Cipher Chain
Enterprise Crypto Intelligence & User Management Platform

CMPE-272 – Enterprise Software Platform

Fall 2025

Authors:

Mohammed Mominuddin

Shreram Palanisamy

December 2025

ABSTRACT

Cipher Chain

Cipher Chain is an enterprise-grade FinTech platform that combines secure multi-user authentication, role-based access control (RBAC), audit logging, and real-time crypto market intelligence. The platform is designed for financial institutions, crypto exchanges, investment firms, and internal enterprise teams that require secure access governance and transparent operational visibility.

End-users such as traders, analysts, and investors access the platform to view dashboards, technical charts, and market insights, while administrators manage user accounts, permissions, and activity monitoring via a dedicated admin console. The system is implemented as a single-page application (SPA) using React and Tailwind CSS, with Firebase providing authentication, Firestore database, and serverless infrastructure. TradingView is used to embed professional-grade financial charts.

This project report documents the motivation, requirements, design, implementation, and testing of Cipher Chain. It demonstrates how enterprise software engineering principles—such as security-by-design, scalability, and observability—can be applied to build a production-ready FinTech platform suitable for real-world deployment.

Acknowledgements

We would like to express our sincere gratitude to our course instructor Mr. Andrew H. Bond for their guidance throughout the CMPE-272 Enterprise Software Platform course. Their feedback on system architecture, security, and enterprise design patterns helped us refine Cipher Chain into a more robust and realistic platform.

We would also like to thank our classmates for their constructive comments during project reviews and presentations, which helped us identify gaps in our design and implementation and improve the final system.

Table of Contents

1. Introduction

1.1. Project goals and objectives
1.2. Problem and motivation
1.3. Project application and impact
1.4. Project results and deliverables
1.5. Market research
1.6. Project report structure

2. Project Background and Related Work

2.1. Background and used technologies
2.2. State-of-the-art technologies
2.3. Literature survey

3. System Requirements and Analysis

3.1. Domain and business requirements
3.2. Customer-oriented requirements
3.3. System function requirements
3.4. System behavior requirements
3.5. System performance and non-function requirements
3.6. System context and interface requirements
3.7. Technology and resource requirements

4. System Design

4.1. System architecture design
4.2. System data and database design
4.3. System interface and connectivity design
    4.3.1. System user interface design
    4.3.2. System component API and logic design
4.4. System design problems, solutions, and patterns

5. System Implementation

5.1. System implementation summary
5.2. System implementation issues and resolutions
5.3. Used technologies and tools

6. System Testing and Experiment

6.1. Testing and experiment scope
6.2. Testing and experiment approaches
6.3. Testing report (or case study results)

7. Conclusion and Future Work

7.1. Project summary

7.2. Future work

List of Tables

Table 1. *Domain and business requirements ……Page No. 12*
of the Cipher Chain platform

Table 2. *Key functional requirements derived from ……Page No.13*
the business needs of Cipher Chain.

Table 3. *Summary of manual test cases for Cipher Chain ……Page No. 21*

1. Introduction

1.1 Project goals and objectives

The goal of Cipher Chain is to build an enterprise-grade FinTech platform that offers secure multi-user access, real-time crypto intelligence, and centralized administration. The project aims to provide a single web portal where traders and analysts can monitor markets, while admins manage users, roles, and system activity. Our objectives include implementing strong authentication with Google SSO, enforcing role-based access control, and capturing detailed audit logs for every sensitive action. Academically, the project demonstrates how enterprise security, scalability, and usability principles can be applied to design and implement a realistic financial software platform.

1.2 Problem and motivation

The problem motivating Cipher Chain is that many student or hobby finance dashboards only display price charts and ignore real enterprise requirements such as security, governance, and compliance. Financial institutions, crypto exchanges, and internal risk teams need platforms where every user action is controlled, monitored, and auditable. Our project addresses this gap by integrating authentication, role-based access, admin controls, and audit logs into a single system. Technically, we contribute a secure architecture using React and Firebase, while academically we explore how enterprise patterns can be applied to modern cloud-based FinTech applications.

1.3 Project application and impact

Cipher Chain can be applied in FinTech startups, crypto exchanges, and internal enterprise teams that need secure access to financial dashboards. Traders and analysts use the platform to monitor real-time crypto markets, indices, and trends, while administrators manage accounts, permissions, and audit logs. This improves visibility, security, and accountability across the organization. The platform also supports compliance teams by providing clear records of who did what and when. The impact is a more controlled and trustworthy environment for financial decision-making, reducing operational risk and demonstrating how enterprise-grade security can be integrated into modern web-based analytics tools.

