**Attention**

**All requests start with "http://****192.168.101.204:7282/api/Users/The Action you need" (for user oprations)**

**All requests start with**

**"http://192.168.101.204:7282/api/** **AuthenticationForToken/The Action you need" (for user Authentication)**

**Authentication**

All endpoints require authentication. Make sure to include the JWT token in the Authorization header for each request.

**Endpoint** : POST authenticate

**Description :** It gives you a token that you can use to authenticate

**Response** :

* 500 Internal Server Error: Indicates a server error.
* 200 OK: Return token api with 24 hour Expire Time
* Unauthorized, if you don’t have permission

**Example Request:**

Post http:// 192.168.101.204:7282/api/AuthenticationForToken/authenticate

Content-Type: application/json

{

"UserName": "mahan",

"Password": "mahan.z.road0908"

}

**Endpoints**

**1. Get Full Information of All Users**

**Endpoint:** GET /FullInforMation

**Description:** Retrieves full information of all users from the database.

**Response:**

* 200 OK: Returns a list of user information.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

GET http:// 192.168.101.204:7282/api/Users/FullInforMation

Authorization: Bearer <JWT\_TOKEN>

**2. Get Full Information of a User by Username and Email**

**Endpoint:** GET /FullInforMationWithUserName

**Description:** Retrieves user information based on the provided username and email.

**Request Body:**

* UserName (string): The username of the user.
* Email (string): The email of the user.

**Response:**

* 200 OK: Returns the user information.
* 404 Not Found: Indicates the user was not found.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

GET http:// 192.168.101.204:7282/api/Users/FullInforMationWithUserName

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"UserName": "exampleUser",

"Email": "example@example.com"

}

**3. Get Users Without Password**

**Endpoint:** GET /GetUserWithoutPassword

**Description:** Retrieves a list of users without including their passwords.

**Response:**

* 200 OK: Returns a list of user information without passwords.

**Example Request:**

GET http:// 192.168.101.204:7282/api/Users/GetUserWithoutPassword

Authorization: Bearer <JWT\_TOKEN>

**4. Register a New User**

**Endpoint:** POST /RegisterUser

**Description:** Registers a new user in the system.

**Request Body:**

* UserName (string): The username of the new user.
* Email (string): The email of the new user.
* Password (string): The password of the new user.
* Bio (string): The Biography of the new user.
* RePassword (string): The RePassword must be like Password.
* IsActive (bool): Indicates if the user account is active. (optional)

**Response:**

* 200 OK: Returns registration success information and activation link.
* 400 Bad Request: Indicates validation errors or if the username/email already exists.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

POST http:// 192.168.101.204:7282/api/Users/RegisterUser

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"UserName": "newUser",

"Email": "newuser@example.com",

"Password": "Password123!",

"RePassword": "Password123!",

"IsActive": false(optional)

}

**5. Partially Update a User (You can Skip This Because we have another update with PUT action)**

**Endpoint:** PATCH /PartialUpdate/{id}

**Description:** Partially updates user information.

**Request Parameters:**

* id (int): The ID of the user to update.

**Request Body:**

* JSON Patch Document: Contains the fields to be updated.

**Response:**

* 204 No Content: Indicates the update was successful.
* 400 Bad Request: Indicates validation errors.
* 404 Not Found: Indicates the user was not found.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

PATCH http:// 192.168.101.204:7282/api/Users/PartialUpdate/1

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json-patch+json

[

{ "op": "replace", "path": "/Email", "value": "updatedemail@example.com" }

]

**6. Fully Update a User**

**Endpoint:** PUT /FullUpdate

**Description:** Fully updates user information.

**Request Body:**

* UserName (string): The username of the user.
* Email (string): The new email of the user.
* Bio (string): The new bio of the user.

**Response:**

* 200 OK: Returns the update success information.
* 400 Bad Request: Indicates validation errors.
* 404 Not Found: Indicates the user was not found.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

PUT http:// 192.168.101.204:7282/api/Users/FullUpdate

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"UserName": "existingUser",

"Email": "newemail@example.com",

"Bio": "This is the updated bio."

}

**7. Delete a User by Username and Email**

**Endpoint:** DELETE /DeletUserWithUserName

**Description:** Deletes a user based on the provided username and email.

**Request Body:**

* UserName (string): The username of the user.
* Email (string): The email of the user.

**Response:**

* 200 OK: Indicates the user was successfully deleted.
* 404 Not Found: Indicates the user was not found.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

DELETE http:// 192.168.101.204:7282/api/Users/DeletUserWithUserName

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"UserName": "exampleUser",

"Email": "example@example.com"

}

**8. Reset User Password**

**Endpoint:** PATCH /ResetUserPassword/{activeCode}

**Description:** Resets the user's password using the provided active code.

**Request Parameters:**

* activeCode (string): The active code for resetting the password.

**Request Body:**

* JSON Patch Document: Contains the new password field.

**Response:**

* 204 No Content: Indicates the password was successfully reset.
* 400 Bad Request: Indicates validation errors.
* 404 Not Found: Indicates the user was not found.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

PATCH http:// 192.168.101.204:7282/api/Users/ResetUserPassword/ABC123

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json-patch+json

[

{ "op": "replace", "path": "/Password", "value": "NewPassword123!" }

]

**9. Activate User Account**

**Endpoint:** POST /ActiveUserAccount/{Id}

**Description:** Activates a user account using the provided active code.

**Request Parameters:**

* Id (string): The active code for the user account.

**Response:**

* 200 OK: Indicates the account was successfully activated.
* 400 Bad Request: Indicates validation errors.
* 500 Internal Server Error: Indicates a server error.

**Example Request:**

POST http://192.168.101.204:7282/api/Users/ActiveUserAccount/ABC123

Authorization: Bearer <JWT\_TOKEN>

**10. Login User**

**Endpoint:** POST /LoginUser

**Description:** Authenticates a user and returns a JWT token if successful.

**Request Body:**

* Email (string): The email of the user.
* Password (string): The password of the user.

**Response:**

* 200 OK: Returns login success information and JWT token.
* 400 Bad Request: Indicates validation errors or inactive account.
* 404 Not Found: Indicates the user was not found.

**Example Request:**

POST http://192.168.101.204:7282/api/Users/LoginUser

Authorization: Bearer <JWT\_TOKEN>

Content-Type: application/json

{

"Email": "example@example.com",

"Password": "Password123!"

}