CST8916 Assignment 1

Real-time stock market monitoring application

Group 16

Shaoxian Duan

Xihai Ren

Introduction

Project Goal: Build a **real-time stock market monitoring application**

- **Core Technologies:**
  REST API: Traditional request-response model
  GraphQL: Flexible query-based API
  WebSockets: Real-time bidirectional communication

- **System Features:**
  Search for stocks
  Subscribe to real-time stock updates
  Data visualization

Section 1: REST and GraphQL for Data Requests and Updates

REST API vs. GraphQL

REST API

- Representational State Transfer
- HTTP Methods
  - GET – Retrieve stock data.
  - POST – Add new stock records.
  - PUT/PATCH – Update existing stock data.
  - DELETE – Remove stock entries.

GraphQL

- Query language for APIs
- Key Operations
  - Queries – Fetch stock data with specific fields.
  - Mutations – Add, update, or delete stock records.
  - Subscriptions – Real-time updates via WebSockets.

Section 1: REST and GraphQL for Data Requests and Updates

REST API vs. GraphQL: Comparative Analysis

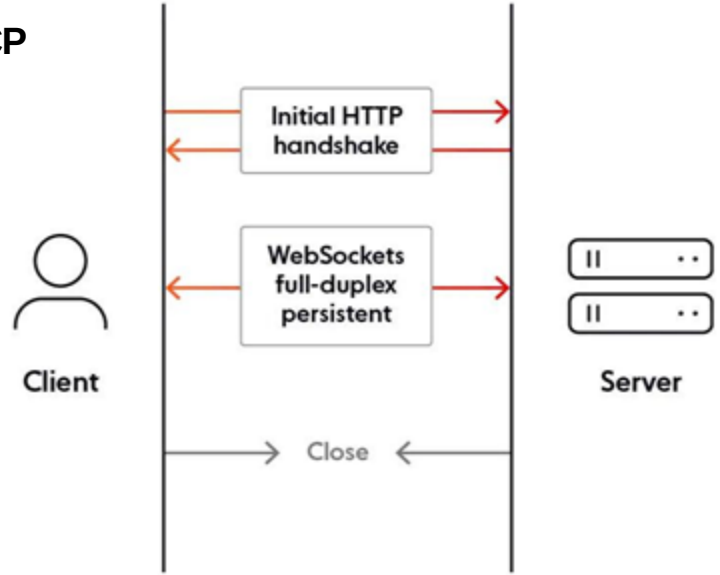
Feature	REST API	GraphQL
Endpoints	Multiple endpoints	Single endpoint
Data Fetching	Fixed response format	Customizable response
Real-time Support	Requires polling	Supports subscriptions
Performance	Can over-fetch or under-fetch data	Optimized for efficiency
Error Handling	HTTP status codes	GraphQL error field

Section 2: WebSockets for Real-time Communication

WebSocket

A WebSocket is **a persistent, two-way(full-duplex) TCP connection** between the server and the client.

- Over TCP
- Persistent connection
- Full-duplex (bi-directional)
- Event-driven
- Low latency



Section 2: WebSockets for Real-time Communication

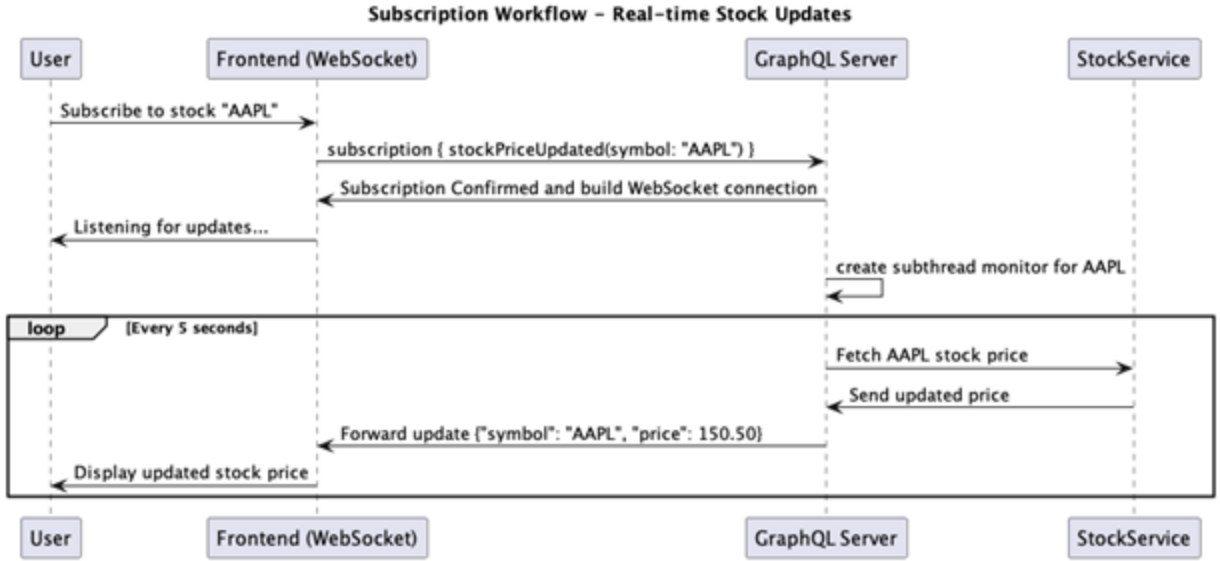
Why Use WebSockets

Feature	REST API	GraphQL Queries	WebSockets
Mechanism	Request-response	Request-response	Persistent connection
Real-time Support	Requires polling	Requires polling	Native real-time updates
Performance	Slower due to repeated requests	Medium (reduces over-fetching)	Fastest – instant updates
Server Load	High (frequent polling)	Medium (efficient queries)	Low (push-based updates)
Best For	Static data fetching	Optimized data selection	Real-time stock price tracking

Section 2: WebSockets for Real-time Communication

Workflow in our use case

1. User subscribes to stock updates.
2. Frontend sends a subscription request (GraphQL Subscription/WebSockets).
3. Server confirms the request and establishes a WebSocket connection.
4. Background process monitors stock prices and detects changes.
5. Real-time updates are pushed to the subscribed clients via WebSockets.
6. Frontend updates UI dynamically without reloading.



Section 3: Technology Recommendation and Justification

- **Hybrid Approach**: Combining REST API, GraphQL, and WebSockets.
- **Technology Roles in the System:**
  - REST API: Fetch bulk stock data from external sources.
  - GraphQL Queries: Allow flexible, client-driven data selection.
  - GraphQL Subscriptions (WebSockets): Enable real-time stock updates.

Factor	REST API	GraphQL Queries	GraphQL Subscription
API Wrapping	• Encapsulates third-party APIs • Standardizes data format • Supports caching	—	–-
Real-time Support	—	—	• Best for real-time updates • Low-latency push via WebSockets
Ease of Use	Simple and Widely supported	Flexible Queries	—

Live Demo

Thank You All