1.4 Project results and deliverables

The project produced a working web application that implements secure authentication, role-based access control, an admin console, and audit logging, along with dashboards for

crypto charts and market insights. Key deliverables include the deployed Cipher Chain platform, source code repository, configuration for Firebase services, and database collections for users and audit logs. We also delivered presentation slides, a demo walkthrough, and this detailed project report documenting requirements, design, implementation, and testing. Together, these deliverables show a complete lifecycle: from concept and enterprise use case definition to a functional prototype that demonstrates realistic FinTech and security capabilities.

1.5 Market research

We analyzed several existing platforms to shape Cipher Chain's design. From FinTech products like Coinbase, Binance, and TradingView, we adopted ideas for interactive charts, watchlists, and market dashboards. From enterprise tools such as Okta, Google Admin Console, and AWS IAM, we studied how user management, permissions, and security controls are presented. We also reviewed security and logging approaches used by companies like Splunk and CrowdStrike. This market research revealed consistent patterns: strong identity management, clear separation of admin and user roles, dashboards with real-time data, and extensive audit trails. Cipher Chain intentionally mirrors these proven enterprise practices.

1.6 Project report structure

This project report is organized to follow the natural flow of how Cipher Chain was conceived, designed, and built. It begins with an introduction that explains the overall idea of the platform, its goals, and why a secure, enterprise-style FinTech system is needed. After setting this context, the discussion moves into background information and related products, showing how existing trading dashboards, admin consoles, and security tools inspired the concept and feature set of Cipher Chain.

Next, the report formalizes what the system must do by defining business needs, user requirements, and the main functional and non-functional requirements. Once the requirements are clear, the narrative transitions into the overall solution design, including the architecture, data model, and user interface flow for both regular users and administrators.

Following the design, the report describes the implementation details: technologies selected, key components, and important technical decisions made while coding the application. After implementation, the focus shifts to testing, where different scenarios

2. Project Background and Related Work

2.1 Background and used technologies

Cipher Chain is built at the intersection of financial technology, cybersecurity, and modern web engineering. The platform targets environments where secure access to crypto and market insights is critical, and where user governance and auditability are mandatory. To support this, we use a modern JavaScript stack centered around React for building a responsive single-page application. Tailwind CSS provides fast, consistent styling and helps us maintain a clean dashboard layout. React Router manages client-side navigation between user and admin views. On the backend side, we adopt Firebase as a serverless platform: Firebase Authentication handles email/password and Google login, while Firestore stores user profiles, roles, and audit logs. Firebase security rules and custom claims are used to enforce role-based access control.

For financial data visualization, TradingView widgets are embedded to deliver professional-grade charts. This combination allows rapid development while still reflecting realistic enterprise patterns and constraints.

2.2 State-of-the-art technologies

Modern enterprise FinTech and security platforms share several state-of-the-art characteristics that influenced Cipher Chain's design. Many systems are cloud-native and rely on serverless or microservices architectures to scale with demand and reduce operational overhead. Identity and access management typically rely on standards such as OAuth2 and OpenID Connect, with strong support for SSO, MFA, and centralized role management.

Real-time data delivery is implemented using streaming APIs, WebSockets, or pub/sub messaging, enabling dashboards to update continuously without manual refresh. On the security and observability side, organizations integrate audit logging, monitoring, and alerting into every layer, often sending logs to centralized tools for analytics and compliance. Cipher Chain mirrors these trends at a smaller scale by using Firebase as a managed backend, Google-based SSO, and structured audit logs. Although it is an academic project, the underlying architecture and technology choices reflect the same principles used by leading FinTech and cybersecurity products.

2.3 Literature survey

The literature and documentation reviewed for Cipher Chain spans three main areas: secure authentication and RBAC, audit logging and compliance, and financial data visualization. Best practices from security guidelines and industry blogs emphasize principles such as least privilege, defense in depth, and secure session management. These sources highlight the importance of clearly separating roles, validating authorization at every sensitive operation, and avoiding implicit trust based solely on UI controls. Research and technical articles on logging and compliance stress that audit trails must be tamper-resistant, time-stamped, and sufficiently detailed to support forensic analysis and regulatory reviews.

