Octa Byte AI Pvt Ltd

CIN: U62099KA2025PTC201620
Registered office address: WeWork Prestige Cube, Site no :26, Laskar Housur Rd, Adugodi, Bangalore South, Bangalore -560030, Karnataka

---

Case Study: Dynamic Portfolio Dashboard with React.Js, TypeScript, Tailwind & Node.Js

---

1. Introduction

Context

Modern investors need real-time insights into their portfolio performance to make informed decisions—whether to buy, sell, hold, or add to positions. This case study challenges you to build a dynamic web application that displays portfolio information, fetching live data from financial APIs.

Goal

Develop a portfolio dashboard using ReactJs/**Next.js** that retrieves stock data from:

- **Yahoo Finance** (for Current Market Price - CMP)
- **Google Finance** (for P/E Ratio and Latest Earnings)

The dashboard should be interactive and visually appealing.

Target Audience

This project assesses your ability to:

- Build a full-stack web application with React.js and Node JS
- Consume and process data from external APIs
- Handle asynchronous operations and data transformations
- Design a user-friendly interface for displaying financial data

---

2. Requirements

Data Sources

- **Yahoo Finance API**
  Used to fetch real-time stock prices (CMP). Note: Yahoo Finance does not have a public official API. Candidates should acknowledge this and propose solutions (e.g., scraping or unofficial libraries).
- **Google Finance API**
  Used to fetch P/E Ratio and latest earnings data. Similar to Yahoo Finance, scraping or unofficial libraries are required.

Data Format: Use a structured format such as JSON to store and manipulate data.

Octa Byte AI Pvt Ltd

CIN: U62099KA2025PTC201620
Registered office address: WeWork Prestige Cube, Site no :26, Laskar Housur Rd, Adugodi, Bangalore South, Bangalore -560030, Karnataka

Functionality

Portfolio Table

Display holdings in a tabular format with the following columns (go through the excel sheet):

- **Particulars** (Stock Name)
- **Purchase Price**
- **Quantity (Qty)**
- **Investment** (Purchase Price × Qty)
- **Portfolio (%)** (Proportional weight in the portfolio)
- **NSE/BSE** (Stock Exchange Code)
- **CMP** (Fetched from Yahoo Finance)
- **Present Value** (CMP × Qty)
- **Gain/Loss** (Present Value – Investment)
- **P/E Ratio** (Fetched from Google Finance)
- **Latest Earnings** (Fetched from Google Finance)

Dynamic Updates

CMP, Present Value, and Gain/Loss should update automatically at regular intervals (e.g., every 15 seconds).

Visual Indicators

Color-code Gain/Loss:

- **Green** for gains
- **Red** for losses

Sector Grouping

Group stocks by sector (e.g., Financials, Technology) with sector-level summaries:

- Total Investment
- Total Present Value
- Gain/Loss

Technology Stack

- **Frontend**: Next.js (React framework)
- **Backend** :NodeJs

- **Styling**: Tailwind CSS, Typescript
- **Data Fetching**: fetch, Axios, or similar

Recommended Libraries:

- react-table – Table display
- recharts – Optional charting for visualizations

---

3. Technical Challenges and Considerations

API Limitations

- **Unofficial APIs / Scraping**:
  Yahoo and Google Finance require scraping or use of unofficial libraries that may break due to site changes.
- **Rate Limiting**:
  Public sources may have rate limits. Use caching, throttling, or batching to prevent blocks.
- **Data Accuracy**:
  Scraped or unofficial data may vary in accuracy. Add disclaimers or verification logic if needed.

Asynchronous Operations

- Use async/await or Promises to handle parallel API requests.

Data Transformation

- Clean and format the raw API data to match the required table schema.

Performance Optimization

- **Caching**: Use Next.js caching or a dedicated tool to reduce API calls
- **Memoization**: Use React.memo or similar for preventing unnecessary renders

Error Handling

- Handle API failures gracefully
- Display clear error messages for users

Security

- Do not expose API keys or sensitive data in client-side code

Real-Time Updates

- Use setInterval for periodic refresh
- Optional: Use WebSockets for more advanced, efficient updates

Responsiveness

- Ensure dashboard layout adapts well across devices

---

4. Implementation Steps

1. **Set Up Next.js Project**
   Use create-next-app to scaffold the project
2. **Design Data Model**
   Define structure for individual stocks and the portfolio
3. **API Integration**
   Write fetch functions for Yahoo and Google Finance data
4. **Create Portfolio Table Component**
   Use react-table to render the portfolio
5. **Implement Dynamic Updates**
   Use setInterval for live CMP and value refresh
6. **Add Sector Grouping**
   Group by sector and show sector-wise summaries
7. **Add Visual Indicators**
   Use CSS for green/red coloring of Gain/Loss
8. **Implement Error Handling**
   Catch and display errors during data fetch or transformation
9. **Optimize Performance**
   Add caching and memoization techniques
10. **Deploy (Optional)**
    Deploy using Vercel, Netlify, or similar platforms

---

5. Evaluation Criteria

Solutions will be evaluated based on:

- **Functionality**: Does it meet the defined requirements?
- **Code Quality**: Is the code clean and maintainable?
- **Performance**: Is the dashboard fast and responsive?
- **Error Handling**: Are failures handled smoothly?

- **API Strategy**: How are scraping/rate limits managed?
- **User Interface**: Is the UI intuitive and visually appealing?
- **Problem Solving**: Are the technical challenges addressed effectively?

---

6. Deliverables

- **Source Code**: Full Next.js application
- **README**: Setup and usage instructions
- **Technical Document**:
  A short write-up explaining key challenges faced and your solutions

---

Please do not use AI generated code since you will need to explain the entire code in interview. Learn and understand technology while doing the assignment.