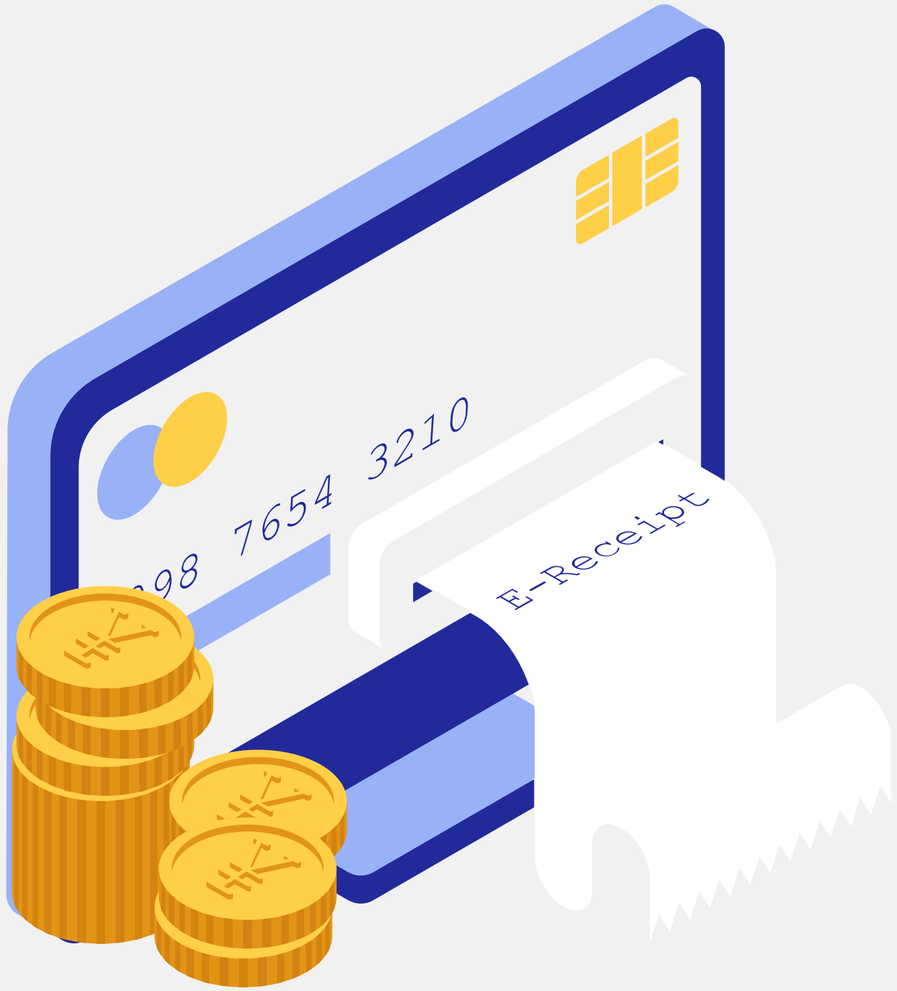


FICC Coding Challenge Presentation Deck

RAHMAN Faiyaz

The Chinese University of Hong Kong

July 15, 2022

Overview

- Completed all essential components, using 7 separate REST APIs, along with a Front-End component for the dashboard and additional components
- Adopted a microservice-based REST API architecture
- Implemented additional component #1: Customizable Reporter
- Created other components to give users a diagnostic view of their trades, and a real-time view of the market across different metrics
- Used: Python/Flask, React.js/MaterialUI, SQLite embedded database

Architecture & Design

The program uses 7 REST APIs to extensively decouple each component from one another.

Advantages:

- Each component can be tested individually
- Easier to develop codebase and find errors (non-monolithic codebase)

Components

i. Event Generator

Create your website address.Reads events from events.json, streams it to (ii, iii) depending on Event type

ii. Market Data Producer

Responsible for keeping track of Bond/FX prices. It publishes prices for components to evaluate trades (iii) or to generate reports (v, vi)

iii. Trade Data Producer

Streams trade events to (iv, v) after evaluating value, depending on order type. It alerts (i) if the order is impossible to execute due to insufficient liquidity, positions, or price history.

iv. Cash Adjuster

Responsible for keeping track of desk liquidity. It validates Buy orders if enough cash is available for a purchase. It also interacts with (v) to make necessary adjustments.

