Server.js

---

1. Get All Stations

- **URL:** `/stations`
- **Method:** GET
- **Description:** Retrieves a list of all bike stations with associated bikes, parking places, and reviews.
- **Response:**
  - Status Code: 200 OK
  - Body: Array of station objects with nested arrays of bikes, parking places, and reviews.

2. Get Station by ID

- **URL:** `/station/:id`
- **Method:** GET
- **Description:** Retrieves a specific bike station by its ID with associated bikes, parking places, and reviews.
- **Response:**
  - Status Code: 200 OK
  - Body: Station object with nested arrays of bikes, parking places, and reviews.

3. Get All Categories

- **URL:** `/categories`
- **Method:** GET
- **Description:** Retrieves a list of all bike categories.
- **Response:**
  - Status Code: 200 OK
  - Body: Array of category objects.

4. Get All Models

- **URL:** `/models`
- **Method:** GET
- **Description:** Retrieves a list of all bike models.
- **Response:**
  - Status Code: 200 OK
  - Body: Array of model objects.

5. Get All Bikes

- **URL:** `/bikes`
- **Method:** GET
- **Description:** Retrieves a list of all bikes with associated reviews.
- **Response:**
  - Status Code: 200 OK
  - Body: Array of bike objects with nested arrays of reviews.

6. Get Bike by ID

- **URL:** `/bike/:id`
- **Method:** GET
- **Description:** Retrieves a specific bike by its ID with associated reviews.

- **Response:**
  - Status Code: 200 OK
  - Body: Bike object with nested arrays of reviews.

7. Get All Model Reviews

- **URL:** `/model_reviews`
- **Method:** GET
- **Description:** Retrieves a list of all model reviews.
- **Response:**
  - Status Code: 200 OK
  - Body: Array of model review objects.

8. Default Route

- **URL:** `/`
- **Method:** GET
- **Description:** Returns a message confirming that the server is running.
- **Response:**
  - Status Code: 200 OK
  - Body: "Server is Running"

module_station.js

---

1. Create Station

- **URL:** `/`
- **Method:** POST
- **Description:** Creates a new bike station with provided details and parking places.
- **Request Body:**
  - JSON object containing station details and an array of parking places.
- **Response:**
  - Status Code: 200 OK
  - Body: "Station Created" if successful.

2. Update Station

- **URL:** `/`
- **Method:** PUT
- **Description:** Updates an existing bike station with provided details.
- **Request Body:**
  - JSON object containing station details.
- **Response:**
  - Status Code: 200 OK
  - Body: "Station Updated" if successful.

3. Delete Station

- **URL:** `/`
- **Method:** DELETE
- **Description:** Deletes an existing bike station and associated data (parking places, reviews).
- **Request Body:**

    - JSON object containing the station ID.
- **Response:**
    - Status Code: 200 OK
    - Body: "Station Deleted" if successful.

module_category.js

---

1. Create Category

- **URL:** `/`
- **Method:** POST
- **Description:** Creates a new bike category with the provided name.
- **Request Body:**
    - JSON object containing the category name.
- **Response:**
    - Status Code: 200 OK
    - Body: "Category Added" if successful.

2. Update Category

- **URL:** `/`
- **Method:** PUT
- **Description:** Updates an existing bike category with the provided name.
- **Request Body:**
    - JSON object containing the category ID and updated name.
- **Response:**
    - Status Code: 200 OK
    - Body: "Category Updated" if successful.

3. Delete Category

- **URL:** `/`
- **Method:** DELETE
- **Description:** Deletes an existing bike category and associated data (bikes, parking places, reviews, models).
- **Request Body:**
    - JSON object containing the category ID.
- **Response:**
    - Status Code: 200 OK
    - Body: "Category Deleted" if successful.

module_model.js

---

1. Create Model

- **URL:** `/`
- **Method:** POST
- **Description:** Creates a new bike model with the provided details.
- **Request Body:**
    - JSON object containing the model details (category ID, name, description, wheel size, manufacturer, brakes type).

- **Response:**
  - Status Code: 200 OK
  - Body: "Model Created" if successful.

2. Update Model

- **URL:** `/`
- **Method:** PUT
- **Description:** Updates an existing bike model with the provided details.
- **Request Body:**
  - JSON object containing the model ID and updated details.
- **Response:**
  - Status Code: 200 OK
  - Body: "Model Updated" if successful.

