## **A logo with a globe and a graduation cap AI-generated content may be incorrect.**

**Benha University**

**Faculty of Computers &**

**Artificial Intelligence**

**E-Commerce Platform with Automated CI/CD Pipeline**

Building a Scalable Laptop Marketplace with Modern DevOps Practices

**DEVOPS TRACK**

**HRV478N\_ONL2\_SWD1\_G2\_DEPI2**

**Project Team**

1- Adham Ashraf Ebrahim ElGanzouri

2- Khaled Abdelmohsen Sayed Mohammed

3- Omar Eid shaban mohamed

4- Mahomoud Mohamed ahmed mohamed

**Under Supervision of**

**Eng/ Ahmed abd-elfatah**

**Feb 2025**

## **1** **Project Planning & Management**

## **1.1 Project Proposal**

**Objective**:  
Develop a scalable **E-Commerce Platform** for selling laptops, supported by an **automated CI/CD pipeline** to ensure efficient development, testing, and deployment.

**Scope**:

1. **E-Commerce Platform**:
   * Build a user-friendly platform with authentication, product management, shopping cart, and order tracking.
   * **Tech Stack**: React.js (Frontend), Express.js (Backend), MongoDB (Database), Material-UI (UI).
   * **Future Enhancements**: Admin panel, payment gateway integration, and product reviews.
2. **CI/CD Pipeline**:
   * Automate the deployment of the e-commerce platform using **Jenkins, Docker, and Ansible**.
   * Ensure seamless integration of code changes, testing, and cloud deployment (AWS/GCP).
   * Optional Kubernetes orchestration for scalable container management.

**Key Integration**:

* The CI/CD pipeline will directly support the e-commerce platform’s development workflow, enabling rapid iteration, testing, and deployment to production.

## **1.2 Project Plan**

## **1.2.1 Gantt chart**

A screenshot of a project

AI-generated content may be incorrect.

## **1.3 Task Assignment & roles**

|  |  |
| --- | --- |
| Role | Responsibilities |
| **Frontend Team** | Develop React UI (product listings, cart), integrate with backend APIs. |
| **Backend Team** | Build Express.js APIs (auth, product/cart logic), MongoDB integration. |
| **DevOps team** | Configure Jenkins pipeline, Dockerize app, write Ansible playbooks for deployment. |

## **1.4 Task Assessment & Mitigation Plan**

|  |  |  |
| --- | --- | --- |
| Risk | Impact | Mitigation Strategy |
| **API/DB Performance Issues** | Slow page loads, user drop-off | Optimize MongoDB queries, implement caching, load-test APIs during CI/CD pipeline runs. |
| **Pipeline Integration Failures** | Deployment delays | Add automated rollback to Jenkins jobs; use staging environments for pre-deployment testing. |
| **Security Vulnerabilities** | Data breaches | Encrypt JWT tokens, sanitize inputs, conduct security scans in the CI pipeline. |
| **Cloud Deployment Errors** | Downtime | Use Ansible idempotent playbooks; monitor with tools like Prometheus/Grafana. |

## **1.4 KPIs (Key Performance Indicators)**

|  |  |  |
| --- | --- | --- |
| Category | Metrics | Target |
| **Platform Performance** | - API response time - System uptime | <500ms 99.9% uptime |
| **User Adoption** | - Registration-to-purchase conversion rate | 80% |
| **Pipeline Efficiency** | - Deployment frequency - Build success rate | 5+ deploys/week 95% success rate |

### **Why This Approach?**

As an expert, I’ve merged both projects into a **single workflow** to emphasize:

1. **End-to-End Ownership**: The CI/CD pipeline is not a standalone project but a critical enabler for the e-commerce platform’s success.
2. **Agility**: Automated testing/deployment reduces manual errors and speeds up feature delivery.
3. **Scalability**: Docker and Kubernetes ensure the platform can handle traffic spikes post-launch.

This structure aligns with industry standards (e.g., Agile-DevOps integration) and ensures stakeholders see the pipeline as integral to the product—not a separate effort. Let me know if you’d like to refine further!

