1. Jaya Sabarish Reddy Remala (jr6421)
2. Sumedh Sandeep Parvatikar (sp7479)

SHEMS - PDS Project

Smart Home Energy Management System **20th December 2023**

https://github.com/sumedhsp/CS-GY-6083-SHEMS

Formatted: Right

Introduction:

We will be designing an efficient Smart Home Energy Management System (SHEMS) using a relational data model to store and display necessary information to the customer. We will be dividing our project into two parts according to the guidelines. In the first part we will deep dive into the relational schema for the system. This report contains all the details from ER diagram to the relational schema along with SQL queries for some questions provided in the handout. For our project, we will be using **PostgreSQL** to design the database and the respective relational model.

As the second part of the project, we implemented a web-based interface where users can log in and view the energy consumption of their service locations. Moreover, they can also create different service locations in the database and assign different devices from a pre-stored list. We implemented the web-based interface using **Django** as our backend framework and **HTML**, **Bootstrap CSS** & **JavaScript** at the front end. In this project, we explored various ways of exchanging information with databases like Raw SQLs, ORMs (their advantages over SQLs, although not used in the project), and connection objects to create cursors and execute multiple queries in order. In addition, we also learnt about different kinds of vulnerabilities the database is exposed to and different techniques to overcome these. We will be discussing these topics in the upcoming sessions of the report on how we handled such security issues. To provide a **user-friendly** and **hassle-free experience**, we employed a simple and aesthetic frontend exposing certain functionalities and information to the users in the initial dashboard for quick access. Apart from that, we generated charts and graphs with the data from the dataset for visual representation of the energy consumption and other factors. We were also able to apply concepts learnt in the class like *Transactions, authentications* using *hashing, hiding sensitive information* to the outer world and many more. We will delve deeper into the concepts and as we proceed in the report.

Updated Schema:

As we started our analysis for the second part of the project with different features, we observed that there had to be some changes done to the previous schema to hold more information and allow functionalities like authentication, service location labeling and more. Hence, we introduced some additional columns which are highlighted in the below schema. More details about the attributes can be found in the section below one.

Address(**address_id**, unit_number, address_line, city, state, zipcode)

Customers(**customer_id**, email, password, phone_num, dob, first_name, Last_name, *billing_address_id*)

ServiceLocations(**sl_id**, *customer_id, address_id*, service_location_label, start_date, apt_area, num_bedrooms, num_occupants)

Device(**model_id**, device_type, model_number, brand, release_year, manufacture_year)

EnrolledDevices(**enrolled_device_id**, *sl_id, model_id*)

DeviceData(**data_id**, *enrolled_device_id*, event_timestamp, event_label, event_value)

EnergyPrices(**zipcode**, **price_timestamp**, price_per_kwh)

1. **Address Table:**
    - **Columns:**
        - `address_id`: Serial primary key.
        - `unit_number`: VARCHAR(50), not null.
        - `address_line`: VARCHAR(100), not null.
        - `city`: VARCHAR(100), not null.
        - `state`: VARCHAR(2), not null.
        - `zipcode`: INT, not null.

This table is designed to store address information, with a unique identifier (`address_id`) as a **primary key** and details such as unit number, address line, city, state, and zipcode.

2. **Customers Table:**
    - **Columns:**
        - `customer_id`: Serial primary key.
        - `first_name`: VARCHAR(100), not null.
        - `last_name`: VARCHAR(100), not null.
        - `email`: VARCHAR(50), not null.
        - `phone_num`: VARCHAR(10), not null.
        - `dob`: date, not null.
        - `password`: VARCHAR(100), not null.
        - `billing_address_id`: INT, not null, foreign key referencing `Address(address_id)`.

This table represents customer information with a unique identifier (`customer_id`) as **the primary key**. It also includes the *billing_address_id* as a **foreign Key**, referencing the `Address` table. New

attributes email, password, dob and phone_num were added to this table. Email and Password attributes were essential in authentication and other aspects like session handling. Phone_num and DOB attributes provided more information about the user as a customer and hence we thought it would make sense to capture this information.

3. **ServiceLocations Table:**
    - **Columns:**
        - `sl_id`: Serial primary key.
        - `service_location_label`: varchar(50), not null
        - `customer_id`: INT, foreign key referencing `Customers(customer_id)`
        - `address_id`: INT, UNIQUE foreign key referencing `Address(address_id)`
        - `start_date`: DATE.
        - `apt_area`: DECIMAL(18,6).
        - `num_bedrooms`: INT.
        - `num_occupants`: INT.

This table stores information about service locations, including the **sl_id** as **primary Key,** service location label giving freedom to the users to have their nicknames for every service location they reside, start date, apartment area, number of bedrooms, number of occupants, and **foreign keys** *customer_id,address_id* (Unique) referencing the `Customers` and `Address` tables. A new attribute Service_location_Label was added to the ServiceLocations table to provide the user the ability to label their respective service location and distinguish one from the other apart from the address of the service location

4. **Device Table:**
    - **Columns:**
        - `model_id`: VARCHAR(100) primary key.
        - `device_type`: VARCHAR(100), not null.
        - `model_number`: VARCHAR(100), not null.
        - `brand`: VARCHAR(100).
        - `release_year`: INT.
        - `manufacture_year`: INT.

This table represents information about electronic devices, with a unique identifier (`model_id`) as the **primary key** and details such as device type, model number, brand, release year, and manufacture year.

Commented [2]: @jr6421@nyu.edu

Commented [3]: 4) Web Based User Interface
a) Authorization Module
b) Basic Features (CRUD Operations)
c) Charts and Graphs
d) SQL Injection & Cross Site Scripting Attacks
e) Transactions & Concurrency
e) Few Query Examples
f) Website Screenshots
g) Extra Features

Commented [4]: 💯

5. **EnrolledDevices Table:**
    - **Columns:**
        - `enrolled_device_id`: Serial primary key.
        - `sl_id`: INT, not null, foreign key referencing `ServiceLocations(sl_id)`.
        - `model_id`: VARCHAR(64), not null, foreign key referencing `Device(model_id)`.

This table links devices to service locations. It includes a unique identifier (`enrolled_device_id`) as the **primary key** and foreign keys *sl_id*, *model_id* referencing both the `ServiceLocations` and `Device` tables.

6. **DeviceData Table:**
    - Stores data related to events from enrolled devices.
    - **Columns:**
        - `data_id` (Serial, Primary Key): Unique identifier for each data entry.
        - `enrolled_device_id` (INT, NOT NULL): Foreign key referencing `EnrolledDevices(enrolled_device_id)`.
        - `event_timestamp` (TIMESTAMP): Timestamp of the event.
        - `event_label` (VARCHAR(100), NOT NULL): Label describing the event.
        - `event_value` (DECIMAL(18,6)): Value associated with the event.

This table stores the records sent by the smart devices being **data_id** as the **primary key and** *enrolled_device_id* as the foreign key references Enrolled Devices Table follows the event timestamp, label and value as the other attributes.

7. **EnergyPrices Table:**
    - **Columns:**
        - `zipcode`: INT, not null.
        - `price_timestamp`: TIMESTAMP, not null.
        - `price_per_kwh`: DECIMAL(18,6), not null.
        - **Primary Key:** (`zipcode`, `price_timestamp`).

This table stores energy prices with a **primary key** consisting of `zipcode` and `price_timestamp`. It includes the price per kilowatt-hour (`price_per_kwh`) at a specific timestamp for a given zipcode.

These tables are interconnected using foreign keys to establish relationships between different entities in the schema. The schema provides a structured way to store and organize information related to addresses, customers, service locations, devices, enrolled devices, and energy prices.

