**A**
**PROJECT REPORT ON**

# QuickTradeHub - A Smart Reselling Platform

SUBMITTED IN
PARTIAL FULFILLMENT OF

**DIPLOMA IN ADVANCED COMPUTING (PG-DAC)**


**BY**

**Mr. Abhishek Kute**

**Mr. Pravin Torave**

**Ms. Sakshi Poshattiwar**

**Ms. Dhanashri Shinde**

**UNDER THE GUIDENCE OF**
**Lalita Shinde**

**AT**
**SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, PUNE**

**SUNBEAM INSTITUTE OF INFORMATION TECHNOLOGY, PUNE.**


# CERTIFICATE

This is to certify that the project

**QuickTradeHub - A Smart Reselling Platform**

Has been submitted by

**Abhishek Kute**
**Pravin Torave**
**Sakshi Poshattiwar**
**Dhanashri Shinde**

In partial fulfillment of the requirement for the Course of **PG Diploma in Advanced Computing (PG-DAC AUG 2024)** as prescribed by The **CDAC** ACTS, PUNE.

Place: Pune Date: 11 Feb 2025

**Lalita Shinde**
**Project Guide**

# ACKNOWLEDGEMENT

We would like to take this opportunity to express our heartfelt gratitude to those who have guided and supported us throughout the journey of this project. A special thanks to Mr. Nitin Kudale (Center Coordinator, SIIT, Pune) and Mr. Yogesh Kolhe, Course Coordinator at SIIT Pune, for their invaluable guidance, unwavering support, and continuous encouragement. Their expertise, constructive feedback, and insightful suggestions at every stage of the project were instrumental in shaping it to its current form.

We are also deeply thankful to the entire faculty and staff members of Sunbeam Institute of Information Technology, Pune, for their support and cooperation.

This project would not have been possible without their assistance, and we are truly grateful for their contributions.

Pravin Torave , Sakshi Poshittiwar , Dhanashri shinde

Abhishek Kute
87009 PG-DAC
SIIT Pune

# ABSTRACT

The project Quick Trade Hub is an online marketplace platform designed providing users with a simple and efficient way to buy, sell, or trade goods and services. The platform enables individuals to list items for sale, negotiate prices, and engage in secure transactions within a localized environment.

The primary objective of Quick Trade Hub is to create an intuitive, user-friendly interface where sellers can post listings and buyers can browse through various categories to find products or services they need. The platform supports key features such as product listing management, buyer-seller communication, location-based filtering, price negotiation tools, and a secure payment gateway to ensure smooth transactions.

The project focuses on optimizing user experience, improving search and recommendation algorithms, and ensuring a secure environment for all participants. Additionally, the platform includes features for user verification, reviews, and ratings, making it reliable and trustworthy for both buyers and sellers.

This report outlines the design, development, and testing processes involved in creating Quick Trade Hub, highlighting the technologies used, challenges encountered, and the steps taken to ensure scalability and performance.

# INDEX

# LIST OF FIGURES

# 1. Introduction

Quick Trade Hub is an online marketplace platform designed to streamline the buying, selling, and trading of goods and services. Inspired by OLX, the platform allows users to list items for sale, browse through various categories, and engage in secure transactions. It offers a user-friendly interface where sellers can post products with descriptions, prices, and images, while buyers can easily search, filter, and find items of interest. The platform's core features include product management, buyer-seller communication, price negotiation, and a secure payment system. The project employs a microservices architecture with dedicated services for authentication, product listings, order management, and notifications, ensuring scalability and maintainability. By integrating modern technologies like Java Spring Boot, Node.js, and C#, Quick Trade Hub aims to create a seamless and reliable environment for online trading, making it an efficient alternative for local commerce.

# 2. PRODUCT OVERVIEW AND SUMMARY

## 2.1 Purpose

The primary purpose of Quick Trade Hub is to create a modern, efficient, and secure online marketplace where users can engage in buying, selling, and trading goods and services. The key purposes are outlined below:

- Simplified Buying and Selling:

  The platform enables users to easily list items for sale or browse products to buy, offering a seamless and intuitive interface for both sellers and buyers.

- Local Commerce Support:

The marketplace is designed to support local commerce, allowing users to search for products based on their geographical location. This fosters community-based trade and makes the exchange process faster and more convenient. Sellers can easily upload product details, including images, descriptions, and prices, and

manage their inventory. Buyers can use search and filter features to quickly find products that meet their needs.

Ensuring transaction safety is a core goal. Quick Trade Hub integrates a secure payment gateway, allowing users to make payments and complete transactions without fear of fraud or loss of data. To build trust among users, the platform implements a robust rating and review system. Buyers can leave reviews about products and sellers, helping others make informed decisions.

The platform provides a direct communication channel between buyers and sellers, facilitating price negotiations and transaction discussions. With its microservices architecture, Quick Trade Hub ensures flexibility and scalability, catering to an expanding user base and diverse product categories, making it adaptable for future enhancements. These purposes collectively drive Quick Trade Hub to be a comprehensive and user-centered online marketplace solution.

## 2.2 Scope

The scope of Quick Trade Hub covers three primary user roles: Sellers, Buyers, and Admins. Each role has distinct functionalities and responsibilities within the platform, ensuring a balanced and effective marketplace experience.

- **Scope for Sellers**

1. It can create, update, and manage their product listings, including uploading images, setting prices, and writing detailed descriptions.
2. It can monitor and manage their inventory, including updating stock levels and marking products as sold.

3.It can directly communicate with buyers through the platform, negotiate prices, and finalize deals.

4.It can receive feedback from buyers in the form of reviews and ratings, which contribute to their reputation on the platform

- **Scope for Buyers**

1 .Buyers are the key users of the platform, and the system is designed to make their experience simple and efficient.

2 .It can search and filter products by categories, prices, and locations to find items that match their needs.

3. It can read reviews and ratings from other users to help them make informed purchasing decisions.

4. Buyers can initiate communication with sellers to inquire about products, negotiate prices, or arrange transactions.

5. Buyers can add products to their cart, place orders, and track the status of their purchases through the platform.

The platform provides secure payment gateways, allowing buyers to make payments safely.

- **Scope for Admins:**

1. Admins have control over user accounts, with the ability to manage user registrations, verify accounts, and handle any disputes between buyers and sellers.
2. They are monitor product listings to ensure they meet platform guidelines, removing inappropriate or fraudulent listings.
3. IT can track and oversee transactions, ensuring compliance with security standards and resolving payment issues if necessary.
4. can access detailed reports and analytics on platform activity, including user engagement, product listings, sales, and more.
5. Admins can manage promotional campaigns, send system-wide notifications, and handle marketing activities to boost user engagement.

### 2.3 Design and Implementation Constraints

The Quick Trade Hub project is developed with specific design and implementation constraints to ensure security, scalability, and efficiency. These constraints define the limitations and considerations that guide the system's architecture and functionality.

1. Design Constraints

These constraints influence how the system is structured and built.

