

# **Software Architecture & Design – Sprint 2**

Team Members:

Evan Balson – BAL18466416

Maahia Rahman – RAH23614335

Hala Bakhtiar – BAK23592238

Mahboubia Rezaei – REZ23579670

Roehampton University

Course Number: CMP020N207S

Lecturer: Dr Shekoufeh Kolaheidouh Rahimi

18/03/2025

Overview .....	3
Key Features:.....	3
Architecture & Design:.....	3
How is this achieved? .....	3
Business Concept Model .....	4
Use Case Model .....	4
System Interfaces .....	5
Business Type Model .....	7
Initial System Architecture .....	7
UML2 Composite Component .....	8
Clean Architecture Principles.....	8
Code-Related Deliverables .....	8
Implementation of Clean Architecture Principles.....	8
Single Responsibility Principle (SRP) .....	8
Open Close Principle (OCP) .....	8
Liskov Substitution Principle (LSP) .....	8
Interface Segregation Principle (ISP) .....	8
Dependency Inversion Principle (DIP) .....	8
Component and Interface Implementation .....	8
Team Management.....	9
Team Coordination .....	9
GitHub Updates .....	10
References: .....	11

## Overview

StockShark is a web application designed to provide users with quick and accessible stock data, powered by real time information fetched from the Yahoo Finance API. Whether you are seasoned investor or just starting to explore the stock market, StockShark offers a streamlined experience to analyze and compare stocks within a two-year timeframe.

### Key Features:

- Effortless stock search: easily find stocks by their company name or ticker symbol.
- Historical data comparison: visualize and compare the performance of multiple stocks over a customizable period, up to two years.
- Real-time data: leverages the Yahoo Finance API to provide up to date stock information.

### Architecture & Design:

StockShark employs clean architecture, a software design that emphasizes separation of concerns and independent from frameworks and external tools. [1]

Clean architecture is a software design pattern aims for:

- Framework independence: allows to easily switch to a different web framework or database without rewriting the heart of StockShark. [1]
- Easy testing: because the business logic is isolated, we can easily write tests to ensure that it works correctly. That leads to a more reliable application. [1]
- User interface independence: the user interface is separate from the core logic. This means we redesign the website without affecting how StockShark processes stock data. [1]
- External API independence: we can change the external API we are using to get data, without impacting the business rules. [1]

### How is this achieved?

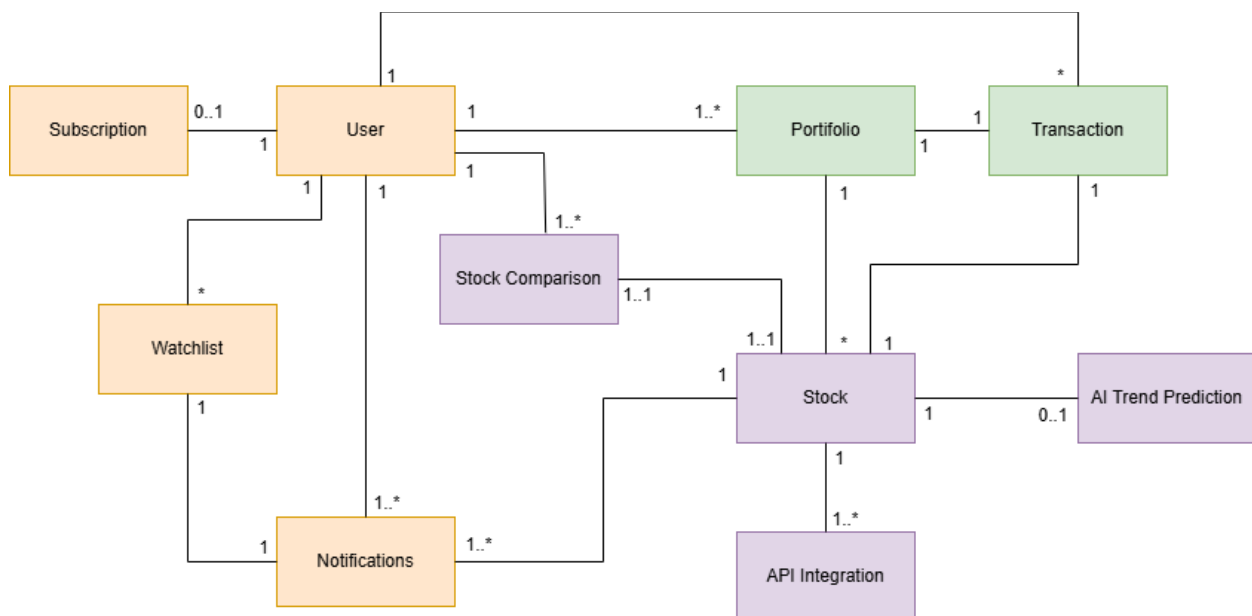
StockShark's architecture is built upon the core principles of Clean Architecture, which aims to create systems that are independent of frameworks, testable, and maintainable. This design follows the SOLID principles, making the system strong and easy to change.

