**Database Structure**

**User Table**

CREATE TABLE user\_tbl (

user\_id UUID PRIMARY KEY,

user\_name VARCHAR(255) NOT NULL,

user\_email VARCHAR(255) UNIQUE NOT NULL,

user\_password VARCHAR(255) NOT NULL

);

**Post Table**

CREATE TABLE post\_tbl (

post\_id UUID PRIMARY KEY,

post\_title VARCHAR(255) NOT NULL,

post\_content TEXT NOT NULL,

user\_id UUID NOT NULL,

FOREIGN KEY (user\_id) REFERENCES user\_tbl(user\_id)

);

**Like Table**

CREATE TABLE like\_tbl (

like\_id UUID PRIMARY KEY,

user\_id UUID NOT NULL,

post\_id UUID NOT NULL,

FOREIGN KEY (user\_id) REFERENCES user\_tbl(user\_id),

FOREIGN KEY (post\_id) REFERENCES post\_tbl(post\_id)

);

**Explanation of Entity Relationships**

1. **User - Post:** A User can have multiple Posts. This is a **One-to-Many** relationship. The **User** entity has a Set<Post> field, and the **Post** entity has a **ManyToOne** relation to **User**.
2. **Post - Like:** A Post can have multiple Likes, and each Like is linked to a specific Post. This is a **One-to-Many** relationship. The **Like** entity has a **ManyToOne** relation to **Post**.
3. **User - Like:** A User can like many posts, and each Like is associated with a User. This is a **One-to-Many** relationship. The **Like** entity has a **ManyToOne** relation to **User**.

**API Endpoints**

* **POST /api/v1/user**: Creates a new user. Returns 201 Created if successful.
* **PUT /api/v1/user/{userId}**: Updates an existing user. Returns 200 OK if successful.
* **DELETE /api/v1/user/{userId}**: Deletes a user. Returns 204 No Content if successful.
* **GET /api/v1/user/{userId}**: Retrieves a user by ID. Returns 200 OK with user data or 404 Not Found if not found.
* **GET /api/v1/user**: Retrieves all users. Returns 200 OK with a list of users.

**Error Codes:**

* **400 Bad Request:** Returned for invalid input (e.g., missing or invalid data).
* **404 Not Found:** Returned when the user cannot be found by the provided ID.
* **500 Internal Server Error:** Returned for unexpected errors.

**Example Request and Response:**

**Create User (POST /api/v1/user)**

**Request Body:**

{

"userName": "John Doe",

"userEmail": "john.doe@example.com",

"userPassword": "password123"

}

**Response:**

{

"userId": "b2b62b4f-3cc6-4b29-9146-4285cb100fa3",

"userName": "John Doe",

"userEmail": "john.doe@example.com"

}

**Update User (PUT /api/v1/user/{userId})**

**Request Body:**

{

"userName": "John Updated",

"userEmail": "john.updated@example.com",

"userPassword": "newpassword123"

}

**Response:**

{

"userId": "b2b62b4f-3cc6-4b29-9146-4285cb100fa3",

"userName": "John Updated",

"userEmail": "john.updated@example.com"

}

**Steps**

1. **Test Login API (POST /api/v1/login):**
   * Send { "email": "test@example.com", "password": "password123" }
   * Response: { "jwtToken": "generated\_token\_here" }
2. **Use JWT Token in Headers for Other APIs:**
   * Include Authorization: Bearer <jwt\_token> in all requests.
3. **Notes:**
   * User, Post, and Like APIs now require a JWT token in the Authorization header.
   * Extracted user email from JWT using SecurityContextHolder.
   * Ensured only authenticated users can access protected endpoints.
   * Updated SecurityConfig to enforce JWT authentication.

