University of Mississippi

eGrove

Honors Theses

Honors College (Sally McDonnell Barksdale Honors College)

Spring 5-1-2021

Analyzing and Optimizing the Energy Operations on Campus

Rohan Agrawal

Follow this and additional works at: https://egrove.olemiss.edu/hon_thesis

Recommended Citation

Agrawal, Rohan, "Analyzing and Optimizing the Energy Operations on Campus" (2021). *Honors Theses*. 1840.
https://egrove.olemiss.edu/hon_thesis/1840

This Undergraduate Thesis is brought to you for free and open access by the Honors College (Sally McDonnell Barksdale Honors College) at eGrove. It has been accepted for inclusion in Honors Theses by an authorized administrator of eGrove. For more information, please contact egrove@olemiss.edu.

ANALYZING AND OPTIMIZING THE ENERGY OPERATIONS ON CAMPUS

By
Rohan Agrawal

A thesis submitted to the faculty of The University of Mississippi in partial fulfillment of the requirements of the Sally McDonnell Barksdale Honors College.

Oxford, MS
May 2021

Approved by

Advisor: Dr. Dawn Wilkins

Reader: Dr. Kristin Davidson

Reader: Dr. Timothy Holston

Copyright Rohan Agrawal 2021
ALL RIGHTS RESERVED

DEDICATION

To Mr. Tim Dolan, for inspiring me to take on this project and work on finding some answers to some important questions. You always raised questions that made me think critically. You brought about the thinker in me that I would have never been without you. Miss you!

ACKNOWLEDGEMENTS

To Dr. Dawn Wilkins, for being patient with me as I worked my way through this thesis. For always being the mentor I needed when I was lost.

To Mr. Dean Hansen, for working with Itron to collect the data for me. For entertaining my calls, emails, and all the impromptu meetings.

To Mr. Jeff Lucas, for always answering all my questions about the dashboard server with a genuine smile. This dashboard wouldn't have been possible without your help.

To Mr. Joseph Carlisle, for helping me finish this dashboard on time. For answering my random knocks on your door to ask questions about the project.

To Dr. Charlie Walter, for spending countless number of hours with me on zoom trying to make my project work. For pushing me to believe that I can finish this project. You invested an equal amount of time in this.

To Dr. Kristin Davidson and Dr. Timothy Holston, for providing your invaluable feedback on this thesis and always willing to help me with everything.

Thank you everyone else not mentioned here who supported me in this process and believed in me as I found my way through.

ABSTRACT

The Energy Dashboard is a way to track the University of Mississippi's energy operations and find ways to optimize them. Data from 200 meters on campus was used to create the dashboard and perform some research. The insights obtained from the data raised some important questions for the University management.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 CAMPUS ENERGY OPERATIONS

The University of Mississippi's energy is managed by the Department of Facilities Management. Mr. Dean Hansen is currently the director of the department managing 200 electricity meters on campus with his team. Besides energy, they also oversee the campus' waste water treatment plant, water supply, and the natural gas plant among other utilities.

Currently there are over 200 smart meters on campus distributed among approximately 170 locations. Some buildings are fitted with more than one meters, which makes the number of meters greater than the number of buildings. These meters have the ability to track the energy utilization for every 15-minute interval on any given day. These meters are run and managed by a third party company, Itron, which sends the data to the University through a database server.

1.2 ENERGY DASHBOARD

A data dashboard is an information management tool that visually tracks, analyzes and displays key performance indicators (KPI), metrics and key data points to monitor the health of a business, department or specific process [1]. They are interactive and visually appealing for the user to be able to draw essential insights.

This University of Mississippi Energy Dashboard, built and designed by me, serves the purpose of providing building administrators and facilities management an effective way of monitoring the energy consumption without having to scroll through hundreds of thousands of lines of data. Through interactive charts and easy navigational methods, this dashboard

will assist in closely analyzing how each building is doing energy-wise and find ways to improve the energy efficiency. This is also an effective tool for faculty and students for understanding the University's electricity consumption.

1.3 IDEA INSPIRATION

In the summer of 2019, while feeling concerned about the unnecessary use of lights at night in many of the campus buildings, I searched for some avenues to find our campus' energy consumption. However, there was no publicly available data on the University website. After researching into the matter more, I found a lot of institutions around the United States having a designated public website for campus operations. One of the models I found very intriguing was by Stanford University having more than 25 campus-level dashboards to aid Energy Management monitoring its services. This dashboard also predicted the campus' performance for a year in the future, as shown in Figure 1.1. Hence, came the idea of building the Ole Miss Energy Dashboard as my senior capstone project.

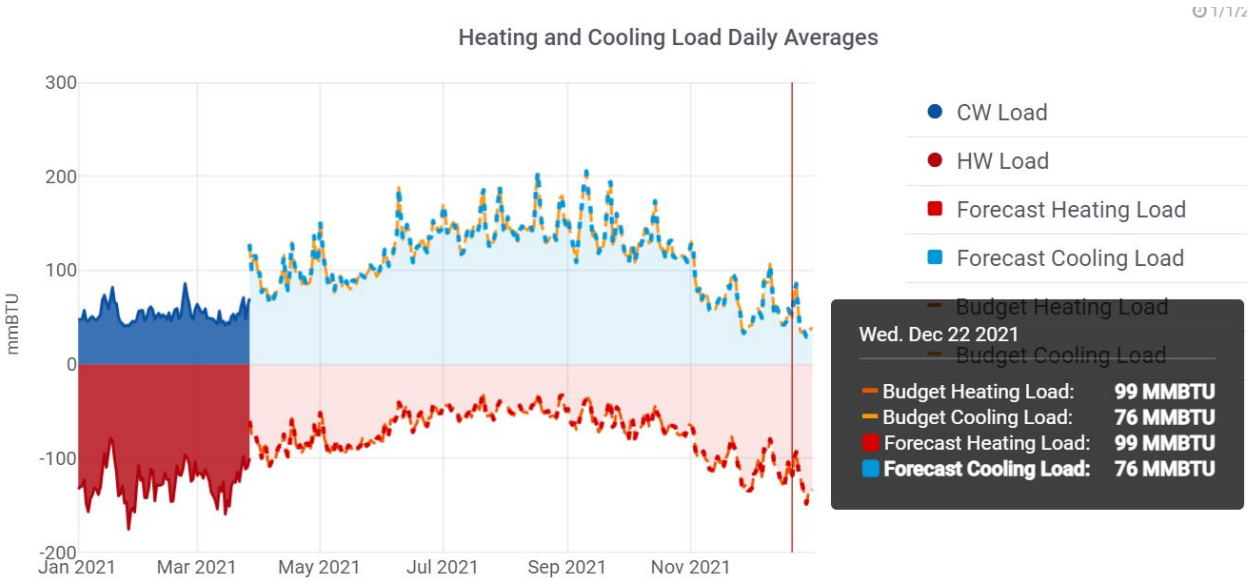

Figure 1.1: Snippet from the Stanford Energy Dashboard showing the predicted performance for Dec 22, 2021 [5]

1.4 SUSTAINABILITY TECH

This project intends to serve as a starting point for more sustainability projects on campus using technological tools. These student-managed dashboards can provide a way for the University to highlight to students how emerging technology can be used for social good. This will also encourage students to become role models in developing emerging technologies for sustainable development. By hiring computer science students to work with sustainability solutions, it enables them to think critically and become better advocates for using technology for solving societal problems.

1.5 PROJECT PITCH

With the same mindset of using technology for good, the project idea was first pitched to the Office of Sustainability and then later to the Department of Facilities Management. The idea was received with warmth by both department heads and was given adequate support to make it possible.

To allow other schools to share the same passion for combining technology with sustainability solutions, the energy dashboard idea was also presented at the Association for the Advancement of Sustainability in Higher Education (AASHE) annual conference in 2019. The idea was well received at the conference and made many sustainability officers pitch the idea to their respective universities.

CHAPTER 2

DATASET DESCRIPTION

The campus energy data was obtained in multiple formats and data tables. The final database was curated using MariaDB, a free supported fork of the MySQL Database Management System. The database is hosted on the turing server of the Computer Science department at the University of Mississippi.

2.1 DATA GATHERING

The first set of Comma Separated Values (CSV) files contained the meters and buildings data. Another CSV file contained information about the various recording channels of the meters that store the data in multiple units as per the user's needs. All these datasets were supplied by the Department of Facilities Management.

The daily meter reading dataset is uploaded on Itron's Simple File Transfer Protocol (SFTP) server at around 6 am the next day to be available for download. The data is downloaded through a Remote Desktop Connection (RDP) with the Facilities Management server and uploaded to the turing server using a SQL script.

2.2 DATABASE CREATION

2.2.1 DATA PREPROCESSING

The obtained files had quite a few missing records, discrepancies in building names and locations, and several records that were not pertinent to the data at hand. To obtain a clean database, a lot of data preprocessing was done and all the building records, containing the location names, Meter IDs, location categories were entered manually.

