Stock Market APP

UI/UX Design Fundamentals - Christ University

Submitted By:

Team Members:-

Full Name & Roll Number:

ALLAN PREM VARGHESE-2462030
ABEL FRANCIS-2462005

College Email id:

allan.prem@btech.christuniversity.in
abel.francis@btech.christuniversity.in

Instructor Name:

Ms. Nagaveena

Index:

1. Abstract

This project is a dynamic web application designed to simulate a basic stock market platform. It features an interactive home page with real-time stock listings, search and filter capabilities, and a detailed stock page with a price chart. The application is built using HTML, CSS, and JavaScript, leveraging popular front-end libraries for a responsive and modern user experience.

2. Objectives

The primary objectives for this project were to:

- Build a functional web-based application that displays stock data.
- Implement features for searching and filtering stock listings.
- Simulate real-time price updates using JavaScript.
- Create a separate details page for each stock, including a price chart.
- Utilize front-end frameworks and libraries to streamline development and enhance the user interface.

3. Scope of the Project

This project is a client-side web application that focuses on the front-end user experience. Its scope includes:

- Displaying a list of mock stocks on the home page (index.html).

- Functionality to search for stocks by company name or ticker and filter by sector.
- Navigating from the home page to a specific stock's details page (details.html).
- Displaying dynamic stock information and a basic price chart on the details page.
- Simulating real-time price changes for all stocks.

The project does not include a backend database, user authentication, or any form of real-time data from a live stock market API.

4. Tools & Technologies Used

Tool/Technology	Purpose
HTML5	Used for the core structure of both web pages.
CSS3	Provides styling to enhance the visual appearance of the site.
JavaScript (ES6)	The primary language for all application logic, including data management, UI updates, and event handling.
jQuery	A library to simplify DOM manipulation and event handling.
Bootstrap 5.3.2	A CSS framework used for responsive design and pre-built components like the navigation bar and tables.
Chart.js	A JavaScript library for creating the interactive price chart on the details page.

5. HTML Structure Overview

The project's HTML files are built to be responsive and modular, using Bootstrap components for a clean layout.

- **index.html**: Contains a navigation bar and a main container for the stock listings. It includes a search input and a sector filter in a responsive row layout, followed by a table to display stock data. The body loads script.js at the end.
- **details.html**: This page also has a navigation bar and a main container. It provides dedicated elements to display a single stock's name, ticker, sector, and price. A canvas element is used as a container for the price chart. It also loads script.js at the end.

6. CSS Styling Strategy

The custom styling for this project is minimal, as it primarily leverages the Bootstrap framework.

- **Body Styling**: A light gray background (#f8f9fa) is set for the entire body.
- **Table Hover Effect**: A custom rule for the table rows (.table-hover tbody tr:hover) changes the background to a light blue (#eaf2ff) and the cursor to a pointer when a user hovers over a row. This provides a clear visual cue that the rows are clickable.

7. Key Features

- **Real-time Price Simulation**: The script.js file includes a setInterval function that updates the price of each stock every 5 seconds, simulating live market activity.
- **Search and Filter Functionality**: Users can dynamically search for a stock by typing in the search bar or filter the table by selecting a sector from the dropdown menu.
- **Dynamic UI Updates**: The loadStocks() and loadDetails() functions in JavaScript handle rendering the stock data to the HTML. They are called after any change (e.g., search, filter, price update) to ensure the UI is always in sync with the data.
- **Interactive Chart**: The details.html page uses Chart.js to render a line chart that visualizes the mock price history for a selected stock.
- **Client-Side Navigation**: Clicking a stock on the home page table triggers a JavaScript function that navigates the user to details.html, passing the stock's ticker as a URL query parameter to dynamically load the correct information.