## **2** **Literature Review**

## **2.1 Feedback & Evaluation**

Lecturer’s assessment:

## **Suggested Improvements**

Areas for improvement

## **Final Grading Criteria**

|  |  |  |
| --- | --- | --- |
| Category | Description | Evaluation |
| **Technical Implementation** | Functionality of E-commerce features (auth, cart, checkout) and CI/CD pipeline. |  |
| **CI/CD Pipeline** | Jenkins automation, Docker/Ansible integration, and successful cloud deployment. |  |
| **Documentation** | Clarity, structure, and depth of technical and project management documentation. |  |
| **Testing** | Creativity in solving challenges (e.g., UI design, pipeline optimization). |  |
| **Presentation** | Professional delivery of demos, diagrams, and responses to feedback. |  |

Overall Evaluation:

## **3 Requirement gathering**

## **3.1 Stakeholder analysis**

|  |  |
| --- | --- |
| Stakeholders | Needs/Expectations |
| **Customers** | - Intuitive UI for browsing and purchasing laptops. - Secure payment and order tracking. |
| **Administrators** | - Tools to manage products, orders, and user accounts. - Analytics for sales trends. |
| **Developers** | - Scalable architecture for future enhancements. - Automated CI/CD workflows. |
| **DevOps Team** | - Reliable deployment pipeline with rollback capabilities. - Monitoring for uptime. |
| **Business Owners** | - High user adoption rates. - Compliance with data protection laws (e.g., GDPR). |

## **3.2 User Stories & Use Cases**

* **User Story**:  
  As a customer, I want to filter laptops by price and specifications so that I can find products within my budget.
* **Use Case**:
  + **Actor**: Customer.
  + **Description**: Browse, filter, and view product details.
  + **Steps**:
    1. Navigate to the homepage.
    2. Use filters (price range, RAM, brand).
    3. View product details and add to cart.

**2. DevOps Pipeline**

* **User Story**:  
  As a DevOps engineer, I want Jenkins to automate testing and deployment so that code changes are deployed without downtime.
* **Use Case**:
  + **Actor**: DevOps Team.
  + **Description**: Deploy updates via CI/CD pipeline.
  + **Steps**:
    1. Push code to Git repository.
    2. Jenkins triggers build, runs tests, and deploys to AWS/GCP.
    3. Monitor deployment status via Jenkins dashboard.

## **3.3 Functional Requirements**

|  |  |
| --- | --- |
| Feature | Description |
| **Authentication** | - Users can register/login with email and password. - JWT tokens for session management. |
| **Product Management** | - Admins can add/edit/delete products. - Customers can view product details and reviews. |
| **Shopping Cart** | - Add/remove items, adjust quantities. - Persist cart data across sessions. |
| **Checkout** | - Secure payment gateway integration (future). - Order confirmation emails. |
| **Order History** | - Customers can view past orders with status (delivered/pending). |
| **CI/CD Pipeline** | - Automated builds, tests, and deployments via Jenkins. - Ansible-managed cloud provisioning. |

## **3.4 Non-Functional Requirements**

|  |  |
| --- | --- |
| Feature | Description |
| **Performance** | - API response time < 500ms. - Handle 1,000 concurrent users. |
| **Security** | - Encrypted passwords (bcrypt). - HTTPS for all transactions. |
| **Usability** | - Mobile-responsive UI (Material-UI). - WCAG 2.1 compliance for accessibility. |
| **Reliability** | - 99.9% system uptime. - Automated rollback on deployment failures. |
| **Reliability** | - Docker containers for horizontal scaling. - Kubernetes support (future). |
| **Maintainability** | - Modular code structure (React/Express). - Comprehensive API documentation. |

**4. System Analysis & Design**

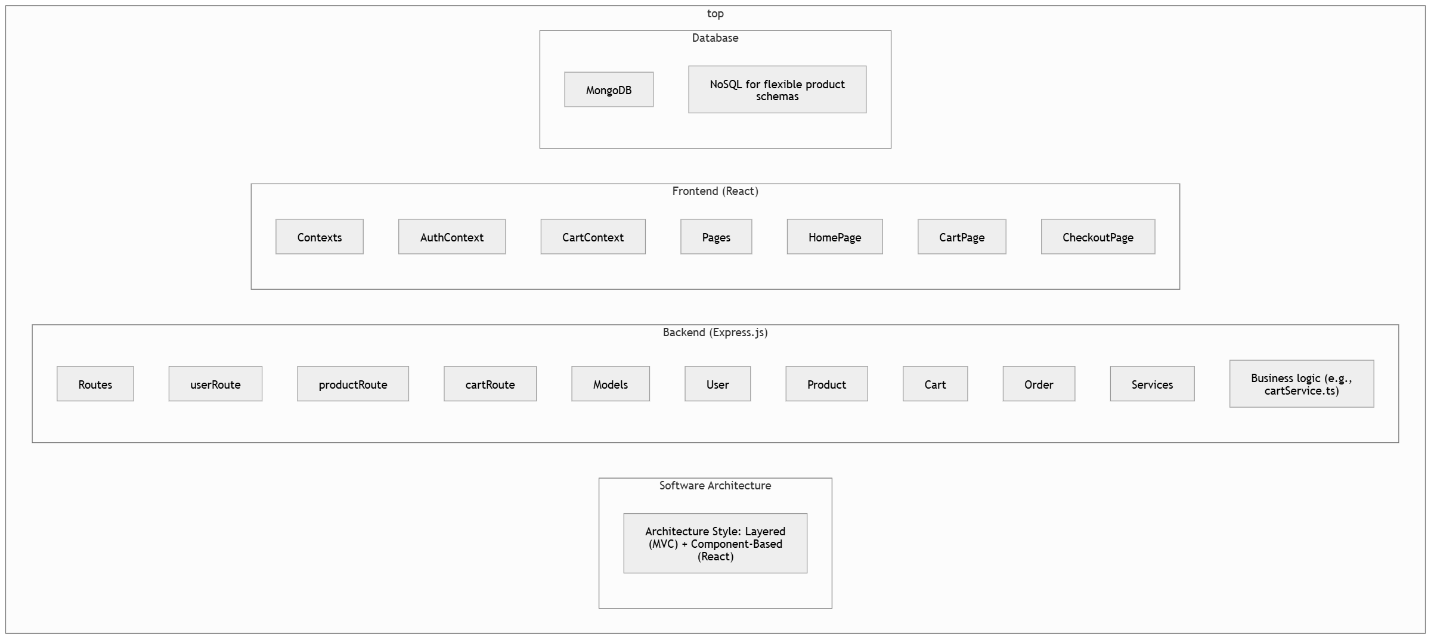
### **4.1 Problem Statement & Objectives**