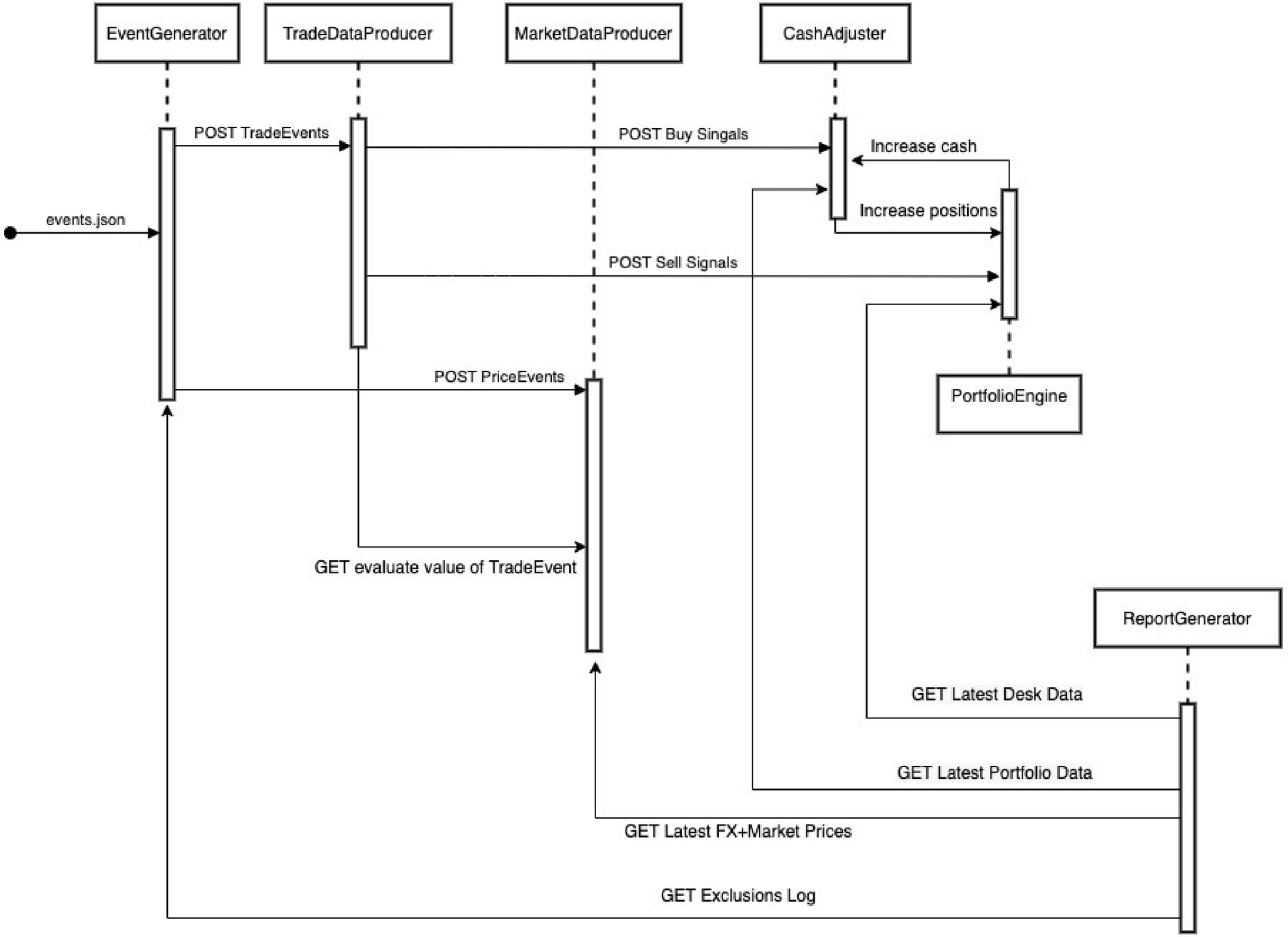
v. Portfolio Engine

Responsible for keeping track of positions in every book. It validates Sell orders if enough positions are available for the sale. It also interacts with (iv) to make necessary adjustments.

vi. Report Generator

Queries (iv, v) to make necessary reports when target events are reached. Primarily makes grouped-summations of data as per criterias in the specification.

