🔐 Secure File Sharing System

This project is a submission for the Back-End Intern Test. It implements a secure file-sharing system between two types of users: **Ops Users** and **Client Users**. It includes role-based authentication, file access restrictions, and encrypted download links.

👥 User Roles

👨‍💼 Ops User

- ✅ Login
- ✅ Upload `.pptx`, `.docx`, `.xlsx` files only

👤 Client User

- ✅ Sign Up (returns encrypted URL)
- ✅ Email Verification (console-based mock)
- ✅ Login
- ✅ Download file via secure encrypted link
- ✅ List all uploaded files

🚀 Features

- Role-based access using JWT
- File type validation for upload (Ops only)

- Secure, encrypted links for file download
- Token-protected APIs using FastAPI
- Email verification (mock implementation)
- Full API documentation via Swagger UI

🛠️ Tech Stack

- **Framework**: FastAPI (Python)
- **Database**: In-memory (can extend to PostgreSQL/MongoDB)
- **Authentication**: JWT Tokens
- **File Storage**: Local file system
- **Email Service**: Console output
- **Tools Used**: Postman, Uvicorn, Git

🔁 API Endpoints

Method	Endpoint	Description
POST	`/ops/login/`	Ops user login
POST	`/ops/upload/`	Upload a file (restricted types)
POST	`/client/signup/`	Signup new client (returns link)
GET	`/client/verify-email/`	Simulated email verification
POST	`/client/login/`	Client user login
GET	`/download-file/{file_id}/`	Secure file download for clients

| GET | `/files/` | List all uploaded files |

▶️ Running the Server Locally

```bash
uvicorn main:app --reload
```

Then open Swagger UI at:

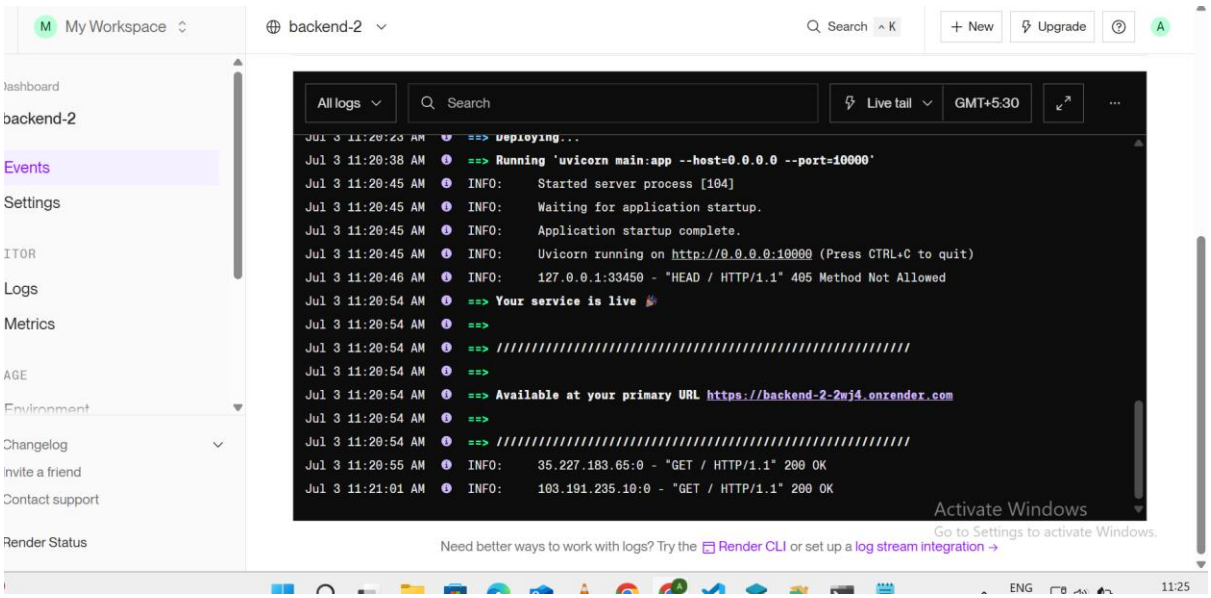
👉 http://127.0.0.1:8000/docs

Use this interface to test all endpoints manually.

