COLLEGE CODE:9605

COLLEGE NAME: CAPE INSTITUTE OF TECHNOLOGY

DEPARTMENT: BE.CSE 3RD YEAR

STUDENT NM-ID:1218D8AA15D7487895F48F1F7073B79F

ROLL NO:960523104002

DATE:08-10-2025

Completed the project named as Phase-5

TECHNOLOGY PROJECT NAME:

IBM-FE-REAL TIME STOCK TICKER

SUBMITTED BY,

NAME: AAROKKIA MAHIM D

MOBILE NO:7845758641

- PHASE 5 – Project Demonstration & Documentation

- Final Demo Walkthrough:
  - Project Setup:
    - **Technologies Used:**
      - **Frontend:** HTML, CSS, JavaScript (or React.js)
      - **Backend (optional):** Node.js + Express (if you use live API calls)
      - **API:** Alpha Vantage / Finnhub / Yahoo Finance API
      - **Real-time update:** setInterval() or WebSocket

    - Steps:
      - Open **VS Code**
      - Create folder `real-time-stock-ticker`
      - Inside it, create files:
        - ✓ script.js
        - ✓ index.html
        - ✓ style.css
      - Link your API key in script.js (if using external API)
      - Run using **Live Server**

  - Demo Flow (Walkthrough)
    - Step 1: Launch the App

      "This is my Real-Time Stock Ticker application. When I open it in the browser, I get a clean dashboard showing stock names and their current prices."

      Explain UI:

      - Header: "Real-Time Stock Tracker"
      - Search bar (optional)
      - Table or Cards showing:
        - ✓ Stock Symbol (e.g., AAPL, TSLA)
        - ✓ Current Price

✓ Change %
✓ Last Updated Time

- Step 2: Show Real-Time Updates
  "Here, the stock prices are automatically updated every few seconds."

  In Code:

```
function functionCall() {
const newsFeedContents = 'Newsfeed.xml';
fetch(newsFeedContents)
.then(response => response.text())

    // DOMparser is the interface which parses
    // a string having XML
    // and return XML document
    .then(str => new window.DOMParser()
    .parseFromString(str, "text/xml"))
    .then(data => {

   //Returns collection of child elements
   // of the matched selector.
    const items = data.querySelectorAll("item");

     let htmlOutput = ``;
/* The concatenation of htmlOutput
```

```
   string is applied for each item
   element of array. querySelector
   fetches first element of the descendant */

items.forEach(itemElement => {
    htmlOutput += `
        <div>
            <h3>
                <a href=
                "${itemElement.querySelector("link").innerHTML}"
                    target="_blank" rel="noopener">
                    ${itemElement.querySelector("title").innerHTML}
                    <button style=
                    "border-radius:8px;
                    background-color:rgb(32, 94, 170);
                    color:white;border:none">
                        RSS
                    </button>
                </a>
            </h3>
            <p>
              ${itemElement
            .querySelector("description").innerHTML}
            <p>
```

```
            </div>
            `;
        });

        // Returns the htmlOutput string
        // in the HTML body element
        // Check whether your query returns null
        var input =
            document.getElementById("RSSfeedID");
        if (input) {
            input.innerHTML = htmlOutput;
        }
        document.body.style.backgroundColor = "rgb(203, 245, 245)";
    });
}
```

"I've used the JavaScript setInterval() function to call the API every 5 seconds and update the prices dynamically without refreshing the page."

- Step 3: Explain API Integration
  "The data is fetched from the Alpha Vantage API using a GET request. Here's a snippet:"

```
async function fetchStockData() {
const response = await fetch(`https://api.example.com/stock/AAPL?apikey=YOUR_KEY`);
const data = await response.json();
document.getElementById('price').innerText = data.price;
```

}

“This function fetches the latest stock price and updates the page automatically.”

- Step 4: User Interaction
  “The user can enter any stock symbol in the search bar, and the application will display its current value.”

