**IDEA DESCRIPTION**

The project is a **Nursery Management Platform** aimed at tracking the care and condition of plants across three types of nurseries (tree nurseries, vegetable nurseries, and ornamental nurseries). Farmers will be able to update plant information such as irrigation status, fertilization status, and pest control. These updates will be tracked over time in a table. On the other hand, managers will have the ability to view updates without making modifications, and they can manage product requests made by farmers for plant care. Farmers can submit product requests, which managers can approve or reject.

**USER STORIES**

**1. Farmer**

* **As a Farmer**, I want to select the irrigation status (e.g., Watered, Needs Watering, Overwatered) for each plant, so I can keep track of the plant’s irrigation status.
* **As a Farmer**, I want to update the fertilization status and pest control status for each plant, so I can properly care for the plants.
* **As a Farmer**, I want to see a table of previous updates for each plant, including date and time, so I can track all the changes made over time.
* **As a Farmer**, I want to set the final condition of the plant (Green, Yellow, Red) and write a note explaining why I chose that condition, so I can document the plant’s current state.
* **As a Farmer**, I want to submit product requests for plant care supplies, so I can get the necessary products for plant care.

**2. Manager**

* As a Manager, I want to view all the updates made by farmers without the ability to edit them, so I can track plant conditions and care for progress.
* As a Manager, I want to view all product requests submitted by farmers, so I can approve or reject these requests based on inventory and need.

**3. Store Controller**

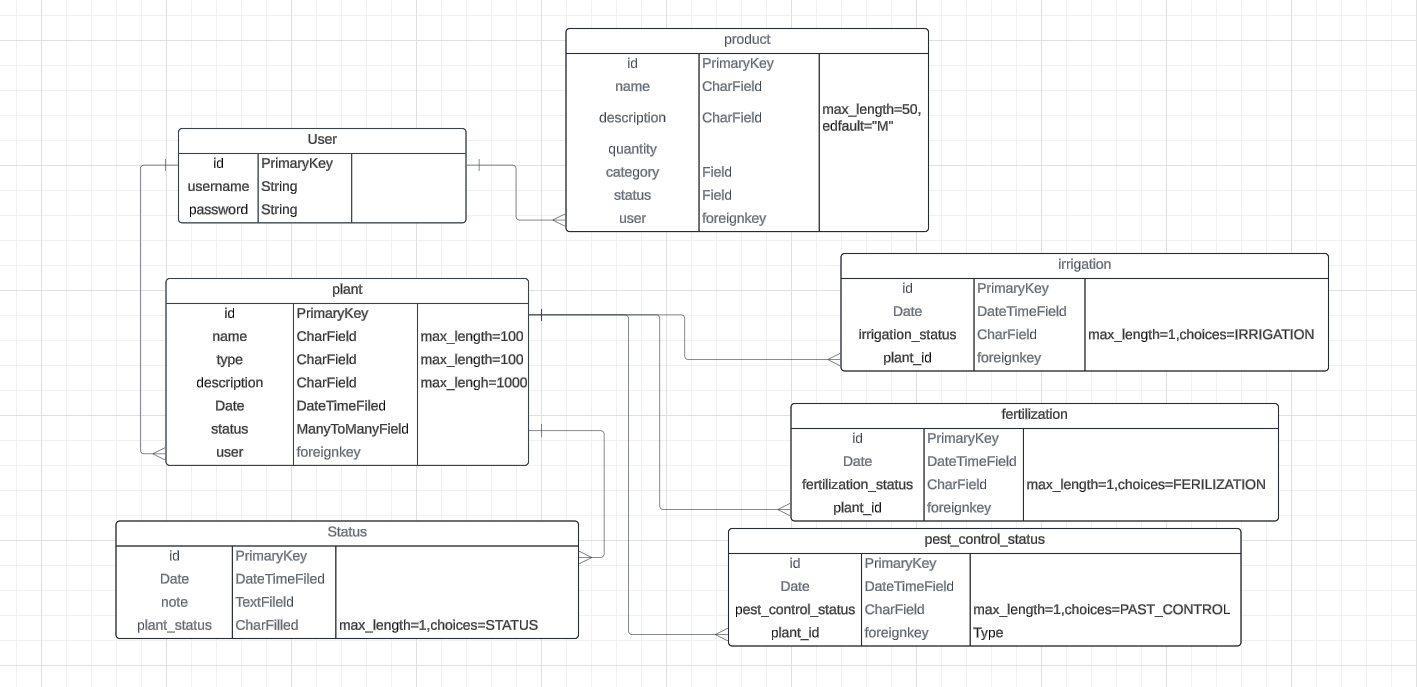
 **As a Store Controller**, I want to add new products to the inventory, including details such as name, description, price, quantity, and category, so that farmers can access a complete and up-to-date list of available products.

