FarmVendor

A Role- Based Digital Platform for Farmer-Vendor Coordination

And Demand Forecasting and vendor optimization

Student Name : Maneesha Eeshwara

Student ID : 300392759

Course : CSIS 4495 Applied Research Project

Section : 03

GitHub Repository : https://github.com/maneeshaprs-rgb/W26_4495_S3_ManeeshaE.git

Midterm Video Demonstration :
https://drive.google.com/file/d/1UjX9kmxRbrBHhqUs3Lwbboizz_h7qboK/view?usp=sharing

Contents

1. Introduction

1.1 Background and Domain Context

Local and small-scale agricultural supply chains continue to struggle with challenges such as demand uncertainty, inventory mismatch, food waste, communication gap between producer and vendor. These challenges are basically visible in British Columbia as many farmers and buyers still depend on informal communication like phone calls, text messages or word of mouth to coordinate with their business partners. This lack of communication may result in overproduction, under-supply, and avoidable waste of perishable goods.

With the increasing adoption of digital platforms in supply chain management, there is an opportunity to develop A Role- Based Digital Platform for Farmer-Vendor Coordination and Demand Forecasting using simulated and User-Generated data. However, most of currently existing systems expensive, complex and most are mobile based applications where unsuitable for small-scale agricultural stakeholders.

1.2 Problem Definition

The main issue addressed in this research is the lack of a structured, secure, and data driven platform that enables farmers and vendors with below challenges.

- Track available and dispatched products
- Plan harvest and procurement based on historical patterns
- Communicate future demand clearly
- Support decision-making through optimization recommendations

1.3 Research Questions

Following research questions are supposed to addressed by the project:

- How can a role-based digital platform improve coordination between farmers and vendors?
- How can simulated and user generated data be used to support early-stage demand forecasting?

- How system can optimize vendor selection for farmers with the combination of distance, requested quantity and historical relationship?
- Can an optimization-based recommendation model support farmer decision-making while human choice and trust prioritized?

1.4 Literature Overview and Research Gap

Literature Overview

Research in agriculture supply chain management highlights the importance of demand forecasting, inventory visibility, and coordination between farmers and vendors. Prior studies show that even basic forecasting techniques prioritised with time and structured communication should reduce food waste, improve supply planning and increase efficiency in agricultural operations. Digital platforms are increasingly used to support these goals by enabling data sharing, record keeping and making decisions on data.

There are several farm management tools and other platforms that partially support those challenges. Here is background research and limitations on existing applications.

- **AgriWebb** : Farm management mobile application that focus on helping farmers manage production records, operational planning and inventory. offer limited direct coordination with vendors such as grocery stores and restaurants
- **FarmLogs** : largely farmer-centric mobile application with good internal visibility. limited vendor supportive.
- **Cropln** : web based Agri-intelligence and AI driven platform for enterprises across the agriculture value chain. Designed only for enterprise level Agri businesses
- **AgriDigital** : Provide cloud based grain supply chain and management platform. Use blockchain technology to secure settlement while digitize grain storage, trading, logistics and financing. Less suitable for small and medium scale farmers where developed models need extensive historical data and long-term adoption.

A key limitation across both academic literature and existing mobile applications and platforms is the high dependability on large historic datasets. While many optimization and forecasting models assume the availability of long-term, real world transactional data newly deployed systems and small – scale agricultural networks face a cold-start problem, where little or no historical data exists.

Most of systems ignore human decision making while fully automated recommendations prioritized where agriculture grow with trust, long-term relationships and farmer experience is vital in decision making and planning.

Research Gap

Below gap remains based on existing research and platforms,

- Limited availability of lightweight, role-based platforms designed specifically for local farmers and vendors.
- Insufficient support for incremental data collection, where forecasting improves gradually over time.
- Lack of optimization models that combines distance, requested quantity and historical relationships.
- Minimal focus on human-in-the-loop decision support rather than full automation.
- Overdependence on real-world, large-scale datasets that are impractical for early stage systems.

1.5 Hypotheses and Expected Benefits

Below assumptions are considered on operation of the research:

- Structured demand communication reduces uncertainty for farmers
- Simulated dataset can serve as a valid foundation for early-stage system evaluation.

Expected benefits include reduced food waste, improved planning accuracy and better decision-making support for farmers and vendors.

2. Summary Of Initially Proposed Project

The proposed research project focuses on designing and implementing a full-stack web-based platform that support role-based interaction between farmers and vendors. The system integrates simulated baseline data and user-generated interactions to support demand forecasting and vendor optimization.

This platform enables farmers to dispatch operations and inventory management while allowing vendors to submit structured demand requests. Historical demand and records of dispatches are used to generate simple forecasting insights based on weekly aggregation and trend comparison.

Additionally, the project introduces an optimization-based recommendation model that ranks potential vendor farmer matches using distance, inventory availability, requested quantity, and historical relationship metrics. The model is designed as a decision-support tool rather than an automated decision engine while ensuring that farmers retain control over final decisions.

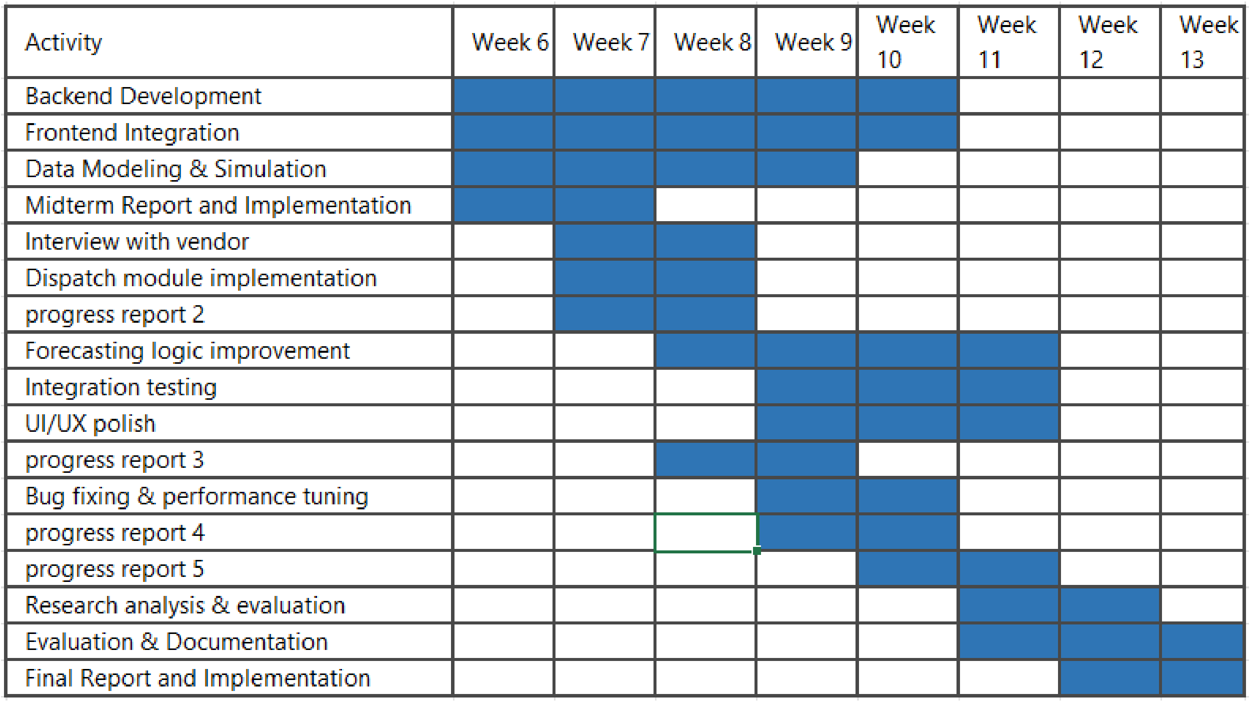
3. Changes to the Proposal

Below modifications were introduced during implementation.

