Software Architecture & Design

Sprint 1 Report

By Group 1

Contents

1. Group Members .....
2. Code of Conduct .....
3. Introduction & Project Details .....
4. Component Specification Diagram .....
5. Meeting Records .....
6. GitHub Links .....

Group Members

Amel Soumia Bouras, A00018158 *Project Owner*

Lyba Ahmad, A00012095 *Scrum Master*

Jochebed Nsiah Saah, NSI22527318 *Team Member*

Rania Malik, A00011818 *Team Member*

Code of Conduct

This code of conduct outlines the basic rules and requirements we will follow during this project to ensure the project runs smoothly and everyone is involved.

Teamwork and Respect

All team members are expected to collaborate respectfully and professionally throughout the project. Everyone should have an equal opportunity to share ideas and opinions. Harassment or disrespect of any kind will not be tolerated. If you experience or witness inappropriate behaviour, speak up immediately. Workloads must be distributed fairly among all members.

Accountability

Each team member is responsible for completing their assigned tasks on time and contributing to the overall success of the group. If you are struggling or falling behind, inform the team as early as possible so support can be provided.

Communication

All project-related communication will take place in the WhatsApp group chat. Members are expected to stay up to date with messages and respond in a timely manner. In addition, the team will hold weekly meetings to review progress, discuss upcoming tasks, and ensure everyone is aligned. These meetings will be kept brief and focused.

Conflict Resolution

Any conflicts should be addressed respectfully between the individuals involved. If the issue cannot be resolved, the team will step in to assist and ensure a fair outcome.

Use of Artificial Intelligence

AI tools must be used in accordance with university guidelines. The team may use AI to support learning and understanding, but members must not rely on it without fully understanding the context or content produced.

Consequences for Non-Performance

If a team member fails to complete their responsibilities, the following staged consequences will apply:

- Stage 1- Verbal Warning
  Issued if no progress is visible on GitHub within the first third of the allocated task timeframe.
- Stage 2- Task Reassignment
  If no progress or minimal progress (10% or less) is shown on GitHub, the task will be reassigned to another team member. The original assignee must buy a drink for the person taking over the task.
- Stage 3- Escalation
  Continued lack of contribution will result in the issue being reported to the lecturer.

Introduction & Project Details

This project includes the design and implementation of a 'Share Price Comparison Webpage', where users can select up to two companies to view and compare share prices over a period of time, ranging up to two years. The webpage also supports offline functionality of the graph as stock data is persisted on the system.

The following points go into more detail about the functionality of the webpage:

- The system fetches daily stock data for a company, from an API, upon a user's request. The data is then saved locally in a database so that the user can access it offline.
- Users select a start and end date, limited to a range of the current date up to two years prior.
- The selected stocks appear on a graph where the y-axis contains price and the x-axis contains days in the specified time range.
- Stocks are labelled with their company's name on the graph, and have distinct colours to distinguish between them.

Functional Requirements:

1. The system shall allow the user to select a valid company by searching through a list of available companies or typing its name/share symbol.
2. The system shall allow the user to add up to two companies to the graph.
3. The system shall allow the user to select two different dates.
4. The system shall ensure that the selectable dates are limited to the current date and dates up to two years before.
5. The system shall check the database if share data is available for the user's request before retrieving data from the API.
6. The system shall request daily share data from the API if not found locally on the database.
7. The system shall persist any new requested data into the database.
8. The system shall display a line graph of the selected company/companies, showing price over time (days).
9. The system shall use distinct colours and labels to differentiate the companies on the graph.

Non-Functional Requirements:

1. Usability- the system shall provide a clean UI that is easy to navigate.
2. Robustness- the system shall validate all inputs and handle errors correctly to avoid crashing.
3. Maintainability- the system shall follow a layered architecture that is well designed and justified.
4. Modularity- the system shall be designed to allow components to be replaceable.

