Digital Egypt Pioneers Initiative

React Front-End Development

Medical Application by React
Care Plus

Supervised by:

Eng. Basma Abdel Halim

Team members:

Banoub Nagy Edwar

Asmaa Mahmoud Ebeed

Mark Moheb Mamdouh

Sabah Abdelbaset Ahmed Ali

Yousef Ayman Metry

Acknowledgement

We extend our heartfelt appreciation to Eng. Basma Abdel Halim for her invaluable guidance and unwavering support throughout our journey in the Digital Egypt Pioneers Initiative. Her mentorship has been instrumental in shaping our experience, providing us with the knowledge, insights, and encouragement needed to excel.

Her technical expertise, constructive feedback, and commitment to fostering innovation have played a crucial role in the development and success of this initiative. She has consistently provided thoughtful advice and practical solutions, helping us overcome challenges and refine our approach.

We are truly grateful for the opportunities to learn, grow, and collaborate under her insightful leadership. Her dedication and passion have left a lasting impact on our journey, and we sincerely appreciate her efforts in guiding and inspiring us every step of the way.

Abstract

The growing demand for medical and healthcare supplies has highlighted the need for efficient, secure, and user-friendly digital platforms to facilitate their accessibility. This project focuses on developing an e-commerce platform for medical and health-related products, utilizing React for the front-end to deliver a seamless and responsive user experience. The platform is designed to streamline the purchasing process, ensuring ease of use for both individual consumers and healthcare institutions. By integrating a dedicated back-end system, it enables secure transactions, effective inventory management, and a structured approach to order processing. The system is tailored to provide a reliable and scalable solution that meets the specific needs of the medical sector.

Through the integration of modern front-end technologies and a well-structured back-end architecture, this project aims to enhance the accessibility and availability of essential medical supplies. It contributes to the digital transformation of healthcare commerce, offering a practical and innovative approach to online medical product distribution.

Table of Contents

GitHub Repository

1. Project Planning & Management 6
1.1. Project Proposal 6
1.1. Project Plan 7
1.2. Task Assignment & Roles 8
1.3. Risk Assessment & Mitigation Plan 9
1.4. KPIs (Key Performance Indicators) 9
2. Literature Review 10
2.1. Feedback & Evaluation 10
2.2. Suggested Improvements 10
2.3. Final Grading Criteria 10
3. Requirements Gathering 11
3.1. Stakeholder Analysis 11
3.2. User Stories & Use Cases 11
3.3. Functional Requirements 12
3.4. Non-functional Requirements 12
4. System Analysis & Design 13
4.1. Problem Statement & Objectives 20
4.2. Database Design & Data Modeling 21
4.3. Data Flow & System Behavior 22
4.4. UI/UX Design & Prototyping 23
4.5. System Deployment & Integration 24
4.6. Additional Deliverables 25

5. Implementation (Source Code & Execution) 19
5.1 Source Code 20
5.2 Version Control & Collaboration 21
5.3 Deployment & Excution 22
6. Testing & Quality Assurance 19
6.1 Test Cases and Test Plan 20
6.2 Automated Testing 21
6.3 Bugs 22
7. Final Presentation & Reports 19
7.1 User Manual 20
7.2 Technical Documentation 21
7.3 Presentation 22

Project Planning & Management

1.1. Project Proposal

Overview:

The project aims to develop a comprehensive medical e-commerce and management application using React, React-Bootstrap, MongoDB, and Context API. This application will serve both administrators and customers, offering a seamless experience for managing medical products, handling online and cash payments, and overseeing transactions efficiently.

Objectives:

- Develop a medical e-commerce platform where customers can browse and purchase medical products.
- Implement a dashboard for administrators to track orders, manage products, and monitor payments.
- Enable secure and efficient online and cash payment options, with all transactions recorded in the admin panel.
- Ensure a scalable architecture to accommodate future expansions in features and services.

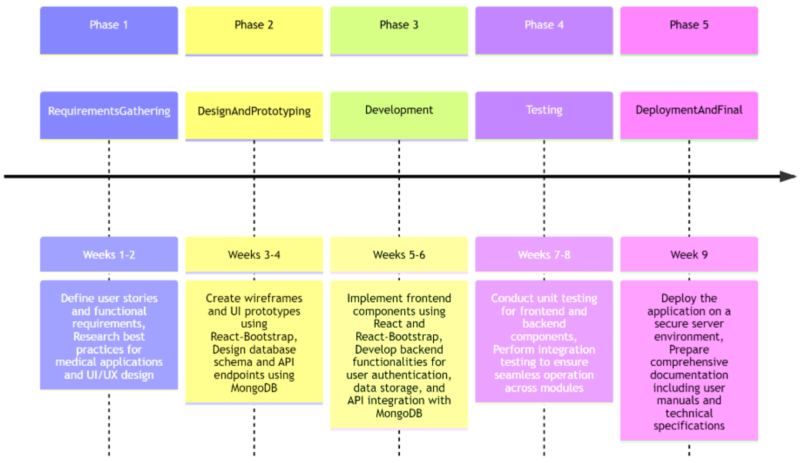
Scope:

The system will have two main interfaces:

1. Customer Portal:
   - Browse and purchase medical products.
   - Choose between online and cash payment methods.
   - View order history and track order status.
2. Admin Dashboard:
   - Manage product inventory, prices, and availability.
   - Monitor customer orders and payment transactions.
   - Generate reports for sales and customer activities.

1.2. Project Plan

Timeline:


Phase 1
Phase 2
Phase 3
Phase 4
Phase 5
RequirementsGathering
DesignAndPrototyping
Development
Testing
DeploymentAndFinal
Weeks 1-2
Weeks 3-4
Weeks 5-6
Weeks 7-8
Week 9
Define user stories and functional requirements, Research best practices for medical applications and UI/UX design
Create wireframes and UI prototypes using React-Bootstrap, Design database schema and API endpoints using MongoDB
Implement frontend components using React and React-Bootstrap, Develop backend functionalities for user authentication, data storage, and API integration with MongoDB
Conduct unit testing for frontend and backend components, Perform integration testing to ensure seamless operation across modules
Deploy the application on a secure server environment, Prepare comprehensive documentation including user manuals and technical specifications

Milestones:

- UI/UX design and database setup.

- Core functionality for product management and payments.

- Successful testing of transactions and admin controls.

- Deployment and final documentation submission.

Deliverables:

- A fully functional medical e-commerce system with admin and customer interfaces.

- Secure payment system with transaction tracking.

- A comprehensive admin dashboard for order and inventory management.

- Detailed Documentation & Presentation covering technical and user aspects.