* **Problem Statement**: Existing e-commerce platforms may not cater specifically to laptop sellers, leading to fragmented product discovery, insecure transactions, and inefficient order tracking.
* **Objectives**:
  + Provide a secure authentication system for users.
  + Display laptops with detailed specifications and pricing.
  + Enable users to manage a shopping cart and checkout seamlessly.

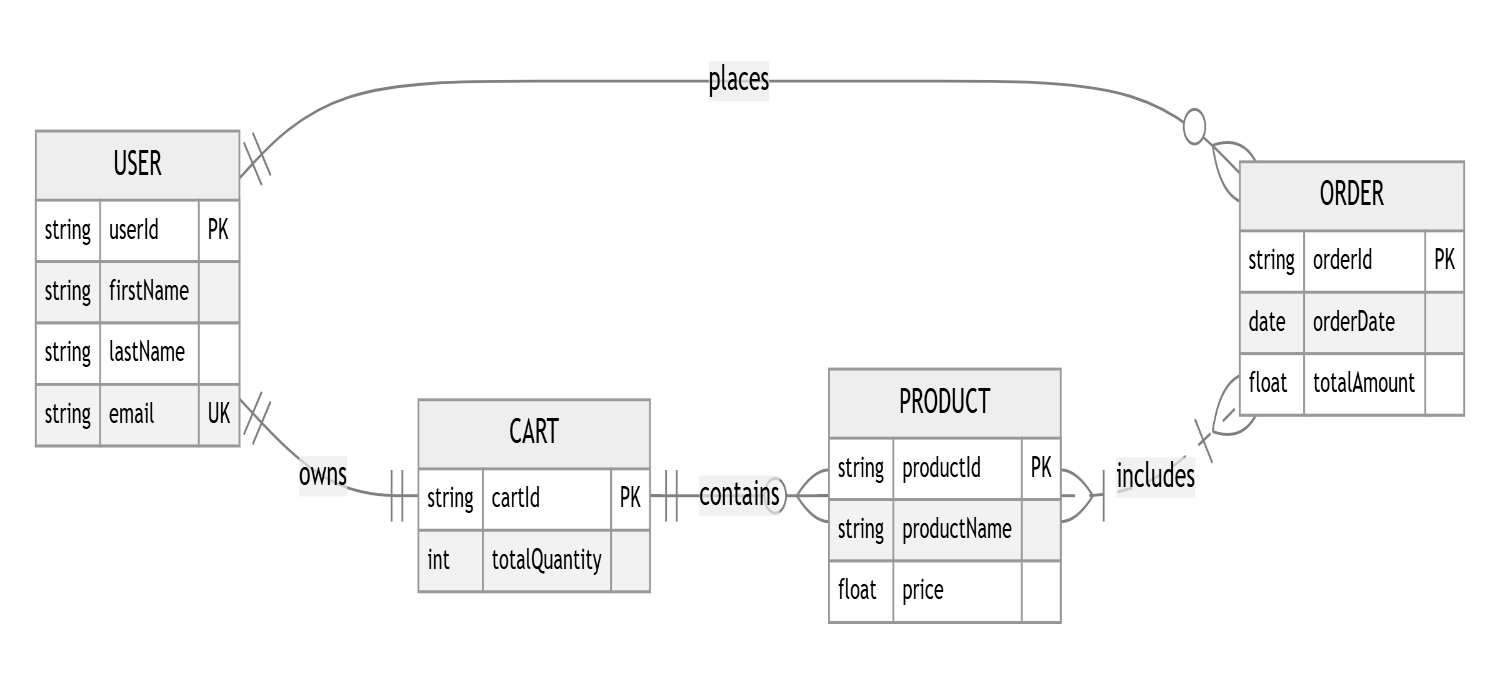
### **4.2 Use Case Diagram & Descriptions**

