Cab Project - Supply
Group 01-01

Group Members:

Casey Lewis lewisc16@tcnj.edu

Alex D'Amico damicoa3@tcnj.edu

Thendral Prabu prabut1@tcnj.edu

Madison Franco francm11@tcnj.edu

Alex Panarese panarea1@tcnj.edu

Farhan Ahsan ahsanf1@tcnj.edu

Inception: “Executive Summary”

When designing the application, our primary stakeholder/market group that we had identified were Paul Romano and the TCNJ Environmental Sustainability Council as well as colleges or other organizations with an interest in sustainability via several different sources of energy.

As a result of this, our approach to solve this issue was to create an application that allowed for the user to view data relating to the energy sources that are available to the user in a variety of ways. The primary ways in which the data could be visualized was through the use of graphs and tables to allow users to see the trends in energy consumption through the different sources, as well as the numbers relating to the points on the graph as well. In addition to this, as new energy sources are made available, database administrators can easily add any corresponding data to the database in order to adapt to any changes that may come up.

The benefits of this software would primarily be that the application is well suited to the needs set forth by the TCNJ Environmental Sustainability Council by looking at the data in both line graph or table form as well as having tables within the database already set for any future data for green energy on campus.

One potential cost of the application would be that as this application replaces already existing software, the application would need to have a location to run and store any data. As a result of this, one possible option would be to host the application on a TCNJ server instead of another hosting service, which would also offer some substantial benefits as well in the form of added data security and reduced costs by avoiding paying for a subscription to a web hosting service.

Elaboration: Project Proposal and Specifications

I. Problem Statement

The electrical grid is a complex, interconnected system of power generating plants, transformers, substations, and power lines through which electricity is supplied from producers to consumers, with the three main components of the electricity supply chain consisting of generation, transmission and distribution. During the generation phase, energy from primary resources such as coal, natural gas, oil, and nuclear energy (nonrenewable sources), as well as wind, solar, geothermal, and hydropower (renewable sources) is converted into electricity, both a secondary source and energy carrier. There are two types of generation—centralized and decentralized—with the former referring to the common, widespread power production manufactured far away from consumption, and the latter occurring significantly closer to consumer demand. In order for centralized electricity to reach end users, power lines and transformers (used to “step up” or “step down” electrical voltages during different stages to increase efficiency, minimize loss, and ensure consumer safety) are relied upon in what is known as the next process: transmission. Finally, substations, smaller transformers, and distributor lines are used to complete the third component of the electricity value chain: distribution.

From the ability to operate appliances to the advancement in technological and medical services provided, electricity is an indispensable staple of modern life, with innumerable uses people may or may not take for granted. Electricity is a necessity, and the electrical grid remains a leading power source for hundreds of millions of residential, industrial, and commercial consumers. According to statista, in 2018, the world’s electricity consumption amounted to approximately 23,398 billion kilowatt hours, or 23,398 terawatt hours; U.S. consumption, alone, totaled approximately 4,194 terawatt, making it the second-largest electricity consumer after China. Electricity, however, is also a perishable commodity, with not only depleting effects of nonrenewable natural reserves, but far-encompassing negative implications and externalities to the environment, public health, and economy. While electricity, in and of itself, is a “clean and relatively safe form of energy when it is used,” the perils lie in the generation and transmission of the energy supply. When fossil fuels and other nonrenewable resources are used to generate electricity, a number of harmful pollutants, such as carbon dioxide and sulfur dioxide, are emitted into the air and water systems, contributing to climate change and global warming.

Today, a number of businesses and facilities are working progressively to reduce their carbon footprint. For that reason, we ask this question, “when is it most economical and least polluting for The College of New Jersey campus to produce its own power as opposed to using an electrical grid on both a site and source basis?