**API Endpoints Overview**

1. **User Registration API (POST /api/v1/register)**  
   **Endpoint:**  
   POST http://localhost:8081/api/v1/register

**Request Body:**

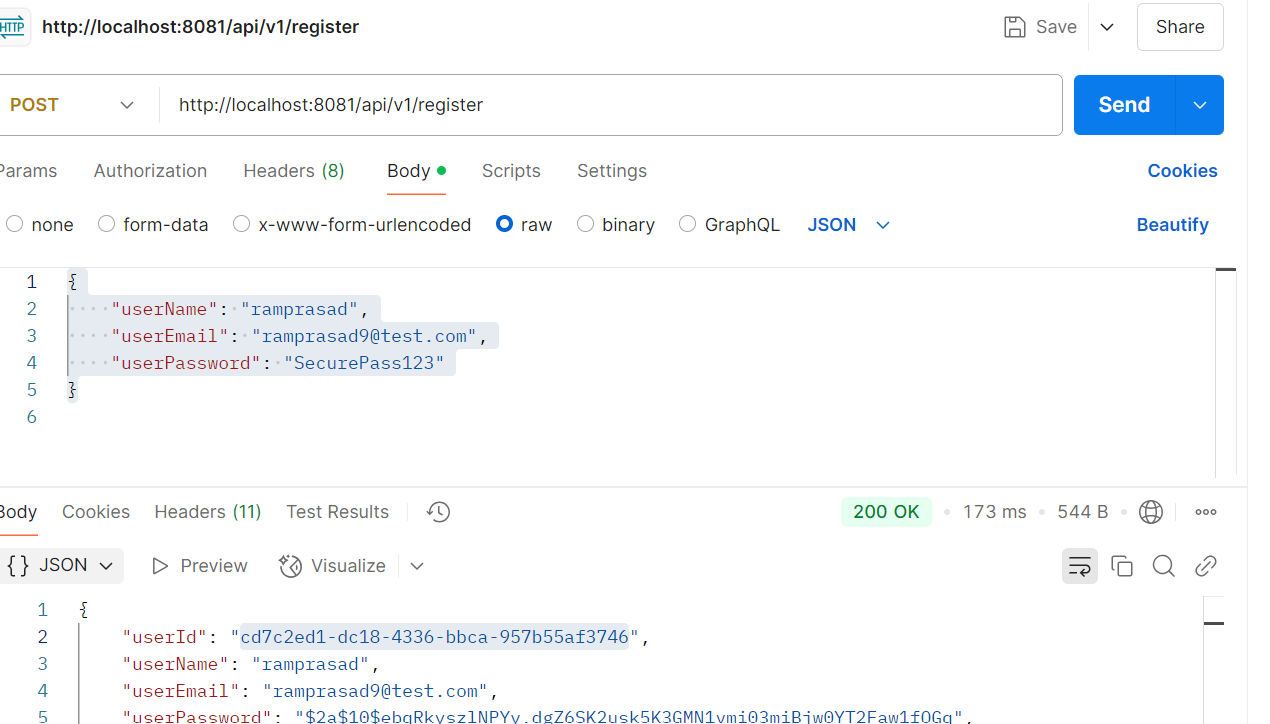
{

"userName": "ramprasad",

"userEmail": "ramprasad9@test.com",

"userPassword": "SecurePass123"

}

****

1. **User Login API (POST /api/v1/login)**  
   **Endpoint:**  
   POST http://localhost:8081/api/v1/login

