Crypto Price Watch

A PROJECT REPORT
Submitted By
Vishek Tyagi
<2200290140183>
Vishal Sen
<2200290140182>

Submitted in partial fulfilment of the
Requirements for the Degree of

MASTER OF COMPUTER APPLICATIONS

Under the Supervision of
Dr. Akash Rajak
Professor

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206
(February, 2024)

CERTIFICATE

Certified that **Vishek Tyagi<2200290140183> and Vishal Sen<2200290140182>** have carried out the project work having "**Crypto Price Watch**" for Master of Computer Applications from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Technical University, Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself / herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Vishek Tyagi (2200290140183)
Vishal Sen (2200290140182)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr Akash Rajak
Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Kumar Tripathi
Professor & Head
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Crypto Price Watch

Vishek Tyagi, Vishal Sen

ABSTRACT

In the dynamic landscape of cryptocurrency markets, staying informed about real-time price movements is essential for investors, traders, and enthusiasts. The Crypto Price Watch project addresses this need by offering a comprehensive web application for monitoring cryptocurrency prices in real-time. Developed using React.js for the frontend and Spring Boot for the backend, the application features a user-friendly interface with functionalities including real-time price updates, search options for cryptocurrencies, and user authentication.

This project report provides a detailed exploration of the development process, starting with a feasibility study that assesses technical, economical, operational, and schedule feasibility. The architectural design section outlines the system's structure, including entity-relationship diagrams, data flow diagrams, and use case diagrams. Additionally, the system design section delves into database schema design, detailing tables for user information, personal data, education, and more.

Implementation details cover the development of key features, such as real-time price monitoring, user authentication, and search functionality, along with challenges faced and solutions devised during development. The testing section outlines test cases for ensuring the functionality and reliability of the application.

Future enhancements are discussed, including advanced analytics, personalized alerts, social features, enhanced customization options, and educational resources, to further enrich the user experience and provide added value.

In conclusion, the Crypto Price Watch project represents a significant step towards empowering users with the tools and information they need to navigate the complex world of cryptocurrency markets effectively. By leveraging modern technologies and continuously evolving to meet user needs, the application aims to become a trusted companion for cryptocurrency enthusiasts worldwide.

ACKNOWLEDGEMENTS

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Akash Rajak** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Vishek Tyagi
Vishal Sen

LIST OF CONTENT

CHAPTER 1

Introduction

In today's rapidly evolving digital landscape, cryptocurrencies have emerged as a significant asset class, captivating the attention of investors, traders, and enthusiasts worldwide. The volatile nature of cryptocurrency markets necessitates real-time monitoring and analysis to make informed decisions. In response to this need, our project, Crypto Price Watch, presents a comprehensive web application designed to facilitate the monitoring of cryptocurrency prices in real-time.

1.1 Purpose

The purpose of this project report is to document the development process and key features of the Crypto Price Watch web application. It serves as a comprehensive guide for stakeholders, developers, and users interested in understanding the functionality and capabilities of the application.

1.2 Scope

The scope of the project encompasses the design, development, and deployment of a web-based platform for monitoring real-time cryptocurrency prices. The application includes features such as real-time price updates, search functionality for cryptocurrencies, user authentication, and a user-friendly interface for seamless navigation.

1.3 Overview

This section provides an overview of the Crypto Price Watch application, highlighting its main functionalities and target audience. It outlines the objectives and goals of the proposed system, setting the stage for a detailed exploration of its features and implementation.

1.4 Goals Of proposed System

The primary goals of the proposed system are to:

Provide users with real-time updates on cryptocurrency prices.
Offer a convenient search feature for finding specific cryptocurrencies.
Ensure user authentication and secure access to the application.
Create an intuitive and user-friendly interface for enhanced user experience.

1.5 Background

Cryptocurrencies have gained significant traction in recent years, attracting interest from investors, traders, and technology enthusiasts alike. The volatile nature of cryptocurrency markets necessitates reliable tools for monitoring price movements and making informed decisions. In response to this need, Crypto Price Watch aims to provide users with a comprehensive platform for tracking cryptocurrency prices in real-time.

1.6 Project Requirements

Table 1.1 Software requirements

Software requirements	
Operating system	**Software requirement**
Windows 7,8,10, Linux, or any other higher version	Google chrome, internet explorer, or any web browser

Table 1.2 Hardware requirements

Hardware requirements	
Processor	**RAM**
Core i3,i5 or i7	2GB or more

1.7 Technology Used

The technologies utilized in the development of Crypto Price Watch include:

React.js: A JavaScript library for building user interfaces.
Spring Boot: A Java-based framework for building web applications.
External APIs: Used to retrieve real-time cryptocurrency price data.
HTML, CSS, and JavaScript: Standard web development technologies for frontend design and interactivity.

CHAPTER 2

Feasibility Study

Depending on the results of the initial investigation the survey is now expanded to a more detailed feasibility study. “FEASIBILITY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources. It focuses on these major questions: What are the user’s demonstrable needs and how does a system meet them? What resources are available for given system? What are the likely impacts of the system on the organization? Whether it is worth to solve the problem? During feasibility analysis for this project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does this. Steps in feasibility analysis Eight steps involved in the feasibility analysis are: Form a project team and appoint a project leader Prepare system flowcharts. Enumerate potential proposed system. Define and identify characteristics of proposed system. Determine and evaluate performance and cost effectiveness of each proposed system. Weight system performance and cost data. Select the best-proposed system. Prepare and report final project directive to management.

2.1 Technical Feasibility

Overview: Technical feasibility assesses whether the proposed system can be successfully developed and implemented using the available technology and resources.

2.2.1 Key Points:

Technology Stack: Evaluate the suitability of the chosen technologies (React.js for frontend, Spring Boot for backend) for building the application. Consider factors such as compatibility, scalability, performance, and support.

Development Tools: Assess the availability and proficiency of development tools, libraries, and frameworks required for implementing the features of the application. Consider factors such as IDEs, version control systems, testing frameworks, etc.

Integration with External APIs: Evaluate the feasibility of integrating external APIs for fetching real-time cryptocurrency price data. Consider factors such as API documentation, rate limits, authentication mechanisms, and data format compatibility.