In parallel, resources on financial dashboards and visualization techniques focus on usability, clarity of chart interactions, and performance. Together, this body of work shaped our decisions to implement RBAC using Firebase custom claims, capture structured audit events for every critical action, and embed professional charting tools to meet both usability and transparency expectations.

3. System Requirements and Analysis

3.1 Domain and business requirements

Cipher Chain operates in the domain of enterprise FinTech and crypto analytics, where secure multi-user access and strong governance are critical. The main business need is to provide organizations—such as crypto exchanges, investment firms, and internal risk teams—with a single platform that combines market intelligence and user management. The core business requirements, including authentication, role separation, user lifecycle controls, and auditability, are summarized in **Table 1**. These requirements ensure that authorized users can safely access dashboards while administrators maintain full control over accounts, roles, and system activity. The solution is expected to reduce operational risk by preventing unauthorized access, detecting misuse early through logs, and simplifying onboarding, disabling, and removal of users, while supporting faster and more confident financial decision-making.

ID	Requirement description	Priority
BR1	System shall allow users to authenticate using email/password and Google SSO.	High
BR2	System shall distinguish between admin and regular user roles.	High
BR3	Admins shall manage user lifecycle (create, disable, delete accounts).	High
BR4	System shall record all critical actions in an audit log	High
BR5	Users shall access real-time crypto and market dashboards.	Medium

Table 1: Domain and business requirements of the Cipher Chain platform

3.2 Customer-oriented requirements

Cipher Chain serves two main customer groups: operational users and governance users. Operational users include traders, analysts, and investors who need fast access to real-time crypto prices, candlestick charts, and global indices. For them, the system should provide an intuitive dashboard, minimal login friction, and clear visualization of

key metrics like price movement, volume, and market news. Governance users include administrators, compliance staff, and internal IT teams. Their requirements focus on accurate user records, easy role assignment, the ability to disable or delete accounts, and visibility into who performed which actions. Both groups expect strong security without a confusing UI. The platform should be accessible from common desktop browsers, require no complex installation, and maintain a consistent look and feel across pages. Error messages, loading states, and feedback on actions (e.g., "user disabled successfully") must be clear so that customers can trust the system and operate efficiently.

3.3 System function requirements

Functionally, Cipher Chain must authenticate users, authorize access, manage data, and present analytics. At a high level, the system must support registration and login, secure session handling, role-based routing to user or admin dashboards, admin operations on accounts, and recording of all critical actions in audit logs. The main functional requirements derived from these needs are listed in **Table 2**, which maps each function to its related business requirement. On the analytics side, the platform must embed TradingView charts, display selected crypto pairs and indices, and allow admins to filter or search user records and audit entries. Together, these functions provide a complete flow from secure login to actionable insights and governance operations within a single web-based interface.

ID	Function description	Related BR
FR1	Authenticate user and start a secure session.	BR1
FR2	Retrieve user role (admin/user) and route to correct dashboard.	BR2
FR3	Allow admin to enable/disable/delete user accounts.	BR3
FR4	Create an audit log entry for each login/logout event.	BR4
FR5	Create an audit log entry for each admin action on a user account.	BR3, BR4
FR6	Display TradingView crypto charts and global indices to logged-in users.	BR5

Table 2: Key functional requirements derived from the business needs of Cipher Chain.

3.4 System behavior requirements

System behavior describes how Cipher Chain responds to various events and inputs over time. After successful authentication, users should be routed automatically to the appropriate dashboard based on their role. If a user with a non-admin role attempts to access an admin-only page, the system should deny access and redirect them to a safe location, such as the main dashboard, while optionally displaying an authorization error message. When an admin changes a user's status to "disabled," that user must be blocked from accessing protected content on any subsequent login attempt. Each critical event—logins, logouts, admin actions—should immediately produce an audit entry, even if the UI fails to update. In case of network or backend failures, the system should show friendly error messages and avoid leaving the UI in an inconsistent state. Behavior should remain predictable and deterministic, so repeated actions by the same role lead to the same outcomes.

3.5 System performance and non-function requirements

Non-functional requirements ensure Cipher Chain is usable and trustworthy in an enterprise context. The application should load the initial dashboard within a reasonable time on a standard broadband connection, and subsequent navigation between views should feel responsive, aided by client-side routing. All communication with Firebase and external APIs must use HTTPS to protect credentials and data. Authentication tokens should be handled securely and never exposed in logs or error messages. The system must enforce authorization checks on the backend side via Firebase rules, not only in the frontend UI. Availability should be high enough for classroom demonstration and small pilot use; Firebase's managed infrastructure supports this requirement. Audit logs are expected to be immutable and retained for a defined period, ensuring traceability of actions. The UI should be accessible on common desktop resolutions, follow consistent styling, and degrade gracefully if external widgets such as TradingView fail to load.

