**SkillSwap User Management (Auth) API Documentation  
  
User Data Format:**{ “email”: “String” (required, must be unique),

“role”: “String” (required, “admin” || “Basic”, default: ”Basic”),

“username”: “String” (required),

“password”: “String” (required),

“description”: “String”,

“isAvailable”: “Boolean”,

“swapPreference”: [“String”] (“in-person” and/or “online” and/or “hybrid”),

“langPreference”: [“String”] (“English” and/or “French” and/or “German” ….),

“location”: {  
 “type”:“String” (must be “Point”),  
 “coordinate”: [Number] (e.g. [-73.856077, 40.848447] },

“skillsOffered”: [{  
 “type”: “Schema.Types.ObjectId”,  
 “ref”: “Skill” }],

“skillsWanted”: [{  
 “type”: “Schema.Types.ObjectId”,  
 “ref”: “Skill” }]

}

**Base URL**

The base URL for all API requests is **https://ws24-skillswap.onrender.com**

**Endpoints**

All endpoints are prefixed with **/api/users**

**1. Create New User**

Allows new users to register.

* **URL:** **/api/users/create**
* **Method:** **POST**
* **Body Parameters:**
  + **username**: String (required)
  + **email**: String (required)
  + **password**: String (required)
  + other params optional
* **Success Response:**
  + **Code:** 201 (Created)
  + **Content:** **{ message: "User successfully registered", user: { id, username, email } }**
* **Error Response:**
  + **Code:** 400 (Bad Request)
  + **Content:** **{ message: "Error message" }**

**2.1 Fetch All Users**

* **Endpoint:** **/api/users**
* **Method:** GET
* **Description:** Fetches data of all users. Returns all Users as JSON Response.
* **Authorization:** Required. JWT token must be included in the HTTP-only cookie.

**2.2 Fetch User Data by ID**

* **Endpoint:** **/api/users/{userId}**
* **Method:** GET
* **Description:** Fetches data for a specific user by their ID. Returns User Info as JSON Response.
* **Authorization:** Required. JWT token must be included in the HTTP-only cookie.
* **URL Parameters:** **id** - The ID of the user whose data is being requested.

**3. Update User Data**

Allows logged-in users to update their own data.

* **URL:** **/api/users/update/{userId}**
* **Method:** **PUT**
* **Headers:**
  + **Authorization**: **Bearer <JWT Token>** (Token retrieved from login cookie)
* **URL Parameters:**
  + **userId**: ID of the user to update
* **Body Parameters:** Any user field(s) to update. (JSON)
* **Success Response:**
  + **Code:** 200 (OK)
  + **Content:** **{ message: "User updated successfully", user: { updated fields } }**
* **Error Response:**
  + **Code:** 401 (Unauthorized) / 403 (Forbidden)
  + **Content:** **{ message: "Error message" }**

**4. Delete User Data**

Allows admins to delete a user's data.

* **URL:** **/api/users/delete/{userId}**
* **Method:** **DELETE**
* **Headers:**
  + **Authorization**: **Bearer <JWT Token>** (Token retrieved from login cookie)
* **URL Parameters:**
  + **userId**: ID of the user to delete
* **Success Response:**
  + **Code:** 200 (OK)
  + **Content:** **{ message: "User deleted successfully" }**
* **Error Response:**
  + **Code:** 401 (Unauthorized) / 403 (Forbidden)
  + **Content:** **{ message: "Error message" }**