3. Delete Model

- **URL:** `/`
- **Method:** DELETE
- **Description:** Deletes an existing bike model and associated data (bikes, reviews).
- **Request Body:**
  - JSON object containing the model ID.
- **Response:**
  - Status Code: 200 OK
  - Body: "Model Deleted" if successful.

module_bike.js

---

1. Create Bike

- **URL:** `/`
- **Method:** POST
- **Description:** Creates a new bike with the provided details.
- **Request Body:**
  - JSON object containing the bike details (model ID, unique ID).
- **Response:**
  - Status Code: 200 OK
  - Body: "Bike Created" if successful.

2. Update Bike

- **URL:** `/`
- **Method:** PUT
- **Description:** Updates an existing bike with the provided details.
- **Request Body:**
  - JSON object containing the bike ID and updated details.
- **Response:**
  - Status Code: 200 OK
  - Body: "Bike Updated" if successful.

3. Delete Bike

- **URL:** `/`

- **Method:** DELETE
- **Description:** Deletes an existing bike.
- **Request Body:**
  - JSON object containing the bike ID.
- **Response:**
  - Status Code: 200 OK
  - Body: "Bike Deleted" if successful.

module_user.js

---

1. Create User

- **URL:** `/`
- **Method:** POST
- **Description:** Creates a new user with the provided details.
- **Request Body:**
  - JSON object containing the user details (email, password).
- **Response:**
  - Status Code: 200 OK
  - Body: "User Created" if successful.

2. Delete User

- **URL:** `/`
- **Method:** DELETE
- **Description:** Deletes an existing user and associated data (model reviews, station reviews).
- **Request Body:**
  - JSON object containing the user ID.
- **Response:**
  - Status Code: 200 OK
  - Body: "User Deleted" if successful.

3. Update Wallet Balance

- **URL:** `/wallet`
- **Method:** PUT
- **Description:** Updates the wallet balance of the authenticated user.
- **Request Body:**
  - JSON object containing the new wallet balance.
- **Request Header:**
  - Authorization: Bearer Token (for user authentication).
- **Response:**
  - Status Code: 200 OK
  - Body: "Wallet updated" if successful.

module_review.js

---

1. Create Model Review

- **URL:** `/model`

- **Method:** POST
- **Description:** Allows authenticated users to create a review for a bike model.
- **Request Body:**
  - JSON object containing the model ID, rating, and review text.
- **Request Header:**
  - Authorization: Bearer Token (for user authentication).
- **Response:**
  - Status Code: 200 OK
  - Body: "Review Created" if successful.

2. Create Station Review

- **URL:** `/station`
- **Method:** POST
- **Description:** Allows authenticated users to create a review for a bike station.
- **Request Body:**
  - JSON object containing the station ID, rating, and review text.
- **Request Header:**
  - Authorization: Bearer Token (for user authentication).
- **Response:**
  - Status Code: 200 OK
  - Body: "Review Created" if successful.

module_login.js

---

1. User Login

- **URL:** `/`
- **Method:** POST
- **Description:** Authenticates user credentials and returns a JWT token upon successful login.
- **Request Body:**
  - JSON object containing user email and password.
- **Response:**
  - Status Code: 200 OK
  - Body:

```
{
    "message": "login successful",
    "login": "user_email",
    "token": "JWT_token",
    "status": "user/admin"
}
```

2. Test User Authentication

- **URL:** `/tryuser`
- **Method:** GET
- **Description:** Dummy route to test user authentication. Requires a valid JWT token in the request header.
- **Response:**

    - Status Code: 200 OK
    - Body: "Authenticated User"

3. Test Admin Authentication

- **URL:** `/tryadmin`
- **Method:** GET
- **Description:** Dummy route to test admin authentication. Requires a valid JWT token in the request header.
- **Response:**
    - Status Code: 200 OK
    - Body: "Authenticated Admin"

module_registration.js

Register User

- **URL:** `/`
- **Method:** POST
- **Description:** Registers a new user with the provided email and password.
- **Request Body:**
    - JSON object containing user email and password.
- **Response:**
    - Status Code: 200 OK
    - Body: "Successful login"

authenticator_user.js

Authentication Middleware

This middleware function verifies the authenticity of a user token. Admin can pass through both user and admin middleware

Usage

```
const jwt = require('jsonwebtoken');
const authenticateUser = require('./authenticator_user.js');

router.get("/protected_route", authenticateUser, (req,res) => {
    // Route code for authenticated users
});
```