- Single/Closed Principle (OCP): StockShark is designed to be open for extension but closed for modification. New features are added by introducing new code, rather than altering existing, stable code.

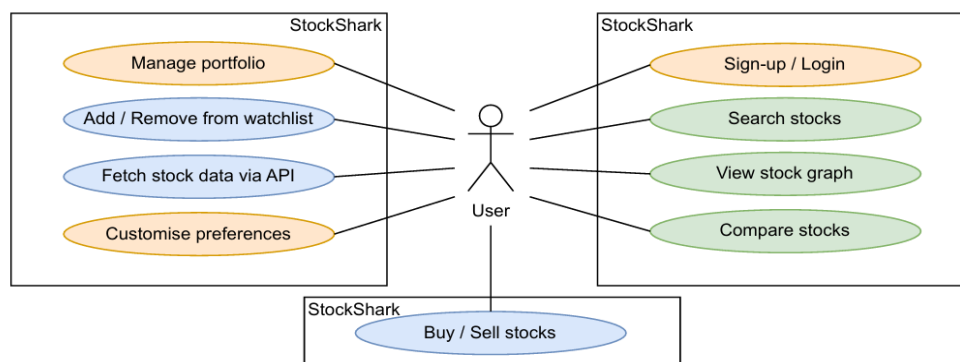
- Dependency Inversion Principle (DIP): High level modules, such as use cases that implement business logic, do not depend on low level modules, such as data access layers. Instead, low level modules depend on abstractions defined by high level modules.

By separating these layers, we create a system that's flexible, maintainable, and easy to understand. This makes it easier for our team to add new features, fix bugs, and adapt to changing requirements, ultimately providing you with a better StockShark experience.

