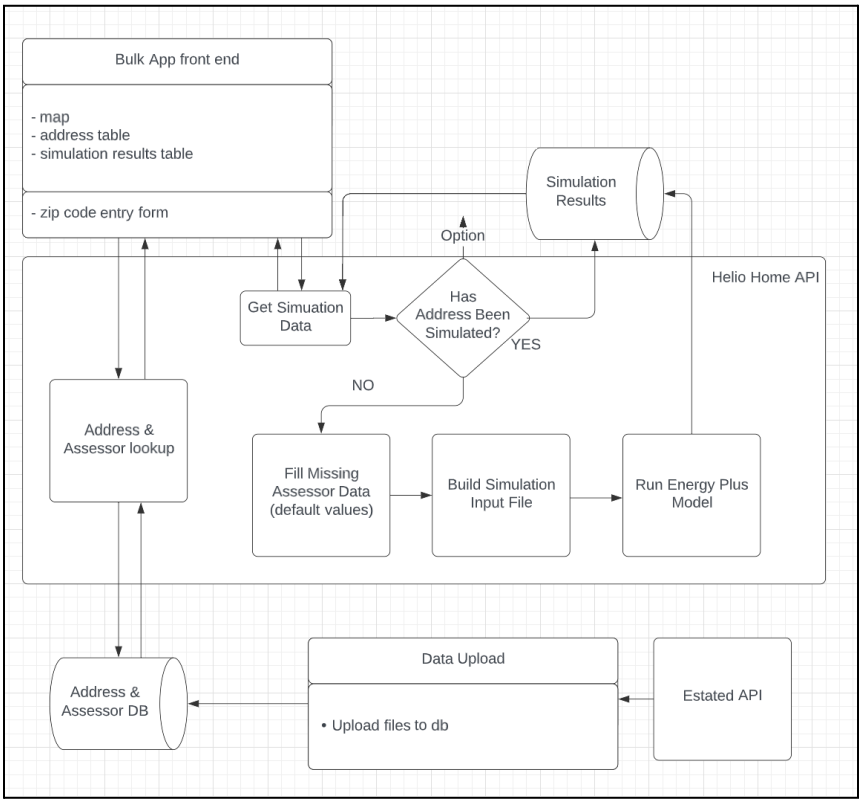

Figure 2: High-level System Architecture

The page shown in Figure 3 is what our team has developed, all other pages on the sidebar were there before our project started and have different functionality to what we are doing. The first element the user interacts with is the zip code search bar at the top. This zip code is then displayed in the map element, and the relevant addresses are displayed in the table below. The columns in the table are the address, the year the home was built, and its area that is air conditioned in square feet. These rows can be selected and then a 'Run Simulation' button runs Helio Home's built in NREL-based API that calculates the energy load of each house. This is displayed in the second table.

