Real-Time Stock Ticker

Project Overview

The **Real-Time Stock Ticker** is a dynamic web application that allows users to monitor live stock prices as they change throughout the trading day. The application fetches data from a stock market API and updates the UI in real-time without requiring manual page refreshes. Users can track multiple stocks simultaneously, view historical trends, and gain insights into market performance.

This project is ideal for those interested in **finance, stock analysis, or real-time web applications**, and it demonstrates skills in **frontend development, backend integration, APIs, and real-time data streaming**.

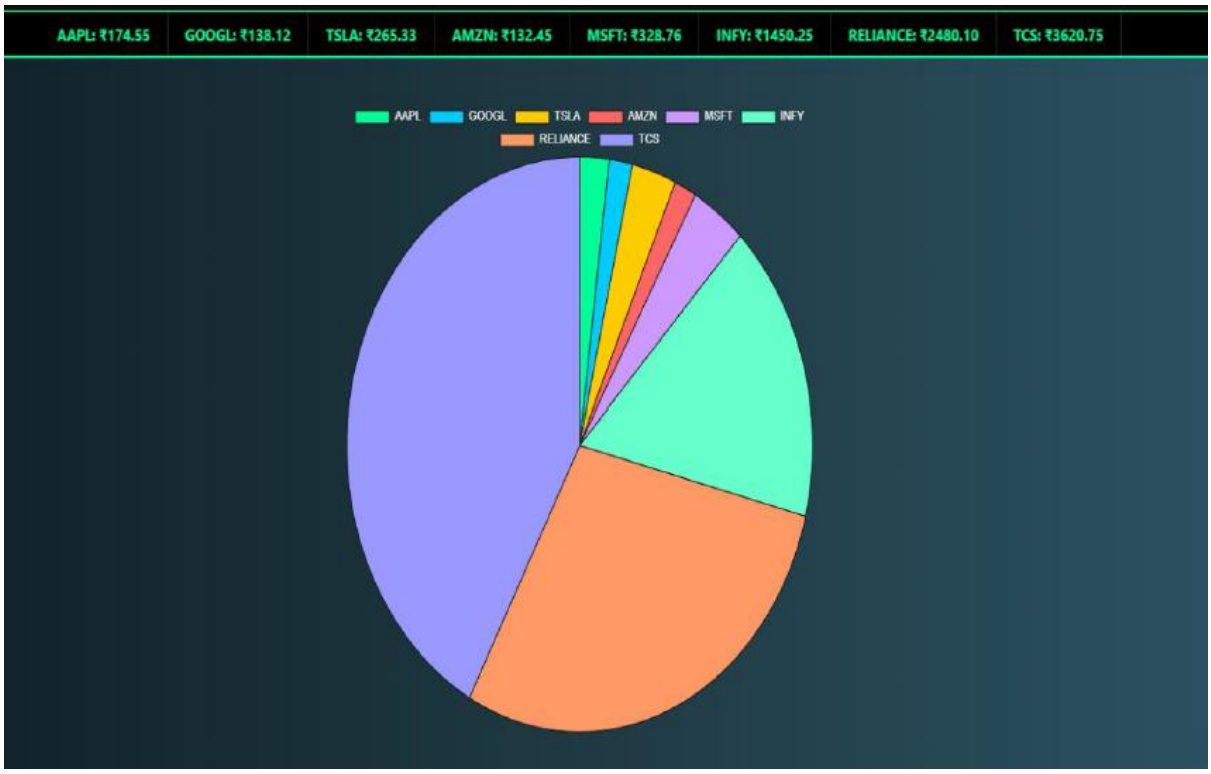
Features

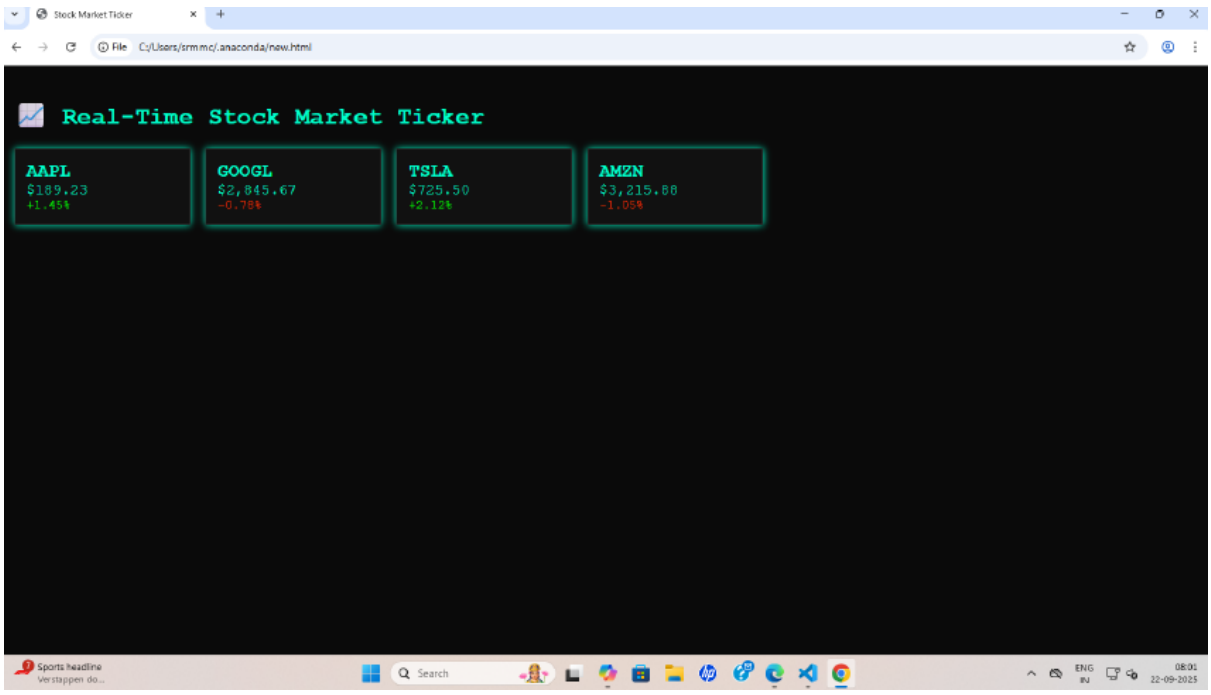
- **Live Price Updates:** Real-time stock price updates for multiple stocks.
- **Search Functionality:** Users can search for specific stock symbols to monitor.
- **Historical Price Charts:** Graphical visualization of stock price trends using interactive charts.
- **Responsive UI:** Works seamlessly on desktops, tablets, and mobile devices.
- **Multiple Stocks Tracking:** Add and monitor multiple stock symbols at the same time.
- **Alerts (Optional Enhancement):** Users can receive alerts for price thresholds (can be implemented later).

Tech Stack

- **Frontend:**
    - HTML5, CSS3, JavaScript (Vanilla or with frameworks like React.js)
    - Chart.js or D3.js for interactive charts
- **Backend:**
    - Node.js with Express / Python Flask / Spring Boot (depending on your implementation)