CategoryNumber	Category
1	Academics & Research
2	Athletics
3	Residential
4	Greek
5	Administrative
6	Cultural
7	Dining
8	Other/Third Party

Figure 2.1: BuildingTypes table showing the different location categories

- There were a few buildings with the name 'Unknown'. These buildings were not added to the database after validating with the Facilities Management office.

- Some buildings having multiple meters had different names for different parts of the building. All these names were changed to one common building name.

- Certain location names having abbreviations or misleading names were changed to more commonly known names. (Note: the original location names provided by the Facilities Management are still preserved in the Meters table)

2.2.2 CREATION OF TABLES

The data tables were created using MySQL. Overall, there are 6 tables in the database containing pertinent information about the buildings, meters, reading channels, and the meter readings. All the records (except in MeterReadings table) were inserted manually through SQL queries. The tables are described below:

- BuildingTypes: This table contains information about the category a particular location falls under. For example, Academic Research, Athletics, Residential, etc. There are 8 building types in this table as shown in Figure 2.1.

- Meters: This table contains all the meter numbers associated with all the meter locations. The names of the locations are not modified in this table and are the same

MeterID	MeterLocation
324021701	Softball Concess
324021702	Sigma Nu - Pool
324021703	Softball MX Shop
324021704	Counseling Center
324021705	Isom Hall
324021706	Phi Kappa Tau
324021707	Health & Safety
324021708	Coy Wallter GH
324021709	Northgate A
324021710	Baseball Coach
324021711	Lamar Hall
324021712	FM Shed
324021713	Library Addition
324021714	Kappa Alpha Theta
324021715	BaxterTelephone Exchange
324027942	Khayat Law Ctr
324027943	FM Shop
324027944	Stockard Hall
324027945	Bryant Hall
324027946	Sam-Gerard Hall
324027947	Lester
324027948	Stewart Hall
324027949	Howry / Falkner Hall

Figure 2.2: A snippet of the Meters table

names as provided by Facilities Management. There are 200 records in this table. A snapshot is shown in Figure 2.2.

- Buildings: This table contains all the unique location names along with their corresponding categories. There are 166 records in this table. A snapshot is shown in Figure 2.3.

- Channels: This table contains information about the Channel number and the unit of measurement each channel represents. There are 9 records in this table as shown in Figure 2.4.

- MeterBuildings: This table combines the Meters and Buildings tables to associate each meter to its respective location. The names of the locations in this table can be different than the ones in the Meters table. This was done to avoid confusion and place more commonly known names to locations for better usability. There are 200 records

BuildingID	BuildingName	Building_Type
1	Softball Stadium	Athletics
2	Counseling Center	Administrative
3	Isom Hall	Academics & Research
4	Phi Kappa Tau	Greek
5	Health & Safety	Administrative
6	Northgate Apartments	Residential
7	Baseball Coach	Athletics
8	Lamar Hall	Academics & Research
9	Facilities Management	Administrative
10	Kappa Alpha Theta	Greek
11	Baxter Telephone Exchange	Administrative
12	Khayat Law Center	Academics & Research
13	Stockard Hall	Residential
14	Bryant Hall	Academics & Research
15	Sam-Gerard Hall	Administrative
16	Lester Hall	Administrative
17	Stewart Hall	Residential
18	Howry / Falkner Hall	Administrative
19	Phi Mu	Greek
20	Delta Gamma	Greek
21	Hefley Hall	Residential
22	Lenoir Hall	Dining
23	Bondurant Hall	Academics & Research

Figure 2.3: A snippet of the Buildings table

ChannelNo	Unit	IntervalInSecs
1	KWh	900
2	KWh_REC	900
21	KVARh	900
41	KVAh	900
101	KWh	NULL
102	KWh_REC	NULL
121	KVARh	NULL
141	KVAh	NULL
150	KW_DEL_MAX	NULL
157	KVAR_DEL_MAX	NULL
400	Vh_A	300
401	V_MIN_A	300
402	V_MAX_A	300
403	Vh_B	300
404	V_MIN_B	300
405	V_MAX_B	300
406	Vh_C	300
407	V_MIN_C	300
408	V_MAX_C	300

Figure 2.4: Channels table showing the various units of measurement for the meter readings

Building	Meters
Softball Stadium	324021701
Softball Stadium	324021703
Softball Stadium	324028069
Counseling Center	324021704
Isom Hall	324021705
Phi Kappa Tau	324021706
Health & Safety	324021707
Northgate Apartments	324021709
Baseball Coach	324021710
Lamar Hall	324021711
Facilities Management	324021712
Facilities Management	324027943
Facilities Management	324028014
Facilities Management	324028017
Facilities Management	324028046

Figure 2.5: A snippet of the MeterBuildings table

in this table. A snapshot is shown in Figure 2.5.

- MeterReadings: This table contains the readings for each meter in the database for every 15-minute interval. The earliest available data in the table is from November 1, 2018. This table fetches the data from a CSV file available daily through the Itron portal. This CSV file is then inserted into the table using an SQL query. A sample reading is shown in Figure 2.6.

2.2.3 HANDLING MISSING DATA

The Meter readings obtained had some missing data for a few meters during certain times. These values are automatically skipped by the data table and the chart still renders a continuous graph. These missing values can be due to multiple reasons, such as a meter failure, server problems, and power outages, among others.

2.2.4 DATABASE SCHEMA/ARCHITECTURE

All the tables in the database are joined using a foreign key relationship. Each table also has a primary key identifier to uniquely identify the records. Refer to the database schema in Figure 2.7

Meter	Channel	ReadingTime	MeterReading
324021701	1	2020-08-21 00:15:00	0.1040
324021701	1	2020-08-21 00:30:00	0.2700
324021701	1	2020-08-21 00:45:00	0.1580
324021701	1	2020-08-21 01:00:00	0.0480
324021701	1	2020-08-21 01:15:00	0.0480
324021701	1	2020-08-21 01:30:00	0.0470
324021701	1	2020-08-21 01:45:00	0.2550
324021701	1	2020-08-21 02:00:00	0.2110
324021701	1	2020-08-21 02:15:00	0.0480
324021701	1	2020-08-21 02:30:00	0.0480
324021701	1	2020-08-21 02:45:00	0.0480
324021701	1	2020-08-21 03:00:00	0.1880
324021701	1	2020-08-21 03:15:00	0.2480
324021701	1	2020-08-21 03:30:00	0.0480
324021701	1	2020-08-21 03:45:00	0.0480
324021701	1	2020-08-21 04:00:00	0.0470
324021701	1	2020-08-21 04:15:00	0.1390
324021701	1	2020-08-21 04:30:00	0.2510
324021701	1	2020-08-21 04:45:00	0.0480
324021701	1	2020-08-21 05:00:00	0.0480
324021701	1	2020-08-21 05:15:00	0.0470
324021701	1	2020-08-21 05:30:00	0.6360

Figure 2.6: A snippet of the MeterReadings table showing readings for meter 324021701 for every 15-minute interval

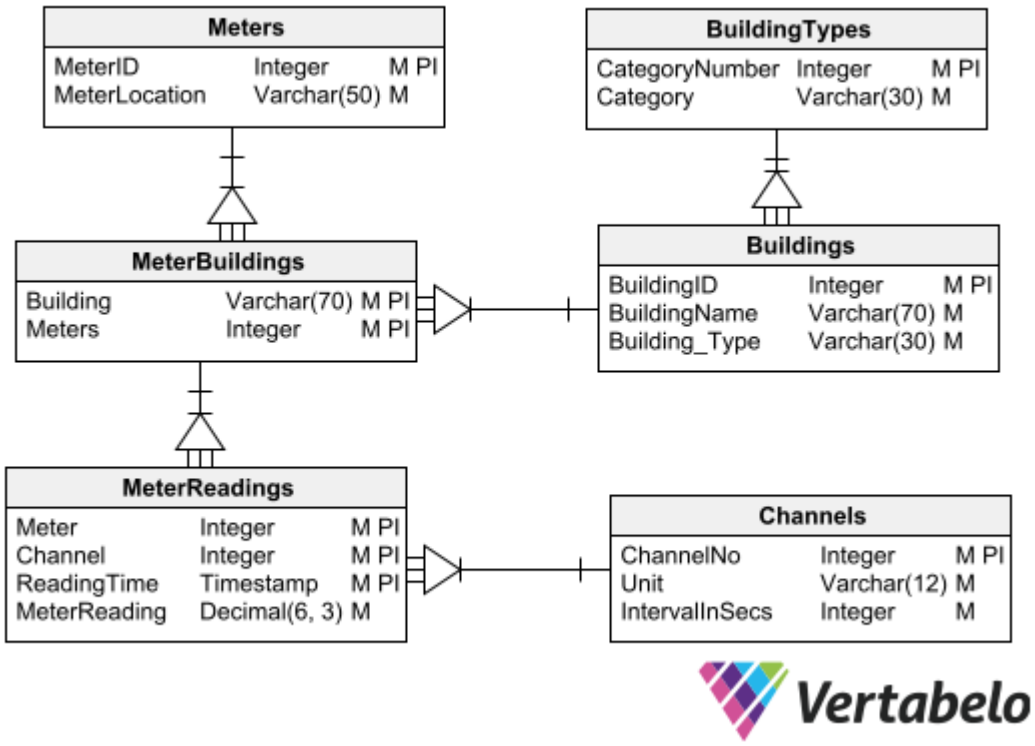

Figure 2.7: Database Schema showing the Entity-Relationship among the tables

CHAPTER 3

DESIGN AND IMPLEMENTATION

The energy dashboard is divided into two parts: the front-end and the back-end. The front-end is a Vue.js client that renders interactive web templates using JavaScript and HTML. It is easy to integrate with other parts of the project that fetch the data and display on the webpage [2]. Both the front-end and back-end are connected using Node.js and Sequelize. The process of pulling the data from the back-end and rendering it on the front end is explained below and in the subsequent flow chart in Figure 3.17.

