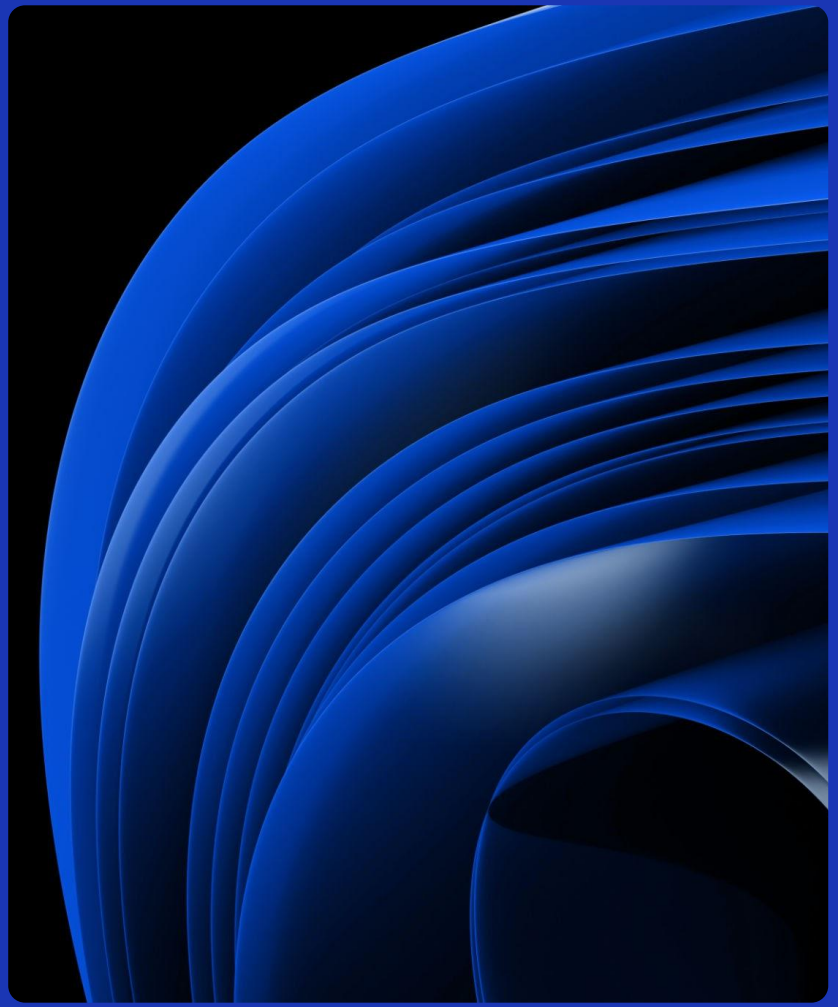
Java Course

Capstone Project

Group 5



Team Roles Distribution

Hadis Delbord

Simulation of the Energy Sources

- Energy source manager
- Battery manager
- Charge of batteries with progress bar
- Logs of energy source (write log)
- Unit test of energy source

Yug Dave

Simulation of the Smart Objects

- Smart object manager
- Consumption of smart objects
- Logs of smart objects (write log)
- Unit tests

Amir Hossein Pakdel

Management system for the house consumption

- User Interface
- Logs management (read, delete, export, filter)

System Requirements

Energy Source Management

The system should use different energy sources for concurrent charging of multiple batteries, allowing smart objects to consume power

Battery Management

The system must be able to handle multiple batteries, support charging and discharging operations for batteries, track battery charge levels.

Smart Object Management

The system must allow adding, activating, and deactivating smart objects. Each smart object should have an energy requirement, and the system should manage concurrent energy consumption by multiple smart objects

