Team 12
Prescriptive Architecture for CryptoBoard

Functional Requirements- a subset of capabilities

Fetching News and Social Media Articles: The system should regularly fetch the required news and social media articles. This can be achieved by **scraping** the required news and social media websites at a fixed interval (24 hrs).

Article Processing and Analysis: The input articles must be processed and analysed for insights, which should be stored in a database and displayed to the user. The following types of analysis must be supported:
(a) Cross-Platform Comparison: Compare two or more data sources to identify which platform is more active regarding trending or specific topics.
(b) Keyword Frequency Analysis: Track the popularity of certain assets or concepts based on the amount of mentions in articles.
(c)Future Price Prediction: Analyze the historical data related to cryptocurrencies and provide to users for price predictions.

Data Storage: Data fetched from the data sources (articles) and the processed data must be stored to provide it to the front end.

Plug and Play Decoupling: The system should allow for the decoupling of different sources and processing. It should be able to change data sources (social media pages, news articles) without affecting the downstream processing pipeline.

Public API: Provide a public API for 3rd party users to access the collected data for further analysis.

UI Dashboard: A website with a dashboard should be created, and accessible to clients via a web browser.

Search Functionality: Users should be able to search for news and insights related to particular cryptocurrencies.

Up-to-date Cryptocurrency Data: The system should stay updated with all new cryptocurrencies by retrieving data from external APIs (such as CoinMarketCap) or Wikipedia to find all available cryptocurrencies.

User Management: The system should allow user login and have the ability to store and manage user details along with user authentication. It should also allow for customization of dashboards and personalized results.

Non-Functional Requirements

Scalability: The system should be scalable to handle an increase in the number of articles from data sources as well as the rise in the number of users.

Availability: The API and the UI must have an availability of 99.99%.

Accuracy: The scraping and processing should produce insights of high quality and correctness.

Compliance: The system should accommodate WCAG and ADA guidelines to ensure accessibility for users with disabilities.

Privacy: User data must be handled in accordance with clear policies that comply with GDPR or other applicable privacy laws. Additionally, the scraping must respect the restrictions of the data source websites.

Security: User data must be kept secure and encrypted both in transit and during storage. The security mechanism used must be robust.

Architecture

- The updated architecture follows a microservices architecture. The frameworks chosen are maven, express (node.js) for the backend, React for UI

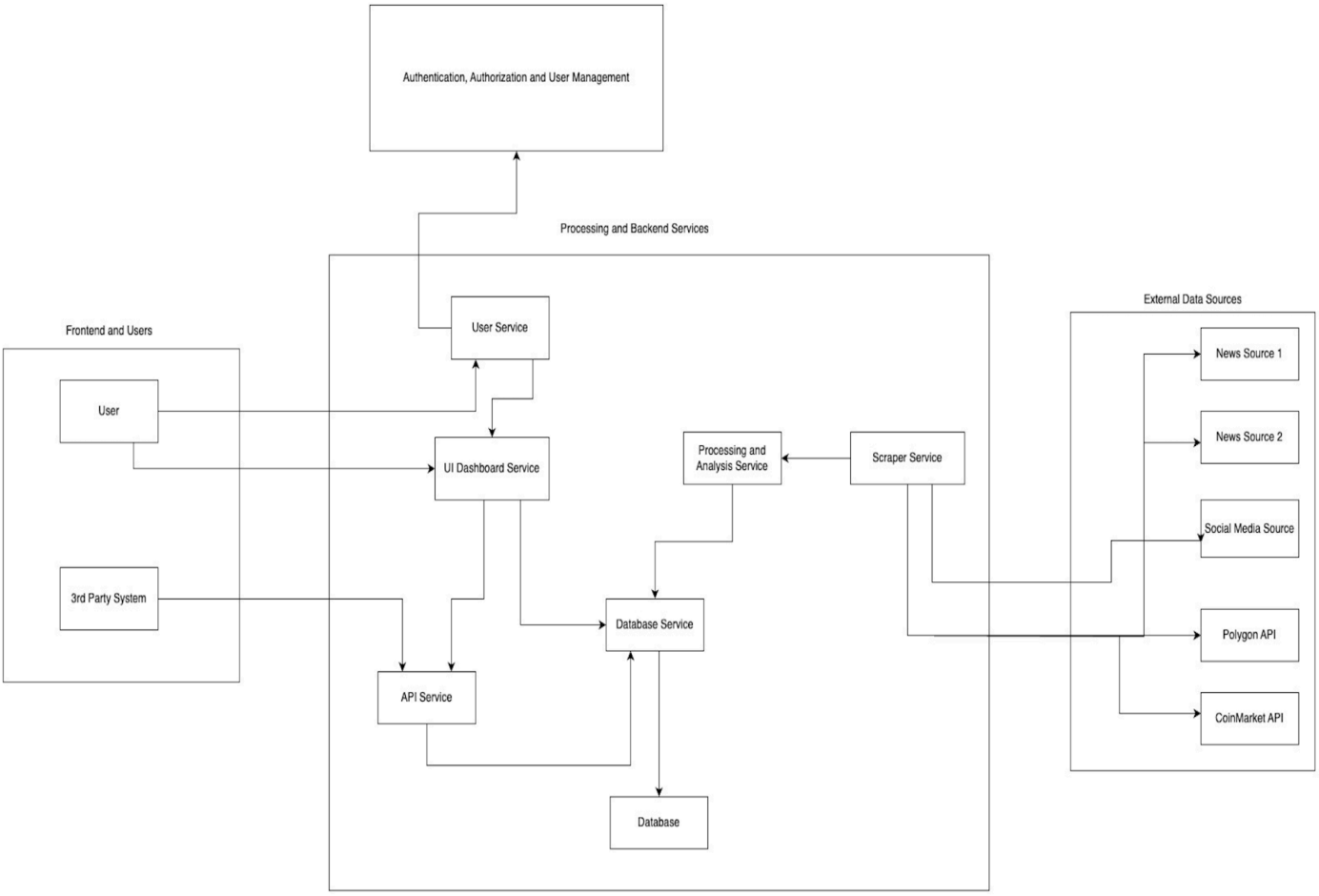
- Team members had language preferences, so microservices architecture aids in realising these requirements

