A Project Report

On

“CryptoTradeX”

By

Arham Darshitkumar Shah (ET22BTIT003)

Dave Dhruv Mitesh (ET22BTIT016)

Desai Parth Anup (ET22BTIT019)

Godhani Neelkumar Rameshbhai (ET22BTIT037)

Veer Shekhda (ET23BTIT811)

under the mentorship of

Prof. Mukesh Patel

Asst. Prof. at Department of Information and Technology

December 2025

DEPARTMENT OF INFORMATION TECHNOLOGY

SARVAJANIK UNIVERSITY

Index

CERTIFICATE

This is to certify that the Project titled **“CryptoTradex”** was submitted by **Arham Darshitkumar(ET22BTIT003), Dave Dhruv Mitesh (ET22BTIT016), Desai Parth Anup (ET22BTIT019), Godhani Neelkumar Rameshbhai (ET22BTIT037)** and **Veer Shekhda (ET23BTIT811)** to the **Sarvajanik College of Engineering and Technology, Sarvajanik University** for the partial fulfilment of the subject credit requirements is a bonafied work carried out by the student.

The work presented in this report is the outcome of the student's independent effort. The findings and conclusions presented are based on the research and development undertaken during the project.

This project work has been carried out under my mentorship and is to my satisfaction.

Date: 12-11-20205
Place: SCET College, Surat

Prof. Mukesh Patel
Assistant Professor,
Department of Information Technology,
Sarvajanik College of Engineering and Technology,
Sarvajanik University, Surat.

Dr. Vivaksha Jariwala
Head, Department of Information Technology,
Sarvajanik College of Engineering and Technology,
Sarvajanik University, Surat.

Examiner’s Signature

Acknowledgements

We express our sincere gratitude to **Sarvajanik College of Engineering and Technology** for providing us with the necessary academic resources and an environment that fostered the successful development of our project CryptoTradeX.

We extend heartfelt thanks to our project guide, **Prof. Mukesh Patel**, for continuous support, valuable insights, and guidance throughout the duration of this work. Their expertise in software engineering and emerging technologies helped shape this project into a structured and meaningful outcome.

We are also thankful to our faculty members, peers, and the Department of Information Technology for their constructive feedback during reviews and presentations. Finally, we acknowledge the importance of open-source tools, development communities, and official documentation that assisted us in building a robust and research-backed project.

Name of Student : Arham Darshitkumar Shah

Enrollment No : ET22BTIT003

Name of Student : Dave Dhruv Mitesh

Enrollment No : ET22BTIT016

Name of Student : Desai Parth Anup

Enrollment No : ET22BTIT019

Name of Student : Godhani Neelkumar Rameshbhai

Enrollment No : ET22BTIT037

Name of Student : Veer Shekhda

Enrollment No : ET23BTIT811

Abstract

The global adoption of cryptocurrency has surged dramatically, exceeding **560 million users worldwide by 2024**. However, as participation expands, many users—especially beginners—continue to face challenges such as complex trading interfaces, fragmented wallet management, lack of real-time market insights, and limited educational resources.

CryptoTradeX is a **web-based, AI-integrated cryptocurrency trading platform** designed to address these issues by unifying **real-time market analytics, secure wallet services, portfolio tracking**, and **intelligent trading support** within a single ecosystem. Developed using **Spring Boot, React, and MySQL**, the platform ensures scalability, efficiency, and seamless user interaction. Enhanced **security layers**, including **JWT authentication** and **Two-Factor Authentication (2FA)**, safeguard user data and transactions.

This report presents the **system architecture, design methodology, implementation details**, and **performance evaluation** of CryptoTradeX. The platform aims to **simplify digital asset trading**, **enhance financial literacy**, and **promote secure and transparent transactions**. Ultimately, CryptoTradeX demonstrates how modern **fintech solutions** can effectively integrate **artificial intelligence, usability, and security** to democratize access to cryptocurrency trading.

.

List of Figures

List of Symbols, Abbreviations and Nomenclature

Symbol/Abbreviation	Term Meaning
CRUD	Create, Read, Update, Delete (basic DB operations)
JPA	Java Persistence API
ORM	Object-Relational Mapping
MVC	Model-View-Controller (architecture pattern)
UI/UX	User Interface / User Experience
HTTP	HyperText Transfer Protocol
HTTPS	Secure HyperText Transfer Protocol
JSON	JavaScript Object Notation
SPA	Single Page Application
DOM	Document Object Model
IDE	Integrated Development Environment
SQL	Structured Query Language
CLI	Command Line Interface
SPA	Single Page Application
CI/CD	Continuous Integration / Continuous Deployment
RBAC	Role-Based Access Control
UAT	User Acceptance Testing
SSL	Secure Sockets Layer
TFA	Two-Factor Authentication (alternative to 2FA)
CRUD	Create, Read, Update, Delete (database operations)
DAO	Data Access Object
DTO	Data Transfer Object

1.INTRODUCTION

The cryptocurrency ecosystem has undergone exponential evolution since Bitcoin's inception in 2009, evolving into a multi-trillion-dollar asset class by 2025. With the total market capitalization surpassing $4 trillion for the first time this year, as reported by a16z crypto's State of Crypto 2025 analysis, digital assets now represent a viable alternative to traditional finance. This surge is fueled by institutional interest, regulatory clarity in regions like the EU and India, and technological advancements such as layer-2 scaling solutions. Yet, beneath this optimism lies a persistent challenge: accessibility. Surveys indicate that 70% of potential users abandon platforms due to steep learning curves, security fears, and disjointed user experiences—issues exacerbated by high-profile hacks and volatile markets.

CryptoTradex is a pioneering AI-integrated trading platform designed to empower users at every level. Developed by us under the mentorship of Prof. Mukesh patel , it leverages modern java full-stack technologies to create a seamless, secure environment. At its core, CryptoTradex addresses these pain points by unifying trading, wallet management, and intelligent assistance into an intuitive interface, making crypto as approachable as everyday banking apps.

