

# Assignment Fourteen

## Please read:

- This is your fourteenth assignment which you need to submit on the LMS. This is an assignment on Backend. The requirements are same as assignment 13.
- Work on the questions given below and be ready with your solution. You have to submit your published API link for your backend below on this page.
- Late Submission: The deadline to submit this assignment is 19th February 2024, 6:00PM IST.

## Important Instructions:

1. Make sure your POSTMAN published link is working properly.
2. Do not copy from someone else as that would be cheating.

## challenge

You can watch the video for reference:

Case Study: Building a TripAdvisor-like Travel Recommendation API

You have been tasked with building the backend API for a travel recommendation platform similar to TripAdvisor. The platform allows users to discover and review travel destinations, read and write travel recommendations, and get information about top-rated and low-rated destinations. The backend uses MongoDB as the database and Mongoose as the ODM.

Your goal is to implement the following features:

1. Travel Destination Management:
  - Create a new travel destination.

- Retrieve travel destination details by name.
- Retrieve a list of all travel destinations.
- Retrieve travel destinations by location (city or country).
- Retrieve travel destinations by their ratings (top-rated and low-rated).
- Update a Travel Destination by ID
- Delete a Travel Destination by ID
- Filter Destinations by Minimum Rating
- Updating Travel Destination Model with User Ratings and Reviews
- Querying Reviews of a Travel Destination

## mongoDB & mongoose

### Step 1: Create a Destination Model

You are tasked with creating a model for a travel destination in your database. Each travel destination will have various attributes. Use Mongoose to create the `Destination` model with the following fields:

1. `name`: The name of the travel destination.
2. `location`: The location of the travel destination, including the city and country.
3. `description`: A brief description or information about the travel destination.
4. `rating`: The rating of the travel destination, represented as a number.
5. `reviews`: An array of user reviews for the travel destination. Each review should include the following fields:
  - `user`: The ID of the user who wrote the review. You can reference the `User` model.
  - `text`: The text content of the review.

### Step 2: Create a New Travel Destination

Create a function `createTravelDestination` that accepts an object containing travel destination data and adds a new destination to the database.

### Step 2: Read a Travel Destination

Create a function `readTravelDestination` that accepts the destination name and retrieves the details of the destination from the database.

## Step 4: Read All Travel Destinations

Create a function `readAllTravelDestinations` that retrieves all travel destinations from the database.

## Step 5: Read All Travel Destinations by Location - Make zomato similar to this

Create a function `readTravelDestinationsByLocation` that accepts a location (city or country) and retrieves all travel destinations in that location.

## Step 6: Read All Travel Destinations by Rating

Create a function `readTravelDestinationsByRating` that retrieves all travel destinations from the database and sorts them in descending order based on their ratings.

## Step 7: Update a Travel Destination by ID

Create a function `updateTravelDestination` that accepts a destination ID and an object with updated data, and updates the destination with the provided ID.

## Step 8: Delete a Travel Destination by ID

Create a function `deleteTravelDestination` that accepts a destination ID and deletes the destination with the provided ID.

## Step 9: Filter Destinations by Minimum Rating

Create a function `filterDestinationsByRating` that accepts a minimum rating and retrieves destinations that have a rating equal to or higher than the specified minimum.

## Step 10: Updating Travel Destination Model with User Ratings and Reviews

1. Update the Travel Destination Model to include a `reviews` field:
2. Create a function to add a review to a travel destination:

## Step 11: Querying Reviews of a Travel Destination

Create a function to retrieve the first 3 reviews of a travel destination, populated with user details:

## Travel Destination API Exercises

### Exercise 1: Creating a Travel Destination API

Challenge:

You are building an Express.js API for a travel destination platform. Create a POST route at `/destinations` that accepts JSON data with destination details. Use the `createTravelDestination` function to save the destination data. Handle errors as well.

1. Create a POST Route: Set up a POST route at `/destinations`.
2. Handle the Request: In the route handler, use the `createTravelDestination` function to save the destination data received in the request body.
3. Success Response: If the destination is successfully added, respond with a success message and the saved destination details.
4. Error Handling: If an error occurs during the process, respond with an error message.

Example:

POST `/destinations`

Request Body:

```
{
  "name": "New Destination",
  "location": "City, Country",
  "description": "Description of the destination.",
  "rating": 4.5,
  "reviews": []
}
```

[COPY](#)

### Exercise 2: Reading a Travel Destination API

Challenge:

Set up a GET route at `/destinations/:name` that allows users to retrieve destination details by providing the destination name as a route parameter. Utilize the `getTravelDestinationByName` function to fetch the destination data. Don't forget error handling!

1. Create a GET Route: Define a GET route at `/destinations/:name` to handle requests for specific destinations.

2. Retrieve Destination Data: In the route handler, use the `getTravelDestinationByName` function to find the destination with the provided name.
3. Success Response: If the destination is found, respond with the destination details.
4. Error Handling: If an error occurs during the process, respond with an error message.

Example:

```
GET /destinations/New%20Destination
```

[COPY](#)

## Exercise 3: Reading All Travel Destinations API

Challenge:

Create a GET route at `/destinations` to fetch all travel destination details. Utilize the `getAllTravelDestinations` function to retrieve the destination data. Handle any potential errors gracefully.

1. Create a GET Route: Define a GET route at `/destinations` to handle requests for all destinations.
2. Retrieve All Destinations: In the route handler, use the `getAllTravelDestinations` function to retrieve all destination data.
3. Success Response: If destinations are found, respond with an array containing all destination details.
4. Error Handling: If an error occurs during the process, respond with an error message.