2.1.2 Challenges and Solutions:

Technology Selection: Choosing appropriate technologies that align with project requirements and development team expertise is crucial. To address this, thorough research and prototyping were conducted to evaluate different technology options and assess their suitability for the project.

API Integration: Integrating external APIs for real-time data retrieval can pose challenges such as handling rate limits, managing authentication tokens, and processing large volumes of data. To overcome this, rate-limiting strategies, caching mechanisms, and efficient data processing techniques were implemented.

2.2 Economical Feasibility

Overview: Economical feasibility examines the financial aspects of the project, including development costs, potential revenue streams, and return on investment.

2.2.1 Key Points:

Development Costs: Estimate the costs associated with development, including software licenses, developer salaries, infrastructure costs, etc. Compare these costs with the available budget to determine financial feasibility.

Revenue Streams: Identify potential revenue streams for the application, such as subscription plans, advertising, affiliate marketing, etc. Evaluate the market demand and competition to assess the viability of these revenue streams.

Scalability and ROI: Consider the scalability of the application in terms of user growth and revenue generation potential. Calculate the projected return on investment (ROI) based on revenue projections and development costs.

2.2.2 Challenges and Solutions:

Cost Estimation: Accurately estimating development costs and projecting revenue streams can be challenging due to uncertainties in market dynamics and user adoption. To address this, a detailed cost-benefit analysis and market research were conducted to inform financial projections.

Monetization Strategies: Identifying viable monetization strategies requires understanding user needs and market trends. Experimentation with different monetization models and continuous refinement based on user feedback and market analysis helped identify effective revenue streams.

2.3 Operational Feasibility

Overview: Operational feasibility assesses whether the proposed system can be effectively integrated into existing operational processes and workflows.

2.3.1Key Points:

User Adoption: Evaluate the willingness of users to adopt the new system and any potential resistance to change. Consider factors such as user training, usability, and perceived benefits of the new system.

Integration with Existing Systems: Assess the compatibility of the new system with existing infrastructure, software, and workflows. Identify potential challenges and opportunities for integration.

Scalability and Maintenance: Consider the long-term scalability and maintenance requirements of the system. Evaluate the availability of resources and expertise required to support ongoing operations and updates.

2.3.2 Challenges and Solutions:

Change Management: Addressing user resistance to change and ensuring smooth transition to the new system requires effective change management strategies. This involved conducting user training sessions, providing documentation and support resources, and soliciting feedback from stakeholders.

Legacy System Integration: Integrating the new system with existing legacy systems can be complex due to differences in technology stacks, data formats, and business processes. To overcome this, APIs, middleware, and data migration tools were utilized to facilitate seamless integration and data exchange.

2.4 Schedule Feasibility

Overview: Schedule feasibility evaluates whether the project can be completed within the specified time frame.

2.4.1 Key Points:

Project Timeline: Define a detailed project timeline with key milestones and deliverables. Allocate resources and estimate task durations to ensure realistic scheduling.

Resource Availability: Assess the availability of human resources, equipment, and infrastructure required for project execution. Identify any potential bottlenecks or resource constraints.

Risk Management: Identify potential risks and uncertainties that may impact project schedule. Develop contingency plans and mitigation strategies to address these risks proactively.

2.4.2 Challenges and Solutions:

Resource Constraints: Limited availability of resources, such as skilled developers or specialized equipment, can impact project scheduling. To mitigate this, resource allocation was optimized, and additional resources were allocated as needed to meet project deadlines.

Scope Changes: Scope changes or unforeseen requirements can disrupt project schedules. Agile development methodologies were adopted to facilitate flexibility and adaptability in response to changing project requirements, allowing for iterative development and continuous refinement of the project schedule.

CHAPTER 3

Architectural Design

Architectural design represents the structure of data and program components that are required to build a computer-based system. It considers the architectural style that the system will take, the structure and properties of the components that constitute the system, and the interrelationships that occur among all architectural components of a system.

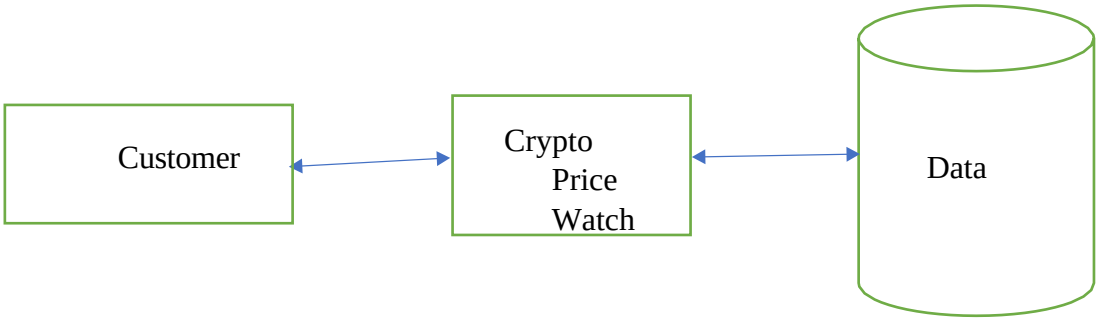

Figure 3.1 Architectural Design

3.1 ER Diagram

The Entity-Relationship (ER) diagram provides a visual representation of the database schema and the relationships between different entities in the system. In the case of Crypto Price Watch, the ER diagram might include entities such as Users, Cryptocurrencies, Prices, etc. Here's a brief description of the entities and their relationships:

User: Represents the users of the application. Each user can have attributes such as UserID, Username, Email, Password, etc.

Cryptocurrency: Represents the cryptocurrencies available for monitoring. Each cryptocurrency can have attributes such as CryptoID, Name, Symbol, etc.

Price: Represents the historical price data for cryptocurrencies. Each price entry can have attributes such as PriceID, CryptoID (foreign key), Timestamp, Value, etc.

User-Cryptocurrency Relationship: Represents the relationship between users and the cryptocurrencies they are monitoring. This may be a many-to-many relationship, as users can

monitor multiple cryptocurrencies and each cryptocurrency can be monitored by multiple users.