Output

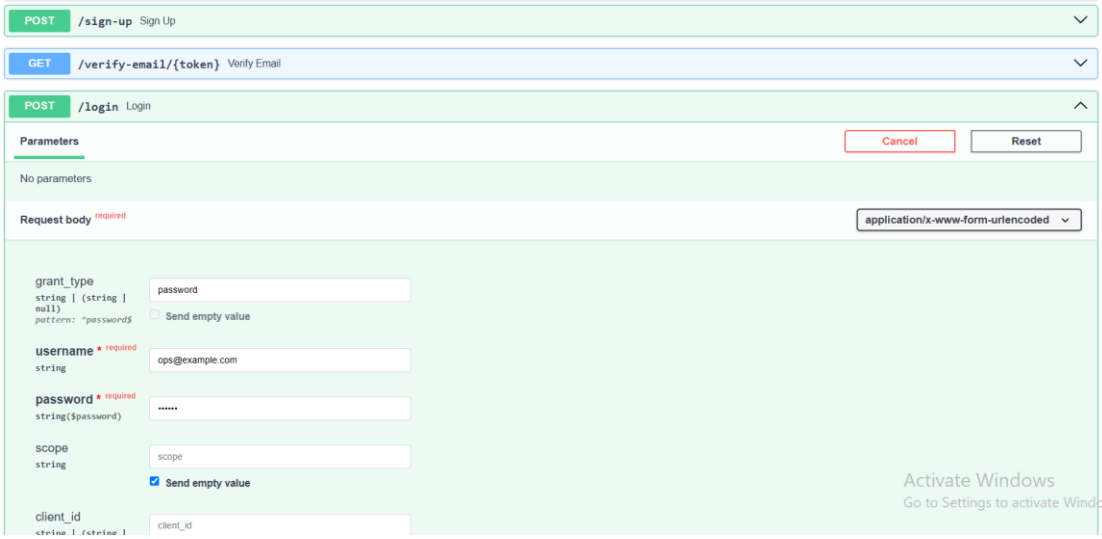
✅ 1. Ops Login

The Ops user logs in using valid credentials. JWT token is returned for authorization in future requests.



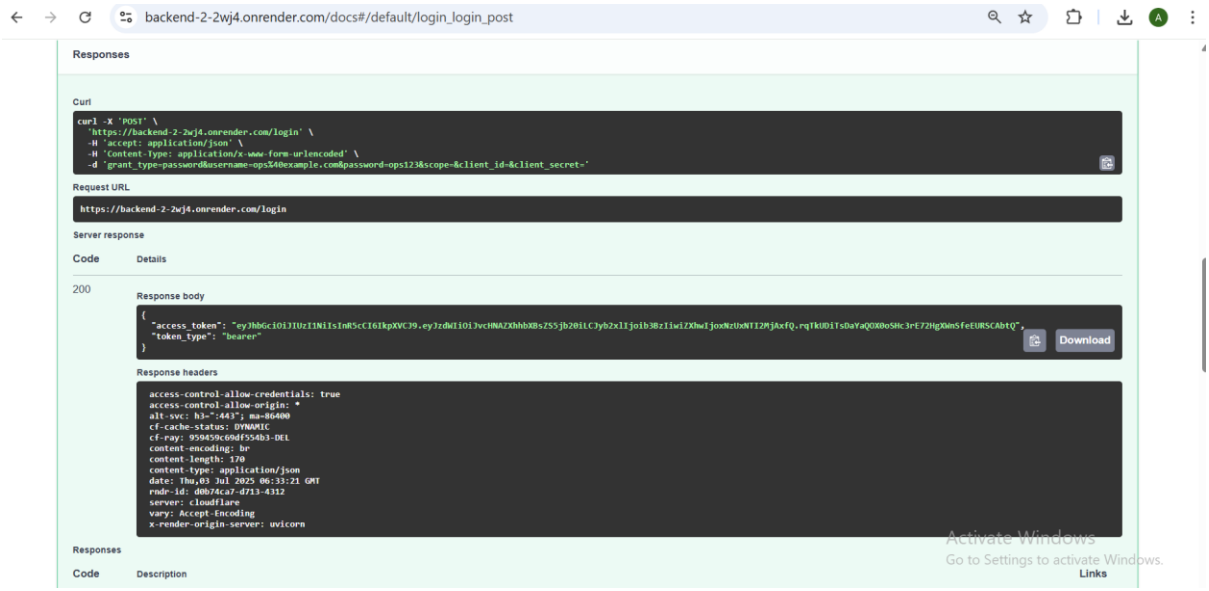
✅ 2. File Upload by Ops- Request

📤 The Ops user uploads a file. Only .docx, .pptx, and .xlsx formats are allowed for security.