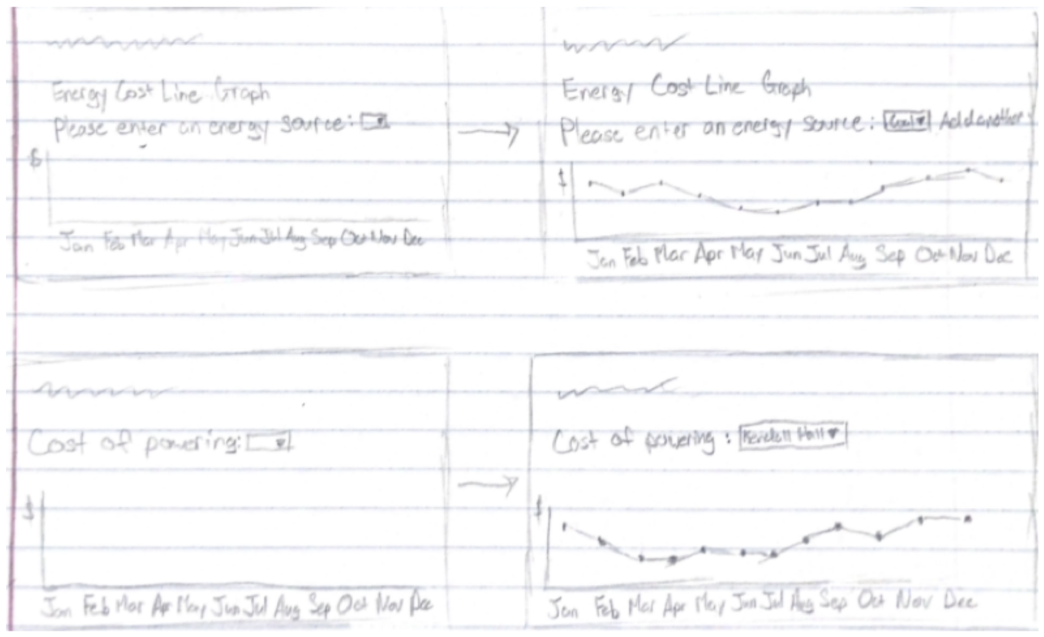
II. Objective

The objective of our module is to create a web application that provides a user important interpretations of data regarding TCNJ’s energy supply. This will be done by displaying the results of useful, complex and efficient queries from our database and constructing data models based on important data points. We will use those spreadsheets provided for us to use in Canvas that are keyed in on energy supply.

III. End Product Description

Our end product will be a web application created using python and flask. This application will use a Postgres database containing data about TCNJ’s energy supply to provide the user with useful information. We expect to see a best scenario end result by the research we gather from both data provided and outside research done regarding the relevant information and variables. We hope to see which option least impacts stakeholders and TCNJ as a whole.

UI Mockup and Use Cases


Energy Cost Line Graph
Please enter an energy source:
$
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Energy Cost Line Graph
Please enter an energy source: Add another
$
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Cost of powering:
$
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
Cost of powering: Eickhoff Hall
$
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

IV. Module Importance

Our module is important for the school because it provides needed information to the user about TCNJ's energy supply. Sustainability is very important and it is crucial that TCNJ is equipped with the necessary information to make decisions about how they will address this topic in the future. Our application will help TCNJ move forward in a way that is both economical and environment-friendly.

V. Research Plan

Our research plan is to calculate and find important values such as the average monthly cost, how this cost changes between months, the average monthly cost per type of meter, and the energy production. The majority of the data that we need to focus on has been given to us by our client, so we can focus our research on studying data rather than obtaining it. We will be analyzing the given data to find the most important data points and relationships that exist between different values. Also, the number of machine hours used. The costs to install and operate components represent additional operational expenses. The increase of usage is a major factor increasing energy supply because the more it is used through a community, the more supply of energy is increased that then increases the cost. Maily demand and supply are the easiest terms to put it. Manufacturing, installment, usage, maintenance. After completing our analysis, we can create meaningful queries and implement our database.

VI. Comparison to Other Systems

Our application is similar to other applications such as Energy Star Portfolio Manager or AASHE STARS, however our application is much more catered to TCNJ's demands to satisfy the sustainability plan set for 2020-2024.

VII. Other Applications

This application can be modified to allow for projections of energy costs in the future given TCNJ's sustainability commitment. In addition, the application can be further modified to allow for cost predictions of different energy sources that may be developed and considered for adoption at TCNJ in the future.

VIII. Performance

Since our web application is relatively small, with our database likely only containing a few thousand instances, application performance should be very efficient.

IX. Security

Github classroom provides security so security should not be a concern for our project. Once the application is live, we will allow two different types of users. One type, an administrator, will be able to add or change data. The other, a regular user, will only be able to view the data. This will prevent unauthorized users from making changes to the application. Once the project departs Github, only authorized users will be able to access and manipulate the data to prevent any misrepresentation of TCNJ's energy supply to the client, which can be implemented with Flask's user authentication functionality.

