DIGITAL EGYPT PIONEERS INITIATIVE

REACT FRONT-END DEVELOPMENT

Home Grown Stores

Supervised by:

Eng. Basma Abdel Halim

Team members:

Eman Saber

Habiba Mohamed

Menna Safwat

Rawda Fakhry

Abdelrahman Mohamed Mukhtar

Nadeem Hassan

Acknowledgement

Invaluable guidance, insightful mentorship, and unwavering support throughout the course of this project. Her constructive feedback and expert advice have been instrumental in shaping not only the technical aspects of our work but also in enhancing our organizational abilities, problem-solving skills, and teamwork capabilities.

Throughout this journey, Eng. Basma has consistently encouraged us to push our boundaries, explore innovative solutions, and maintain a high standard of quality in every task we undertook. Her dedication to teaching and her patience in clarifying complex concepts have created a supportive learning environment where we felt confident to experiment, make mistakes, and learn from them.

We also wish to express our sincere appreciation to the Digital Egypt Pioneers Initiative for providing us with this exceptional opportunity to gain practical experience in full-stack web development and collaborative project work. The initiative has allowed us to apply theoretical knowledge in real-world scenarios, strengthening our understanding of the development process and preparing us for future professional challenges.

This project would not have reached its current level of completion and quality without the guidance, encouragement, and inspiration provided by Eng. Basma Abdel Halim. We are truly grateful for her continuous efforts, for believing in our potential, and for inspiring us to achieve excellence. Her mentorship has left a lasting and meaningful impact on our personal and professional growth, and we will always value the knowledge and insights she has shared with us throughout this journey.

Abstract

The rapid rise in popularity of online marketplaces has opened significant opportunities for small vendors, local artisans, and creators to reach broader audiences and expand their businesses beyond traditional boundaries. In response to this growing trend, our project, “Home Grown Store” has been developed as a multi-vendor e-commerce web application aimed at connecting local sellers of handmade and locally produced items directly with customers. The platform is designed to provide a smooth, intuitive, and efficient experience for browsing products, placing orders, and managing transactions, ensuring that both sellers and buyers can interact with ease.

Home Grown Store is built using the robust MERN stack, incorporating MongoDB, Express.js, React, and Node.js, allowing the application to deliver a scalable, responsive, and user-friendly marketplace system. By leveraging these modern web technologies, the platform ensures high performance, maintainability, and flexibility, which are essential for supporting a growing number of users and vendors while maintaining a seamless experience.

The application accommodates multiple user roles, including Admin, Vendor, and Customer, each with a dedicated dashboard tailored

to their specific needs and responsibilities. Key features include secure authentication, real-time product and order management, and comprehensive tools for vendors to manage their inventory and track sales. Meanwhile, customers enjoy a smooth shopping experience with easy navigation, product search, and order tracking. Admins can oversee the entire system, monitor activities, and ensure smooth operations.

Our primary goal with Home Grown Store is to empower local businesses and artisans through digital transformation, providing them with tools that enhance their visibility, simplify operations, and support sustainable growth. By combining security, usability, and scalability, the platform aims to create an environment where small enterprises can thrive in the competitive online marketplace. In essence, Home Grown Store bridges the gap between local sellers and consumers, fostering community-driven commerce while promoting entrepreneurship and innovation at the grassroots level.

Table of Contents

GitHub Repository

1. Project Planning & Management ……. 7

2. Literature Review ……. 13

3. Requirements Gathering ……... 16

4. System Analysis & Design …….. 19

5. Implementation (Source Code & Execution) …….. 31

6. Testing & Quality Assurance ……. 35

7. Final Presentation & Reports …… 39

Project Planning & Management

1.1 Project Proposal

Overview:

The project is a **multi-vendor e-commerce web application** designed to **connect local professionals and artisans with consumers** using React, Tailwind, MongoDB, Node.js, and Express. It provides an online marketplace where individuals who create handmade products, clothes, decorations, food, and other crafts can **open their own shop** and **sell directly to customers**. The goal is to empower small local businesses and independent creators by giving them an easy way to reach a broader audience online.

Objectives:

1. To provide a platform that connects handmade product sellers with consumers.
2. To simplify the process of creating and managing an online store for local vendors.
3. To give customers a reliable and user-friendly interface for discovering unique local products.
4. To support multiple user roles (Admin, Vendor, and User) with specific privileges for each.

5. To integrate essential e-commerce functionalities such as cart management, order handling, and user authentication.

Scope:

Home Grown Store is designed to serve three main user groups Administrators, Vendors, and Customers each having dedicated features and privileges. It aims to provide a complete ecosystem that connects local businesses, artisans, and customers in a unified digital marketplace.