- Microservices offer easier scalability in case of increased load. It would also enable teams or individuals (in our case) to work independently on different components without facing delays due to conflicts in deployment or development.

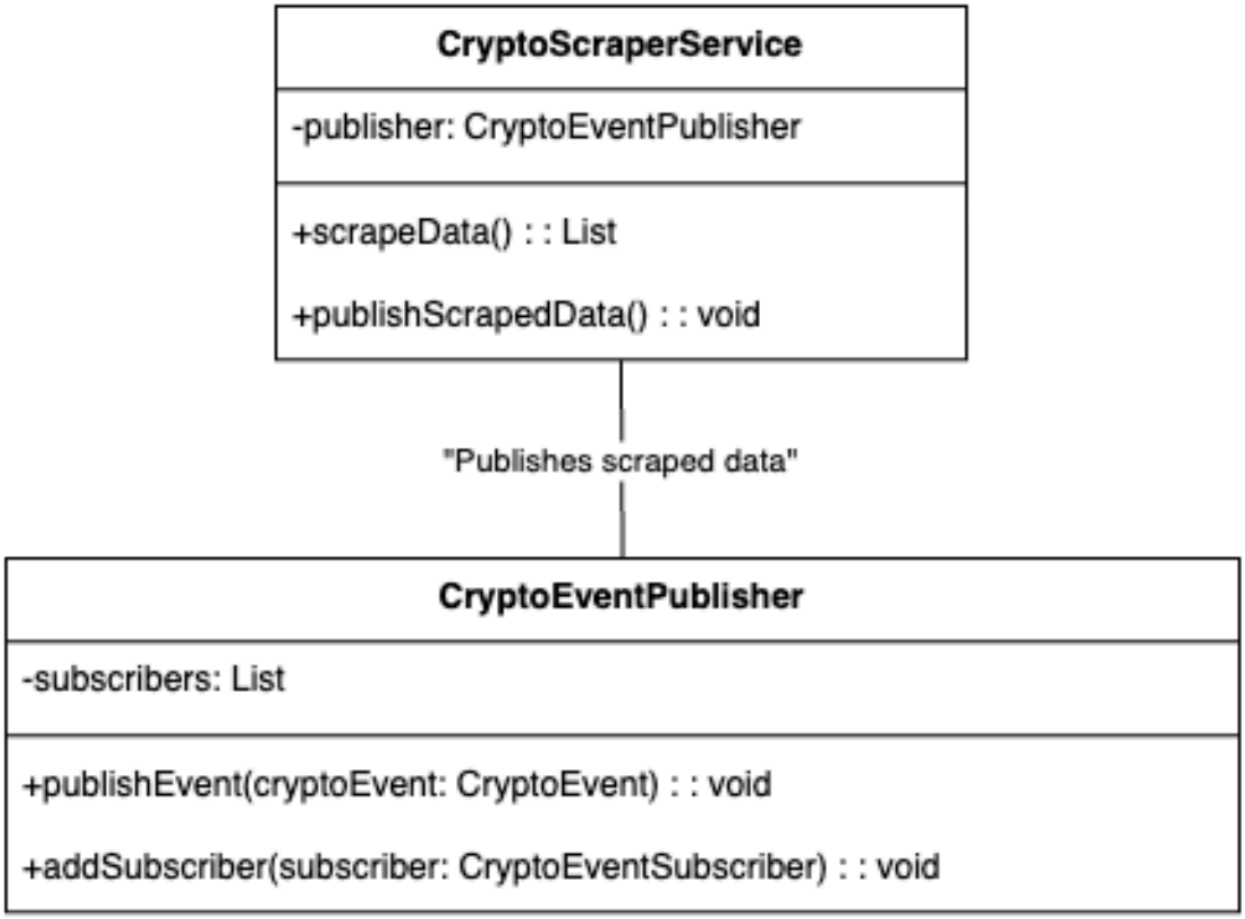
- The Assumption is that it is a service mesh where inter-service communication is taken care of by a platform such as Docker or orchestration platforms like Kubernetes.