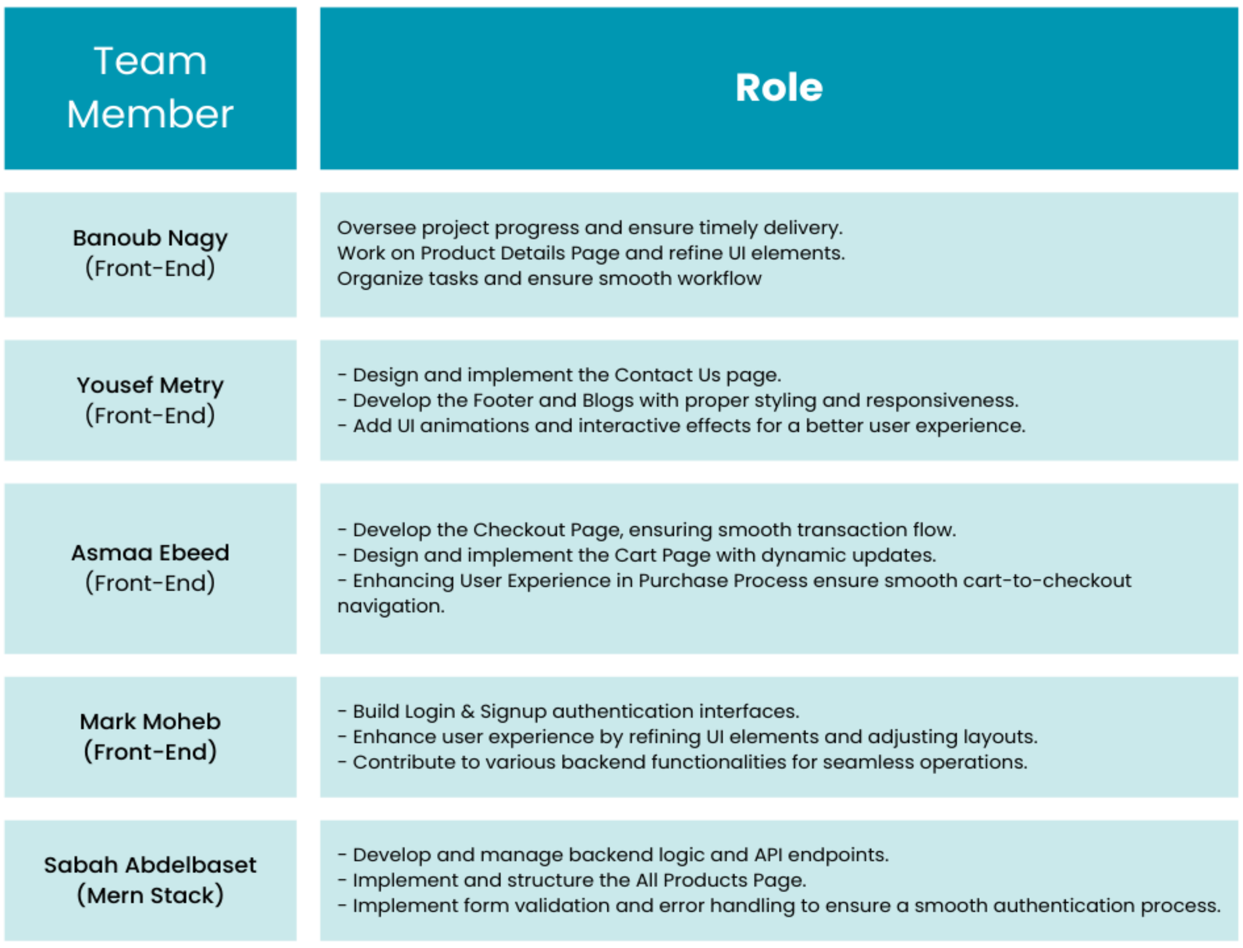
Resource Allocation:

- Development Team: Frontend Devs, Backend Devs, UI/UX .

- Tech Stack: React, React-Bootstrap, MongoDB, Node.js, Context API, Payment Gateway.

1.3. Task Assignment & Roles

Responsibilities for team members:



Team Member	Role
Banoub Nagy (Front-End)	Oversee project progress and ensure timely delivery. Work on Product Details Page and refine UI elements. Organize tasks and ensure smooth workflow
Yousef Metry (Front-End)	- Design and implement the Contact Us page. - Develop the Footer and Blogs with proper styling and responsiveness. - Add UI animations and interactive effects for a better user experience.
Asmaa Ebeed (Front-End)	- Develop the Checkout Page, ensuring smooth transaction flow. - Design and implement the Cart Page with dynamic updates. - Enhancing User Experience in Purchase Process ensure smooth cart-to-checkout navigation.
Mark Moheb (Front-End)	- Build Login & Signup authentication interfaces. - Enhance user experience by refining UI elements and adjusting layouts. - Contribute to various backend functionalities for seamless operations.
Sabah Abdelbaset (Mern Stack)	- Develop and manage backend logic and API endpoints. - Implement and structure the All Products Page. - Implement form validation and error handling to ensure a smooth authentication process.

1.4. Risk Assessment & Mitigation Plan

Delays in Development:

Define clear milestones and conduct regular progress reviews.

Allocate buffer time in the project timeline.

1.5. KPIs (Key Performance Indicators):

1. System Uptime (%) – Ensures system availability and reliability. Target: 99.9% uptime.

2. Response Time (ms) – Measures how quickly the system responds to user requests. Target: < 500ms.

3. Order Processing Time (minutes) – Tracks the average time taken from order placement to confirmation.

4. User Adoption Rate (%) – Percentage of new users actively engaging with the platform within the first month.

5. Customer Satisfaction Score (1-5) – Based on feedback and reviews from users.

6. Security Incidents (count) – Number of security breaches or vulnerabilities detected and resolved.

7. Scalability & Load Handling – Performance of the system under peak usage conditions.

Literature Review

2.1. Feedback & Evaluation

2.2. Suggested Improvements

Enhanced User Review System

Implement a structured review format with rating categories (e.g., delivery speed, product quality, user experience) and Allow users to submit detailed feedback with images to improve credibility.

Real-Time Feedback Collection & Automated Resolution

Add quick survey prompts after order completion for immediate insights.

Develop an automated response system to handle complaints efficiently and improve service.

Advanced Payment & Order Management

Implement multi-currency support to make the platform scalable for global users.

Introduce a refund & cancellation system to enhance customer trust and flexibility.

Admin Dashboard & Analytics Enhancements

Add visual analytics (charts & reports) for tracking sales, user activity, and order status.

Enable pharmacists & doctors to validate medication accuracy, ensuring better service quality.

2.3. Final Grading Criteria

Requirements Gathering

3.1 Stakeholder Analysis:

Customers (End-Users): Need an easy-to-use platform to browse, purchase, and manage medical products with a seamless checkout experience.

Admins & Pharmacists: Require a dashboard to manage products, process orders, track payments, and validate prescriptions if needed.

Business Owners: Expect real-time analytics, sales tracking, and a secure payment system to ensure smooth operations.

Delivery Personnel: Need a system that provides clear order details, delivery addresses, and status updates

3.2 User Stories & Use Cases:

As a customer, I want to browse medical products, add them to my cart, and securely complete my purchase.

As a customer, I want to track my order status and receive real-time updates.

As an admin, I want to manage inventory, approve or reject orders, and monitor payment transactions.

As an admin, I want to access real-time sales reports and user activity analytics.

As a pharmacist, I want to validate prescription-based orders before approval.

3.3 Functional Requirements :

User Authentication: Login & registration system with JWT authentication.

Product Management: Ability to add, update, and remove medical products.

Cart & Checkout System: Customers can add products to the cart, apply discounts, and complete payments.

Payment Integration: Secure online payment (Stripe, PayPal) and cash-on-delivery options.

Order Management: Admins can view, approve, and track orders.

Review & Feedback System: Users can rate and review products and services.

Analytics Dashboard: Real-time reports on sales, order status, and user activity.

3.4 Non-functional Requirements:

Performance: API response time should be under 500ms for a smooth user experience.