Technical Decisions

The API of choice for this project is *Financial Modeling Prep* as it is reliable, has many free features, provides both real-time and historical data for a number of major companies, and can be easily integrated into a database. Overall, it fits our project the best compared to other APIs, although limited, aligning with the requirements.

The use of a relational database rather than persisting JSON/CSV responses in a file ensures data integrity and ease of data filtering through structured querying in the code. For a small-scale local project, SQLite is more appropriate than a server database like MariaDB.

Component Specification Diagram

Synopsis of Software Architecture

Simple Architecture is a software design philosophy that prioritises clarity, maintainability, and separation of responsibilities over complexity. It also provides a clean and organised foundation that meets current system requirements while remaining adaptable to future changes. Thus, it aligns with Clean Architecture principles, reducing coupling and keeping business logic independent of external frameworks and infrastructure. In this project, Simple Architecture supports the core functionality of retrieving, storing, and comparing share price data without over engineering. A simple layered structure is sufficient, enabling focus on correctness, clarity, and maintainability.

Core Architectural Principles

Separation of Concerns – Divides the system into Presentation, Service, Data, and Domain layers to limit the impact of change.

Single Responsibility Principle – Each component has one clear purpose: Share Service for business logic, Share Repository for data, and Api Client for external communication.

Layered Architecture – Dependencies flow from Presentation to Service to Data, with higher-level components relying on abstractions, improving modularity and testability.

Low Coupling and Abstraction – Components interact via interfaces, allowing changes, such as switching storage methods, without affecting other layers.

Why is suitable for this project?

Simple Architecture fits the moderate scope of this project. The layered design supports collaboration, iterative development, and future enhancement, maintaining readability, testability, and extensibility without major restructuring.