System Diagram

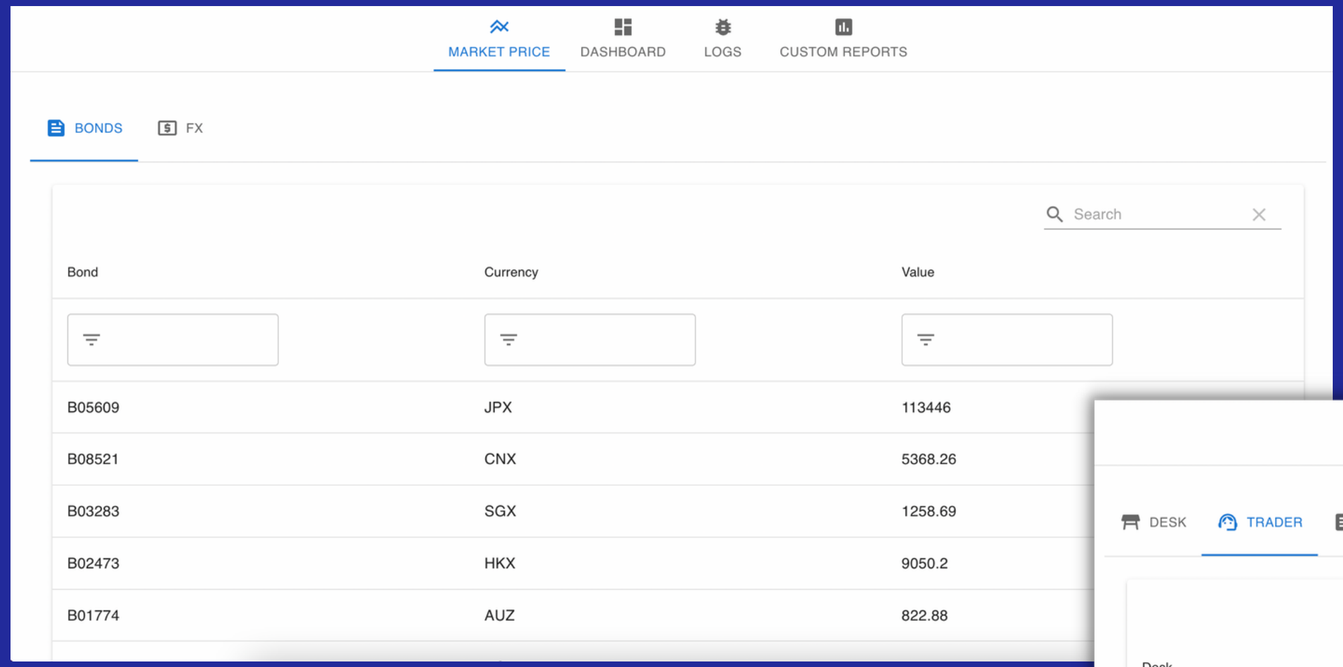


Metrics

The application primarily uses SQLite to store data. The most computationally expensive operation for this application is the calculation of bond prices using the underlying currency's FX rates and its market price. As all of this data is first stored into a hash map before calculations are conducted, algorithmic complexity is near O(n)

