Here's the updated documentation for the API, including the new endpoint for transferring funds without the error details.

User API Documentation

1. **Get Documentation**
   - **Endpoint:** `GET /`
   - **Description:** Sends the documentation file to the user.
   - **Responses:**
     - `200 OK`: Documentation file sent successfully.

2. **Register User**
   - **Endpoint:** `POST /register`
   - **Description:** 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.

3. **Login User**
   - **Endpoint:** `POST /login`
   - **Description:** 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.

4. **Authenticate User**
   - **Endpoint:** `GET /authenticate`
   - **Description:** Authenticates the currently logged-in user.
   - **Responses:**
     - `200 OK`: User is authenticated, returns user details.

5. **Create Gigs**
   - **Endpoint:** `POST /Create/gigs`
   - **Description:** 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.

6. **Fetch Gigs**
   - **Endpoint:** `GET /gigs`
   - **Description:** 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.

7. **Get Gig Media**
   - **Endpoint:** `GET /gigs/media/:gid/:image`
   - **Description:** 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.

8. **Create Order**
   - **Endpoint:** `POST /gigs/order`
   - **Description:** 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.

9. **Fetch Orders**
   - **Endpoint:** `GET /gigs/order`
   - **Description:** 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.

10. **Get Notification**
   - **Endpoint:** `GET /gigs/notification`
   - **Description:** Fetches notifications for the currently logged-in user.
   - **Responses:**
      - `200 OK`: Notifications fetched successfully.

11. **Create Wallet**
   - **Endpoint:** `POST /Create/wallet`
   - **Description:** Creates a wallet for the currently logged-in user.
   - **Responses:**
      - `201 Created`: Wallet created successfully.

12. **Cancel Order**
   - **Endpoint:** `POST /gigs/order/cancel`
   - **Description:** 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.

13. **Get Wallet**
   - **Endpoint:** `GET /user/wallet`
   - **Description:** Fetches the wallet details for the currently logged-in user.

   - **Responses:**
      - `200 OK`: Wallet details fetched successfully.

14. **Transfer Funds**
   - **Endpoint:** `POST /wallet/transfer`
   - **Description:** Transfer funds from the authenticated user's wallet to another user's wallet.
   - **Request Headers:**
      - `Authorization: Bearer <token>` (required)
   - **Request Body:**
      - `receiverUserName` (string): The username of the user to receive the funds. Must be between 3 and 20 characters long and should not contain spaces.
      - `amount` (number): The amount of funds to transfer. Must be greater than zero.
   - **Responses:**
      - `200 OK`: Funds transferred successfully.

Order Management Endpoints

1. **Complete Order**
   - **Endpoint:** `POST /gigs/order/complete`
   - **Description:** Marks an order as completed.
   - **Request Body:**
      - `orderId` (string): The ID of the order to complete. Must be a valid string starting with "order-" and 10-50 characters long.
      - `orderResult` (string): The URL of the order result. Must be a valid URL.
   - **Responses:**
      - `200 OK`: Order completed successfully.

2. **Accept Order**
   - **Endpoint:** `POST /gigs/order/accept`
   - **Description:** Accepts a completed order.
   - **Request Body:**
      - `orderId` (string): The ID of the order to accept. Must be a valid string starting with "order-" and 10-50 characters long.
   - **Responses:**
      - `200 OK`: Order accepted successfully, and the amount has been sent to the seller.

3. **Post Review**
   - **Endpoint:** `POST /gigs/order/review`
   - **Description:** Posts a review for a completed order.
   - **Request Body:**
      - `orderId` (string): The ID of the order to review. Must be a valid string starting with "order-" and 10-50 characters long.

        - `rating` (number): The rating for the order. Must be between 1 and 5.
        - `comment` (string): The comment for the review. Must be between 40 and 1000 characters long.
    - **Responses:**
        - `200 OK`: Review posted successfully.