- The architecture is a combination of event-based and client-server to handle API calls and asynchronous events such as crypto data updates and alerts.

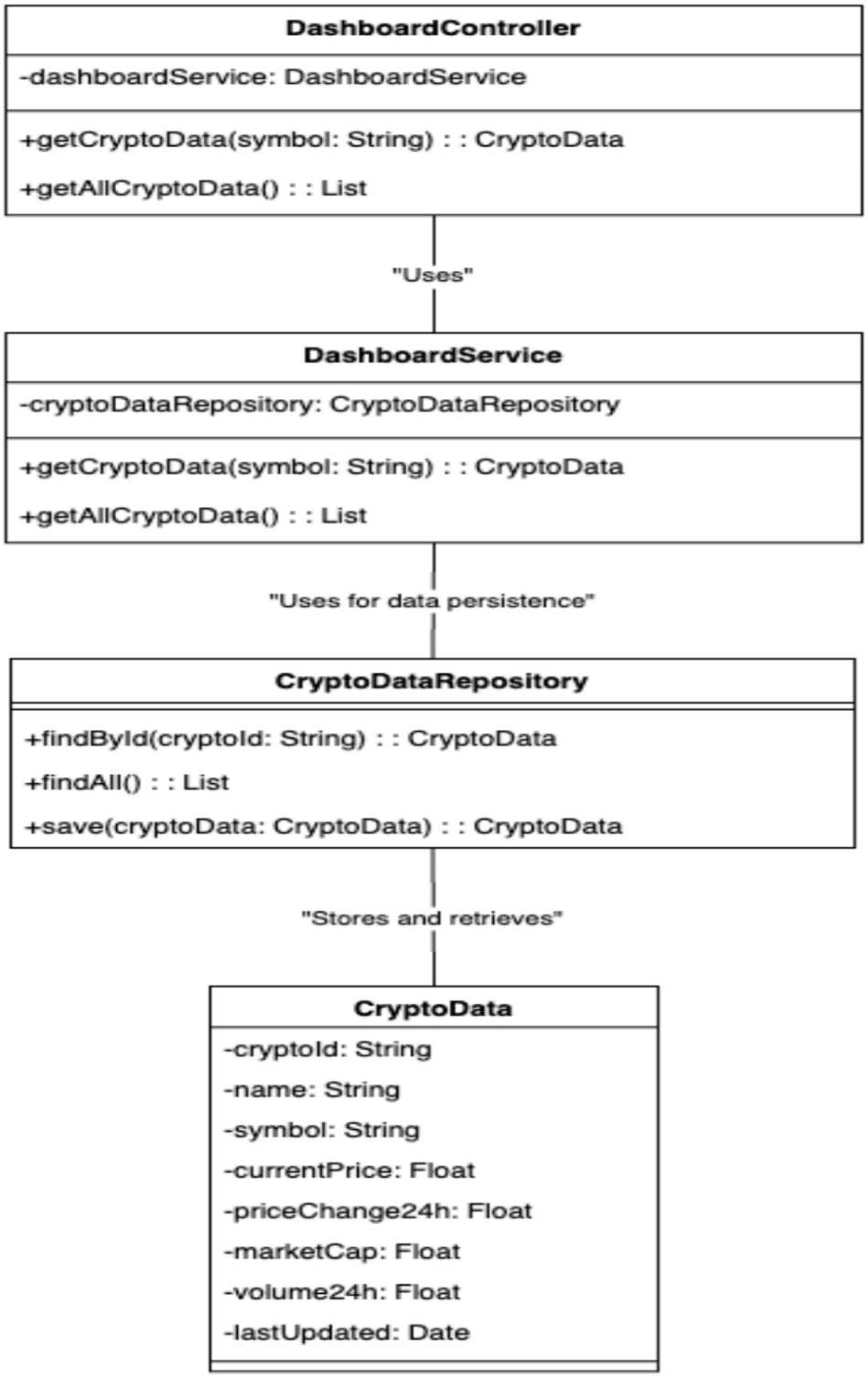
Component Diagram



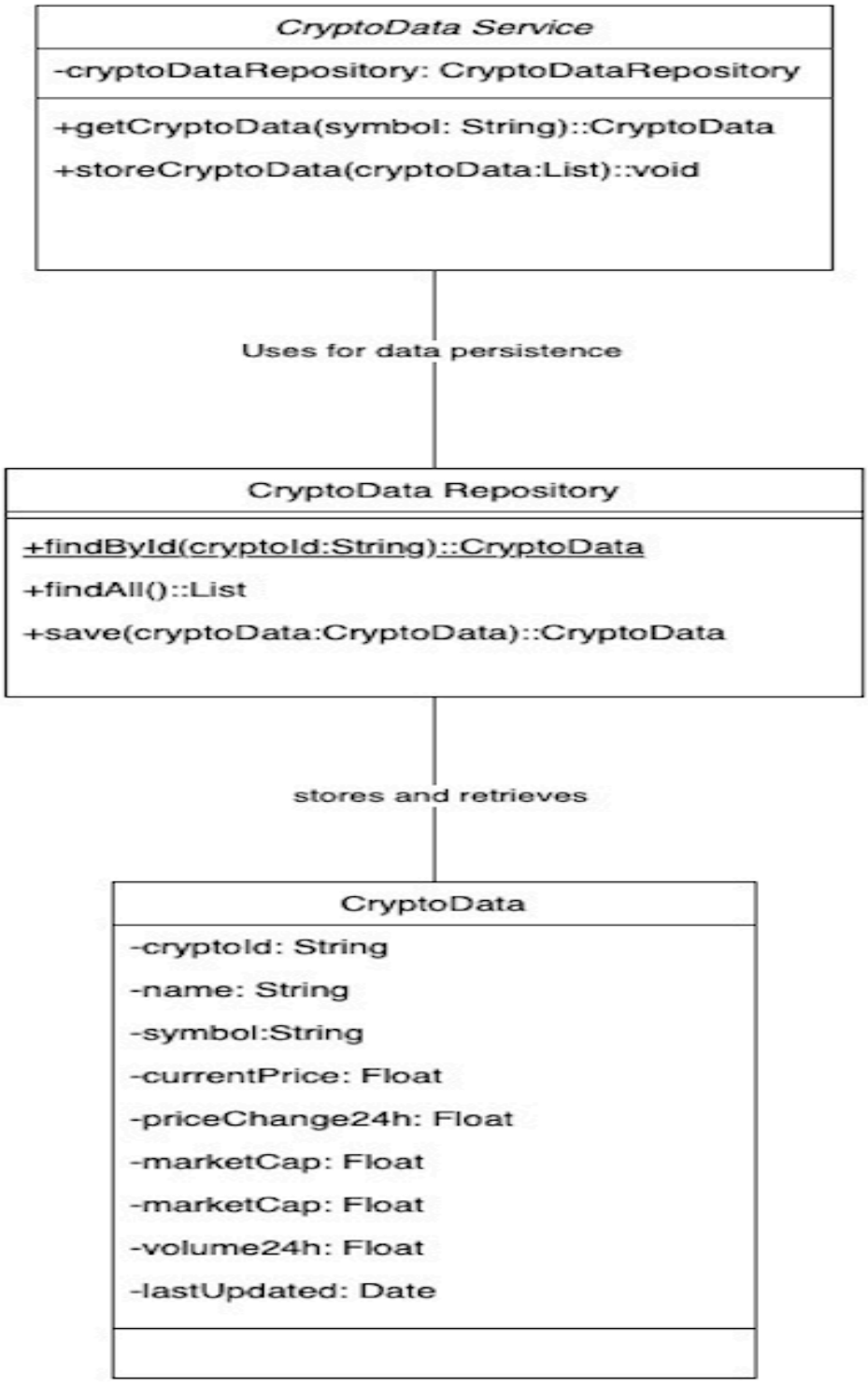
Class Diagram depicting Crypto Data Scraper Component