I. Technology Stack Limitation:

- Backend:

a. The Authentication service is implemented using Java Spring Boot with Spring Security for user authentication and authorization.

b. The Product service is developed using Node.js with Express.js to handle product listings and management.

c. The Order service is built using *C# Web API* to manage order creation, cart, and payment tracking.

- Frontend:

a. The user interface is designed with React.js and Bootstrap to ensure a responsive, user-friendly experience.
b. Database Management:
c. MySQL/PostgreSQL for user data, authentication, and order management.
d. MongoDB is used for flexible, scalable product listings and inventory management.

- Security Considerations:

a.Data transmission over HTTPS to enhance security.

b.Role-Based Authentication and Authorization.

c. Implemented using Spring Security and JWT for stateless session management.

d.Password Security

e.Passwords are hashed using *BCRYPT* to prevent unauthorized access

f. Data Transmission

g. Communication between services and the frontend is secured

2. Implementation Constraints

- .Integration Constraints

a.Payment Gateway Integration

b. Integration with third-party payment gateways (e.g., Azure) to handle secure transactions.

c.Notification Services

d.Nodemailer are used for email and SMS notifications for order updates and promotional messages.

e.Inter-Service Communication

f.RESTful APIs are used for communication between microservices.

- User Roles and Access Control
  - Buyer and Seller Roles:- Buyers can search and purchase products, while sellers can create and manage product listings. Both roles have restricted access based on their permissions.
  - Admin Role: - Admins have full control over the platform, including user management, product moderation, and transaction oversight.

- Scalability & Maintainability

1. Microservices Architecture .

a.The platform uses a microservices architecture to allow independent development, scaling, and maintenance of individual services.

b.Database Scalability .

c.The database schema supports horizontal scaling and efficient querying as the platform expands.

d.RESTful API Design.

e.The API is designed for easy integration with future third-party services and system extensions.

These constraints define technical limitations and restrictions during system development and deployment.

- Database Constraints

a.Each service has its own database, requiring distributed data management and consistency between services.

- ACID Compliance

a.The MySQL/PostgreSQL databases must ensure ACID compliance for reliable transactions and data integrity.

b.Handling Large-Scale Data.

c.The platform must efficiently manage large volumes of product listings, orders, and user data.

3.Performance Constraints:

- Query Optimization: - The system must be optimized for fast query execution, especially for searching and filtering large product datasets.
- Real-Time Notifications:- The platform must handle real-time notifications for order updates and communication between buyers and sellers without affecting overall performance.
- Concurrency

a.The platform should handle multiple concurrent user interactions, particularly during high traffic periods.

4. Deployment Constraints:

- Cloud Deploymen:- The system should be deployable on cloud infrastructure (e.g., AWS) with support for containerization (e.g., Docker) to ensure scalability.
- CI/CD Pipeline :- A *CI/CD pipeline* is recommended for continuous integration and delivery of updates, ensuring bug fixes and new features are rolled out efficiently.

5.Device & Browser Compatibility:

- Responsive Design: - The web application must be responsive and function seamlessly across different device screen sizes (e.g., desktop, mobile, tablet).
- Browser Compatibility:- The platform should be compatible with all modern browsers, including Chrome, Firefox, Edge, and Safari.

# 3 REQUIREMENTS

## 3.1 Functional Requirements

Functional requirements define the specific operations and features the system must support to fulfill user needs. The system caters to three primary user roles: Administrators, Sellers, and Buyers.

### 3.1.1 Use Case for the Administrator

The Administrator is responsible for managing users, overseeing platform operations, and maintaining the security and functionality of the system.

- Use Case: Manage Users

  Actors: Administrator
  Preconditions: The administrator must be logged in with appropriate privileges.
  Flow of Events:

1. View the list of registered users (sellers and buyers).

2. Ban or restrict users violating platform policies.

3. Grant or revoke user access and permissions as needed.

Postconditions: Users are managed effectively, ensuring compliance with platform guidelines and maintaining a safe environment for all participants.

- Use Case: Monitor Business Analytics

  Actors: Administrator

  Preconditions The system must have data on product listings, transactions, and user activity.

Flow of Events:

1. Access reports on product listings, sales, and user activity.

2. View insights such as top-selling products, buyer trends, and peak transaction times.

3. Generate analytical reports for business decision-making.

Postconditions: The administrator can make informed decisions based on analytics, driving improvements in platform performance and user experience.

Use Case: System Maintenance & Security Management
Actors: Administrator
Preconditions The administrator must have security privileges.

Ensure system uptime and monitor performance.

Monitor system uptime and performance.

2. Configure and enforce security policies (e.g., password rules, authentication methods)

3. Manage API integrations, such as payment gateways and notification services.

Postconditions: The system remains secure, functional, and up-to-date, ensuring a safe and reliable marketplace for users.

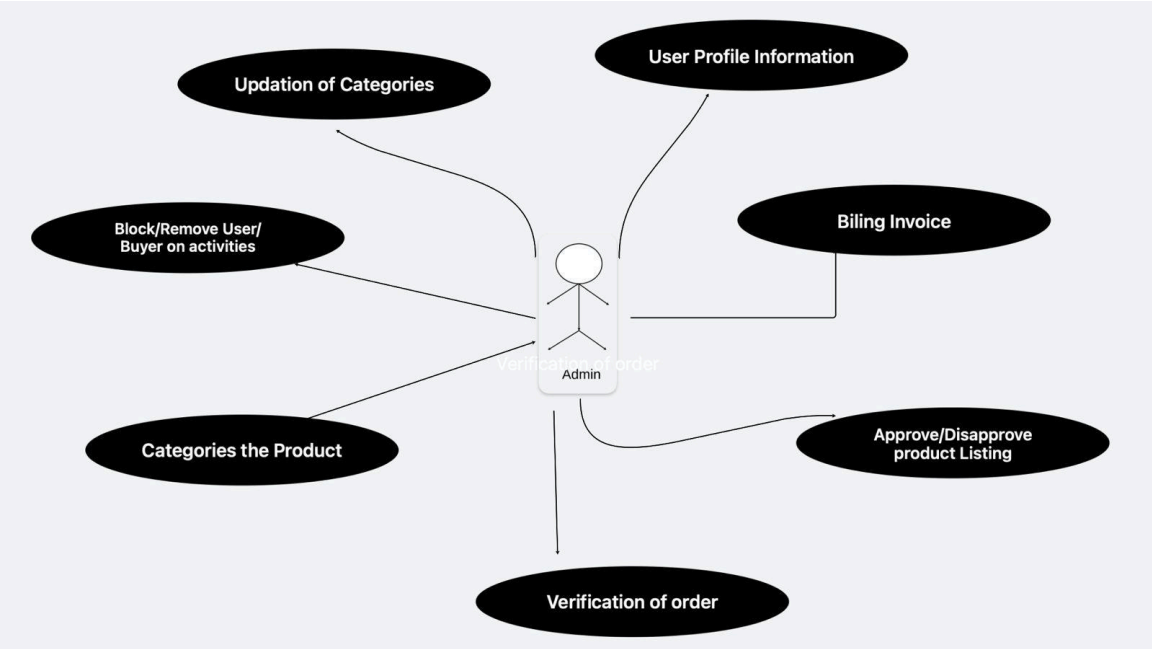


Figure 1 Use Case For Admin

### I.1.2 Use Case for the Seller

The Seller is a user who lists products for sale on the platform, manages their listings, and interacts with potential buyers

- Use Case: Register and Authenticate

  Actors: Seller

  Preconditions : The seller must provide valid credentials.