### A diagram of a user flow AI-generated content may be incorrect.

### **4.3 Software Architecture**



## **4.4 ER Diagram**

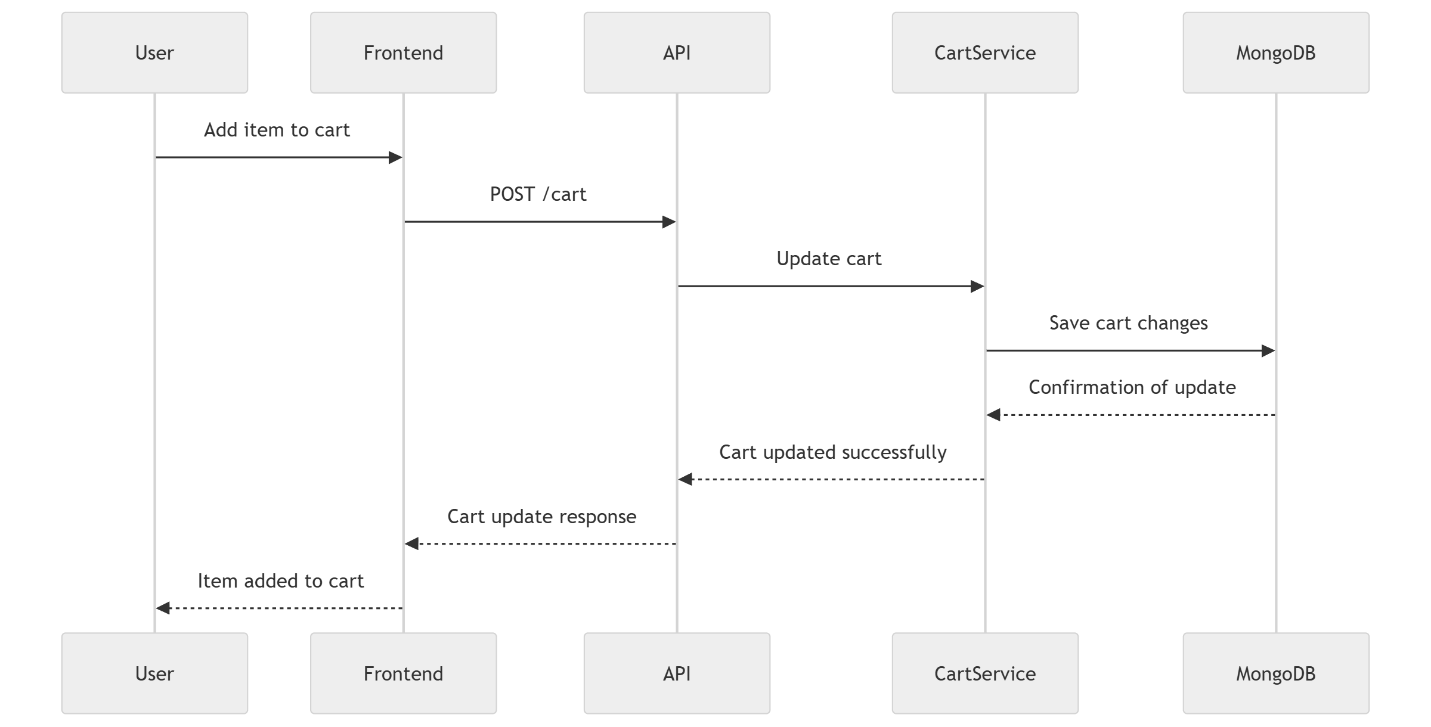


## **4.5 DFD Diagram**

A diagram of a product

AI-generated content may be incorrect.