1.1 Problem Definition

Cryptocurrency trading platforms, while revolutionary, often present significant barriers to entry for novice users. Existing solutions like Binance, Coinbase, and WazirX offer powerful features but suffer from steep learning curves, disjointed wallet functionalities, and insufficient real-time support. Beginners frequently encounter information overload, with 70% reporting difficulties in navigating interfaces. Security vulnerabilities, such as exchange hacks, further erode trust, while the absence of integrated AI guidance leaves users vulnerable to market volatility without contextual explanations.

CryptoTradex tackles these issues by providing a unified, beginner-friendly ecosystem that integrates trading, wallet management, and AI assistance into a single, intuitive interface.

1.2 Motivation

The surge in cryptocurrency adoption—expected to surpass 560 million users globally by 2025, with India leading regionally—underscores the need for accessible tools. Current platforms fragment user experiences across multiple apps for trading, transfers, and analytics, leading to confusion and errors. High-profile security breaches highlight the demand for robust, multi-layered protections.

Our motivation stems from creating a secure, AI-enhanced platform that empowers users with seamless operations and instant insights, making crypto trading as straightforward as traditional banking while promoting financial literacy.

1.3 Relevance

In India, where crypto interest is growing but faces regulatory and usability challenges, CryptoTradex supports the country's digital economy goals. It uses Java full-stack technologies (React frontend, Spring Boot backend) for reliable and scalable performance, ensuring low-latency operations suitable for real-time trading. The AI integration via Gemini API offers simple market analysis, making it easier for beginners to participate.

The platform includes local payment options like Razorpay for secure fiat-to-crypto deposits, following basic KYC requirements. Its database with 17 tables handles user data, wallets, orders, and watchlists efficiently through clear relationships. Features like intuitive dashboards, 2FA security, and an AI chatbot for basic queries help new users understand trading without complexity. Overall, CryptoTradex provides a practical tool to explore crypto trading safely and accessibly.

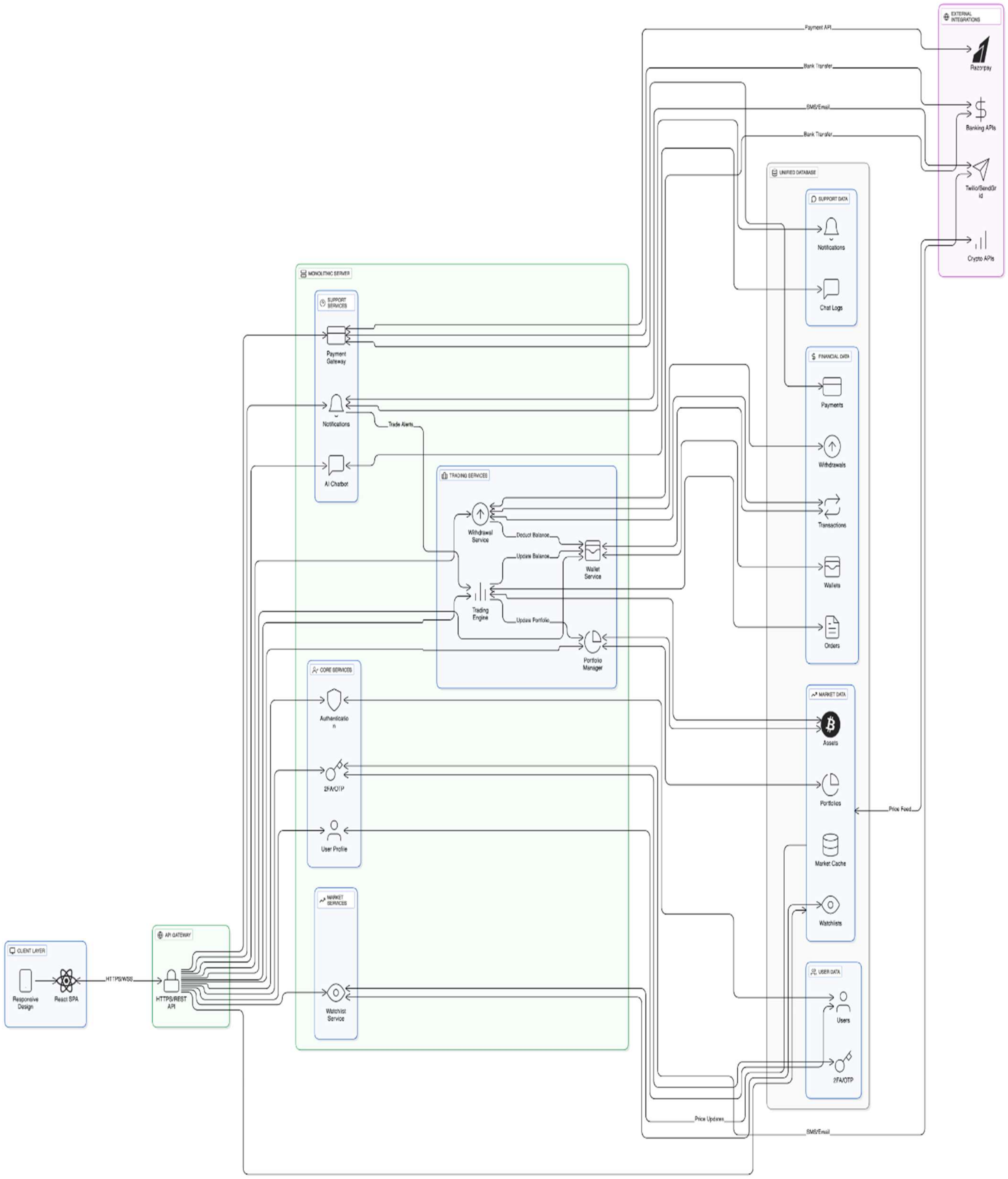
2. DESIGN

2.1 System Architechture

The system architecture of CryptoTradex is designed using a **three-tier model** consisting of the Presentation Layer, Application Layer, and Data Layer. The **Presentation Layer**, built with React, manages all user interactions, including authentication flow, wallet operations, and real-time visualization of market data. It communicates with the backend solely via secure REST API endpoints, ensuring separation of concerns and a smooth experience across devices. The use of Redux and modern UI libraries further improves responsiveness, state management, and user interface consistency.