- Dashboard architecture was updated from static UI to API-driven dynamic data retrieval
- Authentication was implemented using ASP.NET Identity with JWT for secure role separation
- Simulated dataset generation strategy was refined to include multi-period demand and dispatch history
- Optimization approach was simplified to a weighted scoring model to improve interpretability
- Relationship tracking entity was introduced to capture historical interactions between farmers and vendors

These changes were aligned with need for transparent and interpretable models suitable for non technical agricultural users while justifying the feasibility considerations and improved research alignment with the cold-start problem.

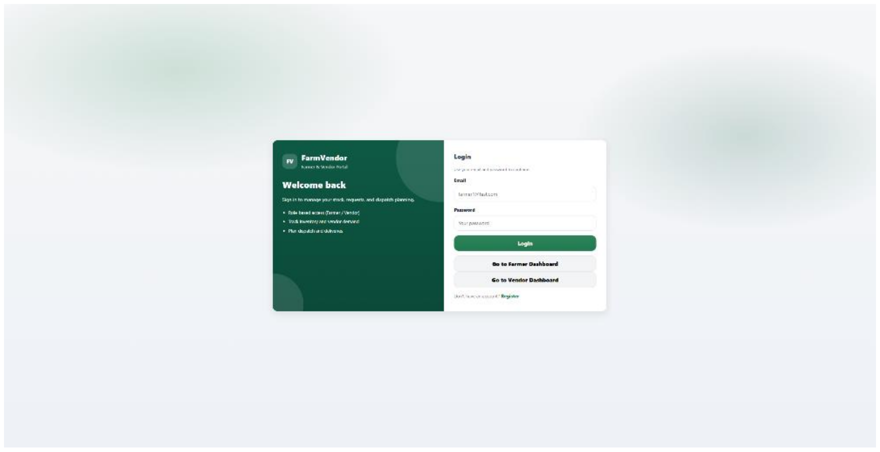
4. Project Planning and Timeline



Activity	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12	Week 13
Backend Development	🟦	🟦	🟦	🟦	🟦			
Frontend Integration	🟦	🟦	🟦	🟦	🟦			
Data Modeling & Simulation	🟦	🟦	🟦	🟦				
Midterm Report and Implementation	🟦	🟦						
Interview with vendor		🟦	🟦					
Dispatch module implementation		🟦	🟦					
progress report 2		🟦	🟦					
Forecasting logic improvement			🟦	🟦	🟦	🟦		
Integration testing				🟦	🟦	🟦		
UI/UX polish				🟦	🟦	🟦		
progress report 3			🟦	🟦				
Bug fixing & performance tuning				🟦	🟦			
progress report 4				🟦	🟦			
progress report 5					🟦	🟦		
Research analysis & evaluation						🟦	🟦	
Evaluation & Documentation						🟦	🟦	🟦
Final Report and Implementation							🟦	🟦

5. Implemented Features

1. **Role Based Authentication(registration & Login)**
    - This feature allows users to login in roles as either Farmer or Vendor. Authentication is implemented using JWT with ASP.NET Identity, which ensure secure access and role based navigation.
    - Login page



FV FarmVendor
Farmer & Vendor Portal

Welcome back

Sign in to manage your stock, requests, and dispatch planning.

- Role based access (Farmer / Vendor)
- Track inventory and vendor demand
- Plan dispatch and deliveries

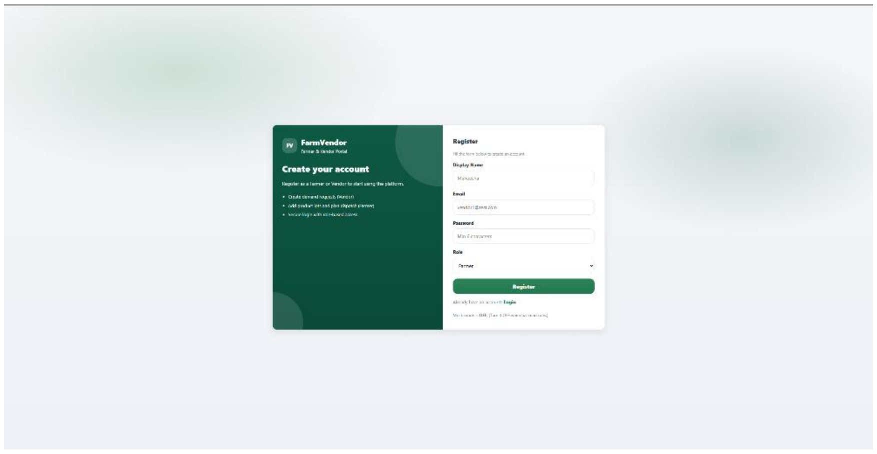
Login

[illegible]

Email: farmer1@test.com

Password: Your password

- Register page



FV FarmVendor
Farmer & Vendor Portal

Create your account

Register as a Farmer or Vendor to start using the platform.

- Create demand requests (Vendor)
- Add product lots and plan dispatch (Farmer)
- Secure login with role-based access

Register

Fill the form below to create an account.

Display Name: [illegible]

Email: vendor1@test.com

Password: Min 6 characters

Role: Farmer

[illegible]

- The design consist of :
    - Registration page with role selection

        - Login page with credential validation
        - Backend JWT token generation
        - Role-based route protection using React guards

    - Implementation

Frontend

- Register.jsx : handles role selection and API call
- Login.jsx : stores JWT and redirects dashboard
- RequireAuth.jsx : protects routes
- RequireRole.jsx : restricts role access

Backend

- AuthController : login and register endpoints
- ApplicationUser : extended Identity user
- JWT token generation logic

Database

- AspNetUsers table stores user accounts
- Roles mapped using Identity role tables

2. **Farmer Inventory management**

   Farmers can add and manage product inventory lots with expiry tracking

   This feature includes:

   - Add product lot form
   - Inventory table view
   - Expiry status badge logic

   Implementation

   Frontend

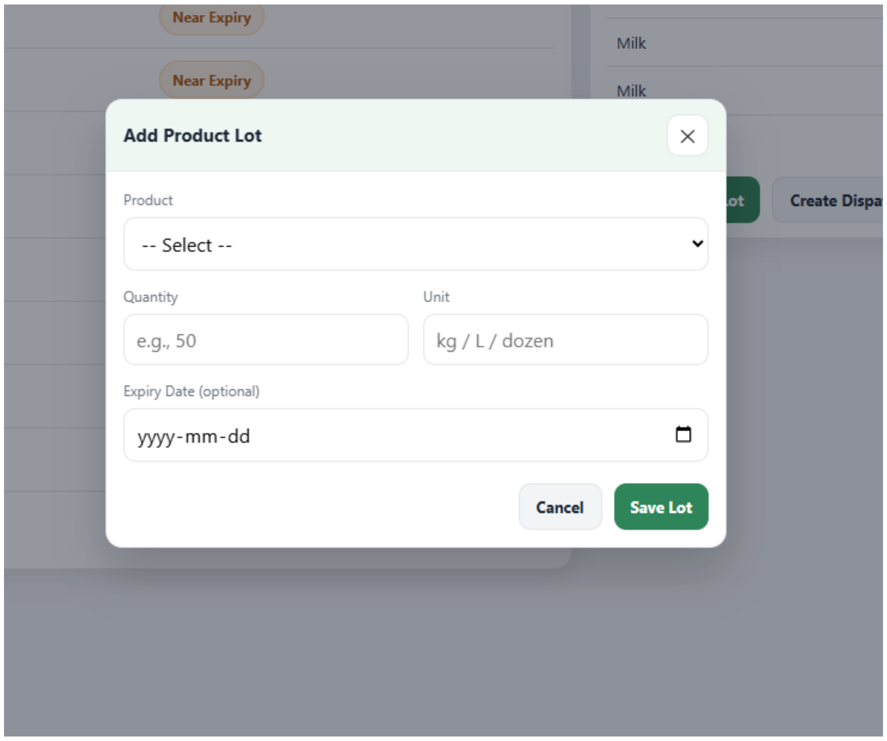
- FarmerDashboard.jsx
- Add Product Lot modal
- Inventory table UI

Backend

- InventoryLotsController
- CreateInventoryLotDto
- InventoryLot entity

Database

- InventoryLot table storing:
    - FarmerId
    - ProductId
    - Quantity
    - ExpiryDate



Add Product Lot

Product

-- Select --

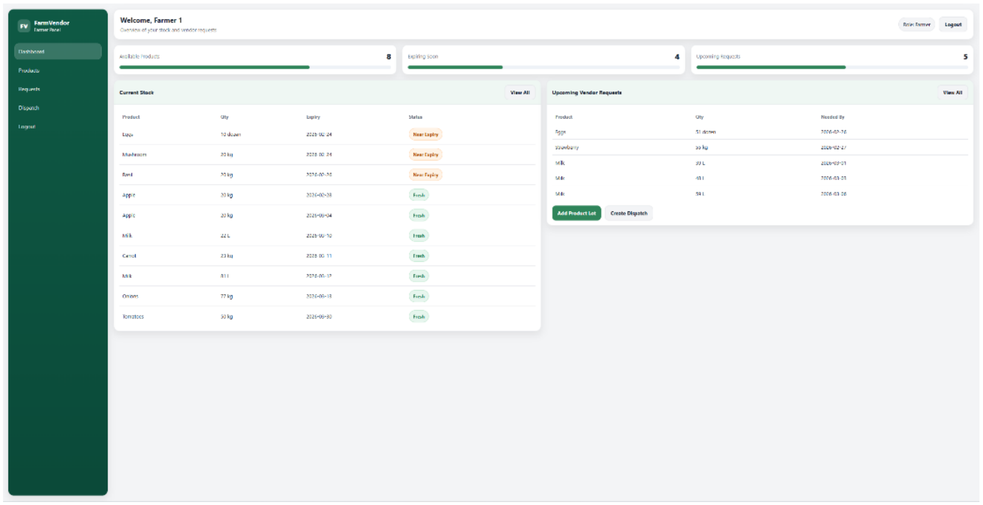
Quantity | Unit

e.g., 50 | kg / L / dozen

Expiry Date (optional)

yyyy-mm-dd

Cancel Save Lot



Welcome, Farmer 1

Overview of your stock and vendor requests

Available Products 8

Expiring Soon 4

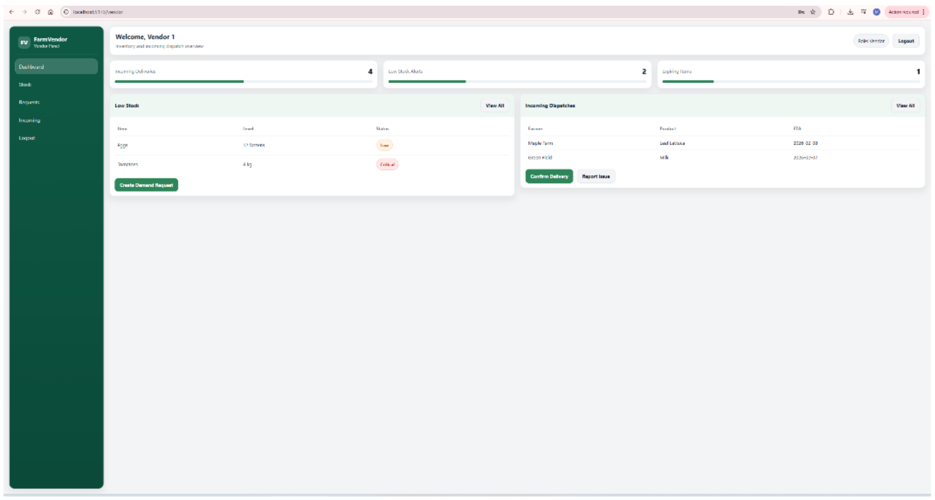
Upcoming Requests 5

Current Stock

Product	Qty	Expiry	Status
Eggs	10 dozen	2026-02-24	Near Expiry
Mushroom	20 kg	2026-02-24	Near Expiry
Beet	20 kg	2026-02-26	Near Expiry
Apple	20 kg	2026-02-28	Fresh
Apple	20 kg	2026-03-04	Fresh
Milk	22 L	2026-03-10	Fresh
Carrot	23 kg	2026-03-11	Fresh
Milk	81 L	2026-03-12	Fresh
Onions	77 kg	2026-03-13	Fresh
Tomatoes	50 kg	2026-03-30	Fresh

Upcoming Vendor Requests

Product	Qty	Needed By
Eggs	51 dozen	2026-02-26
Strawberry	55 kg	2026-02-27
Milk	30 L	2026-03-01
Milk	49 L	2026-03-03
Milk	59 L	2026-03-08