✅ 3. File Upload by Ops – Response

File is successfully stored. The API responds with confirmation and metadata of the uploaded file.

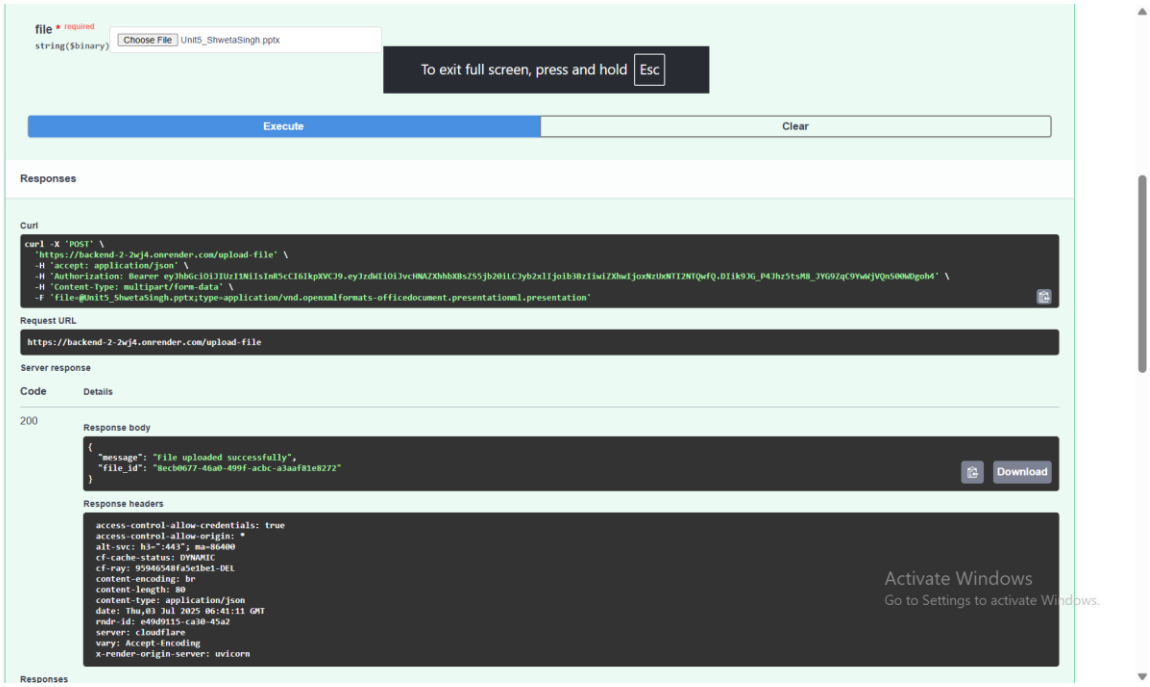


✅ 4. Client Signup with Encrypted URL

A new client registers on the platform. The system returns an encrypted download URL as part of the response.

✅ 5. Email Verification

A mock email (printed in console) is generated for verification. This simulates an actual email delivery system.