The **Application Layer**, developed using Spring Boot, contains the business logic responsible for processing user requests. This includes validating login credentials, generating and verifying JWT tokens, handling deposits or transfers, placing trade orders, and computing portfolio values. The backend also integrates real-time cryptocurrency price feeds through external APIs like Gemini and CoinGecko, ensuring accurate and up-to-date market information for users. It acts as the main orchestration layer, coordinating between the frontend, external services, and the database.

The **Data Layer** uses MySQL to store user accounts, wallet balances, transactions, order history, and portfolio assets. Database normalization, indexing, and relational constraints ensure efficient data retrieval and secure financial recordkeeping. By distributing responsibilities across these three layers, the architecture becomes scalable, modular, and secure, making it suitable for future expansion such as mobile app integration or machine-learning modules.



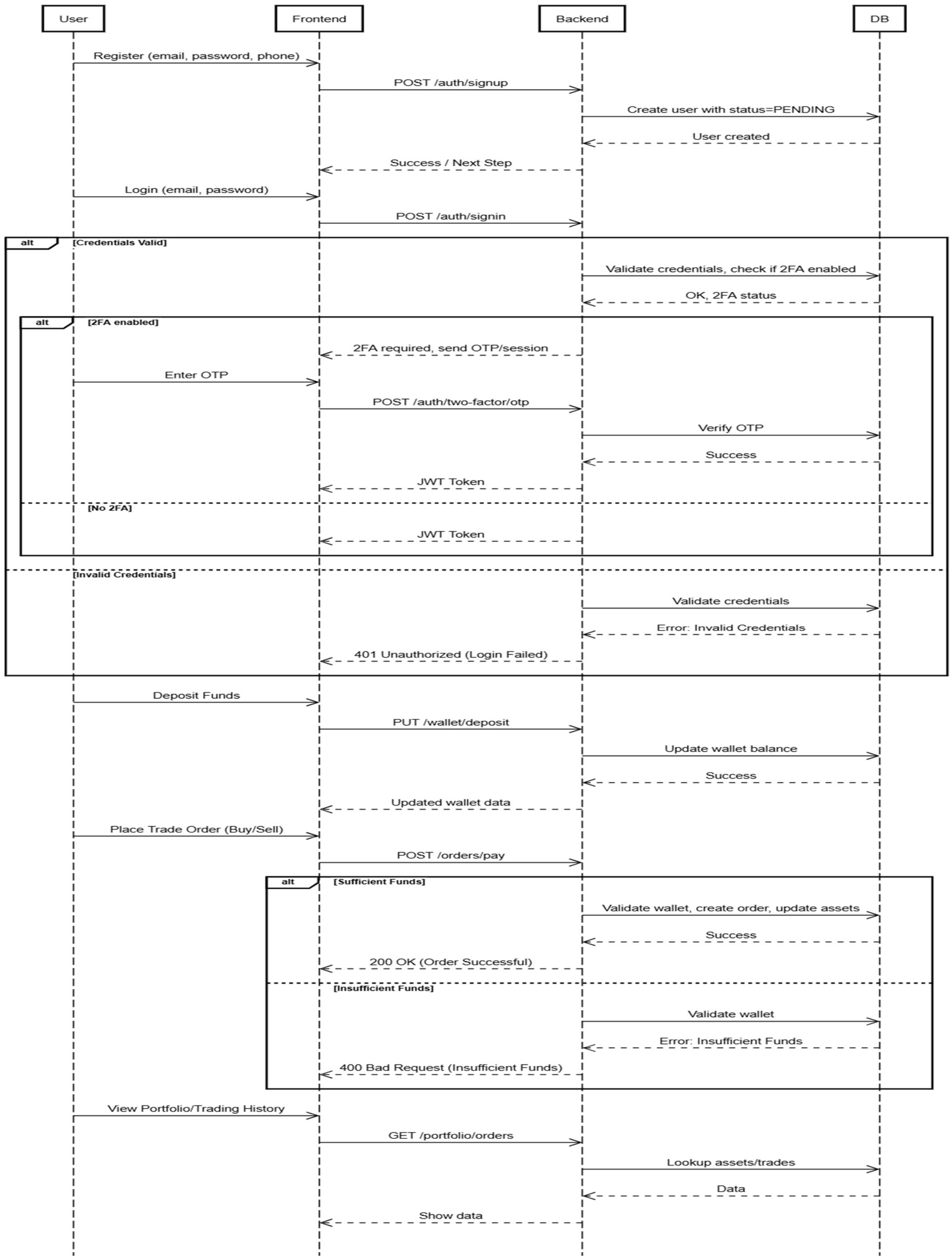
[Figure 2.1: System Architecture Diagram]

2.2 Sequence Diagram

The sequence diagram illustrates the complete interaction flow between the User, Frontend, Backend, and Database within the CryptoTradex system. It begins with the registration and login processes, where the user submits their details and the backend verifies credentials through database lookups. If Two-Factor Authentication (2FA) is enabled, the backend issues an OTP session, verifies the provided OTP, and then returns a secure JWT token. If 2FA is not enabled, the JWT token is issued immediately after credential validation. This ensures that authentication remains both flexible and secure.

After successful login, the user can perform wallet and trading operations. Deposits are handled through a **PUT /wallet/deposit** call, where the backend updates wallet balances and returns confirmation. When a user places a buy or sell order, the backend verifies available funds or holdings before executing the trade. Successful orders result in database updates and a **200 OK** response, while insufficient balance returns an error message. This ensures the accuracy and integrity of all financial actions performed on the platform.

Finally, the user can access their portfolio and trading history through a **GET /portfolio/orders** request. The backend retrieves stored trades and asset data from the database and returns it to the frontend, allowing users to visualize their performance. Overall, the sequence diagram highlights how CryptoTradex coordinates secure authentication, wallet activity, trade processing, and data retrieval in a structured and efficient manner.