 **As a Store Controller**, I want to update existing product information (price, description, stock quantity) so that the product details remain accurate and reflect current availability.

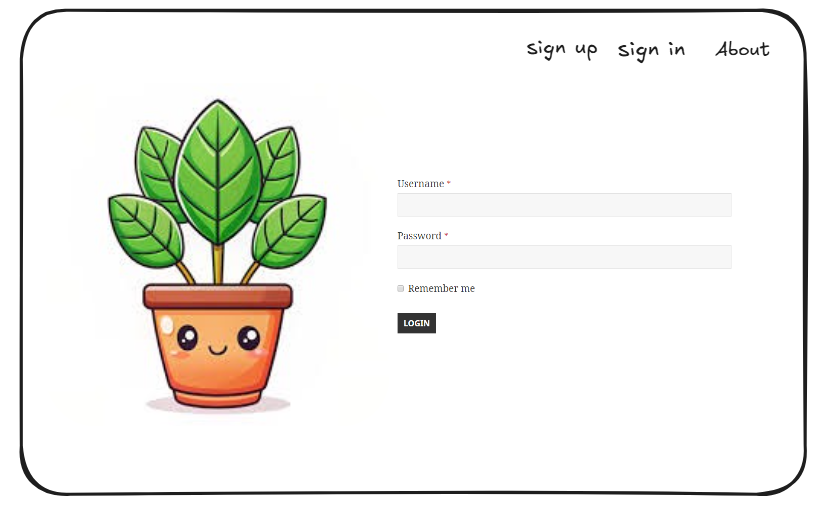
 **As a Store Controller**, I want to delete products that are no longer available from the inventory, ensuring that farmers cannot request products that are out of stock or discontinued.

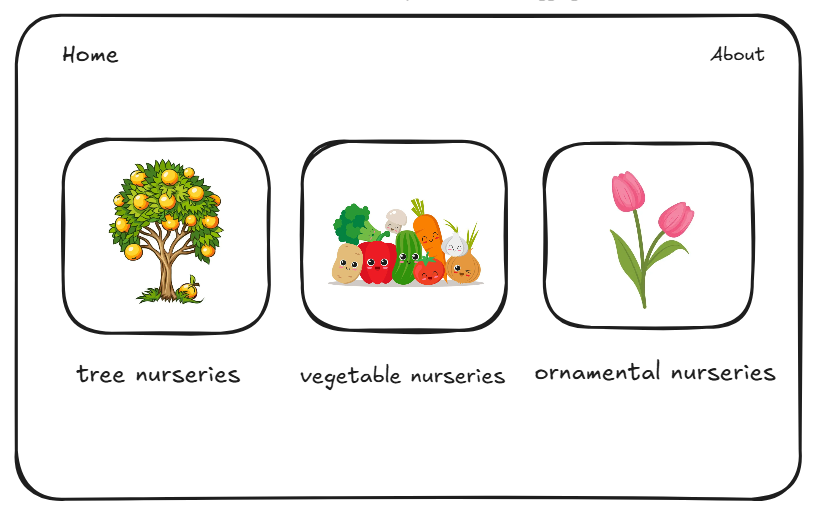
 **As a Store Controller**, I want to view a list of all available products, displaying essential details like name, price, and stock status, so that I can manage and monitor the product inventory efficiently.