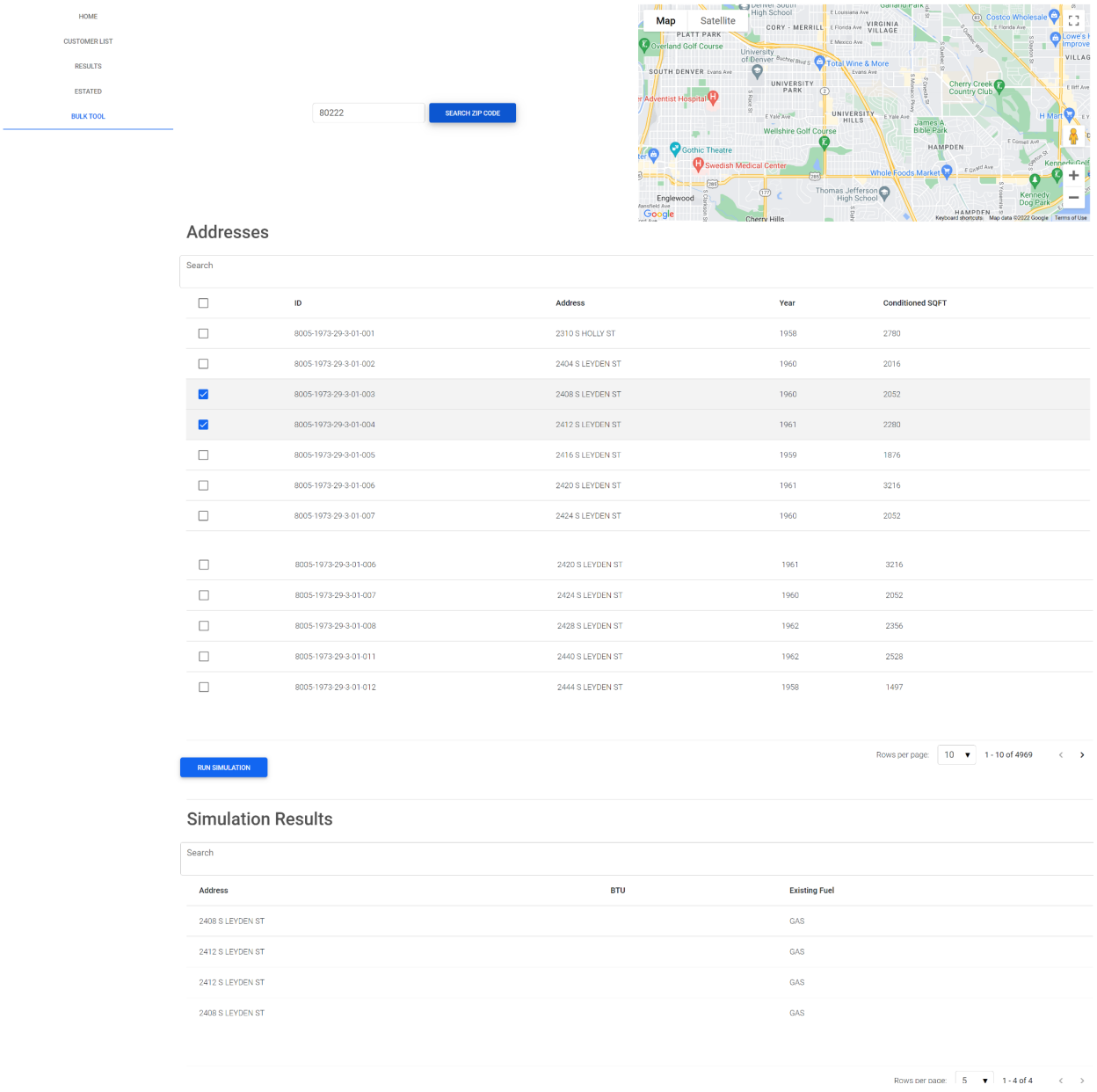

Figure 3: Client's Existing Application with New Page Incorporated

VII. Technical Design

As described above, the program takes information from the Estated API and then runs an NREL simulation with it. This is particularly difficult because an Estated output does not match an NREL input in any way, shape, or form. If the NREL input is not in an extremely specific HPXML format, it does not run. On top of this, an NREL input has more than one-hundred data points that are not present in an Estated output. The team calls the process of reformatting an Estated output into an NREL input and accounting for missing data defaulting.

Two objects were created in the form of an NREL input, converted to JavaScript Object Notation (JSON) format so they are workable. The first object contains a default value mapped to every datapoint.

These values were provided by Helio Home in the form of an Excel spreadsheet. They were then manually converted to a static python file called default.py in the proper format. This “Default Object” exists so that an NREL simulation can run even if Estated has no data on a home. The second object is also in the format of an NREL input, although there is no data. Rather, each datapoint is mapped to a function that will retrieve the value from Estated; if this information cannot be found in Estated then the datapoint is mapped to null. This is called the Source Mapping Object. To begin the defaulting process, a copy of the Default Object is made. This will become the NREL simulation input.

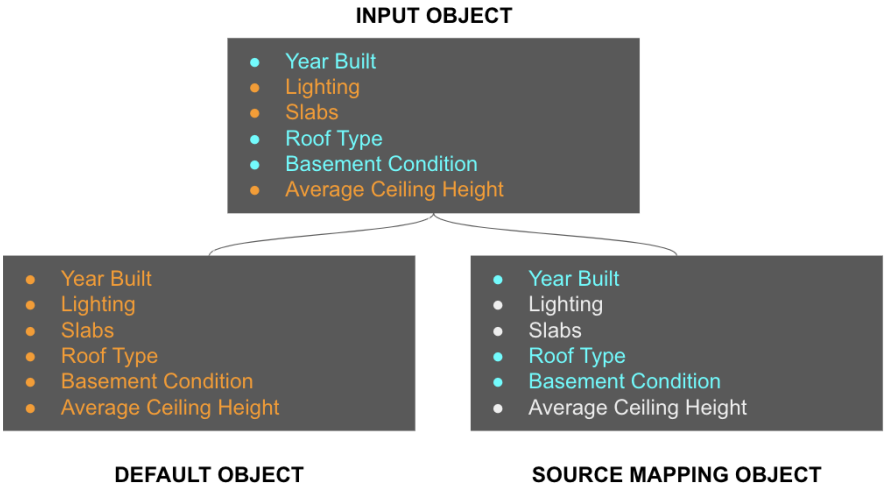

Figure 4: The Process by which Default Data is Combined with Estated Data to Create an NREL Simulation Input