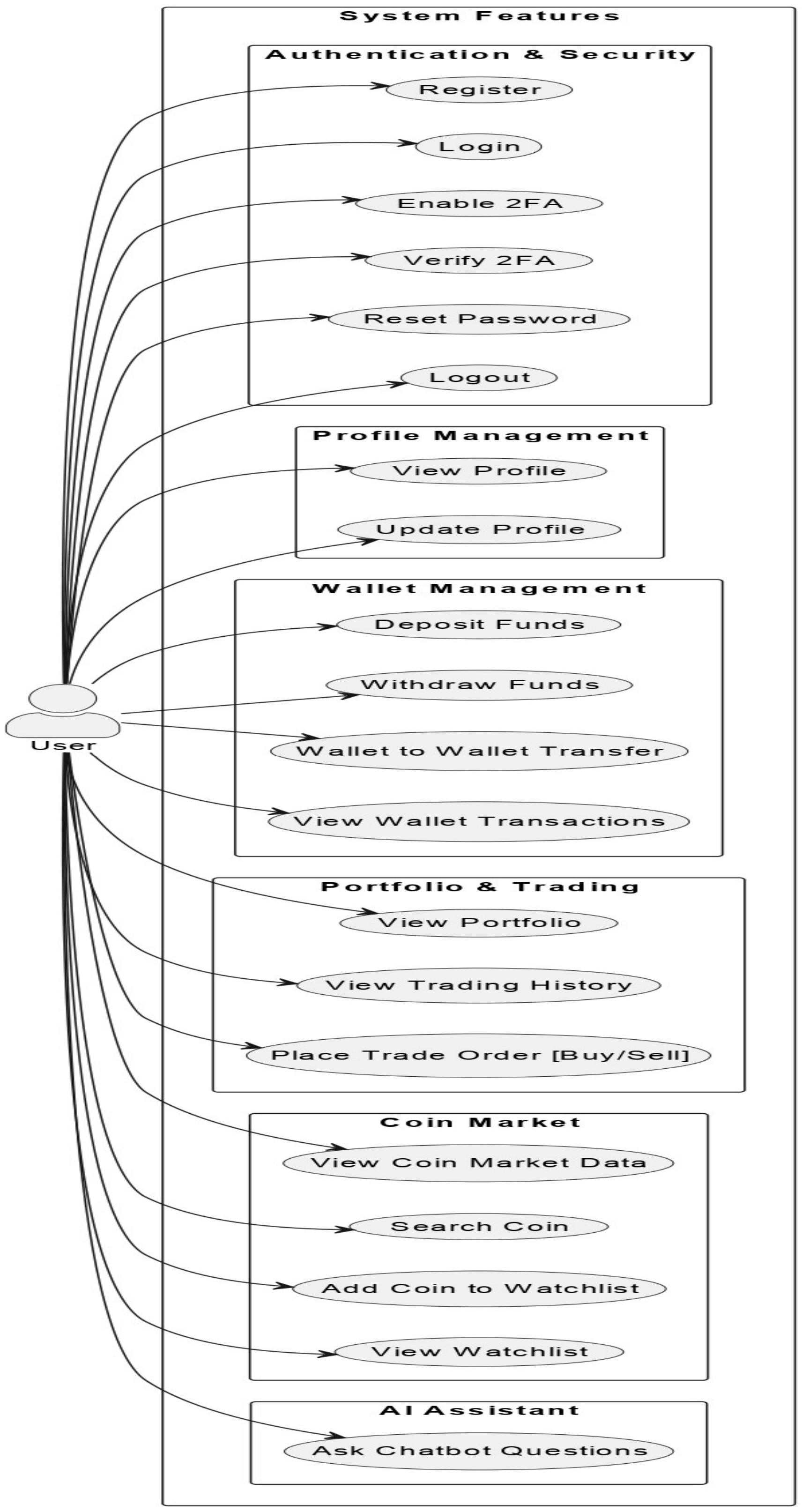
[Figure 2.2: Sequence Diagram]

2.3 Use Case Diagram

The use case diagram outlines all the major interactions between a user and the CryptoTradex platform. The **primary actor**, the User, engages with essential functionalities such as registration, login, enabling 2FA, managing wallet funds, placing buy or sell orders, and checking portfolio performance. These use cases represent the core activities needed for a fully functional trading system.

Supporting actors also enhance the system's capabilities. The **AI Assistant** serves as an intelligent helper that can answer queries, explain crypto terms, and fetch real-time market information. The **Payment Gateway** actor handles external financial operations, including deposits and withdrawals, ensuring smooth integration between the user's bank and the crypto wallet. Meanwhile, the **Database** actor is responsible for securely storing and retrieving persistent records.

Overall, the use case diagram provides a high-level view of how different system components work together to fulfill user needs. It ensures clarity in requirement gathering, helps validate system behavior, and forms a blueprint for later design and implementation tasks.



