USER MANAGEMENT PROJECT
(FINAL REFLECTION DOCUMENTATION)

INSTRUCTOR: **KEITH WILLIAMS**

COURSE:

IS601: WEB SYSTEMS DEVELOPMENT

STUDENT DETAILS:

NAME: **KRISHNA SATHVIKA GANNI**

NJIT ID: **31715411**

📝 MY LEARNINGS FROM THIS COURSE:

- I have learned many practical skills in this course, starting with how to use GitHub effectively for version control and collaboration.
- The course introduced me to a lot of tools and ideas that I did not know about before, and made me learn a lot more about building, testing, and deploying realistic applications.
- The midterm calculator project was a stepping block that enabled me to get acquainted with the development process and prepare myself for the challenges of this Epic User Management project.

💼 MY EXPERIENCE WORKING ON THIS PROJECT:

- Throughout the User Management Project, I gained first-hand experience in debugging, clean code writing, user authentication, media upload handling, and integration with third-party storage services like Minio.
- I also understood the importance of writing thorough test cases and ensuring quality with automated tests and issue tracking.
- This experience not only improved my technical skills but also made me understand how to approach problems in a systematic manner and work as part of a team.
- Most significantly, I achieved a very high level of confidence in creating production-grade applications that are secure, sustainable, and ready for production.

🛠 PROJECT SETUP:

1. **FORK AND CLONE THE REPOSITORY:** ```git clone https://github.com/Krishna-Sathvika-Ganni/FINAL_PROJECT_USER_MANAGEMENT.git```

2. **ENVIRONMENT CONFIGURATION:**
   - Create a local `.env` file.
   - And add the MailTrap SMTP settings for testing email features

3. **DATABASE MIGRATION:** ```docker compose exec fastapi alembic upgrade head```

4. **Start the project with Docker**: ```docker compose up --build```

5. **ACCESS POINTS**

- **User Management API:** `localhost/docs`
- **PGAdmin:** `localhost:5050`
- **Minio Console:** `localhost:9001`

6. **USEFUL COMMANDS**

- To view logs for FastAPI service: ```docker compose logs fastapi -f```

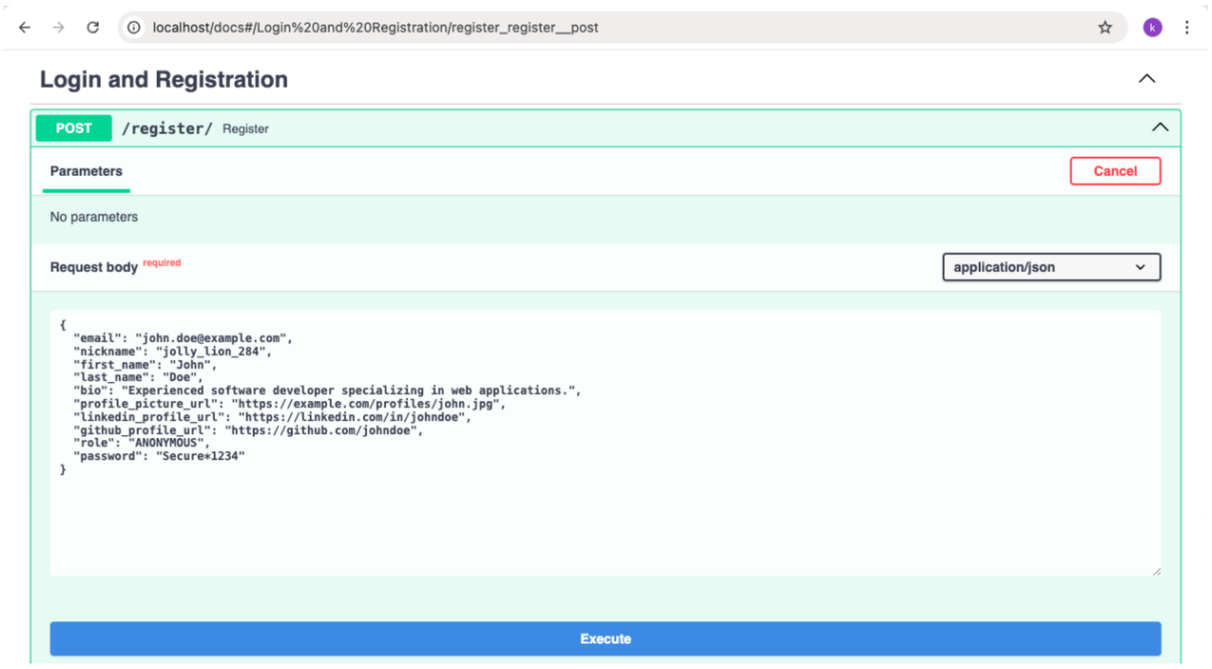
- Execute all tests inside the container: ```docker compose exec fastapi pytest```

🐞 **QA ISSUES:**

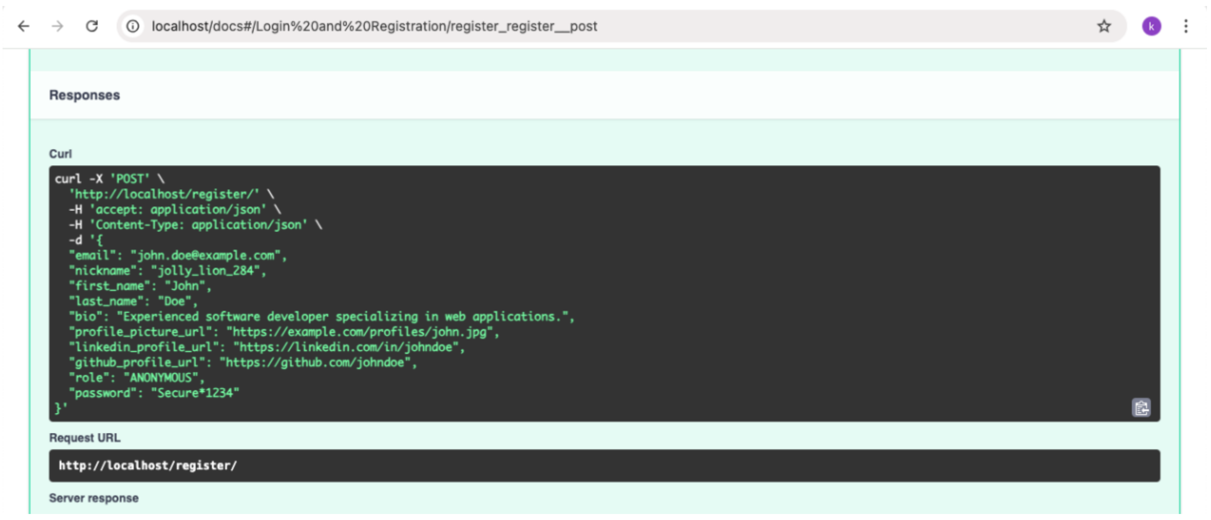
🔄 **ISSUE 1: Mismatch of the Nickname**

Description:

- The issue is that the Create User API doesn't match the nickname stored in the database input to the request.
- It is also seen as inconsistent with API reaction and example values.
- The Nickname gets overwritten during processing, leading to inconsistencies between input, stored data and output.



```
{
  "email": "john.doe@example.com",
  "nickname": "jolly_lion_284",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "password": "Secure*1234"
}
```


localhost/docs#/Login%20and%20Registration/register_register__post

Responses

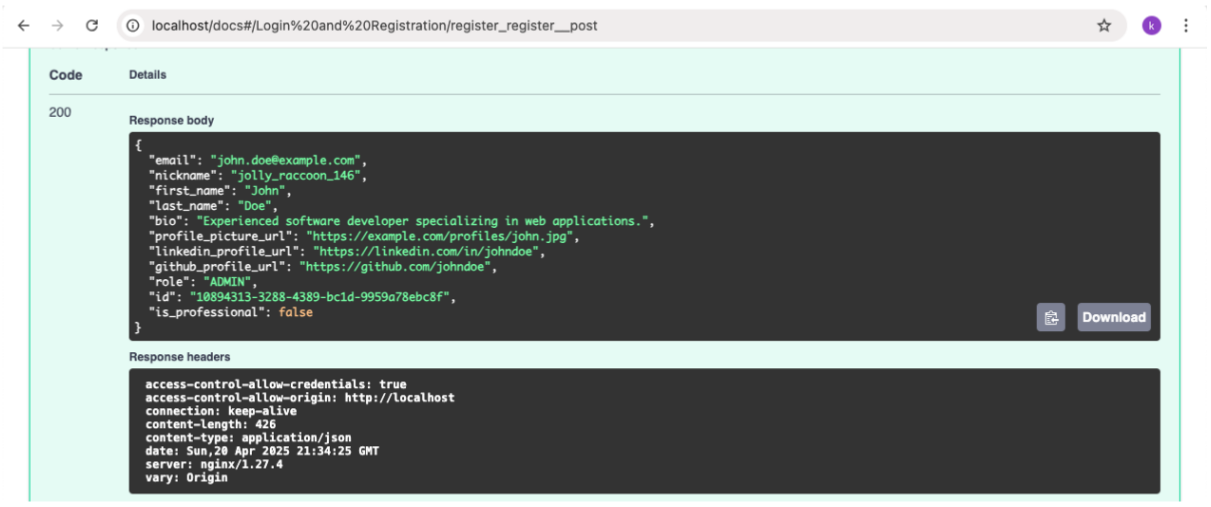
Curl

```
curl -X 'POST' \
  'http://localhost/register/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "john.doe@example.com",
  "nickname": "jolly_lion_284",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "password": "Secure*1234"
}'
```

Request URL

```
http://localhost/register/
```

Server response

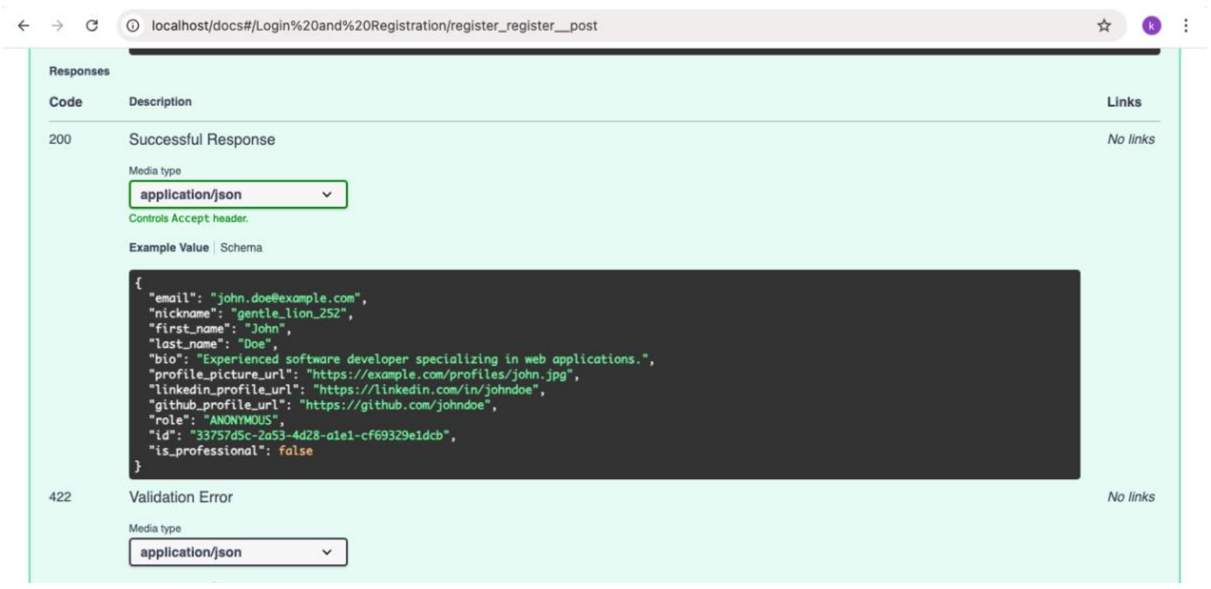

localhost/docs#/Login%20and%20Registration/register_register__post
Download

Code	Details
200	Response body

```
{
  "email": "john.doe@example.com",
  "nickname": "jolly_raccoon_146",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ADMIN",
  "id": "10894313-3288-4389-bc1d-9959a78ebc8f",
  "is_professional": false
}
```

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost
connection: keep-alive
content-length: 426
content-type: application/json
date: Sun,20 Apr 2025 21:34:25 GMT
server: nginx/1.27.4
vary: Origin
```


localhost/docs#/Login%20and%20Registration/register_register__post

Responses

Code	Description	Links
200	Successful Response	*No links*

Media type: application/json

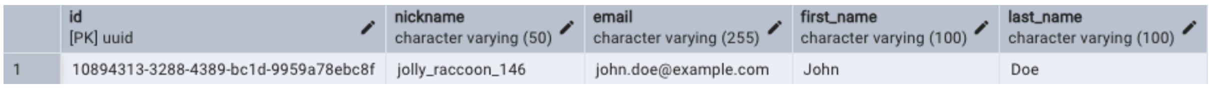
Controls Accept header.

Example Value | Schema

```
{
  "email": "john.doe@example.com",
  "nickname": "gentle_lion_252",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "id": "33757d5c-2a53-4d28-a1e1-cf69329e1dcb",
  "is_professional": false
}
```

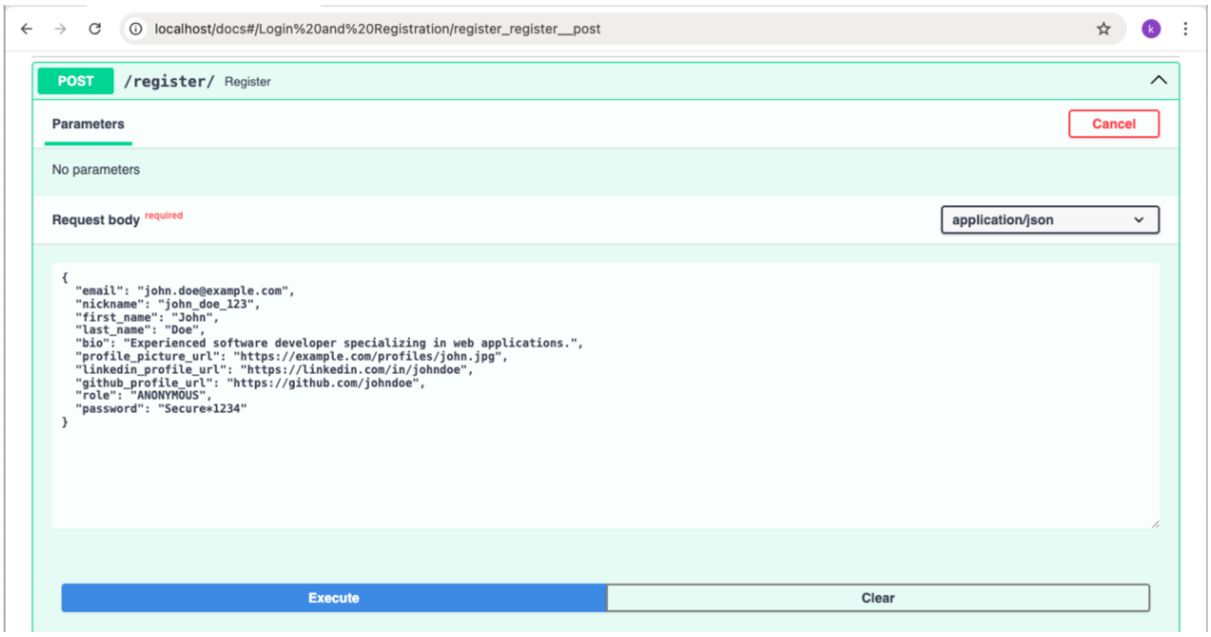
Code	Description	Links
422	Validation Error	*No links*

Media type: application/json

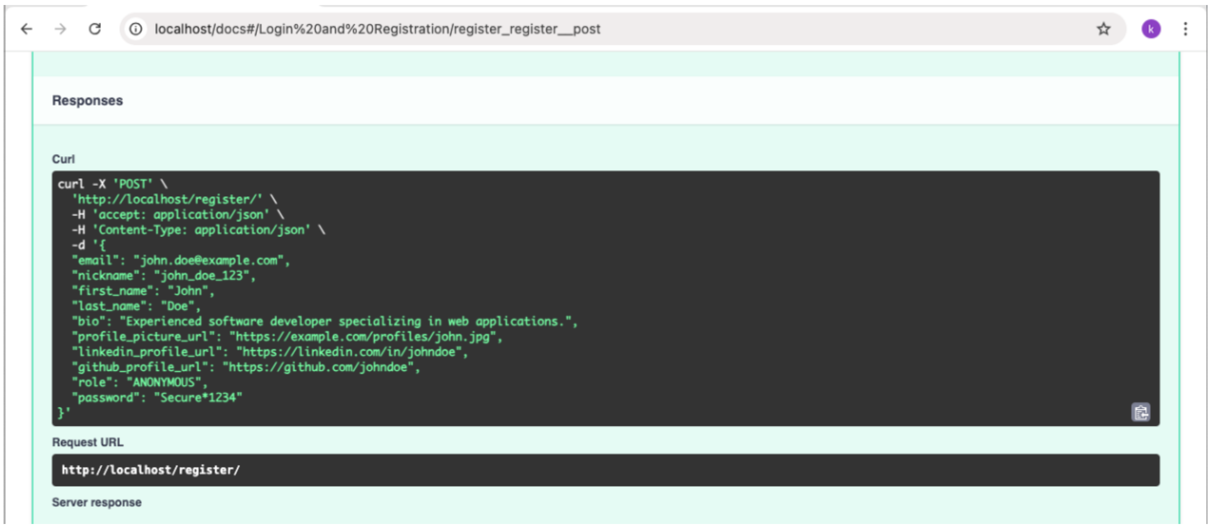


	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)
1	10894313-3288-4389-bc1d-9959a78ebc8f	jolly_raccoon_146	john.doe@example.com	John	Doe

Resolved Issue:

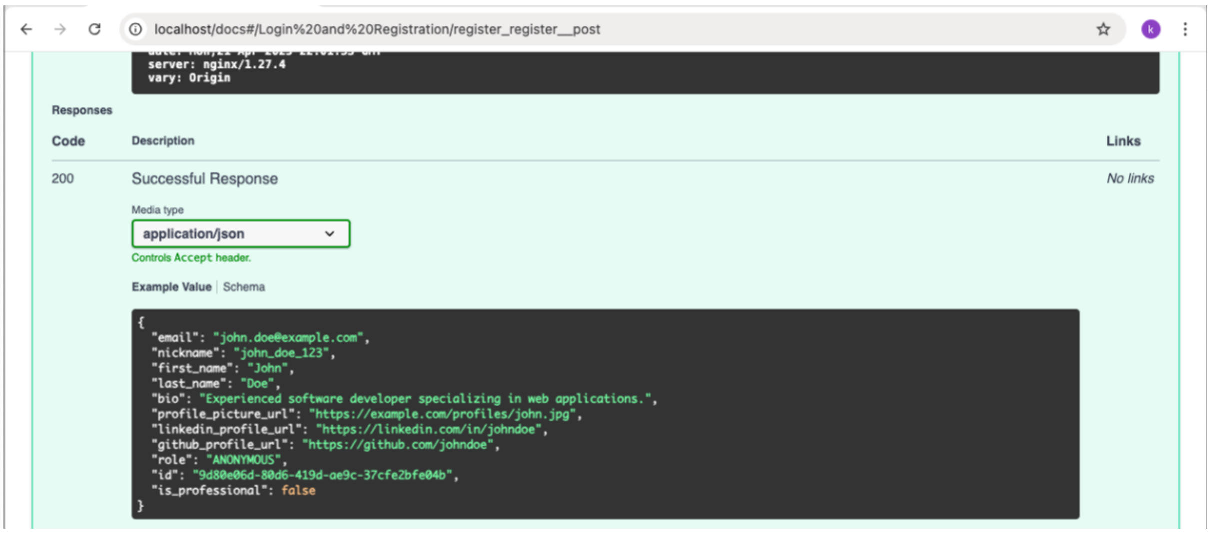