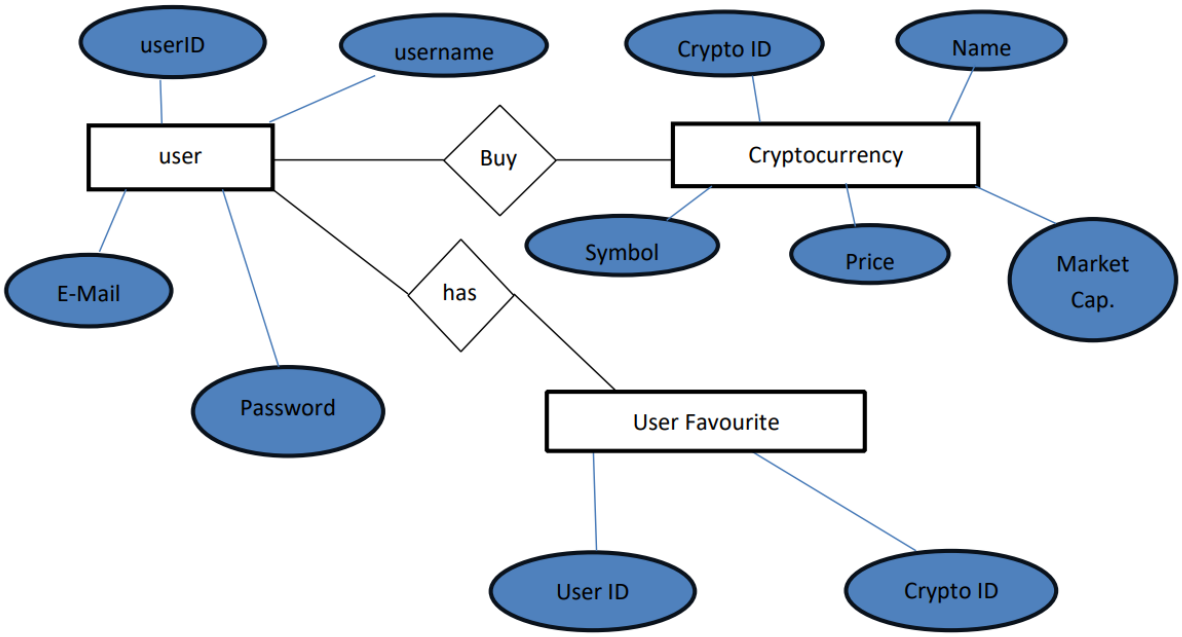

Figure 3.2 ER diagram

3.2 Data Flow Diagram

The Data Flow Diagram (DFD) illustrates the flow of data within the system, including processes, data stores, and data flows. In the case of Crypto Price Watch, the DFD might include components such as User Interface, Backend Server, External APIs, Database, etc. Here's a brief description of the components and their interactions:

User Interface: Represents the frontend interface of the application, where users interact with the system to view cryptocurrency prices and perform searches.
Backend Server: Represents the backend server responsible for processing user requests, fetching data from external APIs, and interacting with the database.
External APIs: Represents the external APIs used to retrieve real-time cryptocurrency price data.
Database: Represents the database used to store user information, cryptocurrency data, and historical price data.

LEVEL 0 DFD

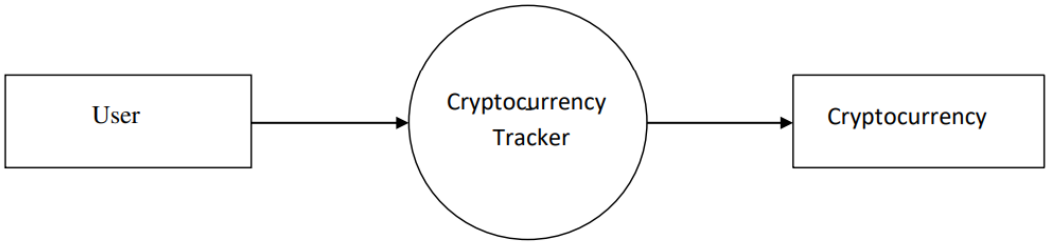

Figure 3.3 Level 0 DFD

Level 1 DFD:

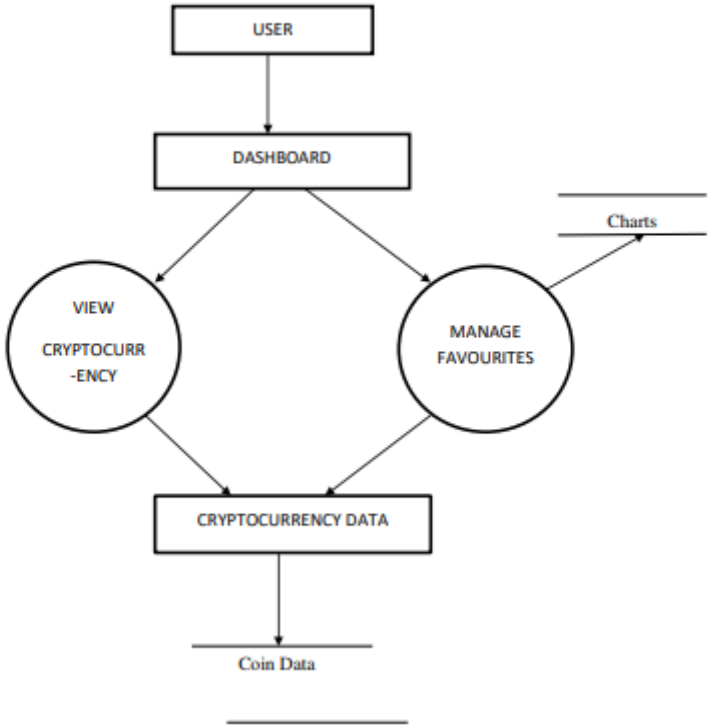

Figure 3.3 Level 1 DFD

3.3 Use Case Diagram

The Use Case Diagram illustrates the various use cases or interactions between actors (users) and the system. In the case of Crypto Price Watch, the Use Case Diagram might include actors such as Registered User, Guest User, Admin, etc. Here's a brief description of the actors and their interactions:

Registered User: Represents users who have registered and logged into the system. Use cases for registered users might include View Prices, Search Cryptocurrencies, Manage Watchlist, etc.

