# Stock Market Application Development Challenge: Reference Document

## 1. Introduction

### Overview:

This assignment involves developing a comprehensive stock market application using the MERN (MongoDB, Express.js, React, Node.js) stack or any other suitable technology stack. The application should provide real-time and historical stock market data by integrating with APIs like Alpha Vantage, Novadax, or any other suitable API. The goal is to create an intuitive, responsive, and feature-rich application focusing on user experience and data accuracy.

### Objectives:

- Fetch and display real-time and historical stock data.

- Design a user-friendly and visually appealing interface.

- Implement stock data visualization using TradingView charts.

- Enable user authentication and a watchlist feature.

- Provide functionalities for data sorting, filtering, portfolio management, trading, alerts, and notifications.

## 2. Step-by-Step Instructions

### Step 1: Setting Up the Development Environment

1. **Install Node.js and npm**:

- Download and install Node.js from [Node.js official website](https://nodejs.org/).

- Verify the installations by running `node -v` and `npm -v` in your terminal.

2. **Set Up Your Project Directory**:

- Create a new directory for the project.

- Navigate to the directory and initialize a new project: `npm init -y`.

3. **Install Dependencies**:

- Backend: `npm install express mongoose cors dotenv`

- Frontend: `npx create-react-app client`

- Additional Dependencies: `npm install axios redux react-redux redux-thunk socket.io socket.io-client jsonwebtoken bcryptjs passport passport-google-oauth20 passport-facebook`

### Step 2: Backend Development

1. **Create Express Server**:

- Inside your project directory, create an `index.js` file.

- Set up an Express server in `index.js`.

- Configure the server to use `cors` and `dotenv` for environment variables.

2. **Database Setup**:

- Connect to MongoDB using `mongoose`.

- Define schemas and models for User, Stock, Watchlist, and Portfolio items.

3. **API Integration**:

- Create routes for fetching real-time and historical data.

- Integrate with APIs (e.g., Alpha Vantage) to fetch data.

4. **User Authentication**:

- Implement social authentication using Passport.js (Google, Facebook).

- Set up routes for login, logout, and user sessions.

5. **WebSocket Setup**:

- Use Socket.io to establish WebSocket connections for real-time data updates.

### Step 3: Frontend Development

1. **Set Up React Project**:

- Navigate to the `client` directory and set up your React application.

- Install necessary UI libraries: `npm install @material-ui/core ant-design chakra-ui react-bootstrap`.

2. **Design User Interface**:

- Create main components: Navbar, Home, Dashboard, StockDetails, Watchlist, Portfolio.

- Use Material-UI, Ant Design, Chakra UI, or React-Bootstrap to design the interface.

3. **Fetching and Displaying Data**:

- Use Axios to fetch data from the backend and display it.

- Display real-time stock prices using Socket.io connections.

4. **Charts Integration**:

- Integrate TradingView charts for data visualization.

- Display different chart types (e.g., line charts, bar charts, candlestick charts).

5. **User Authentication**:

- Implement authentication flows using React-Redux for state management and Passport.js for backend integration.

6. **Watchlist Feature**:

- Allow users to add stocks to their watchlist.

- Provide real-time updates and notifications for watchlist stocks.

7. **Portfolio Management**:

- Enable users to manage their stock portfolio, including adding/removing stocks and viewing performance.

8. **Trading Functionality**:

- Implement buying and selling features for stocks.

- Provide real-time order status updates.

9. **Alerts and Notifications**:

- Allow users to set price alerts and send notifications via email or in-app when the stock price crosses a predefined threshold.

### Step 4: Best Practices

- **Code Quality**: Write clean, modular, and well-documented code. Use linters like ESLint and tools like SonarLint to ensure code quality.

- **Security**: Ensure secure authentication and authorization using JWT, passport.js, and bcrypt for password hashing.

- **Responsive Design**: Ensure your UI is responsive and accessible on different devices and screen sizes.

- **State Management**: Utilize Redux for predictable state management, especially for user sessions, stock data, and portfolio management.

