GUI Documentation

The below are the functionalities of the different classes:

Devices :

The code comprises classes representing different smart devices in a home automation system. The `SmartDevice` class serves as the base, featuring methods for device status manipulation like turning on/off and toggling. Its subclasses, `SecurityCamera`, `SmartLight`, and `Thermostat`, extend these functionalities. The `SecurityCamera` class introduces features for recording and motion detection, relying on an external utility function to simulate motion detection. Meanwhile, the `SmartLight` class manages brightness levels and updates its status based on the brightness setting. Lastly, the `Thermostat` class maintains temperature settings, allowing adjustments and retrieval of the current temperature. Each class encapsulates specific functionalities and behaviors vital to their respective devices within the home automation ecosystem.

AutomationSystem:

The `AutomationSystem` class orchestrates smart devices and rules within a home automation setup. It manages devices by adding, removing, and discovering them on the network. Through methods like `discover_devices`, it mimics the process of discovering IoT devices and adds sample devices like smart lights, thermostats, and security cameras. Additionally, it defines rules such as the `LightMotionRule`, which, when evaluated, triggers actions based on predefined conditions. For instance, the `LightMotionRule` turns on a specific light when motion is detected by a specified camera. This class encapsulates functionalities to execute rules and interact with devices in an organized and manageable manner within a home automation system.

Dashboard:

The provided code defines a `Dashboard` class responsible for displaying and controlling smart devices within a home automation system. This class is built using tkinter and acts as a graphical interface to manage different devices. It includes a list box to showcase available devices and a frame to exhibit the controls associated with the selected device. Upon device selection, the dashboard triggers the creation and display of controls specific to the chosen device using the `ControlFactory`. The `Dashboard` continuously updates the device list and control frames to reflect any changes in device states or configurations. The `update_dashboard` method handles these updates by refreshing the displayed devices and updating the controls according to the current device's state. Additionally, it checks if the current control supports an update method before executing it to ensure compatibility and prevent errors.

DeviceControlling:

The provided code constitutes a `ControlFactory` class and several subclasses of `DeviceControl` designed for managing different smart devices within a graphical user interface implemented using tkinter. The `ControlFactory` maps specific device types to their respective control classes, enabling the creation of device controls dynamically based on the type of device passed. The `DeviceControl` class serves as a template for specific device controls and defines methods such as initialization, creation of GUI components, toggling device states, and updating control displays. Subclasses like `SecurityCameraDeviceControl`, `SmartLightDeviceControl`, and `ThermostatDeviceControl` inherit from `DeviceControl` and implement control features tailored to each device type. These controls manage functionalities like recording status and motion detection for security cameras, light toggling and brightness adjustment for smart lights, and temperature control for thermostats. The code also includes integration guidelines demonstrating how to instantiate controls and update their states within a GUI application.

Simulator:

The presented code introduces a `Simulator` class, which extends the `threading.Thread` functionality to simulate an automated system's behavior within a graphical interface using tkinter. This class orchestrates the interaction between the automation system, the root window, and a dashboard GUI. Upon initialization, the `Simulator` class accepts parameters such as the root window, an automation system, and a dashboard instance. The `setup` method configures the display by packing the dashboard. The `run` method initiates a continuous loop, simulating the operation of the automation system by executing its automation rules and updating the dashboard display at regular intervals using `time.sleep(3)`. The `stop` method allows halting the simulation by setting a flag (`is_running`) to terminate the loop. Overall, the `Simulator` class acts as a control mechanism to visualize the behavior of the automation system through a user interface, adhering to threading principles to avoid blocking the main execution.

