# API Documentation – User Registration & Login

## General Information

This document describes the User Registration and User Login APIs. These endpoints allow users to create an account and authenticate using email and password. After successful authentication or registration, a JWT token is issued to access protected resources.

## Authentication & Security

• The registration and login endpoints do not require authentication.  
• Both return a JWT token that must be included in the Authorization header as: 'Authorization: Bearer <token>' for all subsequent authenticated endpoints.  
• Tokens should be securely stored on the client side.

## Conventions

• All requests and responses use JSON format.  
• Field names use camelCase.  
• Date and time fields (if any) use ISO 8601 format.

## Endpoint Catalog

1. POST /api/user/register – Register a new user and return a JWT token.  
2. POST /api/user/login – Authenticate a user and return a JWT token.

## Endpoint Details

### 1. User Registration API

Method: POST

URL: http://13.48.130.179:4000/api/user/register

Authentication: Not required.

#### Request Body

- email (string): Email address of the user.  
- password (string): Password chosen by the user.

#### Example Request

{  
 "email": "user@example.com",  
 "password": "yourpassword"  
}

#### Successful Response

{  
 "success": true,  
 "message": "User registered successfully",  
 "token": "eyJhbGciOiJIUzI1NiIsInR..."  
}

#### Error Responses

Validation errors (400):  
{  
 "success": false,  
 "errors": [  
 "\"email\" is not allowed to be empty",  
 "\"password\" is not allowed to be empty"  
 ]  
}  
  
Duplicate email (400):  
{  
 "success": false,  
 "message": "Email already in use"  
}  
  
Server error (500):  
{  
 "success": false,  
 "message": "Internal server error"  
}

### 2. User Login API

Method: POST

URL: http://13.48.130.179:4000/api/user/login

Authentication: Not required.

#### Request Body

- email (string): Email address associated with the user account.  
- password (string): Password for the user account.

#### Example Request

{  
 "email": "user@example.com",  
 "password": "yourpassword"  
}

#### Successful Response

{  
 "success": true,  
 "message": "Login successful",  
 "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",  
 "user": {  
 "id": 15,  
 "name": "John Doe",  
 "email": "user@example.com",  
 "role": "JobSeeker"  
 }  
}

#### Error Responses

Email not found (400):  
{  
 "success": false,  
 "message": "Email not found",  
 "field": "email"  
}  
  
Incorrect password (400):  
{  
 "success": false,  
 "message": "Incorrect password",  
 "field": "password"  
}  
  
Validation errors (400):  
{  
 "success": false,  
 "errors": [  
 "\"email\" is not allowed to be empty",  
 "\"password\" is not allowed to be empty"  
 ]  
}

## Data Models

User Object:  
{  
 "id": number,  
 "name": string,  
 "email": string,  
 "role": string ("JobSeeker" | "Recruiter")  
}

## Use Cases / Workflows

• Register a new user and immediately receive a JWT token for authentication.  
• Login an existing user and receive a JWT token for accessing protected endpoints.  
• Store token securely on the client side and include in Authorization header for all authenticated APIs.

## Environments & Integration

Base URL: http://13.48.130.179:4000  
Integration Steps:  
1. Call POST /api/user/register or POST /api/user/login.  
2. Store the returned token securely.  
3. Use the token in Authorization headers for subsequent requests.

## Changelog

v1.0 – Added User Registration and User Login API documentation.