```
{
  "email": "john.doe@example.com",
  "nickname": "john_doe_123",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "password": "Secure*1234"
}
```


localhost/docs#/Login%20and%20Registration/register_register__post
Responses
Curl
Request URL
Server response

```
curl -X 'POST' \
  'http://localhost/register/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "email": "john.doe@example.com",
  "nickname": "john_doe_123",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "password": "Secure*1234"
}'
```

```
http://localhost/register/
```



localhost/docs#/Login%20and%20Registration/register_register__post

```
server: nginx/1.27.4
vary: Origin
```

Responses

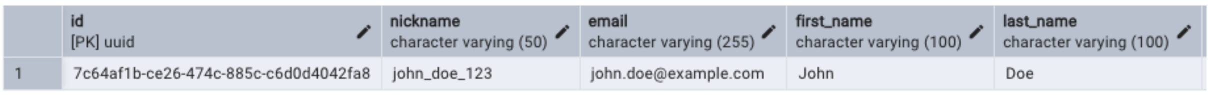
Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls Accept header.

Example Value | Schema

```
{
  "email": "john.doe@example.com",
  "nickname": "john_doe_123",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "id": "9d80e06d-80d6-419d-ae9c-37cfe2bfe04b",
  "is_professional": false
}
```

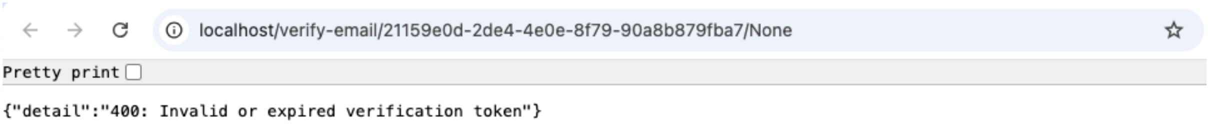


	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)
1	7c64af1b-ce26-474c-885c-c6d0d4042fa8	john_doe_123	john.doe@example.com	John	Doe

🖂 ISSUE 2: User ID passed as None in the Verification Email

Description:

- At the time of user registration, the email verification link was generated with a None user ID.
- Thus the email link becomes invalid or incomplete.

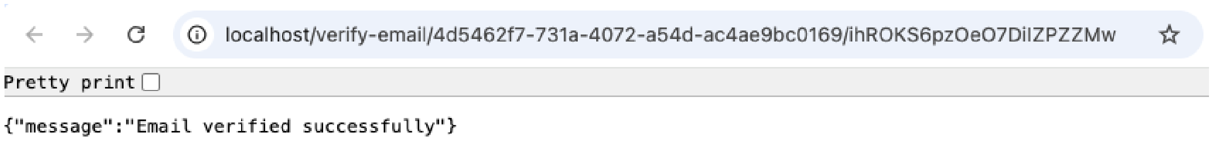


localhost/verify-email/21159e0d-2de4-4e0e-8f79-90a8b879fba7/None

Pretty print ☐

```
{"detail":"400: Invalid or expired verification token"}
```

Resolved Issue:

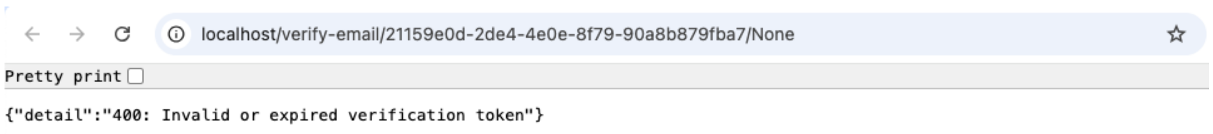


```
localhost/verify-email/4d5462f7-731a-4072-a54d-ac4ae9bc0169/ihROKS6pzOeO7DiIZPZZMw
Pretty print
{"message":"Email verified successfully"}
```

🔍 ISSUE 3: Verification Token Missing or Invalid

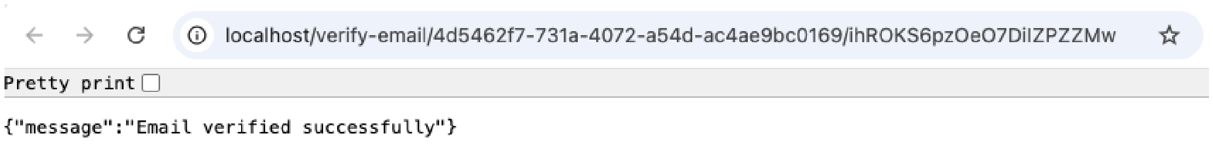
Description:

- User is getting a 400: Invalid or expired token for verification error.
- The verification URL is also "None," i.e., token is not being generated, not being stored, or is not being placed properly inside email verification link.



```
localhost/verify-email/21159e0d-2de4-4e0e-8f79-90a8b879fba7/None
Pretty print
{"detail":"400: Invalid or expired verification token"}
```

Resolved Issue:



localhost/verify-email/4d5462f7-731a-4072-a54d-ac4ae9bc0169/ihROKS6pzOeO7DiIZPZZMw

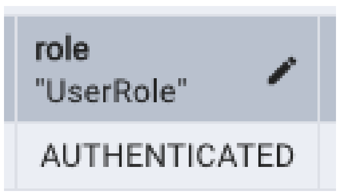
Pretty print

```
{"message":"Email verified successfully"}
```

👮 ISSUE 4: Role Changes from Admin to Authenticated

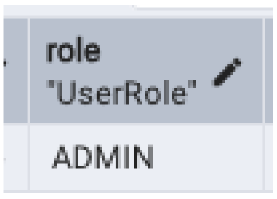
Description:

- Whenever an admin user's email was verified, their role was automatically downgraded to "Authenticated".
- This access control violation was resolved by fixing the buggy logic to preserve the original role.



role "UserRole"
AUTHENTICATED

Resolved Issue:



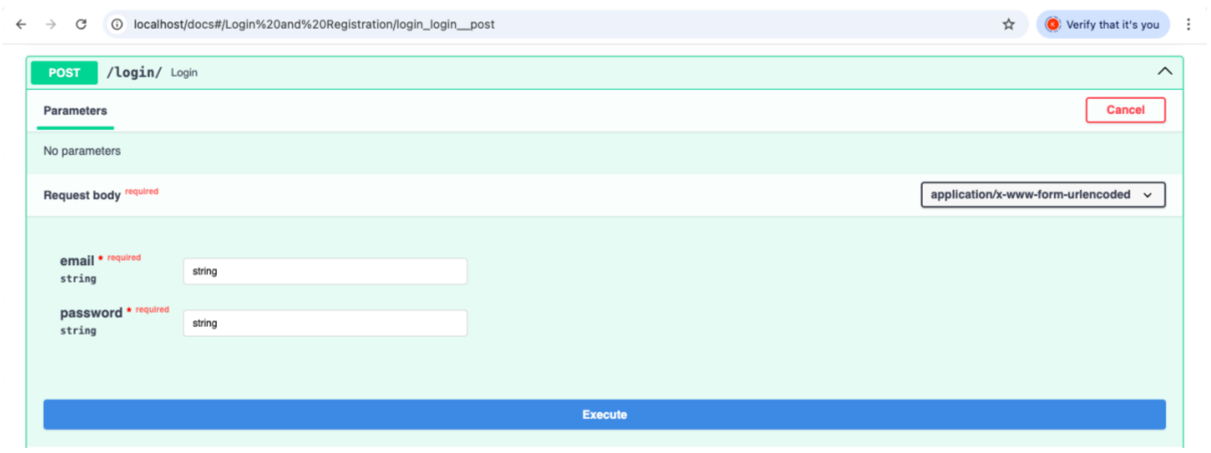
role "UserRole"
ADMIN

🔐 ISSUE 5: Confusing Login Prompt Asking for "Username" Instead of "Email"

Description:

- The login prompt asked for a "username" when the system was looking for an email.
- This inconsistency caused login errors and confusion.

Resolved Issue:


localhost/docs#/Login%20and%20Registration/login_login__post

POST /login/ Login

Parameters — No parameters

Request body required — application/x-www-form-urlencoded

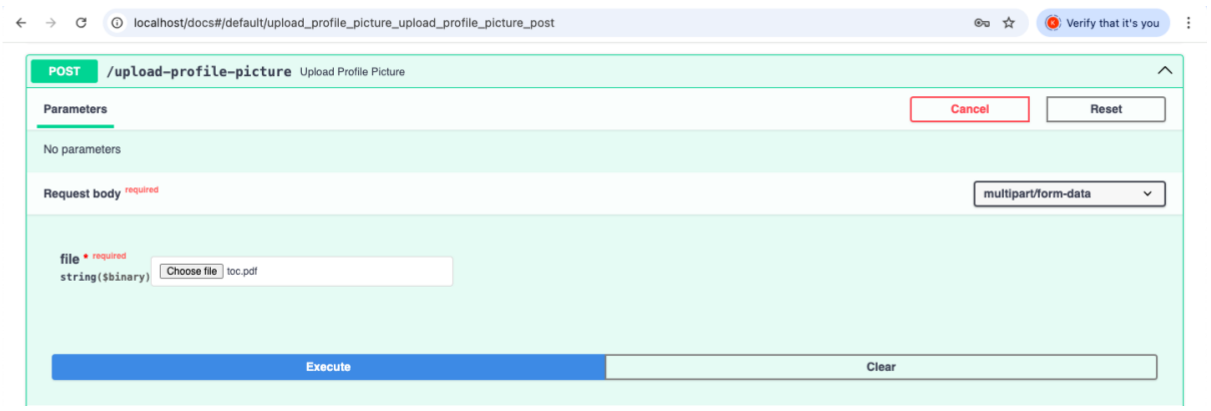
email * required string: string

password * required string: string

📸 ISSUE 6: Image Upload Endpoint Accepts All File Types Instead of Only Images

DESCRIPTION:

- The image upload endpoint was allowing any file type to be uploaded.
- This was not secure and violated expected behavior.
- The validation was updated to permit only JPEG, PNG, or WEBP image file types.


localhost/docs#/default/upload_profile_picture_upload_profile_picture_post

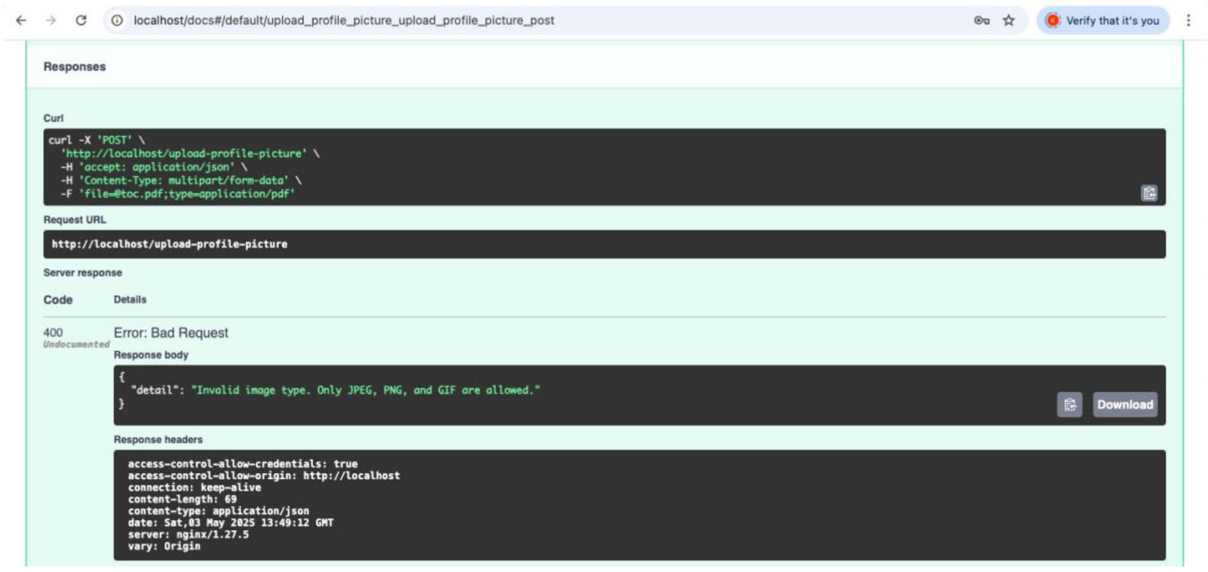
POST /upload-profile-picture Upload Profile Picture

Parameters — No parameters

Request body required — multipart/form-data

file * required string($binary): Choose file toc.pdf

Resolved Issue:


localhost/docs#/default/upload_profile_picture_upload_profile_picture_post
Verify that it's you
Responses
Curl

```
curl -X 'POST' \
  'http://localhost/upload-profile-picture' \
  -H 'accept: application/json' \
  -H 'Content-Type: multipart/form-data' \
  -F 'file=@toc.pdf;type=application/pdf'
```

Request URL

```
http://localhost/upload-profile-picture
```

Server response

Code	Details
400 *Undocumented*	Error: Bad Request

Response body

```
{
  "detail": "Invalid image type. Only JPEG, PNG, and GIF are allowed."
}
```

Response headers

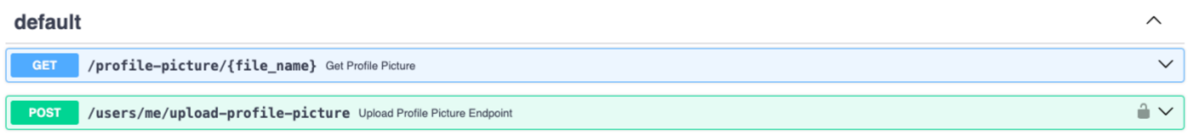
```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost
connection: keep-alive
content-length: 69
content-type: application/json
date: Sat,03 May 2025 13:49:12 GMT
server: nginx/1.27.5
vary: Origin
```

<u>FEATURE: Profile Picture Upload with Minio</u>

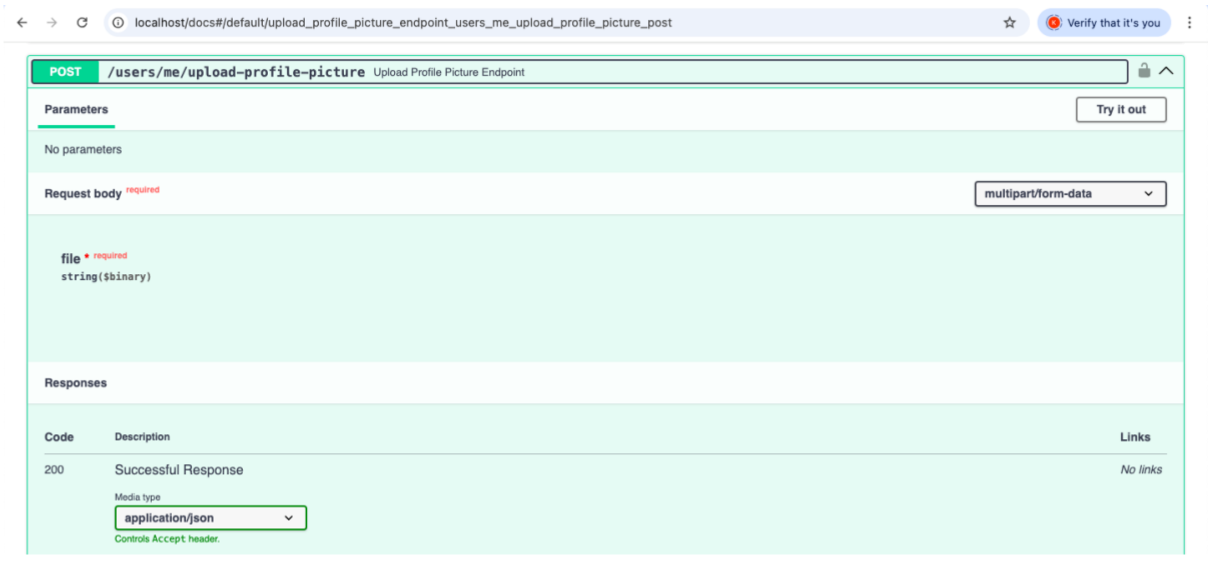
Description:

- A new functionality was added that allows profile images to be uploaded by users, which are stored in Minio (an S3-compatible object store).
- The feature includes validations for the supported image types, UUID-based renaming to ensure uniqueness, and accessible image URLs.
- This greatly enhanced user personalization and cloud storage integration.

Working of the Feature:



```
GET  /profile-picture/{file_name}  Get Profile Picture
POST /users/me/upload-profile-picture  Upload Profile Picture Endpoint
```


localhost/docs#/default/upload_profile_picture_endpoint_users_me_upload_profile_picture_post

POST /users/me/upload-profile-picture Upload Profile Picture Endpoint

Parameters

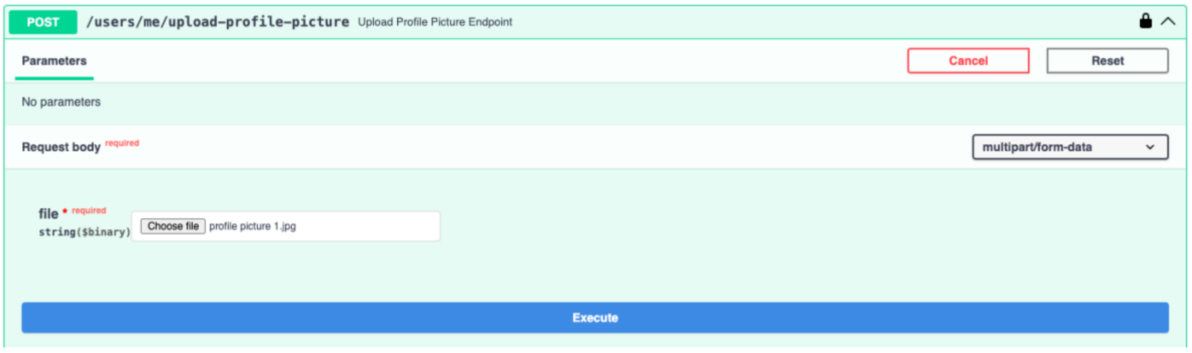
No parameters

Request body required — multipart/form-data

file * required
string($binary)

Responses

Code	Description	Links
200	Successful Response Media type: application/json Controls Accept header.	No links

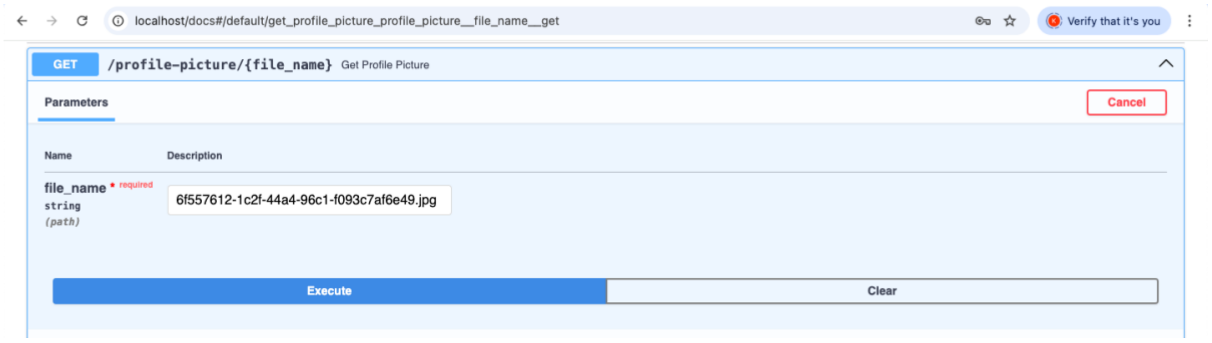

POST /users/me/upload-profile-picture Upload Profile Picture Endpoint

Parameters

No parameters

Request body required — multipart/form-data

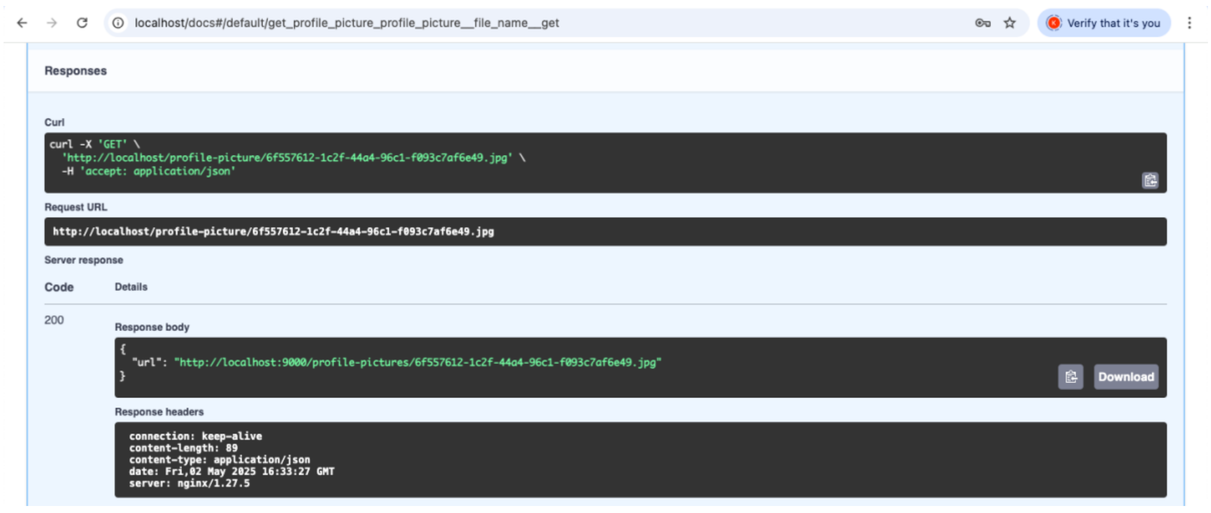
file * required
string($binary) — Choose file: profile picture 1.jpg


localhost/docs#/default/get_profile_picture_profile_picture__file_name__get

GET /profile-picture/{file_name} Get Profile Picture

Parameters

Name	Description
file_name * required string (path)	6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg


localhost/docs#/default/get_profile_picture_profile_picture__file_name__get
Verify that it's you

Responses

Curl

```
curl -X 'GET' \
  'http://localhost/profile-picture/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost/profile-picture/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg
```

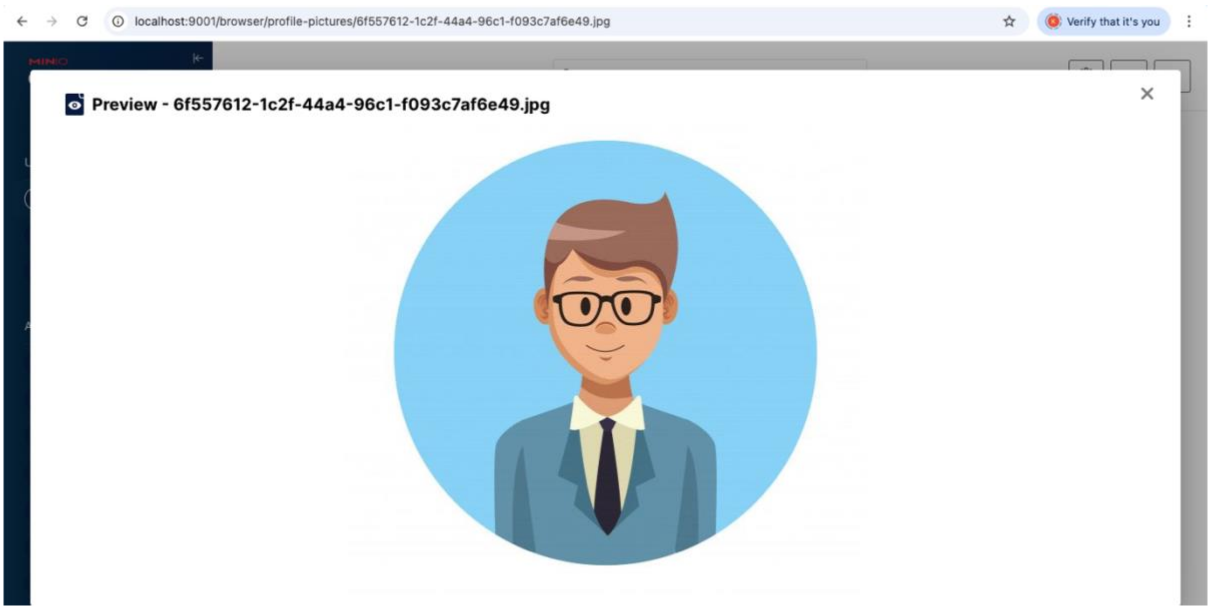
Server response

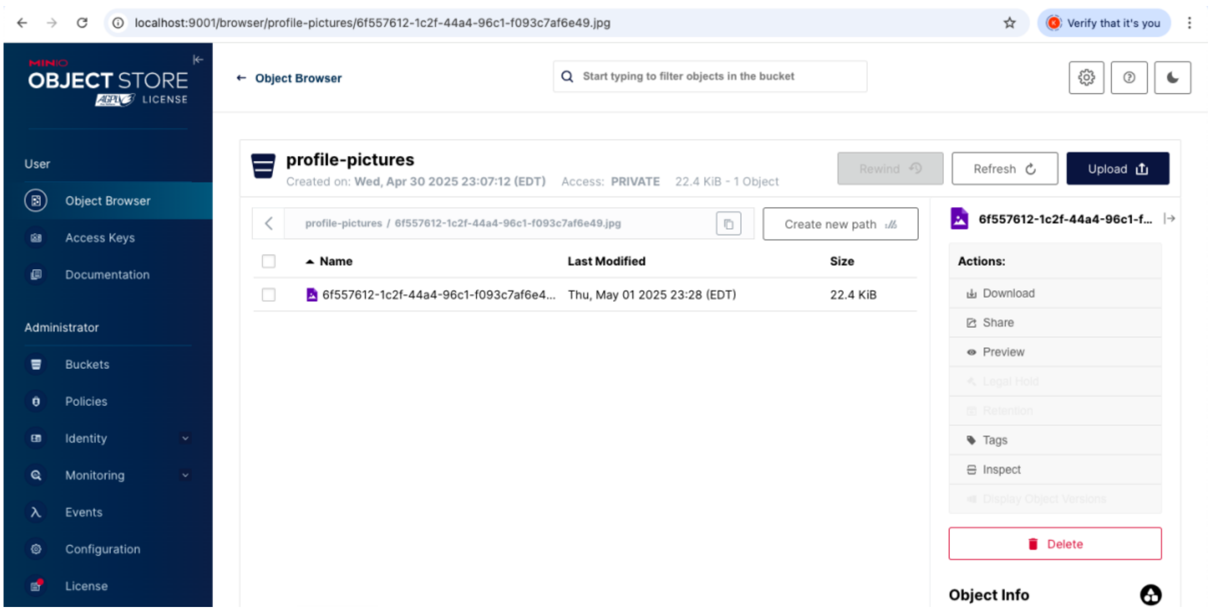
Code	Details
200	Response body

```
{
  "url": "http://localhost:9000/profile-pictures/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg"
}
```

Response headers

```
connection: keep-alive
content-length: 89
content-type: application/json
date: Fri,02 May 2025 16:33:27 GMT
server: nginx/1.27.5
```


localhost:9001/browser/profile-pictures/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg
Verify that it's you
Preview - 6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg



profile-pictures
Created on: Wed, Apr 30 2025 23:07:12 (EDT) Access: PRIVATE 22.4 KiB - 1 Object

profile-pictures / 6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg

Name	Last Modified	Size
6f557612-1c2f-44a4-96c1-f093c7af6e4...	Thu, May 01 2025 23:28 (EDT)	22.4 KiB

6f557612-1c2f-44a4-96c1-f...

☑ TEST CASES:

📤 TEST CASE 1: Basic Upload Functionality for Minio Image

Description:

- Ensures that the core image upload to Minio is working and returns a successful image URL.

🔗 TEST CASE 2: URL Generation for Minio Image

Description:

- Ensures that the get_image_url_from_minio() function returns a valid and well-structured public image URL from a filename.

🗀 TEST CASE 3: Long Filename Handling for Minio Image

Description:

- Tests how the function behaves with extremely long filenames, to make sure they are correctly converted to UUID-based names.

🗂 TEST CASE 4: Custom Bucket Configuration for Minio Image

Description:

- Ensures that image uploads correctly utilize a custom bucket name if provided in the configuration.

📝 TEST CASE 5: URL Format Variations for Minio Image

Description:

- Tests whether different formats of base URLs (e.g., with and without trailing slashes) are properly handled in image URL generation.

📋 TEST CASE 6: List Users with Pagination

Description:

- Tests the endpoint for retrieving users with limit and offset to make sure there is proper pagination of results.

✖ TEST CASE 7: Create User with Missing Email Field

Description:

- Ensures that user registration fails when the required email field is not given, ensuring proper validation.

🔒 TEST CASE 8: Create User with Short Password

Description:

- Tests that passwords shorter than the allowed length are correctly rejected when a user registers.

🚫 TEST CASE 9: Invalid Token Access

Description:

- Test that requests made with an invalid JWT token get blocked with correct unauthorized error.

⊖ TEST CASE 10: Unauthorized User Update Attempt

Description:

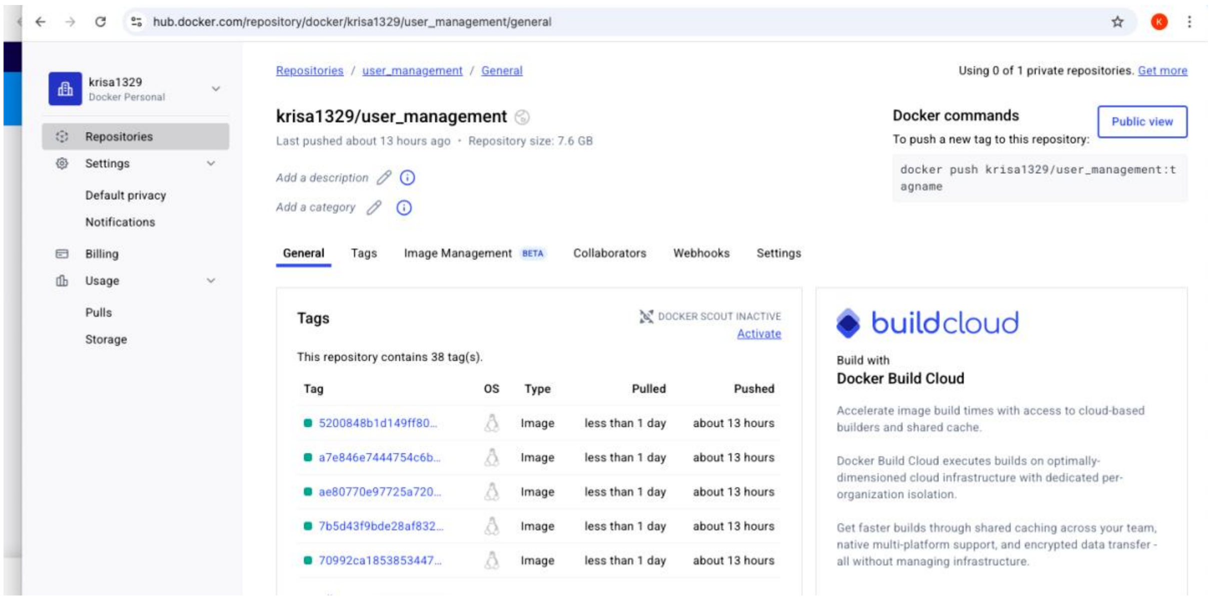
- Prevents a non-admin user from changing the details of another user and ensures proper access control.

DOCKERHUB DEPLOYMENT:

- The application was successfully containerized and deployed to DockerHub.

DockerHub Repository:

https://hub.docker.com/r/krisa1329/user_management


hub.docker.com/repository/docker/krisa1329/user_management/general

krisa1329
Docker Personal

Repositories
Settings
Default privacy
Notifications
Billing
Usage
Pulls
Storage

Repositories / user_management / General

Using 0 of 1 private repositories. Get more

krisa1329/user_management
Last pushed about 13 hours ago • Repository size: 7.6 GB

Add a description
Add a category

Docker commands
Public view
To push a new tag to this repository:

```
docker push krisa1329/user_management:tagname
```

General | Tags | Image Management BETA | Collaborators | Webhooks | Settings

Tags
DOCKER SCOUT INACTIVE Activate

This repository contains 38 tag(s).

Tag	OS	Type	Pulled	Pushed
5200848b1d149ff80...		Image	less than 1 day	about 13 hours
a7e846e7444754c6b...		Image	less than 1 day	about 13 hours
ae80770e97725a720...		Image	less than 1 day	about 13 hours
7b5d43f9bde28af832...		Image	less than 1 day	about 13 hours
70992ca1853853447...		Image	less than 1 day	about 13 hours

buildcloud

Build with
Docker Build Cloud

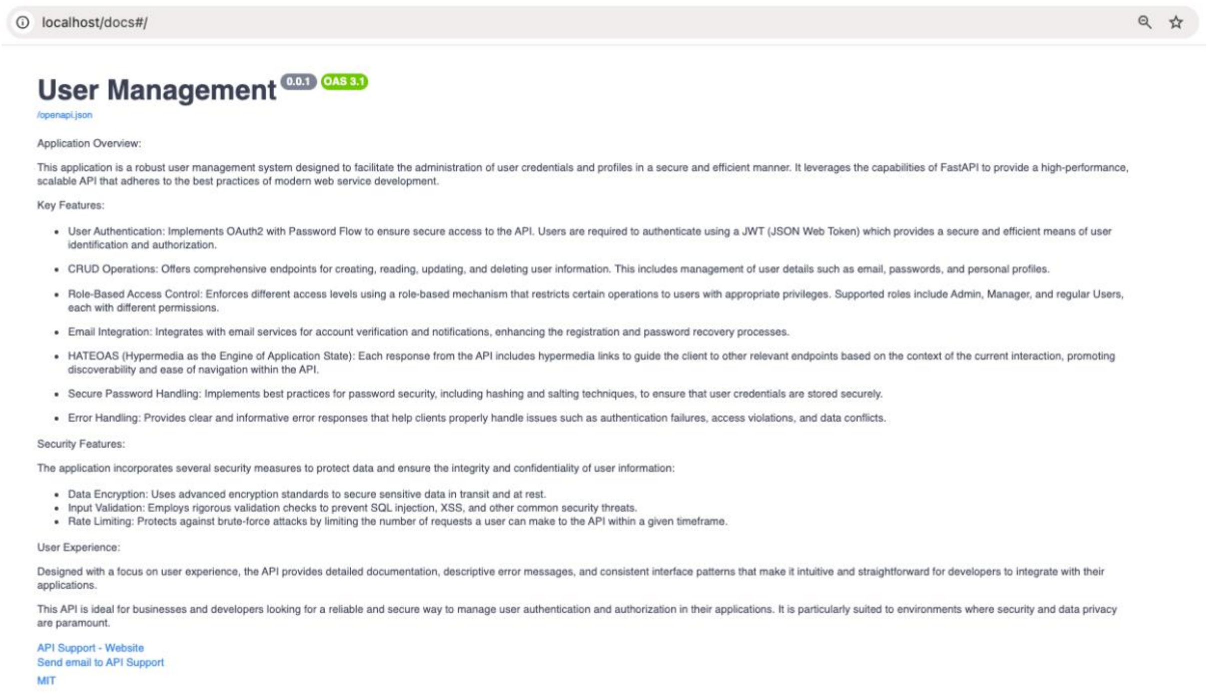
Accelerate image build times with access to cloud-based builders and shared cache.

Docker Build Cloud executes builds on optimally-dimensioned cloud infrastructure with dedicated per-organization isolation.

Get faster builds through shared caching across your team, native multi-platform support, and encrypted data transfer - all without managing infrastructure.

WORKING OF THE WHOLE PROJECT:

~ USER MANAGEMENT PAGE:



localhost/docs#/

User Management 0.0.1 OAS 3.1

/openapi.json

Application Overview:

This application is a robust user management system designed to facilitate the administration of user credentials and profiles in a secure and efficient manner. It leverages the capabilities of FastAPI to provide a high-performance, scalable API that adheres to the best practices of modern web service development.

Key Features:

- User Authentication: Implements OAuth2 with Password Flow to ensure secure access to the API. Users are required to authenticate using a JWT (JSON Web Token) which provides a secure and efficient means of user identification and authorization.
- CRUD Operations: Offers comprehensive endpoints for creating, reading, updating, and deleting user information. This includes management of user details such as email, passwords, and personal profiles.
- Role-Based Access Control: Enforces different access levels using a role-based mechanism that restricts certain operations to users with appropriate privileges. Supported roles include Admin, Manager, and regular Users, each with different permissions.
- Email Integration: Integrates with email services for account verification and notifications, enhancing the registration and password recovery processes.
- HATEOAS (Hypermedia as the Engine of Application State): Each response from the API includes hypermedia links to guide the client to other relevant endpoints based on the context of the current interaction, promoting discoverability and ease of navigation within the API.
- Secure Password Handling: Implements best practices for password security, including hashing and salting techniques, to ensure that user credentials are stored securely.
- Error Handling: Provides clear and informative error responses that help clients properly handle issues such as authentication failures, access violations, and data conflicts.

Security Features:

The application incorporates several security measures to protect data and ensure the integrity and confidentiality of user information:

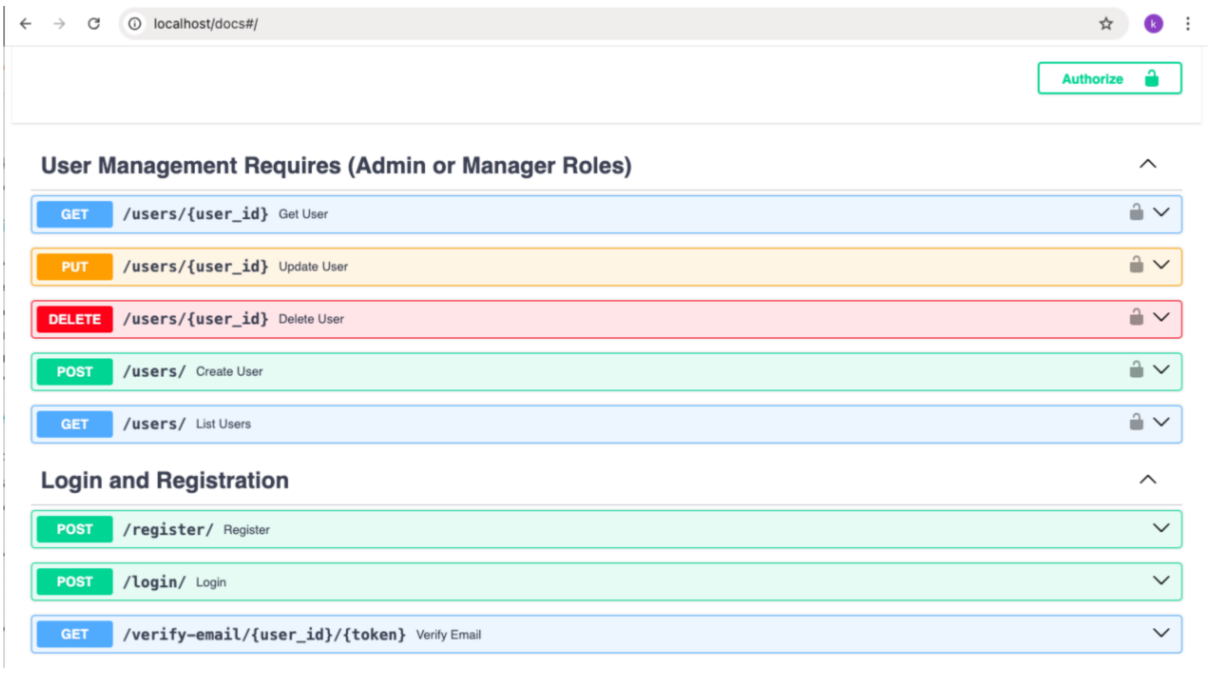
- Data Encryption: Uses advanced encryption standards to secure sensitive data in transit and at rest.
- Input Validation: Employs rigorous validation checks to prevent SQL injection, XSS, and other common security threats.
- Rate Limiting: Protects against brute-force attacks by limiting the number of requests a user can make to the API within a given timeframe.

User Experience:

Designed with a focus on user experience, the API provides detailed documentation, descriptive error messages, and consistent interface patterns that make it intuitive and straightforward for developers to integrate with their applications.

This API is ideal for businesses and developers looking for a reliable and secure way to manage user authentication and authorization in their applications. It is particularly suited to environments where security and data privacy are paramount.

API Support - Website
Send email to API Support
MIT



localhost/docs#/

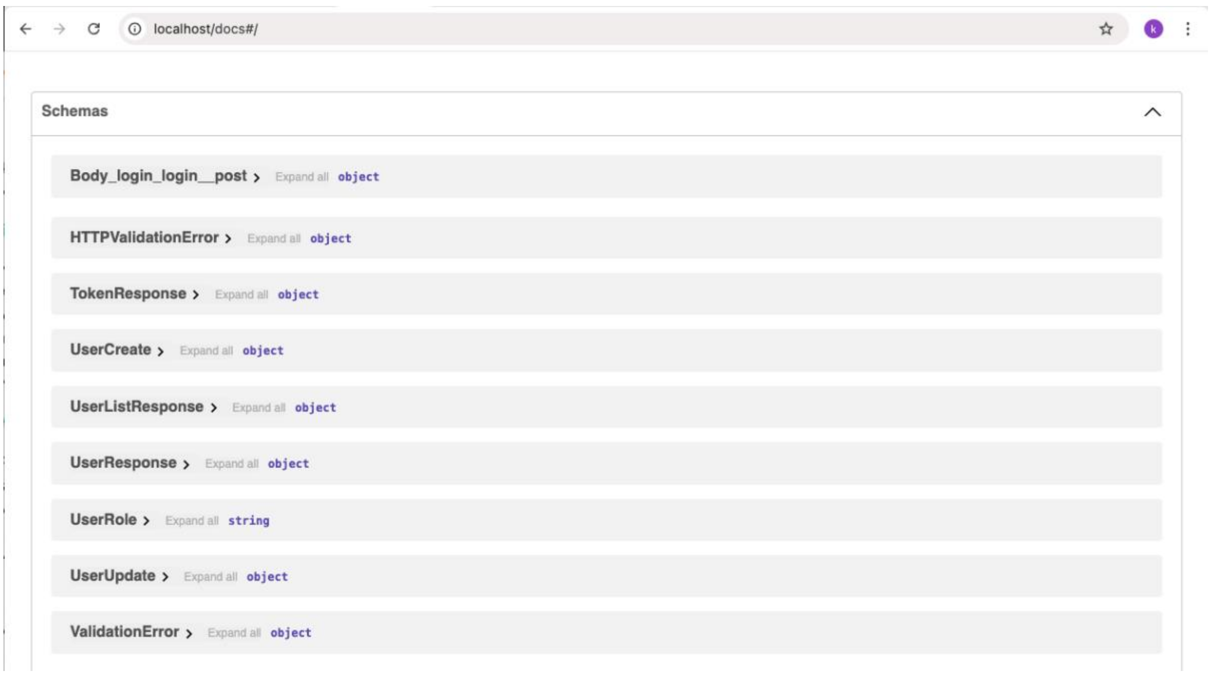
Authorize

User Management Requires (Admin or Manager Roles)

- GET `/users/{user_id}` Get User
- PUT `/users/{user_id}` Update User
- DELETE `/users/{user_id}` Delete User
- POST `/users/` Create User
- GET `/users/` List Users

Login and Registration

- POST `/register/` Register
- POST `/login/` Login
- GET `/verify-email/{user_id}/{token}` Verify Email



localhost/docs#/

Schemas

Body_login_login__post › Expand all object

HTTPValidationError › Expand all object

TokenResponse › Expand all object

UserCreate › Expand all object

UserListResponse › Expand all object

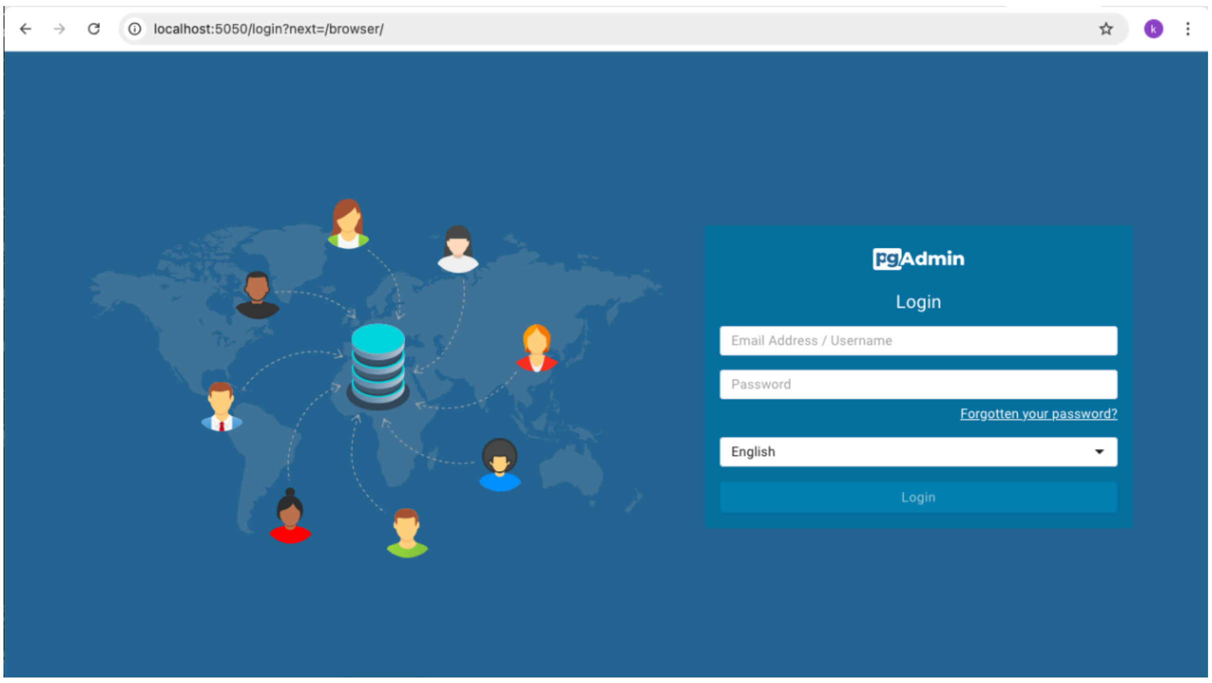
UserResponse › Expand all object

UserRole › Expand all string

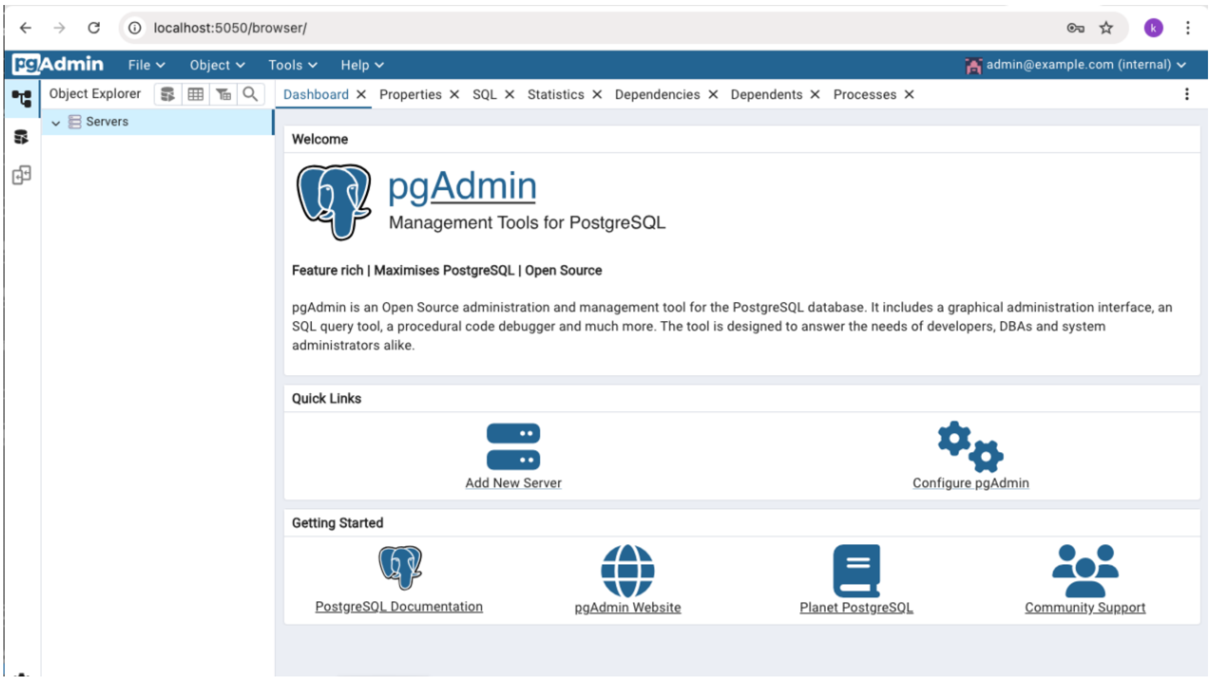
UserUpdate › Expand all object

ValidationError › Expand all object

~ PGADMIN PAGE:



localhost:5050/login?next=/browser/


localhost:5050/browser/

pgAdmin — File, Object, Tools, Help — admin@example.com (internal)

Object Explorer — Servers

Dashboard | Properties | SQL | Statistics | Dependencies | Dependents | Processes

Welcome

pgAdmin
Management Tools for PostgreSQL

Feature rich | Maximises PostgreSQL | Open Source

pgAdmin is an Open Source administration and management tool for the PostgreSQL database. It includes a graphical administration interface, an SQL query tool, a procedural code debugger and much more. The tool is designed to answer the needs of developers, DBAs and system administrators alike.

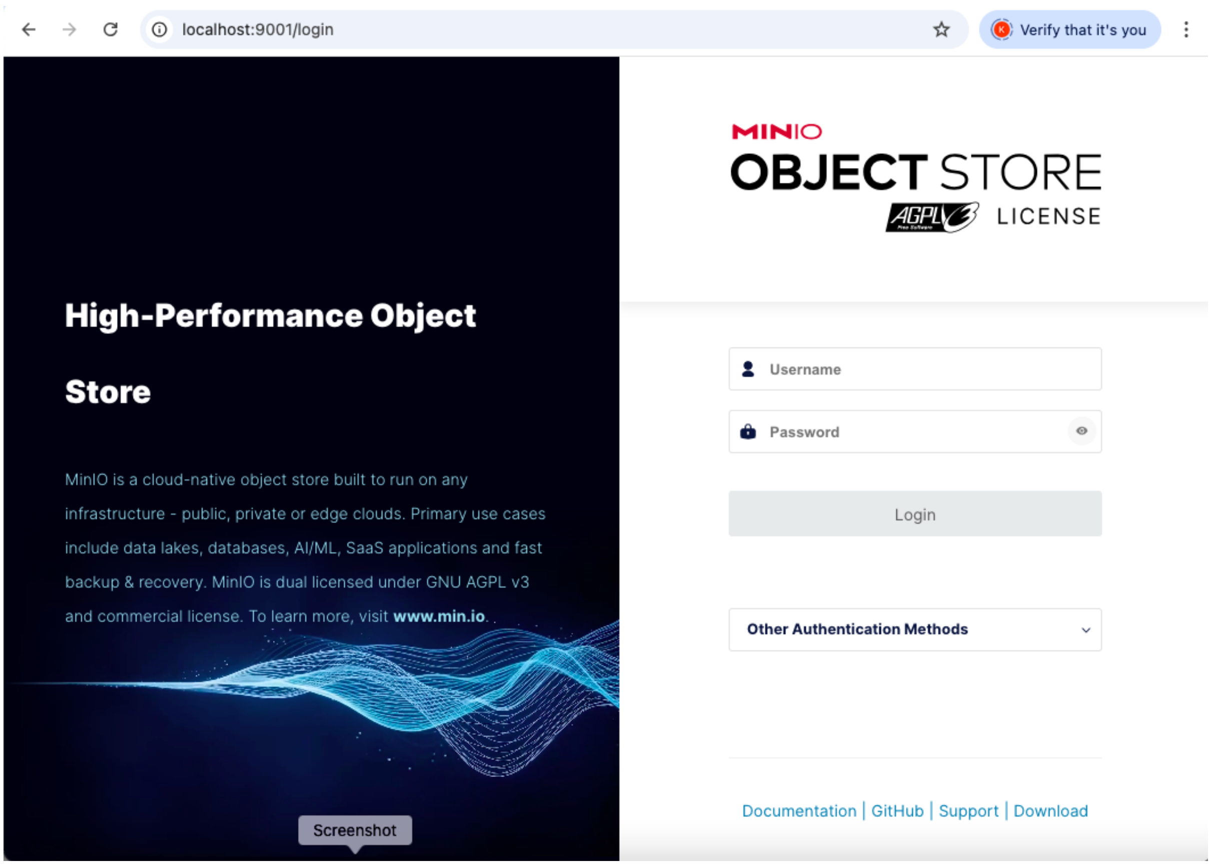
Quick Links

Add New Server | Configure pgAdmin

Getting Started

PostgreSQL Documentation | pgAdmin Website | Planet PostgreSQL | Community Support

~ MINIO PAGE:


localhost:9001/login — Verify that it's you

High-Performance Object Store

MinIO is a cloud-native object store built to run on any infrastructure - public, private or edge clouds. Primary use cases include data lakes, databases, AI/ML, SaaS applications and fast backup & recovery. MinIO is dual licensed under GNU AGPL v3 and commercial license. To learn more, visit **www.min.io**.

MINIO OBJECT STORE AGPLv3 LICENSE

Username

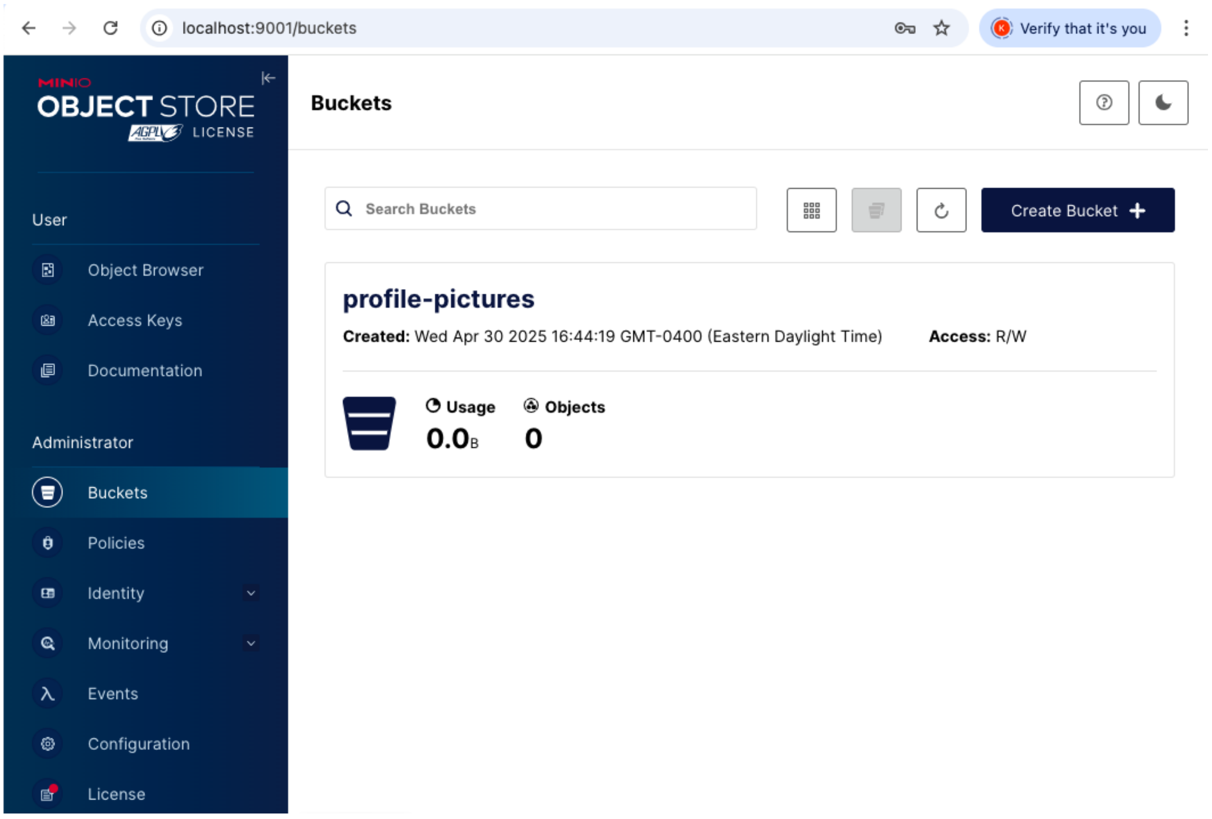
Password

Login

Other Authentication Methods

Documentation | GitHub | Support | Download

Screenshot

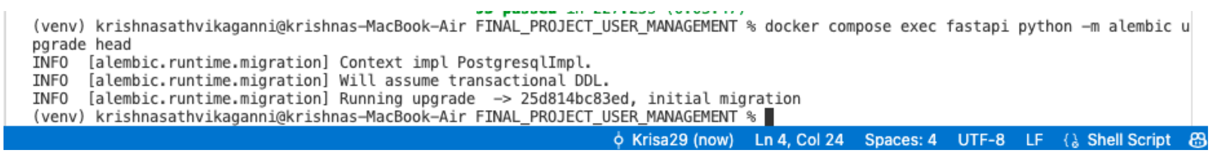


profile-pictures

Created: Wed Apr 30 2025 16:44:19 GMT-0400 (Eastern Daylight Time) **Access:** R/W

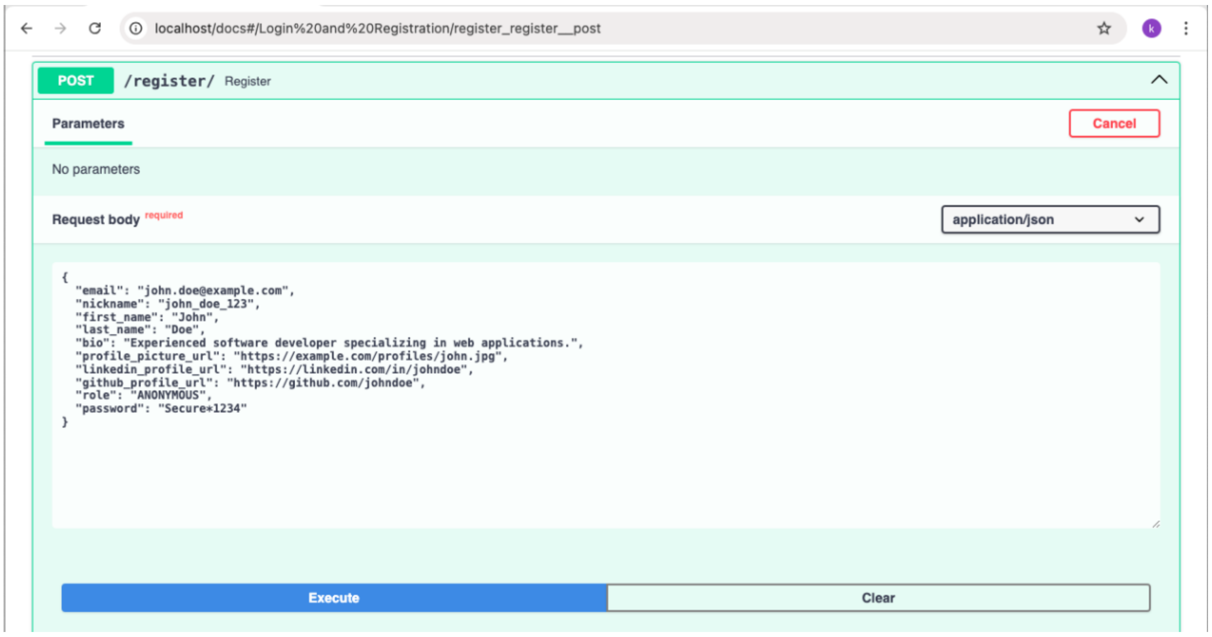
Usage 0.0B Objects 0

~ CREATION OF TABLES IN PGADMIN WITH THE COMMAND:



```
(venv) krishnasathvikaganni@krishnas-MacBook-Air FINAL_PROJECT_USER_MANAGEMENT % docker compose exec fastapi python -m alembic u
pgrade head
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> 25d814bc83ed, initial migration
(venv) krishnasathvikaganni@krishnas-MacBook-Air FINAL_PROJECT_USER_MANAGEMENT %
```

~ REGISTERING OF USER:

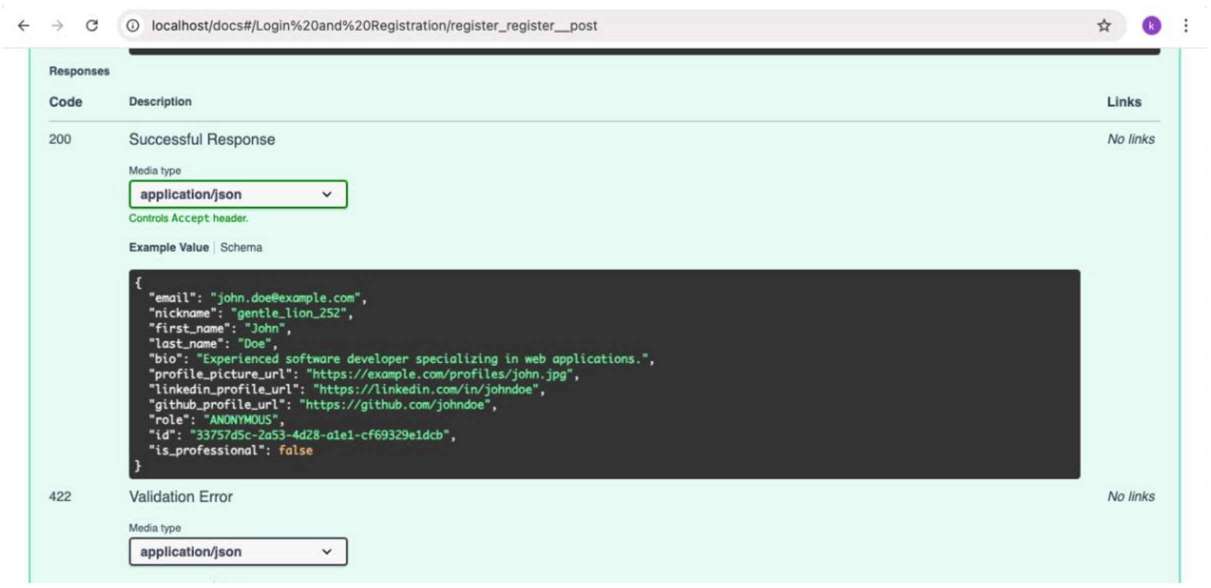

localhost/docs#/Login%20and%20Registration/register_register__post

No parameters

Request body required

application/json

```
{
  "email": "john.doe@example.com",
  "nickname": "john_doe_123",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "password": "Secure*1234"
}
```


localhost/docs#/Login%20and%20Registration/register_register__post

Responses

Code	Description	Links
200	Successful Response	No links

Media type: application/json

Controls Accept header.

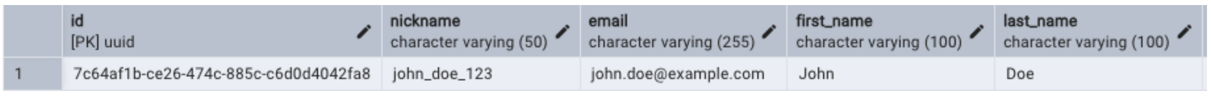
Example Value | Schema

```
{
  "email": "john.doe@example.com",
  "nickname": "gentle_lion_252",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ANONYMOUS",
  "id": "33757d5c-2a53-4d28-a1e1-cf69329e1dcb",
  "is_professional": false
}
```

Code	Description	Links
422	Validation Error	No links

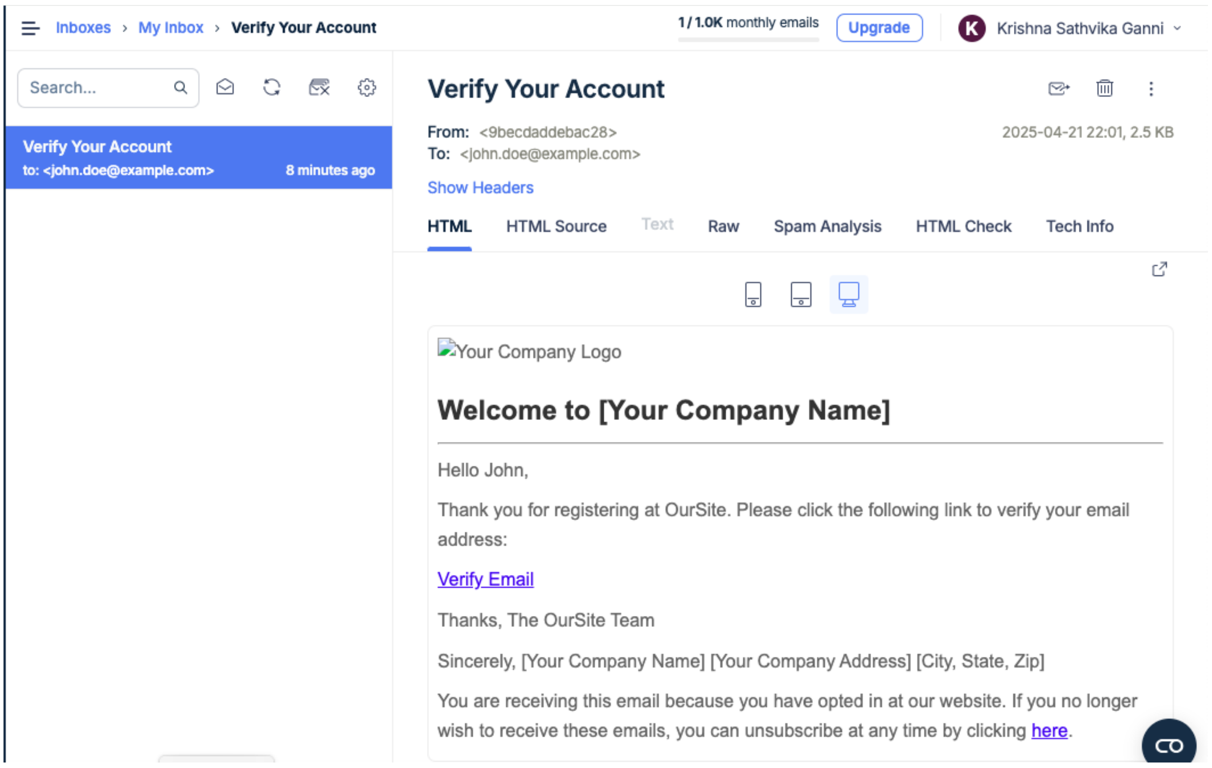
Media type: application/json

~ REGISTERED USER STORED IN DATABASE:



	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)
1	7c64af1b-ce26-474c-885c-c6d0d4042fa8	john_doe_123	john.doe@example.com	John	Doe

~ VERIFICATION OF REGISTERED USER USING EMAIL:



Verify Your Account

From: <9becdaddebac28>
To: <john.doe@example.com>

2025-04-21 22:01, 2.5 KB

Your Company Logo

Welcome to [Your Company Name]

Hello John,

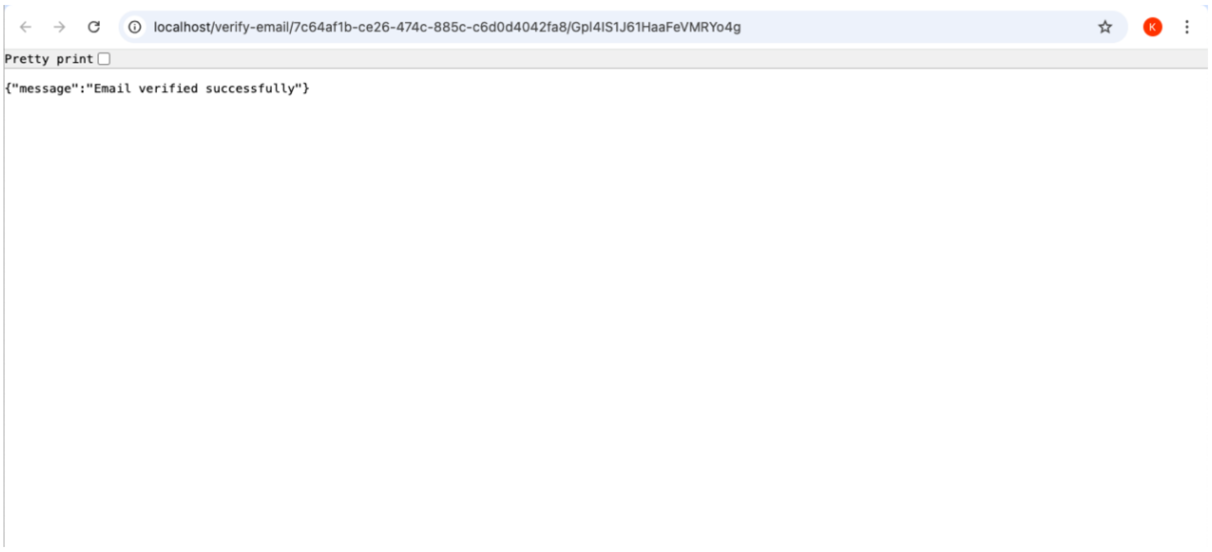
Thank you for registering at OurSite. Please click the following link to verify your email address:

Verify Email

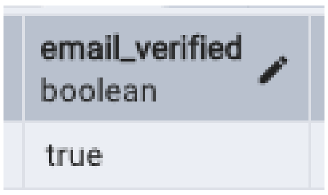
Thanks, The OurSite Team

Sincerely, [Your Company Name] [Your Company Address] [City, State, Zip]

You are receiving this email because you have opted in at our website. If you no longer wish to receive these emails, you can unsubscribe at any time by clicking here.

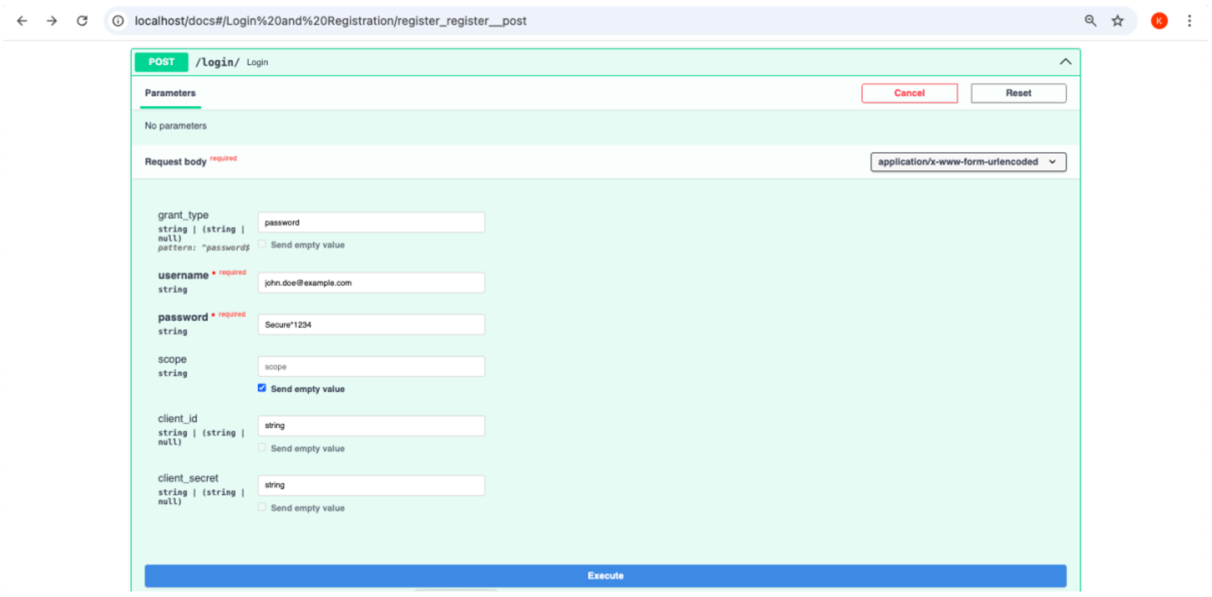

localhost/verify-email/7c64af1b-ce26-474c-885c-c6d0d4042fa8/Gpl4IS1J61HaaFeVMRYo4g
Pretty print