Guest User: Represents users who are not logged into the system. Use cases for guest users might include View Prices, Search Cryptocurrencies, Sign Up, Log In, etc.
Admin: Represents administrators or moderators who have special privileges within the system. Use cases for admins might include Manage Users, Manage Cryptocurrencies, etc.

CHAPTER 4

System Design

System design is a solution, a ‘how to’ approach to the creation of a new system. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Emphasis is on translating the performance requirements into design specifications.
Design goes through logical and physical stages of development. Logical design reviews the present physical system; prepared input and output specifications; details the implementation plan; and prepares a logical design walkthrough. The physical design maps out the details of the physical system, plans the system implementation, devises a test and implementation plan, and specifies any new hardware and software.

4.1 Architecture

The architecture of Crypto Price Watch follows a client-server model, which is a common architectural pattern in web applications. In this model:

Client: The client-side of the application is responsible for presenting the user interface and handling user interactions. In the case of Crypto Price Watch, the client-side is implemented using React.js, a popular JavaScript library for building interactive user interfaces. React.js allows for the creation of dynamic, single-page applications (SPAs) where content is dynamically updated without requiring full page reloads.

Server: The server-side of the application manages the business logic, data processing, and communication with external services. Crypto Price Watch's server-side is built using Spring Boot, a Java-based framework for building web applications. Spring Boot simplifies the process of developing robust, production-ready web applications by providing out-of-the-box features such as dependency injection, security, and configuration.

The client and server communicate with each other using HTTP (Hypertext Transfer Protocol), a standard protocol for transmitting data over the web. The client sends requests to the server, which processes the requests, performs any necessary operations, and sends back responses containing the requested data.

The client-server architecture of Crypto Price Watch provides several benefits, including scalability, maintainability, and separation of concerns. By separating the user interface from the backend logic, the application becomes easier to maintain and scale as each component can be developed, tested, and deployed independently.

4.1.1 Database Structure:

The database structure of Crypto Price Watch is designed to efficiently store and manage various types of data required by the application. The application utilizes a relational database management system (RDBMS) to organize data into structured tables with predefined relationships.

Here's a breakdown of the main tables in the database:

User Table: This table stores information about registered users, such as user ID, username, email, password, registration date, and role. Each user is assigned a unique user ID, which serves as the primary key for the table.

Personal Table: This table stores additional personal information about users, such as their first name, last name, gender, birthdate, address, and phone number. The personal information is associated with the corresponding user through a foreign key relationship.

Education Table: This table stores educational background information about users, including their degree, school, major, and graduation date. Similar to the personal table, the education information is linked to the user through a foreign key relationship.

Other Table: This table serves as a placeholder for additional data that may be required by the application in the future. It can be customized to store various types of data based on the specific requirements of the application.

By organizing data into structured tables with defined relationships, the database structure ensures data integrity, consistency, and efficiency in storing and retrieving information.

4.2 Component Interactions:

The interaction between different components of Crypto Price Watch is crucial for the seamless operation of the application. Here's how the main components interact with each other:

User Interface (UI): The frontend user interface is responsible for presenting the application's features and allowing users to interact with the system. The UI is implemented using React.js components, which render dynamically based on user actions and data received from the server.

Backend Server: The backend server handles user requests, processes business logic, and interacts with the database and external services. It exposes RESTful APIs (Application Programming Interfaces) that the frontend client can communicate with to

perform various actions such as user authentication, fetching cryptocurrency data, and updating user information.

External APIs: The application integrates with external APIs to retrieve real-time cryptocurrency price data. These APIs provide endpoints for accessing market data, historical price information, and other relevant metrics. The backend server communicates with these APIs using HTTP requests to fetch the required data, which is then processed and sent back to the frontend client.

Database: The relational database stores persistent data required by the application, including user information, cryptocurrency data, and historical price data. The backend server interacts with the database using SQL (Structured Query Language) queries to perform CRUD operations (Create, Read, Update, Delete) on the data.

Overall, the interaction between these components forms the backbone of Crypto Price Watch, allowing users to seamlessly monitor cryptocurrency prices in real-time while ensuring data integrity, security, and performance. The client-server architecture, database structure, and component interactions are carefully designed to provide a robust and scalable platform for users to make informed investment decisions in the dynamic world of cryptocurrencies.

CHAPTER 5

Future Enhancements

5.1 Advanced Analytics:

Description: Implement advanced analytics features to provide users with deeper insights into cryptocurrency trends, market sentiment, and price movements. This could include:

Historical Data Analysis: Enable users to analyze historical price data using interactive charts, graphs, and technical indicators. Provide tools for trend analysis, pattern recognition, and statistical modeling to identify trading opportunities.

Sentiment Analysis: Integrate sentiment analysis algorithms to analyze social media, news articles, and other sources for sentiment indicators related to specific cryptocurrencies. Visualize sentiment trends and correlations with price movements to help users make informed decisions.

Portfolio Performance Tracking: Allow users to track the performance of their cryptocurrency portfolios over time. Provide analytics on portfolio diversification, risk exposure, and returns compared to benchmarks or indices.

Challenges: Developing robust analytics features requires advanced data processing, visualization, and statistical analysis capabilities. Integrating external data sources and ensuring data accuracy and reliability are also challenges.

Solutions: Collaborate with data analytics experts and leverage existing libraries, APIs, and frameworks for data visualization and analysis. Implement data validation and quality control measures to ensure accuracy and reliability of analytics results.

5.2. Personalized Alerts and Notifications:

Description: Enhance user engagement and retention by offering personalized alerts and notifications based on user preferences and trading activities. This could include:

Price Alerts: Allow users to set custom price alerts for specific cryptocurrencies. Notify users via email, SMS, or push notifications when price thresholds are reached or significant price movements occur.

News Alerts: Deliver real-time news alerts related to cryptocurrencies, blockchain technology, regulatory developments, and market events. Use natural language processing (NLP) algorithms to filter and prioritize relevant news articles based on user interests.

Event Reminders: Provide reminders for upcoming events such as ICO launches, project updates, conferences, and regulatory announcements. Allow users to subscribe to event calendars and customize their event preferences.

Challenges: Implementing real-time alerts and notifications requires efficient event processing, notification delivery mechanisms, and user preference management. Ensuring timely and reliable delivery of notifications across different communication channels is also a challenge.

