**Project Description**  
The main functionality involves pulling the latest available data for each issuer on the stock exchange, identifying missing data, and filling gaps by fetching transactions from the past. The app uses Prisma ORM to manage database interactions with a PostgreSQL database, storing data efficiently with a structured schema.

The implementation includes four core modules:

1. **Issuer Fetching**: Identifies available stock issuers.
2. **Last Date Checking**: Determines the latest date for which data is available for each issuer.
3. **Missing Data Fetching**: Fetches and parses historical data if gaps are detected.
4. **Database Insertion**: Stores parsed data in the PostgreSQL database.

**Functional Requirements**

1. **Fetch Issuer Data**: Retrieve a list of stock issuers from the stock exchange website.
2. **Get Last Available Date**: Check the database for the latest date of recorded data for each issuer to identify data gaps.
3. **Fetch Historical Data**: Retrieve historical transaction data for missing dates.
4. **Parse and Store Data**: Extract relevant fields (price, quantity, date, etc.) from the data and store them in the PostgreSQL database.
5. **Error Logging**: Log any errors during data fetching or insertion processes for debugging and record-keeping.

**Non-Functional Requirements**

1. **Performance**: Ensure efficient data retrieval and insertion processes to minimize latency and optimize resource usage.
2. **Scalability**: Capable of handling additional issuers and increased transaction volume as the database grows.
3. **Reliability**: Recover from failures by retrying failed operations or handling exceptions without compromising data integrity.
4. **Maintainability**: Modular code structure, separating data retrieval, parsing, and storage logic for easier debugging and future updates.

**Personas and User Scenarios**

**Keiko, the Academic Researcher**

**Persona:** Keiko, 38, is a university finance professor who conducts extensive research on stock volatility for academic publications. With her busy schedule, she needs a reliable source of clean, continuous stock data for her studies. The application's error-logging and gap-filling features enable Keiko to ensure her data remains accurate and up-to-date, even if external factors cause temporary data gaps.

**User Scenario:** Before running simulations on volatility patterns, Keiko uses the app to pull comprehensive stock data across several issuers. If any gaps are identified, the application fills them quickly, saving her time and ensuring she works with complete data sets, which is critical for her rigorous research standards**.** By addressing both functional and non-functional requirements, this application ensures reliability and effectiveness for users who depend on accurate stock data for analysis and decision-making.

**Jordan, the Independent Financial Blogger**

**Persona:** Jordan, 31, writes a financial advice blog and regularly posts stock performance analyses for a wide readership. Since his followers rely on real-time and historical data for personal investments, Jordan uses the application to effortlessly retrieve current and past transaction data. The app’s efficiency in fetching and storing data with minimal latency ensures his blog posts remain timely and relevant.

**User Scenario:** Jordan initiates the app just before finalizing a post, ensuring he has up-to-the-minute data. He appreciates the modular structure of the app that allows him to retrieve data by issuer or date. This way, he can present the latest figures with confidence, enhancing his credibility among his readers.

**Data Population Timer: 1:56.594 (m:ss.mmm)**

**Ali Zhupani 211142  
Malik Alilovski 211532**