**User Authentication System**

This project is a simple user authentication system built using Node.js, Express, MongoDB, JWT (JSON Web Tokens), and bcrypt. It includes user registration, login, and middleware for protected routes. The system securely stores user passwords and generates authentication tokens for user sessions.

**Features**

* **User Registration**: Users can sign up by providing a valid email, username, and password. The password is hashed using bcrypt before being stored in the MongoDB database.
* **User Login**: Users can log in using their email and password. Upon successful login, a JWT is generated and returned to the client.
* **Protected Routes**: The system includes middleware to protect routes that require authentication. The middleware verifies the JWT and extracts user information.
* **Token Storage**: JWTs are stored in HTTP-only cookies for secure client-side storage.

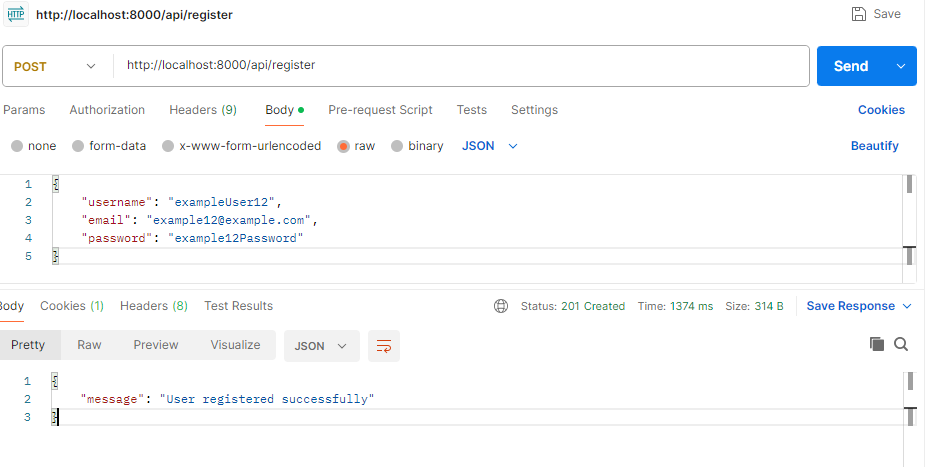
**Technologies Used**

* **Node.js**: JavaScript runtime for server-side development.
* **Express.js**: Web framework for building the API.
* **MongoDB:** NoSQL database for storing user data.
* **Mongoose:** ODM for MongoDB, providing schema validation and query building.
* **JWT (JSON Web Token):** For secure token-based authentication.
* **bcrypt:** Library for hashing passwords.

**API Documentation**

#### 1. ****User Registration****

* **Endpoint:** POST /api/register
* **Description:** Registers a new user by storing their username, email, and password in the database.
* **Authentication Required:** No
* **Request Format:**
  + **Content-Type:** application/json
* **Required Parameters:**
  + username (string): The username of the user.
  + email (string): A valid email address. Must be unique.
  + password (string): The user's password.



**Expected Responses:**

**201 Created:**

{

"message": "User registered successfully"

}

**409 Conflict:**

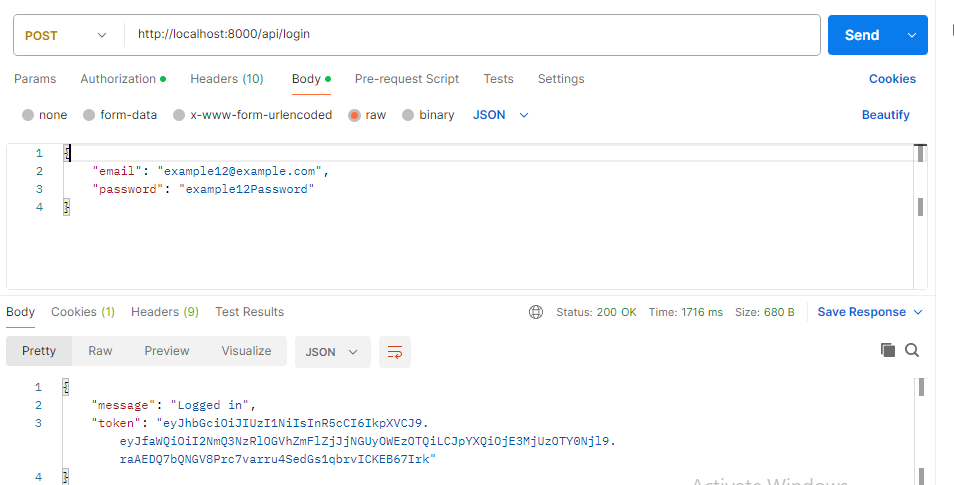
{

"error": "Email already exists"

}

#### 2. ****User Login****

* **Endpoint:** POST /api/login
* **Description:** Authenticates a user and returns a JWT for accessing protected routes.
* **Authentication Required:** Yes
* **Request Format:**
  + **Content-Type:** application/json
* **Required Parameters:**
  + email (string): The email address associated with the user's account.
  + password (string): The user's password.



**Expected Responses:**

**200 OK:**{

"message": "Logged in successfully",

"token": "your-jwt-token"

}

**400 Bad Request:**

{

"error": "Please fill all fields"

}

**401 Unauthorized:**

{

"error": "Invalid email or password"

}