Solutions: Utilize scalable event-driven architecture and messaging systems for processing and delivering notifications in real-time. Implement user preference management systems to allow users to customize their notification settings and delivery preferences.

5.3 Social Features and Community Engagement:

Description: Foster a sense of community and social interaction among users by introducing social features and collaboration tools. This could include:

Discussion Forums: Create discussion forums or community boards where users can share insights, discuss trading strategies, and ask questions. Provide moderation tools to ensure a constructive and respectful community environment.

User Profiles: Allow users to create profiles and connect with other users. Enable profile customization, follower/following relationships, and activity feeds to facilitate social networking and user engagement.

Trading Groups: Enable users to form or join trading groups based on common interests, investment strategies, or cryptocurrency preferences. Provide group chat, collaboration tools, and shared watchlists to facilitate group interaction and collaboration.

Challenges: Building social features requires addressing privacy and security concerns, implementing moderation and content filtering mechanisms, and fostering community norms and guidelines.

Solutions: Implement robust privacy controls and user moderation tools to ensure user safety and compliance with community guidelines. Provide reporting and flagging mechanisms for inappropriate content and behavior. Encourage positive community engagement through gamification, rewards, and incentives.

5.4. Enhanced User Customization Options:

Description: Empower users to personalize their experience and tailor the application to their preferences. This could include:

Customizable Dashboards: Allow users to customize their dashboard layout, widgets, and data visualization preferences. Provide drag-and-drop functionality and widget libraries for easy customization.

Theme Customization: Offer theme options and color schemes to allow users to personalize the application's visual appearance. Provide dark mode/light mode options and customizable UI elements to enhance user comfort and accessibility.

Language Support: Expand language support to cater to a global audience. Provide multilingual interfaces and localization options to accommodate users from different regions and language backgrounds.

Challenges: Implementing user customization features requires flexibility in UI design, backend architecture, and data presentation. Ensuring consistent user experience across different customization options and devices is also challenging.

Solutions: Adopt modular UI design principles and component-based architecture to facilitate easy customization and flexibility. Use localization frameworks and internationalization best practices to support multiple languages and cultural preferences. Conduct user testing and feedback sessions to iterate and refine customization options based on user preferences and feedback.

5.5. Educational Resources and Learning Tools:

Description: Provide educational resources and learning tools to help users understand cryptocurrency markets, trading strategies, and blockchain technology. This could include:

Educational Articles and Tutorials: Publish articles, tutorials, and guides on topics such as cryptocurrency basics, blockchain technology, trading strategies, and investment fundamentals.

Interactive Learning Modules: Develop interactive learning modules, quizzes, and simulations to engage users and reinforce learning concepts. Provide gamified learning experiences and progress tracking to incentivize continuous learning.

Market Insights and Analysis: Offer market insights, analysis reports, and expert commentary on cryptocurrency trends, market dynamics, and regulatory developments. Provide research tools and data visualization tools to help users interpret market data and make informed decisions.

Challenges: Creating high-quality educational content requires expertise in cryptocurrency markets, blockchain technology, and instructional design. Balancing educational content with user engagement and entertainment value is also challenging.

Solutions: Collaborate with subject matter experts, educators, and industry professionals to develop authoritative and engaging educational content. Utilize multimedia formats such as videos, infographics, and interactive tools to enhance learning experiences. Leverage data

analytics to track user engagement and learning outcomes, and iterate on content based on user feedback and performance metrics.

By incorporating these future enhancements, Crypto Price Watch can further differentiate itself in the market and provide added value to its users, fostering long-term engagement and loyalty.

CHAPTER 6

Implementation

In this section, we'll provide an overview of how each feature of the Crypto Price Watch application was implemented, along with the challenges faced during development and how they were overcome.

6.1 Real-Time Price Monitoring:

Overview: Real-time price monitoring is a core feature of Crypto Price Watch, allowing users to track cryptocurrency prices dynamically.

Implementation: This feature was implemented by integrating external APIs that provide real-time cryptocurrency price data. The frontend continuously fetches updated price information from the backend server, which in turn retrieves data from the external APIs. WebSocket technology could also be utilized for real-time data streaming.

6.1.2 Challenges Faced:

API Rate Limits: One challenge was managing API rate limits, especially when dealing with high volumes of user requests. This could lead to throttling or blocking of requests if not handled properly.

Data Synchronization: Ensuring that the displayed prices are always up-to-date and synchronized with the latest market data posed a challenge, particularly when dealing with fluctuating prices and high-frequency updates.
Solutions:

Rate Limiting Strategies: Rate-limiting mechanisms were implemented to control the frequency of API requests and prevent exceeding rate limits. This involved optimizing the frequency of data polling and implementing caching mechanisms to reduce the number of API calls.
Data Caching: Caching mechanisms were employed to store frequently accessed data locally, reducing the need for repeated API calls. This helped improve performance and reduce latency in fetching real-time price data.

6.2 Search Functionality:

Overview: The search functionality allows users to search for specific cryptocurrencies by name or symbol.

Implementation: The search feature was implemented using a combination of frontend and backend components. On the frontend, an interactive search bar was developed to send search queries to the backend server. On the backend, search functionality was implemented to query the database for matching cryptocurrencies based on the user's input.

6.2.1 Challenges Faced:

Optimizing Search Queries: Designing efficient search algorithms to handle a large volume of data and provide quick response times posed a challenge, especially when dealing with complex search queries or partial matches.
Handling Edge Cases: Handling edge cases such as misspelled or incomplete search queries required careful consideration to ensure accurate and relevant search results.
Solutions:

Query Optimization: The search algorithm was optimized to efficiently search through the database and retrieve relevant results in real-time. This involved indexing database fields, optimizing SQL queries, and utilizing database query optimization techniques.
Fuzzy Search: Fuzzy search algorithms were implemented to handle partial matches and misspelled search queries, improving the robustness and accuracy of the search functionality.

6.3 User Authentication:

Overview: User authentication is essential for secure access to the application's functionalities.

Implementation: User authentication was implemented using industry-standard protocols such as JSON Web Tokens (JWT) and bcrypt for password hashing. The backend server handles user registration, login, and session management, while the frontend provides user interfaces for authentication-related actions.