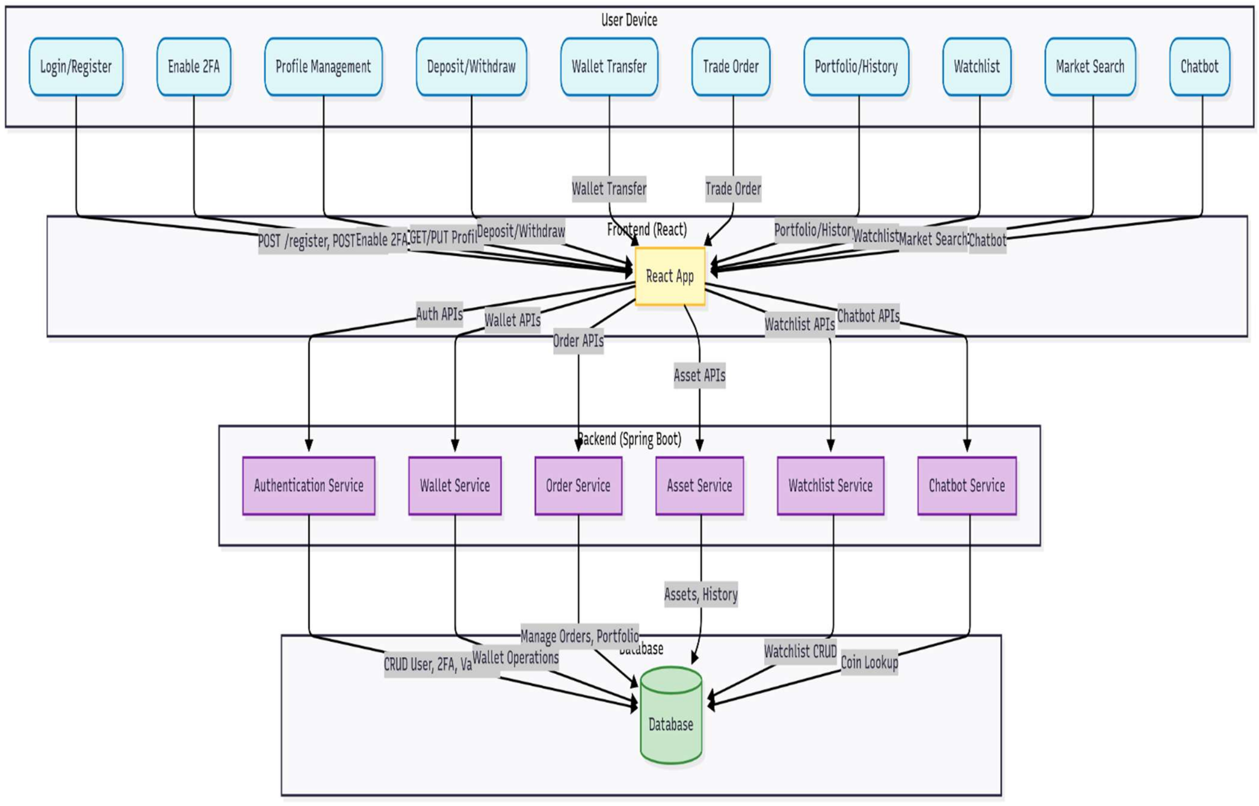
[Figure 2.3: Usecase Diagram]

2.4 Data Flow Diagram (DFD)

The Data Flow Diagram (DFD) highlights how information moves across the different processes of the CryptoTradex platform. When a user interacts with the system—whether by logging in, depositing funds, or placing a trade—the frontend forwards this data to the backend for processing. The backend verifies inputs, applies business logic, and retrieves or updates necessary data in the database. This controlled flow ensures accuracy and consistency across user operations.

External APIs, such as CoinGecko and Gemini, also contribute to the data flow by supplying live market prices. These values are fetched by the backend at regular intervals and sent to the frontend for display. As users place trade orders, the backend checks their wallet balance, validates market prices, and updates the database with new order records. The updated data is then returned to the frontend to reflect the latest information.

By clearly separating data inputs, processes, and outputs, the DFD ensures that every module communicates effectively without redundancy. It also helps identify potential bottlenecks, enabling developers to optimize system performance and maintain reliability during high traffic loads.



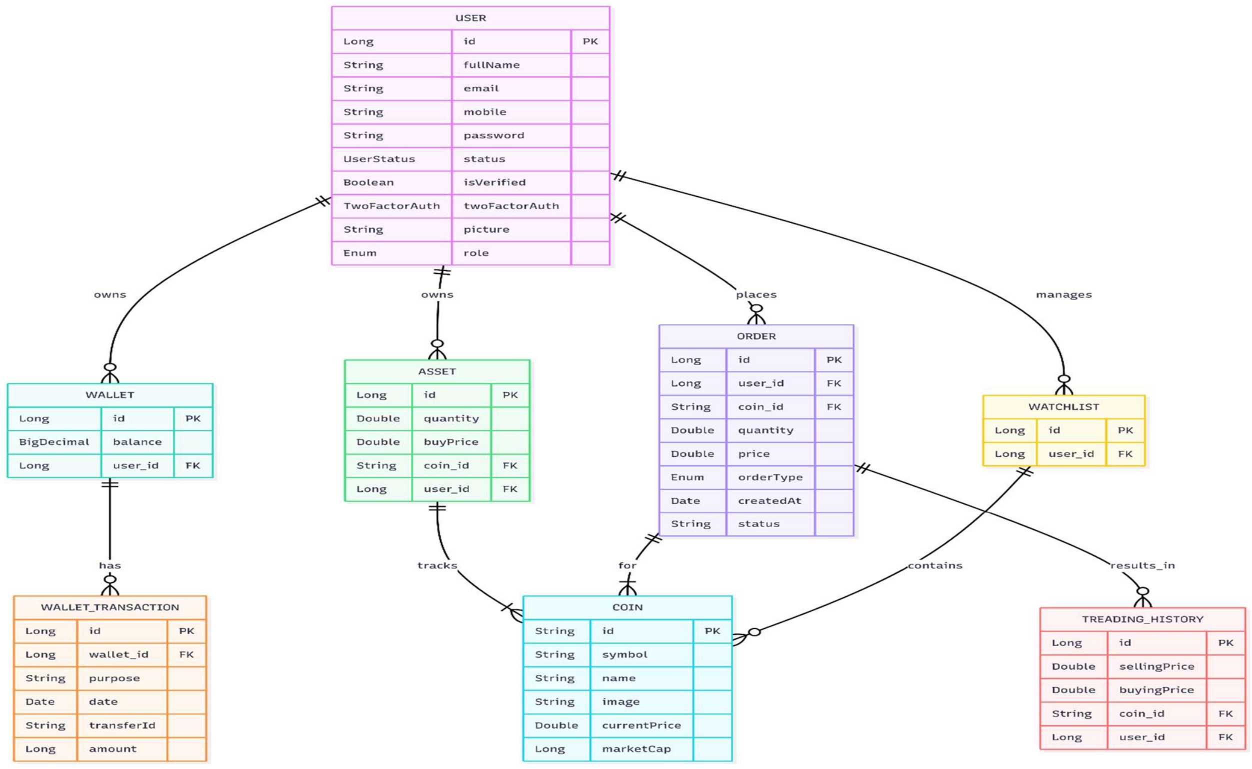
[Figure 2.4: Data Flow Diagram (DFD)]

2.5 Data Diagram

The Entity–Relationship (ER) diagram forms the structural backbone of the CryptoTradex database. It identifies major entities such as **User**, **Wallet**, **Orders**, **Transactions**, and **Portfolio**, each representing a critical component of the trading system. The User entity stores authentication and personal details, while the Wallet entity keeps track of the user's available balance. The Orders entity captures buy and sell operations, storing relevant attributes such as price, quantity, and timestamp.