**4.6 Sequence Diagram**

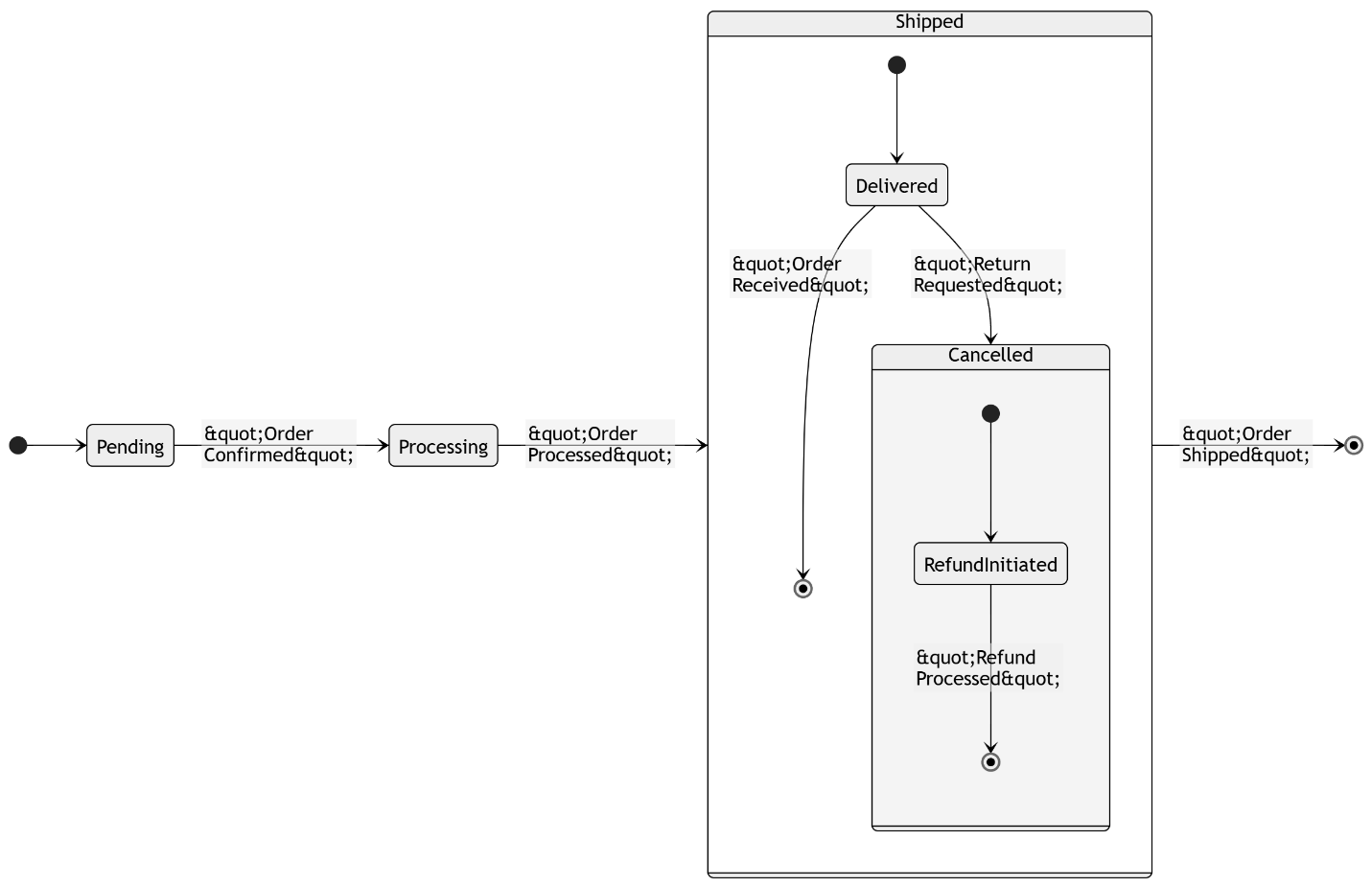
****

**4.7 Activity Diagram (checkout)**

**A diagram of a computer

AI-generated content may be incorrect.**