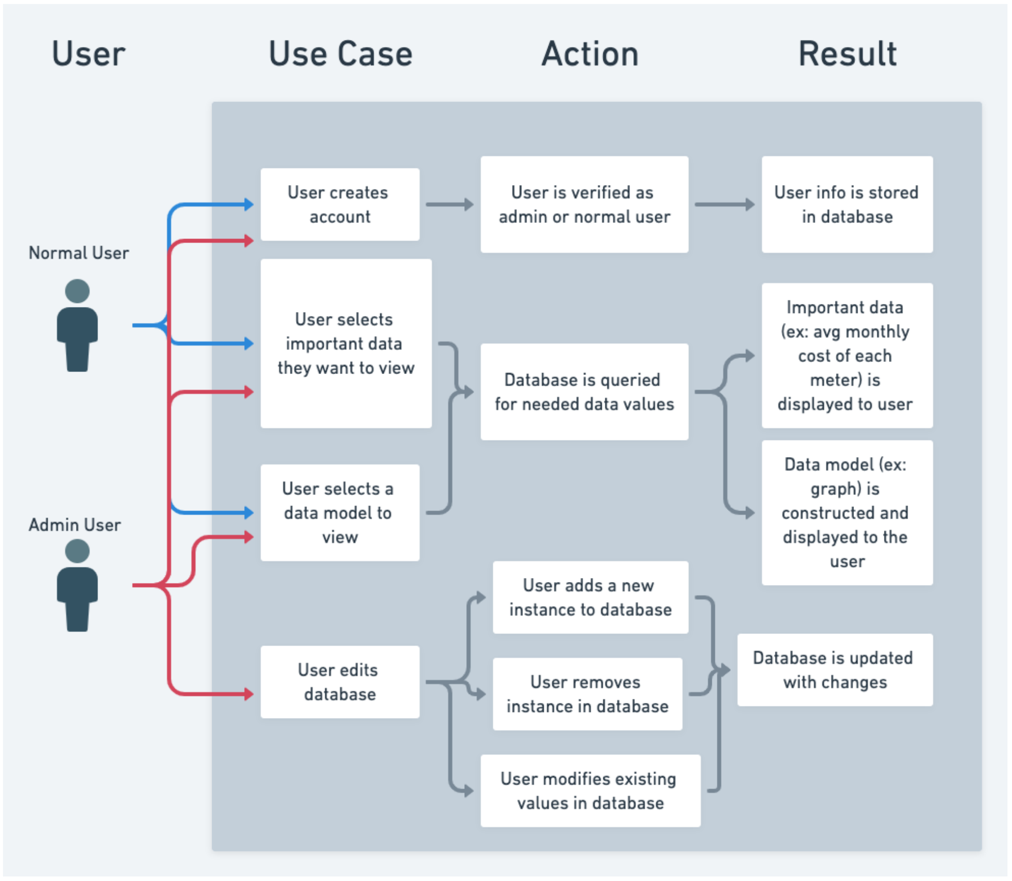
X. Backup and Recovery

Github provides our project a proven backup and recovery system for our project, so we will be leveraging it for this purpose. Once launched and delivered to TCNJ facilities, the project will be backed up on the TCNJ Cloud for simpler recovery. In case of TCNJ facility failure, together Flask and SQL have functionality for both local and remote server backups.