  Example:

  - Type **TSLA**
  - Click **Search**
  - The app displays Tesla’s live stock data

- Step 5: Visual Features
  “I’ve added color indicators — green for price increase, red for decrease — to make the data easy to understand at a glance.”

```
@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@500&amp;family=R
oboto:wght@500&amp;display=swap");


* {
   padding: 0;
   margin: 0;
   box-sizing: border-box;
}




body {
```

```
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #c3c3c3;
}

p {
    font-family: "Roboto", sans-serif;
    opacity: .8;
    line-height: 1.4rem;
}

a {
    text-decoration: none;
}

ul {
    list-style: none;
}

.flex {
    display: flex;
    align-items: center;
}

.container {
```

```
    max-width: 1180px;
    margin-inline: auto;
    overflow: hidden;
}

nav {
    background-color: #484e52;
    color: white;
    position: fixed;
    top: 0;
    z-index: 99;
    left: 0;
    right: 0;
}

.main-nav {
    justify-content: space-between;
    padding-block: 8px;
}



.nav-links ul {
    gap: 16px;
```

```
}

.hover-link {
    cursor: pointer;
}


.nav-item.active {
    color: var(--accent-color);
}

.search-bar {
    height: 32px;
    gap: 8px;
}

.news-input {
    width: 200px;
    height: 100%;
    padding-inline: 12px;
    border-radius: 4px;
    border: 2px solid #bbd0e2;
    font-family: "Roboto", sans-serif;
}
```

```
.search-button {
    background-color: #208c5b;
    color: white;
    padding: 8px 24px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-family: "Roboto", sans-serif;
}



main {
    padding-block: 20px;
    margin-top: 80px;
}

.cards-container {
    justify-content: space-between;
    flex-wrap: wrap;
    row-gap: 20px;
    align-items: start;
}
```

```
.card {
    width: 360px;
    min-height: 400px;
    box-shadow: 0 0 4px #d4ecff;
    border-radius: 4px;
    cursor: pointer;
    background-color: #fff;
    overflow: hidden;
    transition: all 0.3s ease;
}

.card:hover {
    box-shadow: 1px 1px 8px #d4ecff;
    background-color: #f9fdff;
    transform: translateY(-2px);
}

.card-header img {
    width: 100%;
    height: 180px;
    object-fit: cover;
}
```

```
.card-content {
    padding: 12px;
}

.news-source {
    margin-block: 12px;

}

.hover-link:hover {
    color: #b9b9b9;
    text-decoration: underline;
}
```

- Step 6: Responsive Design
  "The layout is responsive, so it adjusts perfectly on both desktop and mobile screens."

- Step 7: Backend / WebSocket (if used)
  "To make it truly real-time, I've implemented a WebSocket that streams live updates from the stock API without repeated API calls."

- Output Demo Summary
  - Displays live stock data
    - Updates automatically
    - Allows searching for different symbols
    - Visual indicators for price changes
    - Responsive & real-time behavior

- Challenges Faced
    - **API rate limits** — handled using caching or reduced fetch frequency
    - **Real-time synchronization** — solved using setInterval/WebSocket
    - **Cross-origin issues (CORS)** — fixed via proxy or server configuration
- Future Enhancements
    - Add **charts** for stock trends
    - Add **user login** to save favorite stocks
    - Use **WebSocket streaming** for true instant updates
    - Implement **dark/light mode**

- Project Report:
    - Objectives

      The main objectives of this project are:
        - To design and develop a **web-based application** that displays **live stock prices**.
        - To fetch and update stock data automatically from an external **financial data API**.
        - To allow users to **search** and **track multiple stock symbols**.
        - To visually indicate **price movement trends** (green for gain, red for loss).
        - To ensure **real-time updates** and **responsive design** for different screen sizes.
    - Problem Statement

Traditional stock-tracking websites may require manual refreshes or contain delayed data feeds. Users need a lightweight, responsive, and real-time tool to monitor live stock values.
This project solves that by implementing automatic data updates and a clean interface suitable for both investors and learners.