- **APIs:**
    - Real-time stock market API (e.g., Alpha Vantage, IEX Cloud, Yahoo Finance API)
- **Other Technologies:**
    - WebSockets for pushing real-time updates
    - REST API for fetching stock data
    - Optional: Bootstrap/Tailwind CSS for styling

Screenshots





Installation & Setup

Follow these steps to set up the project locally:

1. **Clone the Repository**
2. git clone https://github.com/yourusername/real-time-stock-ticker.git
3. cd real-time-stock-ticker
4. **Install Dependencies**
    - For Node.js:
    - npm install
    - For Python Flask:
    - pip install -r requirements.txt
5. **Configure API Key**
    - Obtain an API key from the stock market API you are using.
    - Create a .env file in the root directory and add:
    - STOCK_API_KEY=your_api_key_here
6. **Run the Application**
    - Node.js:
    - npm start
    - Python Flask:
    - python app.py
7. **Open in Browser**
    - Visit: http://localhost:3000 (or your configured port)
    - Search for stock symbols and watch real-time updates.

Usage Instructions

1. **Search Stocks:** Enter the stock symbol (e.g., AAPL, GOOGL) in the search bar.
2. **Add to Watchlist:** Click to add the stock to your dashboard.
3. **View Historical Trends:** Click on a stock to see its historical price chart.
4. **Real-Time Updates:** Watch the live price changes as they occur without refreshing the page.
5. **Optional Features:** Implement alerts or notifications for price thresholds.