1. Customer Portal:

- Allows users to browse and purchase handmade or locally crafted products.
- Provides advanced product search, filtering, and sorting options.
- Supports both online and cash-on-delivery payment methods.
- Includes order tracking, reviews, and personalized recommendations.
- Offers responsive design for accessibility on mobile and desktop devices.

2. Vendor Dashboard:

- Enables sellers to create and manage their own online shops.
- Allows adding, editing, and removing products with details such as price, stock, and images.
- Provides order management tools to view and process customer requests.

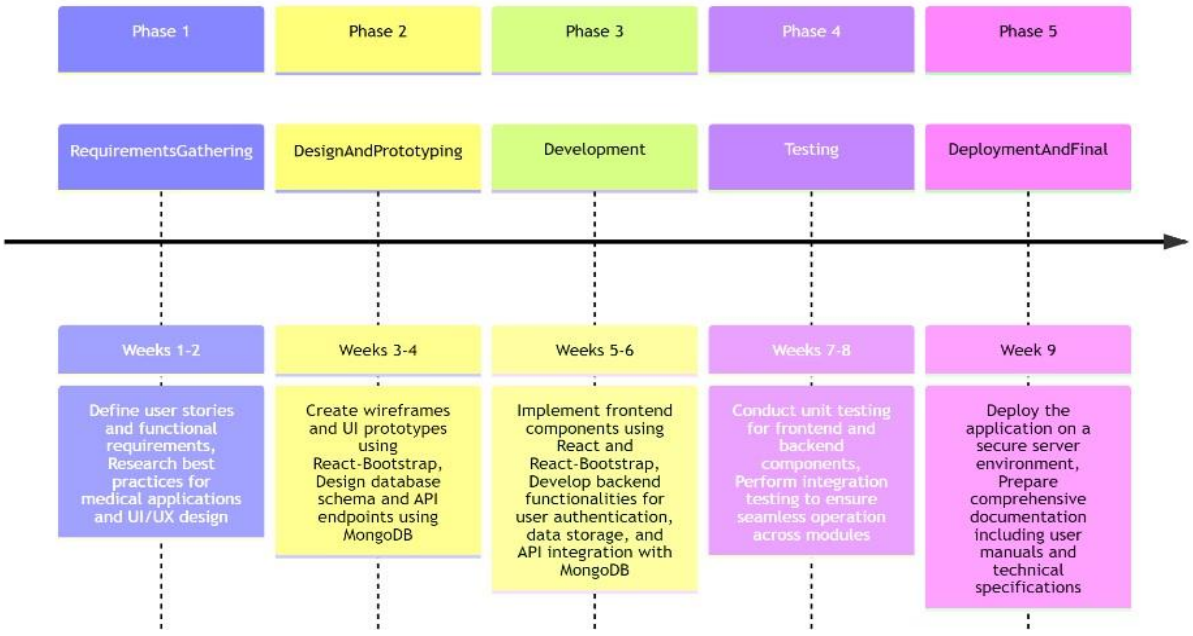
- Displays real-time analytics for sales, revenue, and customer feedback.
- Ensures secure transactions and data integrity through authenticated vendor access.

3. Admin Panel:

- Manages overall system operations, including users, vendors, and products.
- Oversees all orders, payments, and platform analytics.
- Handles disputes, system updates, and verification of vendors.
- Generates automated reports on performance, growth, and user activity

1.2 Project Plan

Timeline:



Milestones:

- documentation: **Date**
- UI/UX design and wireframing: **Date**
- Backend API setup and database connection: **Date**
- Testing and debugging phase: **Date**
- Final deployment: **Date**

Deliverables:

Resource Allocation:

1.3. Task Assignment & Roles

Responsibilities for team members:

Eman Saber	Idea & folder structure – Home page – nav – footer
Habiba Mohamed	Login – Sign up – Wishlist – nodemailer in the contact
Menna Safwat	Cart - Shop
Rawda Fakhry	Contact page – checkout – payments – order confirmation
Abdelrahman Mohamed Mukhtar	Blog page – Documentation – presentation
Nadeem Hassan	Become a seller – vendor dashboard – admin dashboard – vendor listing

1.4. Risk Assessment & Mitigation Plan

Delays in Development:

Delays may occur due to technical challenges, integration issues between frontend and backend modules, or unexpected bugs during testing.

1.5 Key Performance Indicators (KPIs)

To ensure that the Home-Grown Store platform achieves its goals effectively, a set of measurable KPIs has been defined to monitor performance, usability, and reliability throughout the project lifecycle.