6.3.1 Challenges Faced:

Security Vulnerabilities: Implementing robust security measures to prevent unauthorized access, protect user data, and mitigate potential security threats such as SQL injection or cross-site scripting (XSS) attacks posed a significant challenge.
Authentication Flow: Designing a seamless and intuitive authentication flow that balances security with user experience required careful consideration of user interface design and backend authentication mechanisms.

6.3.2 Solutions:

Security Best Practices: Security best practices were followed throughout the development process, including input validation, parameterized queries, and protection against common web vulnerabilities. Regular security audits and penetration testing were conducted to identify and address security vulnerabilities.
Authentication Middleware: Middleware components were implemented on the backend server to handle authentication-related tasks such as verifying JWT tokens, hashing passwords securely, and managing user sessions. This helped streamline the authentication process and ensure consistent security across the application.

6.4 User Interface Design:

Overview: The user interface design plays a crucial role in providing users with a seamless and intuitive experience.

Implementation: The frontend user interface was developed using React.js components, which render dynamically based on user actions and data received from the server. The user interface design focused on usability, responsiveness, and visual appeal to enhance the overall user experience.

6.4.1 Challenges Faced:

Cross-Browser Compatibility: Ensuring consistent user experience across different web browsers and devices posed a challenge, as rendering may vary based on browser compatibility and screen resolutions.
Performance Optimization: Optimizing the performance of the user interface, especially for large datasets and complex components, required careful consideration of rendering performance, code optimization, and resource management.
Solutions:

Responsive Design: Responsive design techniques were employed to adapt the user interface layout and styling based on the user's device screen size and orientation. This involved using CSS media queries, flexible layouts, and responsive design frameworks such as Bootstrap.
Performance Profiling: Performance profiling tools were used to identify and address performance bottlenecks in the user interface. Code splitting, lazy loading, and memorization techniques were employed to optimize rendering performance and improve overall responsiveness.

Overall, overcoming these challenges required a combination of technical expertise, problem-solving skills, and collaboration between frontend and backend development teams. By addressing these challenges effectively, Crypto Price Watch was able to deliver

a robust, secure, and user-friendly application for monitoring cryptocurrency prices in real-time.

Chapter 7

Testing

7.1 Test Case: Search Functionality

The search functionality allows users to search for specific cryptocurrencies by name or symbol. To ensure that this feature functions as intended and provides an optimal user experience, a series of test cases were developed and executed. These test cases cover various scenarios and edge cases to validate the accuracy, efficiency, and robustness of the search functionality.

7.1.1 Test Case 1: Basic Search

Objective: Verify that the search functionality returns accurate results for a basic search query.

Test Data: Enter a valid cryptocurrency name or symbol (e.g., "Bitcoin" or "BTC") into the search bar.
Expected Result: The application should display relevant cryptocurrency information, including the name, symbol, and current price of the searched cryptocurrency.
Actual Result: Verify that the displayed information matches the expected cryptocurrency based on the search query.

7.1.2 Test Case 2: Partial Match

Objective: Verify that the search functionality handles partial matches and returns relevant results.

Test Data: Enter a partial cryptocurrency name or symbol (e.g., "Eth" for Ethereum) into the search bar.
Expected Result: The application should display relevant cryptocurrency information for all matches that contain the partial search query.
Actual Result: Verify that the displayed information includes all relevant cryptocurrencies containing the partial search query.

7.1.3 Test Case 3: No Match Found

Objective: Verify that the search functionality handles cases where no matching cryptocurrencies are found.

Test Data: Enter a search query for a cryptocurrency that does not exist
(e.g., "NonExistentCoin") into the search bar.
Expected Result: The application should display a message indicating that no matching cryptocurrencies were found.
Actual Result: Verify that the application displays the appropriate message when no matching cryptocurrencies are found.

7.1.4 Test Case 4: Case Insensitivity

Objective: Verify that the search functionality is case-insensitive and treats uppercase and lowercase characters equally.

Test Data: Enter a search query with mixed case characters (e.g., "eThErEuM") into the search bar.
Expected Result: The application should treat the search query as case-insensitive and return relevant results regardless of the case of the input.
Actual Result: Verify that the application correctly handles case-insensitive search queries and returns relevant results.

7.1.5 Test Case 5: Empty Search

Objective: Verify that the search functionality handles empty search queries gracefully.

Test Data: Leave the search bar empty and submit the search query.
Expected Result: The application should display a message prompting the user to enter a search query.
Actual Result: Verify that the application displays the appropriate message when the search query is empty.

7.1.6 Test Case 6: Performance

Objective: Evaluate the performance of the search functionality under various load conditions.

Test Data: Conduct multiple concurrent search queries with varying search terms and observe the response time.
Expected Result: The application should maintain responsive performance and return search results within a reasonable time frame, even under heavy load conditions.
Actual Result: Measure the response time for each search query and ensure that the application meets performance requirements.