## **4.8 State Diagram**



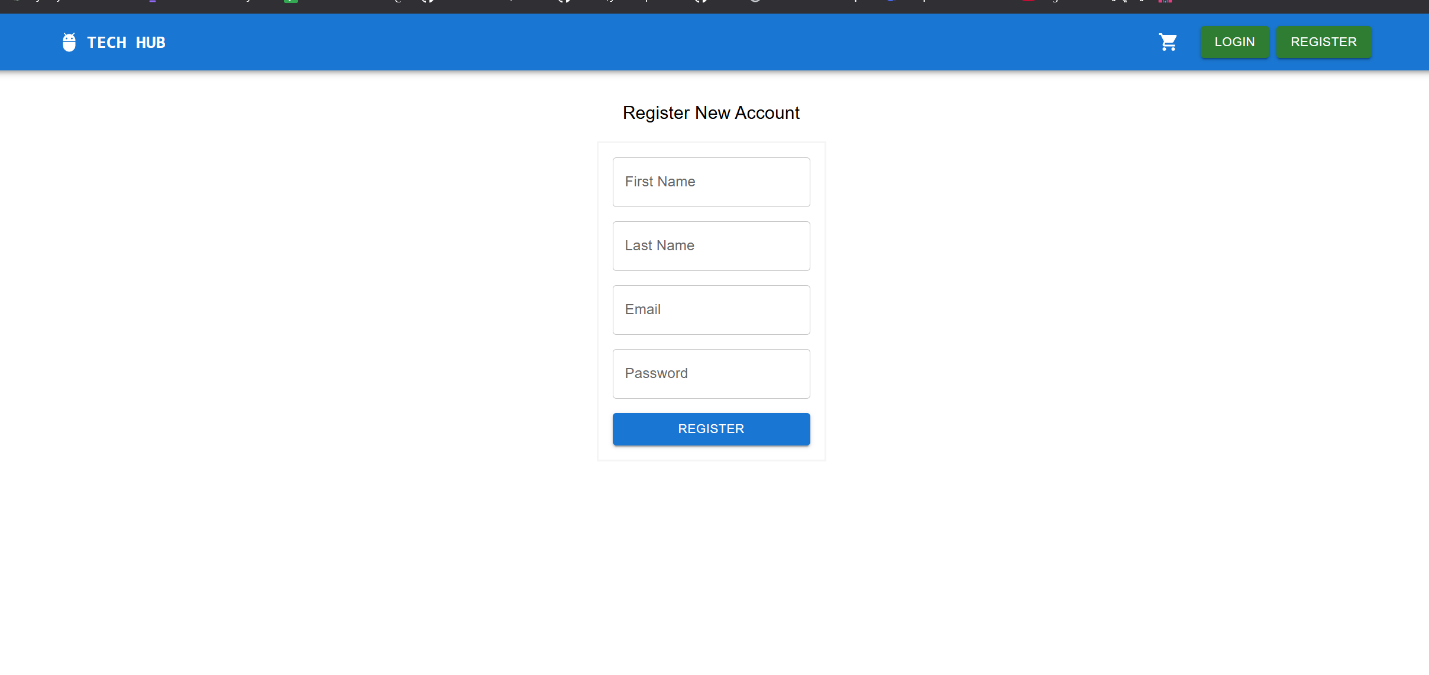
## **4.9 Class Diagram**

A diagram of a product

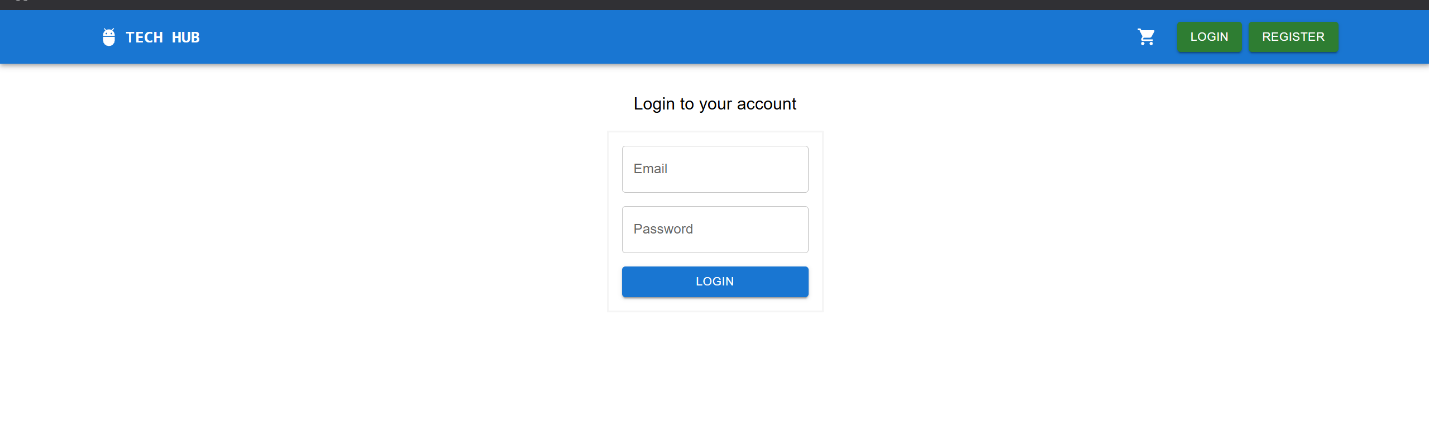
AI-generated content may be incorrect.