- Scope of the Project

The **Real-Time Stock Ticker** application is designed for:

- Individual investors

- Financial students
- Analysts who want to observe stock fluctuations in real time

Scope Includes:

- Displaying real-time stock information
- Fetching data from APIs (e.g., Alpha Vantage / Finnhub / Yahoo Finance)
- Auto-refreshing data without reloading the page
- Responsive layout compatible with desktop and mobile devices

- System Requirements

Software Requirements

- VS Code or any code editor
- Web browser (Chrome, Edge, Firefox)
- Live Server Extension (for testing)
- API key from a stock data provider (e.g., Alpha Vantage)

Hardware Requirements

- Minimum 4 GB RAM
- Dual-core processor
- Internet connection (for API data fetch)

- System Design

Architecture:

- Frontend (Client):
  - ✓ HTML → Structure
  - ✓ CSS → Styling
  - ✓ JavaScript → Fetching and updating stock data dynamically
- API Layer:

  Uses a third-party API (e.g., Alpha Vantage or Yahoo Finance API) to fetch live stock prices.
- Backend:

  Node.js with Express (used if data needs to be cached or processed before sending to frontend)

- **Implementation Details**

  Frontend Code Example

```
<!DOCTYPE html>
<html>

<head>
  <title>
    HTML CSS JavaScript RSS news reader
  </title>
  <link rel="stylesheet"
    href="./sharmi.css">
</head>

<body>
  <div>
    <div class="headerbox">
      <div class="logo">
        <a href="https://www.geeksforgeeks.org/">
          <img src=
"https://media.geeksforgeeks.org/wp-
content/uploads/20231124125438/geeksimage2.png"
            alt="GeeksforGeeksLogo">
        </a>
        <div style="color:green; margin: 10px;
            font-size: 30px;">
          GeeksforGeeks
        </div>
      </div>
      <div style="text-align: center;
```

```
                font-size: 30px;
                padding: 10px;
                font-weight: 700;">
                Create RSS News feed Reader using
                HTML JavaScript CSS
        </div>

        <div style="background-color:rgb(32, 94, 170);
                overflow:hidden;
                text-align: center;">
            <ul style="text-align: center;">
              <li>
                <a class="aClass"
                  href=
"https://www.geeksforgeeks.org/"
                  target="_self">
                   Home
                </a>
              </li>
              <li>
                <a class="aClass" href=
"https://www.geeksforgeeks.org/html/html-tutorial/">
                   HTML Tutorial
                </a>
              </li>
              <li>
                <a class="aClass" href=
"https://www.geeksforgeeks.org/trending/">
                   Latest News
                </a>
              </li>
```

```
                <li>
                    <a class="aClass" href=
"https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-data-
structures-and-algorithms/">
                        Data Structures
                    </a>
                </li>
                <li>
                    <a class="aClass" href=
"https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-data-
structures-and-algorithms/">
                        Algorithms
                    </a>
                </li>
            </ul>
        </div>
    </div>
  </div>
  <div id="RSSfeedID"></div>
  <script src="sharmi.js"></script>
  <script>
    document
    .addEventListener("DOMContentLoaded", functionCall());
  </script>
</body>

</html>
```

- **Features**
    - Real-time stock updates
    - Auto-refresh mechanism
    - Search functionality for custom symbols
    - Color-coded price indicators
    - Responsive and user-friendly UI
    - API-based live data integration

- Future Enhancements

    - Integration with WebSocket for instant updates
    - Adding live **stock charts and graphs**
    - Login and personalized **watchlist** feature
    - Email or SMS alerts for price thresholds
    - AI-based trend prediction module

- **Applications**

    - Real-time financial monitoring
    - Educational purposes for students learning APIs and data visualization
    - Portfolio tracking for small investors
    - Integration with mobile apps or dashboards
    -

- Screenshots/API Documentation
    - **API Documentation**

      API Name: Alpha Vantage (Example)

      Base URL:

      https://www.alphavantage.co/query

Authentication:

Requires a free API key (generated after sign-up on https://www.alphavantage.co

- **Endpoint Used:**

To fetch the **latest stock price**: https://www.alphavantage.co/query?function=GLOBAL_QUOTE&symbol=TSLA&apikey=YOUR_API_KEY