**Request Body:**

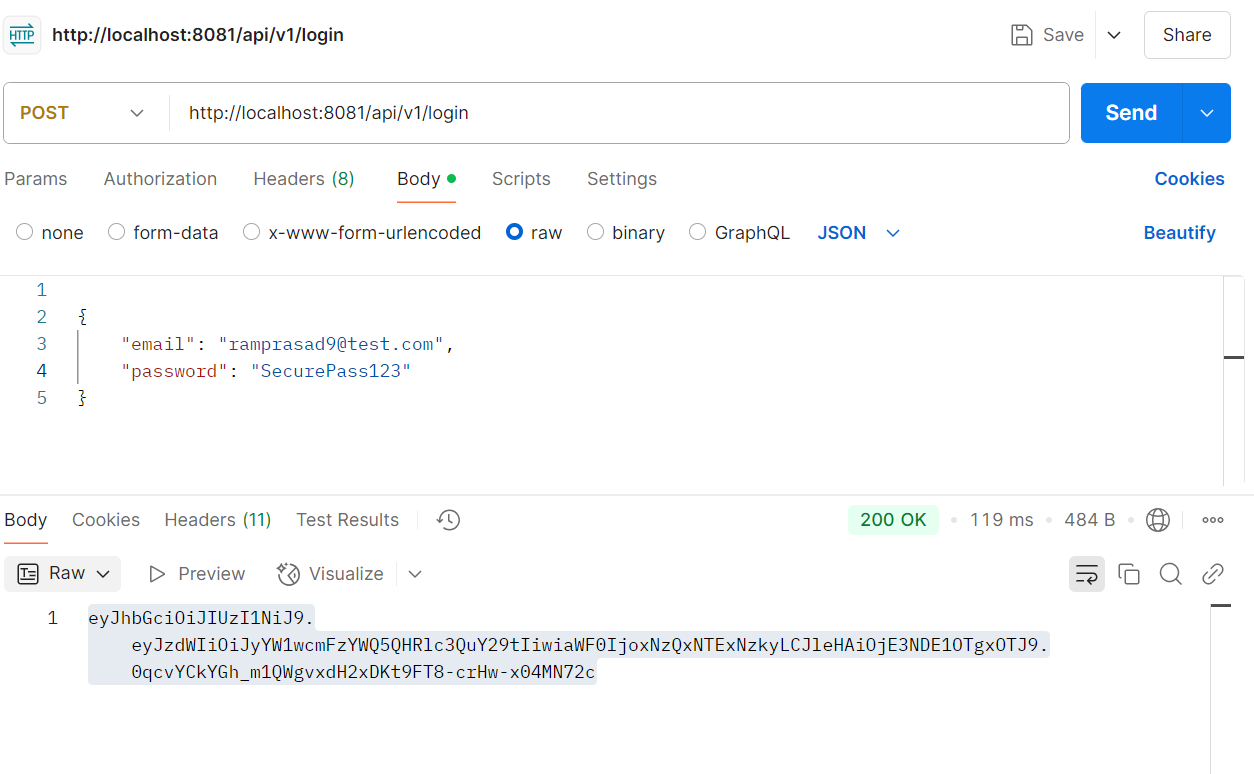
{

"email": "ramprasad9@test.com",

"password": "$2a$10$ebqRkyszlNPYy.dgZ6SK2usk5K3GMN1vmi03miBjw0YT2Faw1fOGq"

}

**Response (Token):**



1. **Create User API (POST /api/v1/user/create)**  
   **Endpoint:**  
   POST http://localhost:8081/api/v1/user/create

