**TutorConnect : Backend Documentation – as of 2/11/25**

# API & Authentication Documentation

## Overview

This document outlines the backend functionality of the TutorConnect application. The backend is built using FastAPI and provides secure access to a database where users can perform basic actions such as registering, logging in, and retrieving tutor matches.

The system ensures authentication using JWT tokens and bcrypt password hashing, which protects user data and prevents unauthorized access.

It is important to note that the current implementation is primarily for **developer testing**; the final product will have a more user-friendly experience.

## Workflow

1. **User Registration:** Users create an account via the /register endpoint. Their email and **hashed** password are stored in the database.
2. **User Login:** Users log in through the /login endpoint. If their credentials are valid, a JWT token is generated.
3. **Accessing Protected Routes:** Users include their **JWT token in the request header** to access protected endpoints.
4. **JWT Verification:** The server validates the token, checks the user role, and grants access if the request is authorized.
5. **Token Validation:** If the token is missing, expired, or invalid, the request is rejected.

## Password Security

* **Passwords are securely hashed** using bcrypt before storage.
* **Passwords are never stored in plain text.**
* **During login, the stored hash is compared** with the provided password.

## A Note on JWT (JSON Web Tokens)

* JWT is a **stateless authentication mechanism** that encodes user data like ID and role.
* The generated token is sent back to the user **on successful login**.
* The token must be included in requests using the **Authorization header**:

Authorization: Bearer <insert\_token>

* Token expiration is set to **120 minutes** (2 hours).

## **API Endpoints**

### **Endpoint: POST /register**

**Registers a new user and stores their secure password in the database.**

#### Expected Request:

{

"name": "John Doe",

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

"password": "plaintextpassword",

"role": "student"

}

#### Expected Response:

{

"message": "User registered successfully",

"user\_id": 1

}

### **Endpoint: POST /login**

**Authenticates user credentials and returns a JWT token.**

#### Expected Request:

Username=example@example.com

Password=plaintextpassword

#### Expected Response:

{

"access\_token": "<insert\_token>",

"token\_type": "bearer"

}

* **Token must be included in requests to access protected routes.**
* If login fails (wrong email/password), the response is:

{

"detail": "Invalid credentials"

}

### **Endpoint: GET /protected-endpoint**

**Restricted to authenticated users.**

#### Expected Response (if authenticated):

{

"message": "You have accessed a protected route!",

"user": {"sub": 1, "role": "student"}

}

#### Expected Response (if unauthorized):

{

"detail": "Invalid or expired token"

}

### **Endpoint: GET /match/{user\_id}**

**Retrieves tutor matches for a student. Requires authentication and role verification.**

#### Security Rules:

Only **students** can request tutor matches.  
Users **can only request matches for their own user\_id** (no cross-account access).  
The request must include a **valid JWT token**.

#### Expected Request:

GET /match/1

Authorization: Bearer <insert\_token>

#### Expected Response (if authorized):

{

"matches": ["Tutor 1", "Tutor 2", "Tutor 3"]

}

#### Expected Response (if user is not a student):

{

"detail": "Only students can request matches"

}

#### Expected Response (if unauthorized access attempt):

{

"detail": "Access denied"

}

## **Final Notes**

* **JWT tokens expire after 120 minutes**, requiring re-authentication.
* **Authorization header must be included** in protected requests.
* **Only students can access** **/match/{user\_id}**, ensuring role separation.
* **The system prevents unauthorized access** by enforcing authentication and role-based restrictions.
* The code currently has multiple dependencies and packages. Run this command in your terminal to download the tools currently in the environment:

pip freeze > requirements.txt

This documentation will be updated as new features are added.