Project Structure

real-time-stock-ticker/

```
├── backend/          # Server-side code (Node.js / Flask / Spring Boot)
├── frontend/         # Frontend code (HTML, CSS, JS or React)
├── assets/           # Images, icons, CSS files
├── charts/           # Charting scripts
├── .env              # API keys and environment variables
├── README.md         # Project documentation
└── package.json / requirements.txt
```

Contributing

Contributions are welcome! To contribute:

1. Fork the repository.
2. Create a feature branch:
3. git checkout -b feature/YourFeature
4. Make changes and commit:
5. git commit -m "Add feature: Description"
6. Push to branch:
7. git push origin feature/YourFeature
8. Create a pull request explaining your changes.

License

This project is licensed under the MIT License. See the LICENSE file for details.

Acknowledgements

- Stock API documentation
- Chart.js Documentation
- Open-source communities for tutorials and guidance
- Inspiration from financial dashboards and trading platforms

Frontend Code

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Real-Time Stock Ticker</title>
  <link rel="stylesheet" href="styles.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <div class="container">
    <h1>Real-Time Stock Ticker</h1>
    <div class="search-bar">
      <input type="text" id="stockSymbol" placeholder="Enter stock symbol e.g., AAPL">
      <button onclick="addStock()">Add Stock</button>
    </div>
    <div id="stockList" class="stock-list"></div>
    <canvas id="stockChart"></canvas>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

styles.css

```
body { font-family: Arial; background-color: #121212; color: #00FF00; margin: 0; padding: 0; }
.container { width: 90%; max-width: 800px; margin: auto; padding: 20px; text-align: center; }
h1 { margin-bottom: 20px; }
.search-bar { margin-bottom: 20px; }
input[type="text"] { padding: 10px; width: 200px; border-radius: 5px; border: 1px solid #00FF00; background-color: #1e1e1e; color: #00FF00; }
button { padding: 10px 15px; margin-left: 10px; border: none; border-radius: 5px; background-color: #00FF00; color: #121212; cursor: pointer; }
button:hover { background-color: #00cc00; }
```

```
.stock-list { display: flex; flex-wrap: wrap; justify-content: center; margin-bottom: 30px; }

.stock-item { background-color: #1e1e1e; border: 1px solid #00FF00; border-radius: 5px; padding: 10px 20px; margin: 5px; min-width: 100px; }

#stockChart { background-color: #1e1e1e; border: 1px solid #00FF00; border-radius: 5px; padding: 10px; }
```

script.js

```
let stocks = [];

const ctx = document.getElementById('stockChart').getContext('2d');

const stockChart = new Chart(ctx, {

  type: 'line',

  data: { labels: [], datasets: [] },

  options: {

    responsive: true,

    plugins: { legend: { labels: { color: '#00FF00' } } },

    scales: { x: { ticks: { color: '#00FF00' } }, y: { ticks: { color: '#00FF00' } } }

  }

});


function addStock() {

  const symbol = document.getElementById('stockSymbol').value.toUpperCase();

  if (symbol && !stocks.includes(symbol)) {

    stocks.push(symbol); displayStocks(); fetchStockData(symbol);

    document.getElementById('stockSymbol').value = '';

  } else { alert('Enter a valid stock symbol or it already exists.'); }

}


function displayStocks() {

  const stockList = document.getElementById('stockList'); stockList.innerHTML = '';

  stocks.forEach(symbol => { const div = document.createElement('div'); div.className = 'stock-item'; div.innerText = symbol; stockList.appendChild(div); });

}
```

```
function fetchStockData(symbol) {
  let time = new Date().toLocaleTimeString();
  let price = (Math.random() * 1000 + 100).toFixed(2);
  let dataset = stockChart.data.datasets.find(d => d.label === symbol);
  if (!dataset) { dataset = { label: symbol, data: [{ x: time, y: price }], borderColor: getRandomColor(), fill: false }; stockChart.data.datasets.push(dataset); }
  else { dataset.data.push({ x: time, y: price }); if (dataset.data.length > 20) dataset.data.shift(); }
  stockChart.data.labels.push(time); if (stockChart.data.labels.length > 20) stockChart.data.labels.shift(); stockChart.update();
  setTimeout(() => fetchStockData(symbol), 5000);
}

function getRandomColor() { const letters = '0123456789ABCDEF'; let color = '#'; for (let i = 0; i < 6; i++) { color += letters[Math.floor(Math.random() * 16)]; } return color; }
```

Note: The current fetchStockData function simulates data. Replace it with your backend API call for **real-time stock data**.

