User API Documentation

1. **Get Documentation**

Endpoint: `GET /`

Sends the documentation file to the user.

Responses:
- `200 OK`: Documentation file sent successfully.

2. **Register User**

Endpoint: `POST /register`

Registers a new user with the provided details.

Request Body:
- `UserName` (string): The username for the new user. Must be 3-20 characters long and should not contain spaces.
- `Name` (string): The full name of the user. Must be 3-50 characters long.
- `Password` (string): The password for the new user. Must be at least 8 characters long, contain at least one lowercase letter, one uppercase letter, one number, and one special character.
- `CPassword` (string): Confirmation of the password. Must match the `Password`.
- `PhoneNo` (string): The phone number of the user. Must be a valid Indian phone number.
- `EmailID` (string): The email address of the user. Must be a valid email.

Responses:
- `201 Created`: Registration successful.
- `400 Bad Request`: Validation errors, such as invalid email, short name, mismatched passwords, etc.
- `500 Internal Server Error`: Unexpected server errors.

3. **Login User**

Endpoint: `POST /login`

Logs in a user with the provided credentials.

Request Body:
- `UserName` (string, optional): The username of the user. Required if `EmailID` is not provided.
- `EmailID` (string, optional): The email address of the user. Required if `UserName` is not provided.

- `Password` (string): The password of the user. Must be a valid strong password.

Responses:
- `200 OK`: Login successful, sets an authentication token.
- `400 Bad Request`: Validation errors, such as invalid credentials.

4. **Authenticate User**

Endpoint: `GET /authenticate`

Authenticates the currently logged-in user.

Responses:
- `200 OK`: User is authenticated, returns user details.
- `404 Not Found`: User not found.
- `500 Internal Server Error`: Unexpected server errors.

5. **Create Gigs**

Endpoint: `POST /Create/gigs`

Creates a new gig with the provided details.

Request Body: Multipart/form-data
- `data` (string): A JSON string containing the gig details:
  - `title` (string): The title of the gig. Must be between 10 and 100 characters long.
  - `cost` (number): The cost of the gig. Must be at least 100.
  - `description` (string): The description of the gig. Must be between 500 and 1500 characters long.
  - `tags` (array of strings): The tags associated with the gig. Each tag must be between 1 and 50 characters long, and a maximum of 10 tags are allowed.
  - `deliveryTime` (number): The delivery time for the gig in hours. Must be at least 1 hour.
  - `category` (string): The category of the gig. Must be between 1 and 50 characters long.
  - `maxPendingOrders` (number): The maximum number of pending orders allowed for the gig. Must be at least 1.
- `files` (array of files): The images for the gig. Must provide between 1 and 5 images.

Responses:
- `201 Created`: Gig created successfully.
- `400 Bad Request`: Validation errors, such as invalid fields or exceeding image limits.
- `500 Internal Server Error`: Unexpected server errors.

6. **Fetch Gigs**

Endpoint: `GET /gigs`

Fetches gigs based on the provided query parameters.

Query Parameters (optional):
- `title` (string): The title of the gig to search for. Maximum 100 characters.
- `tags` (string): A comma-separated list of tags to search for. Each tag must be between 1 and 50 characters long.
- `category` (string): The category of the gig to search for. Maximum 50 characters.
- `recommended` (string): Whether to search for recommended gigs. Should be either "true" or "false".
- `by` (string): The username of the gig creator to search for. Maximum 50 characters.
- `gigId` (string): The ID of the gig to search for. Maximum 35 characters.

Responses:
- `200 OK`: Gigs fetched successfully.
- `400 Bad Request`: Validation errors, such as exceeding character limits.
- `500 Internal Server Error`: Unexpected server errors.

7. **Get Gig Media**

Endpoint: `GET /gigs/media/:gid/:image`

Fetches the specified image for the given gig.

Path Parameters:
- `gid` (string): The ID of the gig.
- `image` (string): The name of the image file.

Responses:
- `200 OK`: Image file sent successfully.
- `404 Not Found`: Image file not found.
- `500 Internal Server Error`: Unexpected server errors.

8. **Create Order**

Endpoint: `POST /gigs/order`

Creates a new order for the specified gig.

Request Body:
- `gigId` (string): The ID of the gig to order. Must be a valid string with a maximum length of 35 characters.

Responses:
- `201 Created`: Order created successfully.
- `400 Bad Request`: Validation errors, such as invalid gig ID or attempting to order own gig.
- `500 Internal Server Error`: Unexpected server errors.

9. **Fetch Orders**

Endpoint: `GET /gigs/order`

Fetches orders based on the provided query parameters.

Query Parameters (optional):
- `status` (string): The status of the orders to search for. Valid values are "cancelled", "pending", "completed", and "accepted".
- `orderId` (string): The ID of the order to search for. Must be a valid string with a maximum length of 35 characters and start with "order-".

Responses:
- `200 OK`: Orders fetched successfully.
- `400 Bad Request`: Validation errors, such as invalid status or order ID.
- `500 Internal Server Error`: Unexpected server errors.

10. **Get Notification**

Endpoint: `GET /gigs/notification`

Fetches notifications for the currently logged-in user.

Responses:
- `200 OK`: Notifications fetched successfully.
- `500 Internal Server Error`: Unexpected server errors.

11. **Create Wallet**

Endpoint: `POST /Create/wallet`

Creates a wallet for the currently logged-in user.

Responses:
- `201 Created`: Wallet created successfully.
- `500 Internal Server Error`: Unexpected server errors.

12. **Cancel Order**

Endpoint: `POST /gigs/order/cancel`

Cancels the specified order.

Request Body:
- `orderId` (string): The ID of the order to cancel. Must be a valid string starting with "order-".

Responses:
- `200 OK`: Order cancelled successfully.
- `400 Bad Request`: Validation errors, such as invalid order ID or attempting to cancel an order at an invalid stage.
- `403 Forbidden`: User does not have permission to cancel the order.
- `500 Internal Server Error`: Unexpected server errors.

13. **Get Wallet**

Endpoint: `GET /user/wallet`

Fetches the wallet details for the currently logged-in user.

Responses:
- `200 OK`: Wallet details fetched successfully.
- `400 Bad Request`: Wallet not created yet.
- `500 Internal Server Error`: Unexpected server errors.