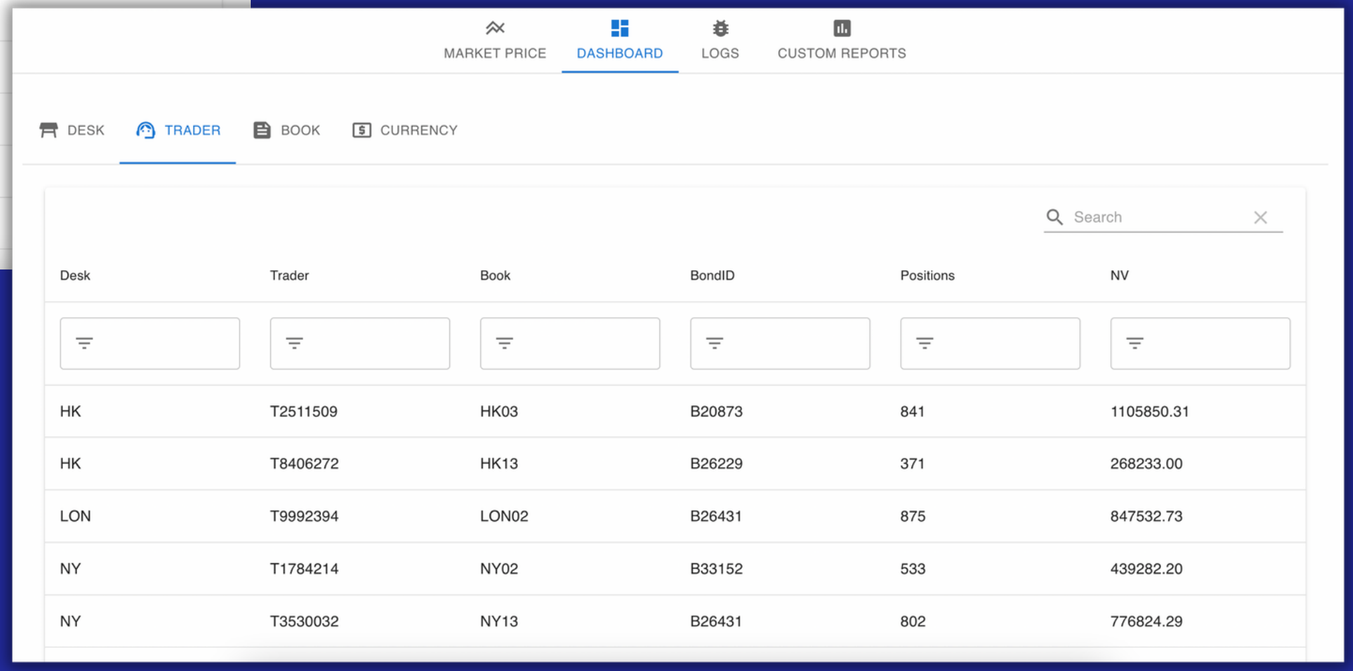
The chief bottleneck in this application would be the network interface. As many requests are sent internally in the application, the overhead can become significant

UI DEMO



DASHBOARD

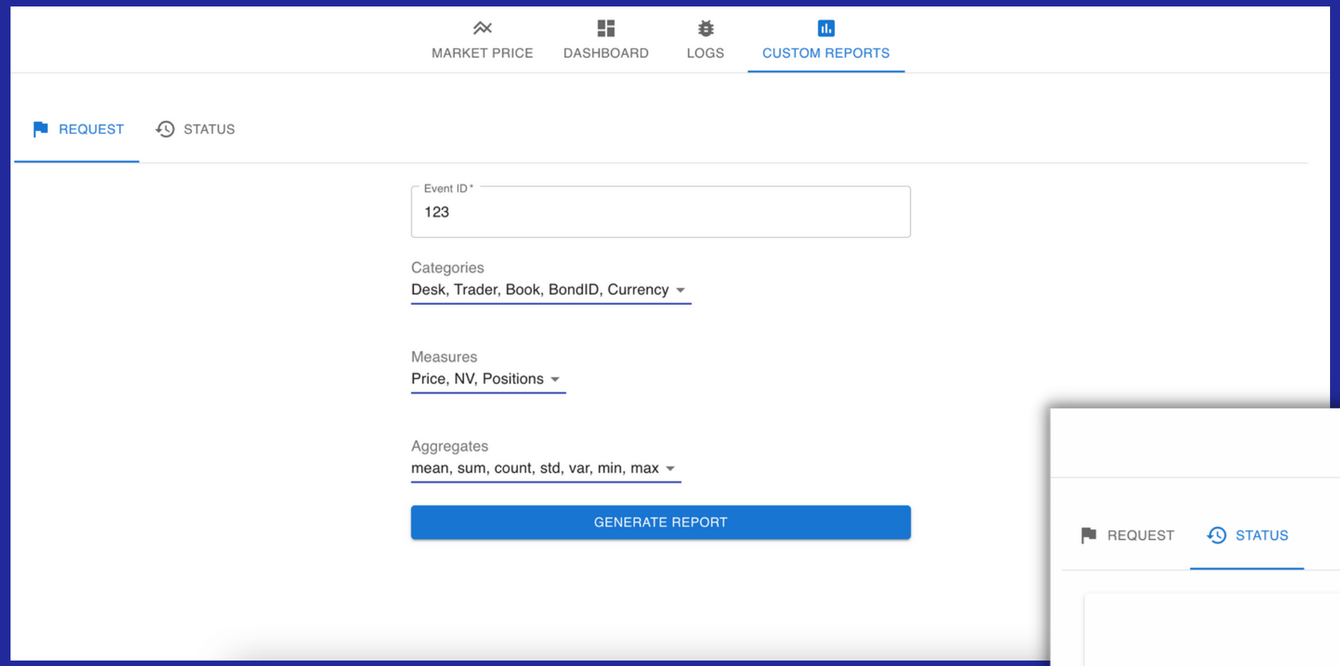
Displays a ticking dashboard of portfolio data grouped by different attributes



