Assignment Sixteen

Please read:

- This is your sixteenth assignment which you need to submit on the LMS. This is an assignment on Backend. The requirements are same as assignment 15.
- 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:

mongoDB & mongoose

Step 1: Define a User Model

Let's start by defining the User Model with fields like email, password, profile picture URL, username, and other relevant details.

Step 2: Create a Function for User Signup

Now, let's implement the signup functionality. Create a function `signup` that accepts an object containing user details and creates a new user in the database.

Step 3: Create a Function for User Login

Create a function `login` that accepts email and password, verifies the credentials, and returns user details upon successful login.

- Get email and password from user - function paramaters
- Need to get email and password from database - findOne
- Make sure both matches. if yes, then login, if no, then bye bye. - if else logic

Step 4: Create a Function to Change Password

Create a function `changePassword` that accepts a user's email, current password, and new password. Update the password if the current password matches.

Step 5: Create a Function to Update Profile Picture

Create a function `updateProfilePicture` that accepts a user's email and the new profile picture URL. Update the profile picture URL for the user.

Step 6: Create a Function to Update Contact Details

Create a function `updateContactDetails` that accepts a user's email and an object containing updated contact details. Update the contact details for the user. Make sure to add the phoneNumber (Number) key in the User model first.

Step 7: Create a Function to Find User by Phone Number

Create a function `findUserByPhoneNumber` that accepts a phone number and retrieves the user associated with that phone number.

Step 8: Updating Movie Model with User Ratings and Reviews

1. Update the Movie Model to include `reviews` field.

2. Create a function to add a review to a movie:

- Breakdown steps

    1. Get the movie to be updated // findOneById
    2. const movieToBeUpdated = await findOneById
    3. movieToBeUpdated.reviews
    4. `{ user: "isissksk3787393", text: "OMG! Loved it!" }`
    5. `movieToBeUpdated.reviews.push({ user: "isissksk3787393", text: "OMG! Loved it!" })`
    6. movieToBeUpdated.save()

Step 9: Querying Top and Bottom Ratings of a Movie

1. Create a function to retrieve the top 5 ratings and reviews of a movie.
2. Create a function to retrieve the bottom 5 ratings and reviews of a movie.

Step 10: Querying Reviews of a Movie

Create a function to retrieve the first 3 reviews of a movie, populated with user details.

express

ex01: user signup API

challenge

Develop a POST route at `/signup` that allows users to create new accounts by providing their user details. Utilize the `signup` function to save the user's information. Handle errors gracefully.

1. Create a POST Route: Design a POST route at `/signup` to handle requests for user signups.
2. Signup User: In the route handler, use the `signup` function to create a new user account based on the provided user details.
3. Success Response: If the user account is successfully created, respond with the saved user details.
4. Error Handling: If an error occurs during the process, respond with an error message.

example

```
POST /users

Request Body:
{
  "email": "user@example.com",
  "password": "secretpassword",
  "username": "exampleuser",
  "profilePicture": "<https://example.com/profile.jpg>"
}
```

COPY

ex02: user login API

challenge

Develop a POST route at `/login` that allows users to log into their accounts by providing their email and password. Utilize the `login` function to authenticate users. Handle errors effectively.

1. Create a POST Route: Set up a POST route at `/login` to handle requests for user logins.
2. Login User: In the route handler, use the `login` function to authenticate the user based on the provided email and password.
3. Success Response: If the user is successfully authenticated, respond with the user's details.
4. Error Handling: If the credentials are invalid or an error arises, respond with an error message.

example

```
POST /login

Request Body:
{
  "email": "user@example.com",
  "password": "secretpassword"
}
```

COPY

ex03: changing password API

```
/user/:userId/password POST
```

COPY

challenge

Design a POST route at `/user/:userId/password` that enables users to update their passwords. Utilize the `changePassword` function to make the necessary changes. Handle errors proficiently.