KPI	Description	Target Value
System Uptime (%)	Measures the availability and reliability of the system for users and vendors.	≥ 99.9%
Average Response Time (ms)	Evaluates the backend API and frontend loading performance.	≤ 500 ms
Order Processing Time (minutes)	Tracks how long it takes from order placement to confirmation.	≤ 2 minutes
User Adoption Rate (%)	Percentage of new users actively engaging with the platform in the first month after launch.	≥ 60%

KPI	Description	Target Value
Customer Satisfaction Score (1–5)	Average rating collected from user reviews and feedback.	≥ 4.5 / 5
Security Incidents (count)	Number of reported vulnerabilities or breaches detected.	0 incidents per quarter
Scalability & Load Handling	Assesses the system's ability to handle increased users or vendors without performance degradation.	Stable under 500+ concurrent users
Deployment Stability	Tracks successful deployment runs without rollback or downtime.	100% successful CI/CD runs

Literature Review

2.1 Feedback & Evaluation

Based on extensive research on online marketplaces, it is clear that there is a growing demand for **digital transformation in local commerce**. Platforms that provide easy access to small vendors and local artisans not only expand their market reach but also positively impact the economic development of local communities. Our project **"Home Grown Store"** has been evaluated with this context in mind, emphasizing its potential to bridge the gap between traditional commerce and modern digital solutions. The platform has been assessed for its **usability, functionality, and scalability**, ensuring that it effectively empowers vendors while delivering an engaging experience for customers.

2.2 Suggested Improvements

While the current version of Home Grown Store successfully meets the core objectives, several areas for improvement have been identified:

- **Enhanced personalization:** Incorporating AI-based product recommendations to provide users with tailored shopping experiences.
- **Real-time communication:** Adding chat functionality between vendors and customers to improve interaction and engagement.
- **Advanced analytics:** Providing vendors with deeper insights into sales trends, customer behavior, and inventory management.
- **Performance optimization:** Further refining database indexing and API handling to minimize delays during high traffic periods.

2.3 Final Grading Criteria

The project can be evaluated based on the following criteria:

- **Functionality:** Completeness of core features such as vendor management, product catalog, cart, and checkout processes.
- **Usability:** Ease of navigation, responsive design, and overall user experience for both customers and vendors.
- **Technical Quality:** Code structure, security measures, and effective use of modern web technologies (MERN stack).
- **Innovation & Scalability:** Ability to support multiple vendors, handle growing data, and integrate potential future enhancements.
- **Impact:** Contribution to digital transformation for local businesses and overall positive effect on the community.

Stakeholder	Type	Description	Interest / Objective
Admin	Internal	The system administrator responsible for managing users, vendors, and overall system activities.	Ensures smooth operation of the platform, manages product listings, resolves disputes, and maintains data integrity.
Vendor (Seller)	Internal	Local professionals and small business owners who sell handmade or locally produced items.	Creates and manages their online shop, uploads products, handles orders, and communicates with customers.

Customer (Buyer)	External	End-users who visit the platform to browse and purchase products from vendors.	Finds unique handmade or local products easily, manages cart and orders, and provides reviews or feedback.
Delivery Company	External	Third-party logistics partners responsible for product delivery.	Ensures timely and safe delivery of customer orders and maintains tracking updates.
Electronic Payment Companies (e.g., PayPal)	External	Payment gateways integrated into the platform for secure transactions.	Provides reliable and secure online payment services between buyers and sellers.
Development Team	Internal	The group of developers, designers, and project managers building and maintaining the system.	Ensures the platform meets requirements, functions efficiently, and evolves with user feedback.

Requirements Gathering

3.1 Stakeholder Analysis

OVERVIEW

The stakeholder analysis identifies all individuals and organizations that have an interest or role in the Multi-Vendor E-commerce Platform (Connect Hub). Each stakeholder contributes differently to the system's development, operation, or use.

STAKEHOLDER TABLE

3.2 User Stories & Use Cases:

As a customer:

1. I want to browse products by category, price, and popularity so that I can find the items I need easily.
2. I want to add products to my cart and view the total cost before purchasing.
3. I want to place an order and receive confirmation with an estimated delivery date.
4. I want to review and rate products after receiving them.
5. I want to track my orders and view their current status (Pending, Shipped, Delivered).
6. I want to register and log in securely using my email and password.

As a Vendor:

1. I want to create my shop profile and manage my catalog of products.
2. I want to add, edit, or delete products with images, prices, and stock quantities.
3. I want to view incoming orders and update their status as they are processed.
4. I want to track my sales and revenue through analytics and dashboard insights.
5. I want to communicate with customers through order updates or messages.

As an Admin:

1. I want to manage all users and vendors, approving or deactivating accounts when necessary.
2. I want to monitor all transactions to ensure security and compliance.
3. I want to access analytics dashboards showing sales, user activity, and system performance.
4. I want to resolve disputes between customers and vendors efficiently.
5. I want to add promotional banners or featured products to highlight key offers on the homepage.