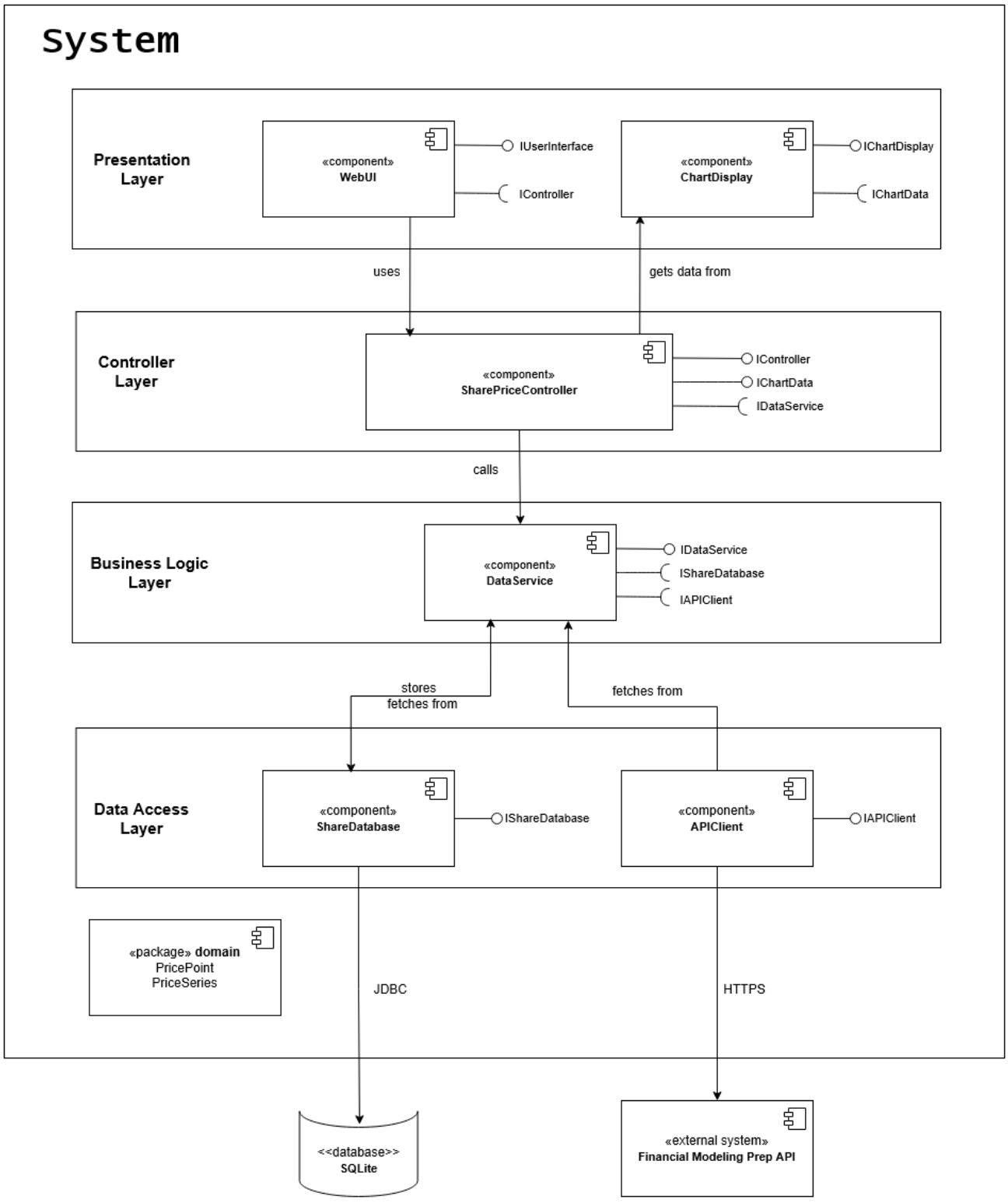
References

[1] Simple Architecture "About," *Simple Architecture*, Bangkok & London. [Online]. Available: https://www.simplearchitecture.net/about (accessed: 15-Feb-2026).

[2] Taylor & Francis Knowledge Centre, "Separation of concerns," *Taylor & Francis*, 2025. [Online]. Available: https://taylorandfrancis.com/knowledge/Engineering_and_technology/Computer_science/Separation_of_concerns/ (accessed: 15-Feb-2026).

[3] The Knowledge Academy, "System Architecture: Components and Types Explained," *The Knowledge Academy Blog*, 12-Feb-2026. [Online]. Available: https://www.theknowledgeacademy.com/blog/system-architecture/ (accessed: 15-Feb-2026)

Component Specification Diagram



Meeting Records

Date and Time	15-02-2026 12:00 – 12:55
Meeting Goal	• Agree on Code of Conduct • Assign roles in the team • Discuss and distribute tasks amongst members
Facilitator	Lyba
Note taker	Rania
Attendees	Amel, Lyba, Rania, Jochebed
Roundtable Updates	First team meeting so no new updates
Discussion points	Code of conduct Component specification diagram Researching architecture aspects Deciding SCRUM Master and Product Owner Product backlog and requirements
Actions	Code of Conduct – *Rania* Component diagram – *Lyba* Product backlog and requirements – *Amel* Research architectural designs – *Jochebed*

Date and Time	17-02-2026 21:15 – 22:45
Meeting Goal	• Verify and finalise Component Specification Diagram • Distribute components amongst members to code
Facilitator	Jochebed
Note taker	Amel
Attendees	Amel, Lyba, Rania, Jochebed
Roundtable Updates	Amel completed the requirements, Jochebed submitted initial research which Lyba used to draft a Component Specification diagram. Rania completed the Code of Conduct.
Discussion points	• Components and structure of the Component Specification Diagram • Initial talks about how to approach code implementation
Actions	Code Business Logic Layer component – *Rania* Code Presentation Layer components – *Lyba* Code Data Access Layer components – *Amel* Code Controller Layer component – *Jochebed*

GitHub Links

Repository:

https://github.com/Joch1n/share-price-analyser

Kanban Board:

https://github.com/users/Joch1n/projects/1