SIMPLELEARN Full Stack Developer - MERN Stack Master's Program

Course – 5 Develop a Reliable Backend with Node and Express

Assessment – Writeup

Student Name : Vedaang Sharma

Project Title: Stock API Access Using Express JS

1. Project Overview

The Stock API Access Using Express JS project is a backend-driven web application designed to provide real-time stock market data and analytics for individual investors and financial professionals. The platform integrates external stock market APIs to fetch live market information and exposes structured RESTful endpoints for consumption by frontend applications or API clients.

Purpose

- Enable access to real-time stock market data through a reliable backend system
- Simplify stock data retrieval for investors and analysts
- Demonstrate backend development skills using industry-relevant tools
- Build a scalable, secure, and API-driven stock data service
- Replace manual stock tracking with automated real-time data fetching

This project focuses on backend development using Node.js, Express.js, MongoDB, and Axios, following industry-standard architectural practices.

---

2. Features

Real-Time Stock Data Retrieval

- Fetches live stock market data from an external Stock Market API (Finnhub)
- Provides real-time updates for multiple stock symbols
- Returns data in structured JSON format

RESTful API Endpoints

```
/api/stock/all – Fetches all available stock symbols and related metadata

/api/stock/:symbol – Fetches real-time price data for a specific stock

/api/stock/:symbol/profile – Retrieves company profile information
```

```
/api/stock/:symbol/history – Retrieves historical stock price data
```

Secure API Key Management

- Stock API keys are stored securely using environment variables
- Prevents hardcoding of sensitive credentials
- Improves security and maintainability

MongoDB Integration

- MongoDB is used for persistent data storage
- Stores user portfolios and tracked stock information
- Supports scalability for large datasets

Modular Backend Architecture

- Clean separation of concerns using MVC pattern
- Independent layers for routes, controllers, and services
- Easy to extend and maintain

Error Handling and Stability

- Graceful handling of API failures and invalid requests
- Proper HTTP status codes and error responses
- Centralized error management

---

3. Implementation Details

Backend Technologies Used

The project uses the following industry-relevant backend technologies:

- Node.js – Runtime environment for scalable backend development
- Express.js – Web framework for routing and middleware handling
- MongoDB – NoSQL database for efficient data storage
- Mongoose – ODM for MongoDB schema modeling
- Axios – For making HTTP requests to external stock APIs

- dotenv – Environment variable management

Project Folder Structure

```
Stock-API/
│
├── config/
│   └── db.js
│
├── models/
│   └── stockModel.js
│
├── routes/
│   └── stockRoutes.js
│
├── services/
│   └── stockService.js
│
├── .env
├── app.js
├── package.json
```

Stock API Integration

- The application integrates with the Finn hub Stock API
- Axios is used to fetch real-time stock data
- API responses are processed and returned via Express routes

Example API Call:

https://finnhub.io/api/v1/stock/symbol?exchange=US

Sample Data Response Structure

```json
{
  "currency": "USD",
  "description": "APPLE INC",
  "displaySymbol": "AAPL",
  "symbol": "AAPL",
  "type": "Common Stock"
}
```

MongoDB Configuration

- MongoDB Community Server is installed locally
- Database connection is established using Mongoose

- Data is stored in collections for future enhancements like portfolios

Connection String:

mongodb://127.0.0.1:27017/stock-platform

---

4. Workflow / Working of the Application

1. Server Initialization

- Express server is initialized
- MongoDB connection is established
- Middleware for JSON parsing is enabled

2. API Request Handling

- Client sends request to /api/stock/all
- Express routes forward request to the service layer
- Axios fetches real-time stock data from the API
- Data is returned as JSON response

3. Real-Time Stock Fetching

- External stock API processes request
- Live market data is returned
- Backend formats and sends response to client

4. Data Storage (Optional Enhancement)

- Stock or portfolio data can be stored in MongoDB
- Enables historical tracking and analytics

5. Response to Client

- User receives real-time stock market data
- Data can be consumed by frontend dashboards or tools like Postman

---

5. Features Summary

Feature	Description
Real-Time Stock Data	Live market data fetched via external API

REST API	Structured endpoints for stock data access
MongoDB Integration	NoSQL database support
Secure API Key	Environment-based key management
Modular Architecture	Clean, scalable backend design
Error Handling	Graceful API failure handling

---

6. Project Deliverables

I am submitting a .zip file of the source code developed by me in the project. The project is also hosted on my GitHub profile and deployed for live viewing through a hosting platform.

Links :

- GitHub Repository : https://github.com/gtathelegend/dynamic-online-stock-trading-platform
- Google Drive Link: https://drive.google.com/drive/folders/1qkw5kz9XH5UJnsHt592HXd6WUshE8reZ?usp=sharing

---

7. Conclusion

This project demonstrates the development of a real-time stock market backend system using modern backend technologies. By integrating Express.js with an external stock market API and MongoDB, the system provides reliable, scalable, and secure access to financial data.

The project reflects practical industry use cases such as:

- Real-time data handling
- API-driven architecture
- Secure credential management
- Backend scalability

Overall, the Stock API Access Using Express JS project successfully fulfils the objective of building a robust backend solution for financial data analysis.