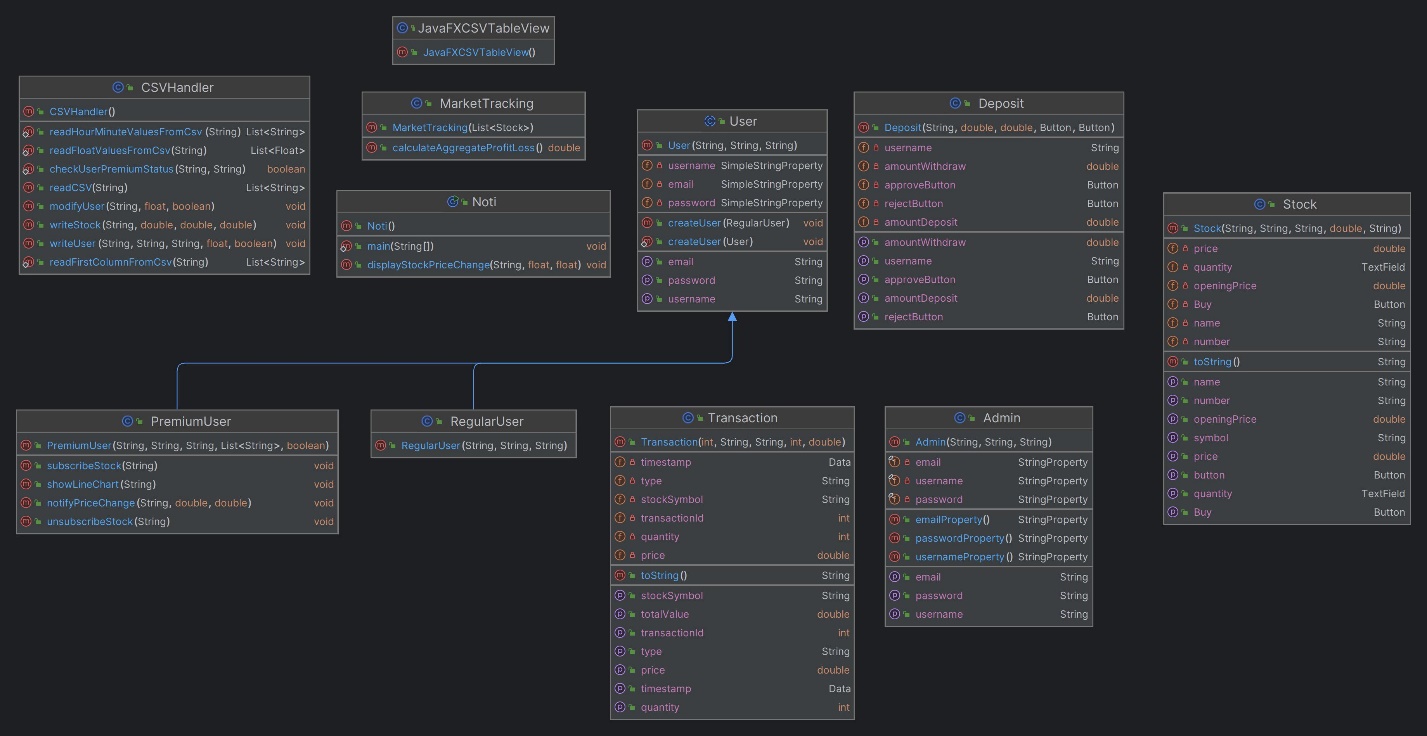
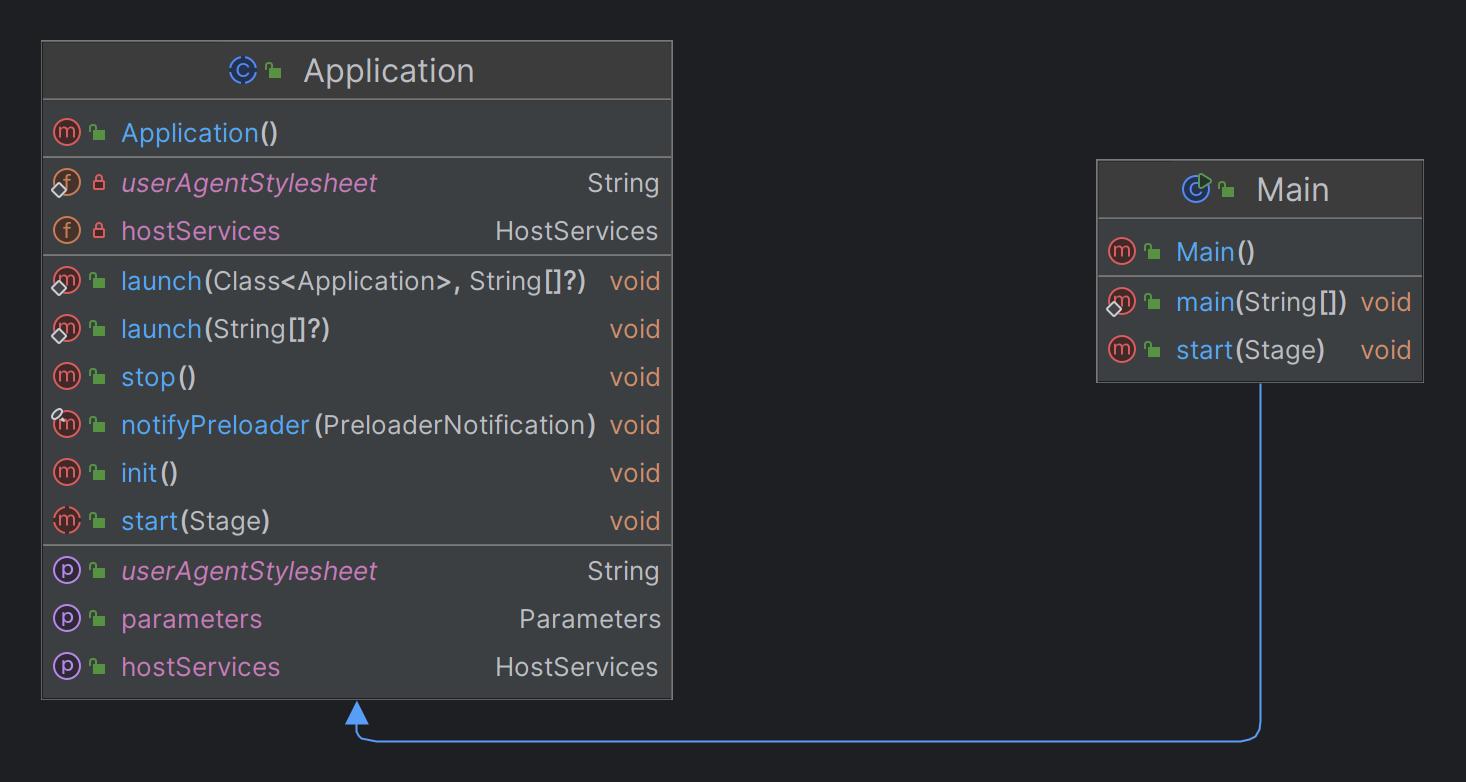
**Stock Exchange Manager App - Detailed Design Documentation**