Normalization:

Normalization is the process of organizing data to reduce redundancy and improve data integrity. The schema exhibits normalization principles:

- **Address Table:**
  - Normalized by having a separate table for addresses, reducing redundancy.
  - Address information is not duplicated; instead, references (foreign keys) are used.
- **Customers Table:**
  - Normalized by using a separate table for customer information.
  - The `billing_address_id` foreign key a relationship with the `Address` table, avoiding duplicate address data.
- **ServiceLocations Table:**
  - Normalized by using separate tables for service location, customer, and address information.
  - Foreign keys link to the `Customers` and `Address` tables, preventing redundancy.
- **Device Table:**
  - Normalized by having a separate table for device information.
  - Device details are not duplicated across multiple tables.
- **EnrolledDevices Table:**
  - Normalized by using foreign keys to establish relationships with the `ServiceLocations` and `Device` tables.
  - Device and service location details are not duplicated.
- **DeviceData Table:**
  - Normalized by using a foreign key to establish a relationship with the `EnrolledDevices` tables.
  - Event_label and Event_value changes according to the record and device.
- **EnergyPrices Table:**
  - Normalized by using a composite primary key to represent the unique identifier for energy prices.
  - The table structure avoids redundant storage of price information for the same zipcode and timestamp.

Note: Even though the customer billing Address and the service location address overlap with the same details we can uniquely identify them via address_id.

Storage Efficiency: In the process of normalization, we observed that there were rare chances of expecting data redundancy at the maximum sample data. DeviceData Table might expect more records as the devices were smart and continuously send data every **10 minutes** in addition to if some events occurred**.**

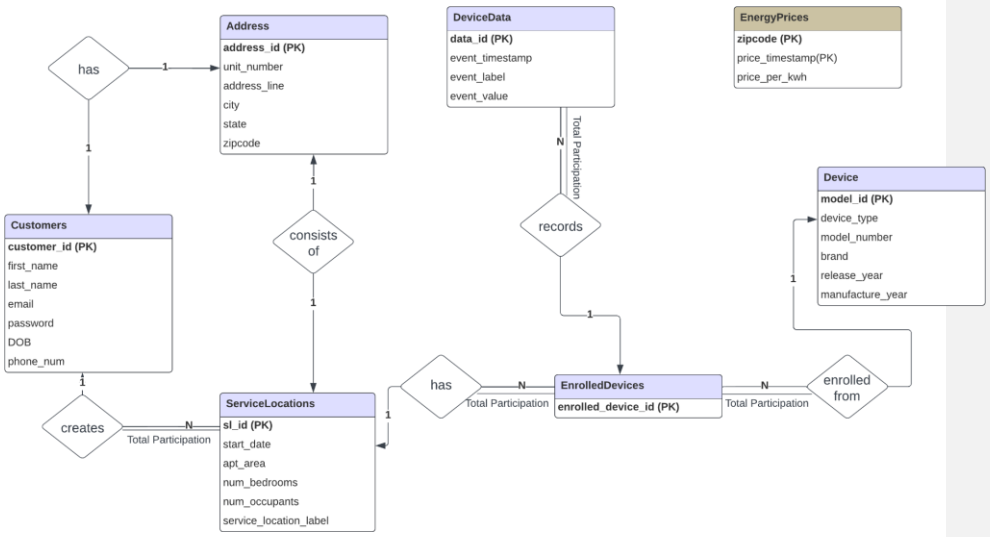
Note: We followed the process to normalize our schema to the highest form (BCNF) by splitting the tables and avoiding redundant data.

Assumptions:

- Of the above schema how the data is being sent/received to SHEMS by smart devices was not a concern at this level.
- The time interval that every electronic smart device such as AC, Refrigerator, Dryer and light will record the data(energy consumed) to SHEMS every **10 minutes** with the **'energy use'** as the event_label continuously till switched off.
- We are considering the energy consumption of every device if the event_label is 'energy use' and the value associated with the record is the amount of energy used in KWH.
- We consider other records as **events** such (switched off, switched on, temp lowered, temp increased, door open, and door closed). For these records the event_value is NULL/0. It means that we are storing this as an event that occurs in b/w the 10-minute interval.
- If the energy consumption is high due to any event that occurs, particularly for the refrigerator when the door opens and closes. We assume the smart devices are capable of calculating the additional energy used until the effect of the particular event is over.
- If the door opens in the middle of 12:10 am - 12:20 am @ 12:18 am, then the door closes at @12:22 am. The data will trigger as **12: 18 am - 'door opened' - NULL → 12: 20 am - 'energy use' - 0.67 KW → 12:22 am - 'door closed' - NULL → 12: 30 am - 'energy use' - 0.89KWH** and follows... Likewise applicable to AC systems as well when the temp lowers or temp increases.
- We assume that the data would be preloaded for each zipcode for every hour of the day along with the price information. The table would be used for reference for calculating the energy prices. The energy prices could be loaded from external sources based on the demand and supply of the energy.
- We assume that there may be a failure detection system in all smart devices considering the minimum amount of power required to send the data when the device is switched off, recording it as an event in the DeviceData table.

- We assume that every customer would have one billing address associated with him/her which can be used as a service location address.
- We make an assumption that the database system would be capable enough to handle the traffic of storing the sensor data into the respective table without DB system failures.
- During the implementation of the web-based application, we assumed that deletion of service location would be completely alright as it is being done part of the project to demonstrate the **CRUD** operations

Updated ER Diagram:



Our **ER diagram** essentially contains 6 entities among which 5 are related to each other in some way and 1 entity (EnergyPrices) is not schematically related to other entities directly.

- ***Customer*** entity and *Address* entity are connected in a 1:1 relationship as we assume that each customer would have one billing address associated with it.
- ***Customer*** entity and *ServiceLocations* entity have a 1:N cardinality relationship as one customer can create and manage multiple service locations and each service location would be associated with only one *customer*. Moreover, all service location records would have a customer entity associated with it, hence providing total participation.
- ***ServiceLocations*** entity and ***Address*** entity have a 1:1 cardinality relationship as every service location would have one unique address associated with it from the address table.

- ***ServiceLocations*** entity connects with the ***EnrolledDevices*** entity and has 1:N cardinality relationship for each service location, hence we can define multiple devices.
- ***EnrolledDevices*** entity gets all the devices from a pre-stored ***Device*** entity which has 1:N cardinal relationship because we can enroll multiple devices from the Device table.
- The ***EnrolledDevices*** entity also connects with the ***DeviceData*** entity which has a 1:N cardinality relationship as the device data table receives multiple records associated with the same enrolled device.
- The ***EnergyPrices*** entity contains hourly energy prices based on zipcode. As the energy prices are based on zipcode and each zipcode can constitute multiple addresses it cannot directly be related to any other entity available in the ER diagram space, thus essentially serving as a reference table consisting of static data about the energy prices. Hence, we've not linked the EnergyPrices entity with any other entity.

Web Based User Interfaces :

We use Django at the backend and PostgreSQL database for our web based application. We use HTML, Bootstrap CSS, and JavaScript at the front end to present the data at the client side. Apart from the above, we used certain packages like *plotly, crypt* and other such libraries to plot charts and hash the passwords. We connect to the database using Django plugins and write raw sql queries to fetch the data from the tables.

1. Authorization Module

We use hashing to store the password in the database. The hashing is done by PostgreSQL using the **crypt** library function which creates the hash by also considering a random salt generated by '**bf' (Blowfish algorithm).**

While logging in, the database simply hashes the new input password provided by the user and then compares it with the hash value in the database. If they match, then the user is allowed to login and perform his/her activities.

2. Basic Features (CRUD Operations)

The CRUD operations stand for **Create, Read, Update and Delete**. These are basic operations which are heavily repeated in the modern database world. As part of this project, we have extensively practiced and implemented features which adhered to the above operations.

CREATE:

We create new Service Locations, Enrolled Devices and Customers as part of the Create operation. The user provides the necessary details through forms and the data is stored in the database.

READ:

We display the Service Location, Devices and other records as part of the read operation. In addition, we also display charts visualizing the information from the databases.

UPDATE:

As part of the Update operation, we implemented features like Edit Profile, Edit Service Locations and Changing the Passwords. These features only update the records for which new values are provided.

DELETE:

As part of the Delete operation, we implemented the deletion feature for the Service Locations. Along with that, we also implemented the deletion of the session entries once the user logs out of their account.