```
{"message":"Email verified successfully"}
```



email_verified boolean
true

~ LOGGING IN OF THE REGISTERED USER:


localhost/docs#/Login%20and%20Registration/register_register__post

POST /login/ Login

Parameters

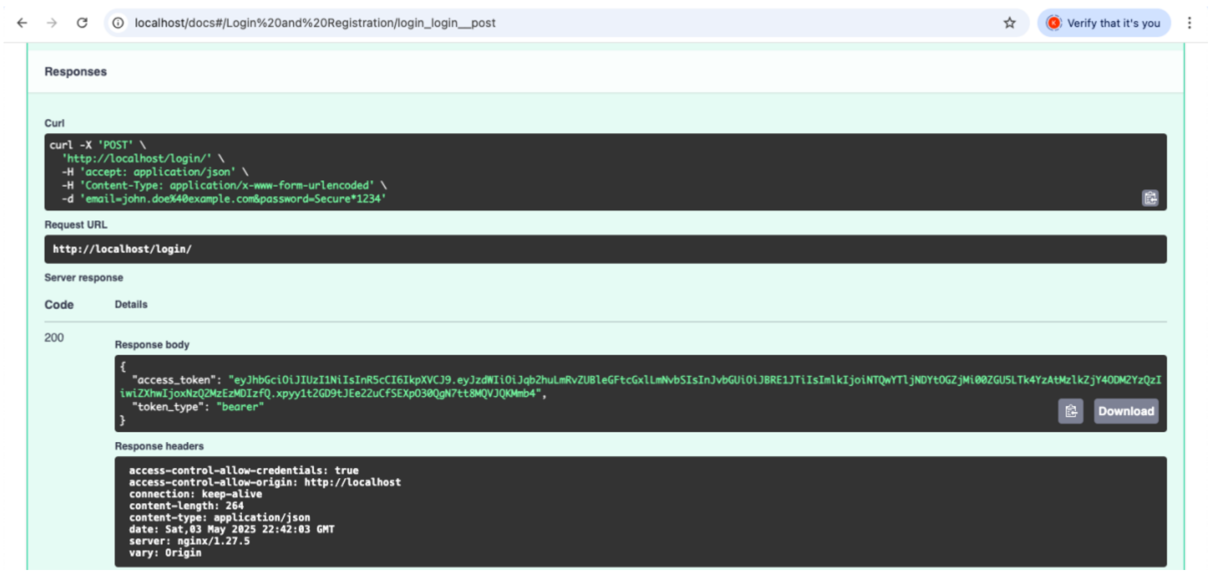
Cancel Reset

No parameters

Request body required application/x-www-form-urlencoded

grant_type string | (string | null) *pattern: ^password$*	password Send empty value
username * required string	john.doe@example.com
password * required string	Secure*1234
scope string	scope [✓ Send empty value]
client_id string | (string | null)	string Send empty value
client_secret string | (string | null)	string Send empty value

Execute


localhost/docs#/Login%20and%20Registration/login_login__post

Verify that it's you

Responses

Curl

```
curl -X 'POST' \
  'http://localhost/login/' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'email=john.doe%40example.com&password=Secure*1234'
```

Request URL

```
http://localhost/login/
```

Server response

Code Details

200

Response body

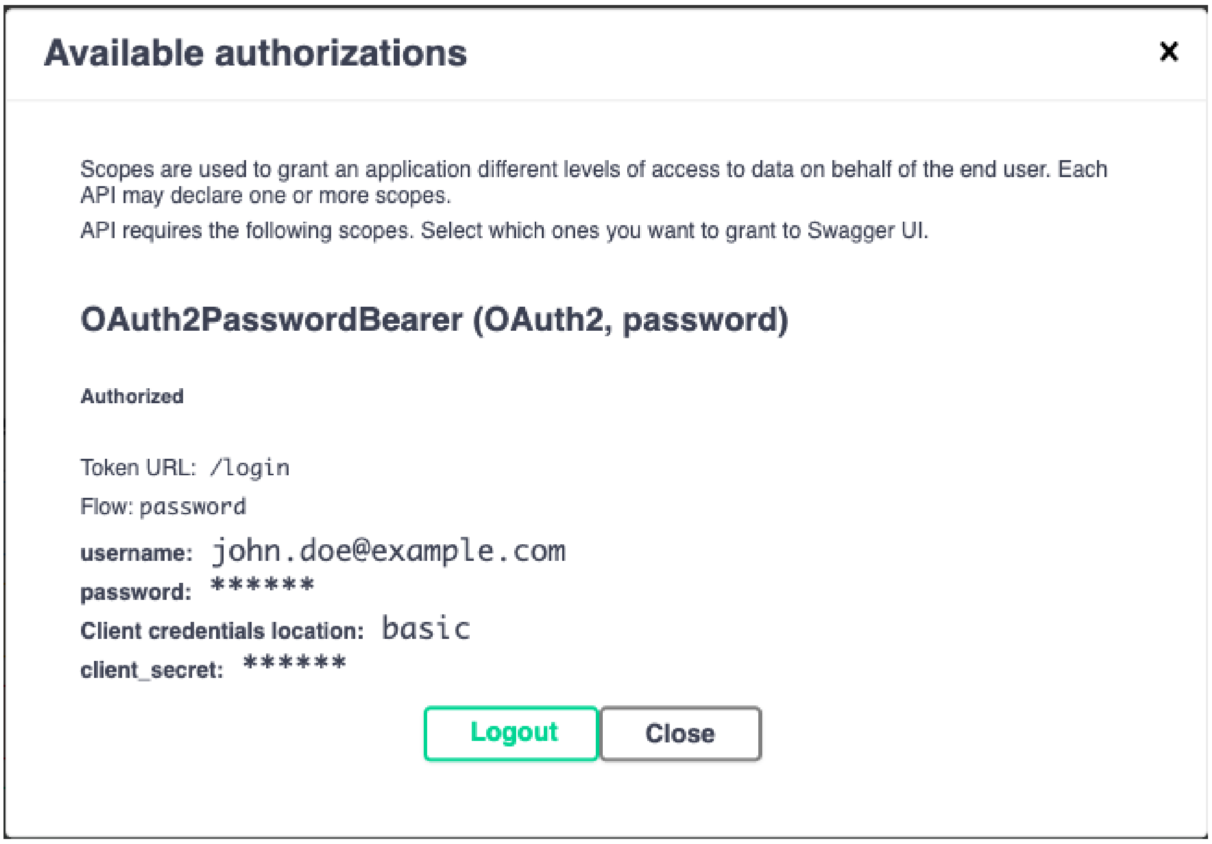
```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huLmRvZUBleGFtcGxlLmNvbSIsInJvbGUiOiJBRE1JTiIsImlkIjoiNTQwYTljNDYtOGZjMi00ZGU5LTk4YzAtMzlkZjY4ODM2YzQzIiwiZXhwIjoxNzQ2MzEzMDIzfQ.xpyy1t2GD9tJEe22uCfSEXpO30QgN7tt8MQVJQKMmb4",
  "token_type": "bearer"
}
```

Download

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost
connection: keep-alive
content-length: 264
content-type: application/json
date: Sat,03 May 2025 22:42:03 GMT
server: nginx/1.27.5
vary: Origin
```

~ AUTHORIZATION OF THE USER:



Available authorizations

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.

API requires the following scopes. Select which ones you want to grant to Swagger UI.

OAuth2PasswordBearer (OAuth2, password)

Authorized

Token URL: `/login`

Flow: `password`

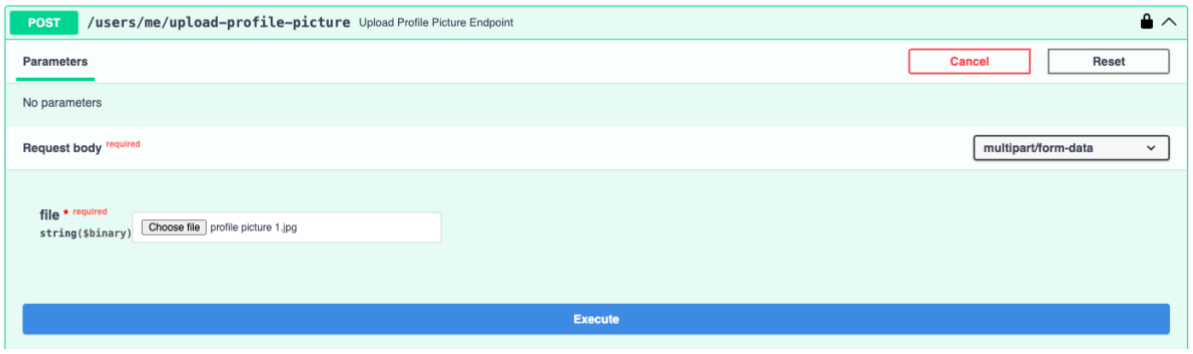
username: `john.doe@example.com`

password: `******`

Client credentials location: `basic`

client_secret: `******`

~ PROFILE PICTURE UPLOAD OF THE REGISTERED USER:



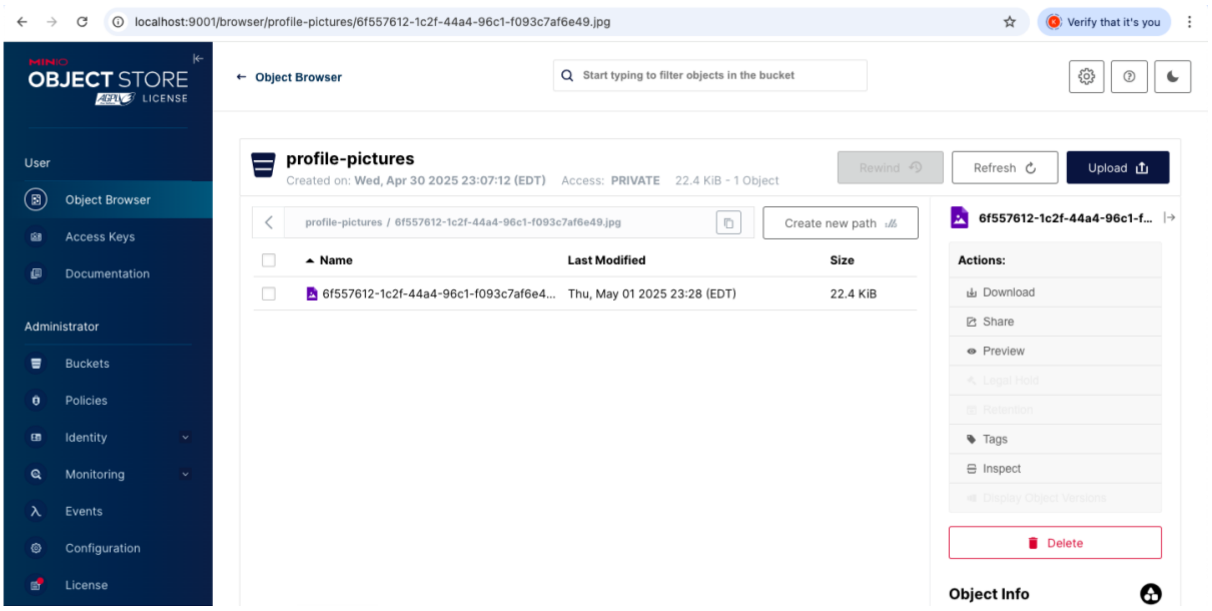
POST `/users/me/upload-profile-picture` Upload Profile Picture Endpoint

Parameters

No parameters

Request body required: multipart/form-data

file * required string($binary): profile picture 1.jpg



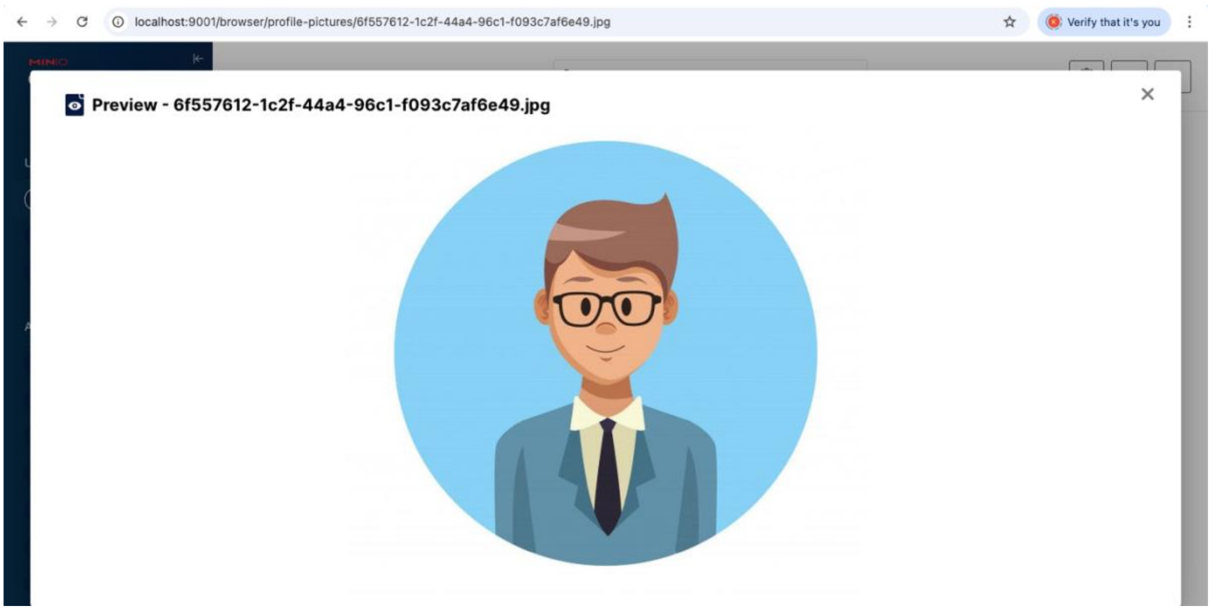
profile-pictures

Created on: Wed, Apr 30 2025 23:07:12 (EDT) Access: PRIVATE 22.4 KiB - 1 Object

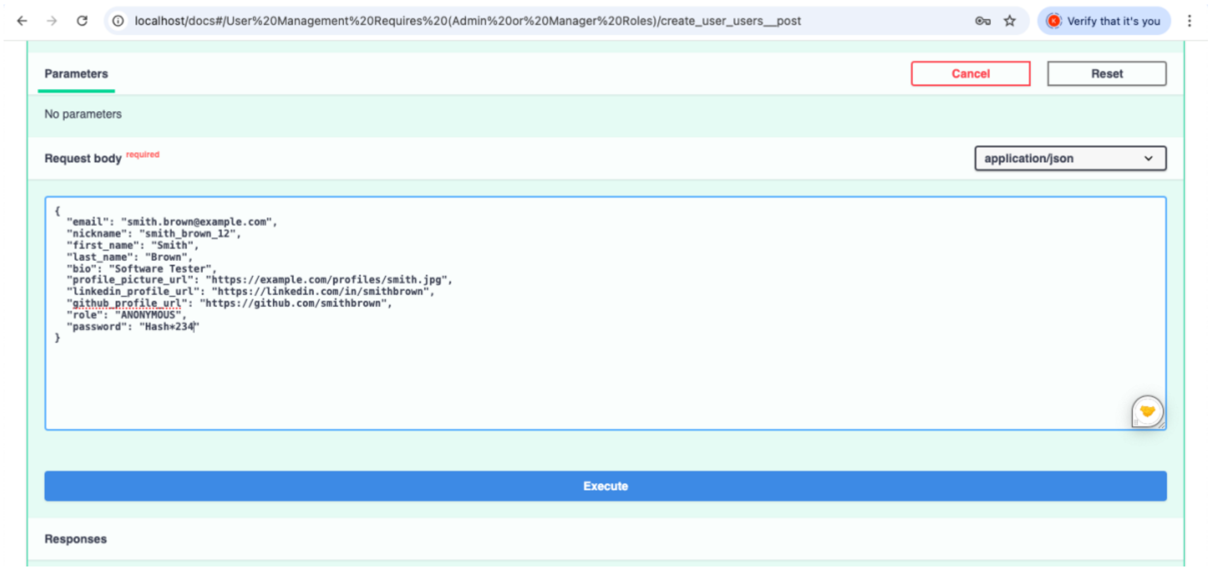
profile-pictures / 6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg

Name	Last Modified	Size
6f557612-1c2f-44a4-96c1-f093c7af6e4...	Thu, May 01 2025 23:28 (EDT)	22.4 KiB

6f557612-1c2f-44a4-96c1-f...


localhost:9001/browser/profile-pictures/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg
Verify that it's you
Preview - 6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg

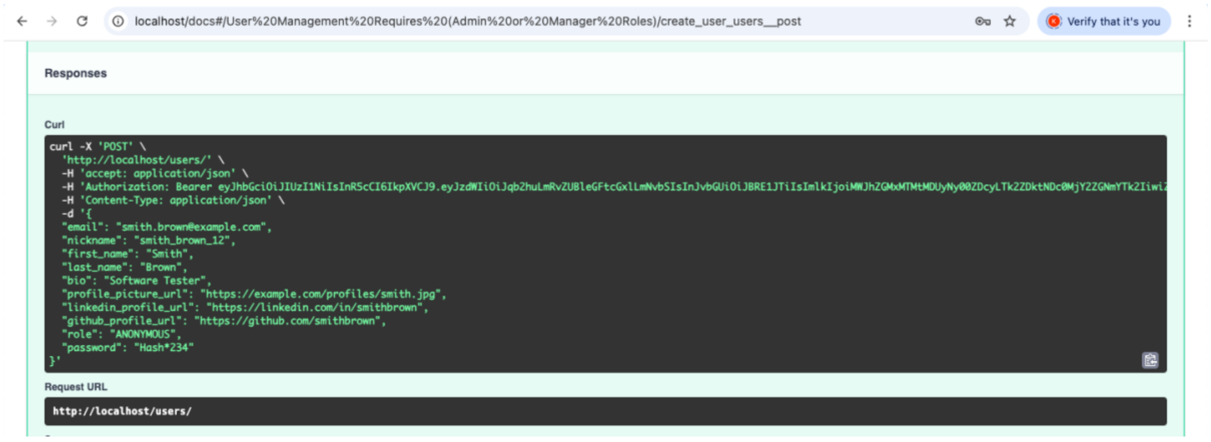
~ REGISTERING OF ANOTHER USER:


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/create_user_users__post

No parameters

Request body required

```
{
  "email": "smith.brown@example.com",
  "nickname": "smith_brown_12",
  "first_name": "Smith",
  "last_name": "Brown",
  "bio": "Software Tester",
  "profile_picture_url": "https://example.com/profiles/smith.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/smithbrown",
  "github_profile_url": "https://github.com/smithbrown",
  "role": "ANONYMOUS",
  "password": "Hash*234"
}
```

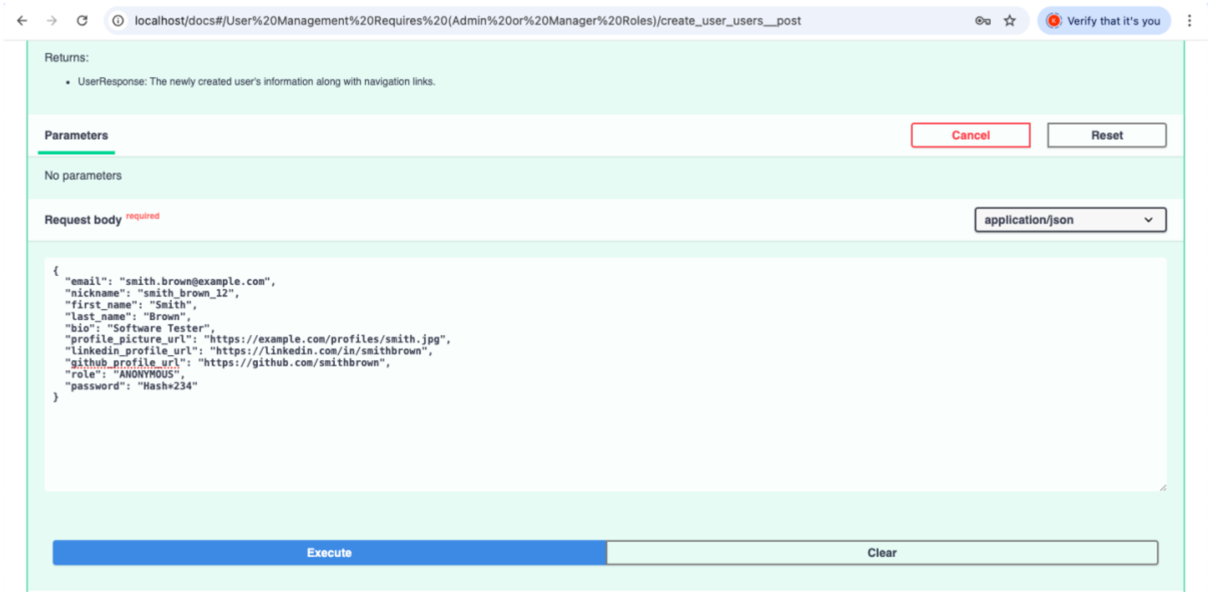

localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/create_user_users__post

Curl

```
curl -X 'POST' \
  'http://localhost/users/' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huLmRvZUBleGFtcGxlLmNvbSIsInJvbGUiOiJBRE1JTiIsImlkIjoiMWJhZGMxMTMtMDUyNy00ZDcyLTk2ZDktNDc0MjY2ZGNmYTk2Iiwi
  -H 'Content-Type: application/json' \
  -d '{
  "email": "smith.brown@example.com",
  "nickname": "smith_brown_12",
  "first_name": "Smith",
  "last_name": "Brown",
  "bio": "Software Tester",
  "profile_picture_url": "https://example.com/profiles/smith.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/smithbrown",
  "github_profile_url": "https://github.com/smithbrown",
  "role": "ANONYMOUS",
  "password": "Hash*234"
}'
```

Request URL

```
http://localhost/users/
```


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/create_user_users__post

Returns:

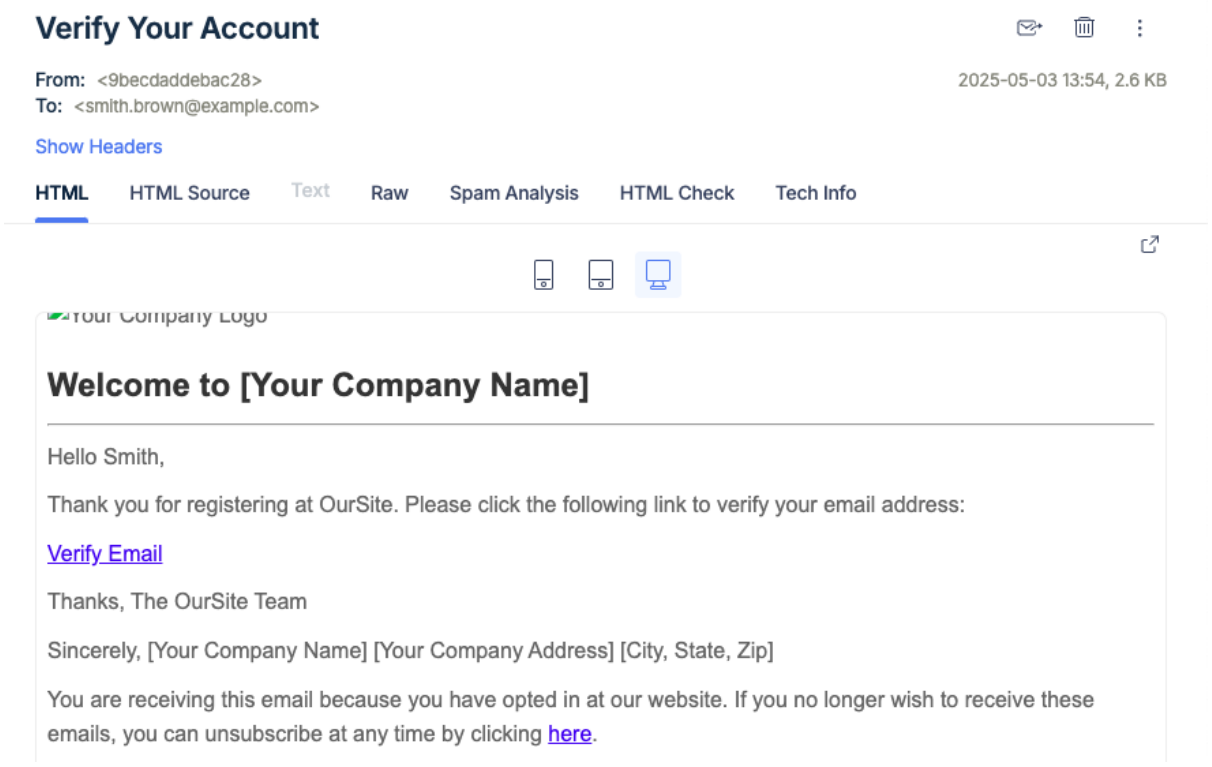
- UserResponse: The newly created user's information along with navigation links.

No parameters

Request body required

```
{
  "email": "smith.brown@example.com",
  "nickname": "smith_brown_12",
  "first_name": "Smith",
  "last_name": "Brown",
  "bio": "Software Tester",
  "profile_picture_url": "https://example.com/profiles/smith.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/smithbrown",
  "github_profile_url": "https://github.com/smithbrown",
  "role": "ANONYMOUS",
  "password": "Hash*234"
}
```

~ VERIFICATION OF ANOTHER USER:



Verify Your Account

From: <9becdaddebac28>
To: <smith.brown@example.com>

2025-05-03 13:54, 2.6 KB

Show Headers

HTML | HTML Source | Text | Raw | Spam Analysis | HTML Check | Tech Info

Your Company Logo

Welcome to [Your Company Name]

Hello Smith,

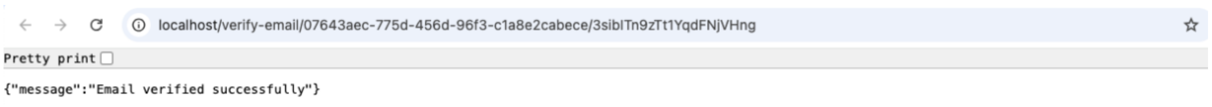
Thank you for registering at OurSite. Please click the following link to verify your email address:

Verify Email

Thanks, The OurSite Team

Sincerely, [Your Company Name] [Your Company Address] [City, State, Zip]

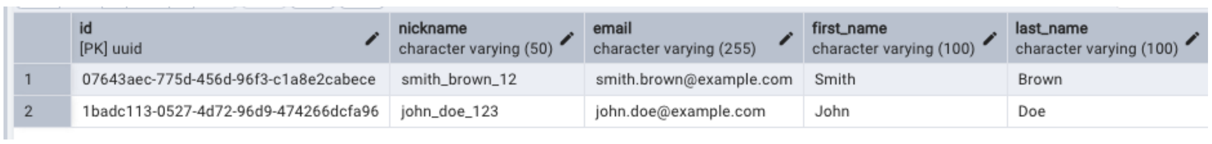
You are receiving this email because you have opted in at our website. If you no longer wish to receive these emails, you can unsubscribe at any time by clicking here.



localhost/verify-email/07643aec-775d-456d-96f3-c1a8e2cabece/3sibITn9zTt1YqdFNjVHng

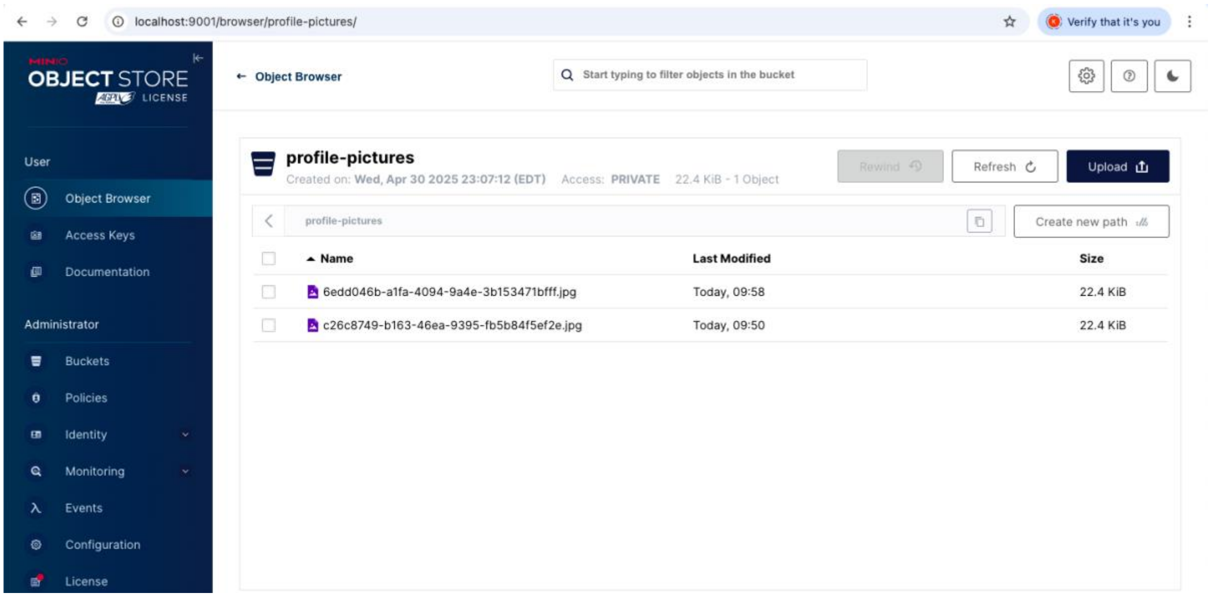
```
Pretty print
{"message":"Email verified successfully"}
```

~ REGISTERED USER STORED IN DATABASE:



	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)
1	07643aec-775d-456d-96f3-c1a8e2cabece	smith_brown_12	smith.brown@example.com	Smith	Brown
2	1badc113-0527-4d72-96d9-474266dcfa96	john_doe_123	john.doe@example.com	John	Doe

~ PROFILE PICTURE UPLOAD FOR ANOTHER USER:

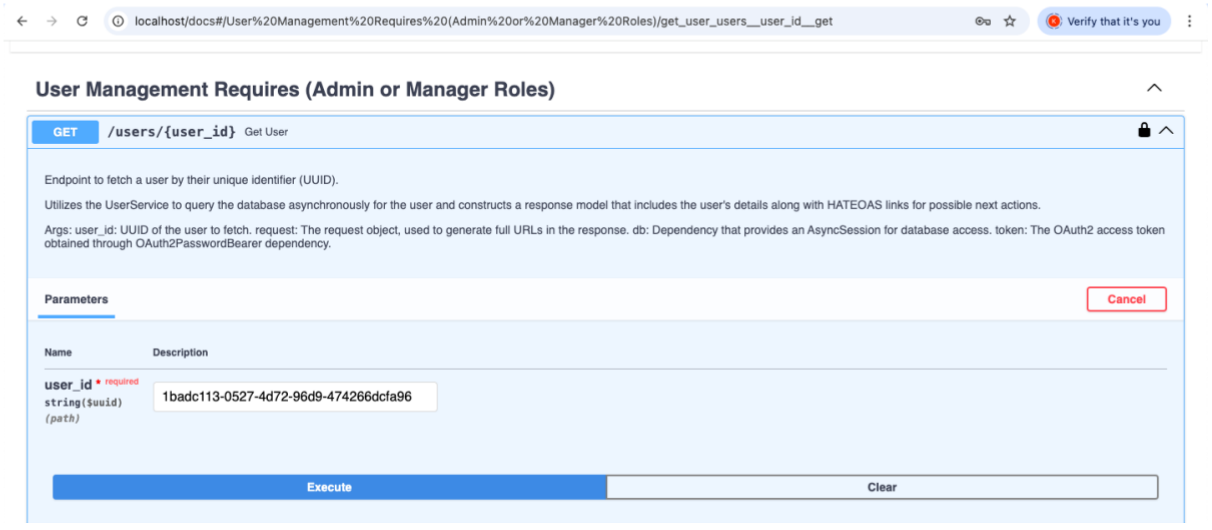


profile-pictures
Created on: Wed, Apr 30 2025 23:07:12 (EDT) Access: PRIVATE 22.4 KiB - 1 Object

profile-pictures

Name	Last Modified	Size
6edd046b-a1fa-4094-9a4e-3b153471bfff.jpg	Today, 09:58	22.4 KiB
c26c8749-b163-46ea-9395-fb5b84f5ef2e.jpg	Today, 09:50	22.4 KiB

~ GETTING THE USER:


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/get_user_users__user_id__get
Verify that it's you
Cancel
Execute
Clear

User Management Requires (Admin or Manager Roles)

GET /users/{user_id} Get User

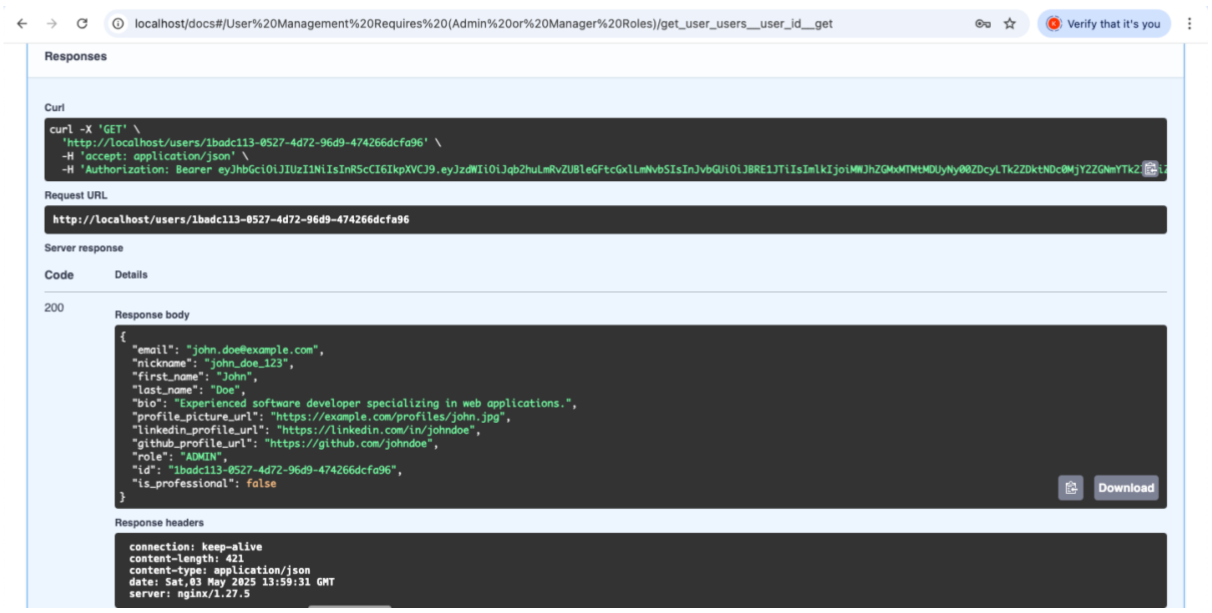
Endpoint to fetch a user by their unique identifier (UUID).

Utilizes the UserService to query the database asynchronously for the user and constructs a response model that includes the user's details along with HATEOAS links for possible next actions.

Args: user_id: UUID of the user to fetch. request: The request object, used to generate full URLs in the response. db: Dependency that provides an AsyncSession for database access. token: The OAuth2 access token obtained through OAuth2PasswordBearer dependency.