- Method:
  `GET`

Request Parameters:

Parameter	Type	Description	Example
function	string	Type of API function to call	`GLOBAL_QUOTE`
symbol	string	Stock ticker symbol	`AAPL`, `TSLA`, `GOOG`
apikey	string	Your personal API key	`demo`

- **Sample API Response (JSON):**


```
{
  "Global Quote": {
    "01. symbol": "AAPL",
    "02. open": "178.20",
    "03. high": "179.00",
    "04. low": "176.50",
    "05. price": "178.95",
    "06. volume": "56324821",
```

```
    "07. latest trading day": "2025-10-25",
    "08. previous close": "177.30",
    "09. change": "1.65",
    "10. change percent": "0.93%"
  }
}
```


- Challenges and Solutions

Developing the *Real-Time Stock Ticker Application* involved several technical and practical challenges.
The table below summarizes the main difficulties encountered and the solutions implemented to overcome them.

S.No	Challenge	Description	Solution / Approach Used
1	**Real-Time Data Fetching**	Retrieving live stock data from external APIs in real time without overloading the system.	Used the **Alpha Vantage API** with controlled fetch intervals (setInterval() every 5 seconds) to balance real-time updates and API limits.
2	**API Rate Limits**	Free APIs restrict the number of requests per minute, causing	Implemented **time delay between requests** and allowed users to fetch

		temporary data fetching failures.	data for selected stocks only to reduce API calls.
3	**CORS (Cross-Origin Resource Sharing) Errors**	Some APIs block direct client-side calls from browsers.	Solved using a **proxy server** or **Node.js backend** to forward requests and bypass CORS issues safely.
4	**Slow Internet / API Response Delay**	Real-time data loading may appear slow during network lag.	Added **loading indicators** and **error handling** messages to inform the user while data is being fetched.
5	**Data Accuracy and Synchronization**	API data may not always sync perfectly due to network latency.	Added **timestamp display** with every update to show last refresh time, ensuring clarity for users.
6	**UI Responsiveness**	Display must adjust to desktop and mobile devices.	Used **CSS Flexbox** and **media queries** to create a **responsive layout** that adapts to screen size.
7	**Color Change for**	Visually representing increase/decrease	Implemented a **conditional CSS color system**

	Price Movement	in stock value dynamically.	(green for up, red for down) using JavaScript DOM manipulation.
8	**API Key Security**	Keeping the private API key secure from unauthorized use.	Stored the **API key in a configuration file** (or backend) instead of exposing it directly in client-side code.
9	**Continuous Refresh without Reload**	Keeping data updated without refreshing the webpage.	Used **AJAX / Fetch API** with asynchronous JavaScript (async/await) to update data smoothly.
10	**Maintaining UI Clarity**	Displaying multiple stocks without cluttering the screen.	Used **tabular or card layout** to present each stock neatly with symbol, price, and change %.

- GitHub README & Setup Guide

  Real-Time Stock Ticker Application

  - **Overview**

    The **Real-Time Stock Ticker** is a web-based application that displays **live stock market prices** using financial data APIs.
    It allows users to search for stock symbols (e.g., *AAPL*, *TSLA*, *GOOG*) and view current prices, changes, and trends — updating automatically without refreshing the page.

  - **Features**
    - Fetches live stock data from public APIs (e.g., Alpha Vantage / Finnhub)
    - Auto-refreshes every few seconds using JavaScript
    - Search bar for custom stock symbols
    - Color indicators — green for gain, red for loss
    - Responsive design (desktop & mobile friendly)
    - Lightweight and simple UI for quick viewing
  - **Setup Guide**

    Step 1 — Clone the Repository

    git clone https://github.com/<your-username>/real-time-stock-ticker.git

    cd real-time-stock-ticker

  - **Step 2 — Open in VS Code**
    - **Open the folder in Visual Studio Code**
    - **Ensure you have the Live Server extension installed**