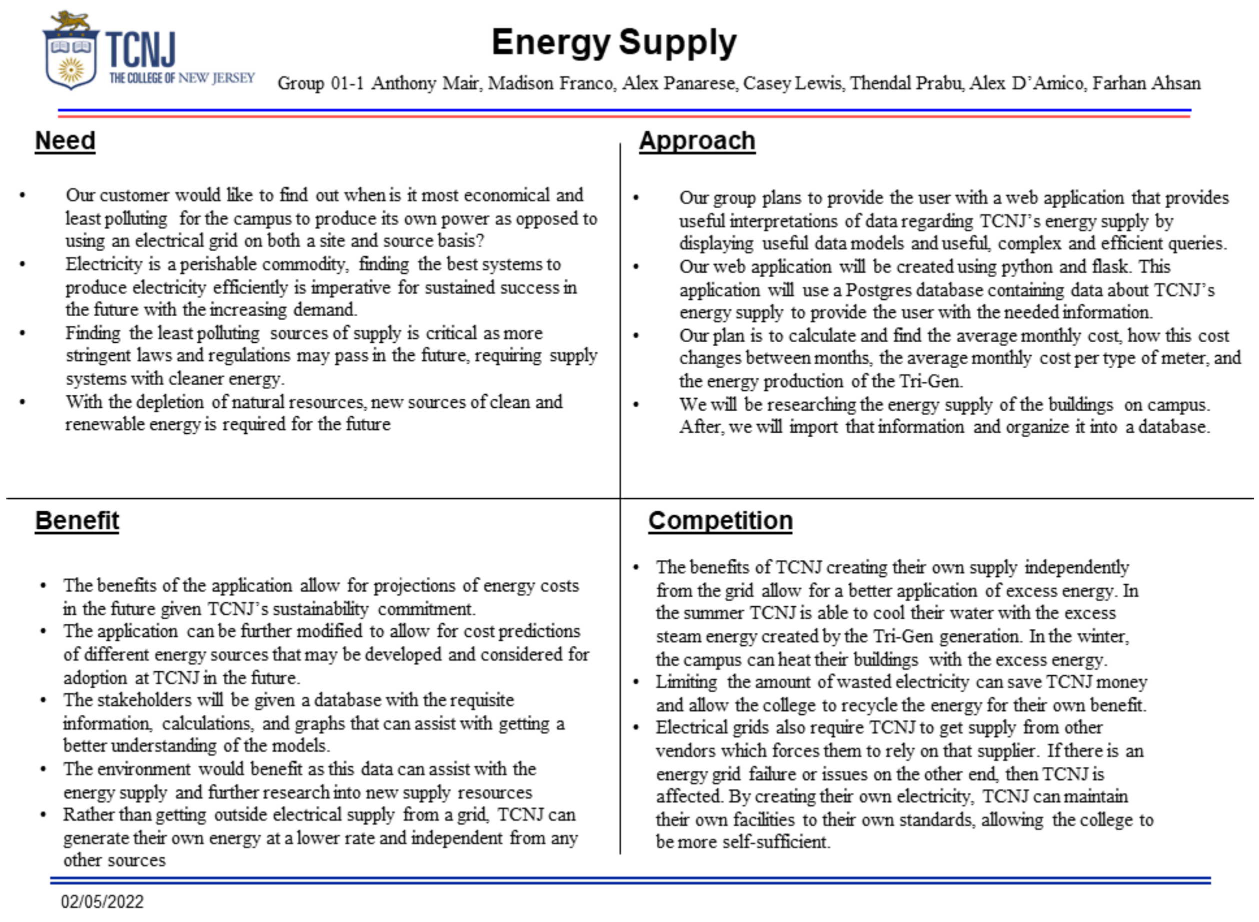
XI. Technological and Database Concepts

We are going to be implementing a relational database using Postgres on a dynamic web application created using python and flask. Understanding these technologies will be crucial to completing this project.

XII. Diagram


User
Use Case
Action
Result
Normal User
Admin User
User creates account
User is verified as admin or normal user
User info is stored in database
User selects important data they want to view
Database is queried for needed data values
Important data (ex: avg monthly cost of each meter) is displayed to user
User selects a data model to view
Data model (ex: graph) is constructed and displayed to the user
User adds a new instance to database
User edits database
User removes instance in database
Database is updated with changes
User modifies existing values in database

XIII. Quad Chart



Energy Supply

Group 01-1 Anthony Mair, Madison Franco, Alex Panarese, Casey Lewis, Thendal Prabu, Alex D'Amico, Farhan Ahsan

Need

- Our customer would like to find out when is it most economical and least polluting for the campus to produce its own power as opposed to using an electrical grid on both a site and source basis?
- Electricity is a perishable commodity, finding the best systems to produce electricity efficiently is imperative for sustained success in the future with the increasing demand.
- Finding the least polluting sources of supply is critical as more stringent laws and regulations may pass in the future, requiring supply systems with cleaner energy.
- With the depletion of natural resources, new sources of clean and renewable energy is required for the future

Approach

- Our group plans to provide the user with a web application that provides useful interpretations of data regarding TCNJ's energy supply by displaying useful data models and useful, complex and efficient queries.
- Our web application will be created using python and flask. This application will use a Postgres database containing data about TCNJ's energy supply to provide the user with the needed information.
- Our plan is to calculate and find the average monthly cost, how this cost changes between months, the average monthly cost per type of meter, and the energy production of the Tri-Gen.
- We will be researching the energy supply of the buildings on campus. After, we will import that information and organize it into a database.

Benefit

- The benefits of the application allow for projections of energy costs in the future given TCNJ's sustainability commitment.
- The application can be further modified to allow for cost predictions of different energy sources that may be developed and considered for adoption at TCNJ in the future.
- The stakeholders will be given a database with the requisite information, calculations, and graphs that can assist with getting a better understanding of the models.
- The environment would benefit as this data can assist with the energy supply and further research into new supply resources
- Rather than getting outside electrical supply from a grid, TCNJ can generate their own energy at a lower rate and independent from any other sources

Competition

- The benefits of TCNJ creating their own supply independently from the grid allow for a better application of excess energy. In the summer TCNJ is able to cool their water with the excess steam energy created by the Tri-Gen generation. In the winter, the campus can heat their buildings with the excess energy.
- Limiting the amount of wasted electricity can save TCNJ money and allow the college to recycle the energy for their own benefit.
- Electrical grids also require TCNJ to get supply from other vendors which forces them to rely on that supplier. If there is an energy grid failure or issues on the other end, then TCNJ is affected. By creating their own electricity, TCNJ can maintain their own facilities to their own standards, allowing the college to be more self-sufficient.