Flow of Events:

1. Sign up using email and password.
2. Verify email using a Nodemailer-based email verification system.
3. Log in securely via Spring Security with JWT-based authentication.

Postconditions: The seller gains access to their account and can start listing products for sale.

- Use Case: Create and Manage Product Listings

Actors: Seller

Preconditions: The seller must be logged in to their account.

- Flow of Events:

1. Create a new product listing by uploading product details, images, and pricing information.

2. Edit or update product details as needed (e.g., changing prices or updating stock).

3. Manage product inventory, marking items as sold or out of stock.

Postconditions: Products are listed and managed efficiently, enabling sellers to market their goods to potential buyers.

- Use Case: Communicate with Buyers

Actors: Seller

Preconditions: The seller must have an active product listing and a buyer inquiry.

Flow of Events:

1. Receive inquiries from buyers regarding listed products.

2. Respond to buyer messages, negotiate prices, and provide additional product information.

3. Finalize terms and prepare for the transaction.

Postconditions: Communication between sellers and buyers facilitates the transaction process and helps in finalizing deals.

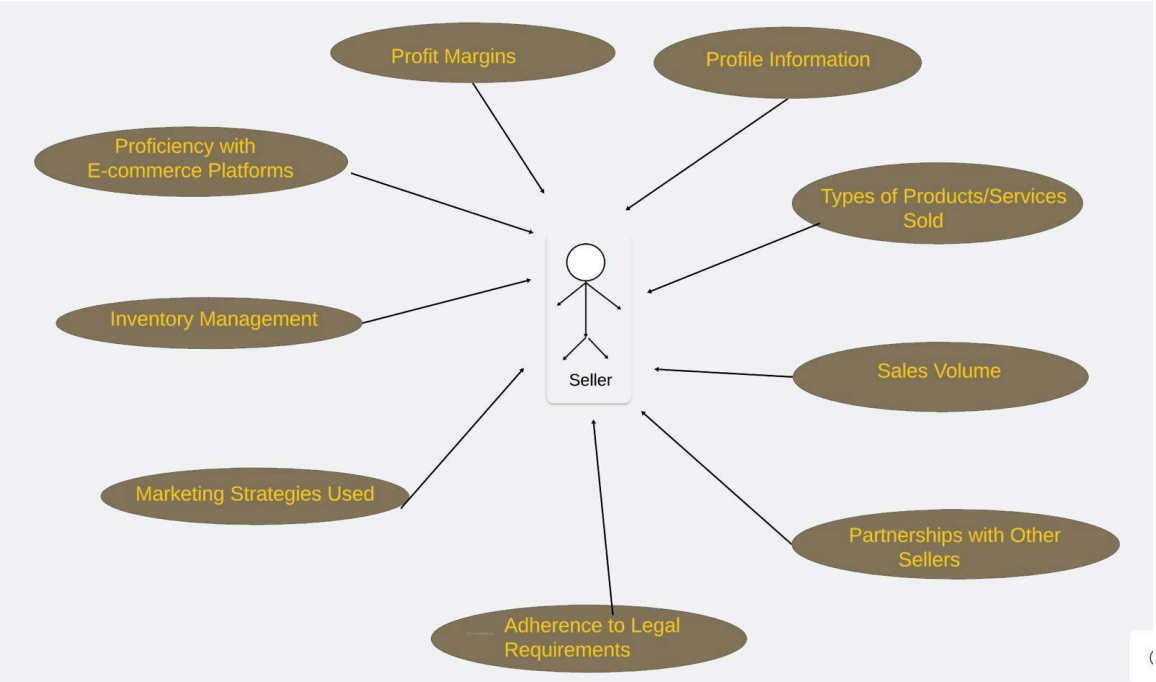


Figure 2 Use Case For seller

### 3.1.3 Use Case for the Buyer

The Buyer is the primary user who searches for products, negotiates with sellers, and completes purchases on the platform.

- Use Case: Register and Authenticate

Actors: Buyer

Preconditions: The buyer must provide valid credentials.

Flow of Events:

1. Sign up using email and password.

2. Verify email through a Nodemailer-based verification system.

3. Log in securely using JWT-based authentication.

Postconditions: The buyer gains access to their account and can begin browsing and purchasing products.

- Use Case: Search and Purchase Products

Actors: Buyer

Preconditions: The buyer must be logged in.

Flow of Events:

1. Search for products using keywords or filters such as category, price, and location.
2. View product listings, including descriptions, reviews, and ratings.
3. Select a product and initiate communication with the seller for negotiation.
4. Confirm the purchase and complete the transaction through a secure payment system.

Postconditions: The product is purchased, and the buyer receives a confirmation notification for the transaction.

- Use Case: Review and Rate Products

Actors: Buyer

Preconditions: The buyer must have completed at least one purchase.

Flow of Events:

1. Select a past purchase and submit a review for the product.
2. Rate the seller based on the quality of service, product condition, and transaction experience.

3. View other customer reviews before making new purchase decisions.

Postconditions: User reviews contribute to overall product and seller quality, helping future buyers make informed decisions.

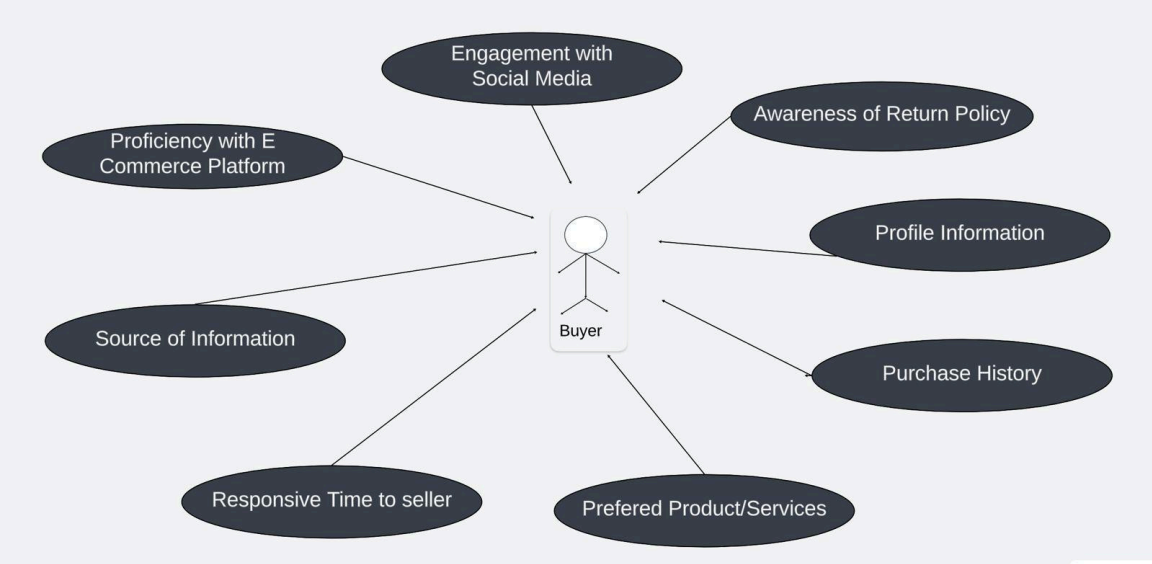


Figure 3: Use Case For buyer

### 3.2 Non-functional Requirements

Non-functional requirements define the quality attributes, system constraints, and overall operational characteristics of Quick Trade Hub. These ensure that the system performs efficiently, securely, and remains scalable for future enhancements.