- **Step 3 — Get a Free API Key**
  - **Go to https://www.alphavantage.co/support/#api-key**
  - Sign up and get your free API key
  - Copy it (you'll paste it inside `script.js`)
- **Step 4 — Configure script.js**

  In your script.js file, update the apiKey variable:

  ```
  const apiKey = "YOUR_API_KEY";
  ```
- Step 5 — Run the Project
  - Right-click `index.html`
  - Click **"Open with Live Server"**
  - The app will open in your browser
  - Enter a stock symbol (e.g., AAPL) and click **"Get Price"**
- **Sample Output**

  Real-Time Stock Tracker

| Symbol | Current Price | Change | Time |

| AAPL | $178.32 | +0.85% | 14:10|

- Final Submission(Repository + Deployed Link)
  - **Scope of the Project**

    The project focuses on **real-time price tracking** using **financial APIs**. It's ideal for:
    - Students studying finance or programming
    - Investors monitoring stock prices

- Developers learning about API integration and asynchronous JavaScript

- **System Design**

Architecture Flow:

User → Web Interface → API Request → Stock API Server → JSON Response → Live Data Display

Modules:

1. **Search Module: User enters stock symbol (e.g., AAPL, TSLA).**
2. **Fetch Module: JavaScript calls the API and retrieves data.**
3. **Display Module: The result (price, change %, time) is shown dynamically.**
4. **Auto-Refresh Module: Uses setInterval() to update every few seconds.**

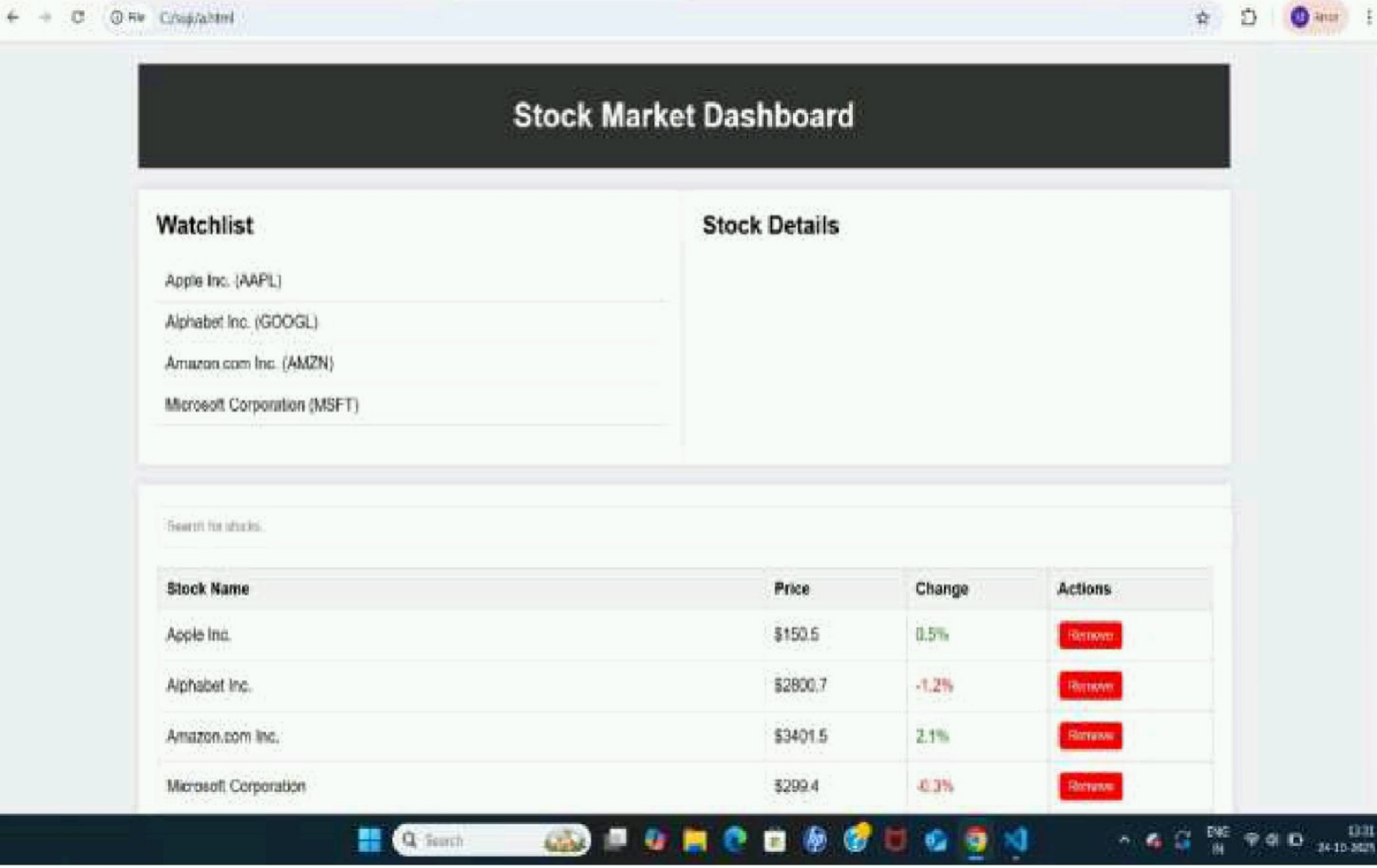
- **Code:**

```
<!DOCTYPE html>
<html>

<head>
   <title>
      HTML CSS JavaScript RSS news reader
   </title>
   <link rel="stylesheet"
       href="./sharmi.css">
</head>

<body>
   <div>
```

