Task-11 API

This is a backend API built with **Node.js**, **Express.js**, **MongoDB**, and **JWT** for user authentication and management. It provides functionalities for user registration, login, and fetching user details.

The API is hosted on Render.

Register: https://task-11-vf7k.onrender.com/auth/register

Login: https://task-11-vf7k.onrender.com/auth/login

Get User: https://task-11-vf7k.onrender.com/auth/user

-- This structure includes the **Authorization** requirement for the `GET /auth/user` endpoint. The token should be passed in the `Authorization` header as `Bearer <JWT_TOKEN>`.

Testing

1. To register a user, send a POST request to /auth/register with the required fields.

2. To log in, send a POST request to /auth/login with the user's email and password.

3. To fetch the logged-in user's details, send a GET request to /auth/user with the Authorization header set to Bearer <JWT_TOKEN>.

Table of Contents

Technologies Used

- **Node.js**: JavaScript runtime environment.
- **Express.js**: Web framework for Node.js.
- **MongoDB**: NoSQL database.
- **Mongoose**: ODM for MongoDB.
- **JWT**: JSON Web Token for authentication.
- **Bcrypt.js**: Library for hashing passwords.

1. POST /auth/register

- Description: Register a new user.
- Request Body:

```
{
  "username": "exampleuser",
  "email": "user@example.com",
```

```
  "password": "password123",
  "role": "user"
}
```

- Response:

```
{
  "message": "User registered successfully"
}
```

2. POST /auth/login

- Description: Login a user and return a JWT token.
- Request Body:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

- Response:
-

```
{
  "token": "your_jwt_token",
  "message": "User logged in successfully"
}
```

3. GET /auth/user

- Description: Fetch the current logged-in user's details. Requires a valid JWT token.
- Headers:
    o Authorization: Bearer <JWT_TOKEN>
- Response:

```
{
  "username": "exampleuser",
  "email": "user@example.com",
  "role": "user"
}
```