Test Strategy Document

Project: KRB AC monitor and control
Team Number: Team 5

Scope

1. The document will be reviewed by both clients and TA .
2. The client and TA must approve the document as this document will signify that the system has been tested and has been verified.
3. We began most of our testing in sprint 3 and sprint 4 as we were supposed to learn some other language we weren't acquainted with which took most of the team's time.

Test Approach

1. The testing mostly involved manual checking. Basically, run a python script and check whether the desired output has been obtained. For example, a script can be created to check the state of the AC. Hence, we can switch the AC off through the app, and verify that the AC is actually off by getting the state of the AC through the script.
2. Testing on software level to check if the server is working or not, and api are being properly served.
3. Testing on hardware level to check if AC is responding to the given input through Bacnet.

Each team member tested the code they have written. The team worked together to integrate the entire application which was built. The application was tested at the end of every code sprint, before pushing the code to the Gitlab repository.

Most of the testing was done manually.

Test Environment

There was no testing environment as such, the “testing environment” was the development environment itself. Some of the development environments were:

- Most of the code was written in javascript (nodejs) or python so python compiler and MERN stack are required for testing.
- Data was stored in MongoDB so that served as the backup.
- Pybacnet library and Hobolink API were used to obtain data.

Testing Tools

- console.log was used to check most of the script outputs.
- APIs were tested for working and correctness using Postman.
- Requests were made to check whether the database was working correctly.
- Input was given manually and checked if everything was working properly.

Use Cases

Following use cases were derived:

Authentication:

1. Login/Logout
2. Registration
3. AC access based on floor

Control:

1. Control AC using pybacnet.
2. Adjust AC temperature using pybacnet.

3. Request access from admin to control AC.

View:

1. View data usage and analytics in a graphical format using data obtained from Bacnet and Hobolink.

Email Alert System:

1. Sends an email alert to the user. Implemented using the nodemailer package in nodejs.

Test Cases

1. **Login**

Can be checked by entering a wrong input and ensuring the app does not allow unauthorized access. Correct input should also be entered, and it should be ensured that authorized access should be granted login permission.

2. **Logout**

On click home page must be rendered.

3. **Register**

Email should be valid, and passwords should be entered twice to ensure the user did not make a mistake while entering the password. It should also be ensured that entering an existing user email should not create a new account.

4. **AC control**

The python script can be run to get the status of the AC and ensure the use case through the app was successful.

5. **Request access**

A dummy user can be created as a space user and can be made to request control access from the Admin. Then, the admin should login and grant access and ensure that the space user can now control requested AC.

6. **Temp Adjust**

Same as AC control, change the AC temperature through the app and run a script to verify that the AC temperature matches the temperature set through the app.

7. **Bacnet**

Checking if a successful connection can be established with bacnet.

8. **Hobolink:**

Checking if a successful connection can be established with hobolink.

9. **View:**

Change the temperature of the AC (through the app/script), and see if the graph updates the temperature of the AC.