Security: Data encryption, secure payment processing, and role-based access control (RBAC).

Usability: Intuitive UI/UX with mobile-friendly design and accessibility support.

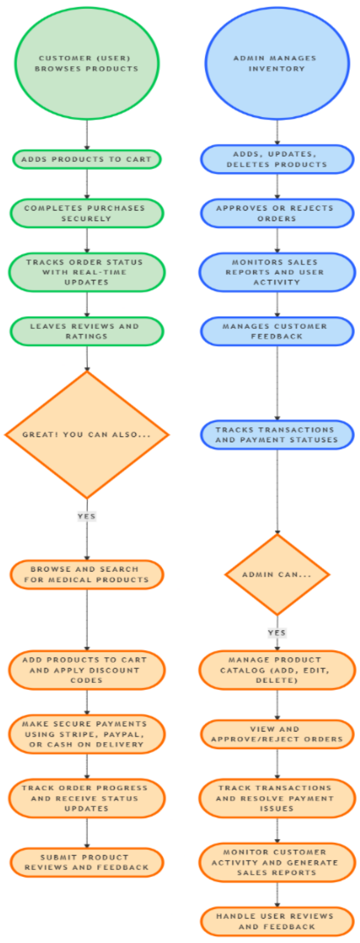
Reliability: 99.9% system uptime with cloud deployment and backup strategies.

System Analysis & Design

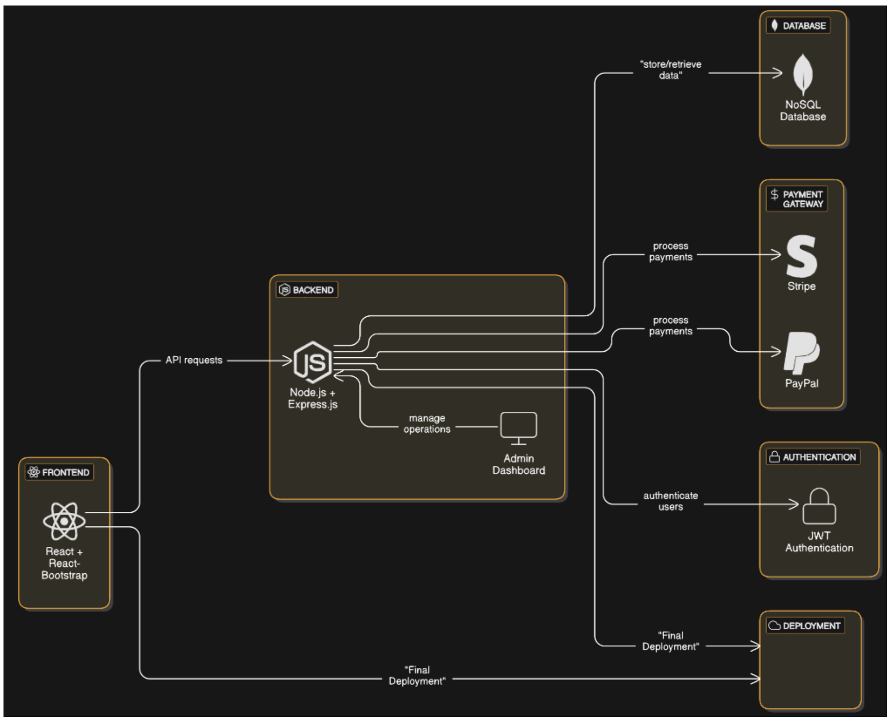
4.1 Problem Statement & Objectives:

Problem Statement:

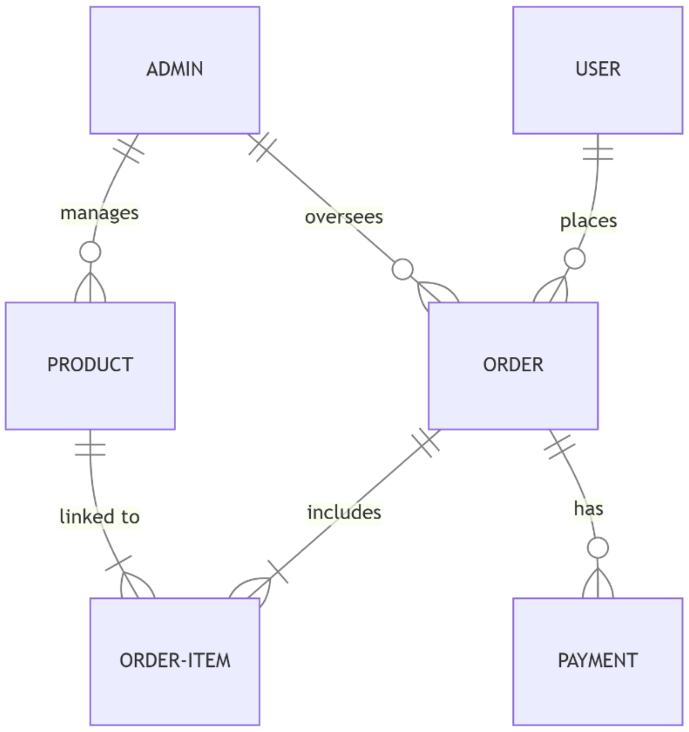
Managing online medical product sales and healthcare transactions can be challenging due to inefficient order processing, lack of real-time tracking, and limited digital payment options. This project aims to build a scalable, secure, and user-friendly medical e-commerce platform that streamlines online purchasing, payment, and admin management.

