Software architecture and design

Sebastian

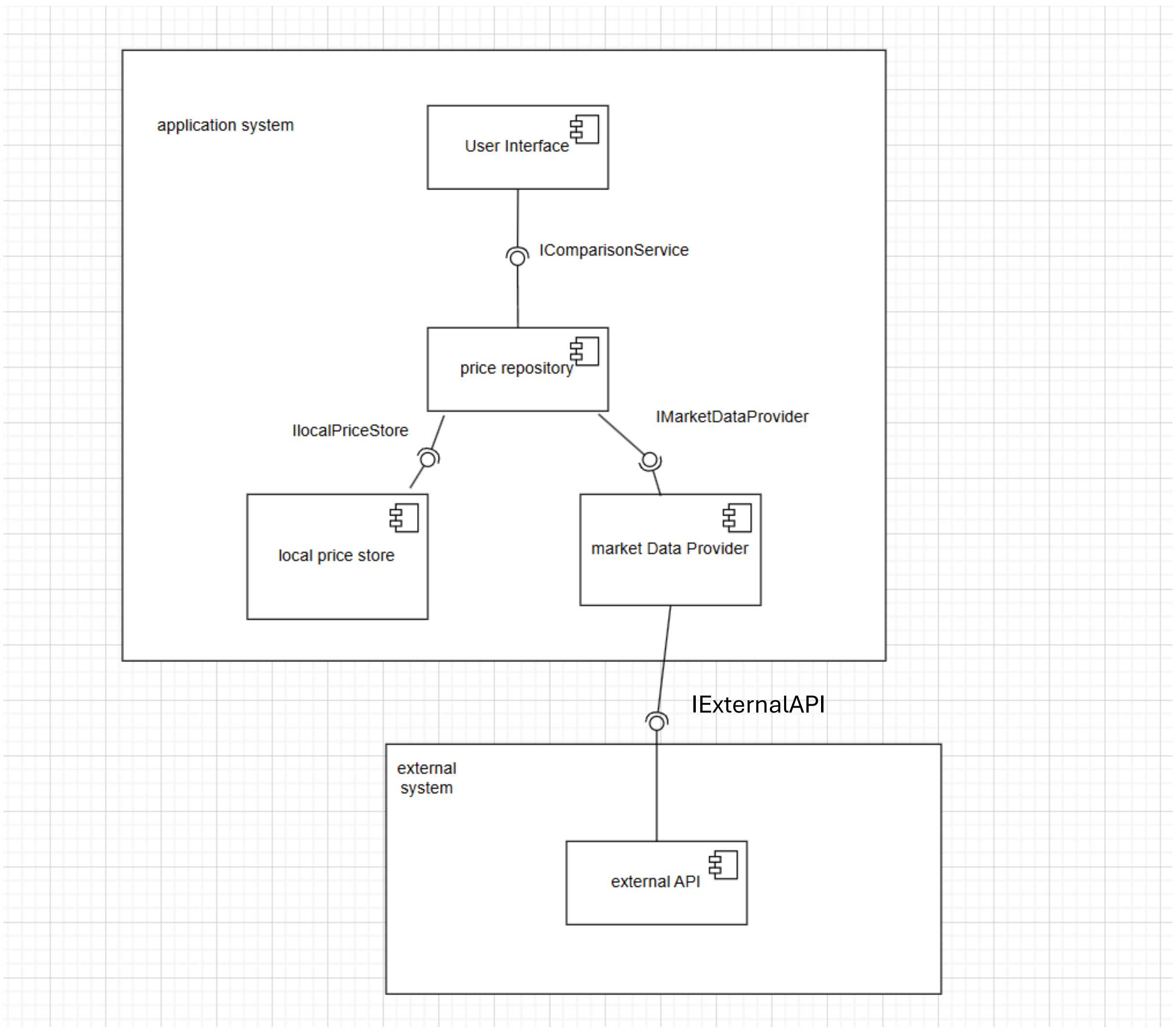
This document presents:

Architectural Pattern

The system has different components to help separate potential concerns that may occur in the system doing so, it promotes maintainability and modularity as each component handles different responsibilities which are described as:

Components	**Responsibility**	**Why It Is Useful**	**How it supports the web application**
User Interface	Handles user interaction and sends requests to the comparison service	Separates UI from business logic; allows UI changes without affecting core functionality	Allows users to select companies and date ranges and display comparison results
Comparison Service	Contains core business logic and orchestrates data retrieval and comparison operations	Centralises system logic; improves maintainability and testability	Performs share price comparison and coordinates repository interactions
Price Repository	Manages data retrieval strategy (local first, external if missing)	Implements caching policy; reduces API calls; supports offline capability	Ensures required share price data is available before comparison
Local Price Store	Stores and retrieves historical share price data locally	Improves performance; enables offline access; persists data between sessions	Allows previously fetched data to be reused without calling external API
Market Data Provider	Fetches data from the external share price service using IExternalAPI	Encapsulates third-party dependency; isolates API logic from core system	Retrieves historical share price data between selected dates
External API	Provides financial market data (outside the system as its not controlled by us)	Clearly models system boundary; represents real-world dependency	Supplies raw share price data required by the system

Component Diagram



Design Intention:

The intention of the sign is so that each component has a single responsibility and that higher level parts such as the UI and service do not depend on low level details like database or API. Furthermore, external feature such as API or storage technology only affects one component and not the whole system.

How it works:

The web application operates through a layered architecture in which the User Interface sends comparison requests to the Comparison Service via the IComparisonService interface. The Comparison Service delegates data retrieval to the Price Repository, which checks the Local Price Store for existing data. If required data is missing, the repository requests it from the Market Data Provider through the IMarketDataProvider interface. The Market Data Provider then communicates with the External API to retrieve historical share price data. Retrieved data is stored locally for future use, and the Comparison Service processes the data before returning results to the User Interface for display.