Parameters

Name	Description
user_id * required string($uuid) (path)	1badc113-0527-4d72-96d9-474266dcfa96


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/get_user_users__user_id__get
Verify that it's you
Responses

Curl

```
curl -X 'GET' \
  'http://localhost/users/1badc113-0527-4d72-96d9-474266dcfa96' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huLmRvZUBleGFtcGxlLmNvbSIsInJvbGUiOiJBRE1JTiIsImlkIjoiMWJhZGMxMTMtMDUyNy00ZDcyLTk2ZDktNDc0MjY2ZGNmYTk2
```

Request URL

```
http://localhost/users/1badc113-0527-4d72-96d9-474266dcfa96
```

Server response

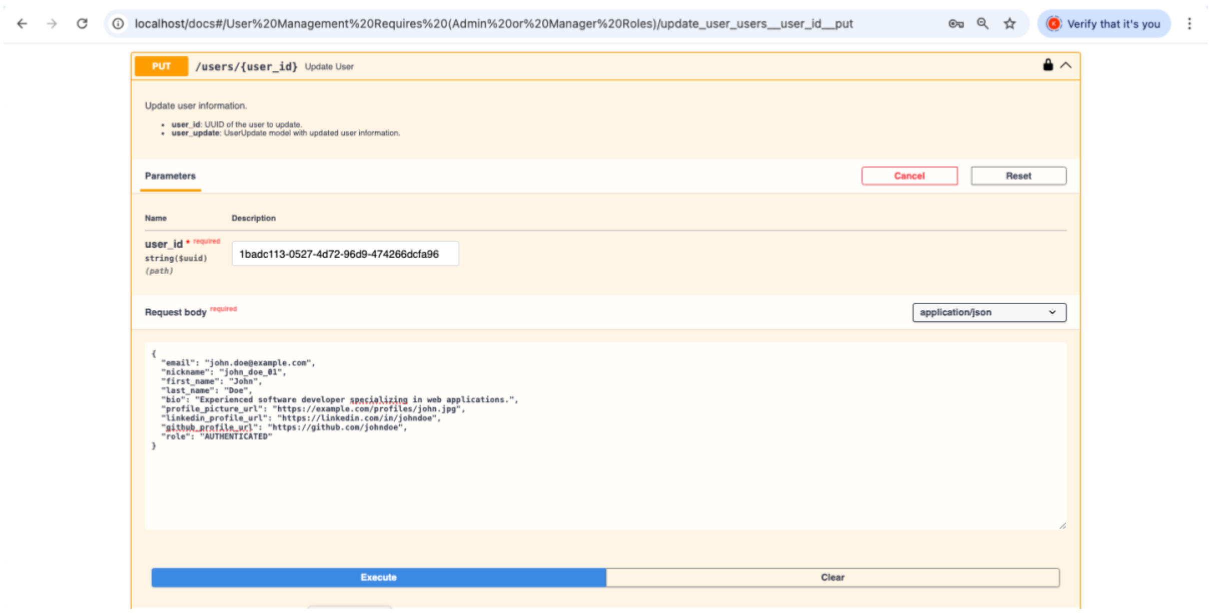
Code	Details
200	Response body

```
{
  "email": "john.doe@example.com",
  "nickname": "john_doe_123",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "ADMIN",
  "id": "1badc113-0527-4d72-96d9-474266dcfa96",
  "is_professional": false
}
```

Response headers

```
connection: keep-alive
content-length: 421
content-type: application/json
date: Sat,03 May 2025 13:59:31 GMT
server: nginx/1.27.5
```

~ UPDATING THE USER:


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/update_user_users__user_id__put
Verify that it's you
PUT

/users/{user_id} Update User

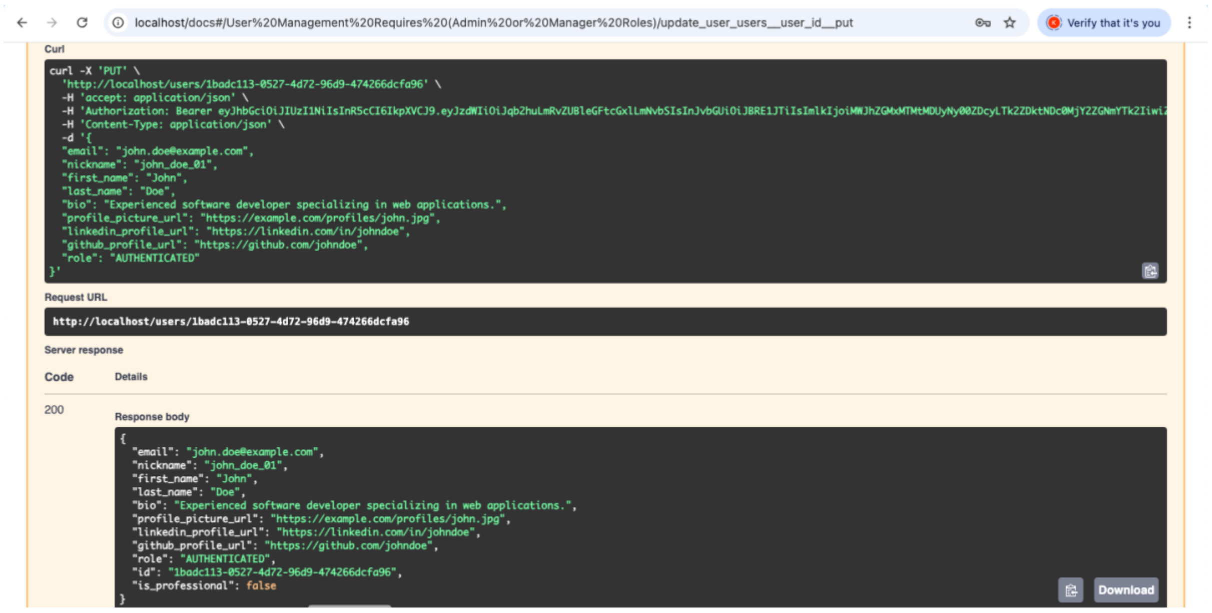
Update user information.

- **user_id**: UUID of the user to update.
- **user_update**: UserUpdate model with updated user information.

Name	Description
user_id * required string($uuid) *(path)*	1badc113-0527-4d72-96d9-474266dcfa96

Request body required

```
{
  "email": "john.doe@example.com",
  "nickname": "john_doe_01",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "AUTHENTICATED"
}
```


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/update_user_users__user_id__put

Curl

```
curl -X 'PUT' \
  'http://localhost/users/1badc113-0527-4d72-96d9-474266dcfa96' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huLmRvZUBleGFtcGxlLmNvbSIsInJvbGUiOiJBRE1JTiIsImlkIjoiMWJhZGMxMTMtMDUyNy00ZDcyLTk2ZDktNDc0MjY2ZGNmYTk2Iiwi
  -H 'Content-Type: application/json' \
  -d '{
  "email": "john.doe@example.com",
  "nickname": "john_doe_01",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "AUTHENTICATED"
}'
```

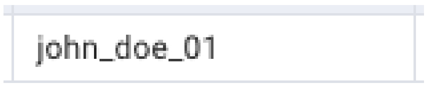
Request URL

```
http://localhost/users/1badc113-0527-4d72-96d9-474266dcfa96
```

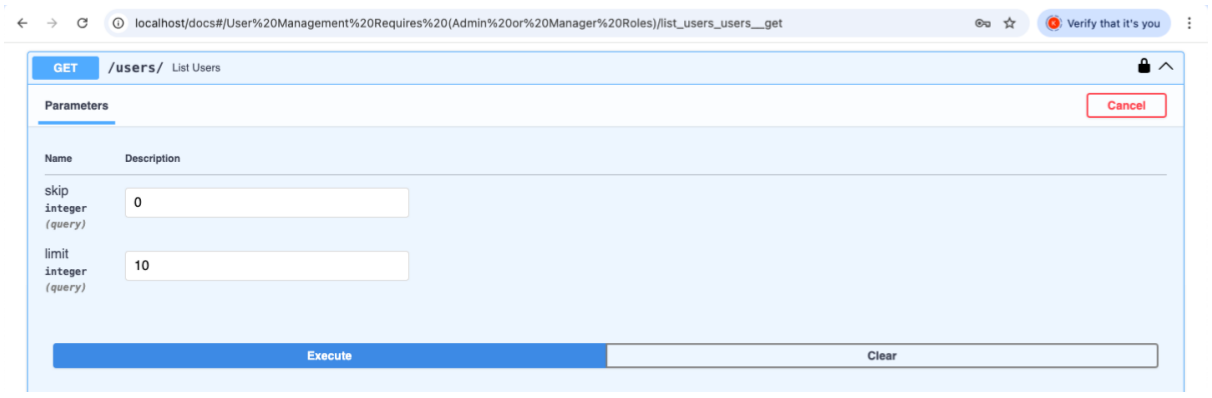
Server response

Code	Details
200	Response body

```
{
  "email": "john.doe@example.com",
  "nickname": "john_doe_01",
  "first_name": "John",
  "last_name": "Doe",
  "bio": "Experienced software developer specializing in web applications.",
  "profile_picture_url": "https://example.com/profiles/john.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/johndoe",
  "github_profile_url": "https://github.com/johndoe",
  "role": "AUTHENTICATED",
  "id": "1badc113-0527-4d72-96d9-474266dcfa96",
  "is_professional": false
}
```


john_doe_01

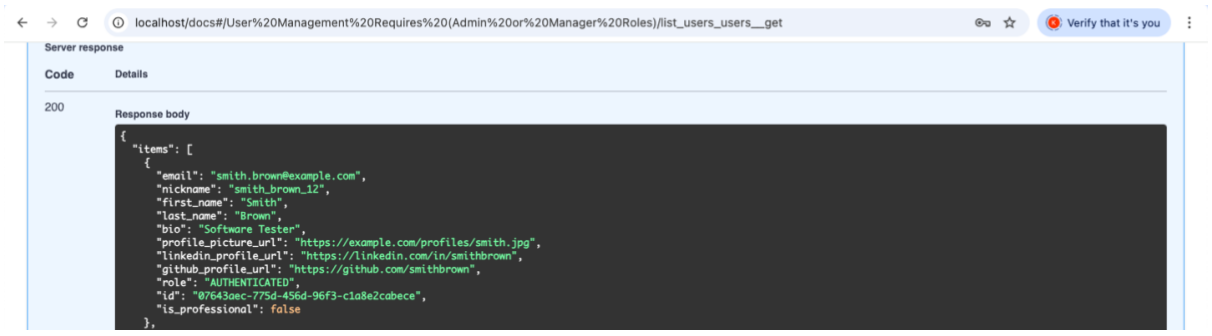
~ LISTING THE USERS:


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/list_users_users__get

GET /users/ List Users

Parameters

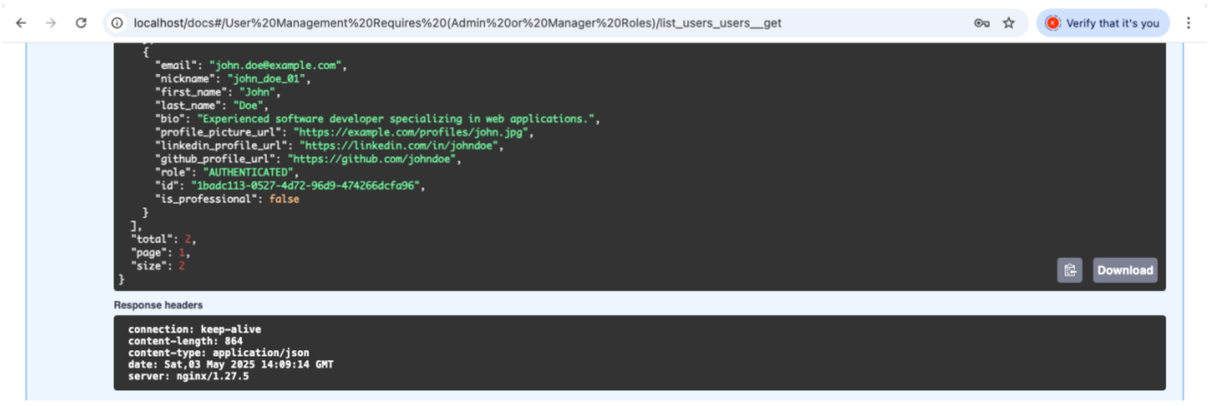
Name	Description
skip integer (query)	0
limit integer (query)	10


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/list_users_users__get

Server response

Code	Details
200	Response body

```
{
  "items": [
    {
      "email": "smith.brown@example.com",
      "nickname": "smith_brown_12",
      "first_name": "Smith",
      "last_name": "Brown",
      "bio": "Software Tester",
      "profile_picture_url": "https://example.com/profiles/smith.jpg",
      "linkedin_profile_url": "https://linkedin.com/in/smithbrown",
      "github_profile_url": "https://github.com/smithbrown",
      "role": "AUTHENTICATED",
      "id": "07643aec-775d-456d-96f3-c1a8e2cabece",
      "is_professional": false
    },
```

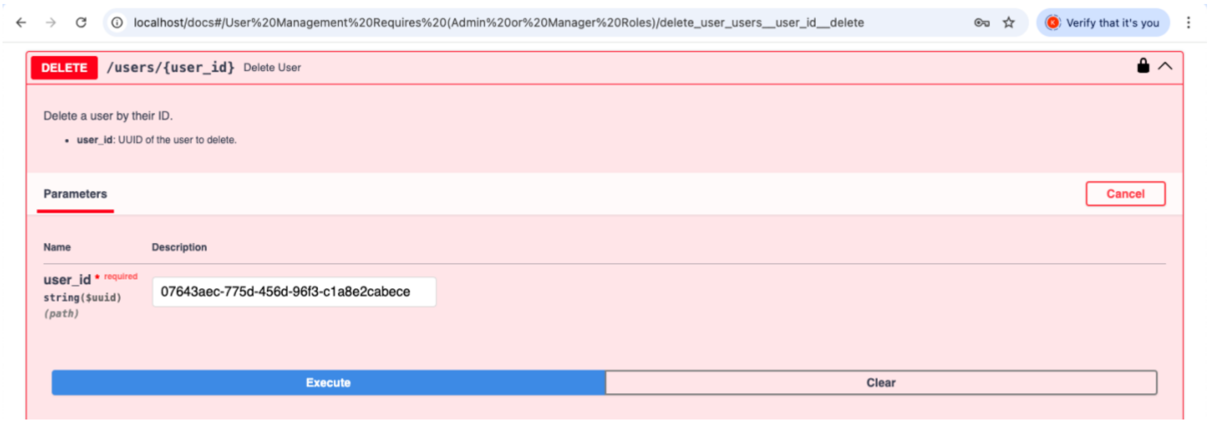

localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/list_users_users__get
Verify that it's you
Download

```
    {
      "email": "john.doe@example.com",
      "nickname": "john_doe_01",
      "first_name": "John",
      "last_name": "Doe",
      "bio": "Experienced software developer specializing in web applications.",
      "profile_picture_url": "https://example.com/profiles/john.jpg",
      "linkedin_profile_url": "https://linkedin.com/in/johndoe",
      "github_profile_url": "https://github.com/johndoe",
      "role": "AUTHENTICATED",
      "id": "1badc113-0527-4d72-96d9-474266dcfa96",
      "is_professional": false
    }
  ],
  "total": 2,
  "page": 1,
  "size": 2
}
```

Response headers

```
connection: keep-alive
content-length: 864
content-type: application/json
date: Sat,03 May 2025 14:09:14 GMT
server: nginx/1.27.5
```

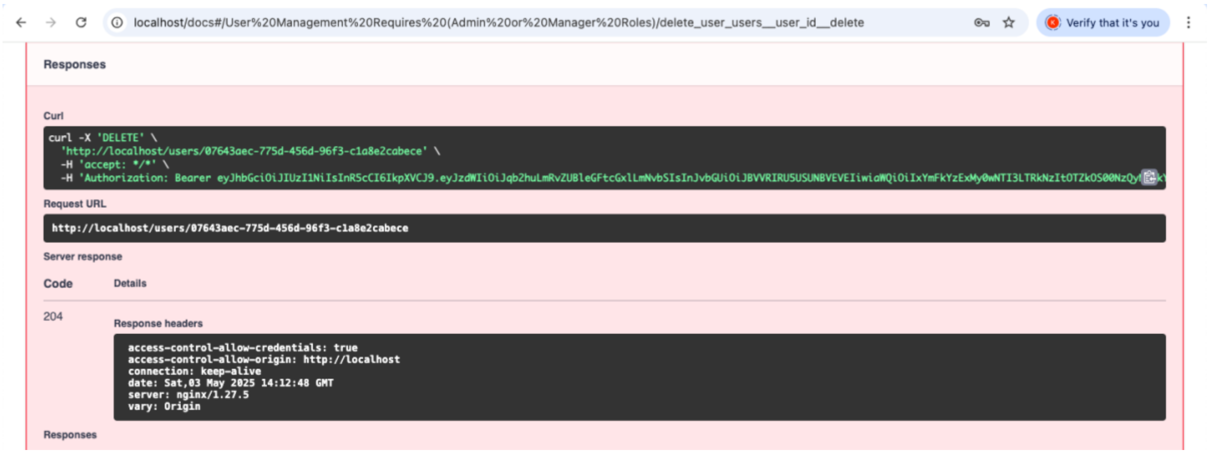
~ DELETING THE USER:


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/delete_user_users__user_id__delete
Verify that it's you
DELETE
/users/{user_id}
Delete User
Parameters
Cancel
Execute
Clear

Delete a user by their ID.

- **user_id**: UUID of the user to delete.

Name	Description
user_id * required string($uuid) (path)	07643aec-775d-456d-96f3-c1a8e2cabece


localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/delete_user_users__user_id__delete
Verify that it's you
Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost/users/07643aec-775d-456d-96f3-c1a8e2cabece' \
  -H 'accept: */*' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huLmRvZUBleGFtcGxlLmNvbSIsInJvbGUiOiJBVVRIRU5USUNBVEVEIiwiaWQiOiIxYmFkYzExMy0wNTI3LTRkNzItOTZkOS00NzQy
```

Request URL

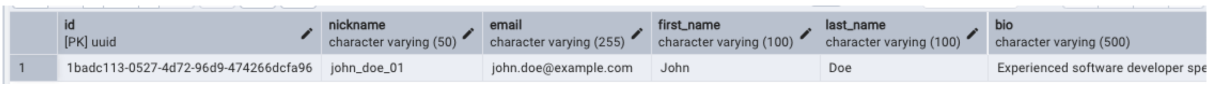
```
http://localhost/users/07643aec-775d-456d-96f3-c1a8e2cabece
```

Server response

Code	Details
204	Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost
connection: keep-alive
date: Sat,03 May 2025 14:12:48 GMT
server: nginx/1.27.5
vary: Origin
```

Responses



	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)	bio character varying (500)
1	1badc113-0527-4d72-96d9-474266dcfa96	john_doe_01	john.doe@example.com	John	Doe	Experienced software developer spe

🏁 FINAL REMARKS:

This project was a significant milestone in my learning. It challenged me to debug, test, and apply DevOps concepts. The iterative development and QA process mimicked real-world processes and has enhanced my technical and collaboration skills.