Software architecture and design

Sebastian

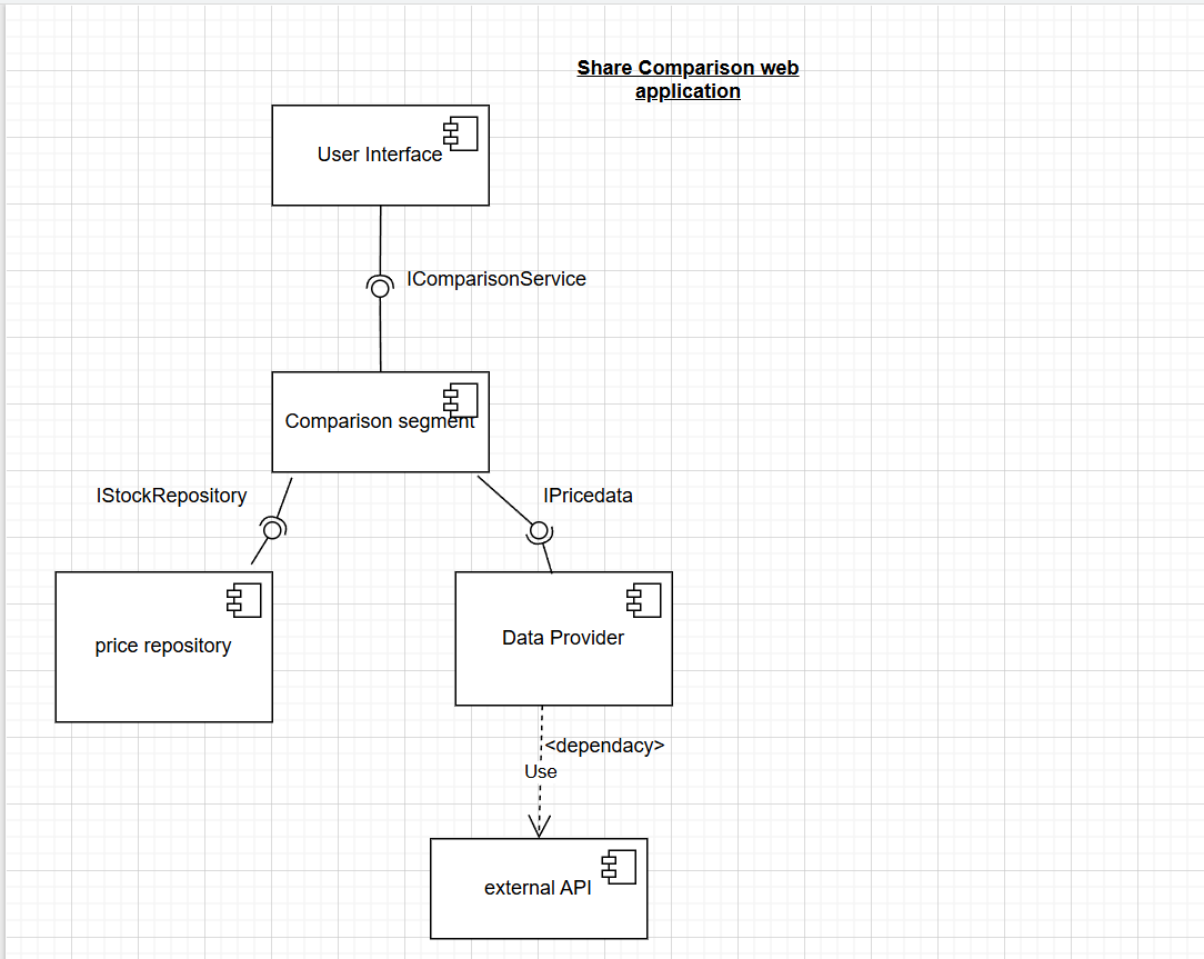
This document presents:

Architectural Pattern

The system is separated into different layers to help separate potential concerns that may occur in the system doing so promotes maintainability and modularity each layer is in charge of different responsibilities which are described as:

Layer	Components Contained	Primary Responsibility	Why It Is Useful
Controller Layer (Presentation Layer)	UI Component / SharePrice Controller	Handles user input, HTTP requests, and displays results	Separates user interface logic from business logic; prevents UI changes from affecting core system logic
Service Layer (Application Layer)	Comparison Engine	Contains core business logic; orchestrates data retrieval, validation, and comparison operations	Centralises business rules; improves maintainability and testability; supports Single Responsibility Principle
Data Provider Layer	Data Provider	Handles communication with external financial API	Encapsulates third-party dependencies; isolates API changes from the rest of the system
Persistence Layer (storage)	Price Repository	Manages local storage and retrieval of share price data	Enables offline access; reduces repeated API calls; improves performance and reliability
External System outside of the system	Yahoo Finance API (External API)	Provides external financial market data	Clearly defines system boundary; models real-world dependency

Component Diagram



How it works:

The UI sends user request to the comparison engine using the IComparisonService. The comparison service oversees the logic of the system by retrieving share price data either from the price repository through its interface (locally stored date) or from the data provide through its interface which is external data the data provider encapsulates communication with the external financial API, isolating third-party dependencies from the core logic