|  |  |
| --- | --- |
| **9230146 SEC.1** | **احمد صلاح جيوشي عبدالفتاح الشناوي** |
| **SEC.2 9230759** | **محمد خالد عبدالحميد عبدالحميد مندور** |
| **SEC.2 9230563** | **عبدالله محمد خليفة منصور** |
| **SEC.1 9230235** | **الحسين ايمن حنفى محمد** |
| **SEC.1 9230395** | **زياد رمضان محمد محمد** |

**1. Introduction**

- The Stock Exchange Manager app is a Java-based application designed to facilitate stock trading, portfolio management, and market analysis in a simulated stock exchange environment. The project emphasizes the principles of Object-Oriented Programming (OOP) and aims to provide a user-friendly interface for traders.  
  
  
**2. Some UML-Diagrams  
  
  
  
  
  
  
  
  
  
3. System Architecture**

Overall Architecture:  
- The system adopts a modular and extensible architecture, encapsulating key components such as stocks, traders, transactions, and market data.  
- Components interact through well-defined interfaces, promoting scalability and maintainability.

Client-Side Architecture:- The client-side architecture utilizes JavaFX for the user interface (UI).- UI components include screens for admin features (single instance) and user features (multiple instances).

- Controllers handle user interactions and update the UI accordingly.

Server-Side Architecture:- While the current implementation is client-only, provisions are made  
for future server-side integration.- Technologies like Spring Boot and Hibernate could be employed for server-side functionality, including user authentication, data persistence, and business logic execution.

**4. Detailed Component Design**

- Components:

- Admin features include screens for user management, stock management, stock orders, price history, approval system, and trading session control.

- User features encompass account management, order management, deposit/withdrawal, financial actions, transaction history, price history, premium features, market performance tracking, charting options, and export functionality.

- Each component provides clear interfaces for performing relevant operations.

- Controllers:

- Controllers handle user inputs, validate data, and orchestrate interactions with service layers.

- Event handling ensures smooth user experience and data consistency.  
  
- Data Model:

- The data model consists of classes representing entities such as User, Stock, Order, Transaction, and Market Data.

- Relationships and attributes are defined to support various functionalities, including stock properties, price history, and user transactions.

- Business Logic:

- Business logic components handle stock trading, portfolio management, market analysis, and user authentication.

- Design patterns like Singleton, Observer, and Factory are employed to address common challenges and promote modularity.

**5. Integration Points**

- Integration points include communication between UI components, controllers, and service layers.

- RESTful API endpoints could be implemented for future integration with external data sources or backend systems.

- Data exchange between client and server follows JSON format over HTTP(S) protocol.

**6. Deployment Architecture**

- The app can be deployed on desktop platforms supporting Java runtime environment (JRE).

- Minimal hardware requirements ensure compatibility with standard desktop or laptop configurations.

- Database server (e.g., MySQL, PostgreSQL) may be required for server-side deployment.

- Continuous integration and deployment pipelines ensure seamless updates and maintenance.

**7. Non-functional Requirements**

- Performance requirements dictate response times for UI interactions and data retrieval.

- Scalability considerations include support for multiple users and concurrent transactions.

- Security measures encompass user authentication, authorization, and data encryption.

- Reliability and fault tolerance mechanisms include transaction rollback and error recovery procedures.

**8. Conclusion**

- The detailed design documentation outlines the architecture and design decisions of the Stock Exchange Manager app.

- Emphasis on OOP principles, design patterns, and best practices in software engineering ensures a robust and extensible system.

- Future enhancements may include server-side integration, advanced analytics features, and optimization for performance and scalability.

This should give a comprehensive overview of the design considerations and implementation details for the Stock Exchange Manager app based on the provided project objectives, guidelines, and features.