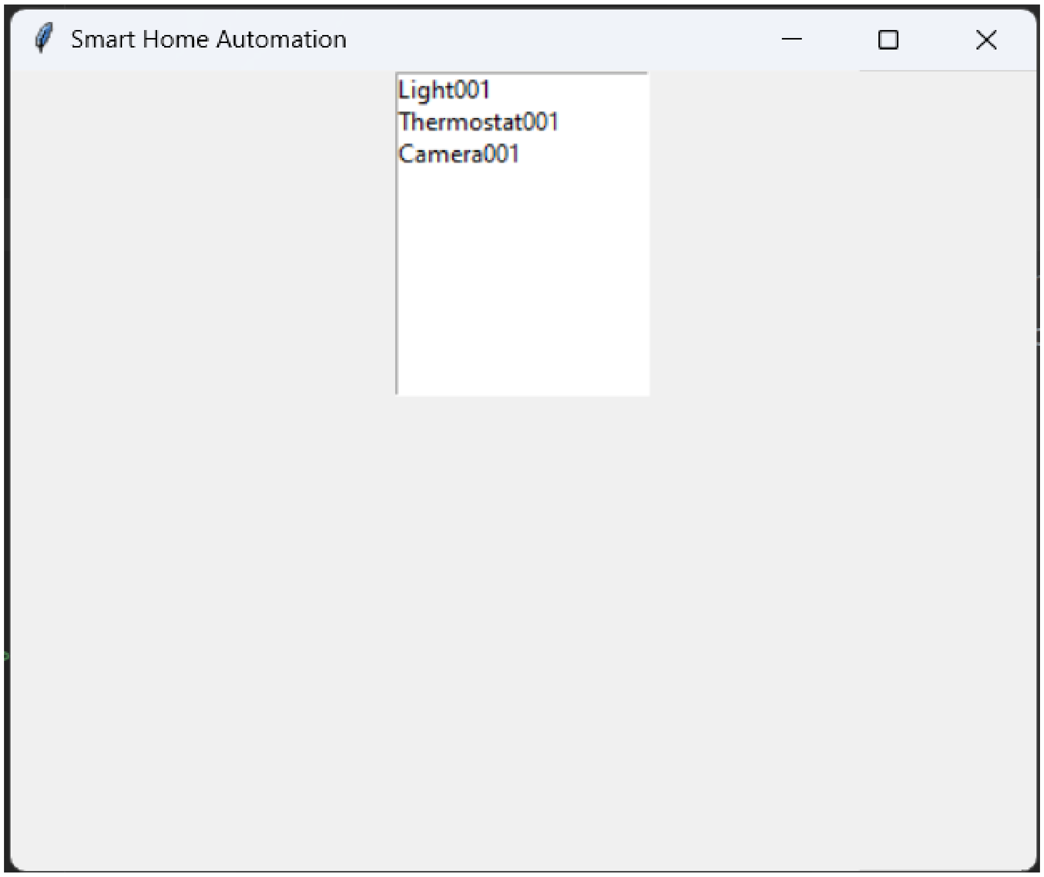
Main method:

The provided code snippet orchestrates the setup and execution of a Smart Home Automation system using Python's tkinter library. It begins by creating a graphical root window that serves as the interface for the automation system. An `AutomationSystem` instance is initialized, where devices are discovered and added to the system. A `Dashboard` is then instantiated, representing the visual interface displaying the status and controls of the connected devices within the system.

