**Backend API Documentation**

**Introduction**

This documentation describes the backend authentication API built using Node.js, Express, PostgreSQL, and Argon2 for password hashing. It handles user registration, login, logout, and retrieving user profile information.

**Base URL**

http://yourdomain.com/api/auth

**Authentication API Endpoints**

**1. Register User**

* **Endpoint**: POST /register
* **Description**: Registers a new user by creating an account with a hashed password.
* **Request Body**:
* {
* "username": "user123",
* "email": "user@example.com",
* "password": "securepassword"
* “role\_id”: 2
* }
* **Responses**:
  + **201 Created**
  + {
  + "message": "User registered successfully",
  + "user": {
  + "username": "user123",
  + "email": "user@example.com",
  + “role\_id: 2
  + "created\_at": "2025-03-11T12:34:56.789Z"
  + }
  + }
  + **400 Bad Request**
  + {
  + "error": "Username or email already exists"
  + }
  + **500 Internal Server Error**
  + {
  + "error": "Registration failed",
  + "details": "Error details"
  + }

**2. Login User**

* **Endpoint**: POST /login
* **Description**: Authenticates a user and initiates a session.
* **Request Body**:
* {
* "email": "user@example.com",
* "password": "securepassword"
* }
* **Responses**:
  + **200 OK**
  + {
  + "message": "Login successful",
  + "user": {
  + "username": "user123",
  + "email": "user@example.com",
  + "role\_id": 2
  + }
  + }
  + **401 Unauthorized**
  + {
  + "error": "Invalid credentials"
  + }
  + **500 Internal Server Error**
  + {
  + "error": "Login failed",
  + "details": "Error details"
  + }

**3. Logout User**

* **Endpoint**: POST /logout
* **Description**: Logs the user out by destroying the session and clearing the session cookie.
* **Request Headers**:
  + Ensure cookies are included in the request (for session-based authentication).
* **Responses**:
  + **200 OK**
  + {
  + "message": "Logout successful"
  + }
  + **401 Unauthorized**
  + {
  + "error": "Not authenticated"
  + }
  + **500 Internal Server Error**
  + {
  + "error": "Logout failed"
  + }

**Clearing the Session Cookie**

When the server calls:

res.clearCookie('connect.sid');

it instructs the **client (browser)** to **remove** the connect.sid cookie (the session cookie) using the default settings. This effectively logs the user out by ensuring the client no longer sends the session cookie in subsequent requests. In many basic setups, simply calling res.clearCookie('connect.sid') is enough to remove the cookie if the path and other settings match the defaults.

**4. Get Profile (Check Authentication Status)**

* **Endpoint**: GET /profile
* **Description**: Retrieves the logged-in user's profile information.
* **Request Headers**:
  + Ensure cookies are included in the request (for session-based authentication).
* **Responses**:
  + **200 OK**
  + {
  + "user": {
  + "id": 1,
  + "username": "user123",
  + "email": "user@example.com",
  + "role\_id": 2
  + }
  + }
  + **401 Unauthorized**
  + {
  + "error": "Not authenticated"
  + }

**Security Measures**

* **Password Hashing**: Argon2 used for secure password storage.
* **Session Management**: Uses cookies for authentication; data stored in PostgreSQL via connect-pg-simple.

**Authentication Flow**

1. **User Registration**
   * User submits username, email, and password
   * System hashes password with Argon2
   * Stores user in database
2. **User Login**
   * User submits email and password
   * System verifies credentials
   * Starts session and stores user data in req.session.user
3. **User Logout**
   * Session is destroyed on the backend
   * res.clearCookie('connect.sid') instructs the client to delete its session cookie
4. **User Profile Retrieval**
   * /profile returns session-stored user data if authenticated

**Best Practices**

* Always return explicit HTTP status codes (200 OK, 201 Created, etc.).
* Include meaningful error messages in responses.
* Use **secure cookies** (httpOnly, secure in production) for session management.
* If your frontend is on a different domain, ensure sameSite: 'None' so cookies function properly.