3.1 BUILDINGS PAGE

This page renders the name of all 166 locations present in the database and allows the user to click on any of the buildings to display its energy trends. The server looks for a *main.js* file as soon as the application is started. The *main.js* file then calls a series of other JavaScript files in order to successfully display the page. The process is explained below:

3.1.1 FRONT-END PROCESS

3.1.1.1 Router Call

The *main.js* file, upon running, calls the *index.js* file stored in the router folder. The *index.js* file uses the *vue-router* package to redirect the pages to different locations based on the paths. One of the routes declared here is the buildings routes that looks for the Buildings component in the components folder as shown in Figure 3.1.

```
{
  path: '/buildings',
  name: 'BuildingList',
  component: Buildings
},
```

Figure 3.1: Sample Code showing the path that opens the Buildings page

```
methods: {
  getBuildingsByType (type) {
    return this.buildings.filter(item => item.Building_Type === type)
  }
}
```

Figure 3.2: Sample code showing the method to filter the buildings by categories

3.1.1.2 Buildings.vue Page

Once the buildings page is called from the router, it runs the Vue file for the buildings. It contains a Vue template to determine the appearance of the web page as well its contents being pulled from the JavaScript.

The script inside the Vue file calls the *BuildingService.js* file to get the Building types and categories from the database and display it on the webpage. In order to display each building in its respective category, a filter method is created as shown in Figure 3.2.

3.1.1.3 Back-end Call

The *Buildings.vue* file calls the *BuildingService.js* file. This file acts as the liasion between the front-end and the back-end through the *Api.js* file. The *Api.js* uses axios to make XMLHTTPRequests from the browser. Axios is a promise-based HTTP client for the browser and node.js [3]. This promise-based request points to the back-end on a specific URL.

The methods in the *BuildingService.js* are shown in Figure 3.3. These methods call the specific routes from the get() method that are stored in the back-end.

```
export default {
  getBuildings () {
    var temp = Api().get('buildings')
    console.log(temp)
    return temp
  },

  getCategories () {
    var cats = Api().get('buildinglist')
    console.log(cats)
    return cats
  }
}
```

Figure 3.3: Sample code showing BuildingService methods to call the back-end

```
<el-tab-pane v-for = 'type in types' :key = "type.CategoryNumber">
  <span slot='label' class='tab_label'>{{ type.Category }}</span>
  <el-row type = 'flex' class='card_flex'>
    <el-col span = '5' v-for = 'item in getBuildingsByType(type.Category)' :key = 'item.BuildingID' :id='item.BuildingName'>
      <el-card shadow = 'hover'>
          <router-link :to="{ name: 'Charts', params: { id: item.BuildingName } }"> {{ item.BuildingName }} </router-link>
      </el-card>
    </el-col>
  </el-row>
</el-tab-pane>
```

Figure 3.4: Sample code showing the HTML script used to filter the building list based on the selected building category

3.1.1.4 Building Filter

The user has the ability to click on any building category to filter the results accordingly. The HTML script in the *BuildingList.vue* template calls the getBuildingsByType() function shown in Figure 3.2. This method filters the buildings from the list of buildings based on the specific category selected by the user. Once the filtered list is obtained, the buildings are listed by the order of their names. An example of the HTML script used to execute this is shown in Figure 3.4.

```
app.get('/buildings', BuildingController.getBuildings)

app.get('/buildinglist', ListController.getCategories)
```

Figure 3.5: Sample code showing the buildings routes that redirects to the BuildingController

3.1.2 BACK-END PROCESS

3.1.2.1 Router Call

Once the front-end API calls the 'buildings' and 'buildinglist' route, it points to the *routes.js* file in the backend folder. This file points to the controllers folder for each different route specified. An example of the back-end routes is shown in Figure 3.5

3.1.2.2 Table Models and Controllers

Sequelize Object Relational Mapping (ORM) was used in order to fetch the requested information from the SQL database. Sequelize is a promise-based Node.js ORM tool for MySQL [4]. It creates a virtual object database that works well with object-oriented programming by making each table model an object with an attribute/field to hold each data item.

To create a virtual table model, an object is defined with the same name and attributes as the table stored in the MySQL database. An example of the Building table model is shown in Figure 3.6

This model stored in the *Building.js* file is referenced by the *BuildingController.js* to run queries on the model. In Sequelize, these queries are method-based instead of the traditional raw SQL queries. The controllers file sends a request to the back-end server that converts the model query to SQL query and sends back an object with the desired results. This object is then received by the *BuildingService.js* file which is then sent to the vue file to be displayed on the webpage. An example of the models controller is shown in Figure 3.7

```
const Building = sequelize.define('Building', {
  BuildingID: {
    type: DataTypes.INTEGER,
    primaryKey: true
  },
  BuildingName: {
    type: DataTypes.STRING,
    unique: true
  },
  Building_Type: {
    type: DataTypes.STRING,
    autoIncrement: false
  }
})
```

Figure 3.6: Sample code showing the definition of the Building model

```
async getBuildings (req, res) {
  try {
    const buildings = await Building.findAll({
      order: ['BuildingName']
    })
    res.send(buildings)
  } catch (err) {
    res.status(500).send(err)
  }
}
```

Figure 3.7: Sample code to get the data from the Buildings table in MySQL

Figure 3.8: A snapshot of the Buildings page displaying a few of the Greek houses on campus

3.1.3 FINAL PAGE VIEW

The final Buildings page displays all 166 locations on campus divided by their categories and listed alphabetically as shown in Figure 3.8 for the Greek buildings. The user can click on any of the categories to display the buildings in that category. Thereafter, the user can click on any of the buildings from the list to view its energy trends.

3.2 CHARTS PAGE

This page renders the energy graph of the selected building from the buildings page.

3.2.1 FRONT-END PROCESS

3.2.1.1 Router Call

When the user clicks on a particular location, vue sends a router call with the parameter as the building name as shown in Figure 3.9. This router points to the *index.js* file in the router that contains the different routes for each web page. The Charts route has an id parameter attached to it that allows it to dynamically load pages based on user selection. See Figure 3.10

```
<router-link :to="{ name: 'Charts', params: { id: item.BuildingName } }"> {{ item.BuildingName }} </router-link>
```

Figure 3.9: Sample code passing the building name as the parameter to the router

```
{
  path: '/charts:id',
  name: 'Charts',
  component: Chart,
  props: true
},
```

Figure 3.10: Sample code showing the route to the Charts page with id as the parameter

3.2.1.2 Charts.vue Page

Once the page is loaded, the created() function gets the building name from the URL parameter as shown in Figure 3.11. Once the building name is obtained from the parameter, it is then passed into the getMeterID() method to get the respective MeterID for that building. This function filters the buildings that have the same name as the one passed in the parameter and pulls out the meter ID associated with that building.

Once the meter ID is obtained, the getReadings() function inside the *MeterReadingService.js* file is called. This function fetches the readings for a particular meterID from the back-end.

3.2.1.3 Date and Time Filter

The user also has the ability to filter the readings displayed on the chart between specific dates and times. This is done through the getReadingsbyDate() function in the *ReadingbyDateService.js* that pulls specific readings between certain dates and times from

```
created () {
  this.id = this.$route.params.id
},
```

Figure 3.11: Sample code showing the building name pulled from the URL parameter

```
getReadings (meterID) {
  var readings = Api().get('readings/' + meterID)
  console.log(readings)
}
```

Figure 3.12: Sample code showing the getReadings() function passing the meterID as a parameter to the back-end

```
getReadingsbyDate (meterID, startDate, endDate) {
    var readings = Api().get('readingsDate/' + meterID + '/' + startDate + '/' + endDate)
    return readings
}
```

Figure 3.13: Sample code showing the getReadingsbyDate() function passing the filters as parameteres to the router

the back-end. The chart is updated after clicking on the Update Chart button on the dashboard.

