Sahara API Documentation

Base URL: `http://localhost:5000/api`

This documentation provides details for all available API endpoints in the Sahara application, including user and admin routes, request/response formats, and important notes for frontend developers.

User Routes

1. Create a New User

- **URL**: `/users`
- **Method**: POST
- **Description**: Registers a new user and sends an OTP for email verification.
- **Request Body**:

```
{
  "fullName": "string",
  "email": "string",
  "password": "string"
}
```

  - `fullName`: Must be at least 3 characters and include both first and last name (space-separated).
  - `email`: Must be a valid email address (e.g., `user@example.com`).
  - `password`: Must be at least 6 characters, contain at least one number and one special character (`!@#$%^&*`).

- **Success Response**:
  - **Status**: 201
  - **Body**:

```
{
  "success": true,
  "message": "User registered successfully. Please check your email f
or OTP verification.",
  "data": {
    "_id": "string",
    "fullName": "string",
    "email": "string",
    "emailVerified": false,
    "createdAt": "string",
    "updatedAt": "string"
  }
}
```

- **Failure Response**:
  - **Status**: 400 (Validation errors) or 500 (Server error)
  - **Body**:
    Examples:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional, included in server errors
}
```

```
{
  "success": false,
  "message": "Please provide all required fields"
}
```

```
{
  "success": false,
```

```
  "message": "User with this email already exists"
}
```

- **Notes for Frontend**:
  - After successful registration, prompt the user to check their email for the OTP.
  - Handle validation errors for full name, email, and password formats.
  - Redirect to OTP verification page after successful signup.

---

2. Verify OTP

- **URL**: `/users/otp/verification`
- **Method**: POST
- **Description**: Verifies the OTP sent to the user's email to complete email verification.
- **Request Body**:

```
{
  "email": "string",
  "otp": "string" // 6-digit OTP
}
```

- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "Email verified successfully"
}
```

- **Failure Response**:

- **Status**: 400 (Invalid input/OTP) or 404 (User not found) or 500 (Server error)
- **Body**:
  Examples:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

```
{
  "success": false,
  "message": "Invalid or expired OTP"
}
```

```
{
  "success": false,
  "message": "User not found"
}
```

- **Notes for Frontend**:
  - Display an input field for the 6-digit OTP.
  - Handle cases where the OTP is invalid or expired by prompting the user to resend OTP.
  - After successful verification, redirect to login or dashboard.

---

3. Resend OTP

- **URL**: `/users/otp/resend`
- **Method**: POST

- **Description**: Resends a new OTP to the user's email if the previous OTP is expired or not received.
- **Request Body**:

```
{
  "email": "string"
}
```

- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "New OTP has been sent to your email"
}
```

- **Failure Response**:
  - **Status**: 400 (Invalid input/No OTP found), 404 (User not found), 429 (Rate limit), or 500 (Server error)
  - **Body**:
    Examples:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

```
{
  "success": false,
```

```
  "message": "Please wait 60 seconds before requesting a new OTP"
}
```

```
{
  "success": false,
  "message": "No OTP found for this user"
}
```

- **Notes for Frontend**:
  - Implement a 60-second cooldown for resend requests to prevent abuse.
  - Show a countdown timer for the remaining time if rate-limited.
  - Prompt the user to check their email after a successful resend.

---

4. Login (User or Counsellor) - only user implemented for now

- **URL**: `/users/login`
- **Method**: POST
- **Description**: Authenticates a user or counsellor and returns a JWT token.
- **Request Body**:

```
{
  "email": "string",
  "password": "string",
  "role": "string" // Either "User" or "Counsellor"
}
```

- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
```

```
  "message": "Login successful",
  "data": {
    "User|Counsellor": {
      "_id": "string",
      "fullName": "string",
      "email": "string",
      "emailVerified": true, // Only for User
      // Additional fields for Counsellor: designation, phone, chargePerH
our, etc.
      "createdAt": "string",
      "updatedAt": "string"
    },
    "token": "string" // JWT token
  }
}
```

- **Failure Response**:
  - **Status**: 400 (Invalid input), 401 (Invalid credentials), 403 (Email not verified for User), or 500 (Server error)
  - **Body**:
    Examples:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

```
{
  "success": false,
  "message": "Please verify your email before logging in"
}
```

```
{
  "success": false,
  "message": "Invalid credentials"
}
```

- **Notes for Frontend**:
  - Store the JWT token in local storage or cookies for subsequent authenticated requests.
  - Include a role selection (User/Counsellor) in the login form.
  - Redirect unverified users to the OTP verification page.
  - Handle role-specific data in the response (e.g., additional fields for Counsellor).