#### 3.2.1 Usability Requirement

- The system must have an intuitive and user-friendly UI designed with React.js and Bootstrap.
- It should support multiple devices, including desktop, mobile, and tablets, ensuring responsive design across various screen sizes.
- The platform must work seamlessly across modern web browsers such as Chrome, Firefox, Edge, and Safari.

- Users (sellers and buyers) should be able to complete key tasks (searching, listing products, purchasing, and payment) within 3-5 clicks for an efficient user experience.

### 3.2.2 Performance Requirement

- The system should handle multiple concurrent transactions (product listings, searches, purchases) without performance degradation.
- The response time for product search queries and transaction-related actions (e.g., adding items to cart, checking out) should be less than 2 seconds.
- Efficient database queries and caching mechanisms (e.g., Redis) should be implemented to optimize speed and performance, especially during high traffic periods.

### 3.2.3 Reliability Requirement

- The platform should maintain a minimum uptime of 99.9% to ensure high availability and reliability for users.
- All transactions, including product listings and payments, must be ACID-compliant to prevent data inconsistencies and maintain data integrity.
- Auto-recovery mechanisms and backup systems must be in place to recover from potential system failures or downtimes.

### 3.2.4 Portability Requirement

- The system must be cloud-deployable on platforms like AWS, Azure, or any cloud provider, with support for containerization tools like Docker and Kubernetes.
- The backend services should be portable across different operating systems, including Windows, Linux, and macOS, without requiring major modifications.
- The frontend should remain responsive and adaptable to different devices, ensuring usability across varying screen sizes and resolutions.

### 3.2.5 Security Techniques

-Spring Security with JWT will be used to ensure secure authentication and session management for both sellers and buyers.

- BCRYPT password hashing will protect user credentials and prevent unauthorized access.

- HTTPS encryption will be enforced to secure all data transmissions between the frontend and backend services.

- Role-Based Access Control (RBAC) will be implemented to restrict unauthorized actions, ensuring that users only have access to the features available for their role (buyer, seller, admin).

- Regular security audits and vulnerability scans will be conducted to identify and fix potential security threats, ensuring platform safety.

