**Stock Dashboard - Report**

**Backend (server.js)**

// Load environment variables from .env file

require("dotenv").config();

const express = require("express");

const http = require("http");

const socketIo = require("socket.io");

const cors = require("cors");

const axios = require("axios");

const app = express();

const server = http.createServer(app);

const io = socketIo(server, { cors: { origin: "\*" } });

app.use(cors());

// Use an environment variable for PORT

const PORT = process.env.PORT || 5000;

// Function to fetch real stock prices (Replace with actual API)

const fetchStockPrice = async (symbol) => {

try {

// Replace this with a real stock API (e.g., Alpha Vantage, Yahoo Finance)

const response = await axios.get(`https://api.example.com/stock/${symbol}`);

return response.data.price;

} catch (error) {

console.error("Error fetching stock price:", error);

return null;

}

};

// Handle WebSocket Connections

io.on("connection", (socket) => {

console.log("Client connected");

// Listen for stock subscription requests from the frontend

socket.on("subscribeToStock", async (symbol) => {

console.log(`Subscribed to stock: ${symbol}`);

// Fetch initial stock price

const price = await fetchStockPrice(symbol);

socket.emit("stockPriceUpdate", { symbol, price });

// Send updated stock prices every 5 seconds

setInterval(async () => {

const updatedPrice = await fetchStockPrice(symbol);

socket.emit("stockPriceUpdate", { symbol, price: updatedPrice });

}, 5000);

});

socket.on("disconnect", () => {

console.log("Client disconnected");

});

});

// Start the Server

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

**Frontend (Dashboard.js)**

import React, { useState, useEffect } from "react";

import axios from "axios";

import { io } from "socket.io-client";

import StockInput from "./StockInput";

import StockList from "./StockList";

import ThemeToggle from "./ThemeToggle";

import "bootstrap/dist/css/bootstrap.min.css";

const socket = io("http://localhost:5000", { transports: ["websocket"] });

const Dashboard = () => {

const [stock, setStock] = useState(""); // Stock input value

const [stockPrice, setStockPrice] = useState(null); // Latest stock price

const [previousSearches, setPreviousSearches] = useState([]); // Search history

const [error, setError] = useState(null); // Error state

// Fetch Stock Price from API

const fetchStockPrice = async () => {

if (!stock) return;

try {

setError(null); // Reset error before fetching

const API\_KEY = process.env.REACT\_APP\_API\_KEY; // Get API Key from .env

const response = await axios.get(

`https://www.alphavantage.co/query?function=GLOBAL\_QUOTE&symbol=${stock}&apikey=${API\_KEY}`

);

// Extract price from API response

const stockData = response.data["Global Quote"];

if (stockData && stockData["05. price"]) {

setStockPrice(stockData["05. price"]);

} else {

setError("Invalid API response. Please try again.");

console.error("Invalid API response", response.data);

}

// Store the stock in previous searches

setPreviousSearches((prev) => [...prev, stock]);

} catch (error) {

console.error("Error fetching stock data", error);

setError("Network error. Please check your internet or API key.");

}

};

// WebSocket Handling (ComponentDidMount Equivalent)

useEffect(() => {

console.log("Component Mounted: Connecting to WebSocket...");

socket.on("connect", () => {

console.log("WebSocket Connected ");

});

socket.on("stockPriceUpdate", (data) => {

console.log("Stock Update Received:", data);

if (data.symbol === stock) {

setStockPrice(data.price);

}

});

return () => {

console.log("Component Unmounted: Disconnecting WebSocket...");

socket.disconnect();

};

}, []);

return (

<div className="container mt-4">

<h2 className="text-center">📈 Real-Time Stock Dashboard</h2>

<ThemeToggle />

<div className="input-group mb-3">

<input

type="text"

className="form-control"

value={stock}

onChange={(e) => setStock(e.target.value)}

placeholder="Enter Stock Symbol (e.g., AAPL)"

/>

<button className="btn btn-primary" onClick={fetchStockPrice}>

Get Price

</button>

</div>

{error && <div className="alert alert-danger">{error}</div>}

{stockPrice !== null && (

<div className="alert alert-info">

Current Price: <strong>${stockPrice}</strong>

</div>

)}

<h4>Previous Searches:</h4>

<ul className="list-group">

{previousSearches.map((s, index) => (

<li key={index} className="list-group-item">{s}</li>

))}

</ul>

</div>

);