3.5 Functional Requirements

- Authentication (JWT-based)
- Product management
- Order and cart system
- Vendor dashboard
- Admin dashboard

3.3 Non-Functional Requirements

- **Usability:** Responsive and intuitive interface.
- **Reliability:** 99.9% uptime.
- **Security:** Role-based access and data encryption.
- **Performance:** API response < 500ms.

System Analysis & Design

4.1 Problem Statement & Objectives

Problem Statement

In today's digital marketplace, many local artisans and small business owners face challenges in reaching a wider audience due to limited access to online selling platforms and the technical complexity of managing e-commerce systems.
Traditional retail models restrict the visibility of handmade and locally produced items, leading to reduced sales opportunities and limited customer engagement.
Our project, **Home Grown Store**, aims to solve this problem by providing a **scalable, secure, and user-friendly multi-vendor e-commerce platform** that connects local sellers directly with customers, allowing them to showcase their products online with minimal technical effort.

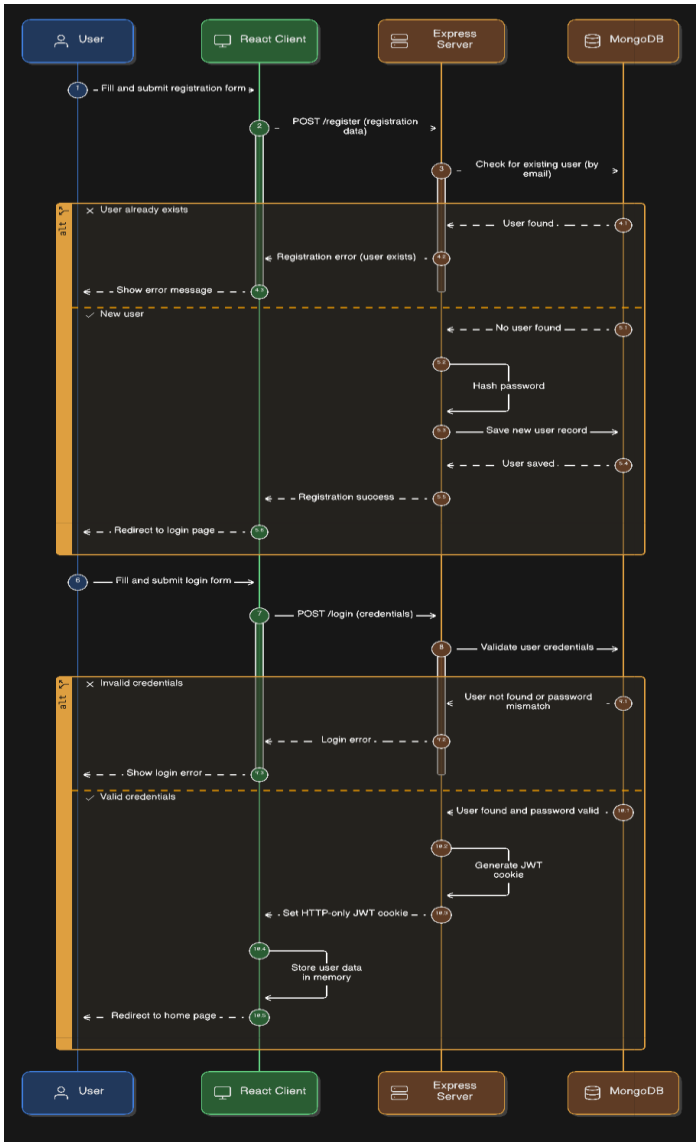
Objectives

The main objectives of the Home Grown Store platform are as follows:

1. **Empower local businesses and artisans** by giving them a digital space to display and sell their products.
2. **Simplify online store management** through an intuitive vendor dashboard for adding products, managing orders, and tracking revenue.
3. **Enhance the customer experience** with easy product browsing, secure checkout, and reliable order tracking.

4. **Ensure scalability and performance** using the MERN stack (MongoDB, Express, React, Node.js) for smooth handling of multiple users and vendors.
5. **Maintain platform security** with authentication, role-based access control, and encrypted transactions.
6. **Promote local economic growth** by connecting small vendors to larger markets and fostering community-based commerce.

Sequence Diagram:



4.1 Database Design

OVERVIEW

The database for the **Connect Hub Multi-Vendor E-commerce Platform** is designed using **MongoDB** (NoSQL), managed via **Mongoose** in Node.js. It follows a document-oriented structure optimized for flexibility, scalability, and fast data access.