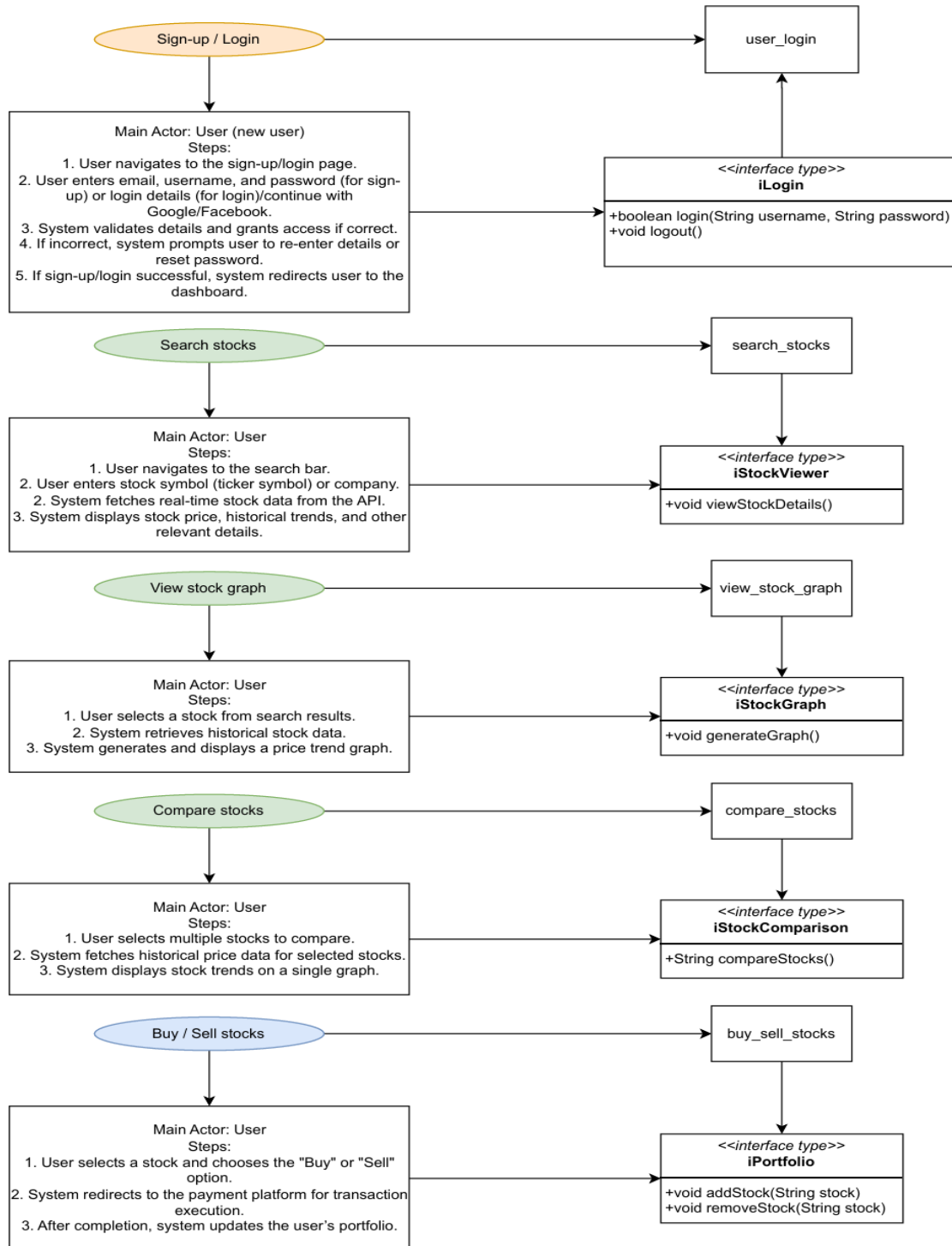
## Business Concept Model

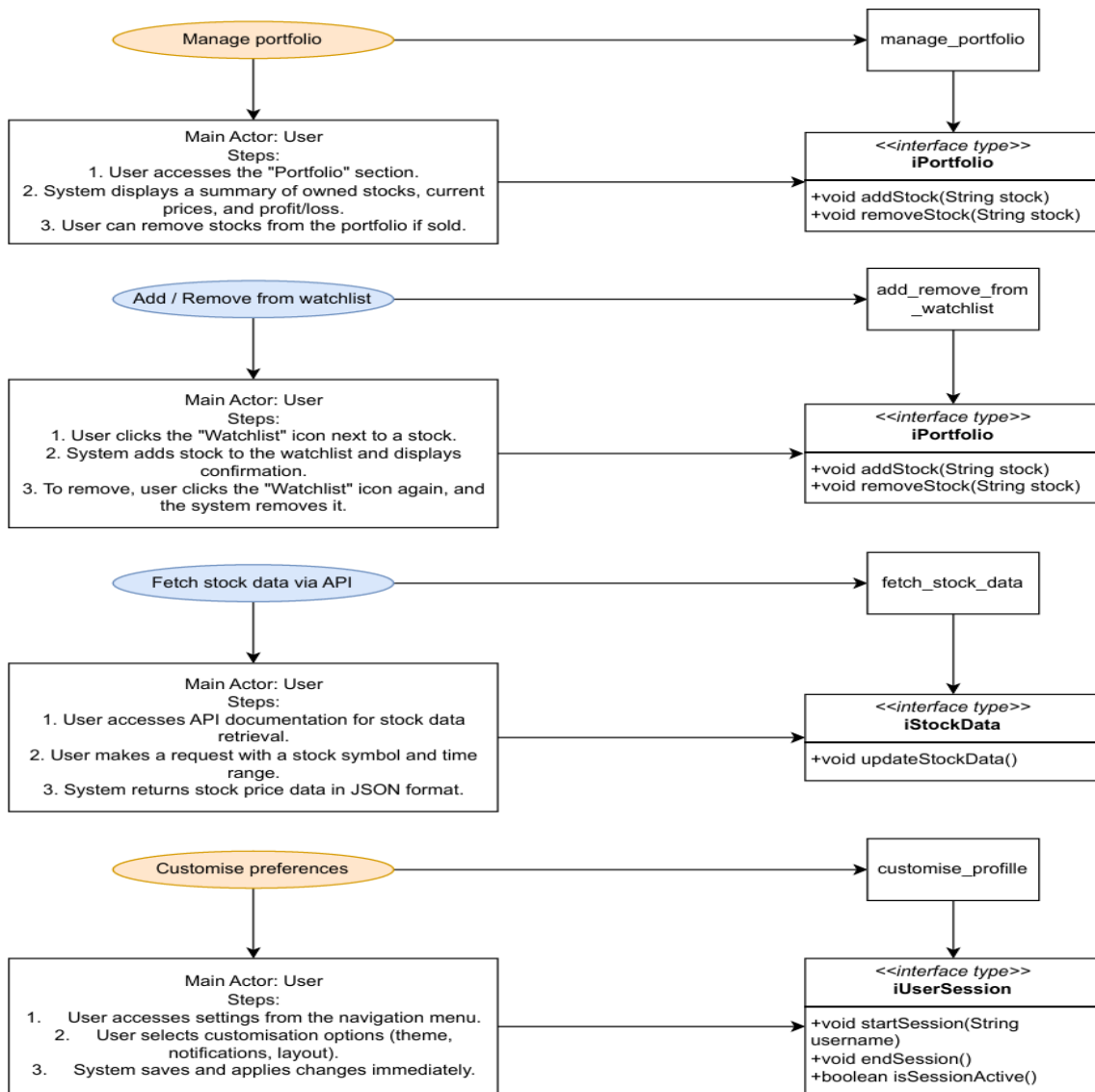


## Use Case Model