## **5 UI/UX Design & Prototyping**

**Register Page**

****

**Login Page**

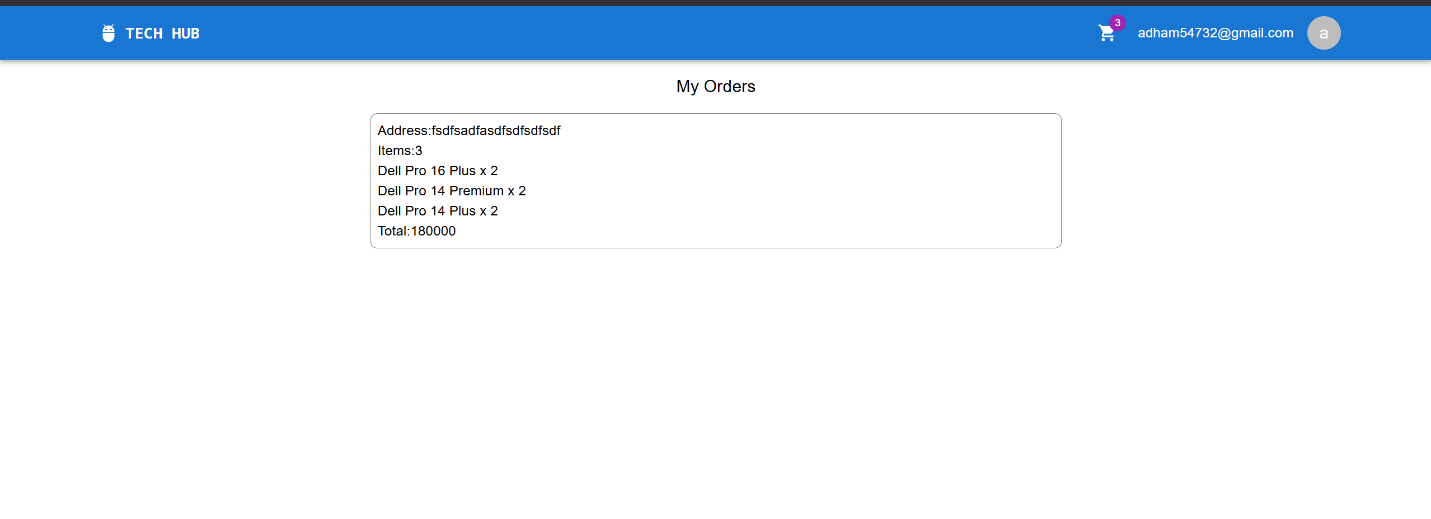


**Main Page**

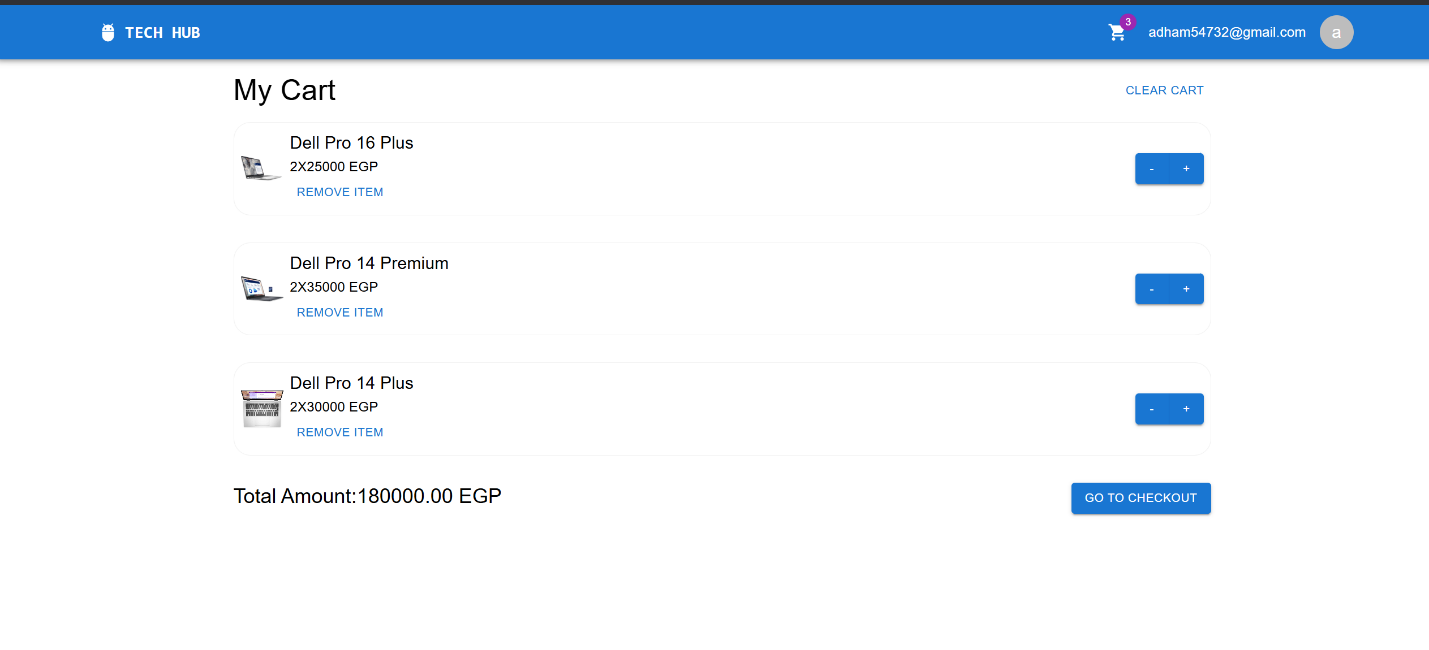
A screenshot of a computer

AI-generated content may be incorrect.

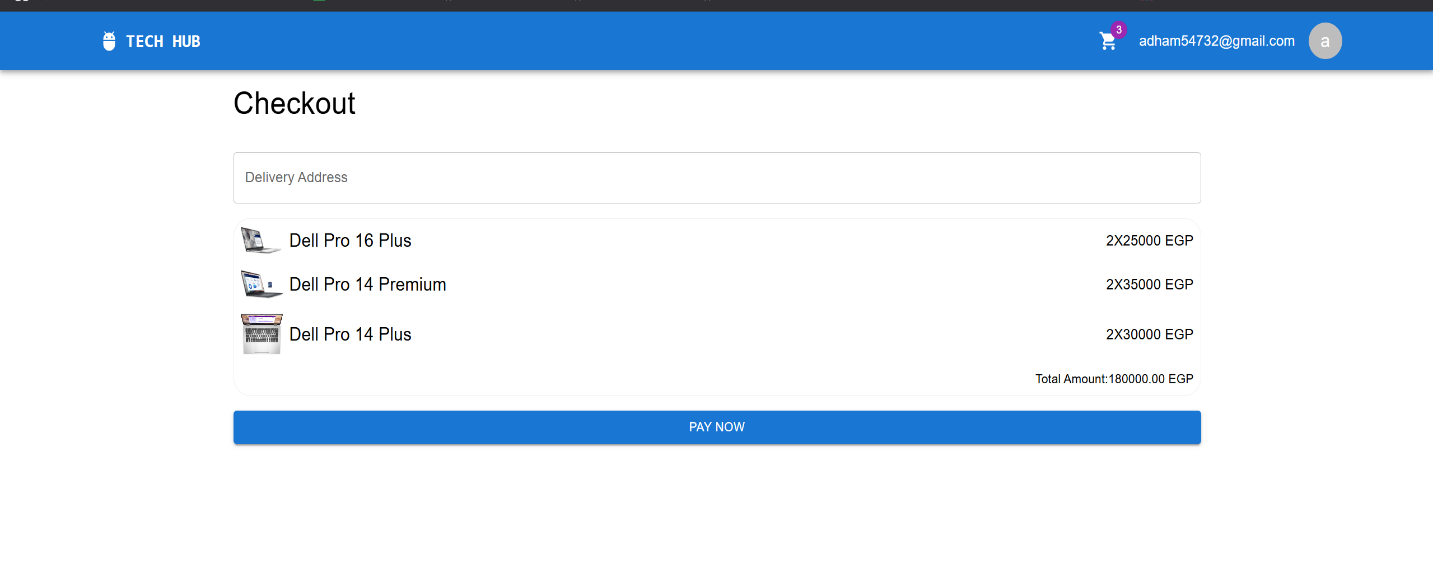
**Orders Page**



**My Cart Page**

****

**Checkout Page**



## **6 System Deployment & Integration**

**6.1 Technology Stack**

|  |  |
| --- | --- |
| **Category** | **Technologies & Tools** |
| **Frontend**  **Frontend** | **React.js (TypeScript), Material-UI (MUI), Vite, Axios (API calls), React Router** |
| **Backend** | **Express.js (TypeScript), Node.js, JWT (authentication), Mongoose (MongoDB ODM)** |
| **Database** | **MongoDB Atlas (Cloud-hosted NoSQL)** |
| **DevOps** | **Jenkins (CI/CD), Docker (containerization), Ansible (configuration management), Kubernetes (orchestration)** |
| **Hosting/Cloud** | |  | | --- | | **AWS EC2 (backend/frontend servers), MongoDB Atlas (database), Docker Hub (image registry)** | |  | |
| **APIs** | **RESTful APIs for authentication, product management, cart, and order operations** |

**6.2 Deployment Diagram**

**A screenshot of a computer

AI-generated content may be incorrect.**

**6.3** **Component DiagramA screenshot of a computer screen

AI-generated content may be incorrect.**