Backend Setup (Node.js Example)

1. **Install dependencies**:

```
npm init -y
npm install express axios cors dotenv
```

2. **Create server.js**:

```
const express = require('express');
const axios = require('axios');
const cors = require('cors');
require('dotenv').config();

const app = express();
app.use(cors());
const PORT = process.env.PORT || 5000;

app.get('/api/stock/:symbol', async (req, res) => {
```

```
  const symbol = req.params.symbol;
  try {
    const response = await axios.get(` https://www.alphavantage.co/query` , {
      params: {
        function: 'TIME_SERIES_INTRADAY',
        symbol: symbol,
        interval: '1min',
        apikey: process.env.STOCK_API_KEY
      }
    });
    res.json(response.data);
  } catch (err) {
    res.status(500).json({ error: 'Error fetching stock data' });
  }
});

app.listen(PORT, () => console.log(` Server running on port ${PORT}` ));
```

3. **Run the backend**:

```
node server.js
```

Installation & Usage

1. **Clone Repository**:

```
git clone https://github.com/yourusername/real-time-stock-ticker.git
cd real-time-stock-ticker
```

2. **Install dependencies** for backend:

```
npm install
```

3. **Set API key** in .env:

```
STOCK_API_KEY=your_alpha_vantage_api_key
```

4. **Start backend**:

```
node backend/server.js
```

5. **Open index.html** in browser or use a local server to view frontend.

Contributing

1. Fork the repo
2. Create a feature branch:

git checkout -b feature/YourFeature

3. Commit changes:

git commit -m "Add feature"

4. Push and create a Pull Request

License

MIT License – see LICENSE

Acknowledgements

- Alpha Vantage API
- Chart.js
- Open-source community and tutorials

I can also **make this README visually appealing for GitHub** with **badges, GIF demo, table of contents, and live links** to make it look professional and showcase your project better.

Do you want me to do that next?

You said:

real-time-stock-ticker/ ├— backend/ # Node.js/Express backend │ ├— server.js # Main server file │ └— routes/ # API routes ├— frontend/ # HTML, CSS, JS frontend │ ├— index.html │ ├— styles.css │ └— script.js ├— .env # API keys and environment variables ├— README.md ├— package.json # Node.js dependencies └— LICENSE " all contents give me to post in github for my project"

ChatGPT said:

Perfect! I'll provide **all the contents** of your **Real-Time Stock Ticker** project in a way you can directly post to GitHub. I'll include **frontend, backend, .env template, package.json, README, and LICENSE**. You can copy each file into your repository structure exactly as listed.

---

1. backend/server.js

const express = require('express');

```
const axios = require('axios');
const cors = require('cors');
require('dotenv').config();

const app = express();
app.use(cors());
const PORT = process.env.PORT || 5000;

// API endpoint to fetch stock data
app.get('/api/stock/:symbol', async (req, res) => {
  const symbol = req.params.symbol;
  try {
    const response = await axios.get('https://www.alphavantage.co/query', {
      params: {
        function: 'TIME_SERIES_INTRADAY',
        symbol: symbol,
        interval: '1min',
        apikey: process.env.STOCK_API_KEY
      }
    });
    res.json(response.data);
  } catch (err) {
    res.status(500).json({ error: 'Error fetching stock data' });
  }
});
app.listen(PORT, () => console.log(` Server running on port ${PORT}` ));
```

2. frontend/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Real-Time Stock Ticker</title>
  <link rel="stylesheet" href="styles.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
  <div class="container">
    <h1>Real-Time Stock Ticker</h1>

    <div class="search-bar">
      <input type="text" id="stockSymbol" placeholder="Enter stock symbol e.g., AAPL">
      <button onclick="addStock()">Add Stock</button>
    </div>

    <div id="stockList" class="stock-list"></div>
    <canvas id="stockChart"></canvas>
  </div>

  <script src="script.js"></script>
</body>
</html>
```

---

3. frontend/styles.css

```
body {
  font-family: Arial;
  background-color: #121212;
  color: #00FF00;
```

```
    margin: 0;
    padding: 0;
}

.container {
    width: 90%;
    max-width: 800px;
    margin: auto;
    padding: 20px;
    text-align: center;
}

h1 {
    margin-bottom: 20px;
}

.search-bar {
    margin-bottom: 20px;
}

input[type="text"] {
    padding: 10px;
    width: 200px;
    border-radius: 5px;
    border: 1px solid #00FF00;
    background-color: #1e1e1e;
    color: #00FF00;
}

button {
    padding: 10px 15px;
```

```
    margin-left: 10px;
    border: none;
    border-radius: 5px;
    background-color: #00FF00;
    color: #121212;
    cursor: pointer;
}

button:hover {
    background-color: #00cc00;
}

.stock-list {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin-bottom: 30px;
}