02/05/2022

Elaboration: Design

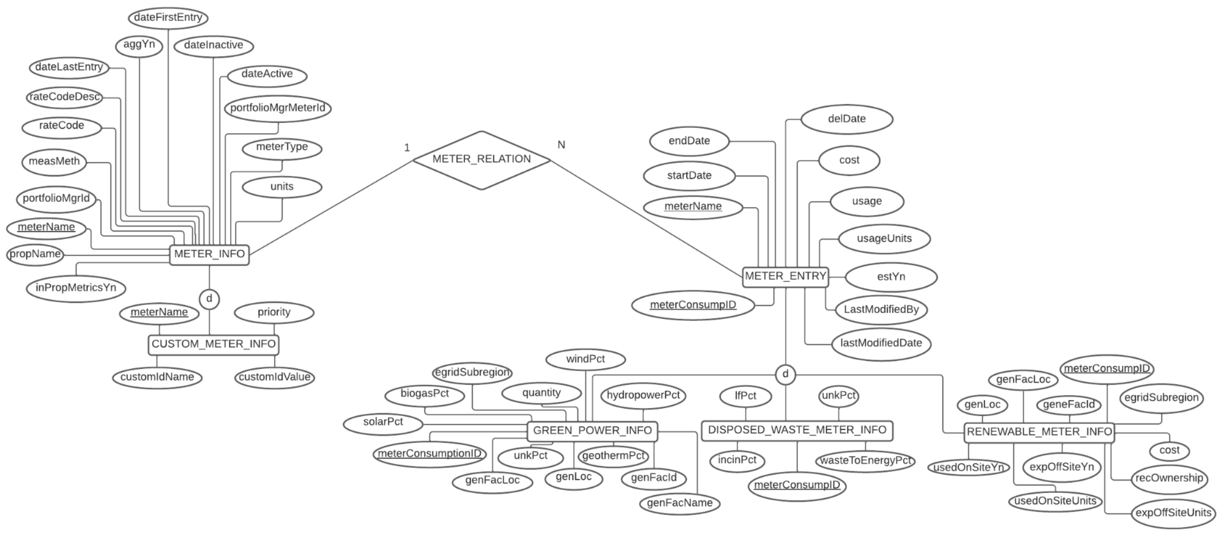
Database Model

Estimations

There are about 1,430 meter entries in the data given. Since this is the bulk of our data, we can assume that the rest of our records will add up to substantially less than this number. In total, we expect between 1,500 and 2,000 records,

We expect to need about 5-10 searches in our web application. The majority of these searches will access data in the METER_ENTRY table since this is where the most important data is. Many of these searches will use aggregate functions to obtain meaningful values such as average costs.

ER Diagram



Entity Types and Relationships

METER_INFO is a strong entity type with the primary key meterName.

CUSTOM_METER_INFO is a subclass of METER_INFO. It is identified by the foreign key meterName and it exists if a meter has a custom name.

METER_ENTRY is a strong entity type with the primary key meterConsumptionId.

GREEN_POWER_INFO, RENEWABLE_METER_INFO, and DISPOSED_WASTE_METER_INFO are subclasses of METER_ENTRY. They are identified by the foreign key meterConsumptionId and they exist if a meter entry has data on them.

The only relationship type is: “a meter has many meter entries”. This means the relationship METER_INFO:METER_ENTRY is a 1:N relationship. This relationship is identified using the foreign key approach. METER_ENTRY has the foreign key meterName to identify the relationship.

Relations and attributes

METER_INFO: propertyName, portfolioManagerId, portfolioManagerMeterId, <u>meterName</u>, meterType, units, measurementMethod, includedInPropertyMetricsYn, dateActive, dateInactive, dateFirstEntry, dateLastEntry, aggregateYn, rateCode, rateCodeDescription

CUSTOM_METER_INFO: <u>meterName</u>, customIdName, customIdValue, priority

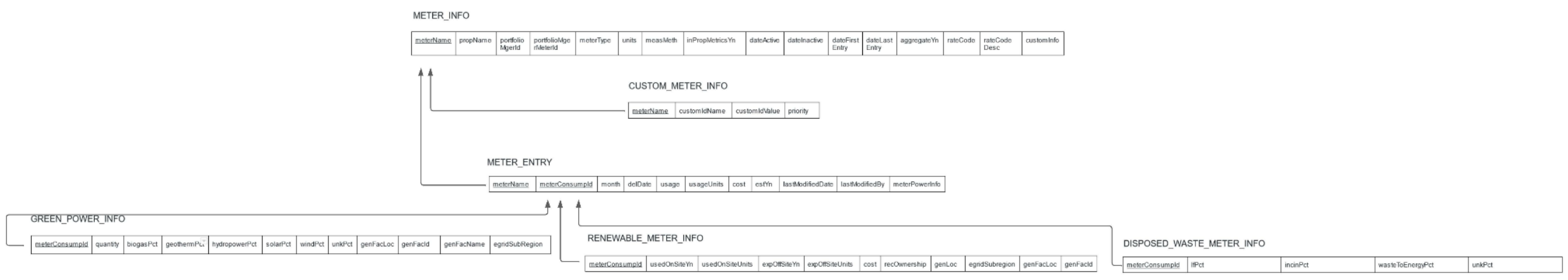
METER_ENTRY: <u>meterName</u>, <u>meterConsumptionId</u>, month, deliveryDate, usage, usageUnits, cost, estimationYn, lastModifiedDate, lastModifiedBy

GREEN_POWER_INFO: <u>meterConsumptionId</u>, quantity, biogasPercent, geothermalPercent, hydropowerPercent, solarPercent, windPercent, unknownPercent, generationFacilityLocation, generationFacilityId, generationFacilityName, egridSubRegion

RENEWABLE_METER_INFO: <u>meterConsumptionId</u>, usedOnSiteYn, usedOnSiteUnits, exportedOffSiteYn, exportedOffSiteUnits, cost, recOwnership, generationLocation, egridSubregion, generationFacilityLocation, generationFacilityId