**Request Body:**

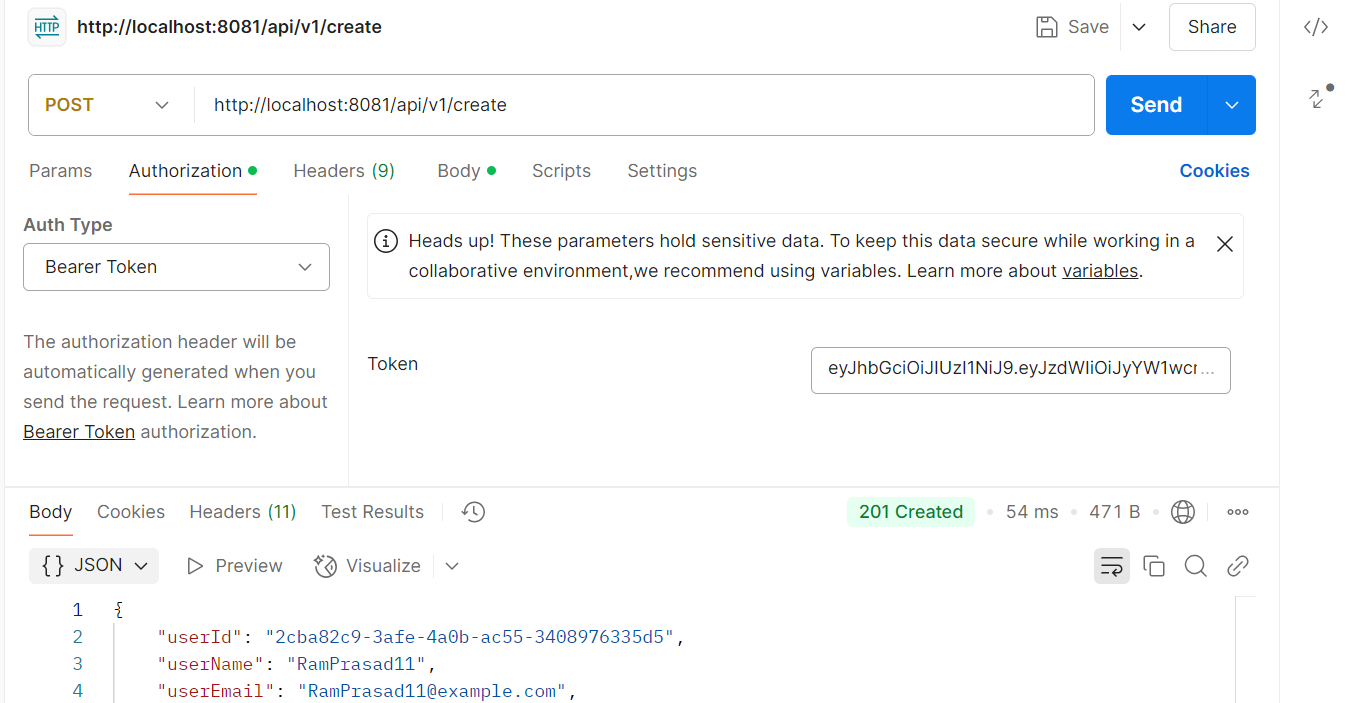
{

"userName": "RamPrasad11",

"userEmail": "RamPrasad11@example.com",

"userPassword": "SecurePass123"

}

****

1. **Update User API (PUT /api/v1/user/{userId})**  
   **Endpoint:**  
   PUT http://localhost:8081/api/v1/user/{userId}

**Request Body (JSON):**

{

"userName": "John Doe Updated",

"userEmail": "johndoe\_updated@example.com",

"userPassword": "NewSecurePass123"