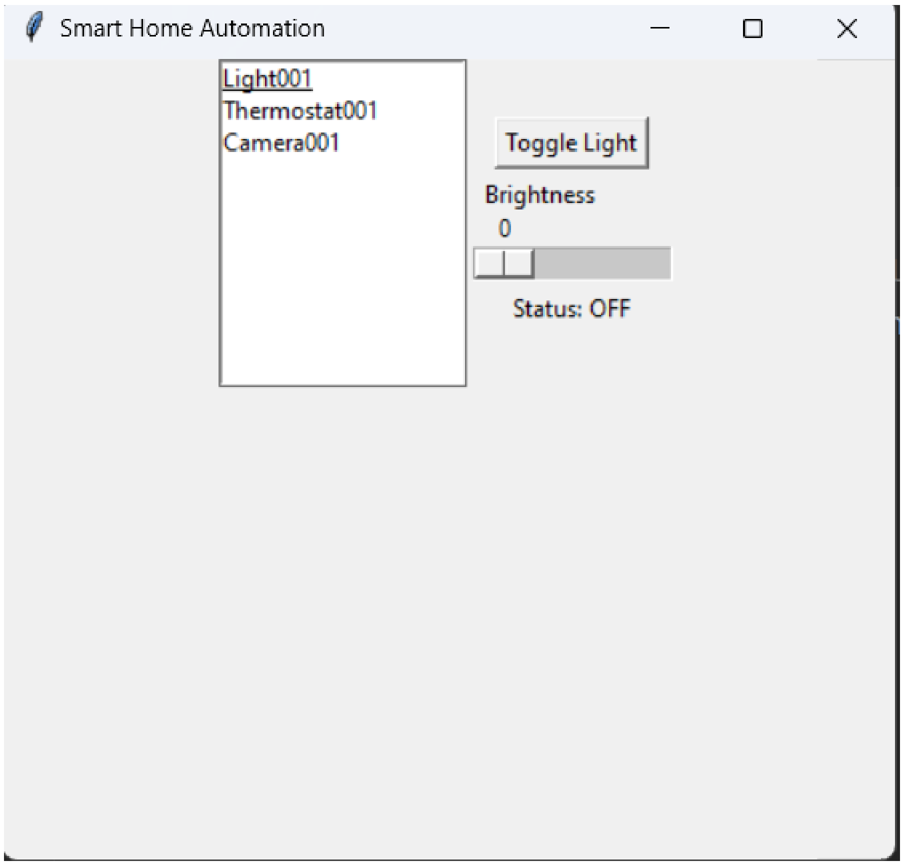
Following this setup, a `Simulator` object is created, responsible for simulating the automation system's behavior. The `Simulator` utilizes the root window, the automation system, and the dashboard to visualize the system's operation. Upon starting the simulation loop, the code enters the main loop for the graphical user interface (`root.mainloop()`), enabling user interaction and continuous simulation of the smart home automation system. This setup enables users to interact with and visualize the behavior of the automation system within a graphical interface in real-time.

Instructions to run the GUI simulation :

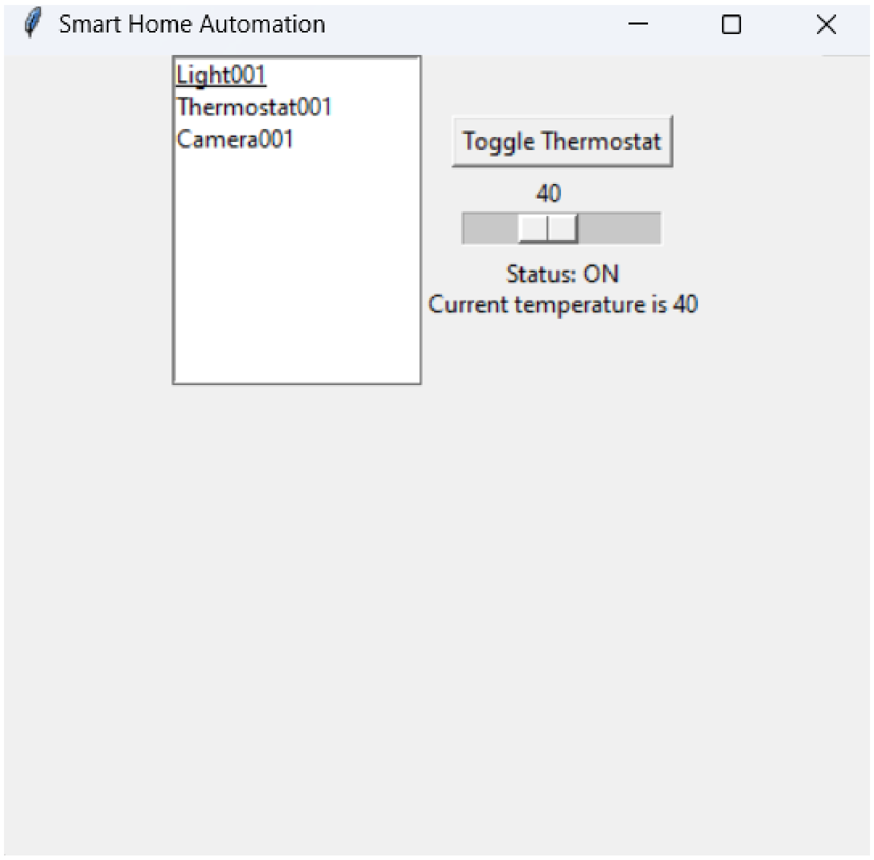
At first run the main method in the main.py class. After thet you will get the GUI like this :



