System Requirements Specification Index

For

Smart Home System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Smart Home System

System Requirements Specification

1 PROJECT ABSTRACT

HomeHub Technologies, a startup focused on simplifying smart home technology, requires a flexible Smart Home System simulation program. This console-based application will demonstrate how different smart devices can be controlled through a central hub, highlighting the importance of standardized interfaces and device interoperability. The system will model various smart devices (lights, thermostats, security cameras, etc.) and show how they can be monitored, controlled, and automated through common APIs regardless of their specific functionality. This educational tool will help customers understand the benefits of an integrated smart home ecosystem.

2 BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. System needs to simulate different types of smart devices (lights, thermostats, security devices) 2. System must support basic smart home operations such as device registration, control, and automation 3. Console implementation must demonstrate object-oriented programming concepts like Classes and objects, Inheritance and polymorphism, Encapsulation, Method overriding

3 CONSTRAINTS

3.1 CLASS CONSTRAINTS

1. `Device` Class (Base Class):
    - Attributes: id, name, is_on, connected, location
    - Class Variables: device_count
    - Methods: display_info(), toggle_power(), connect(), disconnect()
    - Example: Cannot be instantiated directly
2. `Light` Class (inherits from `Device`):
    - Additional attributes: brightness, color
    - Override methods: display_info(), toggle_power()
    - New methods: dim(), change_color()
    - Example: `Light("L001", "Living Room Light", True, True, "Living Room", 80, "White")`
3. `Thermostat` Class (inherits from `Device`):
    - Additional attributes: temperature, mode, target_temp
    - Override methods: display_info(), toggle_power()
    - New methods: set_temperature(), change_mode()
    - Example: `Thermostat("T001", "Main Thermostat", True, True, "Hallway", 22.5, "Heat", 23.0)`
4. `SecurityDevice` Class (inherits from `Device`):
    - Additional attributes: armed_status, sensitivity
    - Override methods: display_info(), toggle_power()
    - New methods: arm(), disarm()
    - Example: Cannot be instantiated directly
5. `Camera` Class (inherits from `SecurityDevice`):
    - Additional attributes: resolution, recording
    - Override methods: display_info(), arm(), disarm()
    - New methods: start_recording(), stop_recording()
    - Example: `Camera("C001", "Front Door Camera", True, True, "Front Door", "Armed", 80, "1080p", False)`

6. `MotionSensor` Class (inherits from `SecurityDevice`):
    - Additional attributes: detection_range, last_triggered
    - Override methods: display_info(), arm(), disarm()
    - New methods: detect_motion(), reset_trigger()
    - Example: `MotionSensor("M001", "Backyard Sensor", True, True, "Backyard", "Armed", 60, 10, None)`
7. `SmartHome` Class:
    - Attributes: name, devices, rooms, automations, mode
    - Methods: add_device(), remove_device(), get_devices_by_type(), get_devices_by_room(), find_device(), execute_automation(), change_mode()
    - Example: `SmartHome("My Home")`

3.2 Operation Constraints

1. Basic Interactions:
    - Devices can be turned on/off through toggle_power() method
        - Implementation may vary by device type
    - Security devices can be armed/disarmed through arm()/disarm() methods
        - Behavior changes based on device type and mode
    - Lights can be dimmed with dim(brightness) method
        - Brightness value must be 0-100
    - Thermostats control temperature with set_temperature(temp) method
        - Valid modes are "Heat", "Cool", "Auto", "Off"
    - Devices can connect/disconnect from the network
2. Input Validation:
    - All device IDs must be unique
    - Brightness values must be between 0-100
    - Temperature values must be between 5-35 (Celsius)
    - Sensitivity values must be between 0-100
    - Valid SmartHome modes are "Home", "Away", "Night", "Vacation"

3. Exception Handling:
    - Must handle DeviceNotFoundException when device not found
    - Must handle InvalidInputException for invalid user inputs
4. Object Oriented Requirements:
    - Must use protected attributes (single underscore prefix)
    - Must use private attributes for SmartHome (double underscore prefix)
    - Must implement inheritance hierarchy as specified
    - Must use polymorphism with method overriding

3.3 Output Constraints

1. Display Format:
    - Device info must show: ID, name, power status, connection status, type-specific attributes
    - Smart Home info must show: name, mode, number of devices, devices by room
2. Output Format:
    - Must show in this order:
        - Show "===== SMART HOME SYSTEM ====="
        - Show "Home: {name}"
        - Show "Mode: {mode}"
        - Show "Total Devices: {count}"
        - Show "Devices by Room:" followed by counts by room
        - Show "Connected Devices: {connected_count}/{total_count}"

4. Template code Structure:

1. Device Classes:

- `Device` (base class)
- `Light` (derived from Device)
- `Thermostat` (derived from Device)
- `SecurityDevice` (derived from Device)
- `Camera` (derived from SecurityDevice)
- `MotionSensor` (derived from SecurityDevice)

2. SmartHome Class:

- `SmartHome`

3. Exception Class:

- `DeviceNotFoundException`
- `InvalidInputException`

4. Program Control:

- `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. View the main menu
3. Select operations:
   - Option 1: Add Device to Smart Home (numeric selection)
   - Option 2: Control Device (turn on/off, adjust settings)
   - Option 3: Run Automation Scenario
   - Option 4: Change Home Mode
   - Option 5: Display All Devices
   - Option 0: Exit
4. Perform operations on the smart home system
5. View results after each operation
6. Exit program when finished