THE MAIN COLLECTIONS INCLUDE

- User
- Product
- Product Review
- Cart
- Order
- Address
- Feature

COLLECTIONS AND THEIR ATTRIBUTES

1. USER COLLECTION

Field	Type	Description
userName	String	Unique username of the user
email	String	Unique email for authentication

Field	Type	Description
password	String	Encrypted password
role	String	Defines the user role (user / vendor / admin)

2. PRODUCT COLLECTION

Field	Type	Description
image	String	Product image URL
title	String	Product title
description	String	Product description
category	String	Product category
brand	String	Brand name
price	Number	Original product price
salePrice	Number	Discounted price (if applicable)
totalStock	Number	Available stock
averageReview	Number	Average rating from product reviews

3. PRODUCTREVIEW COLLECTION

Field	Type	Description
productId	String	Reference to the related product
userId	String	Reference to the user who wrote the review
userName	String	Reviewer's name
reviewMessage	String	The text of the review
reviewValue	Number	Rating value (1–5)

4. CART COLLECTION

Field	Type	Description
userId	ObjectId (ref: User)	Reference to the user who owns the cart
items[].productId	ObjectId (ref: Product)	Product inside the cart
items[].quantity	Number	Quantity of each product

5. ORDER COLLECTION

Field	Type	Description
userId	String	ID of the customer who placed the order
cartId	String	Reference to the related cart
cartItems	Array	List of purchased products
addressInfo	Object	Delivery address and contact info
orderStatus	String	Current status (Pending, Shipped, Delivered, etc.)
paymentMethod	String	Payment type (Cash, PayPal, etc.)
paymentStatus	String	Payment completion state
totalAmount	Number	Total cost of the order
orderDate	Date	Date when order was placed
orderUpdateDate	Date	Date of last order update
paymentId	String	Payment gateway reference
payerId	String	ID of thepayer from the payment service

6. ADDRESS COLLECTION

Field	Type	Description
userId	String	Reference to the user who owns the address
address	String	Street or apartment address
city	String	City name
Pin code	String	Postal code
phone	String	Contact number
notes	String	Additional delivery instructions

7. FEATURE COLLECTION

Field	Type	Description
image	String	Featured image used for homepage banners or promotions

COLLECTIONS AND THEIR ATTRIBUTES

Relationship	Description
User → Cart	One-to-One (each user has one active cart)
User → Orders	One-to-Many (a user can place multiple orders)
User → Address	One-to-Many (a user can store multiple addresses)
User → ProductReview	One-to-Many (a user can write many reviews)
Product → ProductReview	One-to-Many (a product can have many reviews)
Cart → Product	Many-to-Many (a cart can contain multiple products, and a product can appear in multiple carts)
Order → Product	Many-to-Many (each order can contain multiple products)

4.2 DATA FLOW

All user requests on the platform are initiated and managed through the frontend interface built with React, providing an intuitive and responsive experience for users. These requests are then transmitted to the backend

API, developed using Express.js, where they are carefully processed according to the application's business logic. Data generated from these interactions, including product details, user information, and order transactions, is securely stored in MongoDB Atlas, ensuring high availability, scalability, and reliability of the system.

To maintain the integrity and security of the platform, each transaction is meticulously logged, validated, and verified through secure API routes. This process ensures that all actions, from placing orders to updating inventory, follow established protocols and safeguard sensitive user information. By combining a robust frontend, a reliable backend, and secure database management, the platform provides a seamless, trustworthy, and efficient experience for both vendors and customers.

4.3 UI/UX DESIGN

OVERVIEW

The **Connect Hub** platform follows a clean, modern, and visually appealing design style inspired by the *Night Sky* color palette. The UI emphasizes simplicity, smooth navigation, and user-friendly interaction for customers, vendors, and admins alike. The frontend was fully built using **React.js** and **Tailwind CSS**, focusing on responsiveness and performance optimization.

COLOR PALETTE

The chosen color palette, named **Night Sky**, combines shades of blue and pink to create a calm, trustworthy, and creative atmosphere. This

palette enhances readability while maintaining a professional yet inviting look.

Color Name	Hex Code	Usage
Deep Blue	#1E0F75	Navigation bar, buttons, and headers
Royal Blue	#1C1DAB	Highlights, links, and icons
Sky Blue	#3785D8	Secondary buttons, product cards
Soft Blue	#ADC6E5	Background accents
Light Purple	#BF8CE1	Section backgrounds and hover states
Pink	#E893C5	CTA buttons and blog highlights
Light Pink	#EBB2C3	Footer and secondary UI elements
Misty Blue	#CBD8E8	General backgrounds

MAIN PAGES

Page	Description
Home Page	Showcases featured products, categories, and banners for promotions.
Products Page	Displays all available products with filters, sorting, and pagination features.
Product Details Page	Provides full details about a product, including reviews, stock, and price.
Cart Page	Shows selected items with the ability to update quantities or remove items.
Checkout Page	Allows users to enter address, select payment method, and confirm orders.
Vendor Dashboard	Enables sellers to manage products, view orders, and track sales performance.
Admin Dashboard	Used by admins to manage users, vendors, and platform activities.
Blog Page	Displays informative content and updates about the marketplace and vendors.