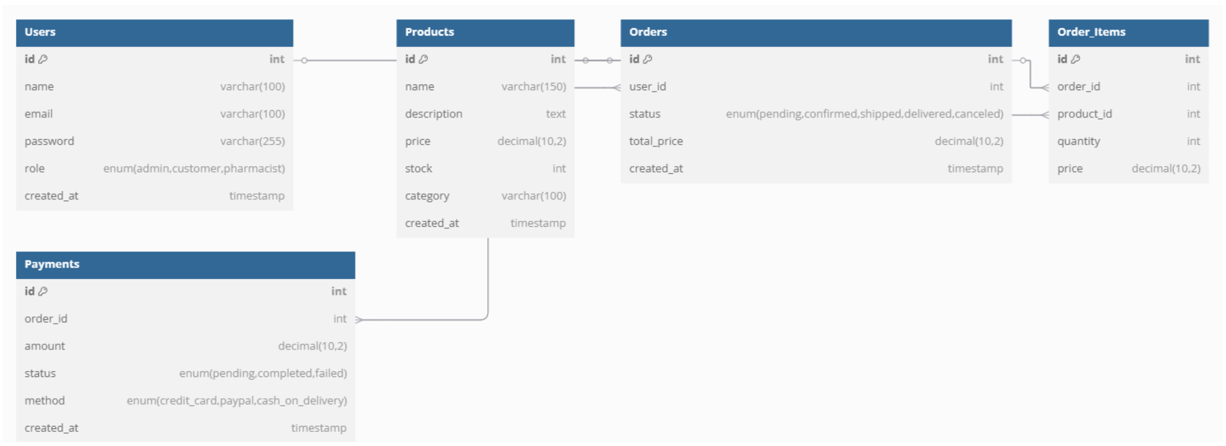

CUSTOMER (USER) BROWSES PRODUCTS
ADDS PRODUCTS TO CART
COMPLETES PURCHASES SECURELY
TRACKS ORDER STATUS WITH REAL-TIME UPDATES
LEAVES REVIEWS AND RATINGS
GREAT! YOU CAN ALSO...
YES
BROWSE AND SEARCH FOR MEDICAL PRODUCTS
ADD PRODUCTS TO CART AND APPLY DISCOUNT CODES
MAKE SECURE PAYMENTS USING STRIPE, PAYPAL, OR CASH ON DELIVERY
TRACK ORDER PROGRESS AND RECEIVE STATUS UPDATES
SUBMIT PRODUCT REVIEWS AND FEEDBACK
ADMIN MANAGES INVENTORY
ADDS, UPDATES, DELETES PRODUCTS
APPROVES OR REJECTS ORDERS
MONITORS SALES REPORTS AND USER ACTIVITY
MANAGES CUSTOMER FEEDBACK
TRACKS TRANSACTIONS AND PAYMENT STATUSES
ADMIN CAN...
YES
MANAGE PRODUCT CATALOG (ADD, EDIT, DELETE)
VIEW AND APPROVE/REJECT ORDERS
TRACK TRANSACTIONS AND RESOLVE PAYMENT ISSUES
MONITOR CUSTOMER ACTIVITY AND GENERATE SALES REPORTS
HANDLE USER REVIEWS AND FEEDBACK

Software Architecture:



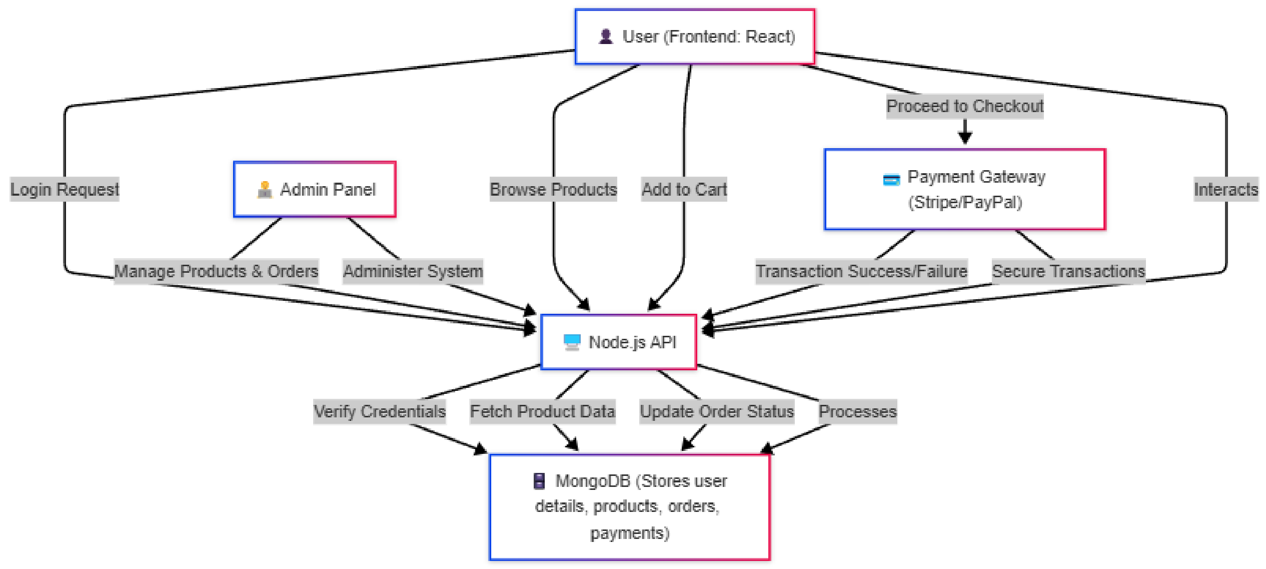
4.2 Database Design & Data Modeling:



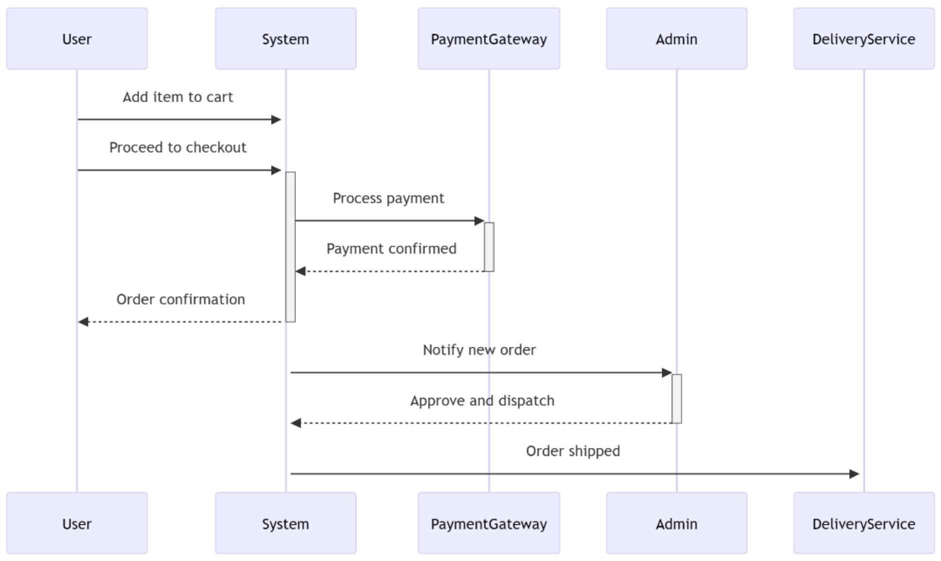


4.3 Data Flow & System Behavior

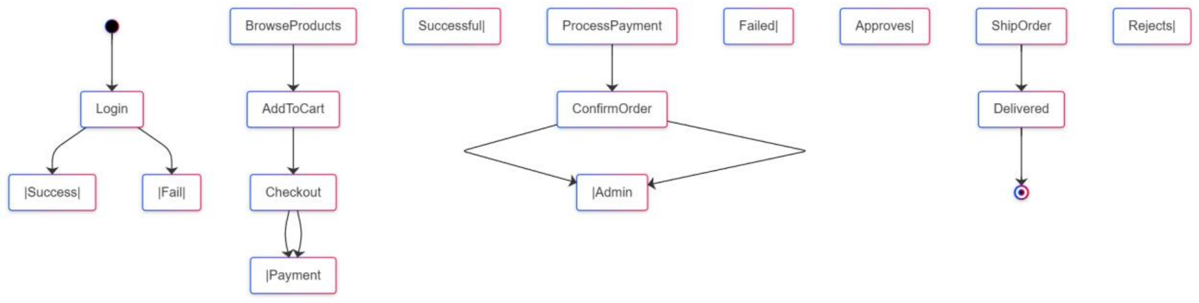
Data Flow Diagram



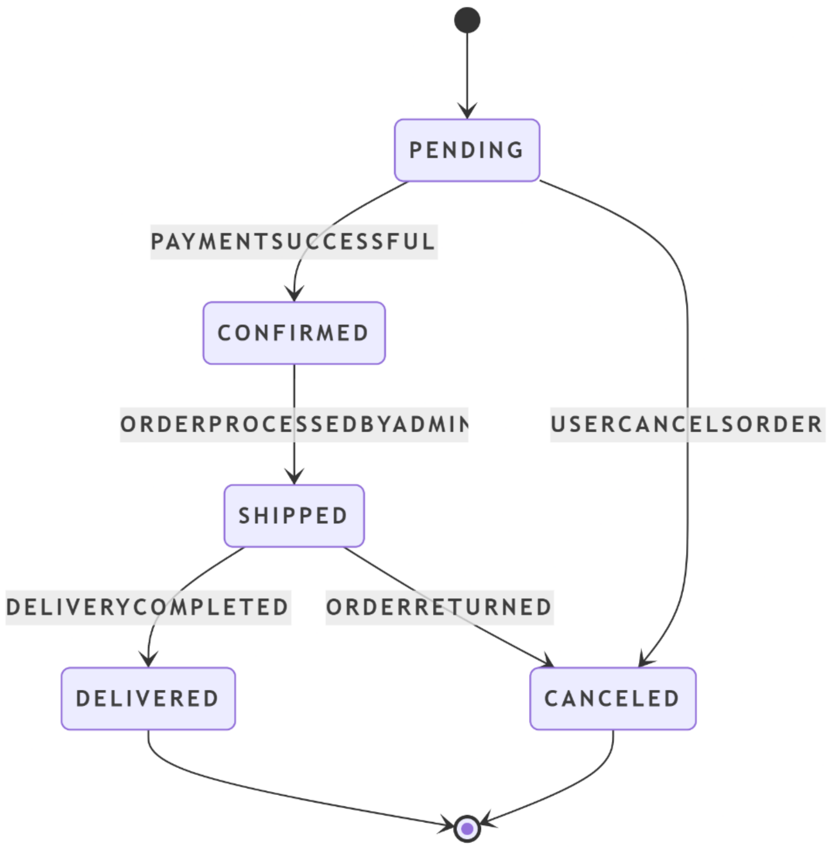
Sequence Diagram



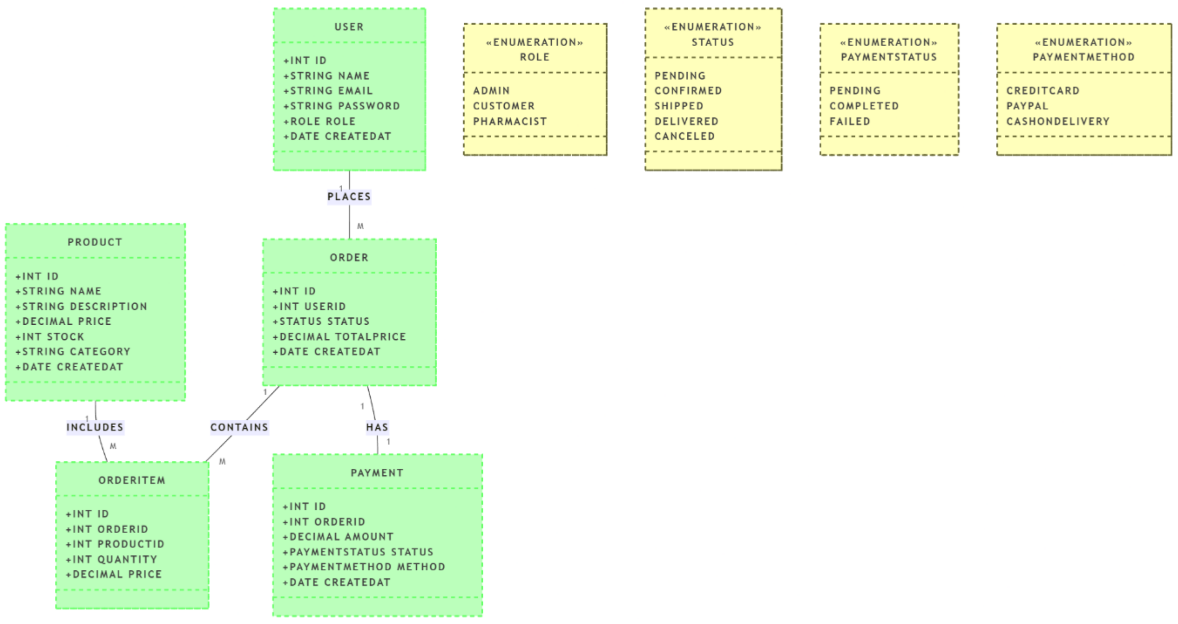
Activity Diagram:



State Diagram:



Class Diagram:



4.4 UI/UX Design & Prototyping:

App Wireframes : Care+

Our Idea Based On Here

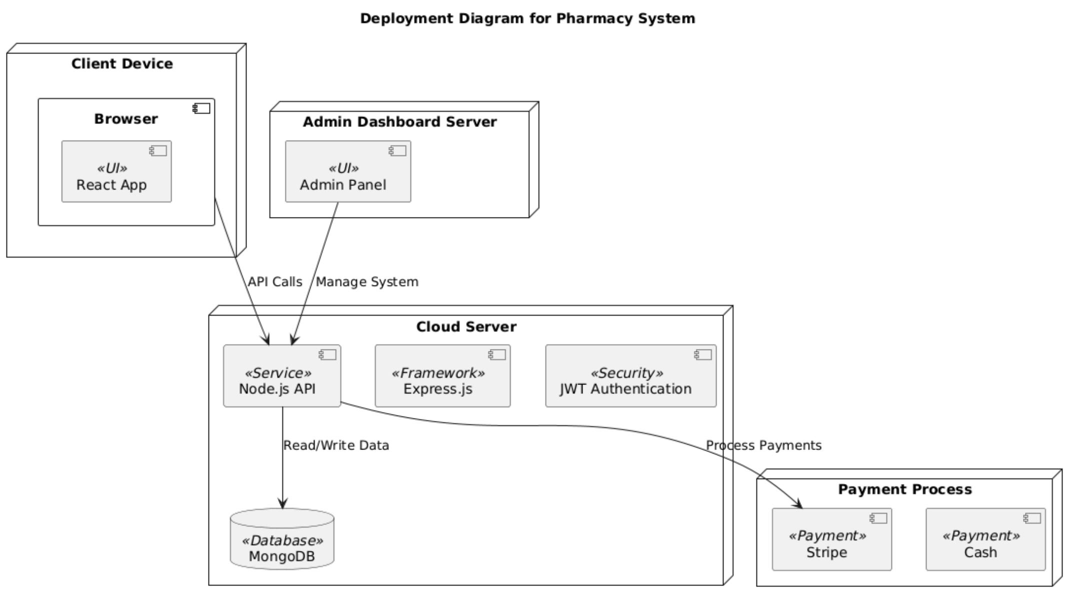
4.5 System Deployment & Integration

Technology Stack:

Frontend: React.js + React-Bootstrap
Backend: Node.js + Express.js
Database: MongoDB (NoSQL)
Payment: Stripe, Cash
Deployment: Vercel

Deployment Diagram:

The Client Device hosts the React App, which runs in the user's browser and interacts with the backend through API calls.
The Cloud Server houses the Node.js API and Express.js framework, handling business logic, authentication, and communication with the database.
MongoDB serves as the primary database, storing user data, orders, products, and transactions.
Payment Gateways (Stripe & PayPal) are integrated with the backend to handle secure payment processing.
The Admin Dashboard Server provides an interface for administrators to manage users, orders, and product inventory