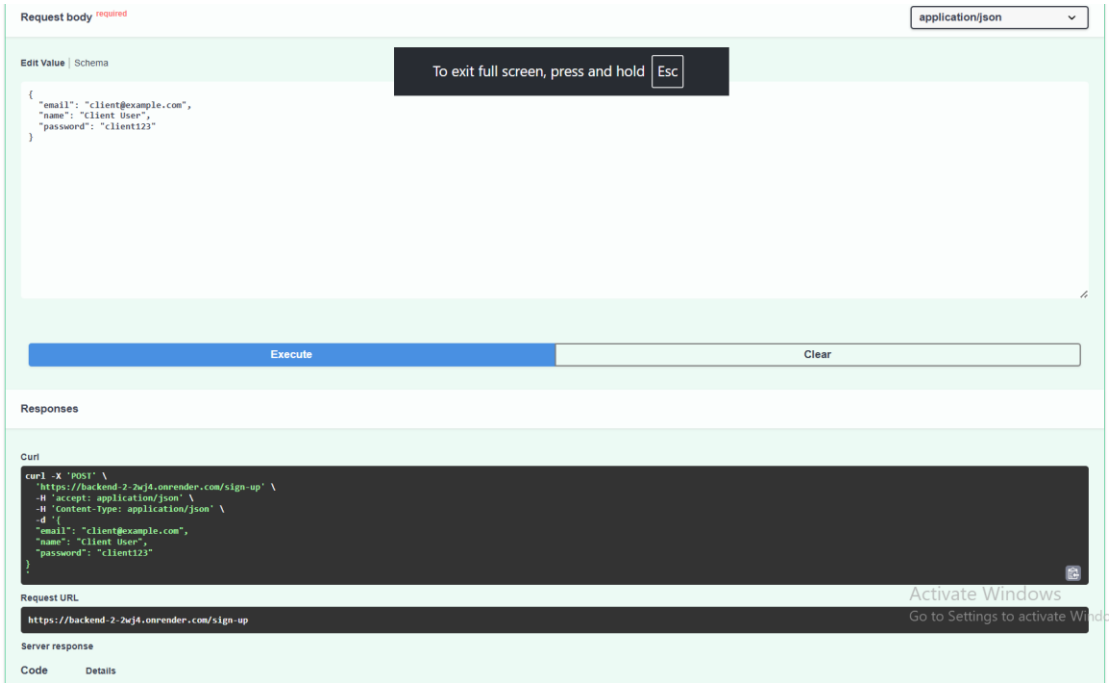


✅ 6. Email Verification Success

The client verifies their email by accessing the verification URL. They are now eligible to access secured routes.

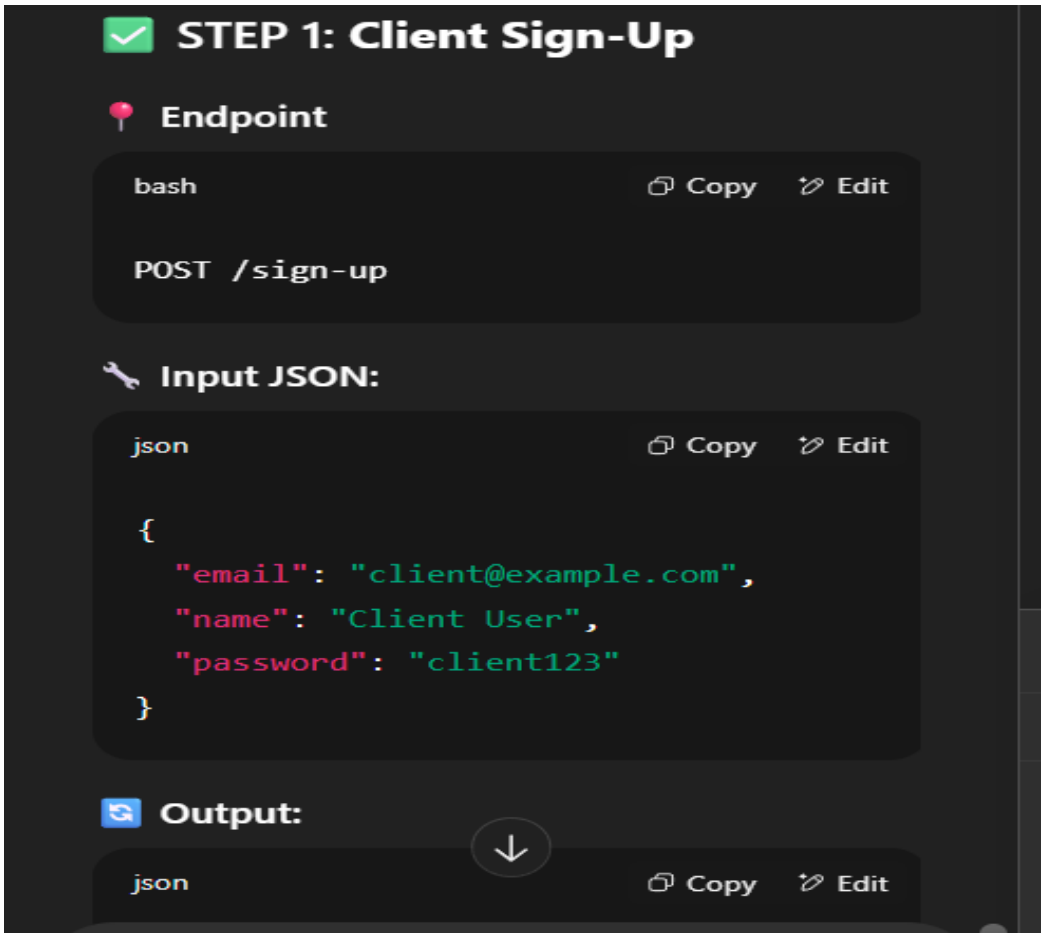
✅ 7. Client Login

🔑 The verified client logs in with credentials. The system checks validity and returns a JWT token.