These entities are linked through well-defined relationships. For instance, a single user can have multiple orders and transactions, forming one-to-many relationships. The Wallet is directly linked to the User, ensuring accurate mapping of financial balances. The Portfolio entity aggregates all holdings for each user, allowing the system to compute real-time valuations and profit/loss metrics. Foreign key constraints reinforce data integrity across all tables.

Together, these relationships create a robust database structure that supports secure storage, efficient retrieval, and error-free transaction handling. The ER diagram therefore acts as a blueprint for database implementation, ensuring that the system remains scalable and consistent as new features or asset types are introduced.



[Figure 2.5: Data Diagram]

3.IMPLEMENTATION

3.1 Frontend Development

The frontend of CryptoTradeX is engineered with a robust stack centered around React, chosen for its component-driven architecture and high performance in developing complex, dynamic interfaces. The core application is scaffolded using modern React paradigms—including hooks and functional components—to ensure maintainable and reusable UI elements. State management is systematically handled by Redux, providing a unidirectional data flow and enabling efficient management of global state across the application. By structuring actions, reducers, and middleware logically, Redux not only supports transactional updates tied to user actions but also powers asynchronous operations such as API polling and session handling.

Styling within the application leverages Tailwind CSS, a utility-first CSS framework that promotes rapid prototyping and consistency through a comprehensive set of design utilities. This is further complemented by integration of shadcn/ui, a suite of polished, accessible React components. These components, adhering to the latest UI/UX best practices, facilitate the creation of a visually cohesive, responsive layout adaptable to different device screens.

Navigation is orchestrated with React Router DOM, employing both declarative and programmatic routing strategies to manage authenticated and non-authenticated flows. Routes are protected using HOC (Higher Order Components) that check JWT tokens persisted in localStorage, ensuring seamless and secure access control to sensitive areas such as the dashboard and trading interfaces.

Core components include feature-rich Login/Register forms with validation logic, a Dashboard that provides real-time portfolio and transaction overviews, and interactive charting modules for market analysis. A standout offering is the Chat interface, powered by Google's Gemini API, which acts as both a contextual assistant and market news provider. The chat leverages advanced NLP to offer insights, answer queries, and augment the user experience with conversational interactions.

All API communications are performed using Axios, which is configured with interceptors for appending JWT tokens to request headers and handling error scenarios such as token expiration or network faults. The tightly integrated Redux middleware ensures that API calls update the store atomically, contributing to a reactive and state-consistent UI.

Moreover, the frontend maintains a high standard for security and compliance, implementing measures like input sanitation, secure storage of JWTs, and proactive session expiry handling, ensuring a trustworthy user experience across all touchpoints.

3.2 Backend Development

The backend infrastructure of CryptoTradeX is underpinned by the Spring Boot framework, chosen for its maturity, scalability, and enterprise-grade features. The application is architected using a multi-layered design that strictly separates concerns, dividing responsibilities among controllers, services, and repositories. This strategy fosters clean code, facilitates testing, and supports future scalability.

Controllers act as the entry point for HTTP requests, exposing RESTful endpoints such as POST /auth/signup for user registration, PUT /wallet/deposit for wallet operations, and POST /orders/trade for executing trades. Each controller method is annotated for clear intent and input validation, integrating smoothly with Spring's request-mapping facilities to ensure precise routing and error handling.

Business logic is encapsulated within dedicated service classes, such as WalletService for handling wallet credit/debit operations, and OrderService for validating and executing trade orders. This layer mediates all interactions between controllers and the data persistence mechanism, encapsulating transaction management, validation, error reporting, and complex business rules—such as enforcing trading limits or composite wallet adjustments.

Data persistence is facilitated through Spring Data JPA and Hibernate ORM, mapping Java entities to MySQL tables with strict schema enforcement. JPA repositories abstract the CRUD operations, enabling rapid prototyping and database-agnostic development. Custom repository methods cater to domain-specific queries, ensuring optimal database interactions and maintaining code clarity.

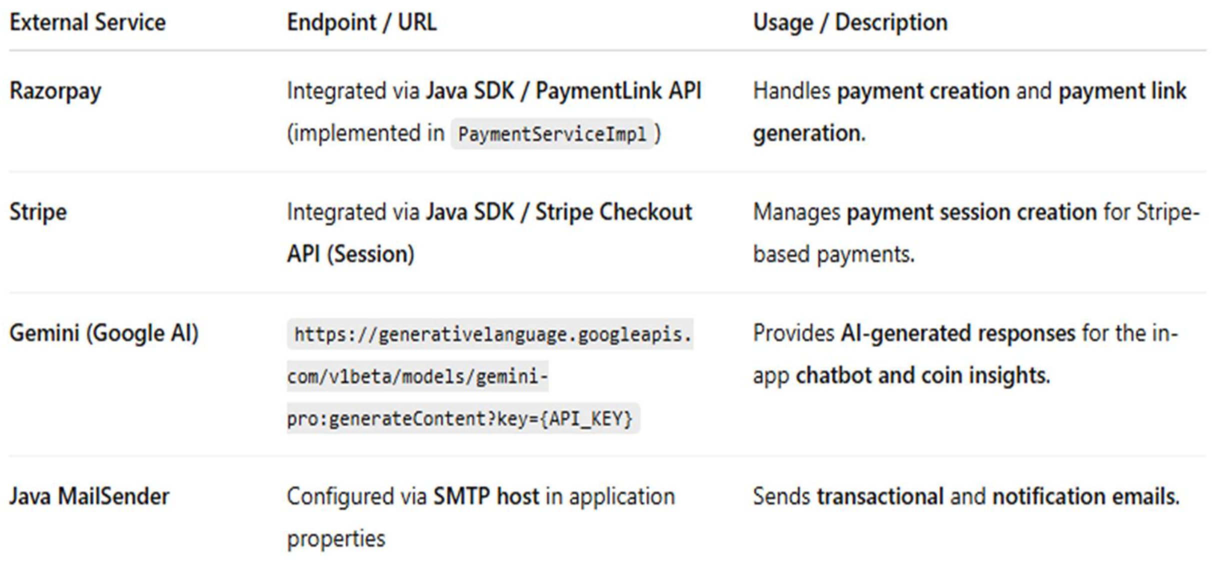
Security is paramount within the backend. The system utilizes Spring Security with JWT-based authentication to protect sensitive endpoints. JWT tokens are issued upon successful login and validated on each request, ensuring stateless, scalable session management. Additionally, granular role-based access control (RBAC) governs user permissions, segregating functionalities between USER and ADMIN roles by configuring method-level security annotations and endpoint whitelisting.