3. Charts and Graphs

To better understand the information from the smart devices and in an efficient manner, we plotted graphs and charts in our web based interface. Our web based interface supports 5 graphs and charts for different use-cases. Here are the chart types:

1) Device Type *Vs* Energy Consumption
2) SL based Time *Vs* Energy Consumption
3) Actual Energy *Vs* Avg Energy Compare
4) Energy Consumption *Vs* Device type *Vs* SL
5) Average Energy Prices in different Zip Codes

4. SQL Injection and Cross Site Scripting Attacks

One of the most popular attacks we were aware of was the SQL Injection attack, where in an intruder through malicious code can retrieve restricted data records thus exposing sensitive information. To avoid and overcome this, we used one of the well-known techniques of **parametrization**. We parameterized all our SQL code wherever an external input was required. Moreover we also validated the variables while performing critical operations like manipulation and deletion of the data.

In this process, we also learnt about a few other attacks namely Cross Site Scripting, Cross Site Request Forgery (CSRF) and also Replay attacks.

For Cross Site Scripting, we made sure to validate the data we capture in the backend before processing and applying on the database. Apart from that, we used the POST method to transfer and communicate the data between the frontend and the backend.

For Cross Site Request Forgery (CSRF), we used Django's in-built library to generate tokens for every form in our application. This is mandatory while building the application according to

Django's standards which helped us understand about the attack and how better equipped we were to handle it.

For Replay attacks, that is when the user clicks on a button and repeats his/her stance thus complicating the state of the database at the backend. To avoid this, we avoided revealing the path of sensitive information and made only some URL paths public and all others were called internally and redirected to one of the public URL paths thus not allowing fabricated multiple calls. Another approach was also thought of to create tokens to track the execution of critical components.

5. Transactions and Concurrency

In our project implementation, we got a chance to apply the concept of Transactions to solve a problem of insertion of records into the Customers and Address Tables. As both the tables were linked by Foreign Key, the records should first be written to the Address table and then the resulting address_id entry is then inserted into the Customers table. As this whole process constitutes one unit of operation i.e., the result should either return failed or success based on execution of all the statements. Hence using transactions here solves the problem and helps us to execute the queries with ease.

For concurrency, we use sessions to track the username and other details. The sessions in our case were being stored in the PostgreSQL database which had an expiring date after which they would be invalid and deleted by the python worker. Moreover, the sessions also help to store certain information about the info of some tables instead of retrieving them everytime from the database. They are also much safer and more efficient than HTTP Cookies.

```
try:
with transaction.atomic(using='default'):
address_insert = "INSERT INTO ADDRESS (unit_number, address_line, city,
state, zipcode) values(%s, %s, %s, %s, %s) RETURNING address_id"
cursor.execute(address_insert, [unit_number, address_line, city, state,
zipcode])

billing_id = cursor.fetchone()

customer_insert = """INSERT INTO CUSTOMERS (email, password, phone_num,
dob, first_name, last_name, billing_address_id)
values(%s, crypt(%s, gen_salt('bf')), %s, %s, %s, %s, %s)"""
cursor.execute(customer_insert, [username, password, phone_num, dob, fname,
lname, billing_id])

rows_affected = cursor.rowcount

# Abort the transaction
except Exception as e:
messages.error(request, "Error occurred while saving. Please try again
```

```
after sometime")
```

6. Few Query Examples

Device Type *Vs* Energy Consumption

```
SELECT
D.device_type,
SUM(DD.event_value) AS total_energy
FROM
DeviceData DD
JOIN EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
JOIN Device D ON ED.model_id = D.model_id
JOIN ServiceLocations SL ON ED.sl_id = SL.sl_id
WHERE
SL.customer_id = %s
AND DD.event_label = 'energy use'
AND EXTRACT(YEAR FROM DD.event_timestamp) = %s
GROUP BY
D.device_type
```

SL based Time *Vs* Energy Consumption

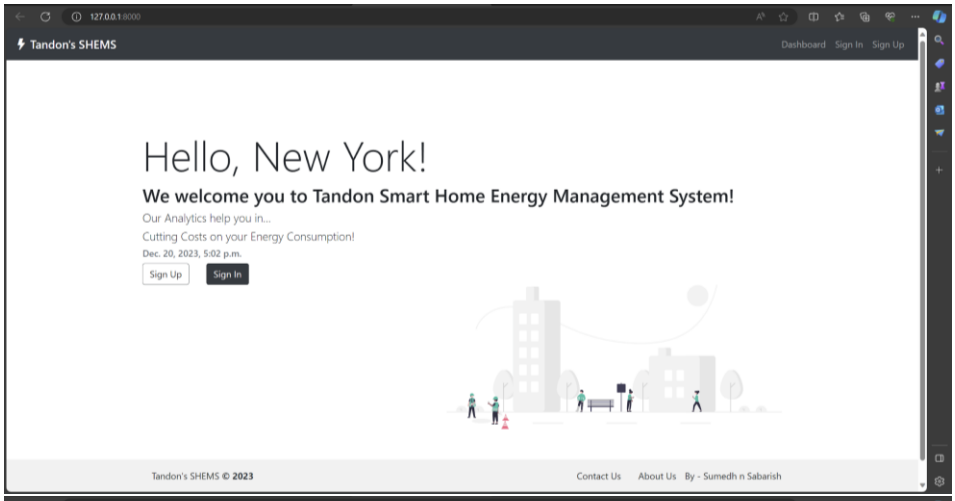
```
SELECT
extract('hour' from DD.event_timestamp) AS hour_of_event,
SUM(DD.event_value) AS total_energy
FROM
DeviceData DD
JOIN EnrolledDevices ED ON DD.enrolled_device_id = ED.enrolled_device_id
JOIN ServiceLocations SL ON ED.sl_id = SL.sl_id
WHERE
SL.customer_id = %s
AND DD.event_label = 'energy use'
AND SL.sl_id = %s
AND DD.event_timestamp >= %s and DD.event_timestamp <= %s
GROUP BY
hour_of_event
```