In this GUI the above white frame is where you can operate the select the device that we wish to select. With three devices namely : Light, Thermostat and Camera001. And in order to control the specific device click on it.

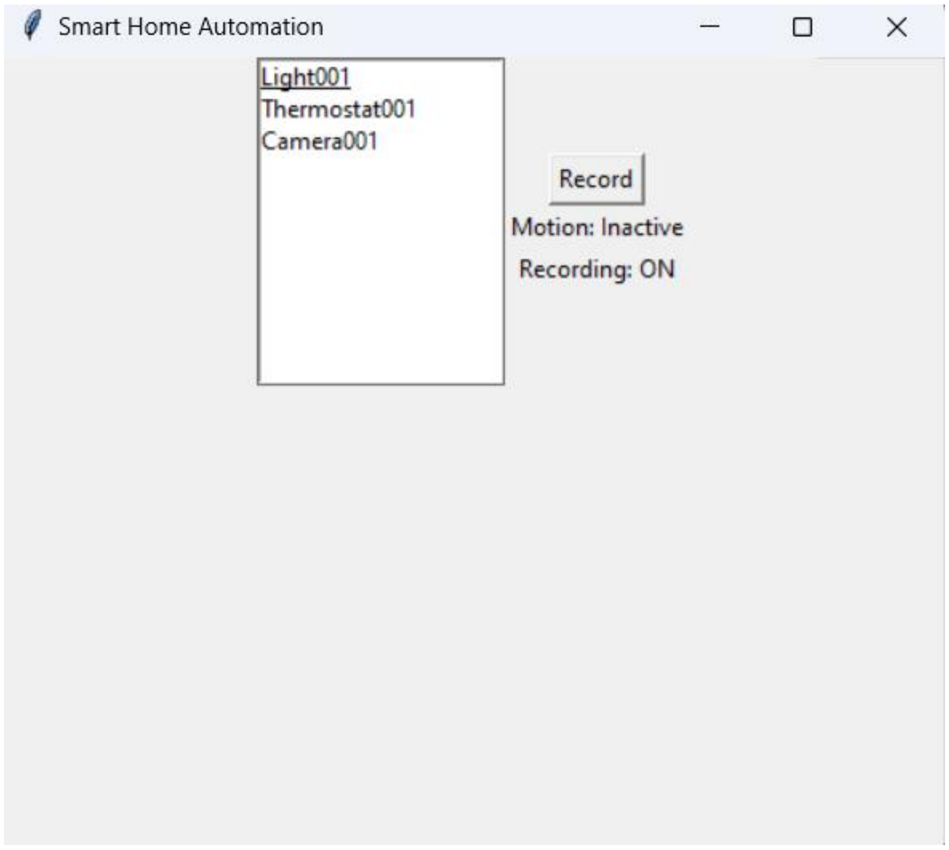


The above instance when the Light is being click in the frame. After click, the controls will be displayed. At first a toggle button “Toggle Light”, when click on it the Light gets on and the Status changes from OFF to ON and we can also adjust the brightness of the light.



Status: ON
Current temperature is 40

Just like the Light This Thermostat has the controls of Toggle button to turn on the temperature with the status being OFF and ON and the brightness level toggle to adjust the brightness level. And after setting the brightness level , the current brightness level is shown.



Motion: Inactive

Recording: ON

And finally the above is the Camera controls where when the record button is pressed the camera starts recording with with recording status ON . Also when the motion is detected there is status active or inactive. If the status is active then the brightness increases adding a layer of security for the house.