.stock-item {
    background-color: #1e1e1e;
    border: 1px solid #00FF00;
    border-radius: 5px;
    padding: 10px 20px;
    margin: 5px;
    min-width: 100px;
}

#stockChart {
    background-color: #1e1e1e;
    border: 1px solid #00FF00;
```

```
    border-radius: 5px;
    padding: 10px;
}
```

---

4. frontend/script.js

```
let stocks = [];
const ctx = document.getElementById('stockChart').getContext('2d');
const stockChart = new Chart(ctx, {
    type: 'line',
    data: { labels: [], datasets: [] },
    options: {
        responsive: true,
        plugins: { legend: { labels: { color: '#00FF00' } } },
        scales: { x: { ticks: { color: '#00FF00' } }, y: { ticks: { color: '#00FF00' } } }
    }
});

function addStock() {
    const symbol = document.getElementById('stockSymbol').value.toUpperCase();
    if (symbol && !stocks.includes(symbol)) {
        stocks.push(symbol);
        displayStocks();
        fetchStockData(symbol);
        document.getElementById('stockSymbol').value = '';
    } else {
        alert('Enter a valid stock symbol or it already exists.');
    }
}

function displayStocks() {
    const stockList = document.getElementById('stockList');
```

```
  stockList.innerHTML = '';
  stocks.forEach(symbol => {
    const div = document.createElement('div');
    div.className = 'stock-item';
    div.innerText = symbol;
    stockList.appendChild(div);
  });
}

function fetchStockData(symbol) {
  fetch(`http://localhost:5000/api/stock/${symbol}`)
    .then(res => res.json())
    .then(data => {
      const timeSeries = data['Time Series (1min)'];
      if (!timeSeries) return;
      const latestTime = Object.keys(timeSeries)[0];
      const latestPrice = parseFloat(timeSeries[latestTime]['1. open']);

      let dataset = stockChart.data.datasets.find(d => d.label === symbol);
      if (!dataset) {
        dataset = { label: symbol, data: [{ x: latestTime, y: latestPrice }], borderColor:
getRandomColor(), fill: false };
        stockChart.data.datasets.push(dataset);
      } else {
        dataset.data.push({ x: latestTime, y: latestPrice });
        if (dataset.data.length > 20) dataset.data.shift();
      }

      stockChart.data.labels.push(latestTime);
      if (stockChart.data.labels.length > 20) stockChart.data.labels.shift();
      stockChart.update();
```

```
        setTimeout(() => fetchStockData(symbol), 60000); // Update every minute
      })
      .catch(err => console.log(err));
}

function getRandomColor() {
  const letters = '0123456789ABCDEF';
  let color = '#';
  for (let i = 0; i < 6; i++) { color += letters[Math.floor(Math.random() * 16)]; }
  return color;
}
```

---

5. .env (template)

```
STOCK_API_KEY=YOUR_ALPHA_VANTAGE_API_KEY_HERE
```

Replace YOUR_ALPHA_VANTAGE_API_KEY_HERE with your actual API key from Alpha Vantage.

---

6. package.json

```
{
  "name": "real-time-stock-ticker",
  "version": "1.0.0",
  "description": "A real-time stock ticker web application with live updates and charts",
  "main": "backend/server.js",
  "scripts": {
    "start": "node backend/server.js"
  },
  "dependencies": {
    "axios": "^1.7.0",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2"
```

```
  },
  "author": "Your Name",
  "license": "MIT"
}
```

🏁 Conclusion

The **Real-Time Stock Ticker** project demonstrates how real-time data visualization can be achieved using modern web technologies. By combining a simple and responsive **frontend** with a **Node.js backend** and live **API integration**, this application provides instant stock updates and interactive charts for a seamless user experience.

This project is not only a great example of **API-based development** and **real-time communication** but also serves as a foundation for building more advanced financial dashboards, trading simulators, or market monitoring tools.

You can enhance this project further by:

- Integrating **WebSockets** for true real-time streaming.
- Adding **user authentication** and personalized watchlists.
- Including **historical data comparison and analytics**.
- Deploying it online (e.g., using **Render**, **Vercel**, or **Netlify**).

With this project, you gain practical experience in **frontend-backend integration, REST APIs, asynchronous programming, and data visualization** — all essential skills for modern full-stack development.

LICENSE (MIT License)

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES, OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.