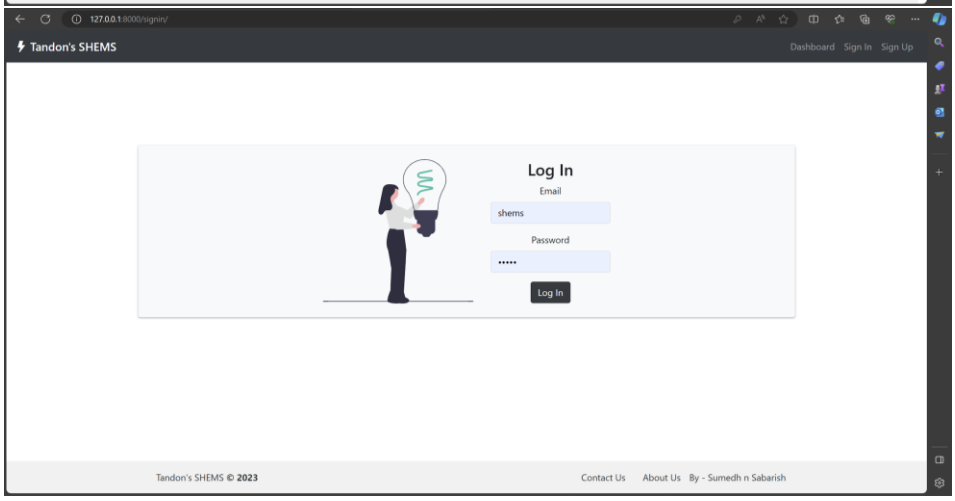
```
ORDER BY
hour_of_event;
```

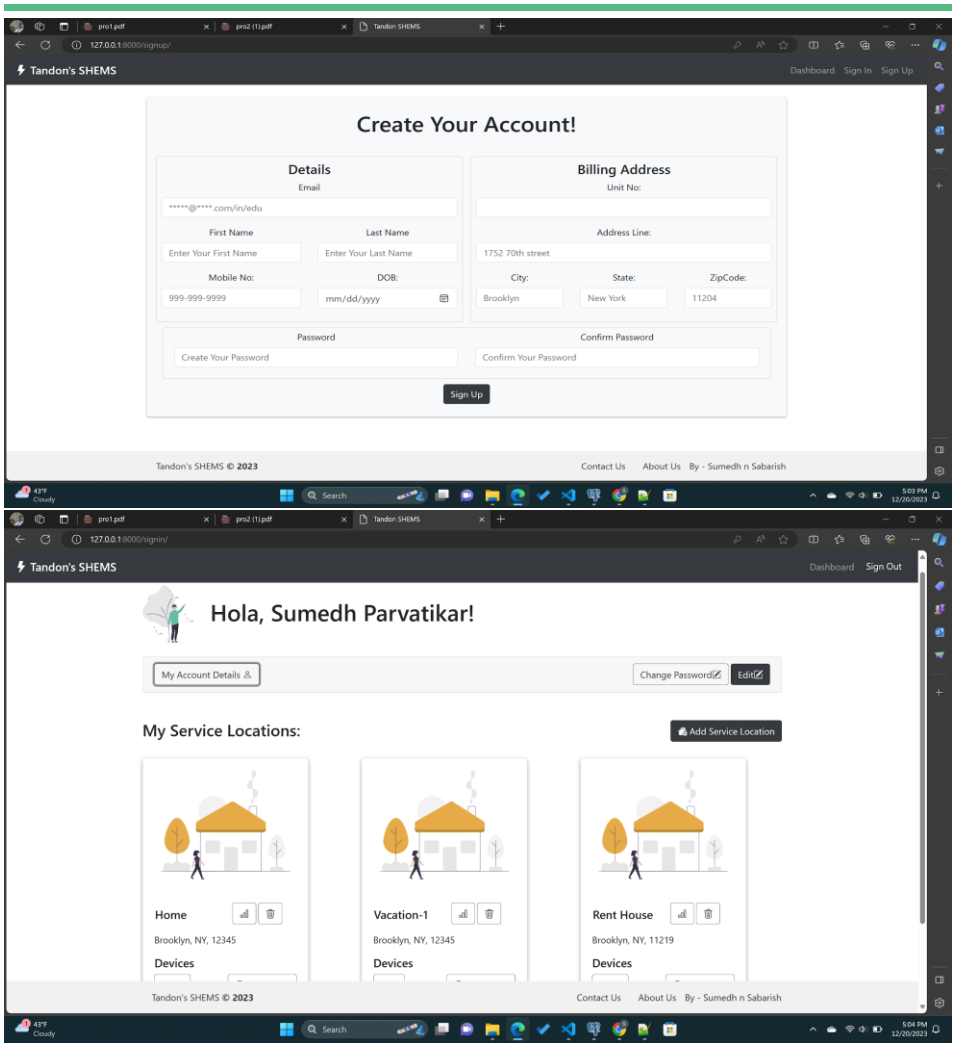
Actual Energy *Vs* Avg Energy Compare

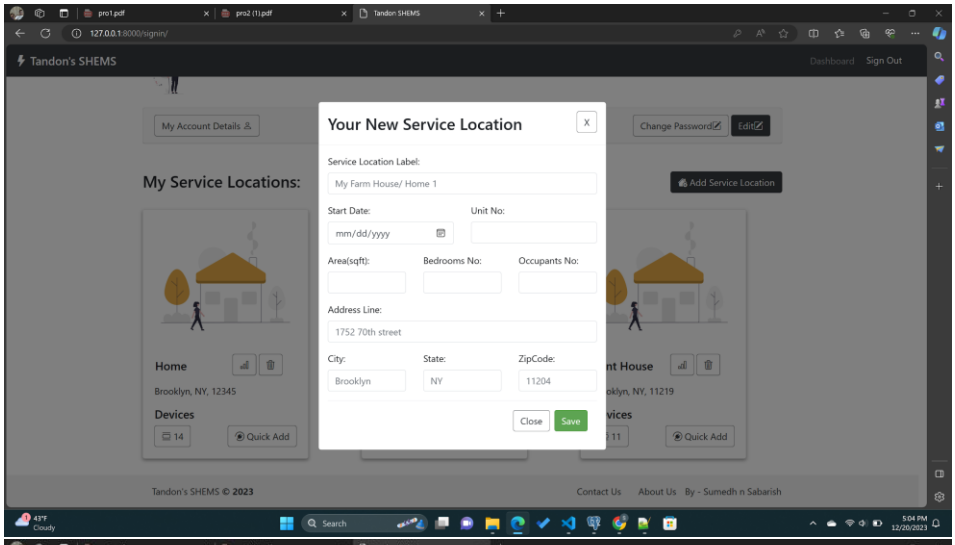
```
Select zipcode, avg(price_per_kwh) from Energy_Prices group by zipcode;
```