3.2.1.4 Back-end Call

The *Charts.vue* file calls the *MeterReadingService.js* to fetch an initial reading data from the back-end. The getReadings() function, as shown in Figure 3.12, receives the meterID from the vue file and passes it as a parameter to the back-end route using Node.js API.

To receive filtered readings based on date and time from the back-end, the *getReadingsbyDate()* function is called. This function takes in the meterID, startDate, and the endDate as the parameters to be passed to the back-end router through the Node.js API. Once the output is received from the back-end, this function returns the filtered readings to the vue page, as shown in Figure 3.13

```
app.get('/readingsDate/:MeterID/:startDate/:endDate?', ReadingsbyDateController.getReadingsbyDate)

app.get('/readings/:MeterID?', ReadingsController.getReadings)
```

Figure 3.14: Sample code showing the two routes used to get the meter readings from the back-end

3.2.2 BACK-END PROCESS

3.2.2.1 Router Call

Once the front-end calls the back-end router, the router tries to fetch the readings from specific controllers based on the routes and parameters passed. These parameters define which controller will have the specific query for it to return the appropriate results. A sample code is shown in Figure 3.14.

3.2.2.2 Table Models and Controllers

The MeterReading model is created, as shown in Figure 3.15, in the "virtual database" for Sequelize. This model allows querying of the MeterReadings table in MySQL without using traditional raw queries.

The *ReadingsbyDateController.js* and *ReadingsController.js* try to query the database to fetch the meter readings based on the different parameters passed by the router. Both these controllers only get the readings for Channel 1, which displays the readings in KWh. To get the readings between specific days and times, the *getReadingsbyDate()* function in the controller fetches the date parameters from the request sent to the function and sends to the back-end query. All the obtained readings are ordered based on the date and time in ascending order, as shown in Figure 3.16.

Once the readings are obtained through the query, they are sent back to the front-end service using the back-end router. The service then returns the data object to the vue file, which displays the final information in a chart.

```
const MeterReading = sequelize.define('MeterReading', {
  Meter: {
    type: DataTypes.INTEGER,
    primaryKey: true
  },
  Channel: {
    type: DataTypes.INTEGER,
    primaryKey: true
  },
  ReadingTime: {
    type: DataTypes.DATE,
    primaryKey: true
  },
  MeterReading: {
    type: DataTypes.DOUBLE
  }
})
```

Figure 3.15: Sample code showing the MeterReadings model created using an ORM

```
async getReadingsbyDate (req, res) {
  try {
    const reads = await MeterReading.findAll({
      where: {
        Channel: 1,
        Meter: req.params.MeterID,
        ReadingTime: {
          [Op.between]: [req.params.startDate, req.params.endDate]
        }
      },
      order: ['ReadingTime']
    })
```

Figure 3.16: Sample code showing the getReadingsbyDate() method in the back-end filtering the readings by date

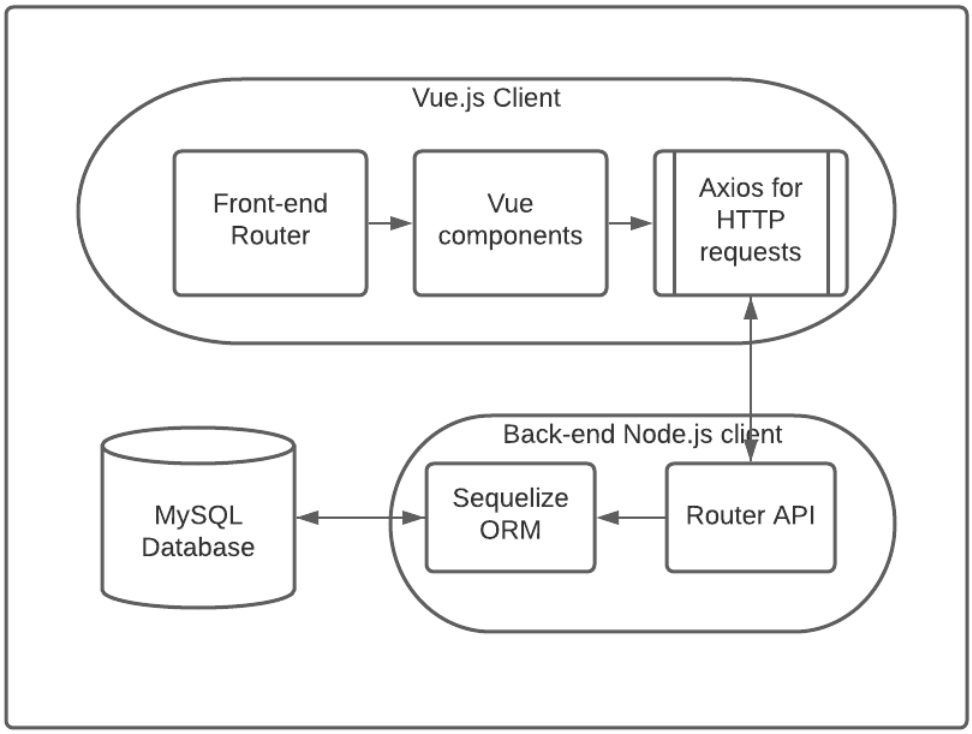

Figure 3.17: Flow chart showing the chain of command in order to render the pages

3.2.3 DISPLAYING THE CHART

To display the final line graph for the meter readings, the Chart.js library was used. Chart.js is an open-source JavaScript-based library for HTML that is used to create interactive charts for the web.

3.2.3.1 Line Graph Library

Chart.js provides a line graph package that can be altered to fit the project's needs. This line graph library was imported into *lineChart.js* file and was customized to change the appearance of the chart on the webpage. An instance of the file is imported into the *Charts.vue* file in order to display the chart with the custom settings.

3.2.3.2 Filling the Data

To display the obtained readings from the back-end onto the line chart, a fillData() method is created in the *Charts.vue* file. This function stores all the metadata about the chart in a *datacollection* variable as shown in Figure 3.18. The chart labels and data are obtained from the other variables inside the file that stores the information obtained from the back-end through the Reading services. The chart-data parameter for the *linegraph* tag takes in the values from *datacollection*, as shown in Figure 3.19, and renders the chart on the web page based on the given custom chart settings.

3.2.3.3 Updating the Chart

In order to display the updated chart based on the selected day and time by the user, an *updateChart()* method is used, as shown in Figure 3.20. This method gets the user-selected start and end date from the web page and passes it to the reading service as a parameter. Once the filtered data is obtained, the process to render the data on the chart is the same as described in the above section.

```
fillData() {
  this.datacollection = {
      labels: this.time,
      datasets: [
      {
          label: 'KWh',
          borderColor: 'lightblue',
          pointBackgroundColor: 'blue',
          borderWidth: 2,
          pointBorderColor: 'red',
          data: this.reading
      }
    ]
  }
},
```

Figure 3.18: Sample code showing the fillData() method used to capture the metadata for the chart

```
<linegraph :chart-data = 'datacollection'>
</linegraph>
```

Figure 3.19: Sample code showing the linegraph tag used to display the chart on the web page

```
async updateChart () {
  var startdate = this.startDate
  var enddate = this.endDate
  this.readings = (await DateService.getReadingsbyDate(this.meter[0], startdate, enddate)).data
  console.log(this.readings)
  this.time = []
  this.reading = []
  this.getReadings(this.readings)
  this.fillData()
}
```

Figure 3.20: Sample code showing the updateChart() method

Figure 3.21: Graph displaying the meter readings for Weir Hall from 8/22/2020, 1:00 to 8/22/2020, 19:00

3.2.4 FINAL CHART VIEW

The final Chart page displays a line graph with an initial 100 readings from the database. Once the user selects a specific date and time, the chart updates with the required data, as shown in Figure 3.21. The user can hover over any of the data points to note the particular reading for that instance.

CHAPTER 4

RESULTS

While looking at various buildings and their energy consumption, a few locations had extremely high energy consumption. Some locations were even found to be using more energy at night time than other locations at day-time. Their Energy Use Intensity (EUI) Some examples are shown below.

4.1 RESIDENTIAL BUILDINGS

The on-campus residential halls are some of the high energy consumers as these buildings remain open at all times for students. These halls are divided into 2 building types - contemporary and traditional.

The traditional-styled buildings have a more vintage dormitory look to them. Each floor in these buildings have a community bathroom with multiple stalls shared by all the residents of that floor.

The contemporary-styled buildings have a more modern look to them. Each bedroom is equipped with a personal bathroom and shower stall.

4.1.1 STOCKARD/MARTIN HALLS

Stockard and Martin Hall are traditional dormitories on campus for first-year students. Stockard Hall houses male students while Martin houses female students, both having 11 floors. Besides the regular fall and spring semesters, the hall is open for part of the summer break to host conferences and camps. However, the hall was completely closed in 2020.

