Group Name: The girls + Kyle
Group Members:

Megan Azmanov - u24575292
Gabi De Gouveia - u24594084
Gabriela Berimbau - u24711013
Rachel Clifford - u24647374
Kyle McCalgan - u24648826
Kahlan Hagerman – u24601358
Sofia Finlayson - u24593240

Functional Requirements:

Note: C (Creational), S (Structural), B (Behavioral)

FR1: Plant lifecycle management (B1)
(Track and manage each plants lifecycle states & Allow forward/backward transitions and final states.)

Feature: Plant lifecycle tracking
Function: State transitions between growth stages
Behaviour: Responds to events by changing states (ie. Seedling -> Growing -> Mature -> ReadyForSale -> Sold).
Patterns: *State*
UML: Plant, PlantState, SeedlingState, GrowingState, MatureState, FloweringState, DormantState
Criteria:

- Given a **Plant** in **SeedlingState**, when time- stepped growth occurs, then the Plant moves to **GrowingState** and lifecycle timestamp updates.
- Given a Plant in **MatureState**, when **markForSale()** invoked, then state becomes **ReadyForSale**.
- There is at least one final state with no outgoing transitions.

FR2: Strategy-based plant care (B2)
(Apply different care behaviors to different types of plants using interchangeable strategies – system must allow switching care strategies at runtime)

Feature: Dynamic care system
Function: Apply care routine depending on plant type
Behaviour: Behaviour changes when care strategy swapped
Patterns: *Strategy*
UML: CareStrategy, SucculentCareStrategy, RosCareStrategy
Criteria:

- Given a **Plant** with **SucculantCareStrategy**, when **applyCare()** invoked, then the correct watering interval and fertilizer rules are applied.
- Given same plant, when strategy switches to **RoseCareStrategy**, then subsequent **applyCare()** uses the new rules.

FR3: Observers for plant condition notifications (B3)
(Notify interested parties about plant state changes via PlantObserver – Observers can attach/detach at runtime)

Feature: Automatic plant monitoring
Function: Notify observers when conditions change
Behaviour: Sends alerts or triggers actions dynamically
Pattern: *Observer*
UML: PlantObserverSubject, PlantObserver, StaffMember, InventoryManager, GreenHouseManager
Criteria:

- Given **WateringObserver** attached to a plant, when moisture < threshold, then **WateringObserver.handleChange()** is called and an alert entry is created.
- Given **Observer** detached, when threshold crossed, then no notification is sent to that observer.

FR4: Encapsulated care and action commands (B4)
(Encapsulated care actions as command objects to allow scheduling, undo, and replay)

Feature: Action scheduling system
Function: Execute, queue, undo watering/fertilizing commands
Behaviour: Executes commands in sequence & logs behaviour
Patterns: *Command*
UML: Command, WaterPlantCommand, FertilizerPlantCommand, AdjustSunlightCommand, CareSchedule
Criteria:

- Given a schedule WaterPlantCommand in invoker queue, when scheduled time arrives, then command executes and plant moisture increases (inoker logs the action)
- Command can be queued and executed in order.

FR5: Centralized greenhouse facade (S1)
(Provide a GreenHouseFacade exposing: plant inventory queries, bulk-care trigger, add/remove plants, and reporting endpoints, so that other subsystems call a single API)

Feature: Simplified interface to subsystems
Function: Query inventory, trigger care, add/remove plants
Behaviour: Handles requests and delegates to correct subsystem
Patterns: *Facade*
UML: GreenHouseFacade, InventorySystem
Criteria:

- Given **SalesFloor** needs to check availability for **PlantType X**, when facade query called, then it returns count and availability status in one call.
- Internal subsystems can change without changing façade API.

FR6: Staff coordination via mediator (B5-6)
(Coordinate staff and customer interactions using NurseryMediator or SalesFloorMediator to route messages without tight coupling.)

Feature: Staff communication system
Function: Route messages & assign tasks
Behaviour: Coordinates & escalates requests between staff
Pattern: *Mediator, Chain of Responsibility*

UML: Colleague, Person, SalesFloor, Greenhouse, NurseryMediator, NurseryCoordinator, customer, corperateCustomer, regularCustomer, walkInCustomer, GreenHouseStaff, staffMembers, FloorManager, SalesFloorStaff, NurseryOwner
Criteria:

- Given customer requests assistance, when **SalesFloorMediator** receives request, then it assigns appropriate **SalesAssistant**.
- Given **SalesAssistant** cannot approve refund, when escalation required, then request passes up chain: **FloorManager -> GeneralManager**

FR7: Inventory CRUD (create,read,unpdate,delete) & transaction-safe updates (S1)
(Add, remove, reserve, and commit plants in the InventorySystem - Inventory updates must be transaction-safe)

Feature: Stock management
Function: Add/remove/update plant inventory
Behaviour: Updates only after successful transaction
Pattern: *Facade*
UML: InventorySystem, GreenHouseFacade
Criteria:

- Given checkout in progress, when payment fails, then inventory remains reserved and not decremented.
- Given payment success, then inventory decremented and order recorded.