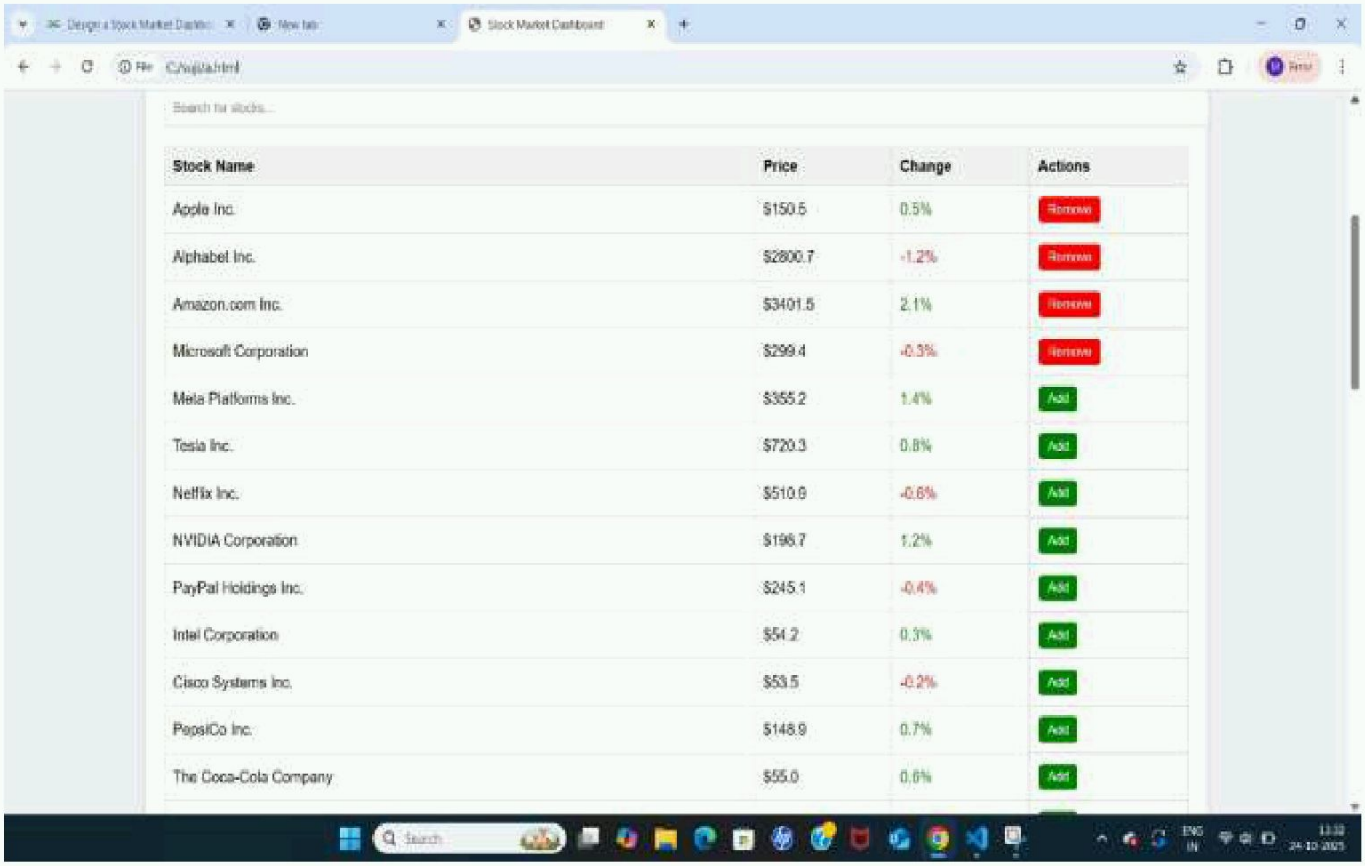
```
        <div class="headerbox">
            <div class="logo">
                <a href="https://www.geeksforgeeks.org/">
                    <img src=
"https://media.geeksforgeeks.org/wp-
content/uploads/20231124125438/geeksimage2.png"
                         alt="GeeksforGeeksLogo">
                </a>
                <div style="color:green; margin: 10px;
                            font-size: 30px;">
                    GeeksforGeeks
                </div>
            </div>
            <div style="text-align: center;
                        font-size: 30px;
                        padding: 10px;
                        font-weight: 700;">
                        Create RSS News feed Reader using
                        HTML JavaScript CSS
            </div>

            <div style="background-color:rgb(32, 94, 170);
                        overflow:hidden;
                        text-align: center;">
                <ul style="text-align: center;">
                    <li>
                        <a class="aClass"
                           href=
"https://www.geeksforgeeks.org/"
                           target="_self">
```

```
                Home
            </a>
        </li>
        <li>
            <a class="aClass" href=
"https://www.geeksforgeeks.org/html/html-tutorial/">
                HTML Tutorial
            </a>
        </li>
        <li>
            <a class="aClass" href=
"https://www.geeksforgeeks.org/trending/">
                Latest News
            </a>
        </li>

        <li>
            <a class="aClass" href=
"https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-
data-structures-and-algorithms/">
                Data Structures
            </a>
        </li>
        <li>
            <a class="aClass" href=
"https://www.geeksforgeeks.org/dsa/dsa-tutorial-learn-
data-structures-and-algorithms/">
                Algorithms
            </a>
        </li>
```