3.6 System context and interface requirements

Cipher Chain is a web-based client running in the browser that interacts with external services over the internet. Its main external interfaces are Firebase Authentication, Firestore, and TradingView widgets. The system must accept user credentials or Google OAuth tokens and exchange them securely with Firebase. The frontend reads and writes user and audit data through Firestore APIs, respecting security rules defined on the backend. TradingView components are embedded via JavaScript and require stable

network connectivity to retrieve market data. From the user's perspective, the primary interface is a navigable set of pages: login, user dashboard, admin dashboard, and audit logs. Input is provided through standard HTML controls such as forms, buttons, and tables, while output appears as charts, tables, and status messages.

3.7 Technology and resource requirements

To build and run Cipher Chain, developers require a machine with Node.js, npm, and a modern browser. A Firebase project must be provisioned with Authentication and Firestore enabled, along with appropriate security rules and custom claims configuration. Access to TradingView's public widgets or scripts is needed for chart embedding. A GitHub repository supports collaboration and version control. During development, tools such as browser devtools, React developer tools, and Firebase console are used for debugging and monitoring. Minimal hosting resources are needed because the frontend can be deployed as static files and the backend is serverless, making the project suitable for a student environment with limited infrastructure.

4. System Design

4.1 System architecture design

Cipher Chain follows a client–server architecture built on a serverless backend. The frontend is a React single-page application that runs entirely in the browser and communicates with cloud services over HTTPS. Authentication and user identity are handled by Firebase Authentication, which supports both email/password and Google sign-in. User profiles, roles, and audit logs are stored in Firebase Firestore, which acts as the primary data store. Access control is enforced using Firebase security rules and custom claims, ensuring that admin-only operations cannot be executed by regular users. TradingView widgets are integrated directly into the frontend to provide real-time crypto charts and indices, while React Router manages navigation between user and admin views. This architecture separates presentation, business logic, and data access, making the system easier to reason about, maintain, and extend in future iterations or deployments.

4.2 System data and database design

From an integration perspective, Cipher Chain connects three main layers: the browser-based client, Firebase services, and third-party visualization tools. All connectivity occurs via secure HTTPS calls using official SDKs or embedded scripts. The React application communicates with Firebase Authentication to handle login, logout, and token management. Once authenticated, the client accesses Firestore through Firebase's JavaScript APIs to read and write user, role, and audit log data. Security rules defined on Firestore ensure that only permitted operations are executed, even if a malicious user attempts to bypass the frontend. TradingView widgets are embedded within dashboard components using provided script tags and configuration objects, allowing charts to load market data directly from TradingView's infrastructure. Internally, the app uses React Router for navigation and context providers to expose authentication state across components. This design keeps the core logic in the client, while delegating identity, data storage, and chart data retrieval to managed cloud services.

4.3 System interface and connectivity design

4.3.1. System user interface design

The user interface of Cipher Chain clearly separates trader-facing features from administrative controls while keeping the experience simple and consistent. After logging in, regular users see a dashboard centered on a TradingView chart for a selected crypto pair, with summary cards for indices or key metrics and a small section for market

highlights or news. Navigation is minimal so users can focus on price movement and trends rather than complex menus. Administrators have extra navigation options leading to the user management console and audit log view. The admin console shows users in a table with columns such as email, role, status, and last login, plus action buttons to enable, disable, or delete accounts. Audit records appear in a sortable, filterable table. Tailwind CSS is used to maintain a clean layout, readable typography, and a responsive design on standard laptop and desktop screens.

4.3.2. System component API and logic design

Cipher Chain's internal logic is structured into modular components and service-style APIs to keep the system organized and easy to extend. An **AuthProvider** wraps the application and exposes the current user, authentication status, and role through React Context, so any component can access this information without prop drilling. Route protection is handled by a **ProtectedRoute** component that checks the user's role before rendering admin-only pages. All direct interaction with Firebase is encapsulated in service modules. For example, a **UserService** manages operations such as fetching users, updating statuses, and deleting accounts, while an **AuditService** provides simple functions to log events like logins, logouts, and admin actions. UI components call these higher-level functions instead of low-level SDK calls. This design follows separation of concerns and façade patterns, improving testability, readability, and future maintainability of the codebase.