System Requirements

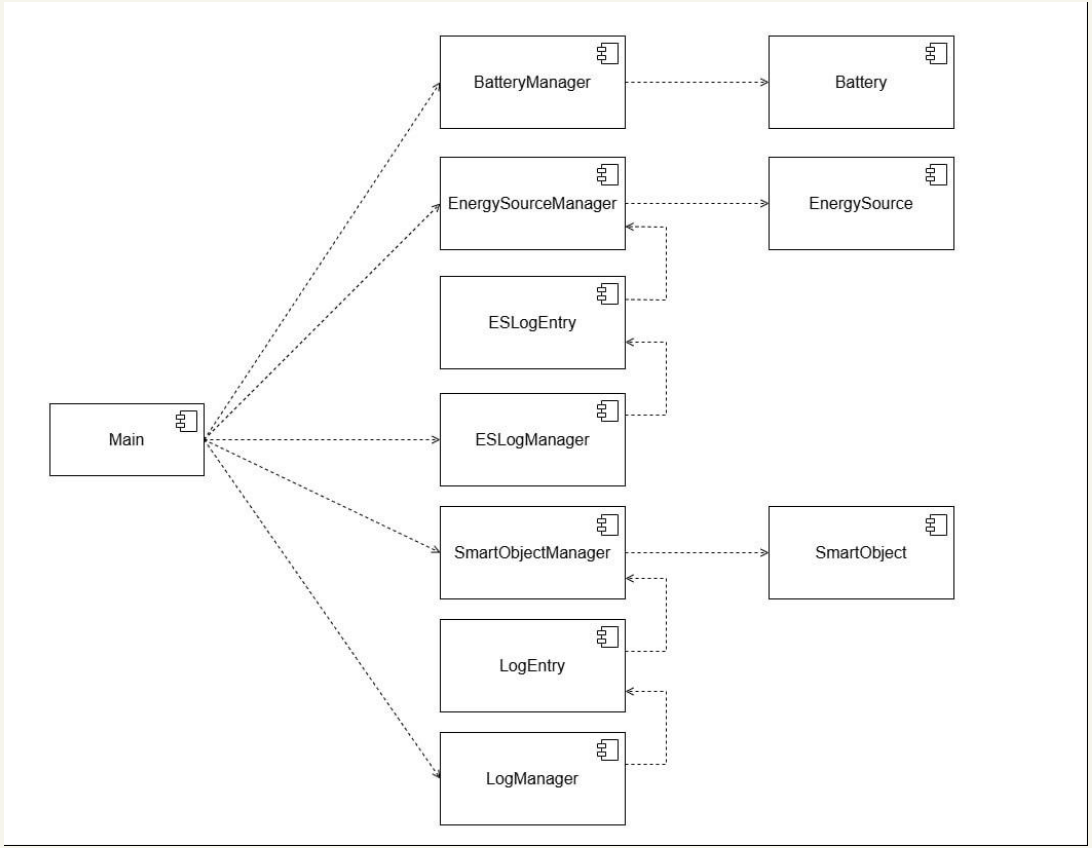
Log Management

The system should maintain logs for actions performed by smart objects and energy sources (e.g., battery usage, switching) and also allows logs to be filtered by object name, energy source name, or date. The system must support exporting logs in CSV format to facilitate record-keeping and data analysis.