# 4 PROJECT DESIGN

## 4.1 Data Model

### 4.1.1 Database Design

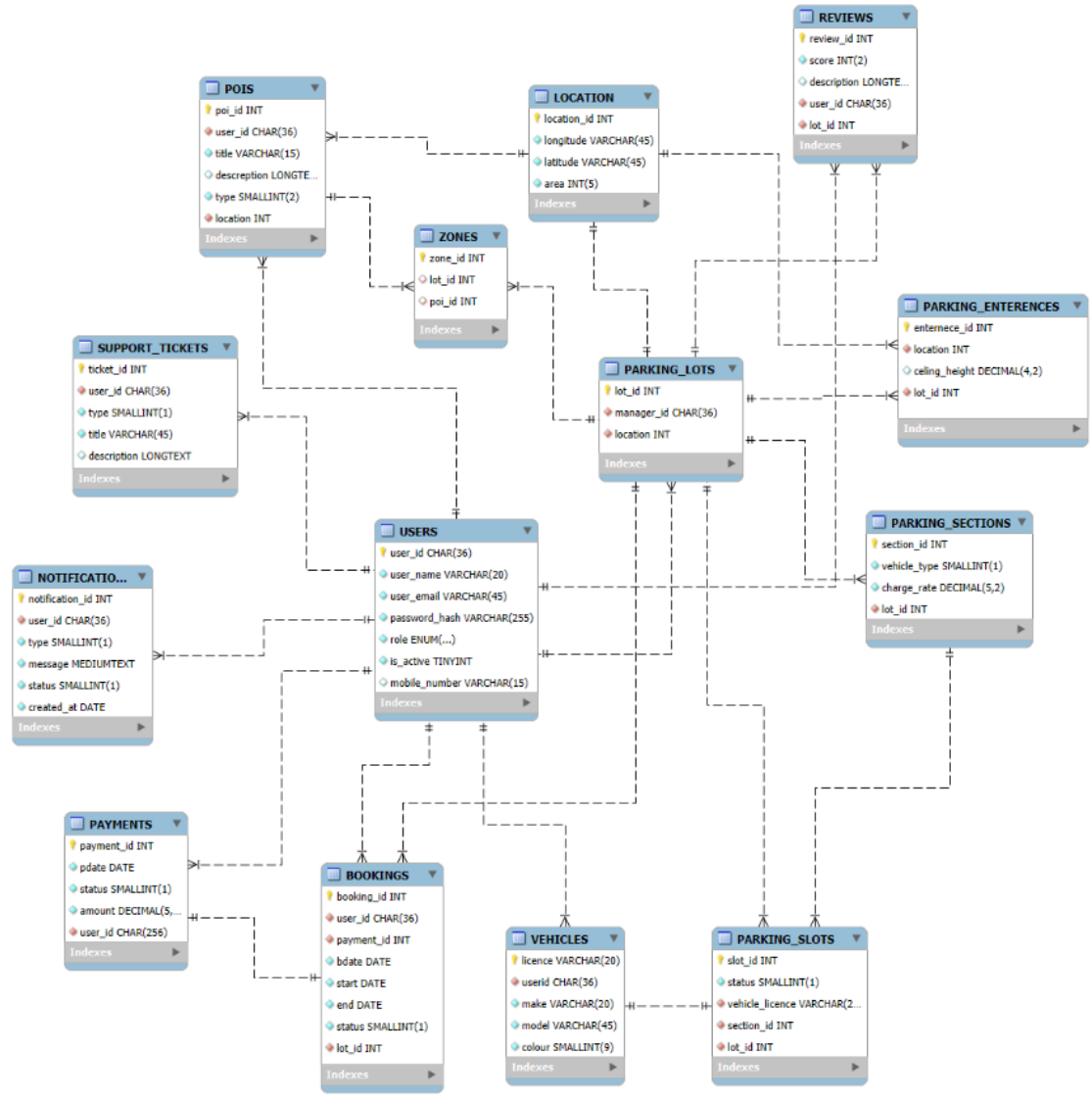


Figure 4 Database Design

# 5 TEST REPORT

| | | | | | | |
|---|---|---|---|---|---|---|
| TC02 | User Login (JWT Authentication) | User registered and email verified | 1. Navigate to login page<br>2. Enter valid credentials<br>3. Submit form | User logged in successfully and JWT token generated | As expected | Pass |
| TC03 | Product Listing by Seller | Seller logged in | 1. Log in as seller<br>2. Navigate to product listing page<br>3. Create and submit a product listing | Product listing created successfully and visible on the platform | As expected | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| TC04 | Product Search by Buyer | Products available | 1. Log in as buyer<br>2. Enter product keywords in search bar<br>3. Apply filters (if necessary) | Search results displayed with relevant product listings and correct filters applied | As expected | Pass |
| TC05 | Add Product to Cart and Checkout | Buyer logged in | 1. Search and select a product<br>2. Add product to cart<br>3. Proceed to checkout<br>4. Complete payment | Product successfully added to cart and payment processed | As expected | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| TC06 | Product Review and Rating by Buyer | Buyer has purchased a product | 1. Log in as buyer<br>2. Navigate to past purchase<br>3. Submit a review and rating | Review and rating submitted and visible for product | As expected | Pass |
| TC07 | Admin User Management (Ban/Restrict User) | Admin logged in with appropriate rights | 1. Log in as admin<br>2. Navigate to user management panel<br>3. Select and ban/restrict a user | User successfully banned/restricted | As expected | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| TC08 | View Business Analytics by Admin | Admin logged in with data available | 1. Log in as admin<br>2. Access business analytics page<br>3. View reports and insights | Business analytics and reports displayed correctly | As expected | Pass |
| TC09 | Email and SMS Notification for Order Updates | Order created and notification service active | 1. Place an order as buyer<br>2. Complete payment<br>3. Check email/SMS for order confirmation | Order confirmation received via email/SMS | As expected | Pass |

