**Frontend API Documentation**

**Base URL**

The backend API is hosted at:

[***http://localhost:5000/auth***](http://localhost:5000/auth)

**REGISTER**

Endpoint: ***POST /auth/register***

Registerign a new user by sending a username, email, and password to the backend. The password is securely hashed before storage.

For example;

With axios;

A computer screen shot of a program

Description automatically generated

Status 201:

{

"message": "User registered successfully",

"user": {

"username": "testUser",

"email": "test@example.com"

}

}

Status 400 (Bad Request):

{

"error": "Username or email already exists"

}

Status 500 (Internal Server Error):

{

"error": "Registration failed",

"details": "Specific error message from server"

}

**LOGIN**

Endpoint: ***POST /auth/login***

Authenticates the user by verifying the **email** and **password**. If successful, logs in the user and **establishes a session.**

For example;

With axios;

A screen shot of a computer

Description automatically generated

Status 200 (OK):

{

"message": "Login successful"

}

Status 401 (Unauthorized):

{

"error": "Invalid credentials"

}

Status 500 (Internal Server Error):

{

"error": " Login failed",

"details": "Specific error message from server"

}

**LOGOUT**

Endpoint: ***POST /auth/logout***

Logs out the currently authenticated user by **destroying the session** and clearing the session cookie.

For example;

With axios;

export const logoutUser = async () => {

try {

const response = await axios.post('http://localhost:5000/auth/logout', {}, { withCredentials: true });

return response.data;

} catch (error) {

throw error.response?.data || { error: 'Logout failed' };

}

};

Status 200 (OK):

{

"message": " Logout successful"

}

Status 500 (Internal Server Error):

{

"error": " Logout failed",

}

**PROFİLE**

Endpoint: ***GET /auth/profile***

Retrieves the logged-in user's profile information **if authenticated**.

For example;

With axios;

export const getProfile = async () => {

try {

const response = await axios.post('http://localhost:5000/auth/profile', {}, { withCredentials: true });

return response.data;

} catch (error) {

throw error.response?.data || { error: ‘Failed to fetch profile’};

}

};

Status 200 (OK):

{

"user": {

"id": 1,

"username": "user123",

"email": "user@example.com",

"role\_id": 2

}

}

Status 401 (Unauthorized):

{

"error": " Not authenticated"

}

Status 500 (Internal Server Error):

{

"error": “Failed to fetch profile",

}

**Profile Endpoint (GET /auth/profile) Requires a Session:**

* The user must **log in first** to establish a session.
* Any request without a valid session returns 401 Unauthorized.

**Request Headers**

* The request must **include cookies** (session-based authentication).
* Axios handles this with { withCredentials: true }.

**Ensure CORS is handled correctly** if the frontend and backend run on different ports (React on 3000, API on 5000).

**Since the backend uses session-based authentication**, ensure the frontend sends and receives cookies:

axios.defaults.withCredentials = true;

**Security Recommendations**:

* Use HTTPS in production.
* Implement proper input validation.
* Consider using JWT authentication for scalability.

# **Frontend Usage Example (Register)**

export default Register;

import React, { useState } from 'react';

import { registerUser } from '../api/authService';

function Register() {

const [username, setUsername] = useState('');

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [message, setMessage] = useState('');

const handleRegister = async (e) => {

e.preventDefault();

try {

const response = await registerUser(username, email, password);

setMessage(response.message);

} catch (error) {

setMessage(error.error || 'Registration failed');

}

};

return (

<div>

<h1>Register</h1>

<form onSubmit={handleRegister}>

<input type="text" placeholder="Username" value={username} onChange={(e) => setUsername(e.target.value)} required />

<input type="email" placeholder="Email" value={email} onChange={(e) => setEmail(e.target.value)} required />

<input type="password" placeholder="Password" value={password} onChange={(e) => setPassword(e.target.value)} required />

<button type="submit">Register</button>

</form>

{message && <p>{message}</p>}

</div>

);

}

export default Register;