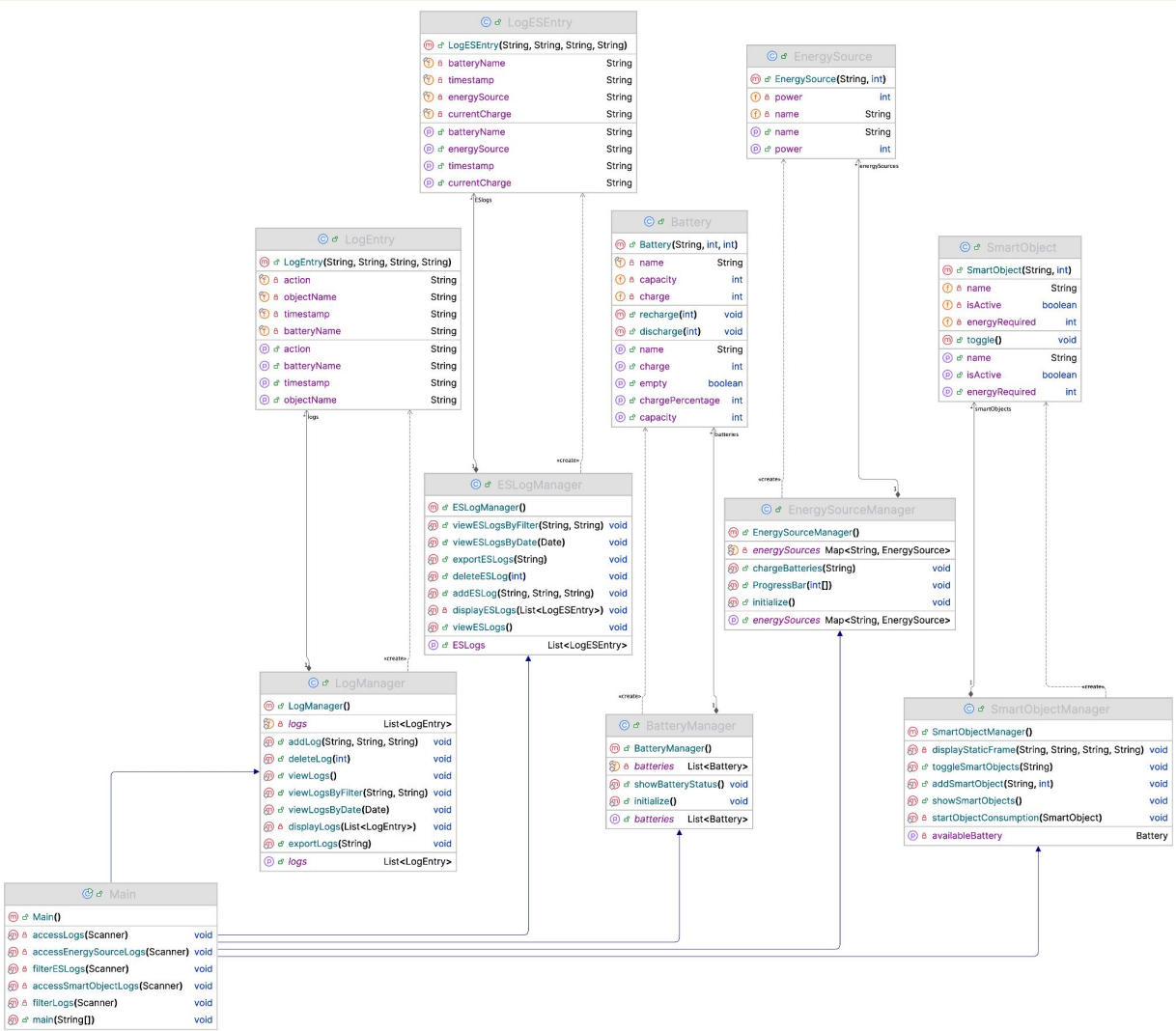
User Interaction

The system should have a user interface to connect energy sources based on weather to batteries, add and manage smart objects (e.g., on/off), display logs, and view battery status.

Component Diagram



UML Class Diagram



Implementation

Management I/O

The system handles input and output for various operations across different modules, enabling user interactions and facilitating concurrent activities like charging batteries and controlling smart objects.

1. **Battery and Energy Source Management**
   - ➔ Users select weather conditions (e.g., sunny, rainy) to choose an energy source.
   - ➔ Charging progress is displayed via dynamic progress bars updated in real time.
2. **Smart Object Management**
   - ➔ Users can define smart objects by providing a name nad required energy
   - ➔ Users can toggle the state of smart objects (on/off), which directly impacts battery consumption.
   - ➔ Consumption progress is displayed
3. **Log management and Exporting**
   - ➔ Users can view logs (LogManager and ESLogManager) by filtering (e.g., by battery or date).
   - ➔ Logs can be exported to files for record-keeping.

User Interface

Menu

- Easy-to-use menu-driven options for managing batteries, energy sources, and smart objects.
- Real-time feedback for every user action (e.g., "Battery fully charged," "SmartLamp turned on").