DISPOSED_WASTE_METER_INFO: <u>meterConsumptionId</u>, landfillPercent, incinerationPercent, incinerationPercent, unknownPercent

Relational Schema


METER_INFO
meterName | propName | portfolio MgerId | portfolioMge rMeterId | meterType | units | measMeth | inPropMetricsYn | dateActive | dateInactive | dateFirst Entry | dateLast Entry | aggregateYn | rateCode | rateCode Desc | customInfo
CUSTOM_METER_INFO
meterName | customIdName | customIdValue | priority
METER_ENTRY
meterName | meterConsumpId | month | delDate | usage | usageUnits | cost | esfYn | lastModifiedDate | lastModifiedBy | meterPowerInfo
GREEN_POWER_INFO
meterConsumpId | quantity | biogasPct | geothermPct | hydropowerPct | solarPct | windPct | unkPct | genFacLoc | genFacId | genFacName | egridSubRegion
RENEWABLE_METER_INFO
meterConsumpId | usedOnSiteYn | usedOnSiteUnits | expOffSiteYn | expOffSiteUnits | cost | recOwnership | genLoc | egridSubregion | genFacLoc | genFacId
DISPOSED_WASTE_METER_INFO
meterConsumpId | lfPct | incinPct | wasteToEnergyPct | unkPct

Database Design

Tables in BCNF

METER_INFO: propertyName, portfolioManagerId, portfolioManagerMeterId, <u>meterName</u>, meterType, units, measurementMethod, includedInPropertyMetricsYn, dateActive, dateInactive, dateFirstEntry, dateLastEntry, aggregateYn, rateCode, rateCodeDescription

METER_INFO is in BCNF because it has the primary key meterName and every other attribute is functionally dependent on it.

CUSTOM_METER_INFO: <u>meterName</u>, customIdName, customIdValue, priority

CUSTOM_METER_INFO is in BCNF because it has the primary key meterName and every other attribute is functionally dependent on it.

METER_ENTRY: <u>meterName</u>, <u>meterConsumptionId</u>, month, year, usage, ~~usageUnits~~, cost, estimationYn, lastModifiedDate, lastModifiedBy

METER_ENTRY was not originally in BCNF because one of our attributes, usageUnits, is functionally dependent on meterName while every other attribute is functionally dependent on the primary key meterConsumptionId. To fix this, we removed usageUnits from the relation, since it already exists in METER_INFO.

*GREEN_POWER_INFO: <u>meterConsumptionId</u>, quantity, biogasPercent, geothermalPercent, hydropowerPercent, solarPercent, windPercent, unknownPercent, ~~generationFacilityLocation, generationFacilityId, generationFacilityName, egridSubRegion,~~ **generationLocationId***

GREEN_POWER_INFO was not originally in BCNF because certain values were dependent on generationFacilityId. After removing those attributes and moving them to their own entity, it is in BCNF because it has the primary key meterConsumptionId and every other attribute is functionally dependent on it.

*RENEWABLE_METER_INFO: <u>meterConsumptionId</u>, usedOnSiteYn, usedOnSiteUnits, exportedOffSiteYn, exportedOffSiteUnits, cost, recOwnership, ~~generationLocation, egridSubregion, generationFacilityLocation, generationFacilityId~~, **generationLocationId***

RENEWABLE_METER_INFO was not originally in BCNF because certain values were dependent on generationFacilityId. After removing those attributes and moving them to their own entity, it is in BCNF because it has the primary key meterConsumptionId and every other attribute is functionally dependent on it.

GENERATION_FACILITY: generationFacilityId, generationLocation, egridSubregion, generationFacilityLocation

GENERATION_FACILITY is a new strong entity. It is in BCNF because it has the primary key generationFacilityId and every other attribute is functionally dependent on it.

DISPOSED_WASTE_METER_INFO: meterConsumptionId, landfillPercent, incinerationPercent, incinerationPercent, unknownPercent

DISPOSED_WASTE_METER_INFO is in BCNF because it has the primary key meterConsumptionId and every other attribute is functionally dependent on it.

Views

1) This view gets the average cost/usage of every month for each individual meter. It can be used by queries to obtain information about which meters cost the most/least per month.

```
CREATE VIEW MeterMonthlyCostAvg
SELECT METER_INFO.meterName, METER_INFO.meterType,
METER_INFO.meterUnits, METER_ENTRY.month, AVG(METER_ENTRY.usage) as
averageUsage, AVG(METER_ENTRY.cost) as averageCost
FROM METER_ENTRY LEFT JOIN METER_INFO ON METER_ENTRY.meterName =
METER_INFO.meterName
GROUP BY METER_INFO.meterName, METER_ENTRY.month;
```