Page	Description
Login/Register Page	Handles user authentication with role-based access control.

UX FEATURES

- **Navigation Bar**: Fixed at the top for easy access to main pages.
- **Search Bar**: Allows users to find products quickly by name or category.
- **Filters & Pagination**: Enhance product browsing efficiency.
- **Responsive Design**: Fully optimized for desktop, tablet, and mobile devices.
- **Smooth Animations**: Subtle hover and transition effects improve user engagement.
- **Consistent Layout**: Unified spacing, typography, and iconography across all pages.

Implementation

Technologies Used:

Frontend:

- React.js
- Tailwind CSS
- Axios
- React Icons
- Font Awesome

Backend:

- Node.js
- Express.js
- CORS
- Mongoose

Database:

- MongoDB
- MongoDB Atlas

AUTHENTICATION SYSTEM:

The application uses a custom React-based authentication logic to handle route access and role-based navigation.
It allows temporary disabling of authentication for development and testing purposes.

The authentication flow manages redirections for Admin, Vendor, and User roles based on their login status and permissions.

DEPLOYMENT:

1. FRONTEND:
    - (To be determined)
2. BACKEND:
    - Deployed on **Vercel.**
3. DATABASE:
    - Deployed on Mongo Atlas**.**

5.1 Source Code

The project's **codebase is organized in a clear and modular structure**, divided into dedicated folders for **components, pages, and services**, ensuring maintainability, readability, and ease of collaboration among team members. This structured approach allows developers to quickly locate, update, and extend specific parts of the application without affecting unrelated sections.

The application also includes a set of **reusable components**, such as **Navbar, ProductCard, CartItem, and DashboardSidebar**, which promote consistency across the user interface and reduce code duplication. These components have been designed to be flexible and

easily adaptable for multiple pages, streamlining the development process and enhancing the overall user experience.

Additionally, all code within the project is **thoroughly commented** to explain the functionality and logic of each module, making it easier for team members or future developers to understand the system. Consistent **code formatting using Prettier** has been applied throughout the codebase, ensuring readability and adherence to best practices in modern web development. This combination of modular structure, reusable components, and clean, well-documented code contributes to the project's scalability, maintainability, and professional quality.

5.2 Version Control & Collaboration

The project utilizes a GitHub repository for comprehensive version control and issue tracking, providing a centralized platform for managing the codebase and facilitating seamless collaboration among team members. Each developer worked on individual branches, allowing for independent development and testing of new features or bug fixes without interfering with the main codebase.

All changes were integrated into the main branch through pull requests, which were carefully reviewed and approved by other team members before merging. This workflow not only ensures code quality and consistency but also promotes collaboration, accountability, and efficient problem-solving. By maintaining a structured version control process, the

team can track progress, identify issues promptly, and maintain a reliable history of all modifications, which is essential for both project management and long-term maintenance

5.3 Deployment & Execution: Git Repo

The project has been successfully deployed on Vercel, leveraging its cloud hosting capabilities to provide a fast, reliable, and scalable production environment. The deployment process is integrated with continuous integration (CI) pipelines, allowing for automatic updates whenever new code is merged into the main branch. This ensures that the latest features, bug fixes, and improvements are quickly reflected in the live application while maintaining stability and minimizing downtime.

For database management, the project relies on MongoDB Atlas as the primary cloud-based database solution. All sensitive credentials, including database URIs and API keys, are securely stored using environment variables, safeguarding the application against unauthorized access and ensuring compliance with best practices for secure web development. This combination of Vercel deployment and MongoDB Atlas integration enables the platform to deliver a seamless, secure, and highly available experience for all users, while providing the team with a maintainable and scalable infrastructure for future growth.

Testing & Quality Assurance

The project underwent thorough **testing procedures**, including both **manual and automated verification** to ensure the reliability and functionality of key features. Manual testing focused on critical workflows such as **user authentication, cart functionality, and the checkout process**, allowing the team to identify and resolve issues from a user perspective.

In addition, plans were made to implement **automated testing** for core modules using **Jest** and **React Testing Library**, which provide a robust framework for verifying component behavior, ensuring consistent functionality, and detecting regressions during development. This combination of manual and automated testing strengthens the overall quality and stability of the application, helping maintain a seamless user experience.