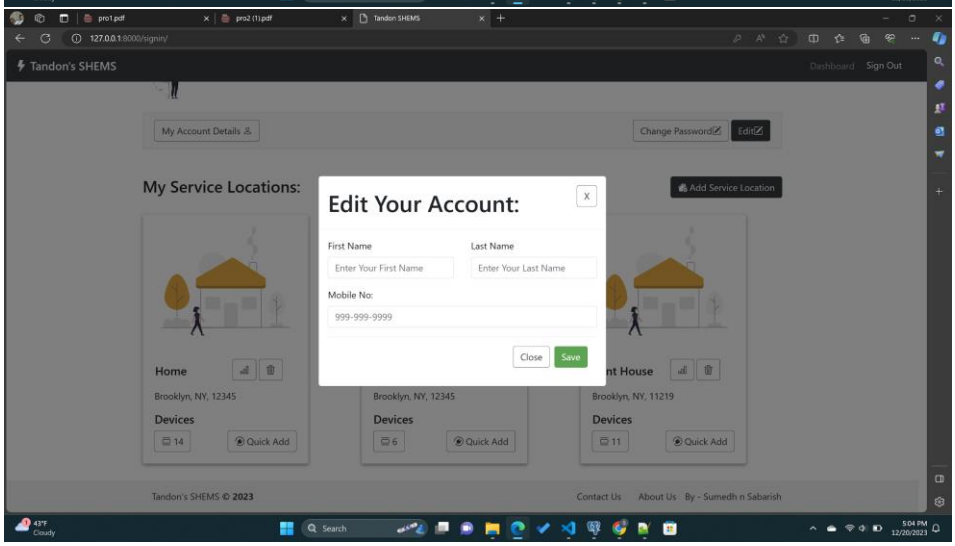
7. **Website Snapshots:**

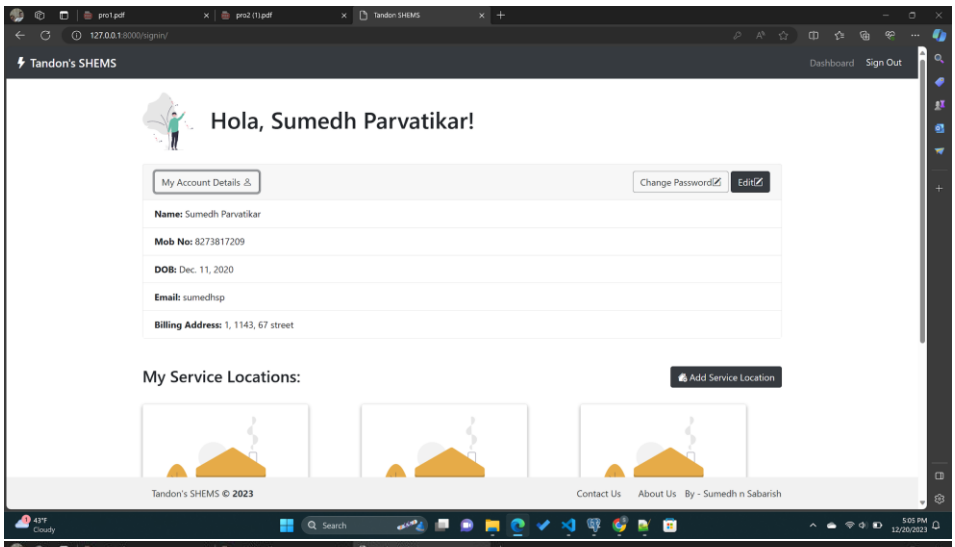


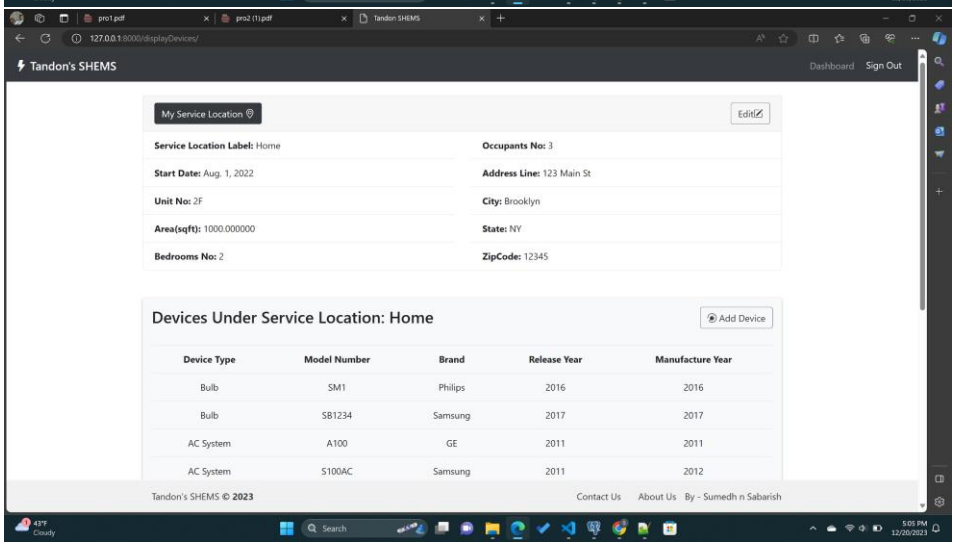


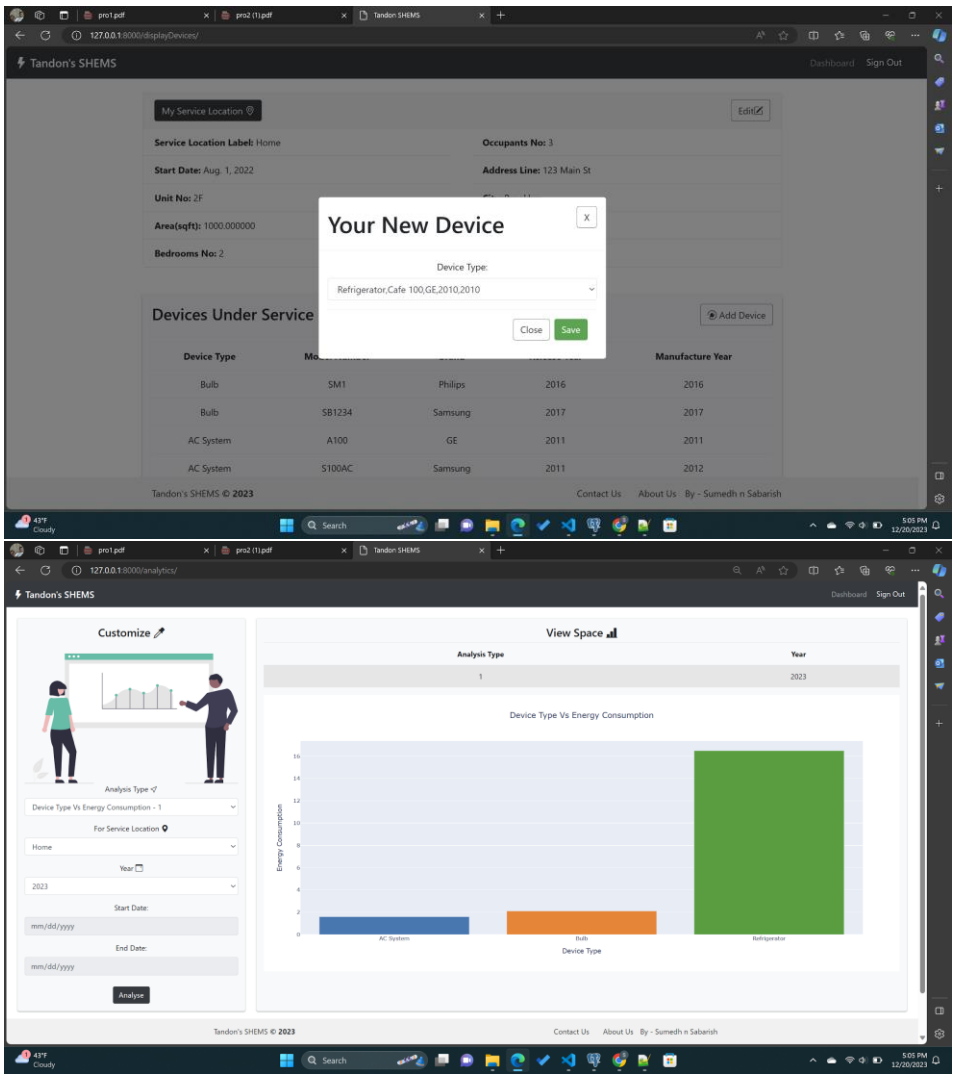

Tandon's SHEMS
Dashboard Sign In Sign Up
Create Your Account!
Details
Email
First Name
Last Name
Mobile No:
DOB:
Billing Address
Unit No:
Address Line:
City:
State:
ZipCode:
Password
Confirm Password
Sign Up
Tandon's SHEMS © 2023
Contact Us About Us By - Sumedh n Sabarish
Tandon's SHEMS
Dashboard Sign Out
Hola, Sumedh Parvatikar!
My Account Details
Change Password
Edit
My Service Locations:
Add Service Location
Home
Brooklyn, NY, 12345
Devices
Vacation-1
Brooklyn, NY, 12345
Devices
Rent House
Brooklyn, NY, 11219
Devices
Tandon's SHEMS © 2023
Contact Us About Us By - Sumedh n Sabarish


Tandon's SHEMS
Dashboard
Sign Out
My Account Details
Change Password
Edit
My Service Locations:
Add Service Location
Your New Service Location
Service Location Label:
My Farm House/ Home 1
Start Date:
mm/dd/yyyy
Unit No:
Area(sqft):
Bedrooms No:
Occupants No:
Address Line:
1752 70th street
City:
Brooklyn
State:
NY
ZipCode:
11204
Close
Save
Home
Brooklyn, NY, 12345
Devices
14
Quick Add
Brooklyn, NY, 11219
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish

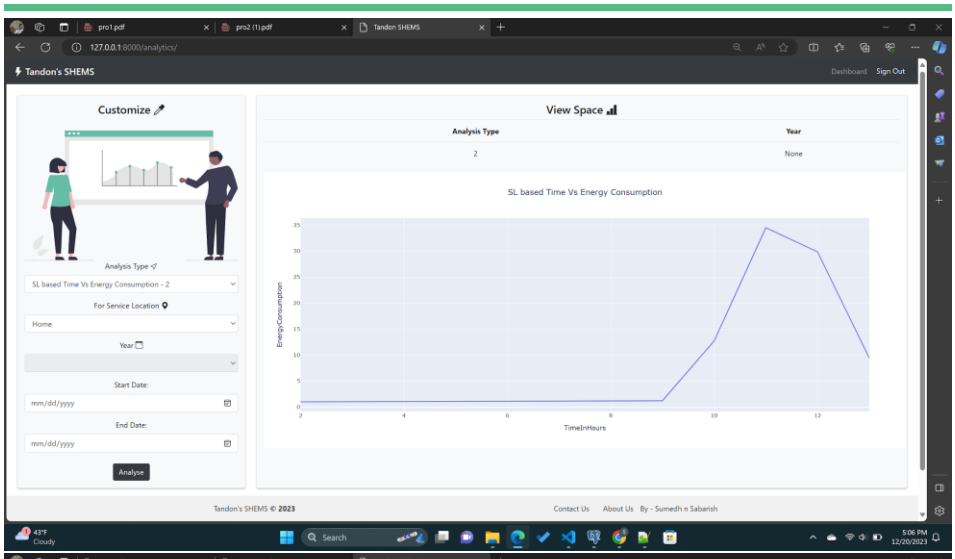

Tandon's SHEMS
Dashboard
Sign Out
My Account Details
Change Password
Edit
My Service Locations:
Add Service Location
Edit Your Account:
First Name
Enter Your First Name
Last Name
Enter Your Last Name
Mobile No:
999-999-9999
Close
Save
Home
Brooklyn, NY, 12345
Devices
14
Quick Add
6
11
Brooklyn, NY, 11219
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish

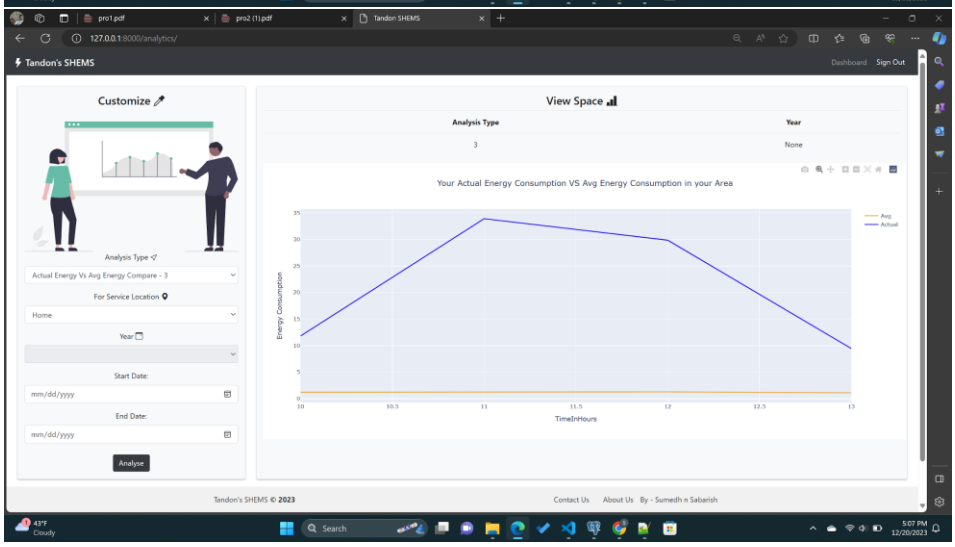

Tandon's SHEMS
Dashboard
Sign Out
Hola, Sumedh Parvatikar!
My Account Details
Change Password
Edit
Name: Sumedh Parvatikar
Mob No: 8273817209
DOB: Dec. 11, 2020
Email: sumedhsp
Billing Address: 1, 1143, 67 street
My Service Locations:
Add Service Location
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish

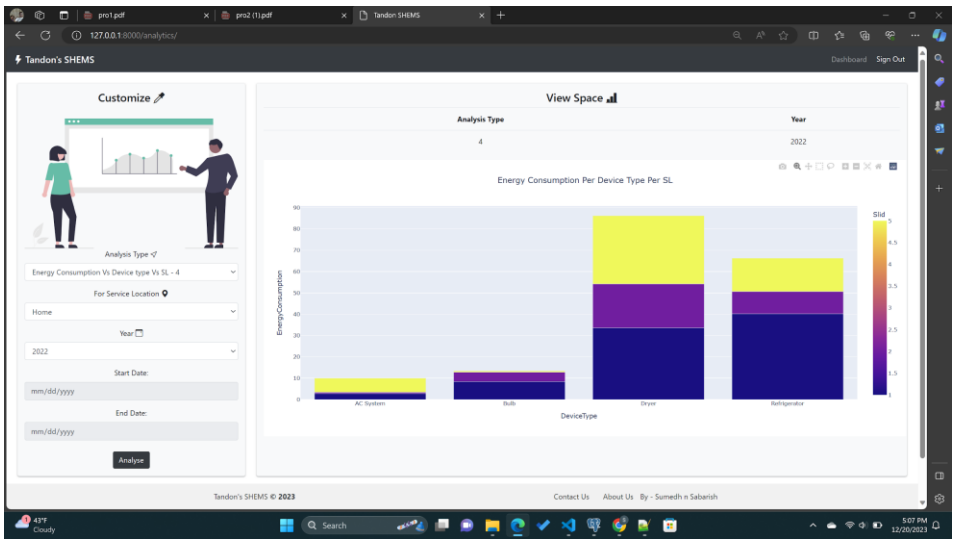

Tandon's SHEMS
Dashboard
Sign Out
My Service Location
Edit
Service Location Label: Home
Occupants No: 3
Start Date: Aug. 1, 2022
Address Line: 123 Main St
Unit No: 2F
City: Brooklyn
Area(sqft): 1000.000000
State: NY
Bedrooms No: 2
ZipCode: 12345
Devices Under Service Location: Home
Add Device
Device Type
Model Number
Brand
Release Year
Manufacture Year
Bulb
SM1
Philips
2016
2016
Bulb
SB1234
Samsung
2017
2017
AC System
A100
GE
2011
2011
AC System
S100AC
Samsung
2011
2012
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish

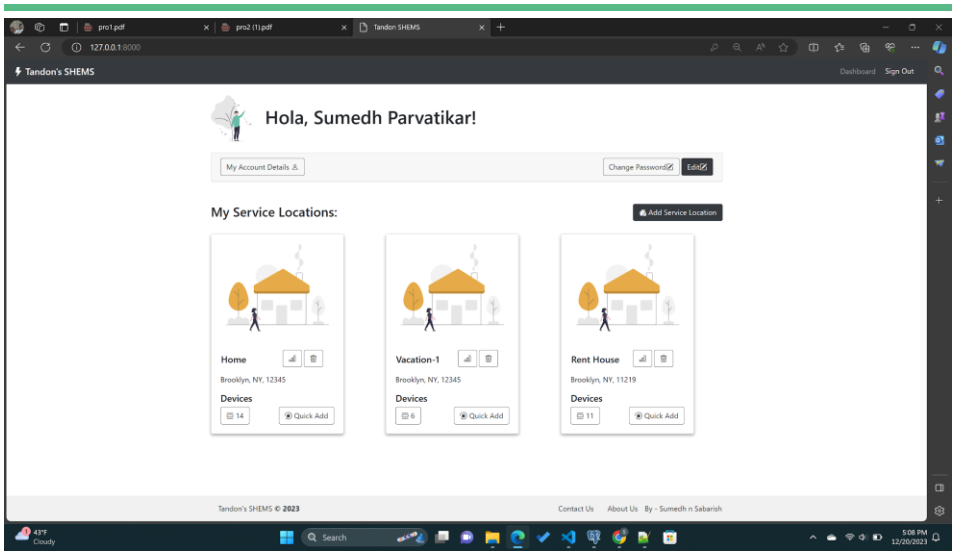

Tandon's SHEMS
Dashboard
Sign Out
My Service Location
Edit
Service Location Label: Home
Occupants No: 3
Start Date: Aug. 1, 2022
Address Line: 123 Main St
Unit No: 2F
Area(sqft): 1000.000000
Bedrooms No: 2
Your New Device
Device Type:
Refrigerator,Cafe 100,GE,2010,2010
Close
Save
Devices Under Service
Add Device
Device Type
Manufacture Year
Bulb
SM1
Philips
2016
2016
Bulb
SB1234
Samsung
2017
2017
AC System
A100
GE
2011
2011
AC System
S100AC
Samsung
2011
2012
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish

Tandon's SHEMS
Dashboard
Sign Out
Customize
Analysis Type
Device Type Vs Energy Consumption - 1
For Service Location
Home
Year
2023
Start Date:
mm/dd/yyyy
End Date:
mm/dd/yyyy
Analyse
View Space
Analysis Type
Year
1
2023
Device Type Vs Energy Consumption
Energy Consumption
Device Type
AC System
bulb
Refrigerator
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish


Tandon's SHEMS
Dashboard
Sign Out
Customize
Analysis Type
SL based Time Vs Energy Consumption - 2
For Service Location
Home
Year
Start Date:
mm/dd/yyyy
End Date:
mm/dd/yyyy
Analyse
View Space
Analysis Type
Year
2
None
SL based Time Vs Energy Consumption
Energy Consumption
TimeInHours
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish


Tandon's SHEMS
Dashboard
Sign Out
Customize
Analysis Type
Actual Energy Vs Avg Energy Compare - 3
For Service Location
Home
Year
Start Date:
mm/dd/yyyy
End Date:
mm/dd/yyyy
Analyse
View Space
Analysis Type
Year
3
None
Your Actual Energy Consumption VS Avg Energy Consumption in your Area
Avg
Actual
Energy Consumption
TimeInHours
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish

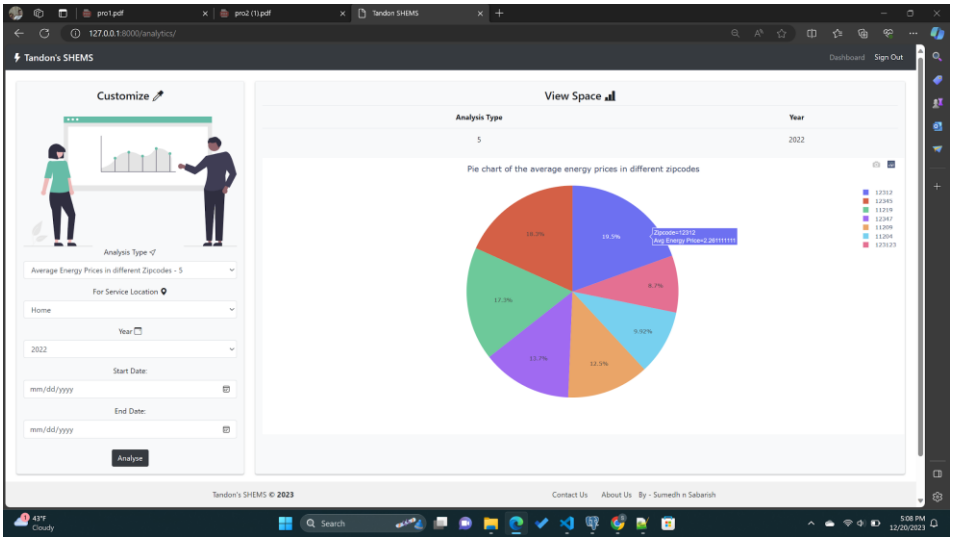

Tandon's SHEMS
Dashboard
Sign Out
Customize
Analysis Type
Energy Consumption Vs Device type Vs SL - 4
For Service Location
Home
Year
2022
Start Date:
mm/dd/yyyy
End Date:
mm/dd/yyyy
Analyse
View Space
Analysis Type
Year
4
2022
Energy Consumption Per Device Type Per SL
Energy Consumption
DeviceType
AC System
Bulb
Dryer
Refrigerator
Sld
Tandon's SHEMS © 2023
Contact Us
About Us
By - Sumedh n Sabarish



Additional Features

1) We implemented an extra graph to visualize the average price per kwh based on different zip codes using a pie chart.

2) We implemented a simple aesthetic look and feel of the website to improve the user experience.



3) We implemented password encryption to improve the security of the data. Moreover we erased all the static path references in different forms and replaced them with django template URI path indicators to avoid delay attacks as well as exposing sensitive paths to the outside world.

INDEX:

```
CREATE TABLE Address(
        address_id SERIAL PRIMARY KEY,
        unit_number varchar(50) NOT NULL,
        address_line varchar(100) NOT NULL,
        city varchar(100) NOT NULL,
        state varchar(2) NOT NULL,
        zipcode int NOT NULL
);


CREATE TABLE Customers(
        customer_id SERIAL PRIMARY KEY,
        email varchar(50) NOT NULL,
        password varchar(100) NOT NULL,
        phone_num varchar(10) NOT NULL,
        dob date NOT NULL,
        first_name  (100) NOT NULL,
        last_name varchar(100) NOT NULL,
        billing_address_id INT NOT NULL,
        FOREIGN KEY(billing_address_id) REFERENCES Address(address_id)
);


CREATE TABLE ServiceLocations(
        sl_id SERIAL PRIMARY KEY,
        customer_id INT,
        address_id INT UNIQUE,
        service_location_label varchar(100) NOT NULL,
        start_date date,
        apt_area decimal(18,6),
        num_bedrooms INT,
        num_occupants INT,
```

```
        FOREIGN KEY(customer_id) REFERENCES Customers(customer_id),
        FOREIGN KEY(address_id) REFERENCES Address(address_id)
);


CREATE TABLE Device(
        model_id serial PRIMARY KEY,
        device_type varchar(100) NOT NULL,
        model_number varchar(100) NOT NULL,
        brand varchar(100),
        release_year INT,
        manufacture_year INT
);




CREATE TABLE EnrolledDevices(
        enrolled_device_id SERIAL PRIMARY KEY,
        sl_id INT NOT NULL,
        model_id varchar(64) NOT NULL,
        FOREIGN KEY(sl_id) REFERENCES ServiceLocations(sl_id),
        FOREIGN KEY(model_id) REFERENCES Device(model_id)
);


CREATE TABLE DeviceData(
        data_id SERIAL PRIMARY KEY,
        enrolled_device_id INT NOT NULL,
        event_timestamp timestamp,
        event_label varchar(100) NOT NULL,
        event_value decimal(18,6),
        FOREIGN KEY(enrolled_device_id) REFERENCES EnrolledDevices(enrolled_device_id)
);


CREATE TABLE EnergyPrices(
        zipcode INT NOT NULL,
        price_timestamp timestamp NOT NULL,
        price_per_kwh decimal(18,6) NOT NULL,
        PRIMARY KEY(zipcode, price_timestamp)
);
```

Sample Data:

Address Table

```
Insert into Address(unit_number, address_line, city, state, zipcode)
values('2F', '123 Main St', 'Brooklyn', 'NY', 12345),
         ('Apt 20', '230 Nevins St', 'Brooklyn', 'NY', 12347),
         ('1F', '1120 67th Street', 'Brooklyn', 'NY', 12358),
         ('1F', '12340 86th Street', 'Brooklyn', 'NY', 12358),
         ('3F', '1250 Grove St', 'Brooklyn', 'NY', 12345)
        ('2F', '123 Mcarther St', 'Manhattan', 'NY',11309),
         ('Apt 20', '230 Nevins St', 'StatenIsland', 'NY', 11809),
         ('1F', '1120 67th Street', 'Queens', 'NY', 12367),
         ('1F', '12340 86th Street', 'Bronx', 'NY', 12987),
         ('3F', '1250 10th St', 'Manhattan', 'NY', 12389)
```

Customer Table

```
Insert into Customers(first_name, last_name, billing_address_id)
values('John','Doe', 1), ('Jane','Smith', 3), ('Rakesh','R', 2),
('Jack','Ryan', 2), ('Rebba','John', 4),('Samantha','Ruth Prabhu', 5),
('Tom','Cruise', 4), ('Klin','John', 7),('Nayantara','S', 8)
```

ServiceLocations Table

```
Insert into
ServiceLocations(customer_id,address_id,start_date,apt_area,num_bedrooms,num_occupants)
values(1, 1, '2022-08-01',1000, 2, 3),
         (1, 2, '2022-08-10',1050, 2, 1),
         (2, 3, '2022-07-30',1200, 3, 5),
         (2, 4, '2022-08-15',980, 1, 2),
         (2, 5, '2022-08-16',1160, 2, 2),
         (4, 6, '2022-09-30',1200, 5, 2),
         (5, 7, '2022-10-15',980, 3, 2),
         (6, 8, '2022-11-16',1160, 4, 5)
```

Device

```
Insert into Device(model_id,device_type,model_number,brand,release_year,manufacture_year)
values('model1','Refrigerator','Cafe 100','GE',2010, 2010),
         ('model2','AC System','A100','GE',2011, 2011),
         ('model3','Refrigerator','S100QZ','Samsung',2011, 2011),
         ('model4','Refrigerator','L100QA','LG',2011, 2011),
         ('model5','AC System','S100AC','Samsung',2011, 2012),
         ('model6','Dryer','D101','GE',2015, 2015),
         ('model7','Bulb','SM1','Philips',2016, 2016),
         ('model8','Bulb','SB1234','Samsung',2017, 2017),
         ('model9','Bulb','A101','Apple',2018, 2018),
         ('model10','Dryer','D1938L','LG',2016, 2016)
```