8. Project Files and Code

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Stock Market Website</title>

  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">

  <link rel="stylesheet" href="style.css">

  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body>

  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

    <div class="container-fluid">

      <a class="navbar-brand" href="index.html">📈 Stock Market</a>

    </div>

  </nav>

  <div class="container my-4">

    <h2 class="mb-3">Stock Listings</h2>

    <div class="row mb-3">

      <div class="col-md-6">

        <input type="text" id="searchInput" class="form-control" placeholder="Search by
company or ticker">

      </div>

      <div class="col-md-6">

        <select id="sectorFilter" class="form-select">

          <option value="">All Sectors</option>

          <option value="Tech">Tech</option>

          <option value="Finance">Finance</option>

          <option value="Energy">Energy</option>

        </select>

      </div>
    </div>
    <table class="table table-striped table-hover">
      <thead class="table-dark">
        <tr>
          <th>Company</th>
          <th>Ticker</th>
          <th>Sector</th>
          <th>Price</th>
        </tr>
      </thead>
      <tbody id="stockTableBody"></tbody>
    </table>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

details.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>Stock Details</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
  <link rel="stylesheet" href="style.css">
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <a class="navbar-brand" href="index.html">📈 Stock Market</a>
    </div>
  </nav>
  <div class="container my-4">
    <h2 id="stockName"></h2>
    <p><strong>Ticker:</strong> <span id="stockTicker"></span></p>
    <p><strong>Sector:</strong> <span id="stockSector"></span></p>
    <p><strong>Current Price:</strong> $<span id="stockPrice"></span></p>
    <div class="card p-3">
      <canvas id="priceChart" height="100"></canvas>
    </div>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

script.js

```
let stocks = [
  { name: "Apple Inc", ticker: "AAPL", sector: "Tech", price: 172 },
  { name: "Microsoft Corp", ticker: "MSFT", sector: "Tech", price: 315 },
  { name: "Goldman Sachs", ticker: "GS", sector: "Finance", price: 365 },
  { name: "Tesla Inc", ticker: "TSLA", sector: "Energy", price: 255 },
  { name: "Reliance Energy", ticker: "REL", sector: "Energy", price: 90 }
];
function loadStocks() {
  if ($("#stockTableBody").length) {
    let search = $("#searchInput").val().toLowerCase();
    let sector = $("#sectorFilter").val();
    $("#stockTableBody").empty();
    stocks.forEach(stock => {
      if (
        (stock.name.toLowerCase().includes(search) ||
         stock.ticker.toLowerCase().includes(search)) &&
        (sector === "" || stock.sector === sector)
      ) {
        let row = `
          <tr onclick="viewDetails('${stock.ticker}')">
            <td>${stock.name}</td>
            <td>${stock.ticker}</td>
            <td>${stock.sector}</td>
            <td>$${stock.price}</td>
          </tr>
        `;
        $("#stockTableBody").append(row);
      }
    });
  }
}
function viewDetails(ticker) {
  window.location.href = `details.html?ticker=${ticker}`;
}
function loadDetails() {
  if ($("#stockName").length) {
    let params = new URLSearchParams(window.location.search);
    let ticker = params.get("ticker");
    let stock = stocks.find(s => s.ticker === ticker);
    if (stock) {
      $("#stockName").text(stock.name);
      $("#stockTicker").text(stock.ticker);
      $("#stockSector").text(stock.sector);
      $("#stockPrice").text(stock.price);
      let ctx = document.getElementById("priceChart").getContext("2d");
      new Chart(ctx, {
        type: "line",
        data: {
          labels: ["Mon", "Tue", "Wed", "Thu", "Fri"],
          datasets: [{
            label: "Price ($)",
            data: [
              stock.price - 5,
              stock.price - 3,
             stock.price,
             stock.price + 2,
             stock.price
            ],
            borderColor: "blue",
            fill: false
          }]
        }
      });
    }
  }
}
$(document).ready(function () {
  loadStocks();
  loadDetails();
  $("#searchInput, #sectorFilter").on("input change", loadStocks);
  setInterval(() => {
    stocks.forEach(s => {
      s.price += (Math.random() * 4 - 2).toFixed(2) * 1;
    });
    loadStocks();
    loadDetails();
  }, 5000);
});
```

style.css

```
body {
  background: #f8f9fa;
}
.table-hover tbody tr:hover {
  background: #eaf2ff;
  cursor: pointer;
}
```

9. Conclusion

This project successfully demonstrates the creation of a dynamic, front-end web application that simulates a stock market interface. By combining a clear HTML structure with JavaScript's real-time capabilities and the powerful styling of Bootstrap, it achieves all its core objectives. The modular design, with separate files for HTML, CSS, and JavaScript, makes the code clean and maintainable. This project serves as a practical example of fundamental front-end development principles.