Welcome, Vendor 1

Inventory and incoming dispatch overview

Incoming Deliveries 4

Low Stock Alerts 2

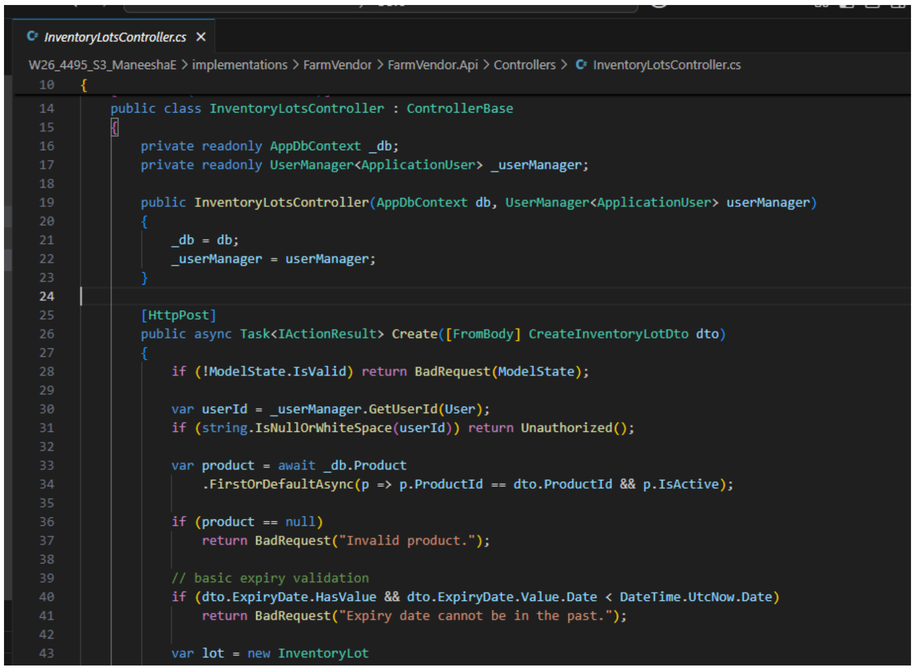
Expiring Items 1

Low Stock

Item	Level	Status
Eggs	12 Dozens	Low
Tomatoes	8 kg	Critical

Incoming Dispatches

Farmer	Product	ETA
Maple Farm	Leaf Lettuce	2026-02-03
Green Field	Milk	2026-02-07


InventoryLotsController.cs

W26_4495_S3_ManeeshaE > implementations > FarmVendor > FarmVendor.Api > Controllers > InventoryLotsController.cs

```
{
    public class InventoryLotsController : ControllerBase
    {
        private readonly AppDbContext _db;
        private readonly UserManager<ApplicationUser> _userManager;

        public InventoryLotsController(AppDbContext db, UserManager<ApplicationUser> userManager)
        {
            _db = db;
            _userManager = userManager;
        }

        [HttpPost]
        public async Task<IActionResult> Create([FromBody] CreateInventoryLotDto dto)
        {
            if (!ModelState.IsValid) return BadRequest(ModelState);

            var userId = _userManager.GetUserId(User);
            if (string.IsNullOrWhiteSpace(userId)) return Unauthorized();

            var product = await _db.Product
                .FirstOrDefaultAsync(p => p.ProductId == dto.ProductId && p.IsActive);

            if (product == null)
                return BadRequest("Invalid product.");

            // basic expiry validation
            if (dto.ExpiryDate.HasValue && dto.ExpiryDate.Value.Date < DateTime.UtcNow.Date)
                return BadRequest("Expiry date cannot be in the past.");

            var lot = new InventoryLot
```

3. Vendor Demand Request System

Vendors can create product demand requests when stock is low

This feature includes :

- Low stock alert table
- Demand request creation
- Status lifecycle tracking

Implementation

Frontend

- VendorDashboard.jsx
- Create demand request button
- Low stock alert UI

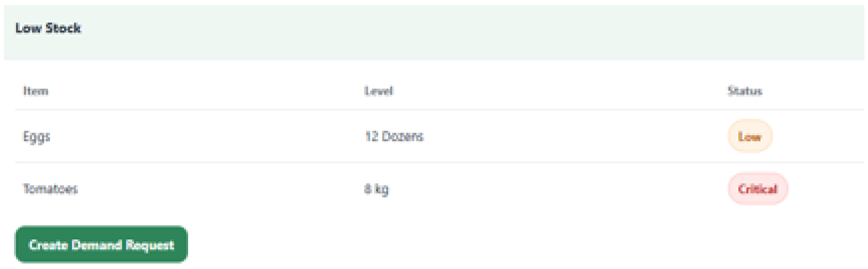
Backend

- DemandRequest entity
- DemandRequest APIs

- Status management

Database

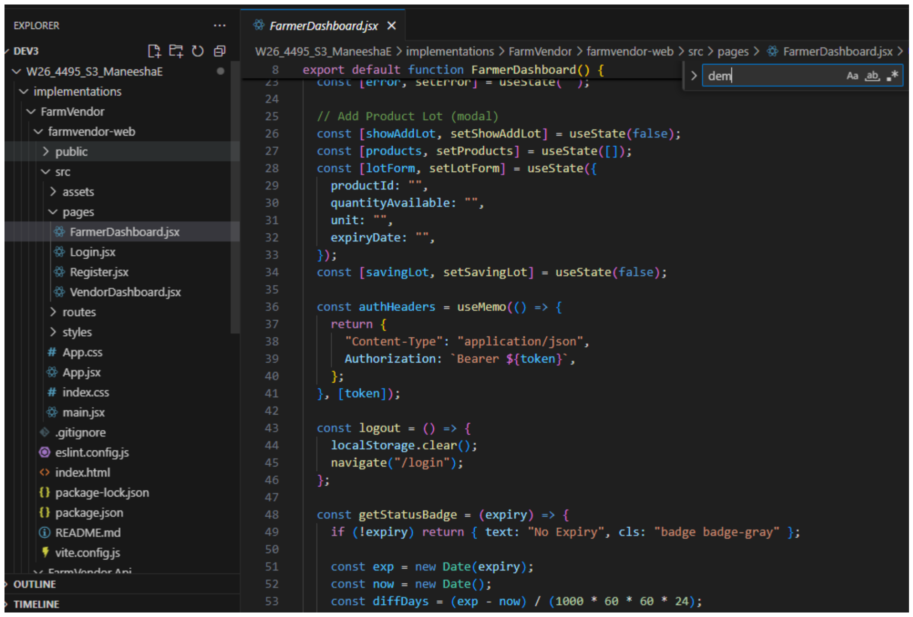
- DemandRequest table storing:
    - VendorId
    - ProductId
    - QuantityRequested
    - Status



Low Stock

Item	Level	Status
Eggs	12 Dozens	Low
Tomatoes	8 kg	Critical

Create Demand Request



```
EXPLORER
DEV3
W26_4495_S3_ManeeshaE
  implementations
    FarmVendor
      farmvendor-web
        public
        src
          assets
          pages
            FarmerDashboard.jsx
            Login.jsx
            Register.jsx
            VendorDashboard.jsx
          routes
          styles
          App.css
          App.jsx
          index.css
          main.jsx
        .gitignore
        eslint.config.js
        index.html
        package-lock.json
        package.json
        README.md
        vite.config.js
      FarmVendor.Api
OUTLINE
TIMELINE
```