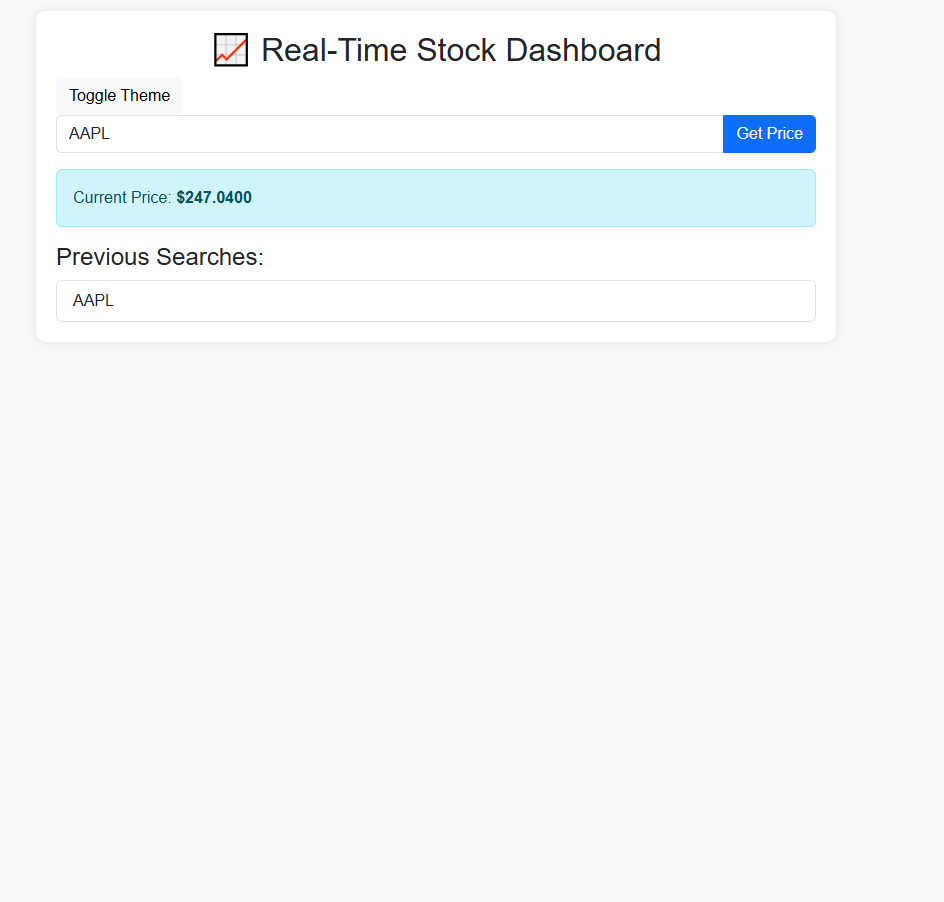
};

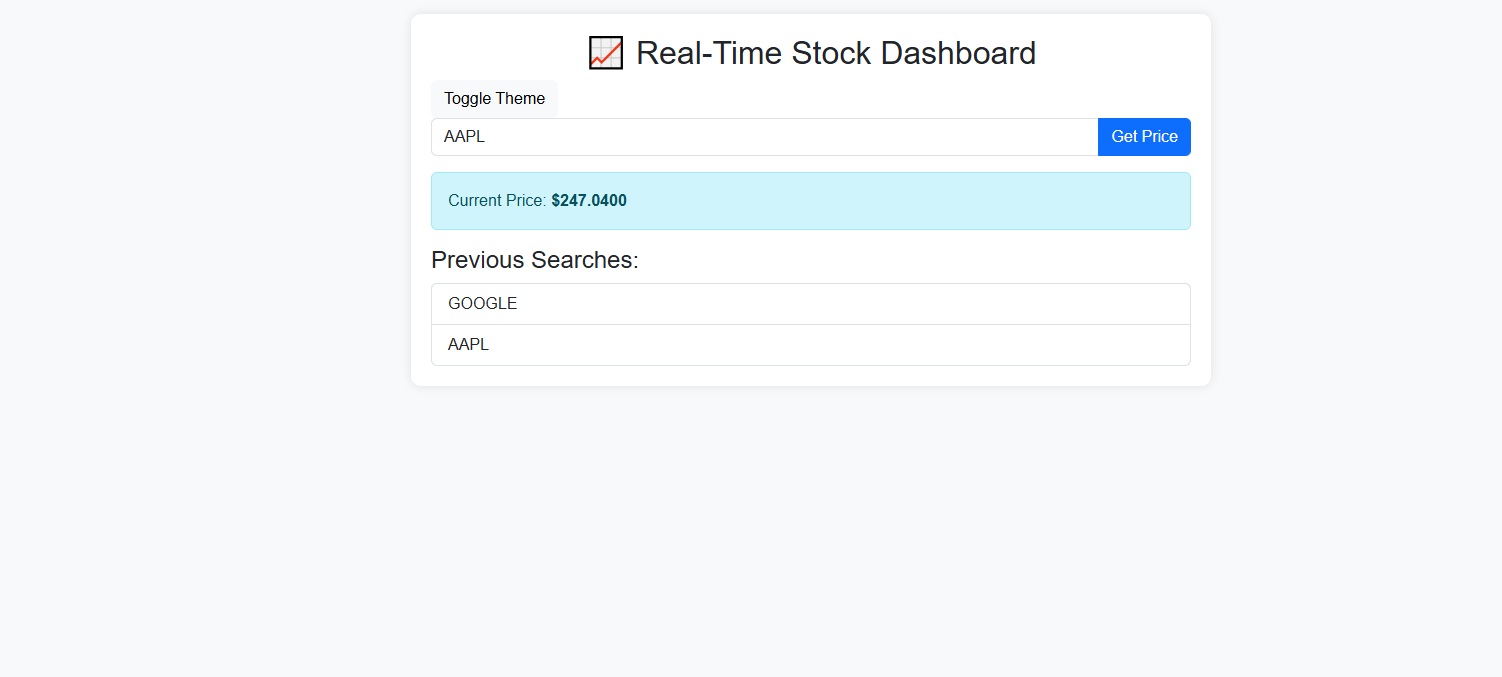
export default Dashboard;

**Next Steps**

1. Ensure your .env files are set up correctly
2. Run the backend:
3. cd backend
4. node server.js
5. Run the frontend:
6. cd frontend
7. npm start
8. Test the real-time stock updates!

**Output:**

****

****