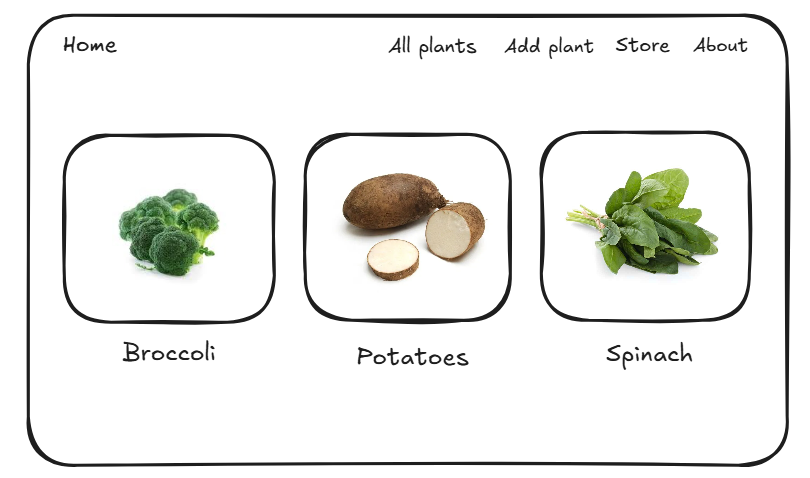
**ERD:**

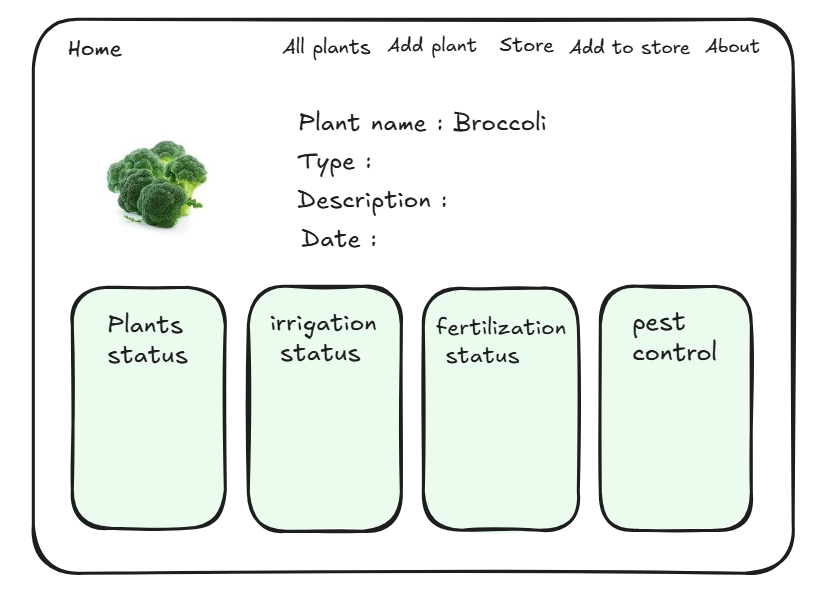
****

**Wire Frames :**

****







**Pseudo Code for Manager:**

class Manager:

def \_\_init\_\_(self, manager\_id, name):

self.manager\_id = manager\_id

self.name = name

# Function to approve or reject a product request

def approve\_or\_reject\_request(self, request\_id, action):

request = find\_product\_request\_by\_id(request\_id)

if request:

if action == "approve":

request.status = "approved"

print(f"Request for product '{request.product.name}' approved.")

elif action == "reject":

request.status = "rejected"

print(f"Request for product '{request.product.name}' rejected.")

else:

print("Invalid action. Please choose 'approve' or 'reject'.")

else:

print("Request not found.")

