

🔒 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.

My Workspace

backend-2

All logs

Search

Live tail

GMT+5:30

Logs

Metrics

AGE

Environment

Changelog

Invite a friend

Contact support

Render Status

```

JUL 3 11:20:23 AM => Deploying...
Jul 3 11:20:38 AM => Running 'uvicorn main:app --host=0.0.0.0 --port=10000'
Jul 3 11:20:45 AM INFO: Started server process [104]
Jul 3 11:20:45 AM INFO: Waiting for application startup.
Jul 3 11:20:45 AM INFO: Application startup complete.
Jul 3 11:20:45 AM INFO: Uvicorn running on http://0.0.0.0:10000 (Press CTRL+C to quit)
Jul 3 11:20:46 AM INFO: 127.0.0.1:33450 - "HEAD / HTTP/1.1" 405 Method Not Allowed
Jul 3 11:20:54 AM => Your service is live 🚀
Jul 3 11:20:54 AM => ===
Jul 3 11:20:54 AM => =====
Jul 3 11:20:54 AM => ===
Jul 3 11:20:54 AM => Available at your primary URL https://backend-2-2wj4.onrender.com
Jul 3 11:20:54 AM => ===
Jul 3 11:20:54 AM => =====
Jul 3 11:20:55 AM INFO: 35.227.183.65:0 - "GET / HTTP/1.1" 200 OK
Jul 3 11:21:01 AM INFO: 103.191.235.10:0 - "GET / HTTP/1.1" 200 OK

```

Activate Windows

Go to Settings to activate Windows.

Need better ways to work with logs? Try the [Render CLI](#) or set up a [log stream integration](#).

ENG 11:25

## ✓ 2. File Upload by Ops- Request

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

POST /sign-up Sign Up

GET /verify-email/{token} Verify Email

POST /login Login

Parameters

No parameters

Request body required

application/x-www-form-urlencoded

grant_type	password
string   (string   null)	<input type="checkbox"/> Send empty value
pattern: "passwords"	
username *	ops@example.com
string	.....
password *	*****
string(\$password)	
scope	scope
string	<input checked="" type="checkbox"/> Send empty value
client_id	client_id
string   (string   null)	

Activate Windows

Go to Settings to activate Windows.

## ✓ 3. File Upload by Ops – Response

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

The screenshot shows a browser window with the URL [https://backend-2-2wj4.onrender.com/docs#/default/login\\_login\\_post](https://backend-2-2wj4.onrender.com/docs#/default/login_login_post). The page displays the API documentation for the login endpoint. It includes a 'Responses' section with a 'Curl' example, a 'Request URL' (https://backend-2-2wj4.onrender.com/login), and a 'Server response'. The response is a 200 OK status with a JSON body containing an access token and a token type. Below the response, there are sections for 'Response headers' and 'Response body'. A 'Download' button is available for the response body. At the bottom right, there are links to 'Activate Windows' and 'Go to Settings to activate Windows.'

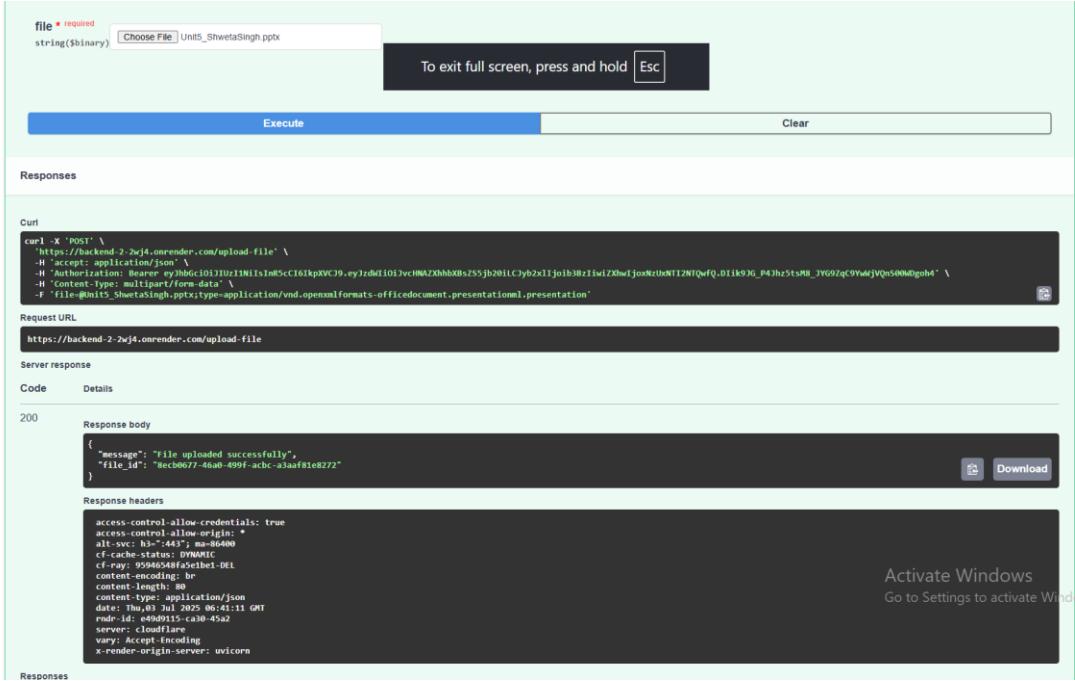
#### ✓ 4. Client Signup with Encrypted URL

