**Full Stack Development with MERN**

**API Development and Integration Report**

|  |  |
| --- | --- |
| Date |  |
| Team ID | PNT2022TMIDxxxxxx |
| Project Name | Project - xxx |
| Maximum Marks |  |

**Project Title:** Stock Trading Web Application  
**Date:** 19/07/24  
**Prepared by:** Adithya Hosur, Mukundan Sriram, Kirthick B Easwar, Anish

**Objective**  
The objective of this report is to document the API development progress and key aspects of the backend services implementation for the Stock Trading Web App project.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** JSONWebToken

**Project Structure**  
Provide a screenshot of the backend project structure with explanations for key directories and files.

**Key Directories and Files**

1. **/controllers**
   * Contains functions to handle requests and responses.
   * Integrated in the server.js file
2. **/models**
   * Includes Mongoose schemas and models for MongoDB collections.
   * Schema.js file contains the 4 schemas of MongoDB collections:
   * UserSchema
   * TransactionSchema
   * StockSchema
   * OrderSchema
3. **/routes**
   * Defines the API endpoints and links them to controller functions.
   * Integrated in the server.js file in the server directory.
4. **/middlewares**
   * Custom middleware functions for request processing.
   * Cross-Origin Resource Sharing (CORS)
   * It is a way to enable cross-domain communication in web applications.
5. **/config**
   * Configuration files for database connections, environment variables, etc.

**API Endpoints**  
A summary of the main API endpoints and their purposes:

**User Authentication**

* **POST /api/user/register** - Registers a new user. Checks if user already exists. Saves data in the MongoDB Collection. The password is hashed.
* **POST /api/user/login** - Authenticates a user and returns a token. Checks if the data is present in the MongoDB Collection and compares the data input.

**User Management**

* **GET /api/user/-** Retrieves user information by ID. Using the ID, we fetch the rest of user data like email, username and password.
* **PUT /api/user/**- Updates user information by ID.

**Transactions**

* **POST /api/deposit** – Finds the user and updates Transaction History with the current balance.
* **POST /api/withdraw** – Finds the user and updates Transaction History and balance.

**Buying/Selling Stocks**

* **POST /api/buyStocks** – Buys new Stocks.
* **POST /api/sellStocks** – Sells Stocks.

**Transactions and Users**

* **GET /api/transactions** - Retrieves all transactions from the MongoDB Collection.
* **GET /api/users** - Retrieves all users from the MongoDB Collection.

**Integration with Frontend**  
The backend communicates with the frontend via RESTful APIs. Key points of integration include:

* **User Authentication:** Tokens are passed between frontend and backend to handle authentication.
* **Data Fetching:** Frontend components make API calls to fetch necessary data for display and interaction.

**Error Handling and Validation**  
Describe the error handling strategy and validation mechanisms:

* **Error Handling:** Centralized error handling using middleware. Every function has a try catch block with specific error prompts in the console.
* **Validation:** Input validation.

**Security Considerations**  
Outline the security measures implemented:

* **Authentication:** Secure token-based authentication.
* **Data Encryption:** Encrypt sensitive data at rest and in transit. Using bcrypt, the passwords stored in MongoDB are hashed and safe from being spied upon by anyone who has the access to the Database.