Project Report

SMART HOME

22CDS0439

P.HAROL MAXILAN

1. Introduction:

This project is a Smart Home Lighting System, which provides users with an easy way to manage lighting in their home through customizable presets. The application allows users to add rooms, toggle lights, and delete presets or rooms while saving configurations in a persistent JSON file for future use.

2. Design Choices:

- Data Structure: The data is structured as a dictionary, with presets as keys and rooms as values (also dictionaries with room names as keys and light statuses as values). This allows for easy addition, removal, and modification of rooms and presets.
- JSON File for Persistence: The use of a JSON file allows the system to save configurations between sessions. When the application starts, it loads the JSON file to retrieve any existing presets and room configurations.
- Modular Design: The system is broken into functions, each handling a specific task (e.g., loading data, adding rooms, toggling light status). This modular approach makes the code more readable and maintainable.

3. System Architecture:

The system follows a command-line interface (CLI) design:

1.Main Menu: Displays options to the user and captures input.

2.Functions for Managing Presets and Rooms:

- load_data() – Loads data from the JSON file.
- save_data(data) – Saves updated data back to the JSON file.
- add_room(preset_data) – Adds new rooms to a preset.
- toggle_light(presets) – Toggles the light status (ON/OFF) for rooms.
- delete_preset(presets) – Deletes an entire preset.
- delete_room_from_preset(presets) – Deletes a room from a preset.

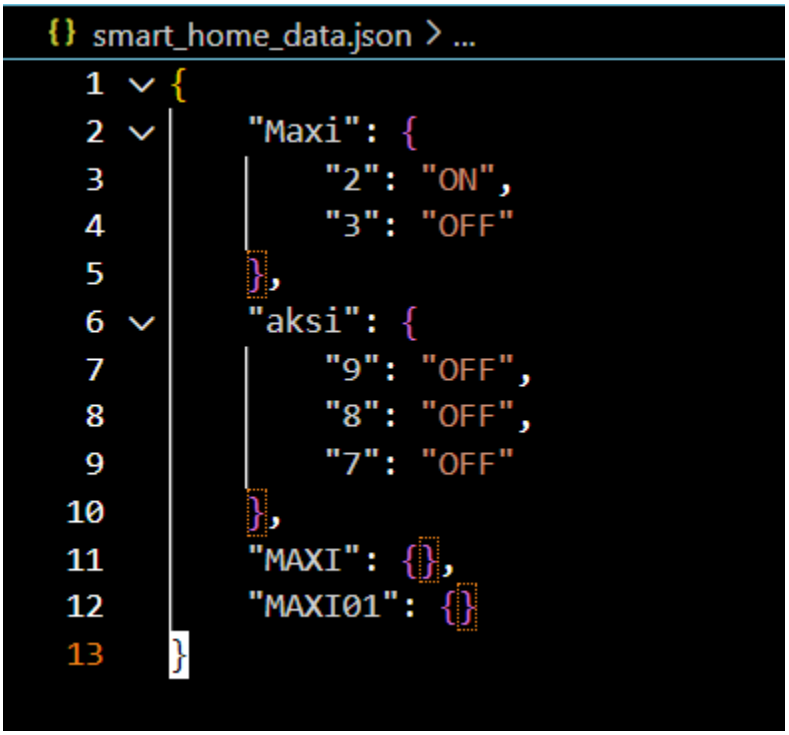
3.Error Handling: Each function checks if the provided preset or room exists before proceeding with the action, ensuring that the program doesn't crash from invalid input.

4. Data Structures:

The main data structure used in the program is a dictionary:

- Presets: Each preset is represented by a dictionary key (e.g., "Living Room Preset").
- Rooms within a preset: Each preset's value is another dictionary, where keys are room names (e.g., "Living Room", "Kitchen") and values are light statuses (e.g., "ON", "OFF").

Example data stored in smart_home_data.json:



5. Functions Overview:

load_data():

- Loads the saved configuration from smart_home_data.json (if it exists).
- Returns the data as a dictionary.

Save_data(data):

- Saves the provided dictionary data to smart_home_data.json.

add_room(preset_data):

- Prompts the user to input room names and adds those rooms to a specified preset.

toggle_light(presets):

- Toggles the light status for rooms in a selected preset.

delete_preset(presets):

- Deletes a chosen preset after confirmation.

delete_room_from_preset(presets):

- Deletes selected rooms from a chosen preset.

6. Challenges Faced and Solutions:

Handling Invalid Input:

- Issue: Users might provide invalid preset names or room names.
- Solution: Implemented error handling that checks if the specified preset or room exists before performing the action. If not, an error message is shown, and the user is prompted to provide a valid input.

Data Persistence:

- Issue: Ensuring that changes to the preset configurations are saved and loaded correctly.
- Solution: Used the json module to read and write data to a file (smart_home_data.json), ensuring persistence between sessions.

User Interface:

- Issue: Making the command-line interface simple and user-friendly.
- Solution: Kept the menu options clear and provided prompts to guide the user through each step, along with error messages when necessary.

7. Future Enhancements:

- Graphical User Interface (GUI): Transitioning from CLI to a GUI would improve usability, especially for non-technical users.
- Remote Control Integration: Integrating the system with mobile apps or smart home devices like Alexa or Google Home could enhance the control experience.
- More Complex Presets: Allow users to create more advanced presets, including scheduling lights to turn on or off at specific times.