The Default Object and Source Mapping Object are iterated through together. If the value mapped to a datapoint in the Source Mapping Object is null (represented by Figure 4 in light gray), then the value from the Default Object (represented in by Figure 4 orange) is not overridden. If the value mapped to the datapoint in the Source Mapping Object is not null (represented by Figure 4 in blue), then the function to retrieve data from the Estated database is called; this value overrides the default value in the Input Object. When this process is complete, the Input Object is converted back to HPXML and is now ready to be used as an NREL simulation input.

Another major technical component to the project is the development of a database to hold all of the Estated data that is requested. The database schema, as pictured in Figure 5 below, shows all of the entities that are necessary to run the simulation along with their attributes and relationships. The main entity is the address table which contains all of the location data for a specific address. Next are the 'Structure' and 'Other Areas' tables, which contain information about the actual structure of the house. These two tables contain most of the information necessary to run the simulation like heating, type, heating fuel, and square footage. The final parcel table contains data about the zoning classification for zip codes and individual addresses which is useful for running the simulation.

Figure 5. Database Schema for Estated Data

On top of the actual database, there is a data upload script that will take the CSV files from the Estated bulk request and upload the data to the database appropriately. There are also endpoints to access the data and process it into an object that can be put into the frontend tables. Another endpoint was developed for defaulting that queries the appropriate data for the simulation input file. These extra features should allow for easy expansion of the Helio Home service to multiple different locations and allow for easier manual simulation runs outside of the bulk tool that we created.

VIII. Software Test and Quality

In order to accomplish all of the functional and non-functional requirements in a way that allows for further improvements and additions, a software quality assurance plan is necessary. This plan defines how the different requirements are tested and how the software is defined as complete.

1. Add a webpage onto the client's existing web application that is interactable and houses all other functionality.
    a. *Purpose of Test:* Ensure that the user can access and view the data and simulation results
    b. *Description of Test:* A series of console logs of the data to ensure backend functions are called (functionality tested in another test)
    c. *Tools Utilized/Required for Test:* Interactable UI, local server, Python environment
    d. *Threshold for Acceptability:* Relate to requirement specification
    e. *Edge Cases:* Incorrect input from user, data doesn't exist
    f. *Results of Testing:* Webpage exists and all implemented functionality works
2. Develop a database to house all of the address and structure data.
    a. *Purpose of Test:* Ensure that the database contains the correct relations and data
    b. *Description of Test:* A series of queries; run and compare to the data that is known
    c. *Tools Utilized/Required for Test:* Django, PSQL
    d. *Threshold for Acceptability:* Data is properly uploaded and the database has the correct constraints
    e. *Edge Cases:* Data is formatted incorrectly
    f. *Results of Testing:* All primary keys and foreign keys are in the database; data is properly uploaded
3. Develop a data upload script for the database.
    a. *Purpose of Test:* Ensure that the user can upload the data (without opening PSQL)
    b. *Description of Test:* Run the script; check the database for the data uploaded
    c. *Tools Utilized/Required for Test:* PSQL
    d. *Threshold for Acceptability:* Data is uploaded properly
    e. *Edge Cases:* Incorrect input from user; data incorrectly formatted
    f. *Results of Testing:* Script functions; assumes perfect user
4. Query database for residential addresses given a zip code input.
    a. *Purpose of Test:* Ensure that the user can upload the data (without opening PSQL)
    b. *Description of Test:* Run the script; check the database for the data uploaded

c. *Tools Utilized/Required for Test:* PSQL

d. *Threshold for Acceptability:* Data is retrieved and added to the frontend table

e. *Edge Cases:* Incorrect input from user; data doesn't exist

f. *Results of Testing:* API works correctly

5. Show map on webpage and zoom to zip code.

    a. *Purpose of Test:* Ensure that the web application visualizes the region that is being shown in the table

    b. *Description of Test:* Search zip code on Google Maps; ensure that the zip code displayed in the map is correct

    c. *Tools Utilized/Required for Test:* NodeJS, Google Geocode

    d. *Threshold for Acceptability:* Relate to requirement specification; must be accurate in displaying center of zip code

    e. *Edge Cases:* Non-American zip codes which are not formatted the same as American ones

    f. *Results of Testing:* Map is shown and zooms to correct zip code

6. List the residential addresses, the year they were built, and their conditioned square footage in a sortable table.

    a. *Purpose of Test:* Show the data in a sortable table and implement sort functionality