Component Diagram:

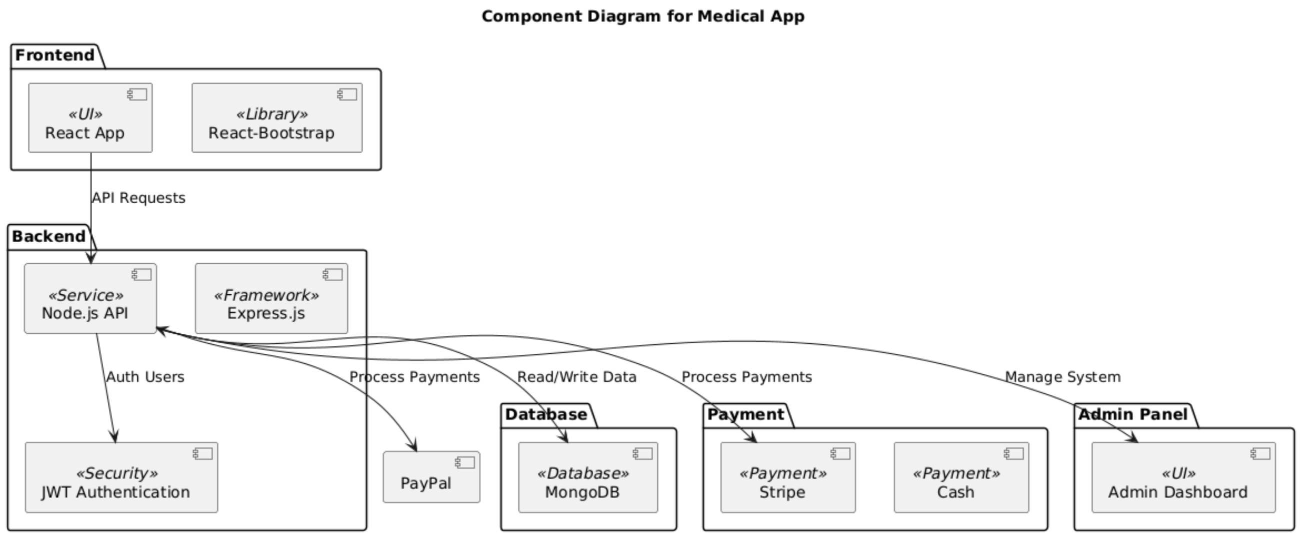
Frontend (React.js + React-Bootstrap): The user-facing interface where customers can browse products, place orders, and track deliveries. It communicates with the backend via RESTful API requests.

Backend (Node.js + Express.js): The core logic of the system, responsible for handling user authentication, processing orders, managing products, and handling payment transactions.

Database (MongoDB): Stores all essential data, including user details, product catalogs, orders, and payment records. It ensures data consistency and efficient retrieval.

Payment (Stripe/Cash): Handles secure transactions and payment processing for orders.

Admin Panel: A dedicated interface for administrators to manage users, products, and monitor system activities.



Implementation (Source Code & Execution)

5.1 Source Code

Structured & Well-Commented Code
We maintained a clean folder structure separating concerns (e.g., \`components\`, \`pages\`, \`services\`, \`contexts\`, etc.). Each component and function is clearly commented to explain its purpose and behavior, making it easier for future developers to understand and work with the codebase.
Coding Standards & Naming Conventions
We followed consistent naming conventions using camelCase for variables and functions, Pascal Case for components. Linting tool like Prettier were integrated into the project to ensure code quality and consistency across the team.
Modular Code & Reusability
The UI was built using reusable React components such as \`ProductCard\`, \`Sidebar\`, \`Navbar\`, \`AdminTable\`,\`ProtectedRoute\`,etc.. Utility functions and custom hooks were created to avoid code duplication, improving maintainability and scalability.
Security & Error Handling
We implemented form validation using libraries like Yup and React Hook Form to prevent invalid user input. Backend requests include proper authentication headers, and sensitive routes are protected via role-based access. We also used try-catch blocks and error boundaries to handle runtime errors gracefully.

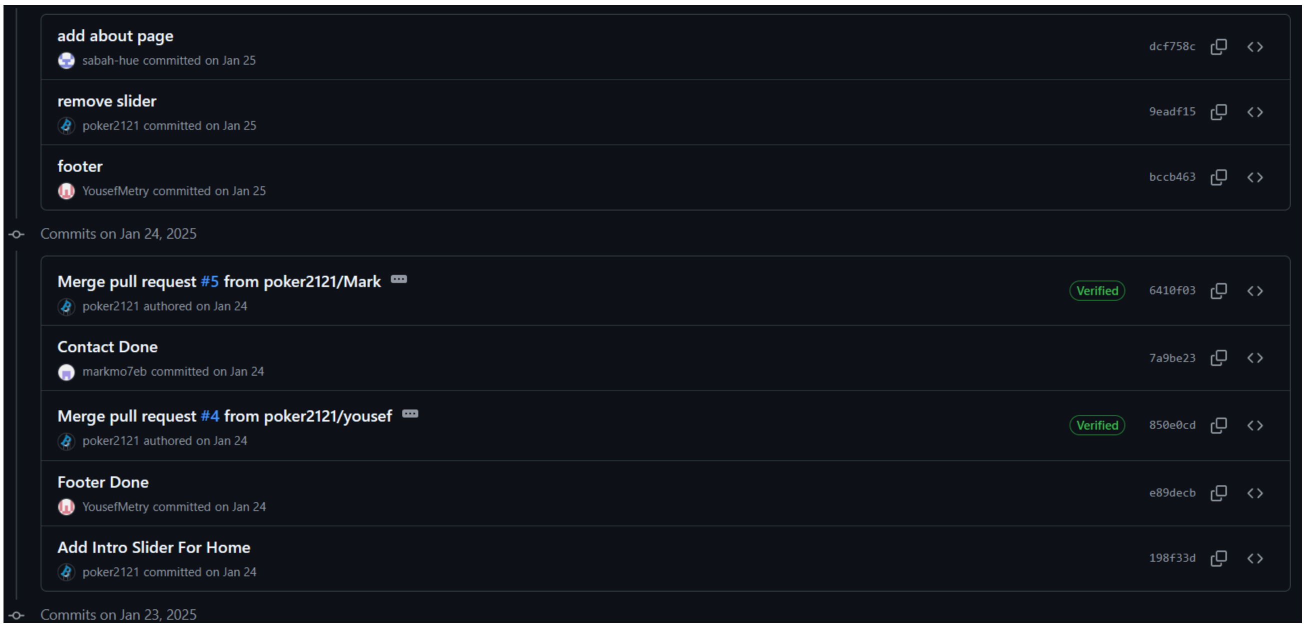
5.2 Version Control & Collaboration
Version Control Repository
The project was hosted on a private GitHub repository for source control and version tracking. In parallel, the app was deployed and continuously updated on Vercel, allowing for real-time previews and feedback throughout the development process.

Branching Strategy

Each team member worked on their assigned features or fixes in separate branches named after their tasks or personal identifiers. This method helped avoid distractions, allowed for focused development, and ensured smoother integration when merging changes.