Example:

```
GET /destinations
```

[COPY](#)

## Exercise 4: Reading Travel Destinations by Location

Challenge:

Create a GET route at `/destinations/location/:location` that allows users to retrieve travel destinations by location (city or country). Utilize the `getTravelDestinationsByLocation` function to fetch the destination data. Handle errors gracefully.

1. Create a GET Route: Set up a GET route at `/destinations/location/:location` to handle requests for retrieving destinations by location.
2. Retrieve Destinations: In the route handler, use the `getTravelDestinationsByLocation` function to find destinations with the provided location.
3. Success Response: If destinations are found, respond with an array containing the details of those destinations.
4. Error Handling: If an error occurs during the process, respond with an error message.

Example:

```
GET /destinations/location/:location
```

[COPY](#)

## Exercise 5: Reading Travel Destinations by Rating

Challenge:

Create a GET route at `/destinations/rating` that retrieves all travel destinations from the database and sorts them in descending order based on their ratings. Utilize the `readTravelDestinationsByRating` function to fetch the data. Handle errors effectively.

1. **Create a GET Route:** Define a GET route at `/destinations/rating` to handle requests for retrieving destinations by rating.
2. **Retrieve and Sort Destinations:** In the route handler, use the `readTravelDestinationsByRating` function to retrieve all destinations and sort them by rating in descending order.
3. **Success Response:** If destinations are successfully sorted, respond with an array containing the sorted destination details.
4. **Error Handling:** If an error occurs during the process, respond with an error message.

Example:

```
GET /destinations/rating
```

[COPY](#)

## Exercise 6: Updating a Travel Destination API

Challenge:

Develop a POST route at `/destinations/:destinationId` that allows users to update a travel destination's details. Utilize the `updateTravelDestination` function to make the necessary changes. Handle errors effectively.

1. **Create a POST Route:** Design a POST route at `/destinations/:destinationId` to handle requests for updating destination details.
2. **Update Destination:** In the route handler, use the `updateTravelDestination` function to modify the specified destination using its ID and the updated data.
3. **Success Response:** If the destination is successfully updated, respond with the updated destination details.
4. **Error Handling:** If an error arises during the process, respond with an error message.

Example:

```
POST /destinations/617a2ed9466e8c00160192e0
```

Request Body:

```
{
```

```
"rating": 4.8
}
```

[COPY](#)

## Exercise 7: Deleting a Travel Destination API

Challenge:

Create a DELETE route at `/destinations/:destinationId` that allows users to delete a travel destination by providing its ID. Utilize the `deleteTravelDestination` function to delete the destination. Handle errors gracefully.

1. **Create a DELETE Route:** Set up a DELETE route at `/destinations/:destinationId` to handle requests for deleting destinations.
2. **Delete Destination:** In the route handler, use the `deleteTravelDestination` function to delete the destination with the provided ID.
3. **Success Response:** If the destination is successfully deleted, respond with a success message and the deleted destination details.
4. **Error Handling:** If an error occurs during the process, respond with an error message.

Example:

```
DELETE /destinations/617a2ed9466e8c00160192e0
```

[COPY](#)

## Exercise 8: Filtering Destinations by Minimum Rating

Challenge:

Create a GET route at `/destinations/filter/:minRating` that allows users to retrieve destinations that have a rating equal to or higher than the specified minimum rating. Utilize the `filterDestinationsByRating` function to fetch the filtered data. Handle errors gracefully.

1. **Create a GET Route:** Set up a GET route at `/destinations/filter/:minRating` to handle requests for filtering destinations by minimum rating.
2. **Retrieve Filtered Destinations:** In the route handler, use the `filterDestinationsByRating` function to find destinations with ratings greater than or equal to the provided minimum rating.
3. **Success Response:** If destinations are found, respond with an array containing the filtered destination details.
4. **Error Handling:** If an error occurs during the process, respond with an error message.

Example:

```
GET /destinations/filter/4.0
```

[COPY](#)

## Exercise 9: Adding Reviews to a Travel Destination

### Challenge:

Develop a POST route at `/destinations/:destinationId/reviews` that allows users to add reviews to a travel destination. Utilize the `addReview` function to add reviews to the destination. Handle errors effectively.

1. **Create a POST Route:** Design a POST route at `/destinations/:destinationId/reviews` to handle requests for adding reviews to a destination.
2. **Add Review:** In the route handler, use the `addReview` function to add a review to the specified destination using its ID and the review data.
3. **Success Response:** If the review is successfully added, respond with a success message and the updated destination details (including reviews).
4. **Error Handling:** If an error arises during the process, respond with an error message.

### Example:

POST `/destinations/617a2ed9466e8c00160192e0/reviews`

Request Body:

```
{
  "userId": "your-user-id-here",
  "reviewText": "Had an amazing trip!"
}
```

[COPY](#)

## Exercise 10: Retrieving Reviews of a Travel Destination

### Challenge:

Create a GET route at `/destinations/:destinationId/reviews` to retrieve the reviews of a travel destination. Utilize the `getDestinationReviewsWithUserDetails` function to fetch the reviews. Handle errors gracefully.

1. **Create a GET Route:** Set up a GET route at `/destinations/:destinationId/reviews` to handle requests for retrieving reviews of a destination.
2. **Retrieve Reviews:** In the route handler, use the `getDestinationReviewsWithUserDetails` function to find the reviews of the specified destination.
3. **Success Response:** If reviews are found, respond with an array containing the reviews along with user details (e.g., username and profile picture URL).
4. **Error Handling:** If an error occurs during the process, respond with an error message.

### Example:

GET `/destinations/617a2ed9466e8c00160192e0/reviews`