EnrolledDevices

```
Insert into EnrolledDevices(sl_id, model_id)
values(1, 'model7'), - Bulb 1
         (1, 'model8'), - Bulb 2
```

```
          (2, 'model2'), - AC 3
          (3, 'model5'), - AC 4
          (3, 'model3'), - Refrigerator 5
          (4, 'model4') - Refrigerator 6
```

DeviceData

```
Insert into DeviceData(enrolled_device_id,event_timestamp,event_label,event_value)
values (1,'2023-12-01 10:30:00', 'energy use', 0.9),
        (2,'2023-12-01 09:30:00', 'energy use', 0.4),
        (3,'2023-12-01 11:30:00', 'energy use', 0.2),
        (4,'2023-12-01 02:30:00', 'energy use', 0.33),
         (2,'2023-12-01 09:40:00', 'energy use', 0.4),
        (3,'2023-12-01 11:40:00', 'energy use', 0.2),
        (4,'2023-12-01 02:40:00', 'energy use', 0.33),
         (2,'2023-12-01 09:50:00', 'energy use', 0.4),
        (3,'2023-12-01 11:50:00', 'energy use', 0.2),
        (4,'2023-12-01 02:50:00', 'energy use', 0.33),
         (1,'2022-08-01 10:30:00', 'switched on', NULL),
         (1,'2022-08-01 10:40:00', 'energy use', 0.8),
         (1,'2022-08-01 10:50:00', 'energy use', 0.8),
         (1,'2022-08-01 11:00:00', 'energy use', 0.8),
        (1,'2022-08-01 11:00:00', 'switched off', NULL),
         (3,'2022-08-01 11:28:00', 'switched on', NULL),
         (3,'2022-08-01 11:29:00', 'temp lowered', 23.6),
         (3,'2022-08-01 11:30:00', 'energy use', 0.6),
         (6,'2022-08-01 11:33:00', 'switched on', NULL),
        (6,'2022-08-01 11:40:00', 'energy use', 0.9),
         (6,'2022-08-01 11:50:00', 'energy use', 0.9),
        (3,'2022-08-01 11:50:00', 'energy use', 0.2),
         (3,'2022-08-01 11:50:00', 'switched off', NULL),
         (6,'2022-08-01 12:00:00', 'energy use', 0.9),
         (6,'2022-08-01 12:10:00', 'energy use', 0.9),
         (6,'2022-08-01 12:15:00', 'door open', NULL),
         (6,'2022-08-01 12:20:00', 'energy use', 0.99),
         (6,'2022-08-01 12:30:00', 'energy use', 0.99),
         (6,'2022-08-01 12:40:00', 'energy use', 1.02),
         (6,'2022-08-01 12:50:00', 'energy use', 1.05),
         (6,'2022-08-01 12:50:00', 'door closed', NULL),
         (6,'2022-08-01 12:50:00', 'energy use', 1.0),
         (6,'2022-08-01 13:00:00', 'energy use', 0.9),
         (6,'2022-08-01 13:10:00', 'energy use', 0.9),
         (6,'2022-08-01 13:20:00', 'energy use', 0.9),
         (6,'2022-08-01 13:30:00', 'energy use', 0.9),
         (6,'2022-08-01 13:33:00', 'door open', NULL),
         (6,'2022-08-01 13:38:00', 'door closed', NULL),
         (6,'2022-08-01 13:38:00', 'switched off', NULL),
         (5,'2022-08-01 10:30:00', 'switched on', NULL),
         (5,'2022-08-01 10:40:00', 'energy use', 0.7),
         (5,'2022-08-01 10:50:00', 'energy use', 0.7),
         (5,'2022-08-01 11:00:00', 'energy use', 0.7),
         (5,'2022-08-01 11:10:00', 'energy use', 0.7),
         (5,'2022-08-01 11:12:00', 'door open', NULL),
         (5,'2022-08-01 11:20:00', 'energy use', 0.9),
         (5,'2022-08-01 11:30:00', 'energy use', 0.9),
```

```
(5,'2022-08-01 11:40:00', 'energy use', 0.9),
(5,'2022-08-01 11:43:00', 'door closed', NULL),
(1,'2022-09-01 10:30:00', 'switched on', NULL),
(1,'2022-09-01 10:40:00', 'energy use', 1.2),
(1,'2022-09-01 10:50:00', 'energy use', 1.2),
(1,'2022-09-01 11:00:00', 'energy use', 1.2),
(1,'2022-09-01 11:00:00', 'switched off', NULL),
(3,'2022-09-01 11:28:00', 'switched on', NULL),
(3,'2022-09-01 11:29:00', 'temp increased', 25.6),
(3,'2022-09-01 11:30:00', 'energy use', 1.0),
(6,'2022-09-01 11:33:00', 'switched on', NULL),
(6,'2022-09-01 11:40:00', 'energy use', 1.4),
(6,'2022-09-01 11:50:00', 'energy use', 1.4),
(3,'2022-09-01 11:50:00', 'energy use', 1.0),
(3,'2022-09-01 11:50:00', 'switched off', NULL),
(6,'2022-09-01 12:00:00', 'energy use', 1.4),
(6,'2022-09-01 12:10:00', 'energy use', 1.3),
(6,'2022-09-01 12:15:00', 'door open', NULL),
(6,'2022-09-01 12:20:00', 'energy use', 1.4),
(6,'2022-09-01 12:30:00', 'energy use', 1.4),
(6,'2022-09-01 12:40:00', 'energy use', 1.4),
(6,'2022-09-01 12:50:00', 'energy use', 1.4),
(6,'2022-09-01 12:50:00', 'door closed', NULL),
(6,'2022-09-01 12:50:00', 'energy use', 1.4),
(6,'2022-09-01 13:00:00', 'energy use', 1.45),
(6,'2022-09-01 13:10:00', 'energy use', 1.45),
(6,'2022-09-01 13:20:00', 'energy use', 1.46),
(6,'2022-09-01 13:30:00', 'energy use', 1.48),
(6,'2022-09-01 13:33:00', 'door open', NULL),
(6,'2022-09-01 13:38:00', 'door closed', NULL),
(6,'2022-09-01 13:38:00', 'switched off', NULL),
(5,'2022-09-01 10:30:00', 'switched on', NULL),
(5,'2022-09-01 10:40:00', 'energy use', 0.5),
(5,'2022-09-01 10:50:00', 'energy use', 0.5),
(5,'2022-09-01 11:00:00', 'energy use', 0.5),
(5,'2022-09-01 11:10:00', 'energy use', 0.5),
(5,'2022-09-01 11:12:00', 'door open', NULL),
(5,'2022-09-01 11:20:00', 'energy use', 0.7),
(5,'2022-09-01 11:30:00', 'energy use', 0.7),
(5,'2022-09-01 11:40:00', 'energy use', 0.7),
(5,'2022-09-01 11:43:00', 'door closed', NULL),
(2,'2022-09-01 11:42:00', 'switched on', NULL),
(2,'2022-09-01 11:50:00', 'energy use', 0.6),
(2,'2022-09-01 12:00:00', 'energy use', 0.6),
(2,'2022-09-01 12:10:00', 'energy use', 0.6),
(2,'2022-09-01 12:20:00', 'energy use', 0.6),
(2,'2022-09-01 12:20:00', 'switched off', 0.6);
```

Energy Prices:

```sql
INSERT INTO EnergyPrices (zipcode, price_timestamp, price_per_kwh)
VALUES
(12345, '2022-08-01 12:00:00', 1.15),
(12347, '2022-08-01 12:00:00', 1.12),
(12358, '2022-08-01 12:00:00', 1.14),
(12345, '2022-08-01 13:00:00', 0.9),
(12347, '2022-08-01 13:00:00', 0.7),
(12358, '2022-08-01 13:00:00', 1.08),
(12345, '2022-08-01 09:00:00', 1.6),
(12347, '2022-08-01 09:00:00', 1.11),
(12358, '2022-08-01 09:00:00', 1.09),
(12345, '2022-08-01 11:00:00', 1.34),
(12347, '2022-08-01 11:00:00', 1.22),
(12358, '2022-08-01 11:00:00', 1.13),
(12345, '2022-08-01 10:00:00', 1.24),
(12347, '2022-08-01 10:00:00', 1.12),
(12358, '2022-08-01 10:00:00', 1.00);
```