- **Testing**: Write unit tests using Jest and integration tests to ensure your application functions as expected.

### Step 5: Submission Guidelines

- **Source Code**:

- Ensure the complete and well-documented source code is included.

- Upload the source code to a GitHub repository.

- **Technical Documentation**:

- Include detailed documentation explaining the architecture, design decisions, and the setup and run procedure for the application.

- **User Guide**:

- Provide instructions for users to navigate and use the application.

- **Deployment**:

- Deploy the application to a cloud platform (e.g., Heroku, AWS, Vercel).

- Provide the deployment link.

### Step 6: Frequently Asked Questions (FAQ)

**Q1: What technology stack should I use for this project?**

A1: Apart from the MERN stack, you can use any tech stack you are comfortable with to develop a full-fledged application.

**Q2: Which APIs can I integrate for fetching stock market data?**

A2: You can integrate APIs like Alpha Vantage, Novadax, or any other suitable API to fetch real-time and historical stock market data.

**Q3: How should I implement real-time data updates?**

A3: Use WebSocket connections for real-time data updates. Socket.io is a great library for enabling real-time, bidirectional communication in your application.

**Q4: What UI libraries are recommended for this project?**

A4: You can use Material-UI, Ant Design, Chakra UI, or React-Bootstrap to design a beautiful and intuitive user interface.

**Q5: How should I handle user authentication?**

A5: Implement social authentication using providers like Google and Facebook. Ensure user data and sessions are secured using appropriate authentication and authorization mechanisms.

**Q6: What charting library should I use to display stock data?**

A6: Use TradingView charts for displaying stock data. TradingView offers professional and fully customizable financial charts that are highly interactive.

**Q7: Can I add additional features beyond the specified requirements?**

A7: Yes, adding additional features is encouraged and will be considered during the evaluation for innovation and creativity.

**Q8: What should be included in the deliverables?**

A8: Your deliverables should include the complete and well-documented source code, technical documentation, a user guide, and a deployed version of the application on a cloud platform with the deployment link.

**Q9: How will the project be evaluated?**

A9: The project will be evaluated based on functionality, user interface quality, code quality, innovation, and successful deployment. Specific criteria include how well the application meets requirements, aesthetics and usability of the UI, code organization and documentation, additional features implemented, and ease of accessing the deployed application.

**Q10: Where can I deploy my application?**

A10: You can deploy your application on cloud platforms like Heroku, AWS, or Vercel. Ensure the application is accessible online and provide the deployment link.

**Q11: Can I use other libraries or tools not mentioned in the requirements?**

A11: Yes, you can use other libraries or tools as long as they help achieve the project's objectives and improve the application's functionality and user experience.

**Q12: What is the submission deadline for the project?**

A12: Please submit your project by the end of the week from the announcement date. Ensure all deliverables are included and the application is deployed and accessible online by this date.

**Q13: How should I handle portfolio management features?**

A13: Implement features to allow users to create and manage a portfolio of stocks and options. Users should be able to add stocks and options to their portfolio, specify quantities, view the total value of their portfolio, and track unrealized gains/losses.

**Q14: What should I include in the technical documentation?**

A14: The technical documentation should detail the application's architecture, design decisions, and steps to set up and run the application. It should provide clear instructions for developers to understand and work with your codebase.

**Q15: How do I implement alerts and notifications?**

A15: Implement a feature where users can set price alerts for specific stocks. Send notifications via email or in-app when the stock price crosses a predefined threshold. Use relevant libraries and APIs to facilitate this functionality.

**Q16: Do I need to implement all the features mentioned above?**

A16: Implement as many features as you can within the available time, but ensure that each implementation is reliable and of high quality.

**Q17: Whom should I contact for additional queries?**

A17: For any additional queries, please contact Vinay at vinay@quantumcona.com.

## Conclusion

By following this reference document, users should be able to develop a comprehensive stock market application that meets the assignment objectives. Be sure to adhere to best practices, maintain high code quality, and implement secure, responsive, and user-friendly functionalities. Good luck!