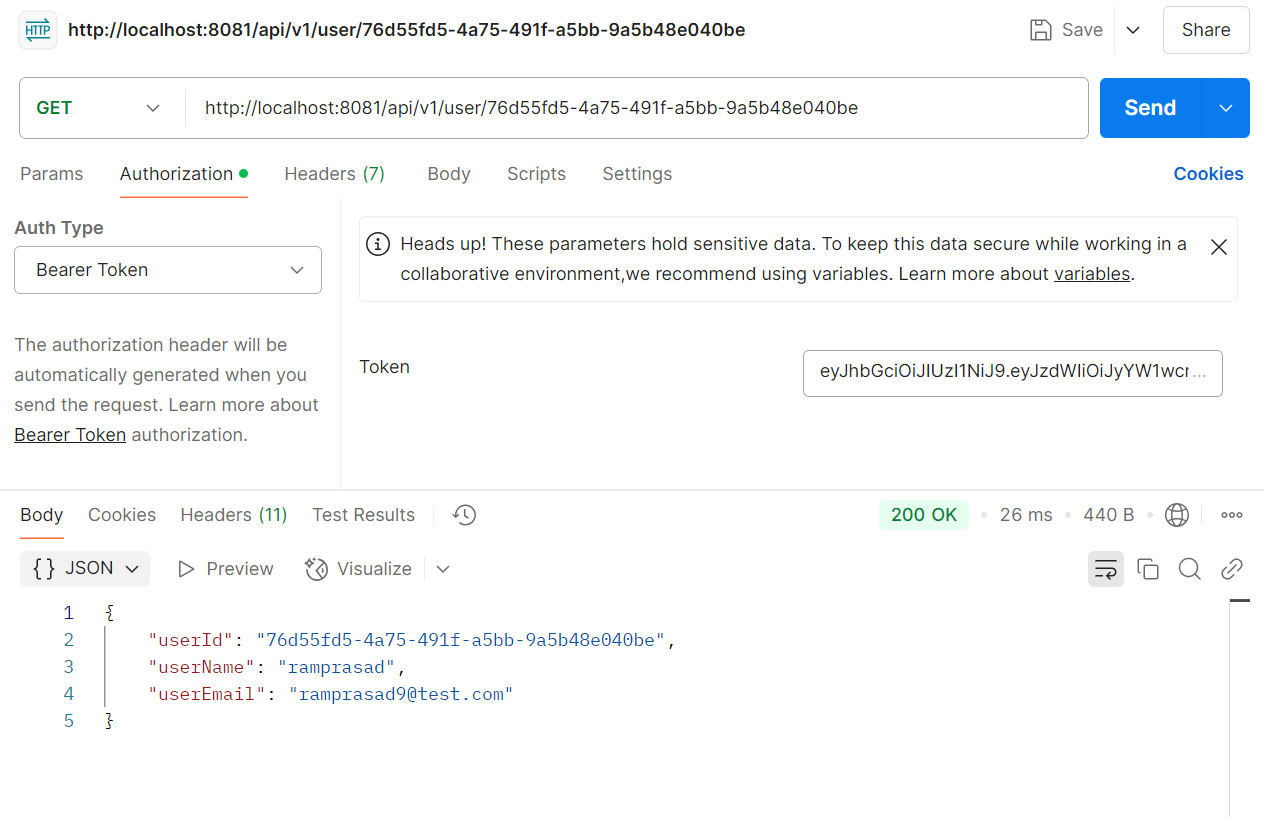
}

1. **Delete User API (DELETE /api/v1/user/{userId})**  
   **Endpoint:**  
   DELETE http://localhost:8081/api/v1/user/{userId}

**Example URL:**  
DELETE http://localhost:8081/api/v1/user/550e8400-e29b-41d4-a716-446655440000

1. **Get User by ID API (GET /api/v1/user/{userId})**  
   **Endpoint:**  
   GET http://localhost:8081/api/v1/user/{userId}

**Example URL:**  
GET <http://localhost:8081/api/v1/user/76d55fd5-4a75-491f-a5bb-9a5b48e040be>



1. **Get All Users API (GET /api/v1/user)**  
   **Endpoint:**  
   GET http://localhost:8081/api/v1/user

**Create Post:**

**1. Create a Post**

* **URL:** POST /api/v1/post

**Request Body:**

{

"userId": "6d4eead8-bf23-4f8b-b9e2-cf8fd2542da6",

"postTitle": "My First Post",

"postContent": "This is the content of my first post!"

}

**2. Get a Post by ID**

* **URL:** GET /api/v1/post/{postId}

**Request Example:**

* **URL:**  
  GET http://localhost:8080/api/v1/post/9c9d0ae2-86f1-42e6-8c8c-7fd64b8a4f89

**3. Delete a Post by ID**

* **URL:** DELETE /api/v1/post/{postId}

**Request Example:**

* **URL:**  
  DELETE http://localhost:8080/api/v1/post/9c9d0ae2-86f1-42e6-8c8c-7fd64b8a4f89

**Explanation:**  
This endpoint deletes a post based on its postId. No request body is required.

**4. Get All Posts by User**

* **URL:** GET /api/v1/post/user/{userId}

**Request Example:**

* **URL:**  
  GET http://localhost:8080/api/v1/post/user/6d4eead8-bf23-4f8b-b9e2-cf8fd2542da6

**Like:**

**1. Like a Post**

* **URL:** POST /api/v1/like

**Request Body:**

{

"userId": "6d4eead8-bf23-4f8b-b9e2-cf8fd2542da6",

"postId": "9c9d0ae2-86f1-42e6-8c8c-7fd64b8a4f89"

}

* **userId:** The ID of the user who is liking the post.
* **postId:** The ID of the post being liked.

**2. Get Like Counts for Multiple Posts**

* **URL:** GET /api/v1/like/count

**Request Example:**

* **URL:**  
  GET http://localhost:8080/api/v1/like/count?postIds=9c9d0ae2-86f1-42e6-8c8c-7fd64b8a4f89,6d4eead8-bf23-4f8b-b9e2-cf8fd2542da6

The postIds are passed as query parameters (comma-separated).

**3. Get Users Who Liked a Specific Post**

* **URL:** GET /api/v1/like/users/{postId}

**Request Example:**

* **URL:**  
  GET http://localhost:8080/api/v1/like/users/9c9d0ae2-86f1-42e6-8c8c-7fd64b8a4f89