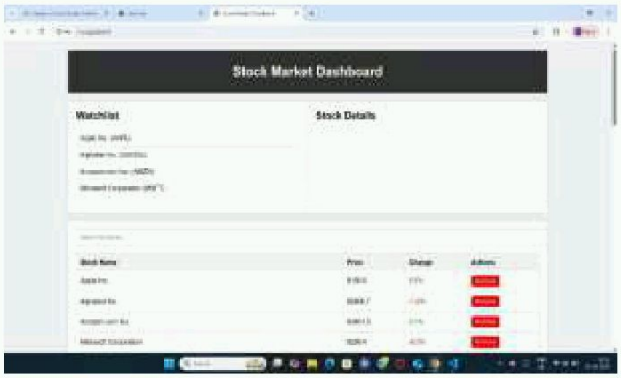
```
            </ul>
          </div>
        </div>
    </div>
    <div id="RSSfeedID"></div>
    <script src="sharmi.js"></script>
    <script>
        document
        .addEventListener("DOMContentLoaded",
functionCall());
    </script>
</body>

</html>
```

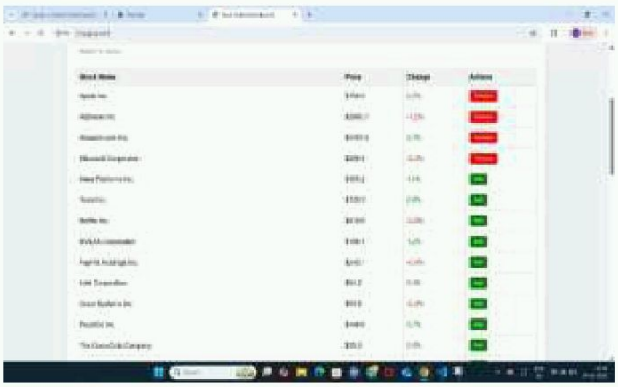

Stock Market Dashboard
Watchlist
Apple Inc. (AAPL)
Alphabet Inc. (GOOGL)
Amazon.com Inc. (AMZN)
Microsoft Corporation (MSFT)
Stock Details
Search for stocks...
Stock Name
Price
Change
Actions
Apple Inc.
$150.5
0.5%
Remove
Alphabet Inc.
$2800.7
-1.2%
Remove
Amazon.com Inc.
$3401.5
2.1%
Remove
Microsoft Corporation
$299.4
-0.3%
Remove

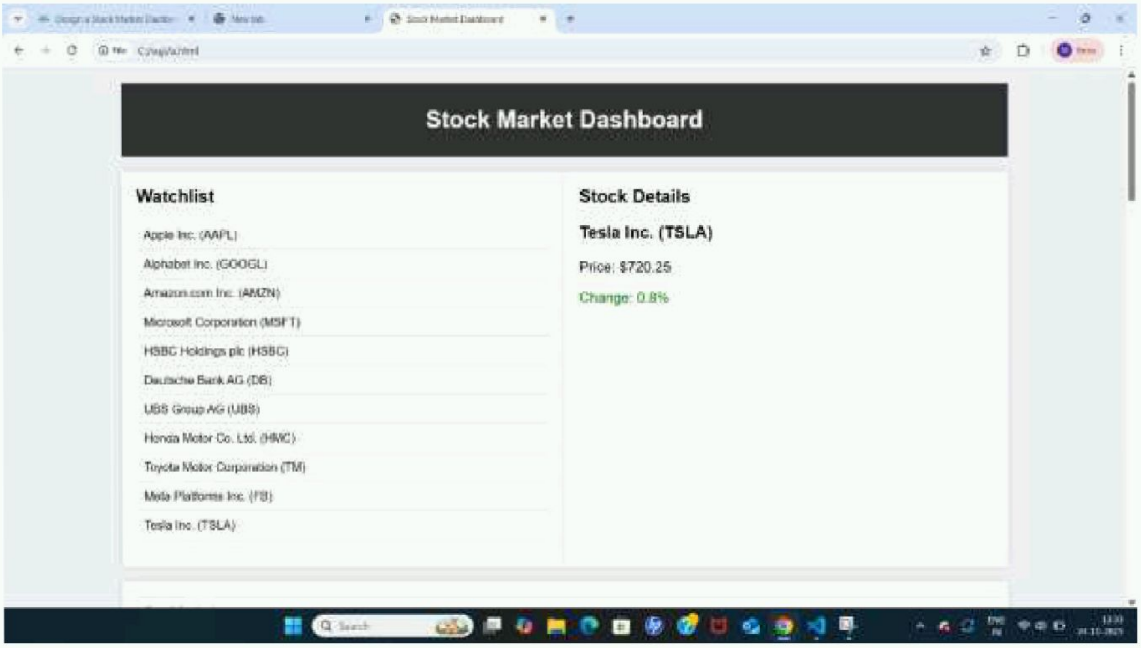
2. Stock Card Component



3.Responsive Layout:







GitHub Repository:

https://github.com/aarokkiamahi/Real-time-stock-ticker-.git