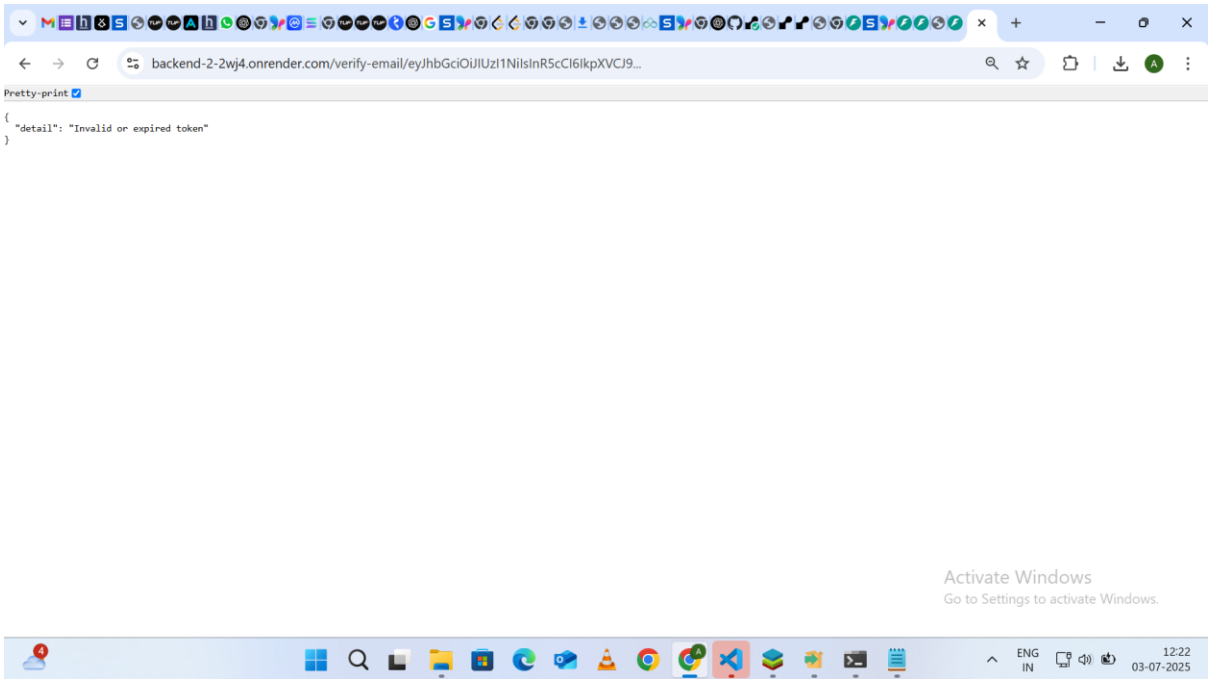
✅ 8. Secure File Download

📥 The client requests to download a file using the secure encrypted URL. Access is granted only to verified clients.



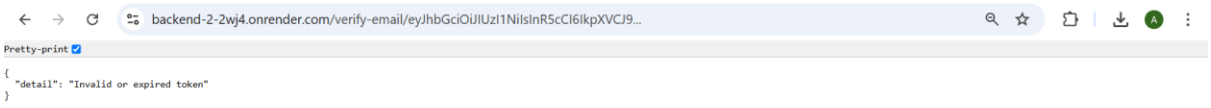
✅ 9. List Uploaded Files

📂 The client fetches a list of all uploaded files using their token. This route is restricted to client users only.



✅ 10. Unauthorized Access Attempt

❌ An unauthorized Ops user attempts to access a client-only route. The system denies access and returns an error.



🧪 Testing

Manual testing performed using:

✅ Swagger UI

✅ Postman Collection

Test cases cover:

Role-based access (Ops vs Client)

File format restrictions

Secure link-based downloads

🧳 Deployment Plan

This project can be deployed on:

- 🔹 Render
- 🔹 Railway
- 🔹 Docker on AWS EC2 / DigitalOcean / Azure

🔐 For production:

Replace in-memory DB with PostgreSQL

Configure .env for secrets

Use Gunicorn + Nginx for serving FastAPI app

📬 Postman Collection

Included: postman_collection.json

You can import this into Postman to test all APIs with sample payloads and tokens.

📎 Repository

🔗 GitHub: https://github.com/NishuKaushik/Backend-Project

👤 Author

Nishu Kaushik

📧 nishukaushik166@gmail.com

🔗 GitHub Profile