FarmerDashboard.jsx

W26_4495_S3_ManeeshaE > implementations > FarmVendor > farmvendor-web > src > pages > FarmerDashboard.jsx >

dem

```jsx
export default function FarmerDashboard() {
  const [error, setError] = useState("");

  // Add Product Lot (modal)
  const [showAddLot, setShowAddLot] = useState(false);
  const [products, setProducts] = useState([]);
  const [lotForm, setLotForm] = useState({
    productId: "",
    quantityAvailable: "",
    unit: "",
    expiryDate: "",
  });
  const [savingLot, setSavingLot] = useState(false);

  const authHeaders = useMemo(() => {
    return {
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    };
  }, [token]);

  const logout = () => {
    localStorage.clear();
    navigate("/login");
  };

  const getStatusBadge = (expiry) => {
    if (!expiry) return { text: "No Expiry", cls: "badge badge-gray" };

    const exp = new Date(expiry);
    const now = new Date();
    const diffDays = (exp - now) / (1000 * 60 * 60 * 24);
```

4. Dispatch Management Module

This feature enables farmers to create dispatches and vendors to confirm deliveries.

This design includes :

- Create dispatch button
- Incoming dispatch table
- Delivery confirmation
- Dispatch status tracking

Implementation

Frontend

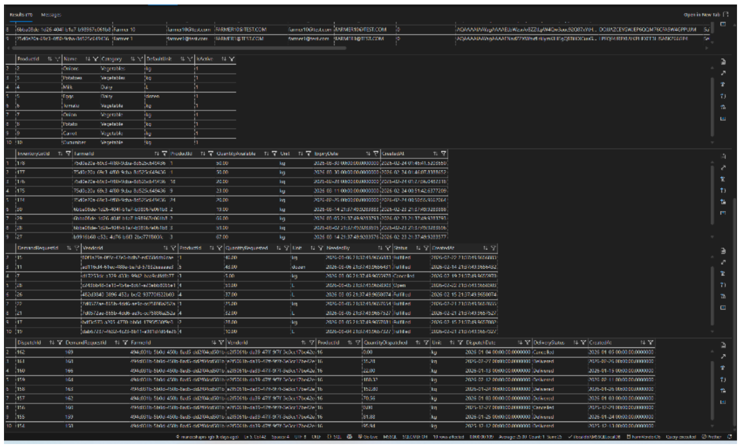
- Create dispatch action in FarmerDashboard
- Incoming dispatch list in VendorDashboard
- Confirm delivery button

Backend

- Dispatch entity
- DispatchController
- Delivery status logic

Database

- Dispatch table storing:
    - FarmerId
    - VendorId
    - ProductId
    - Quantity
    - DispatchDate
    - DeliveryStatus



Results (7) Messages

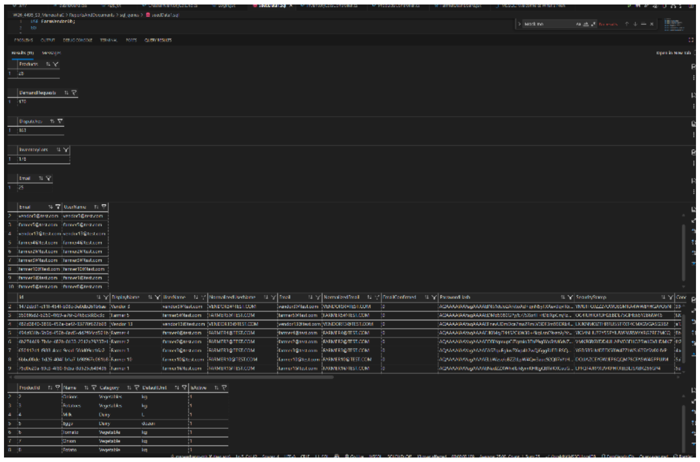
8	6bba08de-1d26-404f-b1a7-b98967c061b8	Farmer 10	farmer10@test.com	FARMER10@TEST.COM	farmer10@test.com	FARMER10@TEST.COM	0
9	75d0e20a-69c1-4f80-9cba-8d525cb49436	Farmer 1	farmer1@test.com	FARMER1@TEST.COM	farmer1@test.com	FARMER1@TEST.COM	0

	ProductId	Name	Category	DefaultUnit	IsActive
2	2	Onions	Vegetables	kg	1
3	3	Potatoes	Vegetables	kg	1
4	4	Milk	Dairy	L	1
5	5	Eggs	Dairy	dozen	1
6	6	Tomato	Vegetable	kg	1
7	7	Onion	Vegetable	kg	1
8	8	Potato	Vegetable	kg	1
9	9	Carrot	Vegetable	kg	1
10	10	Cucumber	Vegetable	kg	1

	InventoryLotId	FarmerId	ProductId	QuantityAvailable	Unit	ExpiryDate	CreatedAt
1	178	75d0e20a-69c1-4f80-9cba-8d525cb49436	1	50.00	kg	2026-03-30 00:00:00.0000000	2026-02-24 01:46:41.5203550
2	177	75d0e20a-69c1-4f80-9cba-8d525cb49436	1	50.00	kg	2026-03-30 00:00:00.0000000	2026-02-24 01:46:07.8388652
3	176	75d0e20a-69c1-4f80-9cba-8d525cb49436	18	20.00	kg	2026-03-28 00:00:00.0000000	2026-02-24 01:27:56.0482518
4	175	75d0e20a-69c1-4f80-9cba-8d525cb49436	9	23.00	kg	2026-03-11 00:00:00.0000000	2026-02-24 00:51:42.6377209
5	174	75d0e20a-69c1-4f80-9cba-8d525cb49436	24	20.00	kg	2026-02-26 00:00:00.0000000	2026-02-24 00:50:55.9567064
6	30	6bba08de-1d26-404f-b1a7-b98967c061b8	2	19.00	kg	2026-03-14 21:37:49.9283888	2026-02-23 21:37:49.9283888
7	29	6bba08de-1d26-404f-b1a7-b98967c061b8	2	66.00	kg	2026-03-05 21:37:49.9281791	2026-02-23 21:37:49.9281793
8	28	6bba08de-1d26-404f-b1a7-b98967c061b8	3	59.00	kg	2026-03-02 21:37:49.9283695	2026-02-23 21:37:49.9283696
9	27	b9916b68-c52c-4d76-b6f3-2bc771f803fc	3	67.00	kg	2026-03-14 21:37:49.9283576	2026-02-23 21:37:49.9283577