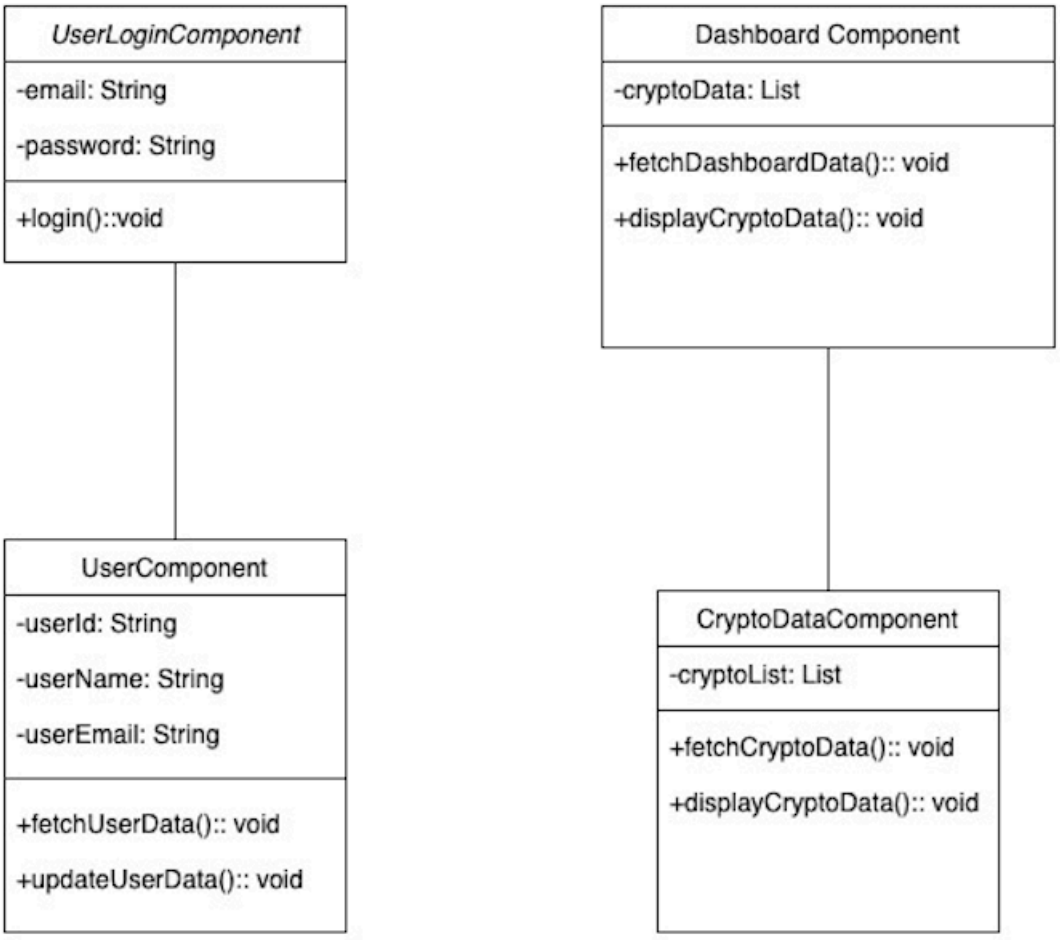
Class Diagram depicting Crypto Data API



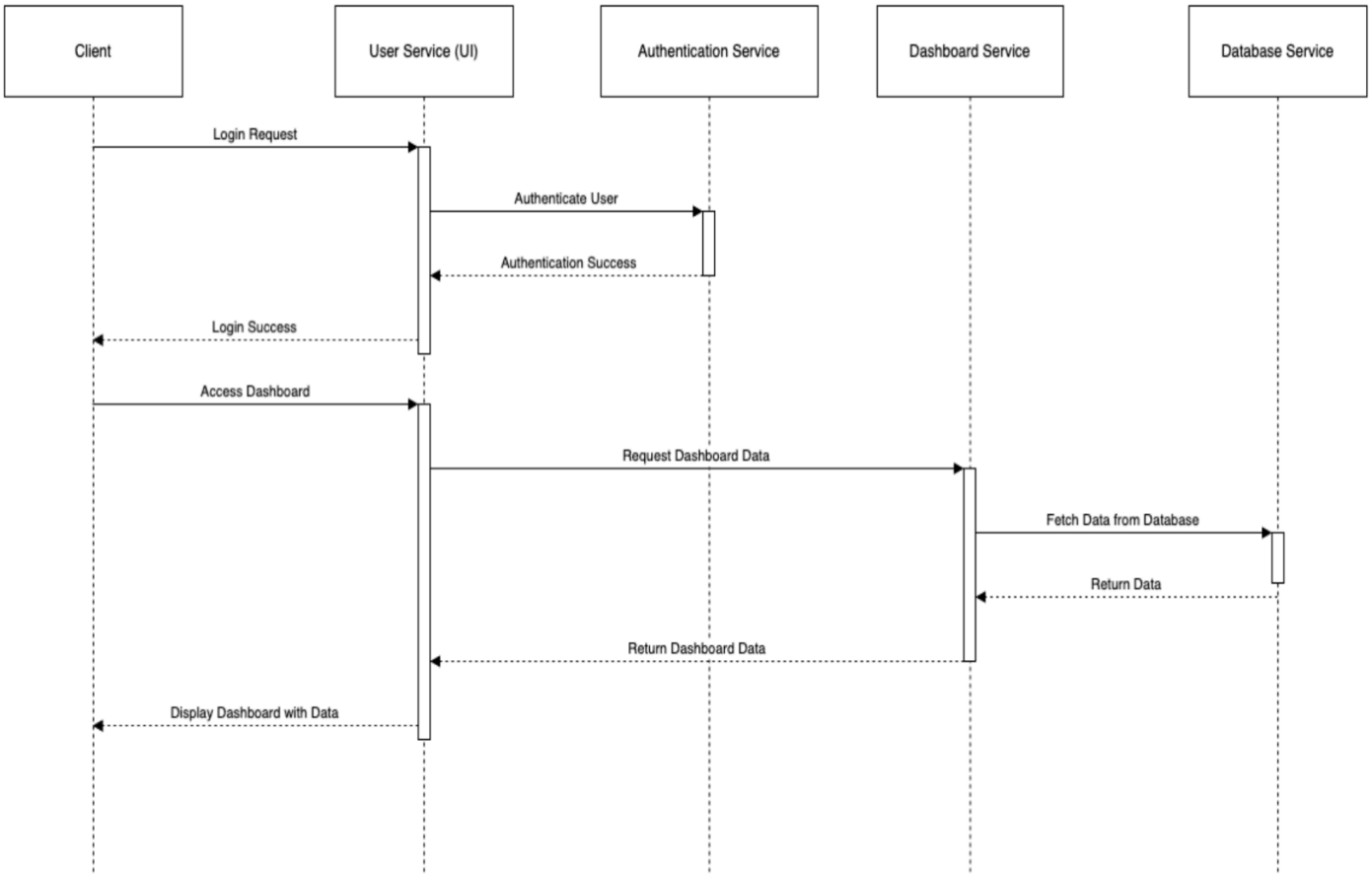
Class Diagram Depicting Crypto Data Processing Service



Component Diagram depicting React UI Components



Sequence Diagram depicting User Dashboard Access



Rationale

1. We have selected Microservices Architecture because of its flexibility in using different technologies, their underlying architectures, deployment, integration and communication along with the properties combining the benefits of event-based and client-server architectures.
2. The set of functional requirements/capabilities has been limited so that the system is useful, neat, easy to use, provides core functionalities, does not confuse users with a lot of unnecessary details and is scalable, accurate with data, available and has a good response time.
3. Component Diagram depicts the necessary components required for executing the functionalities and the interaction between them depicts the execution workflow through message passing, API calls and function calls.
4. Each Class diagram depicts low-level details like properties, methods, and function calls for respective components which help us during the implementation

5. A Sequence diagram at the end depicts how a user accesses the dashboard and informs about one of the important interactions in the front-end which provides us with a good use case of providing neat, useful and visually appealing crypto-currency information to the user.
6. We are using the top ten most popular cryptocurrencies as data relevant to them is available more widely. Those 10 cryptocurrencies are- Bitcoin, Ethereum, Solana, XRP, Dogecoin, Cardano, Tron, Shiba Inu, Avalanche, Toncoin.