Commit History & Documentation

Team members followed a clear commit convention: after completing a task, they wrote descriptive commit messages summarizing their work and informed the team for review. This approach helped maintain clarity and ensured that all updates were reviewed before being merged.



add about page
sabah-hue committed on Jan 25 — dcf758c

remove slider
poker2121 committed on Jan 25 — 9eadf15

footer
YousefMetry committed on Jan 25 — bccb463

Commits on Jan 24, 2025

Merge pull request #5 from poker2121/Mark
poker2121 authored on Jan 24 — Verified — 6410f03

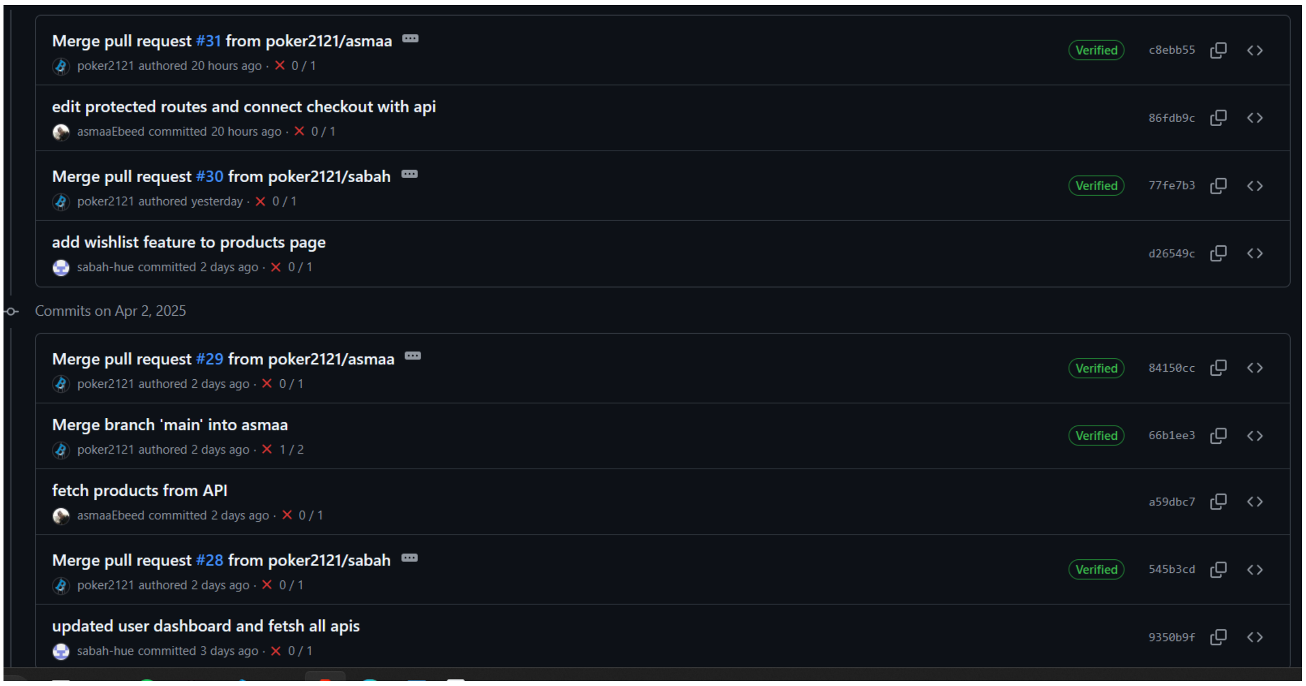
Contact Done
markmo7eb committed on Jan 24 — 7a9be23

Merge pull request #4 from poker2121/yousef
poker2121 authored on Jan 24 — Verified — 850e0cd

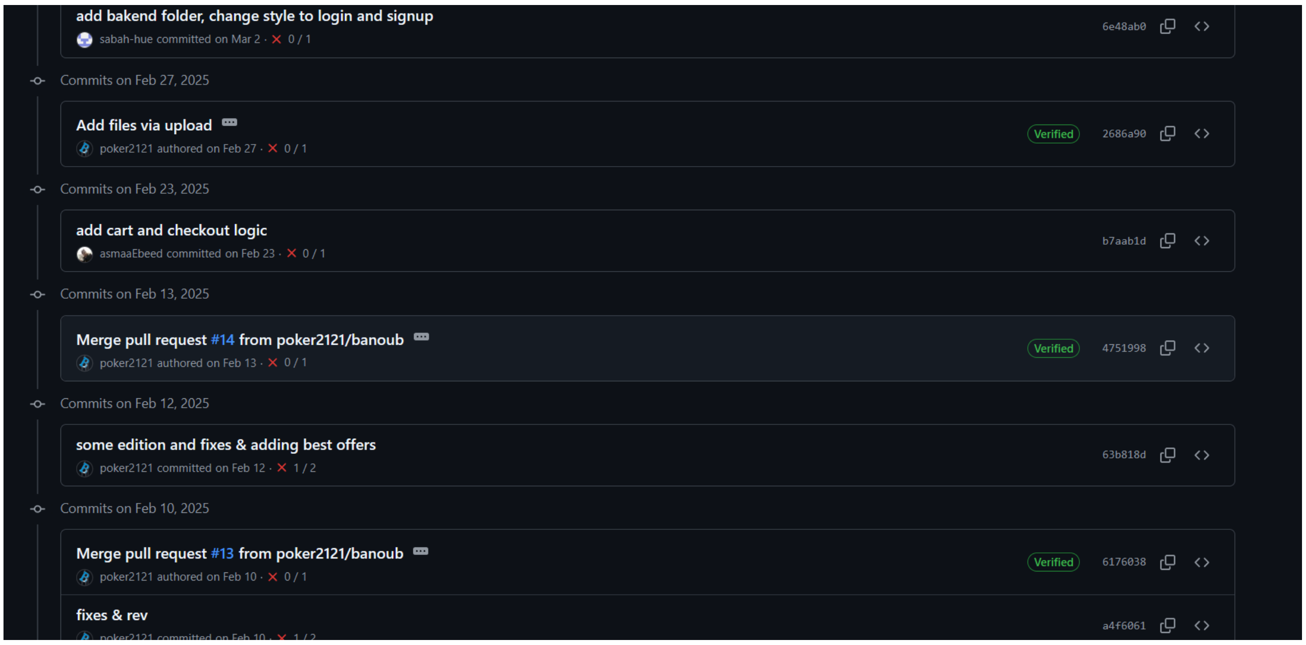
Footer Done
YousefMetry committed on Jan 24 — e89decb