# 6 OUTCOME

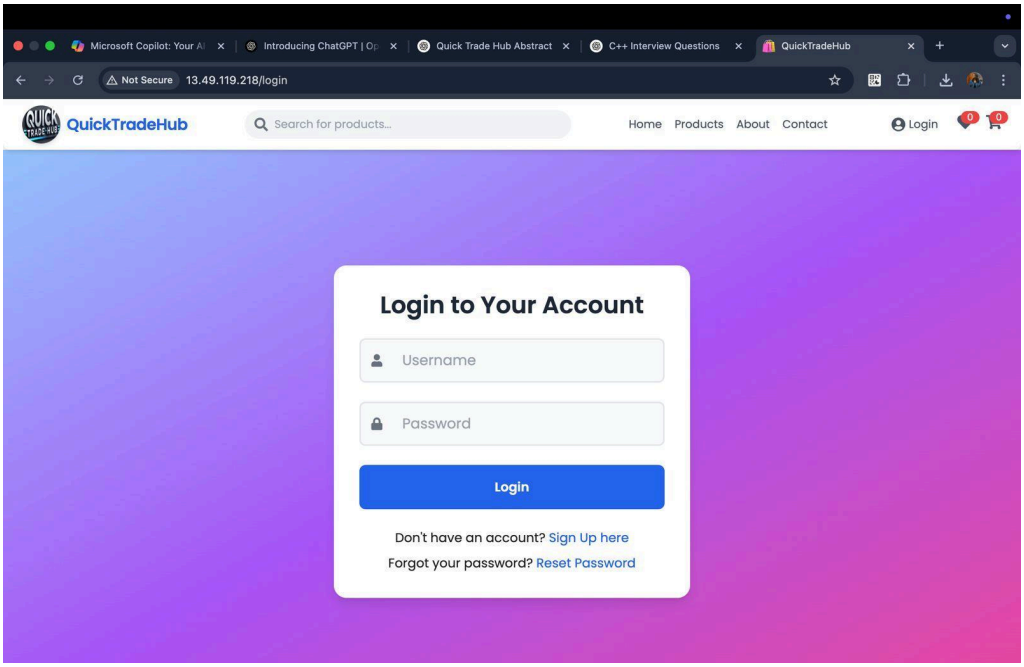


Fig 6.1 login page

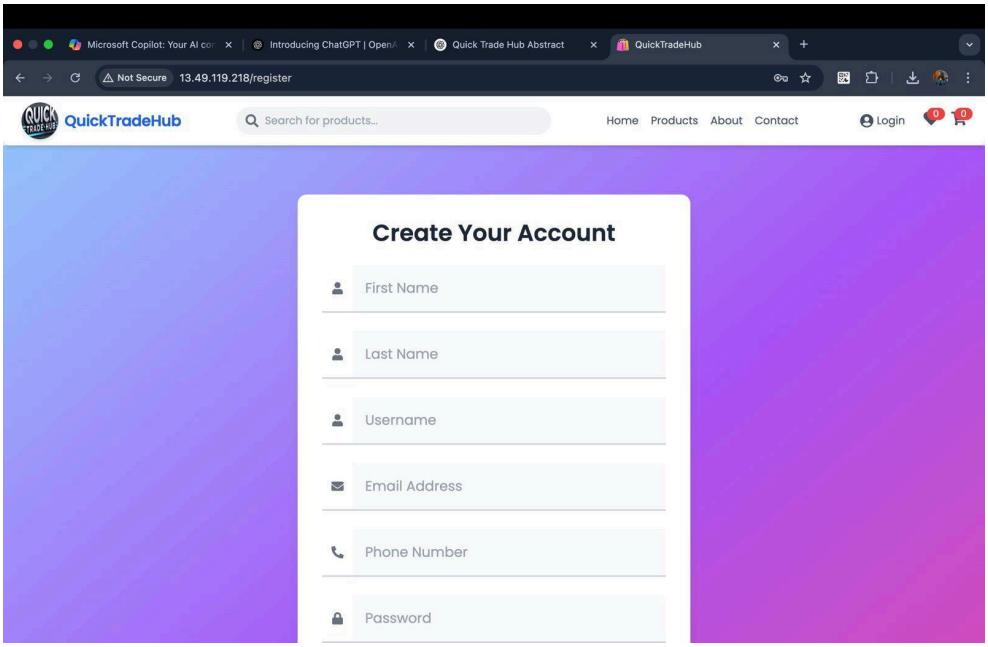


Fig 6.2 Sign up page

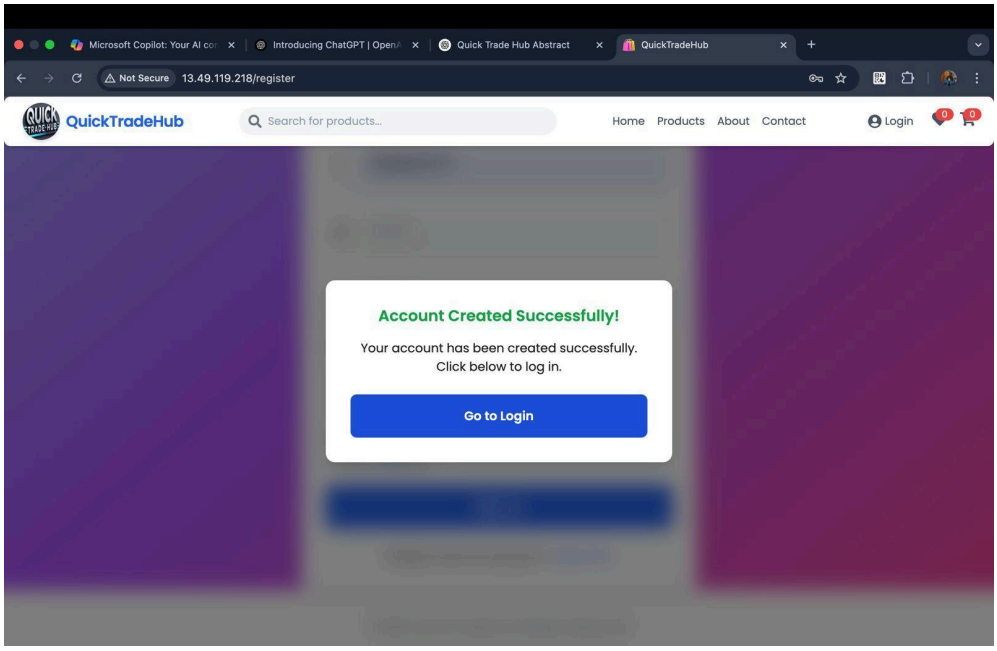


Fig 6.3 Successfull account Create

## 1 . Seller pages

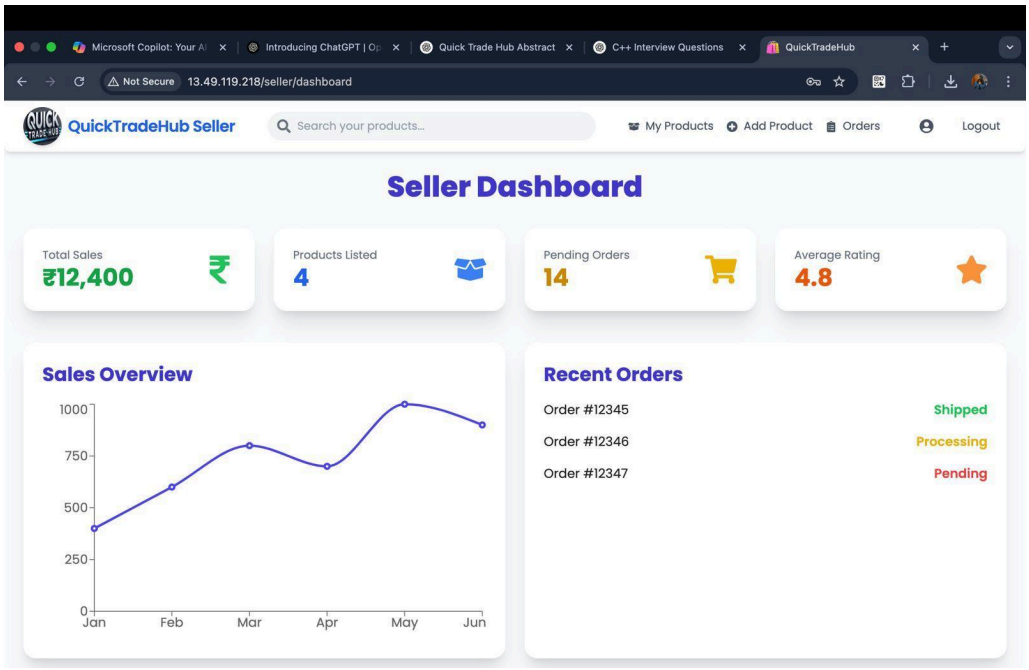


fig 6.4 seller dashboard page

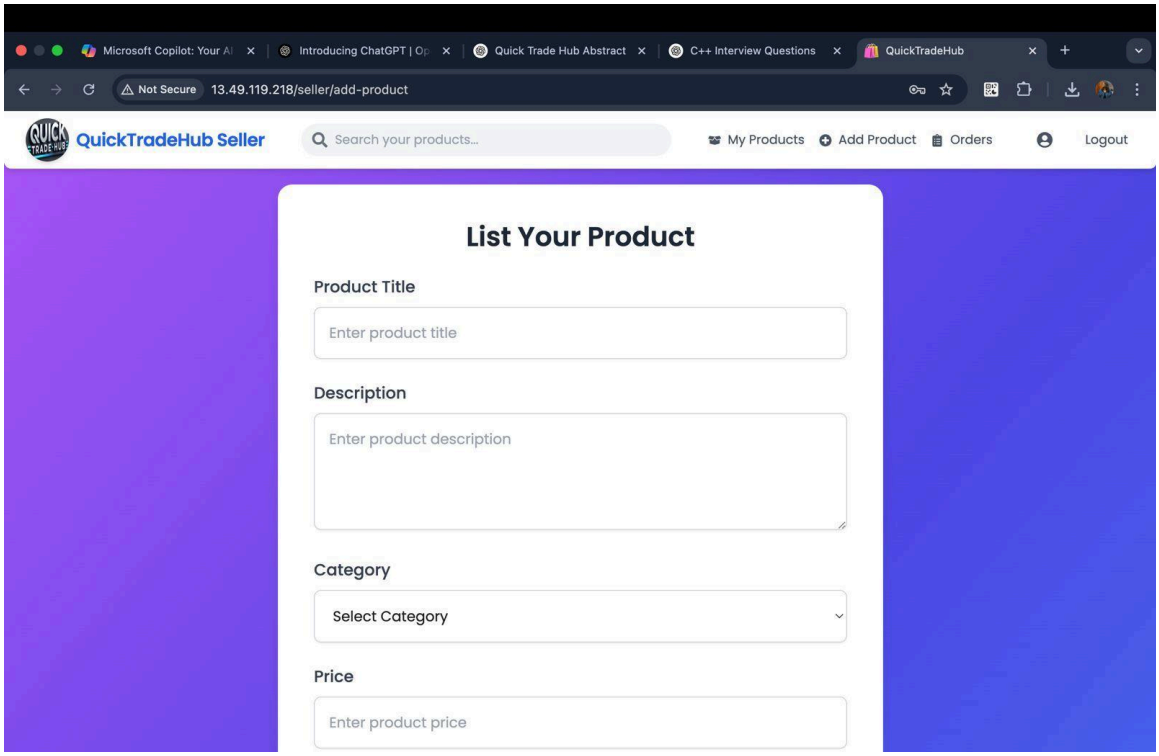


Fig 6.5 seller listing page

## 2. buyer pages:

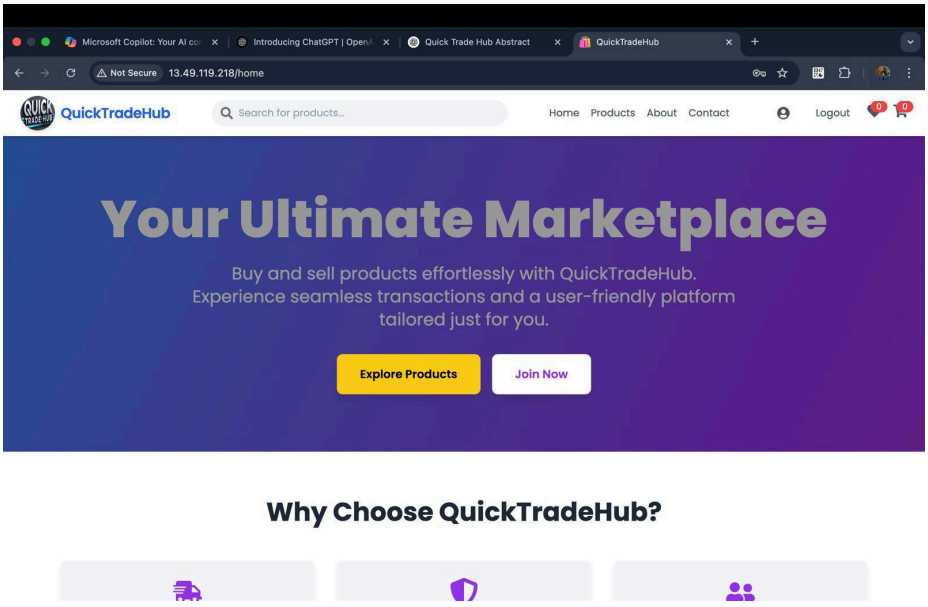


Fig 6.6 buyer dashboard

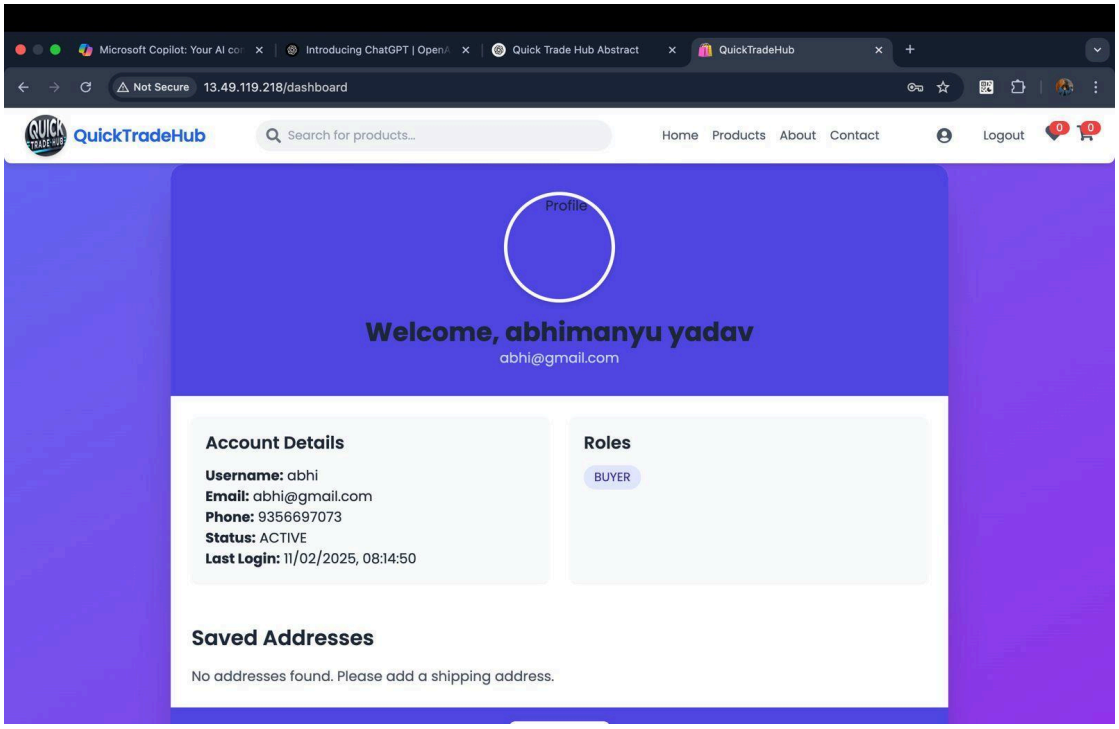


Fig 6.7 buyer profile page

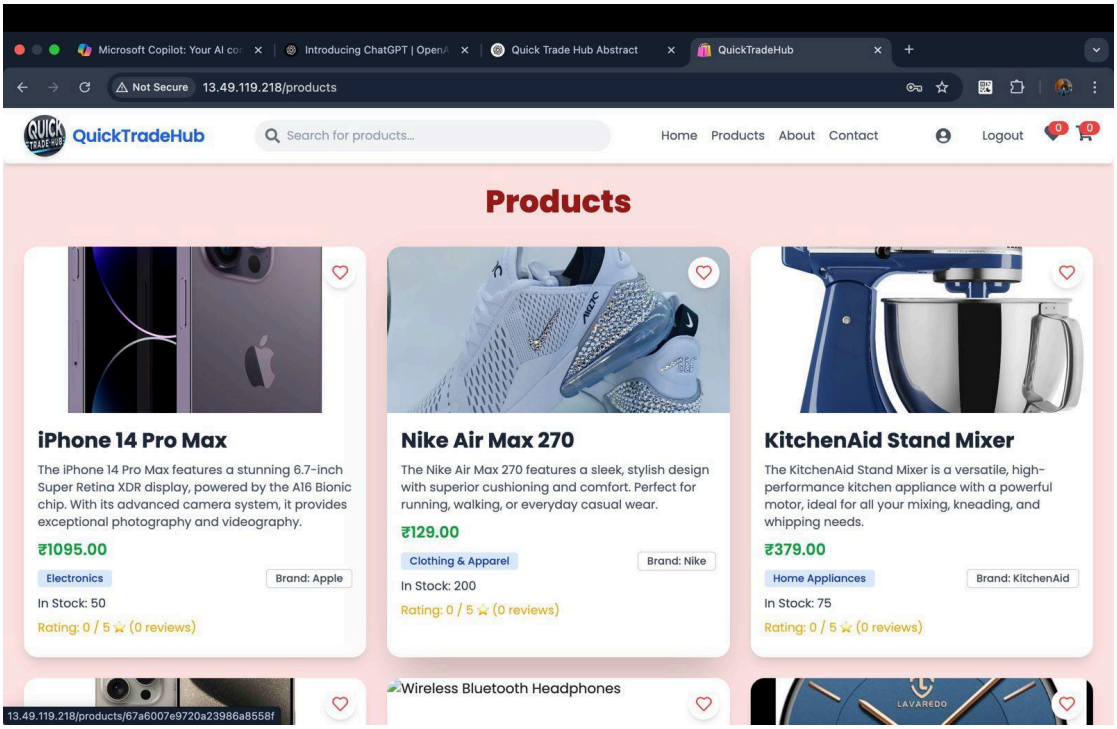


Fig 6.8 buyer product page

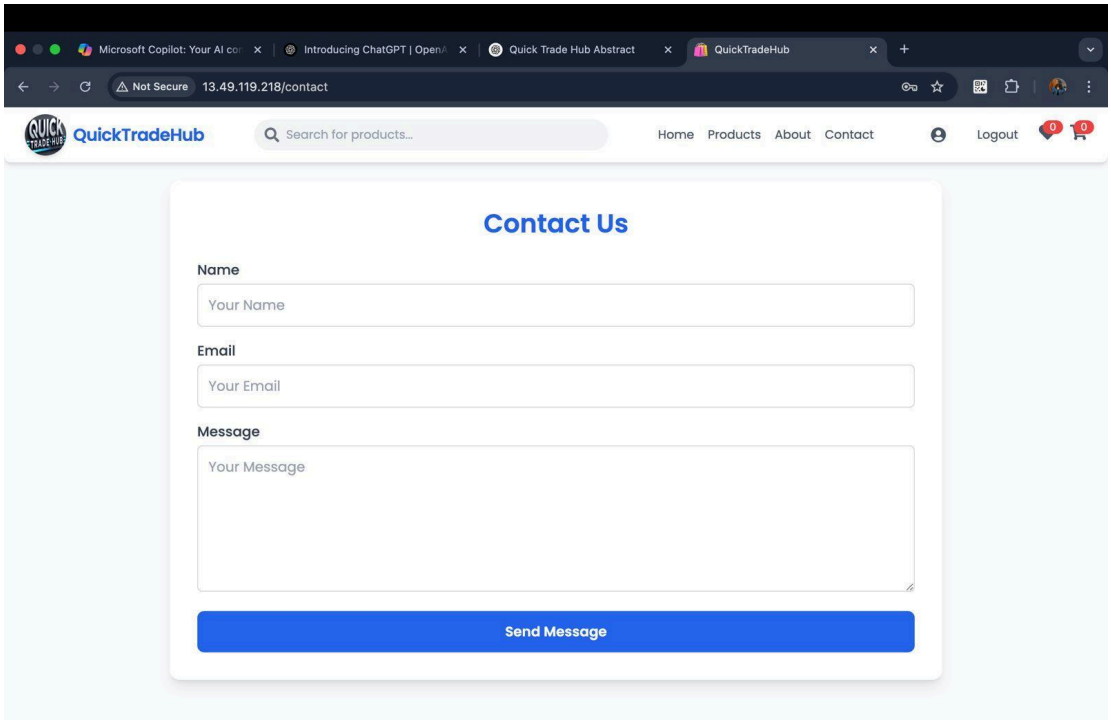


Fig 6.9 contact us page

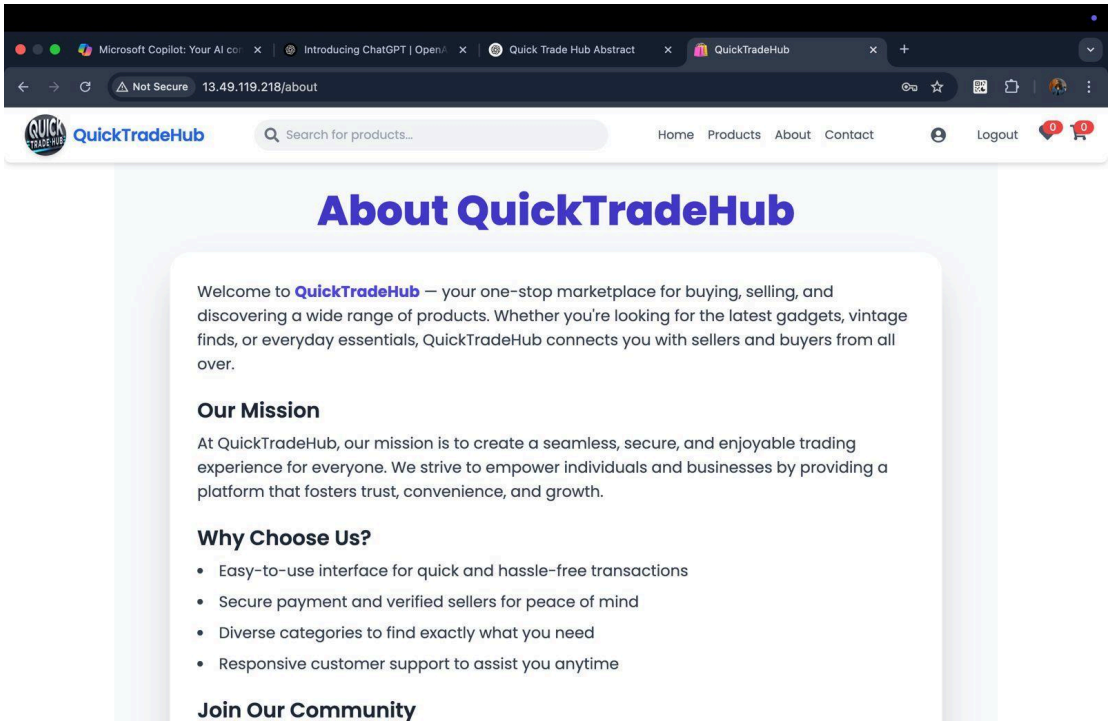


Fig 6.10 about page

# 7 CONCLUSION

The Quick Trade Hub project successfully achieves its goal of creating a modern, user-friendly, and secure online marketplace that facilitates the buying, selling, and trading of goods and services. By adopting a microservices architecture and leveraging advanced technologies such as *Java Spring Boot, **Node.js, **C# Web API, and **React.js*, the platform ensures scalability, flexibility, and maintainability.