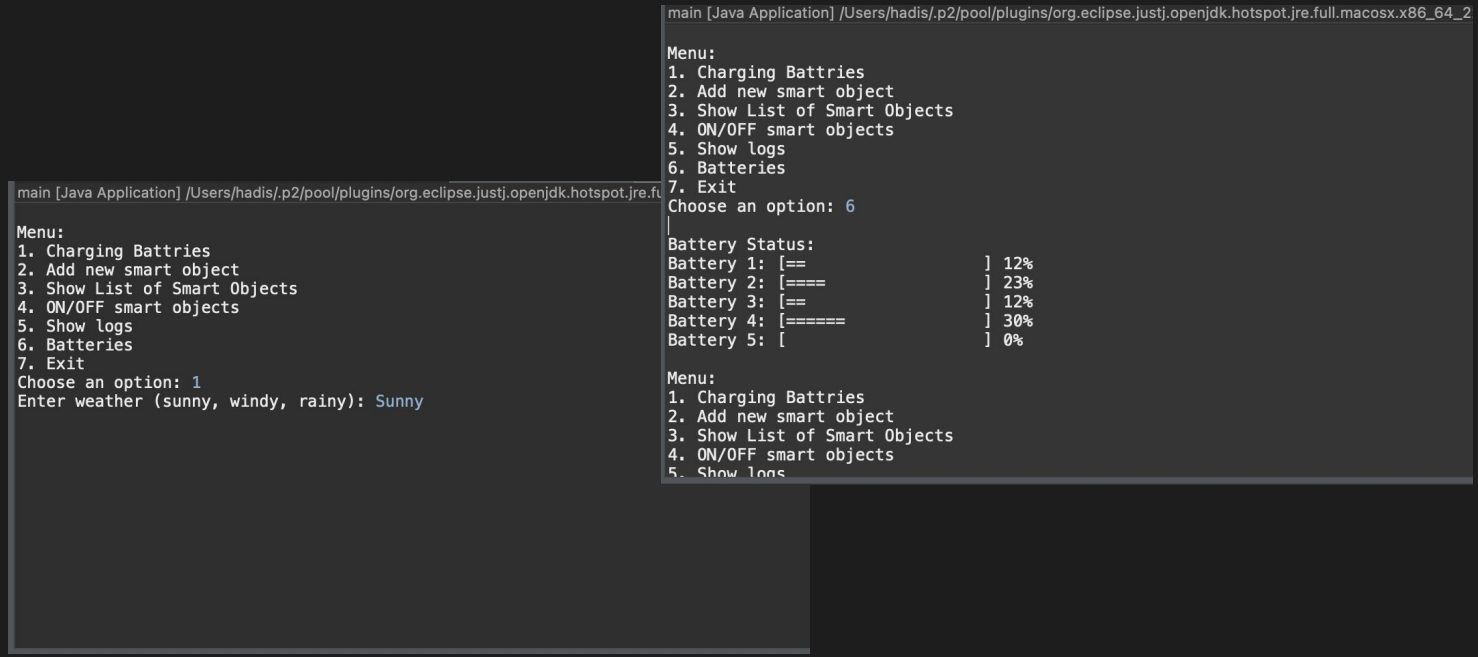
Progress Displays:

- Dynamic progress bars for monitoring battery charging in real time.
- Progress bars while smart object consume batteries

Logs and Reports:

- Display logs in tabular format on the console.
- Export logs to files for detailed analysis.

User Interface



User Interface

```
main [Java Application] /Users/hadis/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jr
Menu:
1. Charging Battries
2. Add new smart object
3. Show List of Smart Objects
4. ON/OFF smart objects
5. Show logs
6. Batteries
7. Exit
Choose an option: 5

===== Log Management =====
1. Access Energy Source Logs
2. Access Smart Object Logs
3. Back to Main Menu
Enter your choice: 2

===== Smart Object Log Management =====
1. View All Logs
2. View Logs by Filter
3. Delete Log by ID
4. Export Logs to File
5. Back to Main Menu
Enter your choice:
```

```
main [Java Application] /Users/hadis/.p2/pool/plugins/org.eclipse.justj.openjdk

Menu:
1. Charging Battries
2. Add new smart object
3. Show List of Smart Objects
4. ON/OFF smart objects
5. Show logs
6. Batteries
7. Exit
Choose an option: 2
Enter object name: TV
Enter energy required: 50
```

```
main [Java Application] /Users/hadis/.p2/pool/plugins/org.eclipse.justj.openjdk.hotspot.jr

Menu:
1. Charging Battries
2. Add new smart object
3. Show List of Smart Objects
4. ON/OFF smart objects
5. Show logs
6. Batteries
7. Exit
Choose an option: 4
Enter object names to toggle (comma-separated): TV,Fan,Lamp
```

Concurrency

Battery Charging:

Uses multithreading to charge multiple batteries simultaneously.

