**Wallet Management System Documentation**

This document provides an overview of the Wallet Management System, including its architecture, endpoints, and usage guidelines. The system is designed to facilitate user registration, authentication, wallet creation, fund transactions, and management of user wallets.

**Architecture Overview**

The Wallet Management System is built using Node.js and Express.js framework. It utilizes MongoDB as the database for storing user information, wallets, and transaction records. JSON Web Tokens (JWT) are employed for user authentication and authorization purposes.

**Key components of the system include:**

**Express.js:** A web application framework for Node.js used to create server-side applications.

**MongoDB:** A NoSQL database used for data storage and retrieval.

**JWT:** JSON Web Tokens used for secure authentication and authorization.

**Swagger:** Swagger UI is integrated for API documentation and testing purposes which can be found here [Wallet Api documentations](https://walletfincra-fdc366187160.herokuapp.com/api-docs/)

**Prerequisites**

Before using the Wallet Management System, ensure the following prerequisites are met:

**Node.js** and npm are installed on the system.

**MongoDB** server is running locally or accessible.

Environment variables are properly configured, including the `SECRET\_KEY` for JWT.

**Getting Started**

To start using the Wallet Management System, follow these steps:

1. Clone the repository from the provided source.

2. Install dependencies using `npm install`.

3. Configure environment variables, including the MongoDB connection string and `SECRET\_KEY`.

4. Run the server using `npm start`.

**Endpoints**

User Management

User Registration

- Endpoint: `POST /api/user/signup`

- Description: Registers a new user in the system.

- Request Body:

- `email`: User's email address.

- `password`: User's password.

- Response:

- `200 OK`: User registered successfully.

- `400 Bad Request`: User already exists.

User Login

- Endpoint: `POST /Api/user/login`

- Description: Authenticates a user and generates a JWT token for authorization.

- Request Body:

- `email`: User's email address.

- `password`: User's password.

- Response:

- `200 OK`: Login successful, returns JWT token.

- `401 Unauthorized`: Invalid email or password.

**Get User Details**

- Endpoint: `GET /api/user/:userId`

- Description: Retrieves user details by user ID.

- Request Parameters:

- `userId`: ID of the user.

- Response:

- `200 OK`: User details retrieved successfully.

- `404 Not Found`: User not found.

**Update User Profile**

- Endpoint: `PUT /api/user/:userId`

- Description: Updates user profile information.

- Request Parameters:

- `userId`: ID of the user.

- Request Body:

- User profile data to be updated.

- Response:

- `200 OK`: User profile updated successfully.

- `404 Not Found`: User not found.

**Wallet Management**

*Create Wallet Account*

- Endpoint: `POST /wallet`

- Description: Creates a new wallet account for the authenticated user.

- Request Headers:

- `Authorization`: JWT token for user authentication.

- Response:

- `201 Created`: Wallet created successfully.

- `400 Bad Request`: User already has a wallet.

*Credit Wallet*

- Endpoint: `POST /wallet/credit`

- Description: Credits the user's wallet with a specified amount.

- Request Headers:

- `Authorization`: JWT token for user authentication.

- Request Body:

- `amount`: Amount to credit.

- Response:

- `200 OK`: Wallet credited successfully.

- `400 Bad Request`: Invalid amount.

*Debit Wallet*

- Endpoint: `POST /wallet/debit`

- Description: Debits the user's wallet with a specified amount and credits the receiver's wallet.

- Request Headers:

- `Authorization`: JWT token for user authentication.

- Request Body:

- `amount`: Amount to debit.

- `walletAccountNumber`: Receiver's wallet account number.

- Response:

- `200 OK`: Wallet debited and credited successfully.

- `400 Bad Request`: Invalid amount or receiver's wallet not found.

*Transaction Management*

*Send Funds*

- Endpoint: `POST /api/transaction/send`

- Description: Sends funds from the user's wallet to another user's wallet.

- Request Headers:

- `Authorization`: JWT token for user authentication.

- Request Body:

- `amount`: Amount to send.

- `receiverAccountNumber`: Receiver's wallet account number.

- Response:

- `200 OK`: Funds sent successfully.

- `400 Bad Request`: Invalid amount or receiver's wallet not found.

**Withdraw Funds**

- Endpoint: `POST /api/users/:userId/wallet/withdraw`

- Description: Withdraws funds from the user's wallet.

- Request Headers:

- `Authorization`: JWT token for user authentication.

- Request Parameters:

- `userId`: ID of the user.

- Request Body:

- `amount`: Amount to withdraw.

- Response:

- `200 OK`: Withdrawal successful.

- `400 Bad Request`: Insufficient funds or user not found.

**Get Transaction History**

- Endpoint: `GET /api/users/:userId/wallet/transactions`

- Description: Retrieves transaction history for the user's wallet.

- Request Headers:

- `Authorization`: JWT token for user authentication.

- Request Parameters:

- `userId`: ID of the user.

- Response:

- `200 OK`: Transaction history retrieved successfully.

**Conclusion**

The Wallet Management System provides a secure and efficient way to manage user wallets and conduct fund transactions. By leveraging Express.js and MongoDB, it offers scalability and reliability for handling user data. For further details, refer to the API documentation provided using Swagger UI.