# Function to view all product requests made by farmers

def view\_all\_requests(self):

print("Product Requests from Farmers:")

for request in all\_requests:

print(f"Request ID: {request.request\_id}, Product: {request.product.name}, Status: {request.status}") return request

# Example usage for Manager

manager = Manager(manager\_id=1, name="Alice Smith")

manager.view\_all\_requests()

manager.approve\_or\_reject\_request(1, "approve")

**Pseudo Code for Farmer:**

class Farmer:

def \_\_init\_\_(self, farmer\_id, name):

self.farmer\_id = farmer\_id

self.name = name

self.requests = [] # List of product requests

# Function to view product details

def view\_product\_details(self, product\_id):

product = find\_product\_by\_id(product\_id)

if product:

print(f"Product Name: {product.name}")

print(f"Description: {product.description}")

print(f"Price: {product.price}")

print(f"Stock: {product.quantity}")

print(f"Category: {product.category}")

else:

print("Product not found.")

# Function to request a product (without purchase)

def request\_product(self, product\_id):

product = find\_product\_by\_id(product\_id)

if product:

self.requests.append(product)

print(f"Product '{product.name}' has been requested.")

else:

print("Product not found.")

# Function to update plant care

def update\_plant\_care(self, plant\_id, irrigation\_status, fertilization\_status, pest\_control\_status):

plant = find\_plant\_by\_id(plant\_id)

if plant:

# Update plant's care details

plant.irrigation\_status = irrigation\_status

plant.fertilization\_status = fertilization\_status

plant.pest\_control\_status = pest\_control\_status

plant.update\_time = get\_current\_time()

print(f"Plant '{plant\_id}' care has been updated.")

else:

print("Plant not found.")

# Function to view plant care updates

def view\_plant\_updates(self, plant\_id):

plant = find\_plant\_by\_id(plant\_id)

if plant:

print(f"Updates for Plant '{plant\_id}':")

for update in plant.updates:

print(f"Time: {update['time']}, Irrigation: {update['irrigation']}, Fertilization: {update['fertilization']}, Pest Control: {update['pest\_control']}")

else:

print("Plant not found.")

# Helper function to get current time (simplified)

def get\_current\_time():

from datetime import datetime

return datetime.now().strftime("%Y-%m-%d %H:%M:%S")

# Example of usage for a Farmer

farmer = Farmer(farmer\_id=1, name="John Doe")

farmer.view\_product\_details(1)

farmer.request\_product(1)

farmer.update\_plant\_care(1,

farmer.view\_plant\_updates(1)

**Pseudo Code for Store Controller:**

class Product:

def \_\_init\_\_(self, name, description, price, quantity, category):

self.name = name

self.description = description

self.price = price

self.quantity = quantity

self.category = category

self.status = "in stock" # default status

self.product\_id = generate\_product\_id()

def add\_product(name, description, price, quantity, category):

# Create a new Product instance

new\_product = Product(name, description, price, quantity, category)

# Save product to inventory (e.g., database or list)

inventory.append(new\_product)

print(f"Product '{name}' added successfully.")

# Function to update an existing product's information

def update\_product(product\_id, new\_name=None, new\_description=None, new\_price=None, new\_quantity=None):

# Search for the product in the inventory using the product\_id

product = find\_product\_by\_id(product\_id)

if product:

if new\_name:

product.name = new\_name

if new\_description:

product.description = new\_description

if new\_price:

product.price = new\_price

if new\_price:

product.price = new\_price

if new\_quantity:

product.quantity = new\_quantity

print(f"Product '{product\_id}' updated successfully.")

else:

print(f"Product with ID {product\_id} not found.")

def delete\_product(product\_id):

# Search for the product in the inventory

product = find\_product\_by\_id(product\_id)

if product:

inventory.remove(product)

print(f"Product '{product\_id}' deleted successfully.")

else:

print(f"Product with ID {product\_id} not found.")

def view\_product\_list():

print("Product List:")

for product in inventory:

print(f"ID: {product.product\_id}, Name: {product.name}, Price: {product.price}, Status: {product.status}")

def find\_product\_by\_id(product\_id):

for product in inventory:

if product.product\_id == product\_id:

return product

return None

def generate\_product\_id():

return len(inventory) + 1

inventory = []

view\_product\_list()

delete\_product(2)

view\_product\_list()