Each battery runs in a separate thread to ensure efficient use of resources.

Charging status is updated in real time

Smart Object Operations:

Allows multiple smart objects to operate concurrently, consuming energy from batteries.

Concurrency (Battery Charging)

```
List<Thread> threads = new ArrayList<>();
        for (int i = 0; i < batteries.size(); i++) {
            int index = i; // Required for use in lambda expression
            Battery battery = batteries.get(i);
            Thread chargingThread = new Thread(() -> {
                while (true) {
                    synchronized (battery) { // Ensure thread-safe access
                        if (battery.isFull()) {
                            break; // Exit if battery is fully charged
                        }
                        battery.recharge(source.getPower()); // Increment battery charge
                        synchronized (percentages) {
                            percentages[index] = battery.getChargePercentage(); // Update percentages
                        }
                        ESLogManager.addESLog(source.getName(), battery.getName(), String.valueOf(battery.getCharge()));
                    }
                    // Simulate charging time
                    try {
                        Thread.sleep(500); // Adjust the speed of charging
                    } catch (InterruptedException e) {
                        Thread.currentThread().interrupt();
                        System.out.println("Recharging interrupted for " + battery.getName());
                        return; // Exit the thread
                    }
                }
            });
```

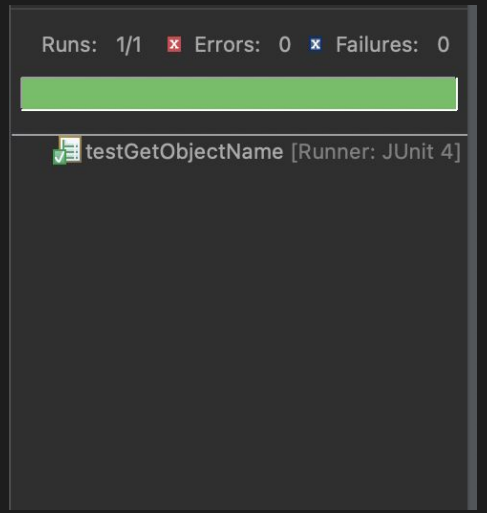
Concurrency (Consuming Energy)

```
Thread consumptionThread = new Thread(() -> {
        try {
            while (object.isActive() && !stopFlag.get()) {
                Battery battery = getAvailableBattery();
                if (battery == null) {
                    displayStaticFrame(object.getName(), "No batteries available", "---", "---");
                    LogManager.addLog(object.getName(), "None", "No Batteries Available");
                    break;
                }
                boolean consumed = false;
                try {
                    batteryLock.lock();
                    if (battery.getCharge() >= object.getEnergyRequired() && !stopFlag.get()) {
                        battery.discharge(object.getEnergyRequired());
                        consumed = true;
                        displayStaticFrame(object.getName(), "Consuming power", battery.getName(), battery.getChargePercentage() + "%");
                        LogManager.addLog(object.getName(), battery.getName(), "Consuming Power");
                    }
                }
                }
                Thread.sleep(1000); // Simulate time taken for energy consumption
            }
        } catch (InterruptedException e) {
            System.out.println("Consumption interrupted for " + object.getName());
            Thread.currentThread().interrupt();
        }
    });
```