	DemandRequestId	VendorId	ProductId	QuantityRequested	Unit	NeededBy	Status	CreatedAt
2	15	50f1a79e-0f7c-47e3-bdb2-ed358dcb6cae	1	40.00	kg	2026-03-06 21:37:49.9656881	Fulfilled	2026-02-22 21:37:49.9656883
3	11	ad116d4-61ec-488e-be7d-57852aaaaad3	5	40.00	dozen	2026-03-06 21:37:49.9656431	Fulfilled	2026-02-14 21:37:49.9656432
4	7	d17253dc-c129-433b-9942-bce9cd4fb77	1	5.00	kg	2026-03-06 21:37:49.9653970	Cancelled	2026-02-19 21:37:49.9653970
5	28	c243bb48-0a15-4b4e-8cb1-e25ebb8065e1	4	50.00	L	2026-03-05 21:37:49.9658303	Open	2026-02-22 21:37:49.9658303
6	26	482d3840-3896-452a-bef2-93770f622b03	4	37.00	L	2026-03-05 21:37:49.9658074	Fulfilled	2026-02-15 21:37:49.9658074
7	22	7d0572ae-855b-4dd6-ae9c-cc75888a252a	1	25.00	kg	2026-03-05 21:37:49.9657654	Fulfilled	2026-02-21 21:37:49.9657655
8	21	7d0572ae-855b-4dd6-ae9c-cc75888a252a	4	32.00	L	2026-03-05 21:37:49.9657527	Fulfilled	2026-02-21 21:37:49.9657527
9	17	bdf5c573-a205-4770-bb0d-1793530f9e3	2	28.00	kg	2026-03-05 21:37:49.9657001	Fulfilled	2026-02-15 21:37:49.9657002
10	19	dab67257-4632-4c25-8b11-a381d7d64e2b	4	10.00	L	2026-03-04 21:37:49.9657327	Fulfilled	2026-02-21 21:37:49.9657327

	DispatchId	DemandRequestId	FarmerId	VendorId	ProductId	QuantityDispatched	Unit	DispatchDate	DeliveryStatus	CreatedAt
2	162	169	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	0.00	kg	2026-01-04 00:00:00.0000000	Cancelled	2026-01-05 00:00:00.0000000
3	161	168	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	35.20	kg	2026-02-27 00:00:00.0000000	Delivered	2026-02-26 00:00:00.0000000
4	160	166	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	22.00	kg	2026-01-13 00:00:00.0000000	Delivered	2026-01-15 00:00:00.0000000
5	159	164	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	160.32	kg	2026-02-12 00:00:00.0000000	Delivered	2026-02-11 00:00:00.0000000
6	158	163	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	152.80	kg	2026-01-24 00:00:00.0000000	Delivered	2026-01-26 00:00:00.0000000
7	157	162	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	70.56	kg	2026-01-03 00:00:00.0000000	Delivered	2026-01-03 00:00:00.0000000
8	156	160	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	0.00	kg	2025-12-27 00:00:00.0000000	Cancelled	2025-12-29 00:00:00.0000000
9	155	159	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	81.88	kg	2026-01-25 00:00:00.0000000	Delivered	2026-01-24 00:00:00.0000000
10	154	158	494d031b-5b0d-450b-8ad5-dd2f04cd501b	e2f5061b-da39-47ff-9f7f-3e3cc17be42e	16	95.94	kg	2025-12-12 00:00:00.0000000	Delivered	2025-12-13 00:00:00.0000000



W26_4495_S3_ManeeshaE > ReportsAndDocuments > sql_queries > seedData1.sql

```
USE FarmVendorDb;
GO
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS

Results (11) Messages

	Products
1	28

	DemandRequests
1	170

	Dispatches
1	163

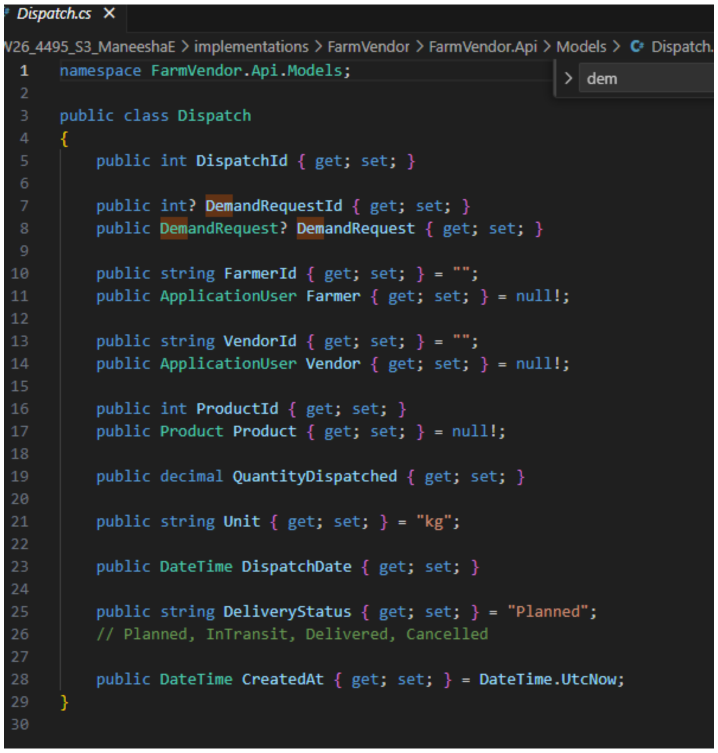
	InventoryLots
1	178

	Email
1	25

	Email	UserName
2	vendor1@test.com	vendor1@test.com
3	farmer5@test.com	farmer5@test.com
4	vendor13@test.com	vendor13@test.com
5	farmer4@test.com	farmer4@test.com
6	farmer2@test.com	farmer2@test.com
7	farmer3@test.com	farmer3@test.com
8	farmer10@test.com	farmer10@test.com
9	farmer1@test.com	farmer1@test.com
10	farmer6@test.com	farmer6@test.com

	Id	DisplayName	UserName	NormalizedUserName	Email	NormalizedEmail	EmailConfirmed	PasswordHash	SecurityStamp
2	1472cb51-c111-4548-b58c-3a0db261b6ae	Vendor 5	vendor5@test.com	VENDOR5@TEST.COM	vendor5@test.com	VENDOR5@TEST.COM	0	[illegible]	[illegible]
3	3508f6d2-6250-4f89-a76f-2f6bc5885cdc	Farmer 5	farmer5@test.com	FARMER5@TEST.COM	farmer5@test.com	FARMER5@TEST.COM	0	[illegible]	[illegible]
4	482d3840-3896-452a-bef2-93770f622b03	Vendor 13	vendor13@test.com	VENDOR13@TEST.COM	vendor13@test.com	VENDOR13@TEST.COM	0	[illegible]	[illegible]
5	494d031b-5b0d-450b-8ad5-dd2f04cd501b	Farmer 4	farmer4@test.com	FARMER4@TEST.COM	farmer4@test.com	FARMER4@TEST.COM	0	[illegible]	[illegible]
6	4b274469-7b4e-482b-0123-2342a2923f2e	Farmer 2	farmer2@test.com	FARMER2@TEST.COM	farmer2@test.com	FARMER2@TEST.COM	0	[illegible]	[illegible]
7	65011241-f800-4ccc-9ead-5fdd09cc1633	Farmer 3	farmer3@test.com	FARMER3@TEST.COM	farmer3@test.com	FARMER3@TEST.COM	0	[illegible]	[illegible]
8	6bba08de-1d26-404f-b1a7-b98967c061b8	Farmer 10	farmer10@test.com	FARMER10@TEST.COM	farmer10@test.com	FARMER10@TEST.COM	0	[illegible]	[illegible]
9	75d0e20a-69c1-4f80-9cba-8d525cb49436	Farmer 1	farmer1@test.com	FARMER1@TEST.COM	farmer1@test.com	FARMER1@TEST.COM	0	[illegible]	[illegible]

	ProductId	Name	Category	DefaultUnit	IsActive
2	2	Onions	Vegetables	kg	1
3	3	Potatoes	Vegetables	kg	1
4	4	Milk	Dairy	L	1
5	5	Eggs	Dairy	dozen	1
6	6	Tomato	Vegetable	kg	1
7	7	Onion	Vegetable	kg	1
8	8	Potato	Vegetable	kg	1



Dispatch.cs

W26_4495_S3_ManeeshaE > implementations > FarmVendor > FarmVendor.Api > Models > Dispatch.