FR8: Purchase customization (S2)
(Allow customers to add customisations to purchases at checkout)

Feature: Customizable orders
Function: Add gift wrap, pot, ribbon to base item
Behaviour: Dynamically adds new features to product
Pattern: *Decorator*
UML: Plant, Decorator, RibbonDecorator, giftWrapDecorator, decoartiveDecorator
Criteria:

- Given base **PlantProd**, when **giftWrapDecorator** applied, then final price increases appropriately and final receipt includes “Gift wrap”
- Multiple decorators can be stacked

FR9: Order cloning for repeat customers (C1)
(Allow fast reorder for returning customers by cloning a previous)

Feature: Quick reordering
Function: Clone existing order object
Behaviour: Creates identical order with new ID & timestamp
Pattern: *Prototype*
UML: Order, ConcreteOrder
Criteria:

- Given **RegularCustomer** with prior **Order A**, when **clone()** invoked, then new Order instance equals **Order A** content-wise but has a different order ID and timestamp.

FR10: Plant construction using Builder (C2)
(Construct plants with many optional parts)

Feature: Complex object creation
Function: Assemble plant with various parts
Behaviour: Builds consistent plant objects step by step
Pattern: *Builder*
UML: PlantBuilder, PlantDirector, buildBasicPlant, GiftArrangement
Criteria:

- Given **PlantDirector** with **RoseBuilder** and configuration, when **build()** invoked, then constructed **Plant** has those attributes and is valid.

FR11: Template Method for payment (B7)
(Handle payments through a consistent checkout sequence with concrete subclasses for different payment types.)

Feature: Unified payment process
Function: Execute a defined sequence of steps: verify → process → confirm → receipt
Behaviour: Core algorithm fixed, but subclasses implement specific payment details
Pattern: *Template Method*
UML: PaymentProcessor, CashPayment, creditCardPayment
Criteria:

- Given customer selects CardPayment, when checkout() executed, then base flow runs with card-specific processing in overridden step.
- Adding a new payment type only requires subclassing PaymentTemplate.

FR12: Chain of Responsibility for complex request handling (B6)
(Route customer requests through a chain of handlers until one can handle it (ie. SalesAssistant -> FloorManager -> GeneralManager))

Feature: Escalation handling
Function: Route requests through hierarchy
Behaviour: Forwards until handled and logs results
Pattern: *Chain of Responsibility*
UML: GreenHouseStaff, staffMembers, FloorManager, SalesFloorStaff, NurseryOwner
Criteria:

- Given refund request beyond sales assistant authority, when request processed, then it is forwarded to **FloorManager** and logged with escalation reason.

Non-Functional Requirements:

NFR1: Scalability and Extensibility:
Description:

- The system must support an increasing number of plants, customers, and staff without requiring major redesign.
- Adding new plant families, care strategies, or customisation options should not require changes to existing code.

Quality Attribute: Scalability, Maintainability
Patterns: *Strategy, Builder, Observer, Facade*

Criteria:

- When 500+ plant objects are added, the system continues to perform lifecycle updates without noticeable slowdown.
- Adding a new CareStrategy class or plant family requires no modification to existing components.

NFR2: Reliability and Consistency:
Description:

- The system must ensure reliable state transitions and consistent data across components, especially during inventory updates, staff coordination, and concurrent operations.

Quality Attribute: Reliability, Consistency
Patterns: *State, Command, Singleton, Facade*
Criteria:

- If a transaction fails, inventory and plant states remain unchanged.
- No duplicate, skipped, or invalid state transitions occur during concurrent updates.

NFR3: Security and Access Control:
Description:

- Sensitive operations (e.g., editing prices, removing plants, processing payments) must be controlled by staff roles and verified by the system before execution.

Quality Attribute: Security
Patterns: *Chain of Responsibility, Mediator, Facade*
Citeria:

- Unauthorized users attempting restricted actions receive “Access Denied,” and all access attempts are logged.
- Role validation occurs before any command or transaction executes.