Full name: Mohamed Ibrahim Title of assignment: Sprint 1
Date of last update: 14/03/2025

Epic Equity Explorer

Overview:

Epic Equity explorer is designed to allow users to simulate and experience exponential investment returns through looking at old-historical data and this is over selected time periods. This will then allow users to pick one or more then one stocks from a active list that can be searched with a function in which the user can type the symbol or name that correlates to a specific stock and if not available the system can suggest similar names or stocks similar in terms of usage as a suggestion to if the user meant another stock instead of suggesting not found. Users will then be able to select a time span of up to two years for their selected stock to see their stock optimization and efficiency.

Architectural Concepts

Requirements needed for project

What will I implement for sprint 1?

Repository name: "Epic Equity Explorer

Main branch: a stable and deployable version of the application, all finalized features are merged after fully accurate testing

Development branch: used for testing the integration of new features, bug fixes and enhancements before merging into the main branch

Feature branch: created from the dev branch for developing certain features for example feature/feature-name

Bugfix branches: created when fixing bugs such as bugfix/bug-name

Release branches: temporary branches used to prepare for a new release named release/version-number

Core classes and their responsibilities

Application: configures and initializes all major components, including server setup and database connectivity

DatabaseConnector: manage database connections and transactions

User: represents the user entity with properties such as roles, password and username

UserRepository: defines operations related to user data management

UserServiceImpl: implements the business logic for the user management, utilizing the UserRepository

AuthenticationController: Handles user authentication requests

StockDataFetcher: Interfaces with financial data APIs to retrieve stock market data

Stock: Data model representing information surrounding stock data

DashboardController: Manages the display of user dashboard data

ConfigLoader: Manages configuration settings throughout the application

APIs and External Services

Financial Data Api: Select a suitable API(Yahoo Finance or Alpha Vantage) based on availability and features required. StockDataFetcher will use this API to fetch real time and historical stock data

Authentication Services: Plan for third party authentication integration

Database services: choose a database solution that fits project needs

Documentation and planning

Project documentation: include a readme.md in my repository detailing project setup, architecture and how to run the project locally

Code documentation: use inline comments to describe important logic and changes

API documentation: document API endpoints

Before Coding

Environment setup: ensure all developers have a consistent development environment, including necessary software, IDEs and API/database access

Database Schema Design: Plan and define database tables, relationships and indexes

