System Architecture Overview

    1. System Components

        The system is built with a combination of a backend REST API (Node.js/Express), a frontend application (React), and WebSocket integration for real-time notifications. It allows users to make purchases, receive earnings based on a referral system, and view purchase and earnings data. The system also includes a WebSocket server for pushing notifications to the user regarding earnings.

    2. Technologies Used

        - Backend: Node.js, Express.js, Sequelize (for database interactions)

        - Frontend: React.js

        - Database: MySQL (using Sequelize as an ORM)

        - WebSocket: Socket.io for real-time notifications

        - Authentication: JWT (JSON Web Token)

        - Version Control: Git (GitHub or GitLab)

    3. System Flow

        User Registration & Authentication: Users can sign up and log in. JWT is used for authentication and maintaining user sessions.

        Purchase Process: Users can make purchases, and based on their referral status, earnings are distributed to the parent users.

        Real-Time Notifications: Whenever earnings are updated due to a purchase, the user’s parent receives a WebSocket notification.

        Purchase & Earnings Tracking: The system stores purchase data and earnings for each user, which can be retrieved and displayed.

API Documentation

    1. Register API

        Endpoint: POST /api/register

        Method: POST

        Description: This endpoint registers a new user into the system.

        Request Body:

        {

        "name": "string",          // Name of the user

        "email": "string",         // Email of the user (unique)

        "password": "string",      // Password (hashed before saving)

        "referredBy": "string",    // Email of the user who referred this user (optional)

        "level": "integer"         // User level (optional, default is 1)

        }

        Response:

            Success (201 Created):

                {

                "message": "User registered successfully",

                "user": {

                    "id": "integer",

                    "name": "string",

                    "email": "string",

                    "referredBy": "string",

                    "level": "integer"

                }

                }

            Failure (400 Bad Request or 500 Server Error):

            {

            "message": "Error message"

            }

    2. Login API

        Endpoint: POST /api/login

        Method: POST

        Description: This endpoint authenticates the user and provides a JWT token for further requests.

        Request Body:

            {

            "email": "string",         // Email of the user

            "password": "string"       // Password of the user

            }

        Response:

            Success (200 OK):

                {

                "message": "Login successful",

                "token": "string",       // JWT token for authorization

                "user": {

                    "id": "integer",

                    "name": "string",

                    "email": "string",

                    "level": "integer"

                }

                }

            Failure (401 Unauthorized or 500 Server Error):

                {

                "message": "Invalid email or password"

                }

    3. Purchase API

        Endpoint: POST /api/purchase

        Method: POST

        Description: This endpoint processes a purchase made by the user, calculates earnings for the referred users (if applicable), and stores purchase data in the database.

        Request Body:

            {

            "userId": "integer",           // ID of the user making the purchase

            "purchaseAmount": "float"      // Amount of the purchase

            }

        Response:

            Success (200 OK):

                {

                "message": "Purchase processed successfully"

                }

            Failure (400 or 500):

                {

                "message": "Error message"

                }

    4. Get Purchase Details API

        Endpoint: GET /api/getPurchaseDetails/:userId

        Method: GET

        Description: This endpoint retrieves all the purchase details made by the specified user.

        Response:

            Success (200 OK):

                [

                {

                    "id": "integer",

                    "userId": "integer",

                    "purchaseAmount": "float",

                    "createdAt": "timestamp",

                    "updatedAt": "timestamp"

                },

                ...

                ]

            Failure:

                {

                "message": "Error fetching purchase details"

                }

    5. Get Earning Details API

        Endpoint: GET /api/getEarningDetails/:userId

        Method: GET

        Description: This endpoint retrieves the earnings details of a user, including both direct and indirect earnings.

        Response:

            Success (200 OK):

            [

            {

                "id": "integer",

                "userId": "integer",

                "referredUserId": "integer",

                "purchaseAmount": "float",

                "directEarnings": "float",

                "indirectEarnings": "float",

                "createdAt": "timestamp",

                "updatedAt": "timestamp"

            },

            ...

            ]

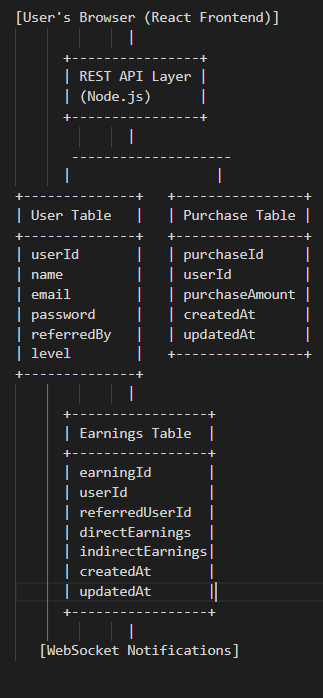
        Failure:

            {

            "message": "Error fetching earning details"

            }

System Architecture Diagram



Flow of Operations

    1. User Registration and Login

        - Users register through the frontend by providing their name, email, and password.

        - Backend validates the data and stores user details in the User Table, hashing the password for security.

        - During login, the system verifies the user’s email and password and issues a JWT token for authenticated access.

        User Table Details

        Fields:

            - userId: Primary key, auto-incremented.

            - name: Name of the user.

            - email: Unique email for the user.

            - password: Hashed password for secure login.

            - referredBy: Email of the referring user (if any).

            - level: User’s level in the hierarchy.

        Operations:

            - Registration: Adds a new user to the table.

            - Login: Authenticates the user and generates a JWT token.

            - Referral Tracking: Links users via the referredBy field.

    2. Purchase Process

        - User initiates a purchase via the frontend.

        - The backend processes the purchase:

            - Saves purchase details in the Purchases table.

            - Calculates and updates earnings in the Earnings table.

            - Sends WebSocket notifications to parent users.

        - The response is returned to the frontend with a success or failure message.

    3. Real-Time Earnings Notification

        - When a new purchase is made, the backend sends real-time WebSocket notifications to parent users about the earnings (both direct and indirect).

        - The frontend listens to these notifications and updates the earnings display.

    4. Viewing Purchase and Earnings Data

        - Users can view their purchase history and earnings by accessing the appropriate API endpoints (/getPurchaseDetails/:userId and /getEarningDetails/:userId).

        - The frontend makes these API calls to display the data in tabular format.