**Name: Erlan Akbaraliev**

**Neptun Code: RTSUMV**

**Smart Home Automation**

Overview:

This Python program simulates a smart home automation system with three main components:

1. SmartLight - Controls the brightness of the living room light.

2. Thermostat - Manages the temperature of the living room.

3. SecurityCamera - Monitors the front door and triggers automation rules when motion is detected.

File Structure:

1. SmartHomeApp.py - The main application file.

2. SmartLight.py - Contains the LightController class for managing the living room light.

3. Thermostat.py - Contains the ThermostatController class for managing the thermostat.

4. Camera.py - Contains the CameraController class for managing the front door camera.

Dependencies:

1. tkinter - Used for creating the graphical user interface (GUI).

2. random - Used for generating random values for simulation.

Classes and Their Responsibilities:

1. AutomationSystem (SmartHomeApp class):

- Initializes the main window for the smart home application.

- Creates instances of LightController, ThermostatController, and CameraController.

- Manages the GUI layout and text display.

2. LightController:

- Manages the SmartLight component.

- Controls the brightness of the living room light.

- Provides simulation of light fluctuations.

- Provides a method to set the status of the SmartLight.

3. ThermostatController:

- Manages the Thermostat component.

- Controls the temperature of the living room.

- Provides simulation of temperature fluctuations.

- Provides a method to set the status of the Thermostat.

4. CameraController:

- Manages the SecurityCamera component.

- Controls the motion detection and automation rules.

- Provides a method to set the status of the SecurityCamera.

- Displays an automation rule related to motion detection.

GUI Elements:

1. Text Widget:

- Displays the status of the SmartLight, Thermostat, and SecurityCamera.

2. Buttons:

- Toggle buttons for turning on/off each component.

- Motion detection button for simulating random motion events.

3. Labels:

- Display labels for each component, indicating their status or value.

- Automation rule label for the CameraController.

Simulation:

The simulation is achieved by periodically updating the status or values of each component.

User Interaction:

1. Toggle On/Off Buttons:

- Clicking on the "Toggle On/Off" buttons for SmartLight, Thermostat, and CameraController

will switch their status between "On" and "Off."

2. Random Detect Motion Button:

- Clicking on the "Random Detect Motion" button in the CameraController simulates a random

motion event. If the front door camera is on, it may trigger the automation rule to turn on lights.

3. Text Display:

- The text widget displays the current status of the SmartLight, Thermostat, and SecurityCamera.

- It updates dynamically as the status or values change during the simulation.

Automation Rules:

1. CameraController Automation Rule:

- "Automation Rule: Turn on lights when motion is detected."

- When motion is detected, the CameraController sets the status of the LightController to "On."

Simulation Loop:

1. LightController:

- The `simulate\_light` method is responsible for simulating light fluctuations.

- It updates the light level every second using the `after` method.

2. ThermostatController:

- The `simulate\_temp` method simulates temperature fluctuations.

- It updates the temperature level every second using the `after` method.

3. CameraController:

- The `detect\_motion` method simulates random motion events.

- If motion is detected and the camera is on, it triggers the automation rule to turn on lights.

- It uses the `after` method for periodic simulation.

Integration:

1. The LightController and ThermostatController instances are created independently.

2. The CameraController instance is created with a reference to the LightController instance.

This allows it to set the status of the SmartLight when motion is detected.

Overall Flow:

1. The main application initializes the GUI and creates instances of the controllers.

2. Controllers manage their respective components and simulate changes over time.

3. The user can interact with toggle buttons and simulate motion events.