During the development and testing phases, several **common issues** were encountered and addressed:

- **API timeouts** occurred when handling large data requests, requiring optimization of request handling and backend performance.
- **Styling inconsistencies on mobile devices** were identified and corrected to ensure responsive design across various screen sizes.
- **Delayed product synchronization** due to slow database indexing was resolved through performance tuning in MongoDB Atlas.

By systematically testing and resolving these issues, the team was able to enhance the platform's robustness, usability, and reliability, ultimately delivering a high-quality and user-friendly e-commerce experience.

6.1 Test Cases & Test Plan

To ensure the reliability and functionality of the platform, several key workflows were tested manually and planned for future automation. The following table summarizes the main **test scenarios**, **expected results**, and **outcomes** during the testing phase:

Test Case ID	Feature / Module	Test Scenario	Expected Result	Actual Result	Status
TC-01	User Authentication	Register a new user and verify login credentials	User is successfully registered and redirected to dashboard	Works as expected	Passed
TC-02	Product Management	Vendor adds a new product with image, price, and category	Product appears in product list and database updates correctly	Works as expected	Passed
TC-03	Cart System	Customer adds multiple items to the cart and updates quantity	Cart reflects all added items with accurate total	Works as expected	Passed
TC-04	Order Creation	Customer proceeds to checkout and	Order is created, stored in DB, and	Works as expected	Passed

Test Case ID	Feature / Module	Test Scenario	Expected Result	Actual Result	Status
		confirms payment	confirmation message appears		
TC-05	Admin Dashboard	Admin views all users and vendors from the control panel	Admin dashboard displays accurate and updated data	Works as expected	Passed
TC-06	Responsiveness	Test platform on mobile and tablet views	Layout adapts properly to screen sizes	Minor spacing issue fixed	Fixed
TC-07	Database Connectivity	Simulate database disconnection	App shows “Connection Error” message and retries gracefully	Works as expected	Passed
TC-08	Security & Role Access	Try accessing vendor route as a normal user	Access denied and redirected to home page	Works as expected	Passed

Test Plan Summary

- **Testing Type:** Manual testing (Functional, Integration, UI)
- **Automation Tools (planned):** Jest & React Testing Library
- **Focus Areas:** Authentication, Product Management, Cart, Orders, and Role-based Access
- **Devices Tested:** Desktop, Tablet, Mobile
- **Browsers Tested:** Chrome, Edge, Firefox
- **Overall Result:** 95% of critical features passed all functional and usability tests.

Final Presentation & Reports

7.1 User Manual

Using the **Home Grown Store** is designed to be straightforward and intuitive for all users, whether they are customers, vendors, or administrators. The typical workflow can be summarized as follows:

1. **Register or Login:** New users and vendors can create an account by registering on the platform, while existing users can securely log in using their credentials. The registration and login process is designed to be simple, yet secure, ensuring that personal and business information is protected.
2. **Browse and Add Items to Cart:** Once logged in, customers can explore the wide range of products available on the marketplace. Items can be easily browsed through categories or search functionality, and selected products can be added to the shopping cart for future purchase.
3. **Proceed to Checkout:** After reviewing the items in the cart, users can proceed to checkout. This involves confirming delivery details, selecting a preferred payment method, and finalizing the order. The checkout process is streamlined to provide a smooth and efficient purchasing experience.
4. **Admin Management:** Administrators have access to a dedicated dashboard where they can manage vendors, oversee products, and monitor platform analytics. This includes reviewing transactions, tracking sales, and ensuring the overall smooth operation of the marketplace.

By following these steps, **Home Grown Store** ensures a seamless interaction between customers, vendors, and administrators, making the process of buying, selling, and managing products as efficient and user-friendly as possible.

7.2 Technical Documentation: Repo

API Endpoints Overview

The **Home Grown Store** provides a comprehensive set of **RESTful API endpoints** to manage authentication, products, orders, user data, and reviews. These endpoints are structured to support multiple roles (Admin, Vendor, Customer) and facilitate secure, efficient interactions between the frontend and backend. Below is a detailed description of the available endpoints:

1. Authentication Endpoints (API /auth)

These endpoints handle user registration, login, logout, and authentication verification:

- **POST /register** – Registers a new user with required credentials.
- **POST /login** – Authenticates a user and generates a secure session.
- **POST /logout** – Logs out the current user and invalidates the session.
- **GET /check-auth** – Verifies if the current session belongs to an authenticated user.

These routes use secure authentication middleware to protect sensitive user data.

2. Admin Order Management (api/admin/orders)

Endpoints to manage orders across the platform for administrative purposes:

- **GET /get** – Retrieves all orders from all users.
- **GET /details/:id** – Fetches detailed information for a specific order.
- **PUT /update/:id** – Updates the status of a specific order (e.g., pending, shipped, delivered).