Chapter 8

Snapshot

7.1 Home Page:

Figure 7.1 Home Page

7.2 Login Page:

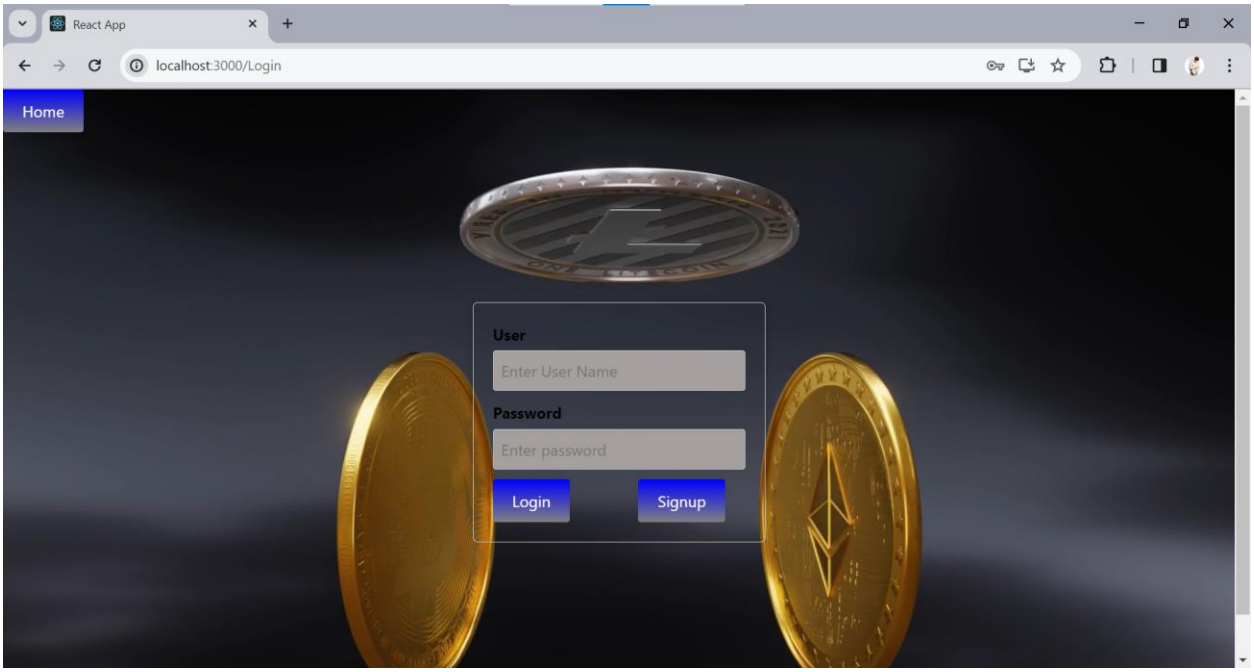

Figure 7.2 Login Page

7.3 Signup Page:

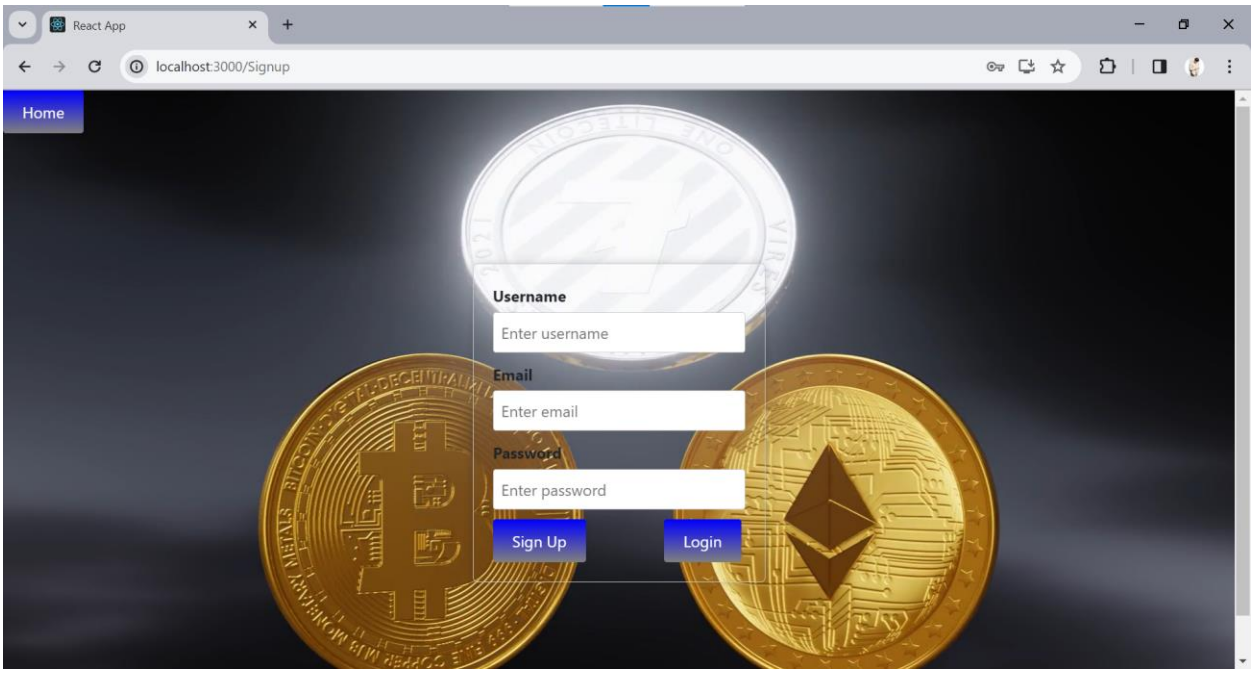

Figure 7.3 Signup Page

7.4 Search Page:

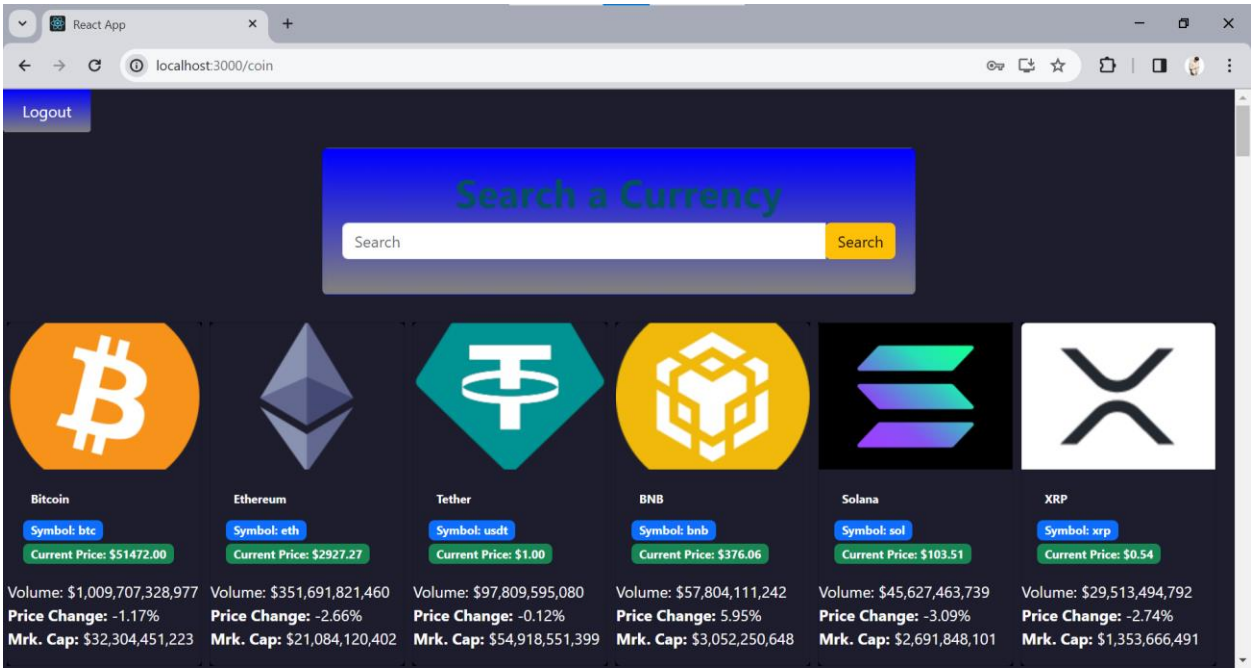

Figure 7.4 Search Page

7.5 Cryptocurrency Search:

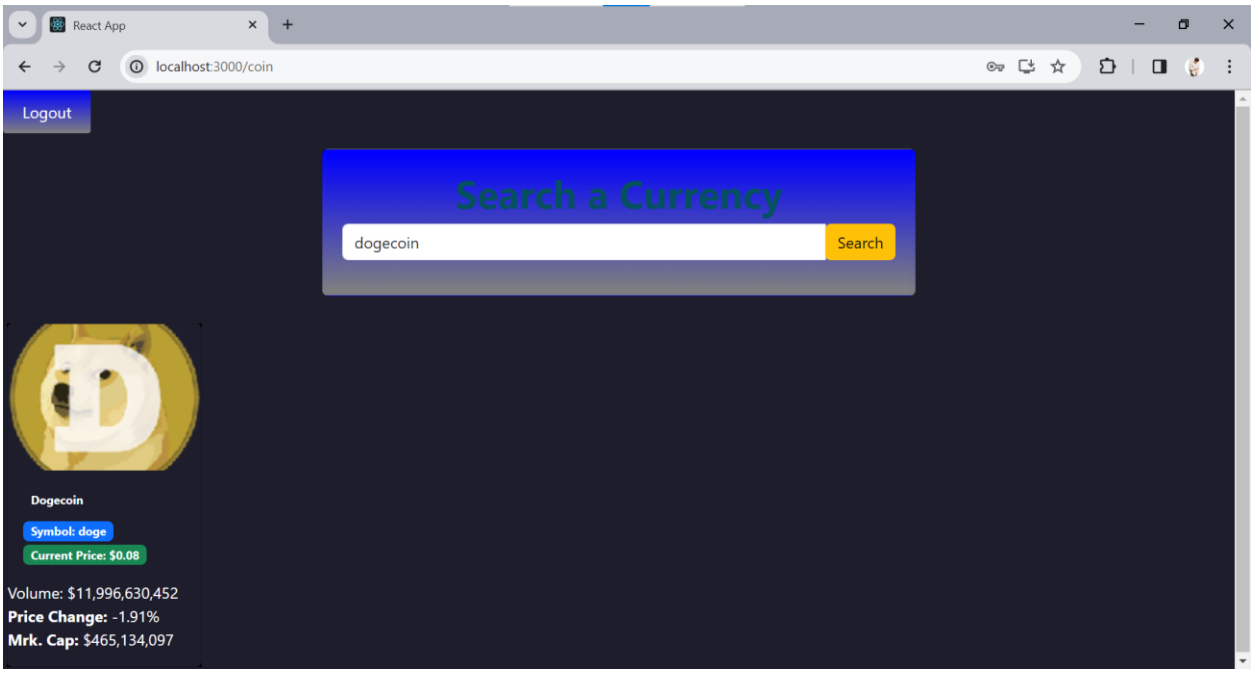

Figure 7.5 Cryptocurrency Page

CHAPTER 9

Conclusion

The Crypto Price Watch project has been successfully developed, aiming to provide users with a convenient and efficient platform for real-time cryptocurrency price monitoring. Throughout the project, various features such as real-time price monitoring, search functionality, and user authentication were implemented using modern technologies like React.js and Spring Boot.

The project report provides a comprehensive overview of the development process, including feasibility studies, architectural design, implementation details, and testing strategies. Key aspects such as technical feasibility, economic feasibility, and operational feasibility were thoroughly evaluated to ensure the viability and success of the project.

7.1 Reflection

The successful completion of the Crypto Price Watch project highlights the effectiveness of collaboration, problem-solving, and agile development methodologies. Despite facing challenges such as API integration complexities, security considerations, and performance optimization, the development team demonstrated resilience and adaptability in overcoming these obstacles.