    b. *Description of Test:* Visual test with multiple different test cases

    c. *Tools Utilized/Required for Test:* NodeJS, micro queries on specific values

    d. *Threshold for Acceptability:* Table displayed with data and implemented sorting and year filtering

    e. *Edge Cases:* No data is returned; incorrect data is displayed

    f. *Results of Testing:* Table displays correctly; sorting works

7. Organize all data into a separate table with address and energy load.

    a. *Purpose of Test:* Show the results of simulation in frontend

    b. *Description of Test:* Visual test with multiple different test cases

    c. *Tools Utilized/Required for Test:* NodeJS running a local server, micro queries on specific values

    d. *Threshold for Acceptability:* Data exists in table; microqueries show that the data is correct

    e. *Edge Cases:* No data is returned; incorrect data is displayed

    f. *Results of Testing:* Table correctly displays response from API

IX. Project Ethical Considerations

Ethics has always been a main consideration when it comes to software engineering projects. There are a few ethical concerns that will be highlighted in this section about our project. Given the scope of the project, we have narrowed it down into two main ethical considerations.

The first ethical consideration has to do with the data we are dealing with. The data from Estated is very specific about certain residential addresses and we now have access to it in our database. While there is no explicit personal information, the information about your home should be kept private, so using it as a sales tool could potentially present some moral conflict. After consulting the IEEE Software Ethics principles, we have determined that using the information in this manner is ethical. Our use of the data is within the bounds outlined by IEEE principle 3.13; we only use lawfully obtained and correct data.

Another ethical consideration that we have highlighted in this project is the IEEE section 1.07, which deals with disparities caused by the product we are developing. The reason why this project must consider this IEEE standard is because of how the NREL simulation runs. We must be aware that the simulation may not be totally unbiased towards older homes or newer homes which typically indicates two different economic classes. Those with older homes tend to be in a lesser economic standing than those who own new homes. This may be a problem because it will be easier to install Helio Home products in new homes which allows those homeowners to save more money. If only new homes can benefit from this product, then it will be unfair to those in a lesser economic state. This is something we must be aware of while developing this product.

X. Project Completion Status

The goal of the project is to create a web app to run NREL simulations on multiple addresses without need for an in person assessor to test the house. In order to accomplish this, we have developed a frontend web application and backend functionality to support it. All of the functional and non-functional requirements have been completed as defined in the above sections. The frontend web application is completely functional with user input for the zipcode search and simulation runs. There is also a map that zooms to the entered zip code, but was unable to meet the stretch goal of visualizing simulation runs on the map. The project has endpoints for querying the database that are used in the frontend to get the data for a zip code. There is also an endpoint for running the simulation on a specific address and an endpoint for getting the simulation result. In the simulation endpoint, there is functionality for querying the database for necessary information and then adding default values to a simulation input file if they are not in the database. All of this functionality has been manually tested utilizing local runs of both the frontend and backend.

Much of the backend functionality is not perfect and can still be improved to allow for concurrent simulation runs or a better storage system for previous simulation runs. However, developing additional functionality for the simulation on top of the bulk frontend and backend tools is too much for a five week project. Unfortunately, we were unable to meet one of the functional requirements for the project. The defaulting endpoint works some of the time, but some of the addresses have data that violates relational constraints in the simulation. Due to this requirement being unfulfilled, our project cannot be considered fully complete.

XI. Future Work

For the frontend, future work involves a few of our stretch goals as well as recommendations for improving the interface's functionality. The first part of future work would be to mark each house within the zip code on the map, color coded by their BTU numbers. Hovering or clicking on the house would show the data for that house. This would allow Helio Home to see if certain areas of the zip code are more viable for optimization, which could allow them to go door-to-door selling their product in certain areas. Another frontend project for the future would be to show graphs for the average data within a zip code. For example, after running the simulation, a graph could show the average numbers for a zip code. This would allow the company to compare zip codes with each other. There is also a possibility to have a new page appear when the simulation is run, which would allow the software to display the information in a more effective way. Although the table is effective, data visualizations would be more effective and easier to understand. Visualization through the use of color, bar graphs, and maps would make this web page much more enjoyable to use.

For the backend, future work would involve better organization and ability for expansion. Specifically, one prime update for the backend would be to improve the defaulting feature. Currently, when the software gets information from Estated about a home, it is likely that Estated does not have all of the required information to run the NREL simulation. So, Helio Home currently has a set of default values used to complete the necessary information. However, these values are based on data from a very limited area, specifically a home in Golden, Colorado. In order for Helio Home to successfully expand nationally, they must be able to implement accurate default values for areas around the country. The company could also aim to have as few values defaulted as possible.