4.4 System design problems, solutions, and patterns

Several design challenges arose during the development of Cipher Chain. One issue was ensuring that route protection and role checks were consistent across the entire app; this was solved by introducing a centralized **ProtectedRoute** component and an **AuthProvider** that exposes user roles globally. Another problem involved keeping the frontend state synchronized with Firebase custom claims after role changes. This was handled by forcing token refreshes and re-fetching user data when admin operations occur. To prevent tight coupling between React components and Firestore, we introduced service modules that act as façades, an application of the façade pattern.

Context and hooks were used to implement a simple form of dependency injection, where components depend on abstracted functions instead of low-level SDK calls. These solutions apply common design patterns—such as separation of concerns and single responsibility—which make the system easier to extend with additional roles, views, or data sources in the future.

5. System Implementation

5.1 System implementation summary

The implementation of Cipher Chain was carried out in iterative stages, starting from the core foundations and gradually adding enterprise features. We first set up the React project structure, routing, and base layout using Tailwind CSS. Next, Firebase Authentication was integrated to support email/password and Google sign-in, along with session handling on the client. Once users could log in, we added role-based access control using Firebase custom claims and implemented **ProtectedRoute** components to separate admin and regular views.

The admin dashboard was then developed to list users, show their status, and provide buttons for enable, disable, and delete operations. After that, we integrated TradingView widgets into the user dashboard to display live crypto charts and indices. Finally, we implemented audit logging using a dedicated Firestore collection and wired it to log key actions such as login, logout, and admin account changes, completing the end-to-end workflow.

5.2 System implementation issues and resolutions

During implementation, several issues appeared that required design and code-level adjustments. One of the first challenges was synchronizing Firebase custom claims with the frontend state after an admin changed a user's role or status. The solution was to trigger a token refresh and re-fetch user data whenever such operations were performed, ensuring that UI access rules updated correctly. Another issue involved handling asynchronous Firestore reads and avoiding flickering or incomplete data in tables. We solved this by introducing loading states and carefully using React hooks to manage subscriptions.

Embedding TradingView in a single-page application also caused re-render problems when navigating between routes. This was addressed by encapsulating the widget in its own component and cleaning up scripts on unmount. These resolutions improved stability and provided a clearer separation between view components, state management, and external integrations.

5.3 Used technologies and tools

Cipher Chain uses a modern web stack that balances rapid development with realistic enterprise patterns. The frontend is built with React, using React Router for client-side navigation and React Context for sharing authentication and role information. Tailwind

CSS is used to implement a consistent, responsive UI suitable for dashboards and admin tables. For the backend services, we rely on Firebase Authentication to handle user sign-in, sign-up, and Google OAuth, and Firestore as a NoSQL database for storing users and audit logs. Access control is enforced through Firebase security rules and custom claims. Financial charts and market views are provided by embedded TradingView widgets. Version control and collaboration are managed through Git and a GitHub repository. Development and debugging are supported by browser devtools, React Developer Tools, and the Firebase console, allowing us to monitor authentication events, database state, and security behavior throughout the project.

6. System Testing and Experiment

6.1 Testing and experiment scope

Testing for Cipher Chain focused on validating the most important user flows rather than full formal verification. The main scope included authentication, role-based access control, admin user management, and basic dashboard behavior. We verified that both regular users and admins could log in, navigate to their respective views, and perform key actions without errors. Additional checks covered how the system behaved when accounts were disabled and whether audit logs were created for critical events. Performance, load, and security penetration testing were not in scope for this course project, but the design leaves room for these in future work.

6.2 Testing and experiment approaches

Testing for Cipher Chain mainly followed a **scenario-based, manual approach**, focusing on realistic user journeys rather than formal test automation. We first identified the most critical flows from a security and usability perspective: logging in as a regular user, logging in as an admin, accessing dashboards, performing admin actions on accounts, and verifying audit logs. For each scenario, we wrote simple test descriptions and executed them step by step in the running application.

Two main perspectives were used: **regular user** and **admin**. For regular users, we tested successful login, viewing charts, and confirming they could not reach admin-only pages even by typing URLs directly. For admins, we tested viewing the user list, disabling/enabling accounts, and then logging in again as those users to verify the effect. We also intentionally tried invalid credentials and repeated logins to see how the system responded.