One of the key lessons learned during the development process is the importance of thorough planning and requirement analysis. Clear understanding of project objectives, user needs, and technical requirements is essential for guiding the development process and ensuring successful project outcomes.

Additionally, effective communication and collaboration among team members played a crucial role in driving project progress and resolving challenges in a timely manner. Regular meetings, code reviews, and feedback sessions facilitated a collaborative environment where ideas could be exchanged, issues addressed, and solutions implemented efficiently.

Moving forward, continuous iteration and improvement are essential for maintaining the success of the Crypto Price Watch application. User feedback, performance metrics, and market trends should be carefully monitored to identify areas for enhancement and refinement.

Overall, the Crypto Price Watch project represents a significant achievement in leveraging technology to address the evolving needs of cryptocurrency enthusiasts. By delivering a robust, user-friendly platform for real-time price monitoring, the project aims to empower users with the tools and information they need to make informed decisions in the dynamic world of cryptocurrencies.

CHAPTER 10

References

1. React.js Documentation. Retrieved from: https://reactjs.org/docs/getting-started.html
2. Spring Boot Documentation. Retrieved from: https://spring.io/projects/spring-boot
3. External APIs for Cryptocurrency Price Data. Retrieved from: [Insert API source here]
4. Online Courses on React.js and Spring Boot on platforms like LinkedIn Learning, Udemy, and Coursera.
5. Various tutorials and guides from Google, YouTube, and community forums for troubleshooting and learning specific development techniques.