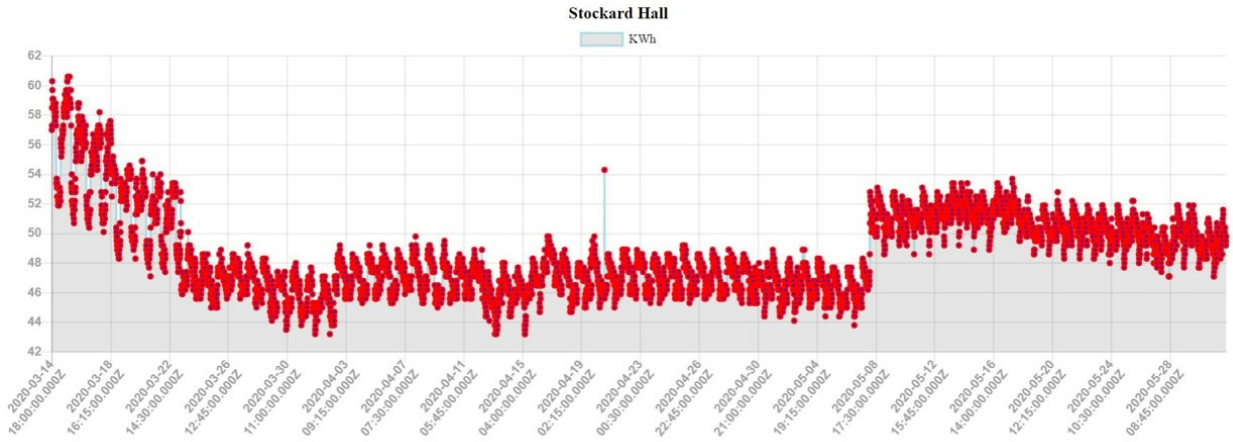

Figure 4.1: Graph showing the energy consumption in Stockard Hall from 2020/03/24 - 2020/05/31

Both buildings are identical to each other and are connected by a long hallway in the middle. They also share the same electricity meter. The insights from the meter readings are shown below:

4.1.1.1 POST-MARCH LOCK-DOWN

As of March 31 2020, the University went into a complete shutdown and students were made to pick up necessary items from their rooms with the hopes of being able to return to campus in a few weeks. However, no one was able to come back until after the end of the semester on May 11 to pick up their remaining items from their rooms. The energy data reflects the same trend as shown in Figure 4.1.

2020/03/24 - 2020/05/06:

Total consumption = 197,503.9 kWh

Avg. Daily consumption = 4,489 kWh

Daily cost = $397.64

2020/05/08 - 2020/05/31:

Total consumption = 116,447.4 kWh

Avg. Daily consumption = 4,851 kWh

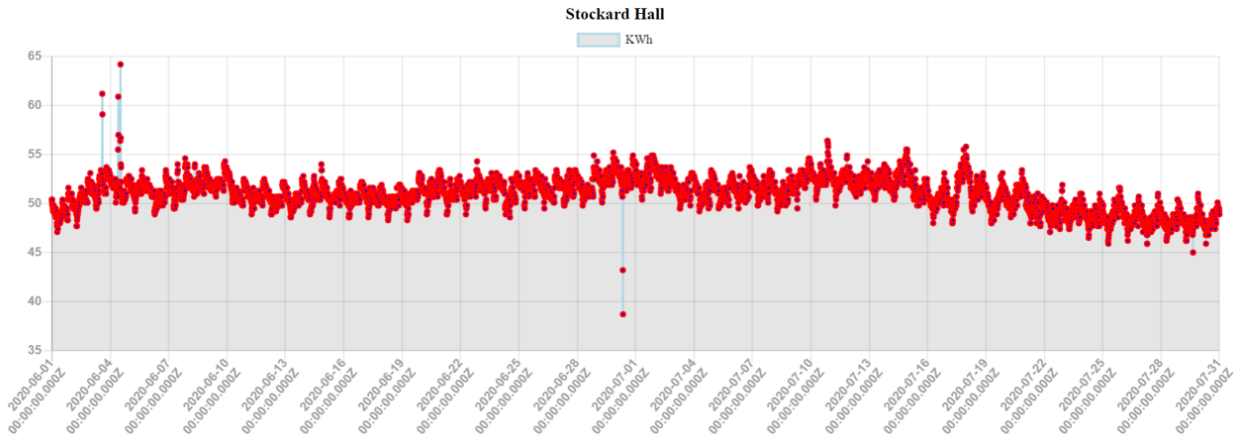

Figure 4.2: Graph showing the energy consumption in Stockard Hall from 2020/06/01 - 2020/07/31

Daily cost = $429.70

The slight increase in energy usage (5 kWh/15 min) in May reflects the influx of students coming back to collect their belongings.

4.1.1.2 SUMMER 2020

Both Stockard and Martin were completely closed for Summer 2020 and part of Spring 2020 post-Spring break due to COVID-19 restrictions, with the exception of May 2020. However, once all the students left after May, the building's energy consumption remained constant and did not go down as shown in Figure 4.2.

Total energy consumption = 298,493.7 kWh

Average Daily consumption = 4,893.3 kWh

Daily cost = $433.44

One of the major and constant energy consumers in the building are the centralized ACs. However, as compared to Minor/Pitmman Halls, which are similar-sized contemporary-styled residence halls, the traditional-style ACs tend to use more power. Minor/Pittman Hall share the same electricity meter and have a similar square footage area as Stockard/Martin Halls. Nonetheless, Stockard/Martin use more power than their contemporary counterparts.

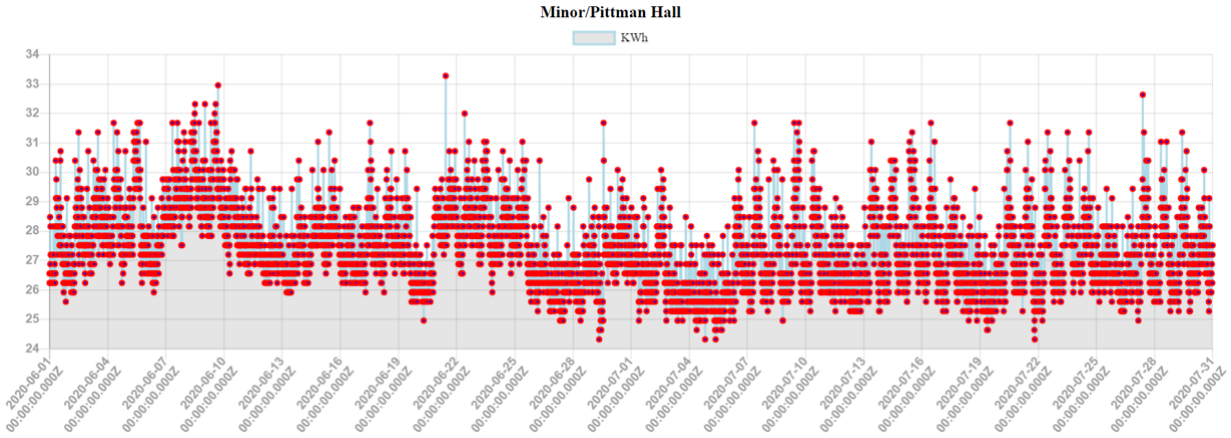

Figure 4.3: Graph showing the energy consumption in Pittman Hall from 2020/06/01 - 2020/07/31

In fact, students in contemporary halls have the ability to control their thermostats while traditional buildings do not. Refer to Figure 4.3.

Total energy consumption = 157,501.12 kWh

Average Daily consumption = 2,582 kWh

Daily cost = $228.71

The daily energy consumption in contemporary halls is almost half that of traditional halls, especially during the months when the halls were closed.

4.1.1.3 YEAR/YEAR COMPARISON

STOCKARD/MARTIN:

2019 Total Consumption = 2,019,632.242 kWh

Energy costs: $178,899.02

2020 Total Consumption: 2,022,070.5 kWh

Energy costs: $179,115

MINOR/PITTMAN:

2019 Total consumption: 1,159,963 kWh

Energy costs: $102,749.52

2020 Total consumption: 1,199,480.64 kWh

Energy costs: $106,249.94

As the data shows, the energy consumption in 2020 was more than that in 2019.

4.1.1.4 ENERGY USE INTENSITY (EUI)

To get further insight into the buildings, the building's EUI was calculated and compared to the Energy Star's EUI reference table.

Stockard/Martin Gross Sq. ft. = 231,330 sq. ft.

Stockard/Martin 2019 EUI = 29.78 kBtu/ft^2

Minor/Pittman Gross Sq. ft. = 171,470 sq. ft.

Minor/Pittman 2019 EUI = 23.08 kBtu/ft^2

Overall, the collective data suggests that, occupied or not, Stockard/Martin units are more inefficient than their contemporary counterparts. The lack of proper insulation in the traditional buildings make the ACs work overtime and consume more power.

4.1.2 CROSBY HALL

Crosby Hall is another traditional residential dormitory on campus. It houses first-year female students and has 10 floors. Similar to Stockard and Martin Hall, Crosby is also closed for the entirety of the summer and open for the regular fall and spring semesters. The building was completely closed for Summer 2020, excluding the month of May.