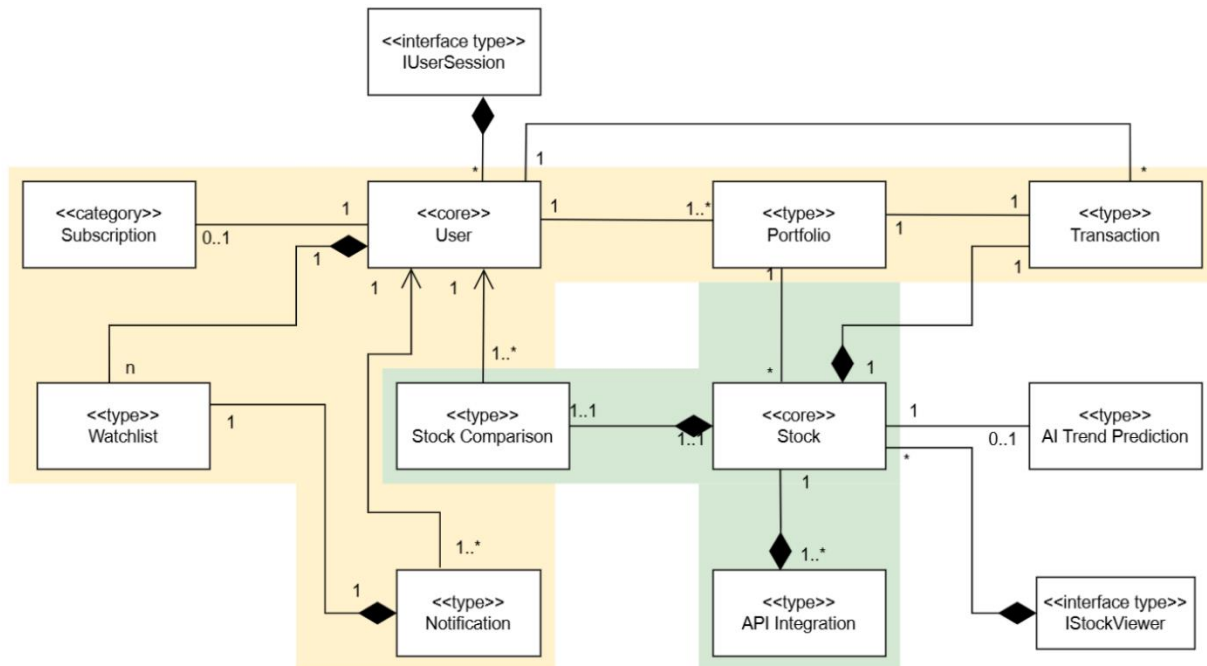


# System Interfaces

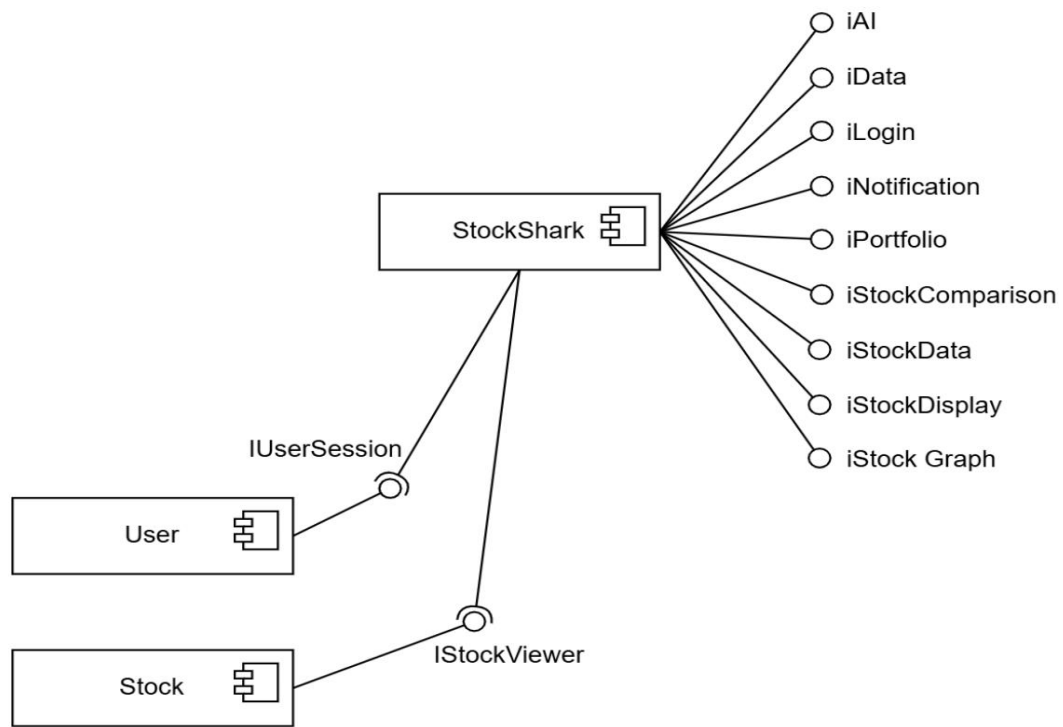




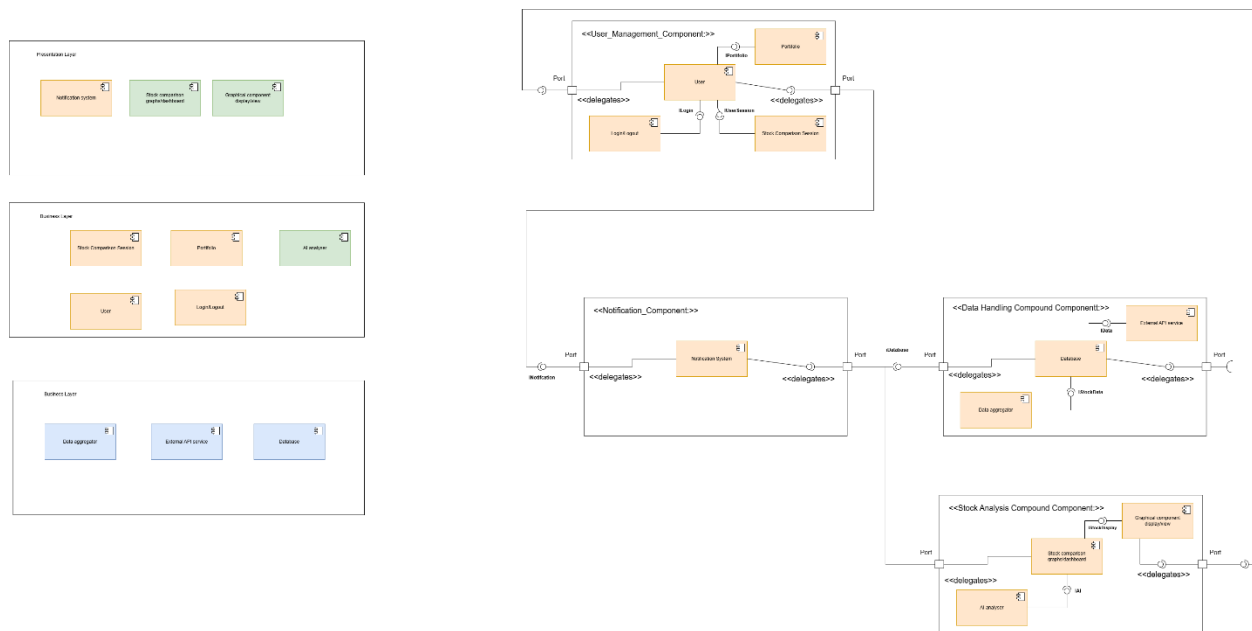
## Business Type Model



## Initial System Architecture



# UML2 Composite Component



## Clean Architecture Principles

How are they being followed?