```csharp
namespace FarmVendor.Api.Models;

public class Dispatch
{
    public int DispatchId { get; set; }

    public int? DemandRequestId { get; set; }
    public DemandRequest? DemandRequest { get; set; }

    public string FarmerId { get; set; } = "";
    public ApplicationUser Farmer { get; set; } = null!;

    public string VendorId { get; set; } = "";
    public ApplicationUser Vendor { get; set; } = null!;

    public int ProductId { get; set; }
    public Product Product { get; set; } = null!;

    public decimal QuantityDispatched { get; set; }

    public string Unit { get; set; } = "kg";

    public DateTime DispatchDate { get; set; }

    public string DeliveryStatus { get; set; } = "Planned";
    // Planned, InTransit, Delivered, Cancelled

    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;
}
```

5. **Dashboard Analytics & Status Badge System**

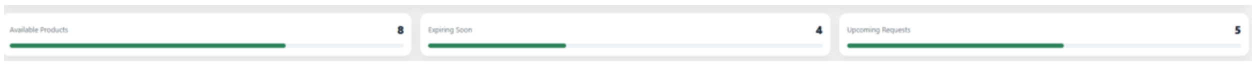
   Dashboard Analytics & Status Badge System

   Design Includes:

   - Analytics summary cards
   - Expiry status badges
   - Low stock alerts

   Logic implemented using:

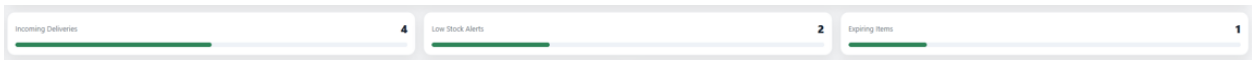
   - Expiry date difference calculation
   - Quantity threshold rules
   - React dynamic rendering



Available Products 8

Expiring Soon 4

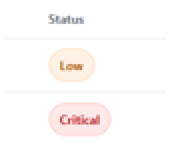
Upcoming Requests 5



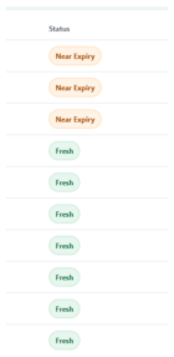
Incoming Deliveries 4

Low Stock Alerts 2

Expiring Items 1

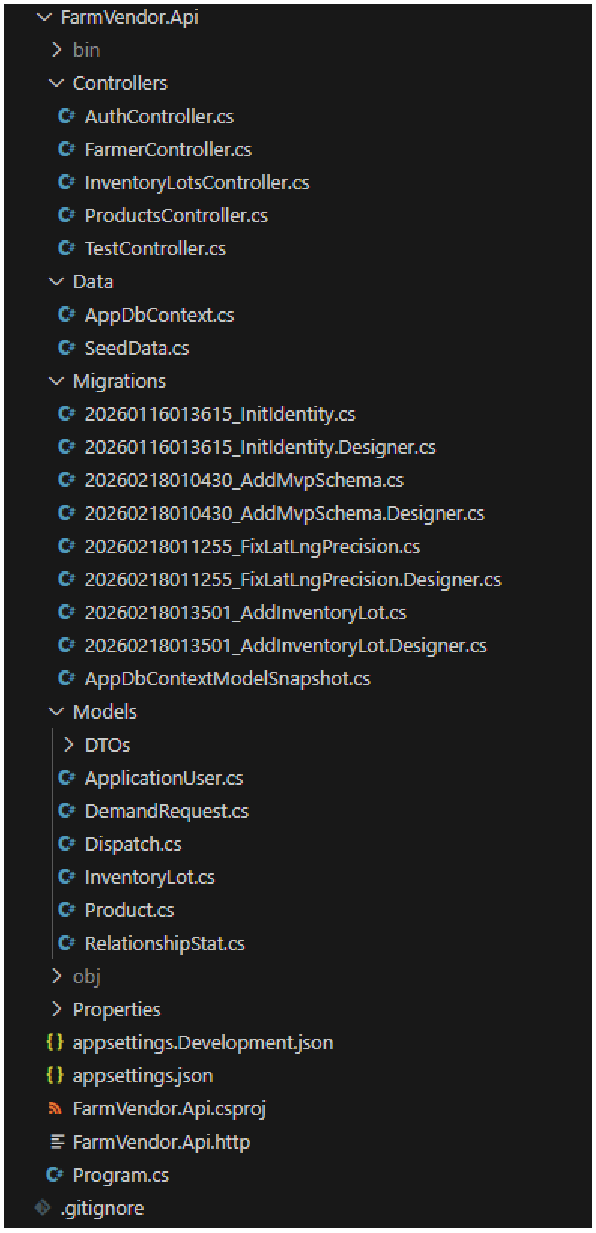


Status
Low
Critical

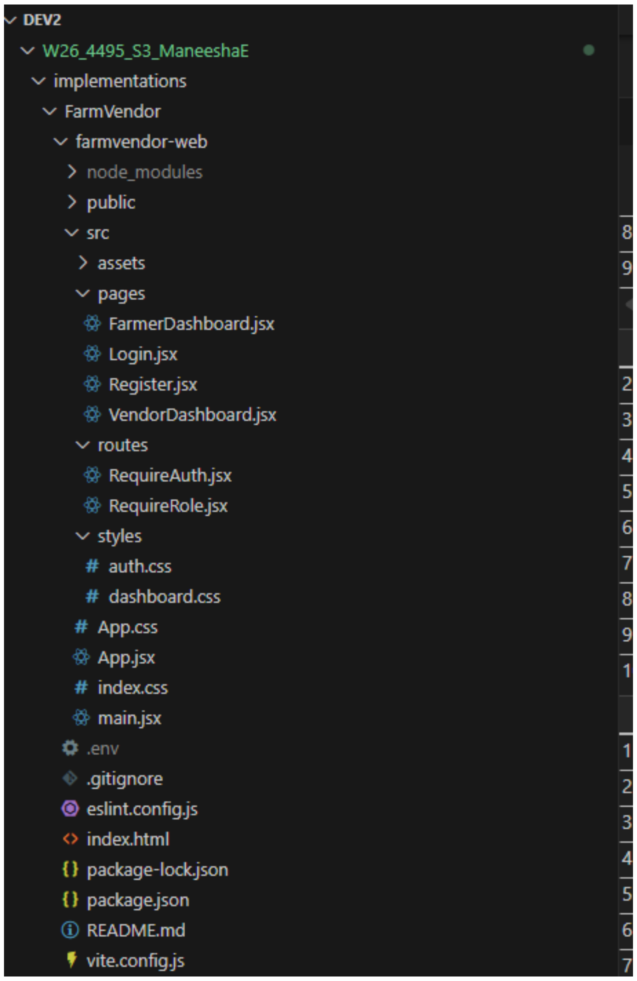


Status
Near Expiry
Near Expiry
Near Expiry
Fresh
Fresh
Fresh
Fresh
Fresh
Fresh
Fresh

- Backend and front end Architecture