MARKET PRICE

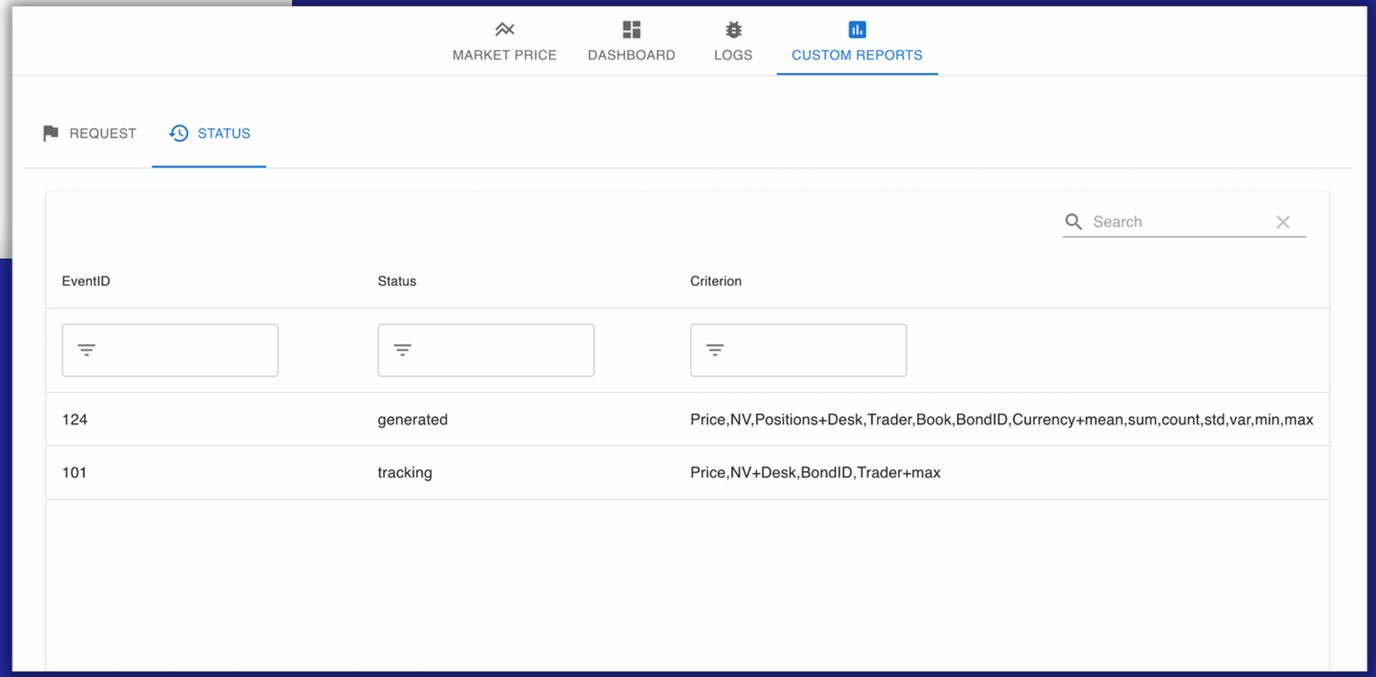
Displays latest bond prices and FX rates