Provide clear documentation on how components and interfaces are implemented according to these principles

## Code-Related Deliverables

### Implementation of Clean Architecture Principles

Single Responsibility Principle (SRP) - each class has one responsibility.

Open Close Principle (OCP) - you can extend but not modify.

Liskov Substitution Principle (LSP) - subtypes are compatible with base class.

Interface Segregation Principle (ISP) - avoiding implementing unnecessary methods.

Dependency Inversion Principle (DIP) - have abstract methods instead of concrete coding implementations.

## Component and Interface Implementation



# Team Management

## Team Coordination:

Team Leader: Evan Balson – provided direction of Sprint 2 tasks, actively tracked progress and ensured completion. In charge of System Interfaces, composite component diagram and high-fidelity wire frames. Coding implementation: Database, AI analyzer and API Service.

Scrum Master: Maahia Rahman – scheduled weekly team meetings. In charge of Business Concept Model and low fidelity wireframes. Coding implementation: Graphical component and stock comparison graph.

Mahboubia: In charge of Business Type Model. Coding implementation: Notification system and stock comparison session.

Hala: In charge of Use Case, Initial Architecture and Business Interfaces. Coding implementation: Portfolio, Login/Logout and User.

Tasks were split equally amongst the group based off strengths and preference. To ensure the main branch was not disrupted, we used pull requests for double checking our implementations.

# GitHub Updates:

Evan-Balson / Projects / Software Design & Architecture Project

Q Type [Z] to search

+

Software Design & Architecture Project

Increased items preview Feedback

Add status update

Tasks

Kanban Board

My items

Current iteration

Next iteration

Prioritized backlog

In review

Repository

New view

Filter by keyword or by field

Discard

Backlog 2

This item hasn't been started

SDA-Spring-Project #23

AI Analyser

SDA-Spring-Project #39

System Interfaces

+ Add item

Ready 1

This is ready to be picked up

SDA-Spring-Project #24

API Service

+ Add item

In progress 4

This is actively being worked on

SDA-Spring-Project #38

Business Concept Model

SDA-Spring-Project #22

Database

SDA-Spring-Project #31

Stock Comparison Graphs

SDA-Spring-Project #30

Graphical Component

+ Add item

In review 5

This item is in review

SDA-Spring-Project #17

Abstract implementation of Architectural components

SDA-Spring-Project #12

Set Up API For Yahoo Finance and Google Finance

SDA-Spring-Project #13

Set up Relational Database for the App

SDA-Spring-Project #25

Notification System

SDA-Spring-Project #37

WireFrames

+ Add item

In progress 4

This is actively being worked on

SDA-Spring-Project #38

Business Concept Model

SDA-Spring-Project #22

Database

SDA-Spring-Project #31

Stock Comparison Graphs

SDA-Spring-Project #30

Graphical Component

+ Add item

In review 5

This item is in review

SDA-Spring-Project #17

Abstract implementation of Architectural components

SDA-Spring-Project #12

Set Up API For Yahoo Finance and Google Finance

SDA-Spring-Project #13

Set up Relational Database for the App

SDA-Spring-Project #25

Notification System

SDA-Spring-Project #37

WireFrames

+ Add item

Done (SPRINT1) 16

This has been completed

SDA-Spring-Project #8

Develop Use Case Diagram

Feb 16, 2025 Feb 5, 2025

SDA-Spring-Project #27

Portfolio

SDA-Spring-Project #18

Personas for the system

SDA-Spring-Project #10

Set Up GitHub Repository & Project Management

Jan 29, 2025 Jan 27, 2025

+ Add item

Done (SPRINT2) 5

This has been completed

SDA-Spring-Project #42

Initial Architecture/Business Interfaces

SDA-Spring-Project #28

Login/LogOut

SDA-Spring-Project #29

User

SDA-Spring-Project #40

Business Type Model

SDA-Spring-Project #37

WireFrames

+ Add item

## References:

[1]. GeeksforGeeks, “*Complete Guide to Clean Architecture*”, [Online], Available: [Complete Guide to Clean Architecture - GeeksforGeeks](#), [Accessed: March. 16, 2025]