```
FarmVendor.Api
  bin
  Controllers
    AuthController.cs
    FarmerController.cs
    InventoryLotsController.cs
    ProductsController.cs
    TestController.cs
  Data
    AppDbContext.cs
    SeedData.cs
  Migrations
    20260116013615_InitIdentity.cs
    20260116013615_InitIdentity.Designer.cs
    20260218010430_AddMvpSchema.cs
    20260218010430_AddMvpSchema.Designer.cs
    20260218011255_FixLatLngPrecision.cs
    20260218011255_FixLatLngPrecision.Designer.cs
    20260218013501_AddInventoryLot.cs
    20260218013501_AddInventoryLot.Designer.cs
    AppDbContextModelSnapshot.cs
  Models
    DTOs
    ApplicationUser.cs
    DemandRequest.cs
    Dispatch.cs
    InventoryLot.cs
    Product.cs
    RelationshipStat.cs
  obj
  Properties
  appsettings.Development.json
  appsettings.json
  FarmVendor.Api.csproj
  FarmVendor.Api.http
  Program.cs
.gitignore
```



```
DEV2
  W26_4495_S3_ManeeshaE
    implementations
      FarmVendor
        farmvendor-web
          node_modules
          public
          src
            assets
            pages
              FarmerDashboard.jsx
              Login.jsx
              Register.jsx
              VendorDashboard.jsx
            routes
              RequireAuth.jsx
              RequireRole.jsx
            styles
              auth.css
              dashboard.css
            App.css
            App.jsx
            index.css
            main.jsx
          .env
          .gitignore
          eslint.config.js
          index.html
          package-lock.json
          package.json
          README.md
          vite.config.js
```

6. AI Use section

AI Tool Name	Version, Account Type	Specific feature for which the AI tool was used	Value Addition What value did you add over and above what AI did for you?
ChatGPT	Free version	Research on existing tools for farm vendor management	Critical revie while refinement of idea while focusing missing features from existing tools
		System backend design guidance	Developed my own backend activities
		Worklog, documentation and report structuring	Rewrote and refined documentation with personal implementation details and validation
		Rewrote and refined documentation with personal implementation details and validation	Research brainstorming for forecasting and optimization module
		Designed own research novelty including optimization and demand prediction logic	Designed custom dashboards, styling, routing and responsiveness independently

Appendix : Ai prompt history

Prompts and responses used for system design, refinement on ideas will be included in the appendix.

- Provide examples of existing apps for farmer vendor management systems

- Provide steps to develop ASP .NET project backend for role-based website.
- Provide relevant dashboard ui to be included in a supply chain management system

7. Work Log

Date	No of Hours	Description of work done
13-Jan	2	Finalised research idea for FarmVendor application by defining the core problem and scope
	1	Installed development tools: .NET SDK, VS Code, SQL Server LocalDB, Node.js (React)
14-Jan	4	Created backend ASP.NET Core Web API and tested in browser by resolving installation dependency issues
15-Jan	4	Integrated ASP.NET Core Web API (.NET 8.0)
		Configured SQL Server LocalDB (FarmVendorDb)
		Implemented ASP.NET Identity
		Defined roles: Farmer and Vendor
		Implemented JWT authentication
17-Jan	2	Tested working endpoints on Postman (POST /Api/auth/register, POST /Api/auth/login)
20-Jan	0.5	Project idea re-evaluation with Professor (Madam Priya) and refined research idea
21-Jan	2	Initial research on existing mobile and web-based applications and background research
22-Jan	2	Initial research proposal draft development
24-Jan	2	Research on React integration with ASP.NET backend
	2	Continued research proposal development
25-Jan	2	Continued research proposal development
26-Jan	1	Finalised research proposal and submission
27-Jan	3	Reviewed proposal with professor and applied modifications to data collection strategy
29-Jan	4	Resolved JWT authentication and protected API access issues
		Corrected Postman authorization configuration
2-Feb	2	Moved .gitignore to repository root and cleaned tracked artifacts
4-Feb	3	Created frontend sketches and planned React component structure
		Installed necessary React dependencies
5-Feb	2	Resolved Node.js and npm installation issues

8-Feb	4	Implemented Login.jsx, Register.jsx, FarmerDashboard.jsx, App.jsx and dashboard styling
		Modified routing in App.jsx
9-Feb	5	Implemented VendorDashboard.jsx and updated routing
		Uploaded modified proposal to GitHub
		Updated worklog for progress report 1
		Verified repository contents
10-Feb	5	Fixed UI alignment issues and corrected auth.css padding problems
		Updated login and registration with 2-panel interface
12-Feb	4	Connected frontend login with backend authentication
		Created SQL Server LocalDB database using migrations
15-Feb	5	Resolved Vite environment loading issues
		Fixed registration and login user errors
17-Feb	6	Fixed frontend registration issue caused by mock mode configuration
		Completed role-based authentication, dashboards, DB persistence, JWT and protected routing
		Fixed RelationshipStats FK constraint issue and recreated database
		Created MVP schema: Product, DemandRequest, Dispatch, RelationshipStat
		Implemented InventoryLot table
		Added real DB data to dashboards
		Updated FarmerDashboard with real DB data
18-Feb	6	Planned next phase tasks: data generation, forecasting module, optimization model, relationship scoring and evaluation
23-Feb	8	Implemented Add Product Lot feature (React form + API POST)
		Created API endpoint POST /api/farmer/inventorylots using CreateInventoryLotDto and InventoryLotsController
		Final Midterm video recording
		Final Midterm report completion and submission

8. Closing and References

I would like to thank our Professor, Padmapriya Arasanipalai Kandhadai, the institution, Douglas College for the opportunity to design and present this project.

AgriWebb. (2024). *AgriWebb farm management software* [Mobile application]. https://www.agriwebb.com

FarmLogs. (2024). *FarmLogs: Farm management and record keeping* [Mobile application]. https://farmlogs.com

CropIn Technology Solutions. (2024). *CropIn smart farm management platform* [Mobile application]. https://www.cropin.com

AgriDigital. (2024). *AgriDigital grain supply chain platform* [Web-based platform]. https://www.agridigital.io

This will be updated accordingly as well after the project is done