Add Intro Slider For Home
poker2121 committed on Jan 24 — 198f33d

Commits on Jan 23, 2025



```
Merge pull request #31 from poker2121/asmaa                    Verified  c8ebb55
poker2121 authored 20 hours ago · ✕ 0 / 1
edit protected routes and connect checkout with api                      86fdb9c
asmaaEbeed committed 20 hours ago · ✕ 0 / 1
Merge pull request #30 from poker2121/sabah                    Verified  77fe7b3
poker2121 authored yesterday · ✕ 0 / 1
add wishlist feature to products page                                    d26549c
sabah-hue committed 2 days ago · ✕ 0 / 1
Commits on Apr 2, 2025
Merge pull request #29 from poker2121/asmaa                    Verified  84150cc
poker2121 authored 2 days ago · ✕ 0 / 1
Merge branch 'main' into asmaa                                 Verified  66b1ee3
poker2121 authored 2 days ago · ✕ 1 / 2
fetch products from API                                                  a59dbc7
asmaaEbeed committed 2 days ago · ✕ 0 / 1
Merge pull request #28 from poker2121/sabah                    Verified  545b3cd
poker2121 authored 2 days ago · ✕ 0 / 1
updated user dashboard and fetsh all apis                                9350b9f
sabah-hue committed 3 days ago · ✕ 0 / 1
```



```
add bakend folder, change style to login and signup                      6e48ab0
sabah-hue committed on Mar 2 · ✕ 0 / 1
Commits on Feb 27, 2025
Add files via upload                                           Verified  2686a90
poker2121 authored on Feb 27 · ✕ 0 / 1
Commits on Feb 23, 2025
add cart and checkout logic                                              b7aab1d
asmaaEbeed committed on Feb 23 · ✕ 0 / 1
Commits on Feb 13, 2025
Merge pull request #14 from poker2121/banoub                   Verified  4751998
poker2121 authored on Feb 13 · ✕ 0 / 1
Commits on Feb 12, 2025
some edition and fixes & adding best offers                              63b818d
poker2121 committed on Feb 12 · ✕ 1 / 2
Commits on Feb 10, 2025
Merge pull request #13 from poker2121/banoub                   Verified  6176038
poker2121 authored on Feb 10 · ✕ 0 / 1
fixes & rev                                                              a4f6061
poker2121 committed on Feb 10 · ✕ 1 / 2
```

CI/CD Integration

Although advanced CI/CD tools weren't fully set up, we maintained a manual testing workflow before merging any code. The Vercel integration also helped us catch layout or functional issues early by generating automatic previews for each branch.

5.3 Deployment & Execution

The project includes a comprehensive README.md file that guides developers, testers, and users through setup and usage in Git Repo

Includes:

• Installation steps
• System requirements (hardware/software dependencies)
• Configuration instructions
• Execution guide (running the project locally or accessing a deployed version)
• API documentation in Repo and Technical Documentation
• Executable Files & Deployment Link

Testing & Quality Assurance

6.1 Test Cases & Test Plan

Test Scenarios were documented and organized into categories, such as User Authentication, Admin Dashboard functionality, and Order Process. Each scenario included clear steps, inputs, and expected outcomes.

The Test Plan focused on critical workflows such as:

- User login/logout functionality
- Medication search and filter accuracy
- Online order creation and payment processing
- Admin dashboard data display (inventory, orders, and user management)
- Responsiveness of the user interface across different devices

6.2 Automated Testing

While Automated Testing was planned, we didn't implement full automation in this phase. However, we did focus on manually verifying core features such as user authentication, medication search, and order processing to ensure quality.

In the future, we plan to implement automated unit tests using Jest and React Testing Library to enhance coverage and ensure ongoing code stability.

6.3 Bug Reports

Common Issues:

- API response delays or failed requests
- UI misalignments on mobile screens
- Broken or outdated links in the navigation

Final Presentation & Reports

7.1 User Manual

1. Access the Website
Open a web browser and navigate the live application URL:
[https://final-project-tawny-xi.vercel.app]

2. Create an Account / Login
- Sign Up: If you are a new user, click on the "Sign Up" button, fill in your details (Name, Email, Password), and click "Register".
- Login: If you already have an account, click "Login", enter your credentials (Email, Password), and access your personalized dashboard.

3. Browse and Search for Medications
- You can browse the available medications by category or search for specific drugs using the search bar.
- Filters are available to refine the search based on categories such as medication type, price range, or availability.

4. Add to Cart and Checkout
- Once you've found the medication(s) you want, click on the "Add to Cart" button.
- Go to your cart by clicking the cart icon in the header.
- Review your items and click on "Proceed to Checkout".
- Select your payment method (Credit Card, PayPal, etc.), confirm your shipping details, and complete the payment process.

5. Order Confirmation & Invoice
- After successfully placing your order, a confirmation message will appear with the order summary.
- A detailed invoice will be automatically generated and sent to your registered email.

- You can also view and download your invoice at any time from the Orders section in your account.
- Once complete, you may continue browsing for additional medications or log out.

6. Admin Panel (For Admin Users Only)
- Dashboard: Admin users can access the dashboard to manage the inventory, view customer orders, and add/update medications.
- Inventory Management: Admins can add new medications, update existing stock levels, and remove medications from the catalog.
- Order Management: Admins can view order details, track order statuses, update delivery status, and manage any issues with shipments or payments.

7.2 Technical Documentation : Repo

The backend follows RESTful API structure with the following key routes:

Category
- POST /api/categories – Creates a new category for organizing products.
- PUT /api/categories – Updates an existing category's details.
- GET /api/categories – Retrieves a list of all categories.
- DELETE /api/categories – Deletes a specific category.
- GET /api/categories/:id – Retrieves details of a single category by ID.

Auth
- POST /api/auth/signup – Registers a new user account.
- POST /api/auth/login – Authenticates a user and returns a token for access.
- POST /api/auth/sendcode– Sends a verification or reset code to the user (e.g., for password reset).
- PUT /api/auth/resetpass – Resets the user’s password using a valid code.

User
- GET /api/users/:id – Retrieves details of a specific user by ID.
- GET /api/users – Retrieves a list of all users.
- PUT /api/users/profile – Updates the user's profile information.
- GET /api/users/profile – Retrieves the profile details of the authenticated user.
- DELETE /api/users – Deletes a specific user account.

Coupon
- POST /api/coupons – Creates a new coupon for discounts.
- PUT /api/coupons – Updates an existing coupon's details.
- DELETE /api/coupons – Deletes a specific coupon.
- GET /api/coupons – Retrieves a list of all coupons.
- GET /api/coupons/qrcode – Generates a QR code for a specific coupon.

Products
- POST /api/products – Creates a new product in the system.
- PUT /api/products – Updates an existing product's details.
- DELETE /api/products – Deletes a specific product.
- GET /api/products/pagination – Retrieves a paginated list of products.
- GET /api/products/search – Searches for products based on query parameters.
- GET /api/products/:id – Retrieves details of a specific product by ID.
- GET /api/products – Retrieves a list of all products.

Cart
- POST /api/cart – Adds or updates items in the user's cart.
- GET /api/cart – Retrieves the user's cart details.
- DELETE /api/cart – Deletes a specific item from the user's cart.
- DELETE /api/cart/clear – Clears all items from the user's cart.

Order
- POST /api/orders – Creates a new order for the user.
- PATCH /api/orders/cancel – Cancels an existing order.
- GET /api/orders – Retrieves a list of all orders.
- PUT /api/orders/status – Updates the status of a specific order.

Blog
- POST /api/blogs – Creates a new blog post.
- PUT /api/blogs – Updates an existing blog post.
- GET /api/blogs – Retrieves a list of all blog posts.
- GET /api/blogs/:id – Retrieves details of a specific blog post by ID.
- DELETE /api/blogs – Deletes a specific blog post.

Reviews
- POST /api/reviews – Creates a new review for a product.
- PUT /api/reviews – Updates an existing review.
- DELETE /api/reviews – Deletes a specific review.
- GET /api/reviews/product/:id – Retrieves all reviews for a specific product.

7.3 Project Presentation

In this presentation, we showcase the CarePlus project by highlighting the main features, key challenges we faced, the solutions we implemented, and the final outcomes of the platform. here

7.4 Video Demonstration

Here we will find a quick look at the project "Demo"