---

5. Add Journal Entry

- **URL**: `/users/journals`
- **Method**: POST
- **Description**: Creates a new journal entry for the authenticated user.
- **Authorization**: Bearer token required in `Authorization` header.
- **Request Body**:

```
{
  "title": "string",
  "explicitEmotion": "string",
  "content": "string",
  "shareStatus": boolean
}
```

  - `title`: Must be at least 3 characters.
  - `explicitEmotion`: User's explicit mood (e.g: bad, low, neutral) .
  - `content`: Journal content.

- `shareStatus`: Whether the entry can be shared with counsellor (true/false).

- **Success Response**:
  - **Status**: 201
  - **Body**:

```
{
  "success": true,
  "message": "Journal entry created successfully",
  "data": {
    "_id": "string",
    "user": "string",
    "title": "string",
    "explicitEmotion": "string",
    "content": "string",
    "shareStatus": boolean,
    "createdAt": "string",
    "updatedAt": "string"
  }
}
```

- **Failure Response**:
  - **Status**: 400 (Invalid input), 401 (Unauthorized), or 500 (Server error)
  - **Body**:
    Examples:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

```
{
  "success": false,
```

```
  "message": "Please provide all required fields"
}
```

```
{
  "success": false,
  "message": "No token provided"
}
```

- **Notes for Frontend**:
  - Include the JWT token in the `Authorization` header as `Bearer <token>`.
  - Display validation errors for missing fields or title length.
  - Note: A commented-out restriction limits users to one journal entry per day; this may be enabled later.

---

6. Get User's Journal Entries

- **URL**: `/users/journals`
- **Method**: GET
- **Description**: Retrieves all journal entries for the authenticated user.
- **Authorization**: Bearer token required in `Authorization` header.
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "Journal entries retrieved successfully",
  "data": [
    {
      "title": "string",
      "content": "string",
```

```
      "explicitEmotion": "string",
      "emotionalTone": "string",
      "confidenceScore": number,
      "shareStatus": boolean,
      "createdAt": "string",
      "date": "string", // YYYY-MM-DD
      "time": "string" // HH:MM:SS
    },
    ...
  ]
}
```

- **Failure Response**:
  - **Status**: 401 (Unauthorized) or 500 (Server error)
  - **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Display journal entries in reverse chronological order (newest first).
  - Use `date` and `time` fields for user-friendly formatting.
  - Handle unauthorized errors by redirecting to login.

---

7. Get User's Mood History

- **URL**: `/users/mood/history`
- **Method**: GET
- **Description**: Retrieves the last 7 journal entries' mood data for the authenticated user.

- **Authorization**: Bearer token required in `Authorization` header.
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "Mood history retrieved successfully",
  "data": {
    "history": [
      {
        "day": "string", // e.g., "Mon"
        "mood": "string" // e.g., "happy"
      },
      ...
    ],
    "feltBetterCount": number // Count of "good" or "great" moods
  }
}
```

- **Failure Response**:
  - **Status**: 401 (Unauthorized) or 500 (Server error)
  - **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Display mood history as a timeline or chart (e.g., last 7 days).
  - Use `feltBetterCount` to show positive mood statistics.

- Handle empty history gracefully.

---

8. Request Password Reset

- **URL**: `/users/password/reset`
- **Method**: POST
- **Description**: Initiates a password reset by sending an OTP to the user's email.
- **Request Body**:

```
{
  "email": "string",
  "newPassword": "string"
}
```

  - `newPassword`: Must be at least 6 characters, contain at least one number and one special character (`!@#$%^&*`).
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "OTP has been sent to your email for password reset veri
fication"
}
```

- **Failure Response**:
  - **Status**: 400 (Invalid input), 404 (User not found), or 500 (Server error)
  - **Body**:

```
{
  "success": false,
```

```
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Prompt the user to check their email for the OTP.
  - Redirect to OTP verification for password reset after success.
  - Validate new password format before submission.

---

9. Verify OTP and Reset Password

- **URL**: `/users/password-reset/otp/verification`
- **Method**: POST
- **Description**: Verifies the OTP and resets the user's password.
- **Request Body**:

```
{
  "email": "string",
  "otp": "string" // 6-digit OTP
}
```

- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "Password has been reset successfully"
}
```

- **Failure Response**:

- **Status**: 400 (Invalid input/OTP), 404 (User not found), or 500 (Server error)
- **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Redirect to login page after successful password reset.
  - Handle invalid/expired OTP by prompting to resend OTP.

---

10. Get Counsellor Details

- **URL**: `/users/counsellors`
- **Method**: GET
- **Description**: Retrieves details of up to 15 randomly selected active counsellors.
- **Authorization**: Bearer token required in `Authorization` header.
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "data": {
    "counsellors": [
      {
        "fullName": "string",
        "designation": "string",
```

```
      "email": "string",
      "chargePerHour": number,
      "profilePhoto": {
       "filename": "string",
       "originalName": "string",
       "path": "string",
       "size": number,
       "mimetype": "string"
      }
     },
     ...
   ],
   "total": number
  }
}
```

