# 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.

**Functional requirements**

Investment amount – users can be able to select the amount to invest and the system then regulates this amount into a numbered input

**Stock Selection**

Users can be able to select and search stocks using a auto feature

**Non-functional requirements**

Usability – the interface can provide clear feedback and will be user friendly

Performance – the system should be able to show the results and calculate it within a certain time

Security – API data transactions which are secure and inputs of data from user

Accessibility – Keyboard navigation and screen readers

Reliability – the system should calculate when an error has occurred for which tells the user

Scability – the application should be able to handle large amounts of requests from user and data which is stored in the database

**Technical Requirements**

Integration of API – Usage of financial data such as Yahoo Finance to see historical data

Handling of data -use classes for looking and storing data

**Risks and Mitigation strategies**

Api reliability: if the API does not calculate the correct data or shows any incomplete error handling

Data accuracy: Not being able to validate all data and showing not showing inaccuracies in order to fix the error so that leaders are not mislead.

**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

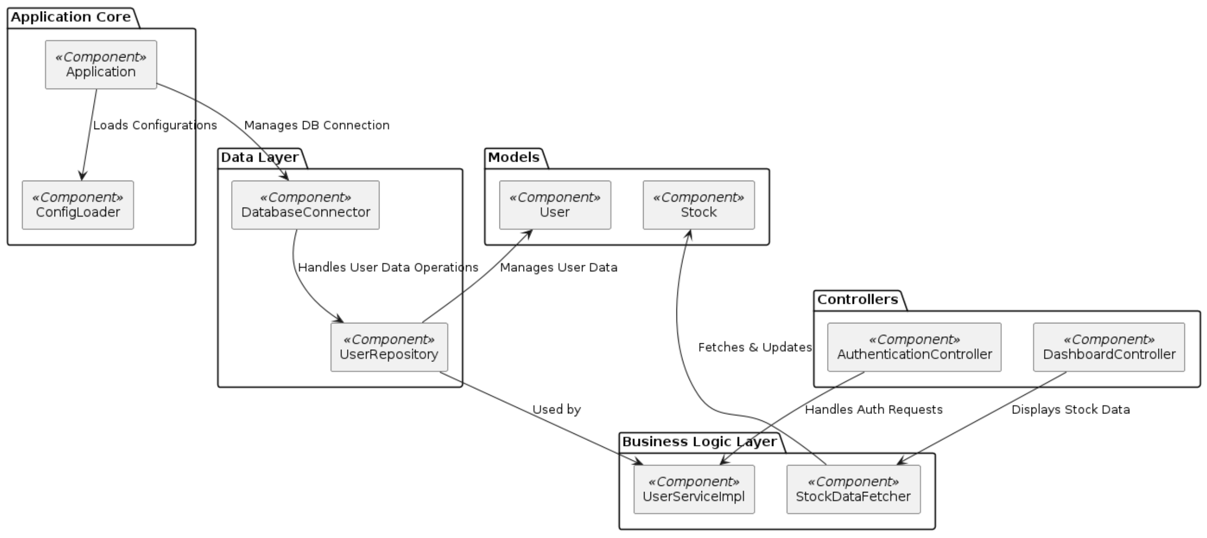
Environment setup: ensure all developers have a consistent development environment, including necessary software, IDEs and API/database access

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

## Component specification diagram



https://spring-boot-app-925354215034.us-east4.run.app/plantuml/150dd07c-31fd-4b5a-b242-281011b9093e.png