Ancillary backend features include scheduled jobs for transaction reconciliation, integration with payment gateways (such as Razorpay and Stripe) via provider-specific services, and a notification subsystem leveraging JavaMailSender for transactional emails. Asynchronous processing is adopted for long-running tasks, maintaining application responsiveness and supporting high concurrency.

Finally, the backend's configuration and deployment leverage environment-specific profiles, externalized configuration properties, and containerized delivery (optional), ensuring robust operation in both development and production environments. Comprehensive logging, monitoring, and exception tracing are built in to support rapid diagnostics and maintain uninterrupted service for all users.

3.3 Integration with External APIs

CoinGecko API fetches real-time coin data (prices, market caps) via REST endpoints. Gemini API powers the chatbot for natural language queries on trends. Razorpay/Stripe integrations process deposits/withdrawals, with webhooks for transaction confirmations. Email notifications use Java Mail Sender for 2FA OTPs and alerts.



External Service	Endpoint / URL	Usage / Description
Razorpay	Integrated via **Java SDK / PaymentLink API** (implemented in `PaymentServiceImpl`)	Handles **payment creation** and **payment link generation.**
Stripe	Integrated via **Java SDK / Stripe Checkout API (Session)**	Manages **payment session creation** for Stripe-based payments.
Gemini (Google AI)	`https://generativelanguage.googleapis.com/v1beta/models/gemini-pro:generateContent?key={API_KEY}`	Provides **AI-generated responses** for the in-app **chatbot and coin insights.**
Java MailSender	Configured via **SMTP host** in application properties	Sends **transactional** and **notification emails.**

4. CONCLUSION

The development of **CryptoTradex** successfully addressed the fundamental challenges associated with modern cryptocurrency trading platforms. By integrating secure authentication methods, real-time market data, AI-assisted support, and unified wallet operations, the system offers a balanced combination of usability, security, and efficiency. The project demonstrates how a well-designed, full-stack architecture can simplify digital asset trading for beginners while still providing the depth and robustness required by more experienced users. Through the use of technologies such as Spring Boot, React, MySQL, and external market-data APIs, the system achieves both functionality and technical reliability.

Throughout the project, critical emphasis was placed on ensuring data integrity, secure communication, and smooth transactional flow. Features like **JWT-based login**, **Two-Factor Authentication**, and **encrypted data handling** significantly strengthen the platform's security posture. Similarly, the wallet and trading modules were designed to operate with precision, validating balances, updating records, and handling potential errors gracefully. The interaction between system components—validated through design diagrams such as the sequence, use case, DFD, and ER diagrams—confirms that the system performs cohesively as a unified product.

The project also highlighted the importance of user experience in fintech systems. The inclusion of an AI chatbot, intuitive dashboards, transaction history modules, and structured portfolio views reflects an intentional effort to empower users with clarity and guidance. These design decisions improve accessibility and lower the barrier to entry for individuals unfamiliar with cryptocurrency markets. As a result, CryptoTradex stands not only as a software implementation but also as a step toward enhancing financial literacy and confidence among new investors.

In conclusion, CryptoTradex meets its objectives of creating a secure, intelligent, and user-friendly platform for cryptocurrency trading. The system lays a strong foundation for future improvements such as automated trading, machine-learning-based market predictions, multi-chain wallet integration, and a dedicated mobile application. With further refinement and expansion, the platform has the potential to evolve into a comprehensive fintech solution capable of supporting large-scale user adoption in the growing digital asset ecosystem.

5. Future Scope and Further Enhancement of the Project

5.1 Future Scope and Further Enhancements

The future scope for the CryptoTradeX platform includes broadening its capabilities both technically and functionally. Planned enhancements and ambitious goals for the project are as follows:

- Expansion to Additional Cryptocurrencies and Exchanges: Support for a wider range of coins and direct integration with multiple exchanges (such as Binance, Kraken) will be added, allowing users greater flexibility and diversification opportunities.
- Advanced Trading Features: Implement features such as margin trading, automated trading bots, stop-loss/take-profit mechanisms, and customizable trading strategies for power users.
- Real-Time Analytics and Charting: Integrate live market feeds and advanced charting tools (candlestick charts, technical indicators) for in-depth analysis.
- Mobile Application Development: Launch mobile versions (Android/iOS) of the platform to make trading and portfolio management seamless and accessible on-the-go.
- Enhanced Security: Introduce biometric authentication, regular security audits, and machine learning-based fraud detection systems to protect user assets and data.
- Social and Community Features: Develop forums, leaderboards, and community trading competitions to increase user engagement and learning.
- Regulatory Compliance Automation: Automate KYC/AML processes and add reporting tools for compliance with international financial regulations.
- Integration with DeFi Protocols: Enable users to interact with decentralized finance ecosystems for lending, borrowing, staking, and yield farming.
- Personalized Dashboards and Alerts: Allow customizable dashboards, real-time notifications for price movements, trade executions, and news headlines.
- API Public Release: Provide secure, documented RESTful APIs for third-party developers to build applications and services on top of CryptoTradeX.