ADDITIONAL COMPONENT: CUSTOM REPORT



STATUS

Displays status of custom report generation requests



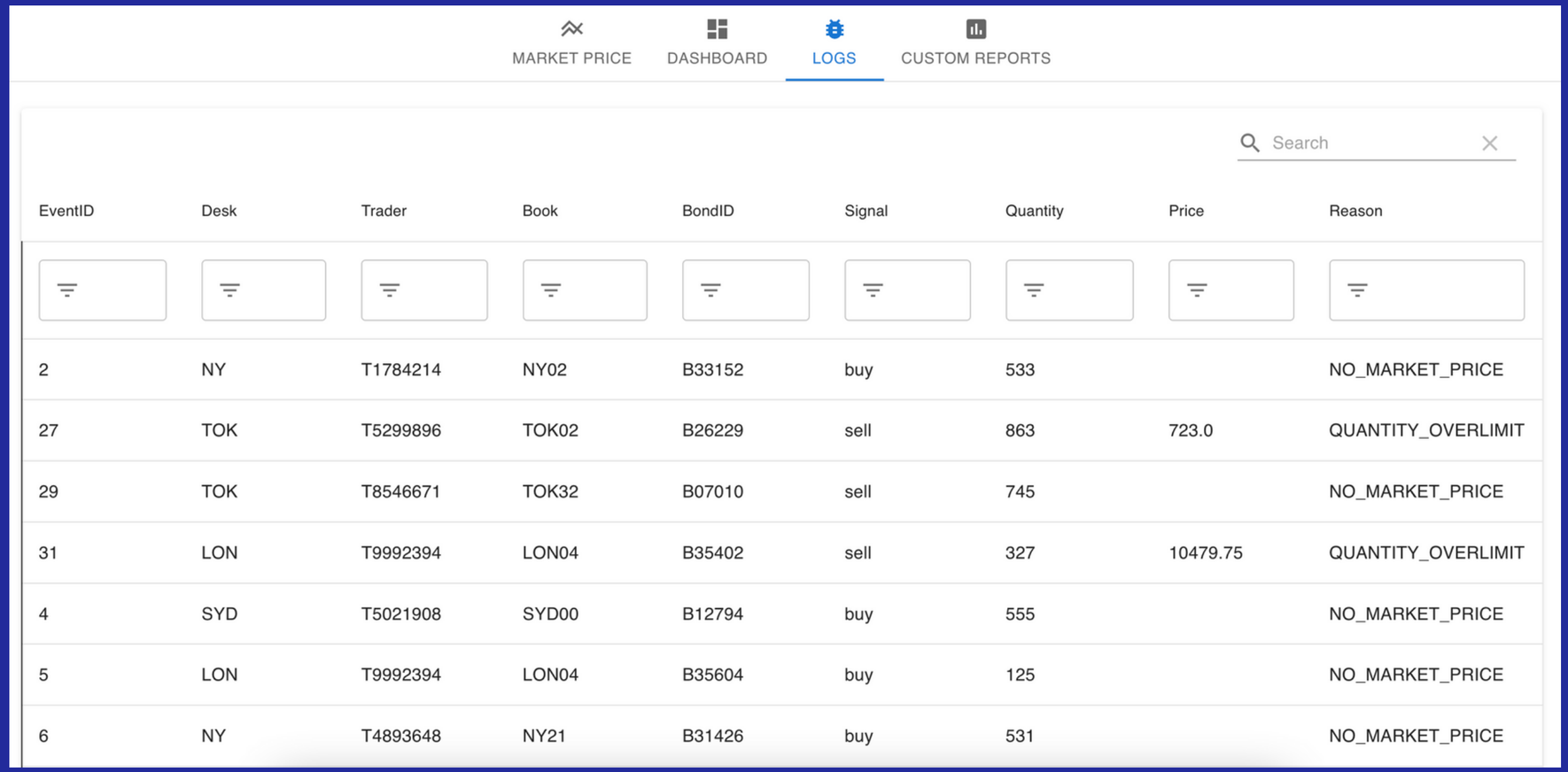
REQUEST

Send request to generate custom reports

ADDITIONAL COMPONENT: LOGS

LOGS

Displays records of invalid events



EventID	Desk	Trader	Book	BondID	Signal	Quantity	Price	Reason
2	NY	T1784214	NY02	B33152	buy	533		NO_MARKET_PRICE
27	TOK	T5299896	TOK02	B26229	sell	863	723.0	QUANTITY_OVERLIMIT
29	TOK	T8546671	TOK32	B07010	sell	745		NO_MARKET_PRICE
31	LON	T9992394	LON04	B35402	sell	327	10479.75	QUANTITY_OVERLIMIT
4	SYD	T5021908	SYD00	B12794	buy	555		NO_MARKET_PRICE
5	LON	T9992394	LON04	B35604	buy	125		NO_MARKET_PRICE
6	NY	T4893648	NY21	B31426	buy	531		NO_MARKET_PRICE

Key Takeaways

AUTOMATION

Most automation, such as CI or scripting, is usually worth the investment, since time spent breaks even quite quickly

INCREMENTED DEVELOPMENT

Incremental developments are always better than large monolithic ones, as it makes testing and debugging easier

Thank You