1. Create a POST Route: Set up a POST route at `/user/:userId/password` to handle requests for changing user passwords.
2. Change Password: In the route handler, use the `changePassword` function to update the user's password if the current password is correct.
3. Success Response: If the password is successfully updated, respond with the updated user details.
4. Error Handling: If the credentials are invalid or an error arises, respond with an error message.

example

```
POST /change-password

Request Body:
{
  "email": "user@example.com",
  "currentPassword": "oldpassword",
  "newPassword": "newpassword"
}
```

COPY

ex04: updating Profile Picture API

```
/user/:userId/profile
```

COPY

challenge

Develop a POST route at `/update-profile-picture` that allows users to modify their profile pictures. Utilize the `updateProfilePicture` function to make the necessary changes. Handle errors effectively.

1. Create a POST Route: Design a POST route at `/update-profile-picture` to handle requests for updating user profile pictures.
2. Update Profile Picture: In the route handler, use the `updateProfilePicture` function to update the user's profile picture URL based on their email.
3. Success Response: If the profile picture is successfully updated, respond with the updated user details.
4. Error Handling: If the user is not found or an error arises, respond with an error message.

example

```
POST /update-profile-picture

Request Body:
{
  "email": "user@example.com",
  "newProfilePictureUrl": "<https://example.com/new-profile.jpg>"
}
```

COPY

ex05: updating Contact Details API

challenge

Develop a POST route at `/update-contact/:email` that allows users to modify their contact details. Utilize the `updateContactDetails` function to make the necessary changes. Handle errors proficiently.

1. Create a POST Route: Design a POST route at `/update-contact/:email` to handle requests for updating user contact details.
2. Update Contact Details: In the route handler, use the `updateContactDetails` function to update the user's contact details based on their email.
3. Success Response: If the contact details are successfully updated, respond with the updated user details.
4. Error Handling: If the user is not found or an error arises, respond with an error message.

example

```
POST /update-contact/user@example.com

Request Body:
{
  "phoneNumber": "1234567890",
  "address": "123 Main St, City"
}

**Note: We are about to add new fields to the existing data, so make sure to add phoneNumber
```

COPY

ex06: finding User by Phone Number API

challenge

Design a GET route at `/users/phone/:phoneNumber` that allows users to search for a user by their phone number. Utilize the `findUserByPhoneNumber` function to retrieve the user data. Handle errors effectively.

1. Create a GET Route: Set up a GET route at `/users/phone/:phoneNumber` to handle requests for finding a user by phone number.
2. Find User: In the route handler, use the `findUserByPhoneNumber` function to locate the user based on the provided phone number.
3. Success Response: If the user is found, respond with the user's details.
4. Error Handling: If the user is not found or an error arises, respond with an error message.

example

```
GET /users/phone/1234567890
```

COPY

ex07: adding Rating and Review API

challenge

Develop a POST route at `/movies/:movieId/rating` that enables users to submit ratings and reviews for a movie. Utilize the `addRatingAndReview` function to handle the process. Handle errors effectively.

1. Create a POST Route: Set up a POST route at `/movies/:movieId/rating` to handle requests for adding ratings and reviews to a movie.
2. Add Rating and Review: In the route handler, use the `addRatingAndReview` function to add the provided rating and review to the specified movie.
3. Success Response: If the rating and review are successfully added, respond with the updated movie details including the populated reviews.
4. Error Handling: If the movie is not found or an error arises, respond with an error message.

example

```
POST /movies/61524b6b6a7a11abc1234567/rating

Request Body:
{
  "userId": "61524c806a7a11abc1234568", // use your created user's id
  "rating": 8,
        "review": "Good Movie"
}
```

COPY

ex08: movie Reviews with User Details API

challenge

Develop a GET route at `/movies/:movieId/reviews` that allows users to fetch movie reviews with associated user details. Utilize the `getMovieReviewsWithUserDetails` function to handle the retrieval. Handle errors proficiently.

1. Create a GET Route: Set up a GET route at `/movies/:movieId/reviews` to handle requests for fetching movie reviews with user details.
2. Retrieve Reviews: In the route handler, use the `getMovieReviewsWithUserDetails` function to retrieve the movie reviews with associated user details for the specified movie.

3. Limit to Top 3 Reviews: Limit the reviews to the top 3, extracting the review text and user details.
4. Success Response: If the reviews are successfully fetched, respond with the array of reviews along with user details.
5. Error Handling: If the movie is not found or an error arises, respond with an error message.

example

```
GET /movies/61524b6b6a7a11abc1234567/reviews
```