**Full Stack Development with MERN**

**API Development and Integration Report**

| Date | 10 JUL 2024 |
| --- | --- |
| Team ID | SWTID1720004076 |
| Project Name | Wise Wallet: Your Budgeting Partner |
| Maximum Marks |  |

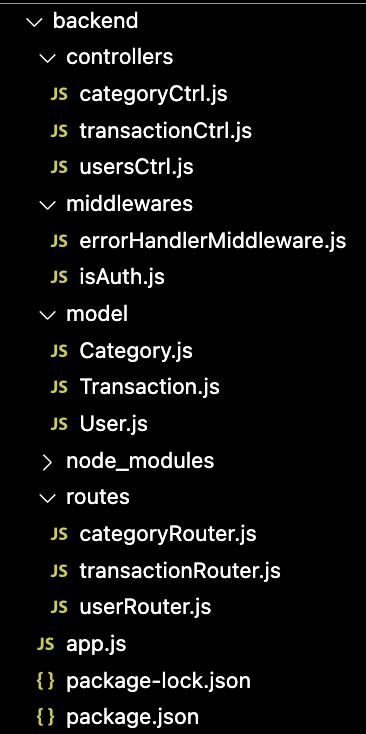
**Objective**

The objective of this report is to document the API development progress and key aspects of the backend services implementation for the Wise Wallet: Your Budgeting Partner project.

**Technologies Used**

* **Backend Framework:** Node.js with Express.js
* **Database:** MongoDB
* **Authentication:** JWT (JSON Web Tokens)

**Project Structure**

Project Directory:  


**Explanation:**

### **Key Directories and Files**

**controllers**

* **categoryCtrl.js:** Contains functions to handle category-related requests and responses.
* **transactionCtrl.js:** Manages transaction-related operations such as creating and fetching transactions.
* **usersCtrl.js:** Handles user-related actions including user registration, login, and profile management.

**middlewares**

* **errorHandlerMiddleware.js:** A centralized error handling middleware that captures and processes errors across the application.
* **isAuth.js:** Middleware to check if a user is authenticated by verifying the JWT token.

**models**

* **Category.js:** Defines the Mongoose schema and model for categories.
* **Transaction.js:** Defines the Mongoose schema and model for transactions.
* **User.js:** Defines the Mongoose schema and model for users, including methods for password hashing and comparison.

**routes**

* **categoryRouter.js:** Defines the API endpoints for category-related operations and links them to the respective controller functions.
* **transactionRouter.js:** Defines the API endpoints for transaction-related operations and links them to the respective controller functions.
* **userRouter.js:** Defines the API endpoints for user-related operations, such as registration and login, and links them to the respective controller functions.

**app.js**

* The main application file that sets up the Express server, connects to the MongoDB database, and initializes middleware and routes.

### **API Endpoints**

A summary of the main API endpoints and their purposes:

#### **User Authentication**

* **POST /api/users/register**: Registers a new user. Requires details such as username and password to create a new account.
* **POST /api/users/login**: Authenticates a user and returns a JWT token. Verifies user credentials and provides a token for accessing protected routes.

#### **User Management**

* **GET /api/users/**: Retrieves detailed information about a specific user by ID. Includes user profile data and other related details.
* **PUT /api/users/**: Updates user information by ID. Allows changes to user details like email and password.

#### **Category Management**

* **GET /api/categories**: Retrieves a list of all categories. Provides information on categories used for organizing transactions.
* **POST /api/categories**: Creates a new category. Requires details for the new category to be added to the system.

#### **Transaction Management**

* **GET /api/transactions**: Lists all recorded transactions. Provides data related to financial transactions.
* **POST /api/transactions**: Adds a new transaction. Requires details such as amount, date, and associated category.

### 

### **Integration with Frontend**

The backend integrates with the frontend through RESTful APIs to ensure smooth data exchange and user authentication. Key integration points include:

* **User Authentication**: Authentication tokens are issued to users upon login and used by the frontend to access protected routes and functionalities.
* **Data Fetching**: The frontend components make API calls to retrieve and display data, such as user profiles, categories, and transactions, ensuring a dynamic and interactive user experience.

### **Error Handling and Validation**

#### **Error Handling**

Errors are managed centrally through middleware. This approach captures any issues that arise during request processing and sends consistent error responses, facilitating easier debugging and user feedback.

#### **Validation**

Input validation is performed using libraries like express-validator to ensure that incoming data meets the required criteria before it is processed by the application logic.

### **Security Considerations**

#### **Authentication**

User authentication is secured using JWT (JSON Web Tokens). Tokens are generated upon successful login and used to authenticate requests, ensuring that only authorized users can access certain endpoints.

#### **Data Encryption**

Sensitive data is encrypted using industry-standard techniques both at rest and in transit. This involves encrypting user passwords and any other sensitive information to maintain privacy and security throughout the application.