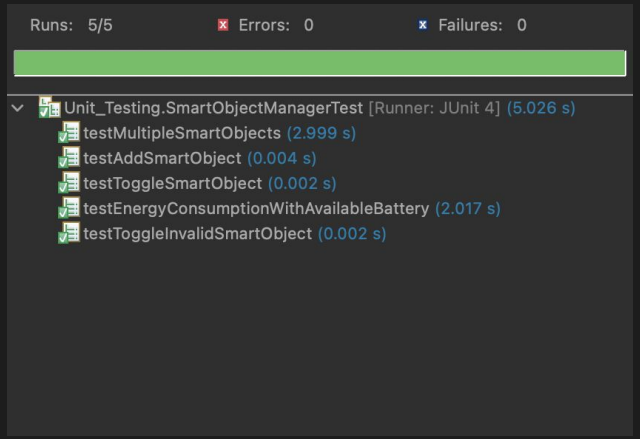
Unit Test

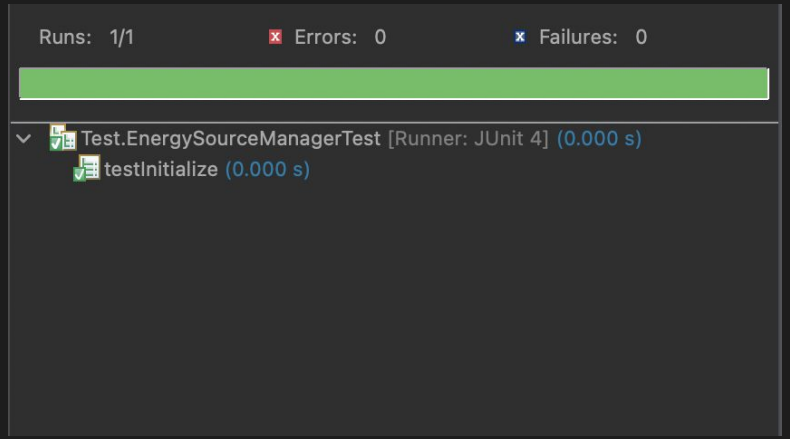
- Main Test
- Energy Source Manager Test
- Smart Object Manager Test
- Energy source Log Manager Test
- Energy Source Log Entry Test
- Smart Object Log Manager Test
- Smart Object Log Entry Test
- Get Object Name Test

Unit Test

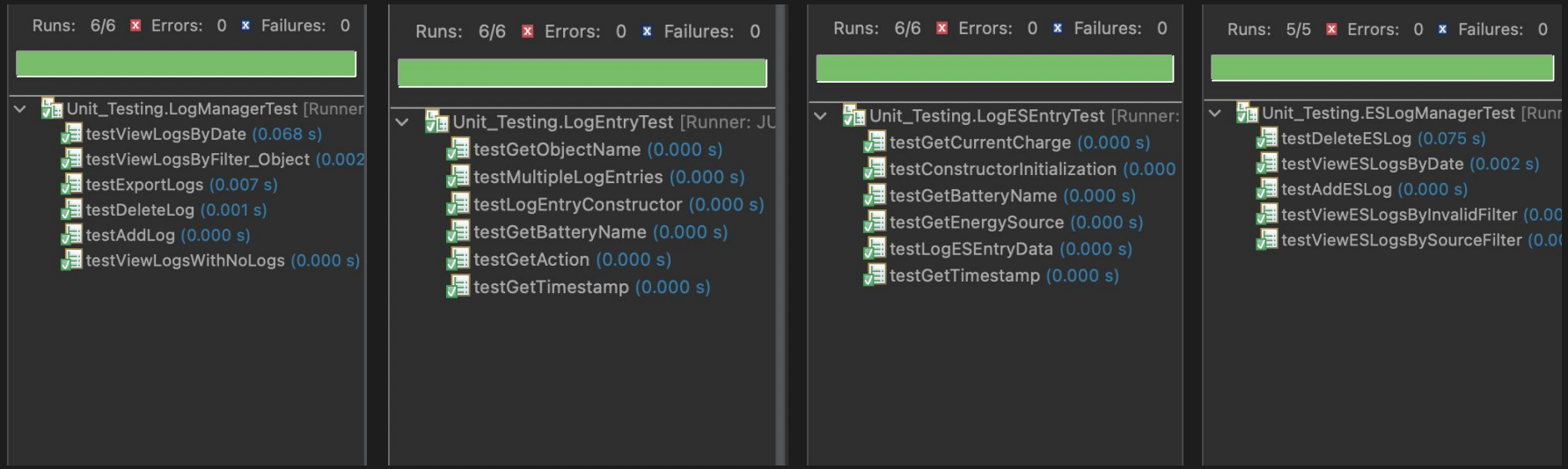


Unit Test





Unit Test



Thank You!