Another aspect of future backend work would be to allow asynchronous calls of the simulation run endpoint. Right now, the simulations can only be run in serial. This drastically increases run times, making for an inefficient simulation run. More future work for efficiency would be to optimize endpoints for handling large amounts of data. As Helio Home starts to expand, they will need to handle massive amounts of data at once. If they can streamline this process to be more efficient, then they will have much more success in expanding.

Finally, one more step Helio Home can take to help their growth is to streamline the data entry process. Currently, data is uploaded manually via a Python script. If Helio Home automates this process, they will be able to run a mass amount of zip codes much easier than right now. Helio Home's growth process will be highly dependent on how efficiently they can get data, which is why the above future work is what the group recommends next.

XII. Lessons Learned

The group took away many valuable lessons from the project. The first is that we need a more specific, realistic definition of done upon starting projects. The project consisted of many different tools that the group was unfamiliar with, such as Django and React, creating a hefty learning curve at the beginning of the project. This also led the group to spend valuable time researching ideas that ended up being impossible for the project. For example, 3 of the members spent a few days researching how to get

data from the Google Places API when the data was in Estated all along. Better planning would have prevented this issue.

Another takeaway from this project is the value of version control organization and Git branching. Throughout the project, the team did a good job creating pull requests and branching from the main branch. However, during the second to last week, the client requested a dedicated frontend and backend branch. This messed up a lot of version control since someone needed both the frontend and backend running locally in order to test the system. This led to issues where branches with both sections were being worked on, but were not able to be merged into the frontend and backend branches since they contained extra files. Furthermore, a good deal of code had to be rewritten to accommodate the frontend and backend being in two different repositories. This led to a lot of issues for the final submission since many people were working on different systems.

However, there were also many good takeaways from the project. The group gained valuable experience in the field. They also were part of an important step in growth for a small business which aims to positively impact the environment. The group also got the opportunity to learn many new languages, such as JavaScript, Django, MD Bootstrap, and skills like scrums, Agile, waterfall, and more. In conclusion, this project, as well as its mistakes, have provided the Helio Home team with an abundance of valuable knowledge going forward in their computer science careers.

XIII. Team Profile

Name: John (Jack) Kohlsaat

Major: Chemical Engineering and Computer Science, 2023

Hometown: Overland Park, KS

Work Experience: UI design, pathfinding, and control systems for automated drilling rig

Future Plans: I hope to work in a career where I can utilize my full knowledge in chemical engineering and computer science. Ideally, this would mean working as a computer scientist or data analyst in a chemical engineering field.

Name: Ben Kamins

Major: Computer Science, 2024

Hometown: Los Angeles, CA

Work Experience: Teachers Assistant for Intro to Data Science

Future Plans: I hope to work in industry, combining my love of data science with my interest in cybersecurity to help protect people from malactors worldwide.

Name: Blake Ripp

Major: Computer Science, 2023

Hometown: Overland Park, KS

Work Experience: Internship at Jacobs Entertainment- worked on creating useful dashboards in Cognos BI, worked with SQL and the database to create useful data visualizations and tables.

Future Plans: I have a passion for solving problems and working in teams so I would love to pursue software engineering after graduation. I am willing to relocate and would love to find a company with a team oriented environment.

Name: Tori Geis

Major: Computer Science, 2023

Hometown: Omaha, NE

Work Experience: Ranch Hand at Scatter Joy Acres, Guest Services Representative at Nothing Bundt Cakes

Future Plans: I hope to work in software engineering or data science upon entering the field. However, I am open to possibilities and am always happy to learn a new skill.

Name: Eric Hayes

Major: Computer Science, 2022

Hometown: Bend, OR

Work Experience: Kansas Federal Reserve intern working with python backend

Future Plans: I will pursue a career in software engineering. I enjoy programming things with tangible results such as machinery, robotics, or guis. I hope to find a job that seeks to fix issues I am passionate about.

Appendix A - Key Terms

Term	Definition
API	Application Programming Interface
AWS	Amazon Web Services
BTU	British Thermal Unit
Defaulting	The process by which values needed for an NREL simulation input that are not present in an Estated output are accounted for
Estated	API that gives information about specific addresses
GUI	Graphical User Interface
HPXML	Home Performance Extensible Markup Language
JSON	JavaScript Object Notation
NREL	National Renewable Energy Laboratory
PSQL	Postgre Structured Query Language
QA	Quality Assurance

Appendix B - Figures