2) This view gets the average cost/usage of every month for each type of energy. It can be used by queries to obtain information about which energy sources cost the most/least per month.

```
CREATE VIEW TypeMonthlyCostAvg
```

```
SELECT METER_INFO.meterType, METER_INFO.meterUnits, METER_ENTRY.month,
AVG(METER_ENTRY.usage) as averageUsage, AVG(METER_ENTRY.cost) as
averageCost
FROM METER_ENTRY LEFT JOIN METER_INFO ON METER_ENTRY.meterName =
METER_INFO.meterName
GROUP BY METER_INFO.meterType, METER_ENTRY.month;
```

3) *This view gets the average total cost of every month. It can be used by queries to obtain information about which months are the most expensive.*

```
CREATE VIEW OverallMonthlyCostAvg
SELECT month, AVG(cost) as averageCost
FROM METER_ENTRY
GROUP BY month;
```

Queries

1) Get average total cost of every month

```
SELECT *
FROM OverallMonthlyCostAvg;
```

2) Get month with the highest average total cost

```
SELECT month, MAX(averageCost)
FROM OverallMonthlyCostAvg;
```

3) Get month with the lowest average total cost

```
SELECT month, MIN(averageCost)
FROM OverallMonthlyCostAvg;
```

4) Get average monthly cost of every meter for every month

```
SELECT *
FROM MeterMonthlyCostAvg
ORDER BY meterName, month;
```

5) Get the month that has the highest monthly average cost for each meter

```
SELECT meterName, meterType, month, MAX(averageCost)
FROM MeterMonthlyCostAvg
GROUP BY meterName
ORDER BY meterName;
```

6) Get the month that has the lowest monthly average cost for each meter

```
SELECT meterName, meterType, month, MIN(averageCost) FROM
MeterMonthlyCostAvg GROUP BY meterName ORDER BY meterName;
```

7) Get average monthly cost of every meter type for every month

```
SELECT *
FROM TypeMonthlyCostAvg
ORDER BY meterType, month;
```

8) Get the month that has the highest monthly average cost for each meter type

```
SELECT meterType, month, MAX(averageCost)
FROM TypeMonthlyCostAvg
GROUP BY meterType
ORDER BY meterType;
```

9) Get the month that has the lowest monthly average cost for each meter

```
SELECT meterType, month, MIN(averageCost)
FROM TypeMonthlyCostAvg
GROUP BY meterType
ORDER BY meterType;
```

Construction: Tables, Queries, and User Interface

Tables and Queries Release (v5.0.0)

https://github.com/TCNJ-degoodj/cab-project-1-1/releases/tag/v5.0.0

Tables:

```sql
DROP VIEW IF EXISTS MeterMonthlyCostAvg;
DROP VIEW IF EXISTS TypeMonthlyCostAvg;
DROP VIEW IF EXISTS OverallMonthlyCostAvg;
DROP TABLE IF EXISTS GREEN_POWER_INFO;
DROP TABLE IF EXISTS DISPOSED_WASTE_METER_INFO;
DROP TABLE IF EXISTS RENEWABLE_METER_INFO;
DROP TABLE IF EXISTS GENERATION_FACILITY;
DROP TABLE IF EXISTS METER_ENTRY;
DROP TABLE IF EXISTS CUSTOM_METER_INFO;
DROP TABLE IF EXISTS METER_INFO;


CREATE TABLE METER_INFO (
meterName varchar(50) PRIMARY KEY,
propertyName varchar(255),
portfolioManagerId varchar(7),
portfolioManagerMeterId varchar(8),
meterType varchar(255),
units varchar(255),
measurementMethod varchar(255),
includedInPropertyMetricsYn BOOLEAN,
dateActive Date,
dateInactive Date,
dateFirstEntry Date,
dateLastEntry Date,
aggregateYn BOOLEAN,
rateCode varchar(10),
rateCodeDescription varchar(255)
);


CREATE TABLE CUSTOM_METER_INFO (
meterName varchar(50) REFERENCES METER_INFO,
customIdName varchar(255),
customIdValue varchar(255),
priority int
);


CREATE TABLE METER_ENTRY (
meterConsumptionId bigint PRIMARY KEY,
meterName varchar(50) REFERENCES METER_INFO,
month int,
year int,
usage NUMERIC(10,2),
cost NUMERIC(10,2),
estimationYn BOOLEAN,
lastModifiedDate Date,
lastModifiedBy varchar(255)
);


CREATE TABLE GENERATION_FACILITY (
generationFacilityId varchar(255) PRIMARY KEY,
generationLocation varchar(255),
egridSubregion varchar(255),
generationFacilityLocation varchar(255)
);


CREATE TABLE GREEN_POWER_INFO (
meterConsumptionId int REFERENCES METER_ENTRY,
quantity int,
biogasPercent NUMERIC(5,3),
geothermalPercent NUMERIC(5,3),
hydropowerPercent NUMERIC(5,3),
solarPercent NUMERIC(5,3),
windPercent NUMERIC(5,3),
unknownPercent NUMERIC(5,3),
generationFacilityId varchar(255) REFERENCES GENERATION_FACILITY
);


CREATE TABLE RENEWABLE_METER_INFO (
meterConsumptionId int REFERENCES METER_ENTRY,
usedOnSiteYn BOOLEAN,
usedOnSiteUnits int,
exportedOffSiteYn BOOLEAN,
exportedOffSiteUnits int,
Cost NUMERIC(20,2),
recOwnership varchar(255),
generationFacilityId varchar(255) REFERENCES GENERATION_FACILITY
);
```

```
CREATE TABLE DISPOSED_WASTE_METER_INFO (
meterConsumptionId int REFERENCES METER_ENTRY,
landfillPercent NUMERIC(5,3),
incinerationPercent NUMERIC(5,3),
unknownPercent NUMERIC(5,3)
);


CREATE VIEW MeterMonthlyCostAvg AS
SELECT METER_INFO.meterName, METER_INFO.meterType, METER_INFO.units, METER_ENTRY.month,
AVG(METER_ENTRY.usage)::numeric(10,2)  as averageUsage,
AVG(METER_ENTRY.cost)::numeric(10,2)  as averageCost
FROM METER_ENTRY LEFT JOIN METER_INFO ON METER_ENTRY.meterName = METER_INFO.meterName
GROUP BY METER_INFO.meterName, METER_ENTRY.month;


CREATE VIEW TypeMonthlyCostAvg AS
SELECT METER_INFO.meterType, METER_ENTRY.month, AVG(METER_ENTRY.usage)::numeric(10,2) as
averageUsage, AVG(METER_ENTRY.cost)::numeric(10,2) as averageCost
FROM METER_ENTRY LEFT JOIN METER_INFO ON METER_ENTRY.meterName = METER_INFO.meterName
GROUP BY METER_INFO.meterType, METER_ENTRY.month;


CREATE VIEW OverallMonthlyCostAvg AS
SELECT month, AVG(cost)::numeric(10,2) as averageCost
FROM METER_ENTRY
GROUP BY month;
```