5.2 Addressing Scalability and Performance

As user adoption grows, scalability and performance become critical for sustained reliability and user satisfaction. Key strategies and enhancements planned include:

- Microservices Architecture: Transitioning backend services to a microservices-based architecture, leveraging containerization and orchestration tools (Docker, Kubernetes) to ensure modular scalability, fault isolation, and ease of deployment.
- Cloud-Based Infrastructure: Migrate infrastructure to cloud platforms (AWS, Azure, GCP) for elastic scaling, global distribution, and managed security.
- Database Optimization: Implement advanced database techniques such as sharding, replication, and caching (using Redis/Memcached) to improve data access speeds and ensure high availability.
- Load Balancing and Auto-Scaling: Employ load balancers and auto-scaling groups to dynamically distribute user requests and maintain optimal resource utilization under heavy loads.
- Asynchronous Processing: Introduce asynchronous task queues and event-driven architecture for non-blocking operations such as trade execution, notification delivery, and third-party API interactions.
- Performance Monitoring and Alerts: Integrate end-to-end monitoring (using tools like Prometheus, Grafana, ELK stack) to track latency, throughput, and error rates, with automated alerts for anomalies.
- API Rate Limiting and Throttling: To protect against excessive traffic or abuse, deploy rate limiting strategies on all APIs and key services.
- Codebase Refinement: Continuously refactor and optimize both frontend and backend code for efficiency, reduce render-blocking resources, and optimize use of third-party libraries.
- User Experience Optimization: Optimize response times, minimize downtime, and reduce transaction waiting periods for an efficient, seamless user experience even during peak usage.

By systematically implementing these future enhancements and scalability improvements, CryptoTradeX aims to evolve into a robust, high-performance, and industry-leading platform for cryptocurrency trading and portfolio management.

References

- CoinGecko API Documentation. Retrieved from https://www.coingecko.com/
- Gemini API Services. Retrieved from https://www.gemini.com/
- Eraser Diagramming Tool. Retrieved from https://www.eraser.io/
- Mermaid Chart for Flowcharts. Retrieved from https://www.mermaidchart.com/
- Razorpay Payment Gateway. Retrieved from https://razorpay.com/
- Stripe Financial Infrastructure. Retrieved from https://stripe.com/
- Spring Boot Official Documentation. Retrieved from https://spring.io/projects/spring-boot
- React.js Framework Guide. Retrieved from https://react.dev/

Appendix

Appendix A: API Endpoint Reference

Endpoint (Path)	Method	Provider/Feature	Purpose/Description	Required Parameters
/api/auth/signup	POST	Authentication	User registration	username, email, password
/api/auth/login	POST	Authentication	User login	email, password
/api/auth/2fa/verify	POST	Authentication, 2FA	Two-Factor Authentication verification	code (OTP), JWT
/api/payment/{paymentMethod}/amount/{amount}	POST	Payments (Razorpay, Stripe)	Initiate payment and generate payment URL	paymentMethod, amount, JWT
/api/withdrawal/{amount}	POST	Payments, Withdrawal	Request withdrawal from user wallet	amount, JWT
/api/admin/withdrawal	GET	Admin	Retrieve withdrawal requests (admin view)	JWT (admin)
/api/chatbot/ask	POST	Chatbot	Conversational queries and trading assistant (AI)	question text, JWT

Endpoint (Path)	Method	Provider/Feature	Purpose/Description	Required Parameters
		(Gemini API)		
/api/mailsender/send	POST	Mailing	Send notification or transactional email	to, subject, content, JWT

Appendix B: Example Payloads

1. Payment Request (Razorpay / Stripe)

Endpoint:

```
POST /api/payment/RAZORPAY/amount/5000
```

Headers:

```
Authorization: Bearer <jwt>
```

Request Body:

```
{}
```

Response:

```
{
  "payment_url": "https://razorpay.com/.../abc123"
}
```

2. Withdrawal Request

Endpoint:

```
POST /api/withdrawal/2000
```

Headers:

```
Authorization: Bearer <jwt>
```

Response:

```
{
  "status": "success",
  "message": "Withdrawal request submitted.",
  "transaction_id": "TXN123456"
}
```

3. Chatbot Query (Gemini AI Integration)

Endpoint:

```
POST /api/chatbot/ask
```

Headers:

```
Authorization: Bearer <jwt>
```

Request Body:

```
{
  "question": "What is Bitcoin?"
}
```

Response:

```
{
  "answer": "Bitcoin is a decentralized digital currency..."
}
```

4. Common Error Response Format

Example Response:

```
{
 "timestamp": "2025-11-11T12:30:45Z",
 "status": 401,
 "error": "Unauthorized",
 "message": "Invalid JWT token",
 "path": "/api/payment/RAZORPAY/amount/1000"
}
```

Appendix C: Configuration Snippets

Application.properties (Spring Boot Backend)

```
spring.datasource.url=jdbc:mysql://localhost:3306/cryptotrade

spring.datasource.username=root

spring.datasource.password=your_db_password

spring.jpa.hibernate.ddl-auto=update

jwt.secret=YourSuperSecretKeyHere

stripe.api.key=pk_test_XXXXX

razorpay.api.key=rzp_test_XXXXX

razorpay.api.secret=yourRazorpaySecret

mail.host=smtp.gmail.com

mail.port=587

mail.username=your-email@gmail.com

mail.password=your-app-password
```

.env (React Frontend)

REACT_APP_API_BASE_URL=https://api.cryptotradex.com

REACT_APP_GEMINI_API_KEY=AIzaSyXXXXXX

JWT Setup

- JWT secret used on backend for signing tokens: jwt.secret=YourSuperSecretKeyHere
- Tokens are sent in Authorization headers as: Authorization: Bearer <token>

Appendix D: Screenshots

1. Login/Registration Page
   - Shows fields for email/password, register button, and error alerts.
2. User Dashboard
   - Portfolio summary, ROI chart, positions table.
3. Wallet Page
   - Deposit, withdrawal, transaction history.
4. Payment Page
   - Razorpay/Stripe selection, payment success/failure dialogs.
5. Chatbot Page
   - Gemini-powered interface, message history, quick suggestions.