4.1.2.1 POST-MARCH LOCK-DOWN

During the lock-down period from late March until the beginning of May, Crosby hall had an unusual energy graph as demonstrated in Figure 4.4.

2020/3/24 – 2020/5/6:

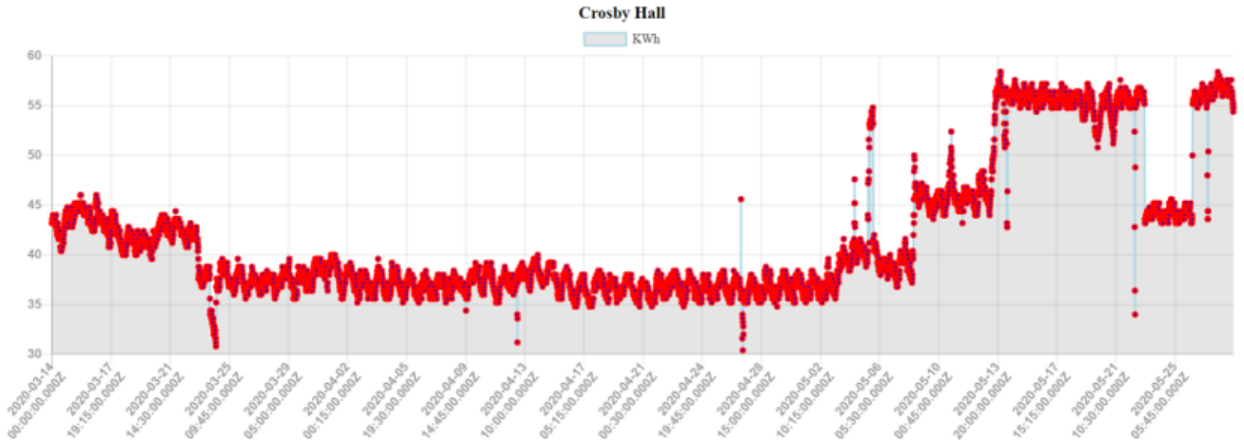

Figure 4.4: Graph showing the energy consumption in Crosby Hall from 2020/03/14 - 2020/05/31

Total consumption = 151,075.2 kWh

Avg. Daily consumption = 3,501.7 kWh

Daily cost = $310.18

2020/5/8 – 2020/5/31:

Total consumption = 116,447.4 kWh

Avg. Daily consumption = 4,678.48 KWh

Daily cost = $413.95

The data shows an increase in energy consumption in May reflecting the influx of students coming back to campus.

4.1.2.2 SUMMER 2020

Crosby Hall was completely closed for Summer 2020 and part of Spring 2020 post-Spring break due to COVID-19 restrictions, with the exception of May2020. However, once all the students left after May, the building's energy consumption remained constant into the month of June and did not go down as shown in figure 4.5.

Total Energy Consumption: 242,414.4 kWh

Avg. Daily Energy Consumption: 3,974 KWh

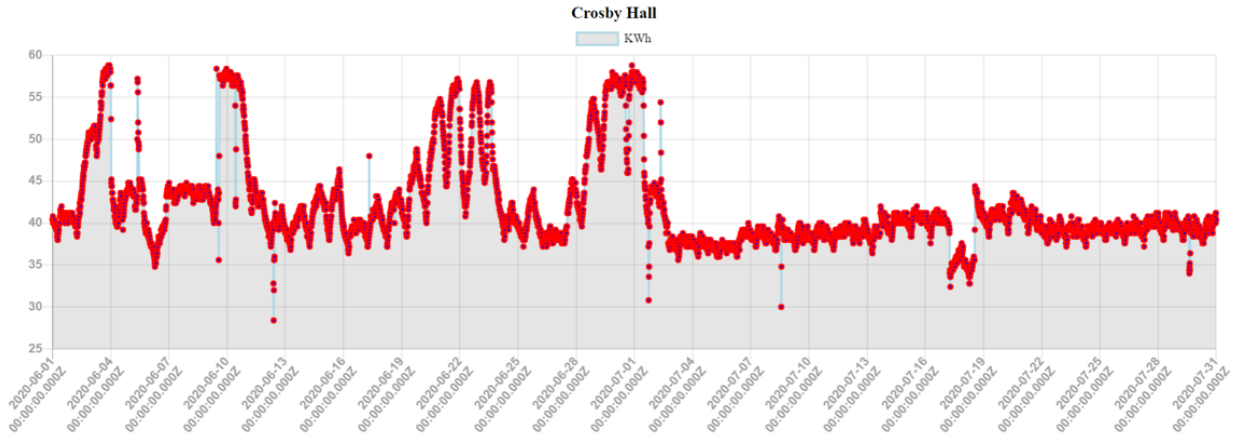

Figure 4.5: Graph showing the energy consumption in Crosby Hall from 2020/06/01 - 2020/07/31

Avg. Daily Cost: $352.02

The average daily cost went up in the summer months compared to the post-March lockdown period in the same year. Although the ACs require more energy during summers to keep the buildings at a stable temperature, it was informed by Student Housing that Crosby's AC system was broken for part of the summer. Despite that, the energy consumption is unusually high.

4.1.2.3 YEAR/YEAR COMPARISON

To better understand Crosby's energy consumption, it was compared with RH1's energy data. RH1 is a contemporary residence hall for upperclassmen and first-year students. The building has 4 floors and is equipped with 2 electricity meters. Below is the data from 2019 and 2020.

CROSBY:

2019 Total consumption: 1,494,231.00 kWh

Energy costs: $132,209.56

2020 Total consumption: 1,558,321.20 kWh

Energy costs: $138,036.09

RH1:

2019 Total Consumption: 612,654.38 kWh

Energy costs: $54,268.92

The data shows that Crosby had more power consumption in 2020 than 2019. However, RH1 used less power in 2020 than in 2019.

4.1.2.4 ENERGY USE INTENSITY (EUI)

To get further insight into the buildings, the building's EUI was calculated and compared to the Energy Star's EUI reference table.

Crosby Gross Sq. ft. = 174,371 sq. ft.

Crosby 2019 EUI = 30.4 kBtu/ft^2

RH1 Gross Sq. ft. = 81,234 sq. ft.

RH 1 2019 EUI = 25.73 kBtu/ft^2

Overall, similar to Stockard/Martin, the collective data suggests that, occupied or not, Crosby units are more inefficient than their contemporary counterparts. The lack of proper insulation in the traditional buildings make the ACs work overtime and consume more power. In fact, despite having indivdual bathrooms in contemporary bathrooms, traditional halls tend to use more power.

4.2 ATHLETIC/SPORTING VENUES

As an NCAA-I University, the University of Mississippi is known for its athletics department. Football, Basketball, Softball are some of the major sports at the University having their individual stadiums. These stadiums are massive power consumers on campus during game season as well as off-season.

4.2.1 PAVILION

The Pavilion is the University's basketball stadium. Opened in 2016, the Pavilion replaced the Tad Smith Coliseum as the basketball stadium [6]. The Pavilion is equipped

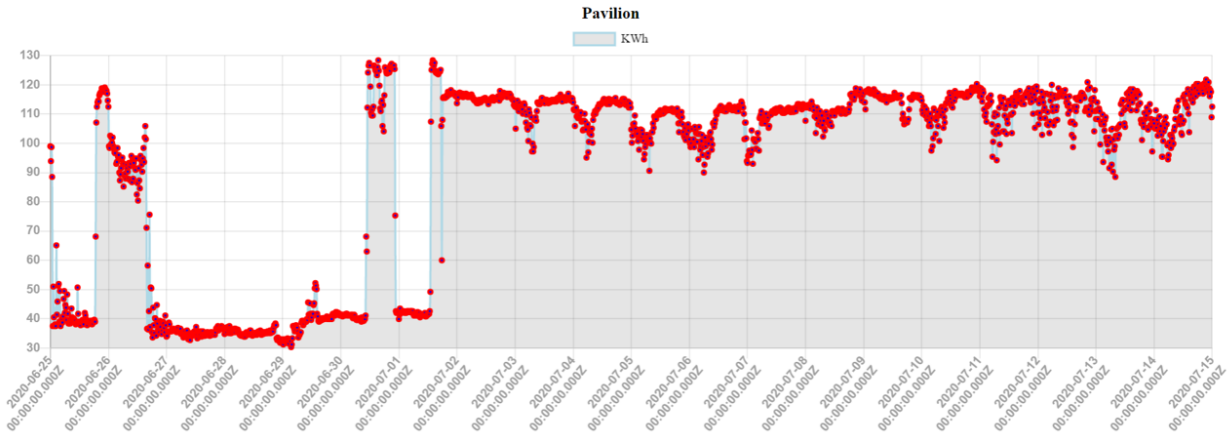

Figure 4.6: Graph showing the energy consumption at the Pavilion from 2020/06/25 - 2020/07/15

with 2 electricity meters, one each on the North and South ends.