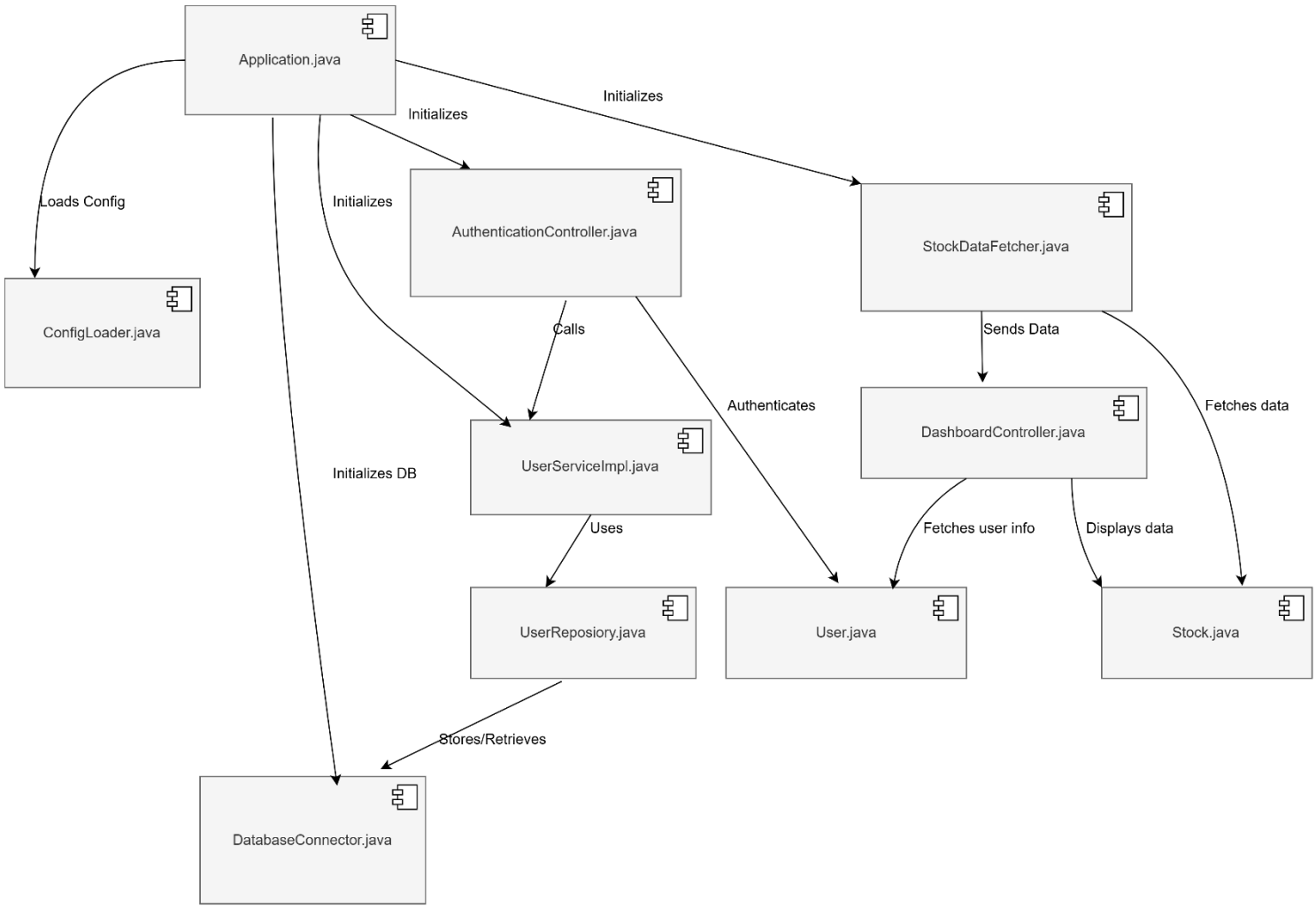
Mockups and UI design: create UI mockups for dashboards and other interfaces to guide frontend development

Security planning: develop a security strategy, including securing API endpoints, database protection and user authentication measures

Full name: Mohamed Ibrahim
Date of last update: 14/03/2025

Title of assignment: Sprint 1

Component specification diagram



Core Classes and Their Responsibilities

1. Application

- Initializes and configures all key components of the application, such as server setup and database connectivity.
- Serves as the entry point, ensuring the environment is properly set up for smooth operation.

2. DatabaseConnector

- Manages database connections and transactions, ensuring reliable and secure data access.
- Maintains transaction integrity to prevent data corruption or inconsistency.

3. User

- Represents a user entity with essential attributes like username, password, and roles.
- Acts as the primary data model for user-related information.

4. UserRepository

- Defines CRUD (Create, Read, Update, Delete) operations for managing user data.
- Facilitates interaction with the database to retrieve, modify, or remove user records.

5. UserServiceImpl

- Implements the business logic for user management.
- Uses UserRepository to process user-related operations according to business needs.

6. AuthenticationController

- Handles user login and authentication requests.
- Manages security checks to provide a safe and controlled access point for users.

7. StockDataFetcher

- Connects with financial data APIs to fetch real-time stock market data.
- Acts as a bridge between external stock data sources and the application.

8. Stock

- Represents the stock data model, storing key details such as prices, trading volume, and timestamps.

9. DashboardController

- Controls data presentation on the user dashboard.
- Oversees data retrieval and visualization to enhance the user experience.

10. ConfigLoader

- Manages configuration settings across the application.
- Loads and processes configuration files to make settings accessible to other components

Full name: Mohamed Ibrahim Title of assignment: Sprint 1
Date of last update: 14/03/2025

When I run the program, the console output occurs in this order because the Application.java main() method invokes three methods sequentially. DatabaseConnector.initializeConnection() is invoked first and prints 'Database connection initialized.' ConfigLoader.loadConfigurations() is invoked afterwards and prints 'Configurations loaded.' Finally, DashboardController.displayDashboard() is invoked, which prints 'Dashboard displayed.' The steps are performed sequentially by the program to simulate the process of setting up an application with database connectivity, system configurations, and a user interface.