# Project 6 Update

Developer(s): Samiksha Patil

Project Title: SmartHomeSim: C++ Simulator for Smart Device Control

Work Done :

In the first week of Project 6, I implemented core system components for simulating a smart home environment using C++. Key tasks completed include:

- Implemented base class SmartDevice with toggle/set state logic and observer management.
- Created device subclasses: Fan, Light, and Thermostat, each with specific behaviors.
- Implemented Sensor class as a subject that notifies subscribed devices of environmental changes.
- Built DeviceLogger, which records state changes and actions using the Observer Pattern.
- Integrated CLI logic in main.cpp to support user commands like adding devices, toggling, and triggering sensor events.
- Connected the DeviceFactory to dynamically instantiate devices from user input using the Factory Pattern.
- Configured the Thermostat to dynamically change strategies (EcoMode, ComfortMode) based on sensor input.

Changes or Issues Encountered :

The overall structure remained faithful to the Project 5 design. However, the following refinements were made during early implementation:

Patterns Used So Far :

The following design patterns have been successfully implemented and are contributing to a modular, extensible design:

- Factory Pattern: DeviceFactory creates new device instances based on user input without exposing subclass logic.

- Observer Pattern:
  Sensor acts as the subject, notifying SmartDevice observers.
  DeviceLogger is attached to devices to track and log state changes.

- Strategy Pattern:
  Thermostat uses a TemperatureStrategy pointer, dynamically switching between EcoMode and ComfortMode based on sensor input.

These patterns allow for loose coupling, runtime flexibility, and improved clarity in simulation logic.

Plan for Next Iteration (Project 7 Goals) :

To complete the full design presented in Project 5 and meet all functional requirements, the following features are planned for implementation in the next iteration:

Use Case 2: Schedule Device Action -

Implement a scheduling framework that allows users to assign a strategy (e.g., turn on/off after X seconds or at a specific simulated time).

Introduce a new class (e.g., Scheduler) or expand the DeviceController to manage timed tasks.

Devices will be assigned scheduling strategies that encapsulate when and how the device should change state.

Strategy options may include periodic activation, delayed toggle, or time-of-day behavior.

CLI enhancement: Add a command like schedule <device> <on/off> <time>.

Use Case 4: View Logs -

Already partially implemented using the DeviceLogger class, which observes and records all device state changes.

In the next iteration, expand logging to also include sensor values and strategy changes (e.g., when Thermostat switches from EcoMode to ComfortMode).

Enhance CLI to support filtering or exporting logs, e.g., by device or time range.

Format logs more clearly with tags like [SENSOR], [STATE], or [STRATEGY] to improve readability.