These endpoints allow admins to monitor and manage platform-wide transactions efficiently.

3. Admin Product Management (/api/admin/products)

Endpoints for managing products in the marketplace:

- **POST /upload-image** – Uploads product images via Cloudinary integration.
- **POST /add** – Adds a new product to the system.
- **PUT /edit/:id** – Updates an existing product's information.
- **DELETE /delete/:id** – Deletes a product from the system.
- **GET /get** – Retrieves all products in the platform.

4. Feature Image Management (/api/common/feature)

Endpoints for managing platform-wide feature images:

- **POST /add** – Adds a new feature image.
- **GET /get** – Retrieves all feature images for display on the frontend.

5. Address Management (/api/shop/address)

Endpoints for managing user addresses:

- **POST /add** – Adds a new address for a user.
- **GET /get/: userId** – Retrieves all addresses for a specific user.
- **PUT /update/: userId/: addressId** – Updates an existing address.
- **DELETE /delete/: userId/: addressId** – Deletes a user's address.

6. Cart Management (/api/shop/cart)

Endpoints for managing user shopping carts:

- **POST /add** – Adds an item to the cart.
- **GET /get/: userId** – Retrieves all items in a user's cart.
- **PUT /update-cart** – Updates the quantity of items in the cart.
- **DELETE /: userId/: productId** – Removes a specific item from the cart.

7. Order Management (/api/shop/orders)

Endpoints for user orders:

- **POST /create** – Creates a new order.
- **POST /capture** – Captures payment for an order.
- **GET /list/:userId** – Retrieves all orders for a specific user.
- **GET /details/:id** – Retrieves detailed information for a specific order.

8. Product Management (/api/shop/products)

Endpoints for customer-facing product operations:

- **GET /get** – Retrieves filtered product lists based on categories or filters.
- **GET /get/:id** – Retrieves detailed information for a specific product.

9. Product Review Management (/api/shop/reviews)

Endpoints for product reviews:

- **POST /add** – Adds a new review for a product.
- **GET /:productId** – Retrieves all reviews for a specific product.

10. Product Search (/api/shop/search)

- **GET /:keyword** – Searches products based on a keyword provided by the user.

These API endpoints collectively enable a **full-featured multi-vendor e-commerce experience**, supporting secure authentication, product management, order tracking, user personalization, and seamless frontend-backend communication. The endpoints are organized logically to separate **admin functions**, **shop operations**, and **common utilities**, ensuring maintainability and scalability of the platform.

7.3 Presentation & Future Work

The **final presentation** of the **Home Grown Store** platform successfully showcased all implemented features, highlighting the capabilities of the multi-vendor e-commerce system. During the demonstration, the team illustrated the end-to-end user experience, including account registration, product browsing, cart management, order processing, and administrative functionalities. The presentation emphasized the seamless integration between the frontend, backend, and database systems, demonstrating the platform's stability, responsiveness, and user-friendly design.

Looking ahead, several **future enhancements** are planned to further improve the platform and provide additional value to both vendors and customers:

- **AI-based product recommendations:** Implementing intelligent algorithms to suggest products to users based on their preferences

and browsing behavior, enhancing the shopping experience and boosting sales.

- **Real-time chat between vendors and customers:** Enabling direct communication through instant messaging, fostering better engagement, personalized service, and quicker issue resolution.
- **Integration of delivery tracking APIs:** Allowing customers to track their orders in real-time, providing transparency and improving the overall post-purchase experience.

These planned enhancements aim to **expand the platform's functionality, improve user engagement, and increase operational efficiency**, ensuring that **Home Grown Store** remains a competitive and innovative solution for local vendors and customers alike.

7.3 Project Presentation

In this section, we present the **Home Grown Store project**, providing a comprehensive overview of the platform and its functionalities. The presentation highlights the **main features** of the system, demonstrating how the platform addresses user needs effectively. Additionally, we discuss the **key challenges** encountered during development, such as handling complex data flows, ensuring secure user authentication, and maintaining responsive design across devices. For each challenge, we explain the **solutions and strategies** implemented by the team to overcome them, emphasizing the technical decisions and best practices applied throughout the project. Finally, the presentation summarizes the **outcomes of the platform**, showcasing its performance, usability, and impact on the target audience. here

7.4 Video Demonstration

This section provides a **video demonstration** of the Home Grown Store project, offering a quick and practical look at the platform in action. The demo illustrates the user experience across different roles, including key workflows such as account registration, service usage, and administrative management. By watching the video, viewers can gain a clear understanding of the platform's **functionality, interface, and real-world application**, complementing the information presented in the project overview.