The platform's key features, including secure user authentication with JWT, product management for sellers, real-time product search for buyers, and seamless payment integration, contribute to a smooth and efficient user experience. Furthermore, security measures such as role-based access control, BCRYPT password hashing, and HTTPS encryption guarantee a safe environment for users to engage in transactions.

Quick Trade Hub addresses the common challenges of trust, performance, and ease of use in online marketplaces. It provides essential tools for sellers and buyers to interact efficiently and securely while offering administrators control over the platform's operations and data analytics demands, Quick Trade Hub is well-positioned to serve a growing user base and enhance local commerce.

# 8 REFERENCES


1. Richardson, C. (2018). Microservices Patterns: With Examples in Java. Manning Publications. A comprehensive guide on microservices design patterns and implementation using Java.
2. Nascimento, H., & Netto, J. (2020). Spring Boot in Action
3. Manning Publications. A practical guide to building modern, scalable Java applications with Spring Boot.
4. Osmani, A. (2020). Learning JavaScript Design Patterns. O'Reilly Media. A guide to design patterns in JavaScript for building scalable web applications.
5. 4. Geier, J. (2021). Node.js Design Patterns: Design and Implement Production-Grade Node.js Applications Using Proven Patterns and Techniques. Packt Publishing. Focused on design and implementation patterns in Node.js applications.
6. 5. Eberhardt, S. (2019). Mastering C# and .NET Framework. Packt Publishing. Provides a deep dive into developing scalable web APIs using C#.
7. 6. Proven, R. (2021). RESTful Web API Design with Node.js 10. Packt Publishing. Focuses on designing RESTful APIs with Node.js and Express, covering best practice
8. Allamaraju, S. (2017). RESTful Web Services Cookbook. O'Reilly Media. A collection of best practices and strategies for designing secure, scalable RESTful web services.
9. Ferdinand, M. (2020). MongoDB: The Definitive Guide. O'Reilly Media.