Basic **cross-browser checks** were performed using at least two browsers (for example, Chrome and another commonly used browser) to ensure layout, navigation, and TradingView widgets behaved consistently. Although automated unit or integration tests were not fully implemented due to time constraints, the manual approach still provided useful confidence that the core security-sensitive flows worked as designed.

6.3 Testing report

Although testing was limited, it confirmed that the core flows of Cipher Chain work as intended. Login and logout behaved correctly for both user roles, and admin-only pages were inaccessible to regular users. Disabling a user account successfully blocked

subsequent access, which matches the security goal of preventing reused credentials. Audit log entries were created for logins and admin actions, providing a minimal but functional activity trail. A few minor issues were found, such as missing loading states and unclear error messages, and were addressed during development. A summary of key manual test cases is shown in Table 3.

Test ID	Scenario	Steps	Expected Results	Status
T1	User login (regular)	Login with valid user credentials	Redirected to user dashboard; audit log entry created	Passed
T2	Admin login	Login with valid admin credentials	Redirected to admin dashboard; audit log entry created	Passed
T3	Disabled user access	Admin disables user; user attempts to log in again	Login blocked or access denied; no dashboard shown	Passed
T4	Access admin page as regular user	Regular user manually enters admin URL	Access denied and redirected to user dashboard/error page	Passed
T5	Admin action logging	Admin disables a user and opens audit log screen	Corresponding audit log entry visible with correct details	Passed

Table 3: Summary of manual test cases for Cipher Chain

7. Conclusion and Future Work

7.1 Project summary

Cipher Chain was conceived as a compact but realistic enterprise FinTech platform that combines secure authentication, role-based access control, admin governance, and crypto market analytics in a single web application. The project set out to bridge the gap between simple student dashboards and the kinds of systems actually used by exchanges, investment firms, and internal compliance teams. Over the course of the implementation, we designed a React-based single-page application, integrated Firebase Authentication and Firestore, embedded TradingView charts, and implemented an admin console for managing user accounts and reviewing basic audit logs.

Even though the system is a prototype, it demonstrates key enterprise concepts: separation of roles, secure multi-user access, and traceability of critical actions. Through this project we gained practical experience in architecting a cloud-backed web app, modeling requirements, designing data structures, and dealing with real integration issues such as token handling, asynchronous data, and third-party widgets. Cipher Chain ultimately validates that a small student team can design and build a focused platform that still reflects real-world enterprise concerns like security, governance, and observability.

7.2 Future work

There are several directions in which Cipher Chain could be extended to resemble a more complete production system. From a security perspective, future work could add multi-factor authentication, stronger session management, IP-based restrictions, and rate limiting for login attempts. Role-based access control could be expanded beyond a simple admin/user split to include roles such as “read-only analyst,” “operations,” or “auditor,” each with more granular permissions.

On the analytics side, the dashboard could be enhanced with portfolio tracking, watchlists, alerts, and support for additional asset classes or on-chain metrics. The audit logging system could evolve into a richer observability layer, including filters by severity, export to external log tools, and simple anomaly detection for suspicious activity. Another important step would be introducing automated testing—unit tests for services, integration tests for critical flows, and possibly end-to-end tests using a browser automation framework. Finally, deploying Cipher Chain with a CI/CD pipeline and environment configuration (dev, test, demo) would make it easier to maintain and demonstrate as a long-term project or foundation for future coursework.

References

[1] Google LLC, “Firebase Authentication Documentation,” Firebase, 2024.

[2] Google LLC, “Cloud Firestore Documentation,” Firebase, 2024.

[3] TradingView Inc., “Charting Library & Widget Documentation,” TradingView, 2024.

[4] OWASP Foundation, “OWASP Cheat Sheet Series: Authentication Cheat Sheet,” 2023.

[5] OWASP Foundation, “OWASP Cheat Sheet Series: Authorization and Access Control,” 2023.

[6] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley, 2003.

[7] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. Sebastopol, CA, USA: O’Reilly Media, 2021.

[8] M. Howard and D. LeBlanc, *Writing Secure Code*, 2nd ed. Redmond, WA, USA: Microsoft Press, 2002.

[9] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA, USA: Addison-Wesley, 1994.

[10] N. Subramanian and A. Jeyaraj, “Recent security challenges in cloud computing,” *Computers & Electrical Engineering*, vol. 71, pp. 28–42, 2018.

[11] S. N. Shiralkar et al., “A survey of financial dashboards and visual analytics,” *International Journal of Computer Applications*, vol. 175, no. 27, pp. 1–6, 2020.