After analyzing the data from 2019 and 2020, some invaluable insights were obtained into the power usage and raised some important questions about the consumption at unusual times.

4.2.1.1 SUMMER 2020

In the summer of 2020, when the University was under a complete shutdown, the Pavilion recorded some unusually high meter readings on the South end meter.

Starting July 1, the Pavilion's South end meter had an average energy consumption of about 112 kWh/15-min. This is almost double the average in June, which was around 54 kWh/15-min. Refer to Figure 4.6 and Figure 4.7.

June 25 – July 1:

Average consumption = 54.08 kWh/15-min

July 2 – July 14:

Minimum consumption = 88.5 kWh/15-min

Average consumption = 111.11 kWh/15-min

July 21 – July 31:

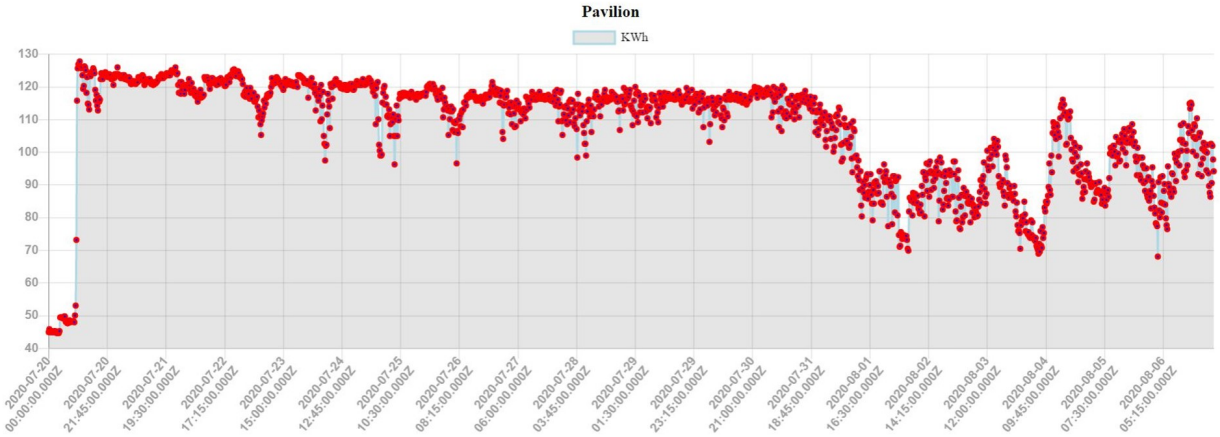

Figure 4.7: Graph showing the energy consumption at the Pavilion from 2020/07/21 - 2020/07/31

Average Consumption: 117.19 kWh

These statistics and graphs show that the average energy consumption during this month was almost double the consumption in the previous months. Moreover, the minimum consumption was more than the maximum consumption in the previous months. This shows that some of the big stadium lights were probably left on during the night times as well when the venue was probably not being used to the same capacity as during the day.

4.2.1.2 YEAR/YEAR COMPARISON

The data below is from February 1 - December 31 for each year. Both meters' data was combined to get the final consumption.

2019 Total Energy Consumption: 4,038,160.8 kWh

2019 Total Cost: $357,700

2020 Total Energy Consumption: 4,236,253.2 kWh

2020 Total Cost: $375,247

There was approximately 5% increase in power consumption in 2020 as compared to 2019. This was in a year when there were fewer games than the previous year as well as the stadium was running at half the capacity than previously. Despite that, there was more

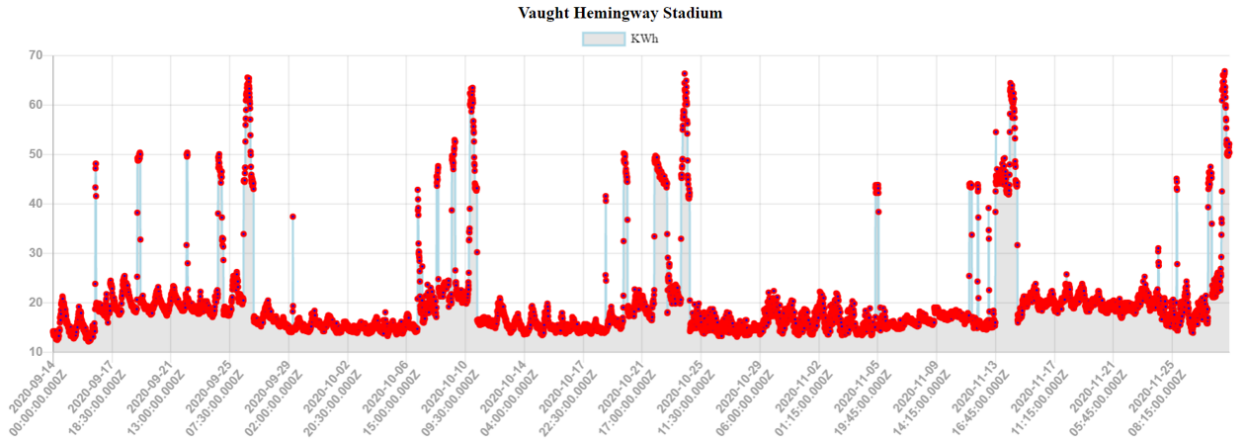

Figure 4.8: Graph showing the energy consumption at the Football stadium (7968) from 2020/09/01 - 2020/11/30

energy usage in 2020 than in 2019.

4.2.2 VAUGHT-HEMINGWAY STADIUM

The Vaught-Hemingway Stadium is the University's football stadium. The stadium is equipped with 7 electricity meters. However, for analysis purposes, only 2 of the 7 meters were used that had the highest energy usage. The location of these meters could not be confirmed by Facilities Management.

4.2.2.1 FOOTBALL SEASON VS. OFF-SEASON

Meters 7968 and 3749 were the two meters chosen for analyzing the energy consumption. As expected, both meters showed a spike in the energy consumption on home-game weekends, while the other days were moderately lower as shown in Figure 4.8 and Figure 4.9.

From 2020/09/01 – 2020/11/30 (both meters combined):

Max. 15-min consumption = 175.28 kWh

Avg. 15-min consumption = 55.914 kWh

Min. 15-min consumption = 27.36 kWh

However, in the month of December, when the football season and the Fall semester

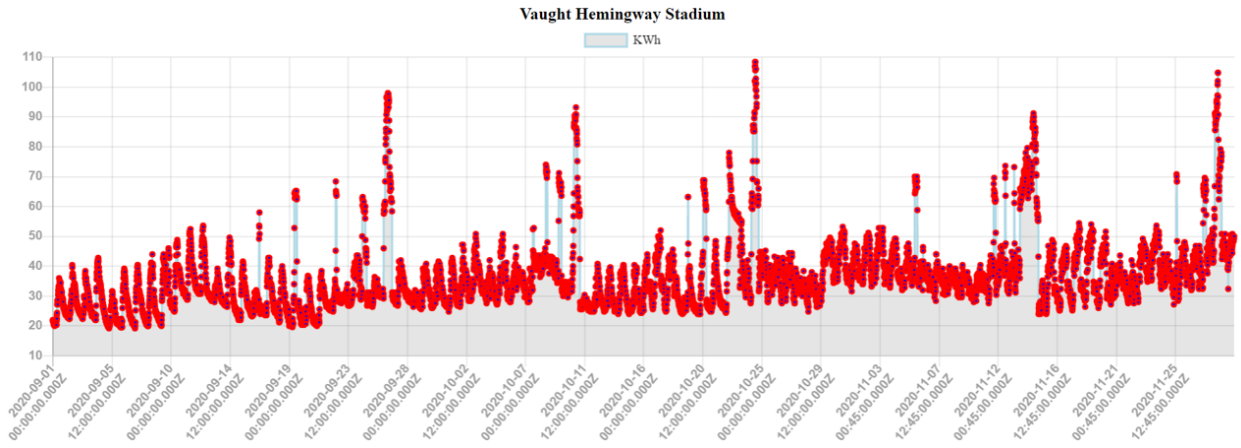

Figure 4.9: Graph showing the energy consumption at the Football stadium (3749) from 2020/09/01 - 2020/11/30

were both over, the stadium's average energy consumption was higher than that during the football season. Refer to Figure 4.11 and Figure **??**.

From 2020/11/30 – 2020/12/30 (both meters combined):

Max. 15-min consumption = 96.4 kWh

Avg. 15-min consumption = 71.85 kWh

Min. 15-min consumption = 49.36 KWh

Both the minimum and average consumption during the winter break is more than that during the regular semester. The average consumption in the winter is 28.5% more than that during the semester.

4.2.2.2 YEAR/YEAR COMPARISON

The data is from February 1 - December 31 of each year and for meters 3749 and 7968.

