**Functional Requirements**

Below are the proposed functional requirements for the Plant Nursery.Each FR is paired with the design pattern(s) that implement it :

**1 ) Plant creation & variety (Factory)**  
The system will create Plant objects of different types (Succulent, Rose, Lavender, Baobab, Cactus, etc.) via an Abstract Factory so new plant types can be added without changing client code.  
*A*dding a new concrete Plant subclass and registering it with the PlantFactory allows creation via PlantFactory::create("NewPlant") .

**2 ) Building a plant**   
The system will assemble the plant environment before the ConcretePlant’s factory method returns the product(as it will need that specific environment to be created).

**3 ) Different care strategies (Strategy)**  
The system will apply different care strategies (watering schedule, sunlight exposure, fertiliser frequency) depending on plant type by swapping CareStrategy implementations at runtime.  
*Acceptance:* for a given Plant, calling plant->performCare() executes the appropriate CareStrategy behavior and logs that action (e.g., “Baobab seed : watered 200ml”).

**4 ) Plant life-cycle & state transitions (State)**  
Each plant will model its life-cycle states (Seed → Sprouting → Mature → Wilted) using the State pattern; state changes will alter behavior ,and optimally the plant will end up being in a state to be available for sale.

**5 ) Plant health notifications (Observer)**  
Environment monitors and staff will be notified when plant properties cross thresholds (low water, disease risk, ready-for-sale) via the Observer pattern.  
.

**6 ) Regulate building a Plant (Template Method)**  
We are going to standardize and simplify the process of creating a plant along with its nurturing environment and equipment (e.g., pot/container, soil setup, and initial care routines).  
The system will use the Template Method pattern to define the step-by-step process for creating a plant and its environment.

**7 ) Inventory manager**   
The system must maintain a centralized InventoryManager () that tracks all plants , stock counts, and provides queries.

**8 ) Orders (Command + Composite)**  
The customer will place an order for their desired plant(s) with the sales clerk.An order can just be singular or be a bulk of other orders.

**9 ) Customer browsing (Iterator & Mediator)**  
Customers must be able to browse the sales-floor inventory using an Iterator that supports filtering by properties such as sunlight requirements, price, or size.  
When a customer selects or purchases a plant, the Mediator coordinates communication between the sales floor, inventory manager, and staff — automatically notifying relevant parties (e.g., restocking staff, greenhouse) if stock levels drop below a predefined threshold.

**10 ) Personalisation of purchases (Decorator)**  
Customers must be able to personalise a plant purchase (decorative pot, gift wrapping, arrangement) using the Decorator pattern for addons.

**11 ) Staff task execution & scheduling (Command)**  
Staff actions (water plant X, fertilize bed Y, restock inventory, assist customer) will be represented as Commands that can be queued, executed, logged, and possibly undone.

**12 ) Save order details(Prototype**  
The system can use the Prototype pattern to allow saving and cloning of complete plant orders or decorated plant objects.Then when a similar order is needed in the future, the system can duplicate an existing prototype (e.g., a Valentines Day Basket).

**13 ) Order Execution (Prototype & Command)**  
The system will use the Iterator pattern to traverse through all stored orders and identify those that are ready to be processed (e.g., “plants matured,” “all components available”).  
For each such order, a corresponding Command object (e.g., PrepareOrderCommand, PackageOrderCommand, DeliverOrderCommand) will be created and executed by the staff or system scheduler which will ensure flexible, queued, and undoable order handling.

**Non-Functional Requirements**

Below are our Non-Functional Requirements which describe the qualities our system will have :

**1 ) Scalability (Performance / Scalability)**  
The system shall support a considerable amount of plant objects and simultaneously maintain communication between staff/customer sessions on a single desktop-class machine without perceptible UI lagging (UI response under 300ms for browsing queries).

**2 ) Reliability / Availability (Reliability)**  
Inventory and transaction operations shall be ACID-like within the simulator: a purchase must either fully complete (inventory updated and order recorded) or fully roll back on error. The system will not be allowed to lose committed orders during normal operation.

**3 ) Usability (Usability)**  
The system is going to provide clear actions for core users (staff and customers) with contextual help.Typical browsing and purchase flows should require no more than 7 clicks for a customer to find, customise, and purchase a plant.

**4 ) Observability for Security and Accountability**  
Most of our staff actions and customer transactions will be logged with timestamps and actor IDs so that inventory changes and care actions can be audited. Logs must be exportable.