Queries:

```
-- Relevant Queries for OverallMonthlyCostAvg
-- Total average cost of every month
SELECT *
FROM OverallMonthlyCostAvg
ORDER BY month;


-- Month with highest average cost
SELECT * FROM OverallMonthlyCostAvg
WHERE averageCost
IN
(SELECT MAX(averageCost)
FROM OverallMonthlyCostAvg);
```

```sql
-- Month with lowest average cost
SELECT * FROM OverallMonthlyCostAvg
WHERE averageCost
IN
(SELECT MIN(averageCost)
FROM OverallMonthlyCostAvg);


-- Relevant Queries for MeterMonthlyCostAvg
-- Get all monthly averages for a certain meter
SELECT *
FROM MeterMonthlyCostAvg
WHERE meterName = ?
ORDER BY month;


-- Get all monthly averages for a certain month
SELECT *
FROM MeterMonthlyCostAvg
WHERE month = ?
ORDER BY meterName;


-- Month with highest average cost per meter
SELECT meterName, meterType, month, averageCost
FROM MeterMonthlyCostAvg
WHERE averagecost IN
(SELECT MAX(averageCost)
FROM MeterMonthlyCostAvg
GROUP BY meterName)
ORDER BY meterName;


-- Month with lowest average cost per meter
SELECT meterName, meterType, month, averageCost
FROM MeterMonthlyCostAvg
WHERE averagecost IN
(SELECT MIN(averageCost)
FROM MeterMonthlyCostAvg
GROUP BY meterName)
ORDER BY meterName;
```

```
-- Relevant queries for TypeMonthlyCostAvg
-- Get all monthly averages for a certain meter
type
SELECT *
FROM TypeMonthlyCostAvg
WHERE meterType = ?
ORDER BY month;


-- Get the monthly averages for each meter type
a certain month
SELECT *
FROM TypeMonthlyCostAvg
WHERE month = ?
ORDER BY meterType;


-- Average monthly cost per meter type
SELECT meterType, AVG(averageCost)
FROM TypeMonthlyCostAvg
GROUP BY meterType;


-- Month with highest average cost per meter
type
SELECT meterType, month, averageCost
FROM TypeMonthlyCostAvg
WHERE averagecost IN
(SELECT MAX(averageCost)
FROM TypeMonthlyCostAvg
GROUP BY meterType)
ORDER BY meterType;


-- Month with lowest average cost per meter
type
SELECT meterType, month, averageCost
FROM TypeMonthlyCostAvg
WHERE averagecost IN
```

```
 (SELECT MIN(averageCost)
 FROM TypeMonthlyCostAvg
 GROUP BY meterType)
 ORDER BY meterType;
```

User Interface Release (v5.1.0)

https://github.com/TCNJ-degoodj/cab-project-1-1/releases/tag/v5.1.0

Transition: Maintenance

https://github.com/TCNJ-degoodj/cab-project-1-1

Transition: Product Hand Over

Public Repository: https://github.com/caseyjohn47/cab-project-public
Final Release in Private Repository:
https://github.com/TCNJ-degoodj/cab-project-1-1/releases/tag/v7.0.0