- **Failure Response**:
  - **Status**: 401 (Unauthorized) or 500 (Server error)
  - **Body**:

```
{
 "success": false,
 "message": "string",
 "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Display counsellor details in a list or grid format.
  - Use `profilePhoto.path` to display images.
  - Handle cases where `total` is 0 (no active counsellors).

Admin Routes

1. Admin Login

- **URL**: `/admin/login`
- **Method**: POST
- **Description**: Authenticates an admin and returns a JWT token.
- **Request Body**:

```
{
  "email": "string",
  "password": "string"
}
```

- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "message": "Login successful",
  "data": {
    "admin": {
      "_id": "string",
      "email": "string",
      "createdAt": "string",
      "updatedAt": "string"
    },
    "token": "string" // JWT token
  }
}
```

- **Failure Response**:
  - **Status**: 400 (Invalid input), 401 (Invalid credentials), or 500 (Server error)
  - **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Store the JWT token for authenticated admin requests.
  - Redirect to admin dashboard on success.

---

2. Add New Counsellor

- **URL**: `/admin/counsellors`
- **Method**: POST
- **Description**: Creates a new counsellor with uploaded profile photo and documents.
- **Authorization**: Bearer token required in `Authorization` header (Admin role).
- **Request Body**: Form-data
  - `fullName`: String (min 3 characters, includes first and last name)
  - `email`: Valid email address
  - `password`: Min 6 characters, at least one number and one special character (`!@#$%^&*`)
  - `phone`: 10-digit number
  - `designation`: String
  - `chargePerHour`: Number
  - `esewaAccountId`: 10-digit number
  - `profilePhoto`: File (single image)
  - `documents`: Files (up to 3 documents)
- **Success Response**:

- **Status**: 201
- **Body**:

```
{
  "success": true,
  "message": "Counsellor added successfully",
  "data": {
    "id": "string",
    "name": "string",
    "email": "string",
    "phone": "string",
    "designation": "string",
    "chargePerHour": number,
    "esewaAccountId": "string",
    "profilePhoto": {
      "filename": "string",
      "originalName": "string",
      "path": "string",
      "size": number,
      "mimetype": "string"
    },
    "documents": [
      {
        "filename": "string",
        "originalName": "string",
        "path": "string",
        "size": number,
        "mimetype": "string"
      },
      ...
    ]
  }
}
```

- **Failure Response**:

- **Status**: 400 (Invalid input), 401 (Unauthorized), or 500 (Server error)
- **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Use `multipart/form-data` for file uploads.
  - Validate all fields client-side before submission.
  - Display uploaded file details and handle file cleanup on failure.

---

3. Get Admin Profile

- **URL**: `/admin/profile`
- **Method**: GET
- **Description**: Retrieves the authenticated admin's profile.
- **Authorization**: Bearer token required in `Authorization` header (Admin role).
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "data": {
    "admin": {
      "_id": "string",
      "email": "string",
      "createdAt": "string",
      "updatedAt": "string"
```

```
    }
  }
}
```

- **Failure Response**:
  - **Status**: 401 (Unauthorized) or 500 (Server error)
  - **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Display admin profile details in a dashboard or profile page.
  - Handle unauthorized errors by redirecting to login.

---

4. Get Total Users Count

- **URL**: `/admin/total-users`
- **Method**: GET
- **Description**: Retrieves the total number of registered users.
- **Authorization**: Bearer token required in `Authorization` header (Admin role).
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "data": {
    "totalUsers": number
```

```
  }
}
```

- **Failure Response**:
  - **Status**: 401 (Unauthorized) or 500 (Server error)
  - **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Display the count in an admin dashboard or analytics section.
  - Handle unauthorized errors appropriately.

---

5. Get Total Counsellors Count

- **URL**: `/admin/total-counsellors`
- **Method**: GET
- **Description**: Retrieves the total number of registered counsellors.
- **Authorization**: Bearer token required in `Authorization` header (Admin role).
- **Success Response**:
  - **Status**: 200
  - **Body**:

```
{
  "success": true,
  "data": {
    "totalCounsellors": number
```

```
  }
}
```

- **Failure Response**:
  - **Status**: 401 (Unauthorized) or 500 (Server error)
  - **Body**:

```
{
  "success": false,
  "message": "string",
  "error": "string" // Optional
}
```

- **Notes for Frontend**:
  - Display the count in an admin dashboard or analytics section.
  - Handle unauthorized errors appropriately.

General Notes for Frontend Developers

- **Authentication**:
  - Protected routes require a JWT token in the `Authorization` header as `Bearer <token>`.
  - Handle token expiration (401 errors) by redirecting to the login page.
- **Error Handling**:
  - Always check the `success` field in responses.
  - Display user-friendly error messages based on the `message` field.
  - Log `error` field details for debugging (if present).
- **File Uploads**:
  - Use `multipart/form-data` for endpoints that accept files (e.g., adding a counsellor).

- Validate file sizes and types client-side to reduce server load.