2019 Total Energy Consumption: 715,500 kWh

2019 Total Cost: $63,307.44

2020 Total Energy Consumption: 1,374,106 kWh

2020 Total Cost: $121,580.90

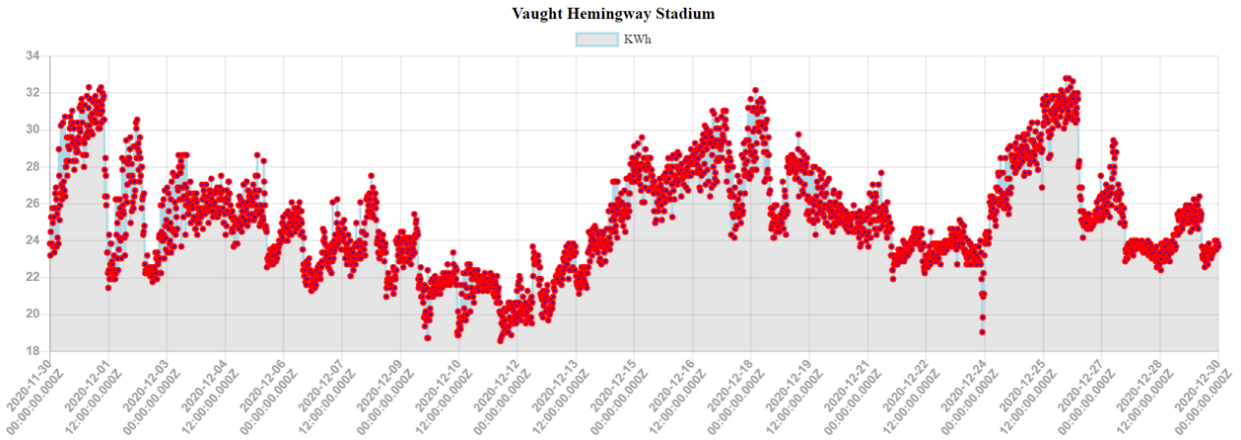

Figure 4.10: Graph showing the energy consumption at the Football stadium (7968) from 2020/11/30 - 2020/12/30

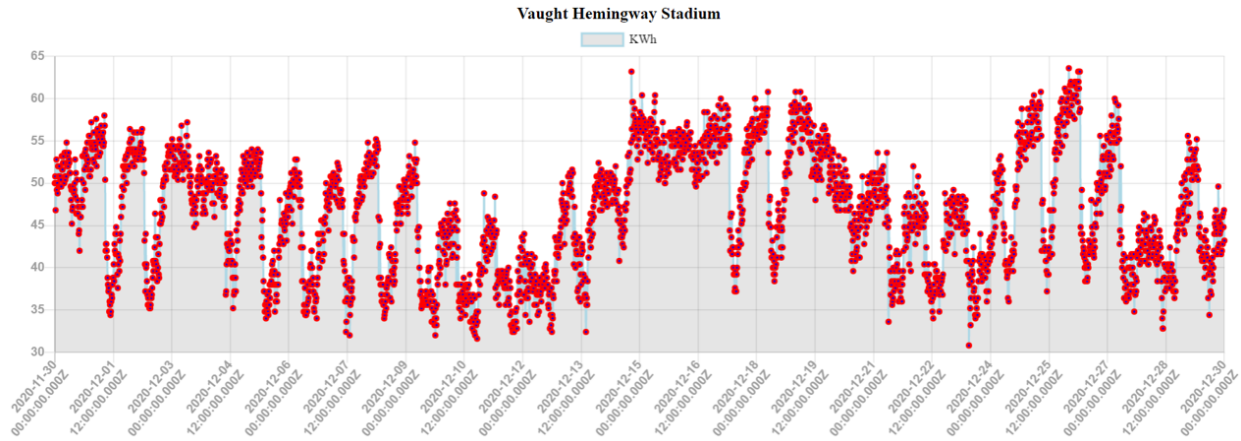

Figure 4.11: Graph showing the energy consumption at the Football stadium (3749) from 2020/11/30 - 2020/12/30

There was approximately 92% increase in the power consumption in 2020 as compared to 2019. Similar to the basketball stadium, the football stadium was also running on a lower stadium capacity of around 2,000 students as compared to the full capacity of over 9,000 people. Despite that, the football stadium used more power in 2020 than in 2019.

CHAPTER 5

CONCLUSION

It was generally observed that 2020 had more power consumption than 2019. This raises a lot of questions on multiple basis:

- Despite most buildings being closed for the summer and part of the Spring semester in 2020, most residential and athletic buildings used a lot higher energy as compared to 2019.
- 2019 was the hottest summer on record in the Northern Hemisphere [7]. This implies the ACs needing more power to keep the buildings at a set temperature. Despite that, the power consumption was higher in 2020.
- All athletic venues were reduced to half capacity in 2020, which should also lead to a reduced power usage, albeit not substantial.

Although not much information was able to be obtained from the Facilities Management, it was found that the University was required to run the buildings at a minimum energy consumption based on the state/federal regulations. Further research needs to be done on this and is only a speculative assumption at this time.

There is an enormous need for using such dashboards to analyze the University's performance in all utilities supplied. By using such dashboards, the University can save anywhere from \$10K - \$30K per building annually (or approximately \$2.5M on average). This money saved can instead be used on funding more research opportunities and bringing other programs for students.

This ambitious goal requires some bold steps to be taken by the University management.

- One of the major steps would be to deploy energy retrofitting projects in the older buildings on campus that still run on faulty/old equipment which require a lot more power. Although expensive in the short-term, these projects will help save millions of dollars over a period of a few years that will eventually benefit the University in the long-run.

- By creating an internal partnership between the Sustainability office and the Facilities Management office, a better plan can be developed to tackle such challenges.

- Facilities management needs more employees doing an energy audit for campus and finding areas of improvement.

- By partnering with the Computer Science department, Facilities management can employ student workers to expand the dashboard to cover natural gas, water, and sewage plants on campus as well as add more features such as future prediction models. These models can help understand the future energy demands and make the University better prepared to handle the loads.

Mississippi is well-behind their efforts in sustainability. The state's flagship University has the opportunity now to take the lead on introducing sustainability efforts on campus and in the state by collaborating with existing leaders in the field. It is time to take some bold steps towards creating a sustainable future and changing the stereotypes about Mississippi.

Overall, the project was a good initial point for starting a trend at the University and encouraging other students to take up similar sustainability projects. It also gave way for students in tech to explore sustainability as an avenue to apply their skills.

CHAPTER 6

FUTURE WORK

Due to the limited amount of time and the constraints in obtaining complete and accurate data from the Facilities management, this project was not able to reach its complete potential. However, there is a lot more work that can be done on this dashboard in the future.

- Add a campus heat map to display all the buildings on a map. This will allow the user to pick the buildings directly from the map.
- View the aggregated data per day, month, and year. Currently, the data is only available for every 15-minute interval.
- The User Interface is currently not as appealing to the general user as it should be. By making the interface more aesthetically pleasing, the dashboard can be made accessible to a larger audience and can be used for further research purposes by students, faculty, and staff.
- Add the option to compare two or more buildings with each other by stacking the graphs together. This will allow for a better comparison between buildings that have a similar structure/function.
- Allow the user to create and save their custom dashboards that they created using filters and comparisons. This will prevent the user from creating the same complex architecture over and over again.
- Make the data dynamic so it is updated daily on the dashboard. Currently, the data is inputted manually through an SQL script.

- Permanently host the dashboard on the Facilities Management website in order for better accessibility for the greater University community. Currently, the dashboard is hosted on the Computer Science department's server.

Besides the extra functionalities added to the dashboard, there is a lot that could have been improved in the way the research was conducted.

- On top of using the EUI to compare buildings, adding a per capita metric would also add an extra dimension to the research. This would allow to better understand some of the residential buildings' data.

- If given access to more data from before 2019, there could be a more thorough year/year comparison of the buildings to understand the needs.

CHAPTER 7

BIBLIOGRAPHY

[1] "Data Dashboards. Definition, Design Ideas plus 3 Examples." Klipfolio.com, www.klipfolio.com/resources/articles/what-is-data-dashboard.

[2] "Introduction." Vue.js, v3.vuejs.org/guide/introduction.html.

[3] "Axios." Npm, www.npmjs.com/package/axios.

[4] "Sequelize." Sequelize ORM, sequelize.org/.

[5] "Campus Performance." Sustainable Stanford - Stanford University, sustainable.stanford.edu/can action/campus-performance.

[6] "The Pavilion at Ole Miss." Wikipedia, Wikimedia Foundation, 11 Apr. 2021, en.wikipedia.org/wiki/The_Pavilion_at_Ole_Miss.

[7] "Summer 2019 Was Hottest on Record for Northern Hemisphere." Summer 2019 Was Hottest on Record for Northern Hemisphere — National Oceanic and Atmospheric Administration, www.noaa.gov/news/summer-2019-was-hottest-on-record-for-northern-hemisphere.