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

The screenshot shows a browser window with the URL [https://backend-2-2wj4.onrender.com/docs#/default/upload\\_file\\_upload\\_file](https://backend-2-2wj4.onrender.com/docs#/default/upload_file_upload_file). The page displays the API documentation for the upload-file endpoint. It includes a 'Responses' section with a 'Code' column (200) and a 'Description' column ('Successful Response'). Under the 'Successful Response' row, there is a 'Media type' dropdown set to 'application/json'. Below it, there are 'Example Value' and 'Schema' sections, both showing a JSON object with 'access\_token' and 'token\_type' fields. A 'Validation Error' row is also present, with a similar structure. At the bottom right, there are links to 'Activate Windows' and 'Go to Settings to activate Windows.'

## 5. Email Verification

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



The screenshot shows a web-based interface for file upload. At the top, there is a file input field labeled "file \* required" with "Choose File" button and "Units\_ShwetaSingh.pptx" selected. Below the file input is a "Curl" section containing a complex curl command for file upload. Underneath is a "Request URL" section with the URL "https://backend-2-2wj4.onrender.com/upload-file". The main area is titled "Responses" and contains two tabs: "Code" and "Details". The "Code" tab shows a status code of 200. The "Details" tab displays the JSON response body: {"message": "file uploaded successfully", "file\_id": "8ecb6077-46a0-49f-acbc-a3aaef81e8272"}. Below the response body is a "Response headers" section with a long list of HTTP headers. A watermark for "Activate Windows" and "Go to Settings to activate Windows." is visible in the bottom right corner of the responses area.

## 6. Email Verification Success

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

The screenshot shows a REST API documentation interface. At the top, it says "x-render-origin-server: uvicorn". Below that is a "Responses" section with two entries:

- Code**: 200 **Description**: Successful Response  
Media type: application/json (dropdown)  
Example Value | Schema  
"string"
- Code**: 422 **Description**: Validation Error  
Media type: application/json (dropdown)  
Example Value | Schema  
{"detail": [{"loc": ["string"], "msg": "string", "type": "string"}]}

Links: No links

## ✓ 7. Client Login

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

The screenshot shows a REST API documentation interface for a "Client Login" endpoint. It includes a "Request body" section with a required field, a "Responses" section with a "Curl" example, and a "Request URL" section.

**Request body** required

application/json (dropdown)

Edit Value | Schema

```
{ "email": "client@example.com", "name": "Client User", "password": "client123" }
```

To exit full screen, press and hold Esc

**Responses**

Curl

```
curl -X 'POST' \ https://backend-2-2wj4.onrender.com/sign-up' \ -H 'accept: application/json' \ -H 'Content-Type: application/json' \ -d { "email": "client@example.com", "name": "Client User", "password": "client123" }
```

Activate Windows  
Go to Settings to activate Windows.

Request URL

https://backend-2-2wj4.onrender.com/sign-up

Server response

Code Details

## ✓ 8. Secure File Download

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

### ✓ STEP 1: Client Sign-Up

📍 Endpoint

bash      ⚒ Copy ⚒ Edit

POST /sign-up

⚡ Input JSON:

json      ⚒ Copy ⚒ Edit

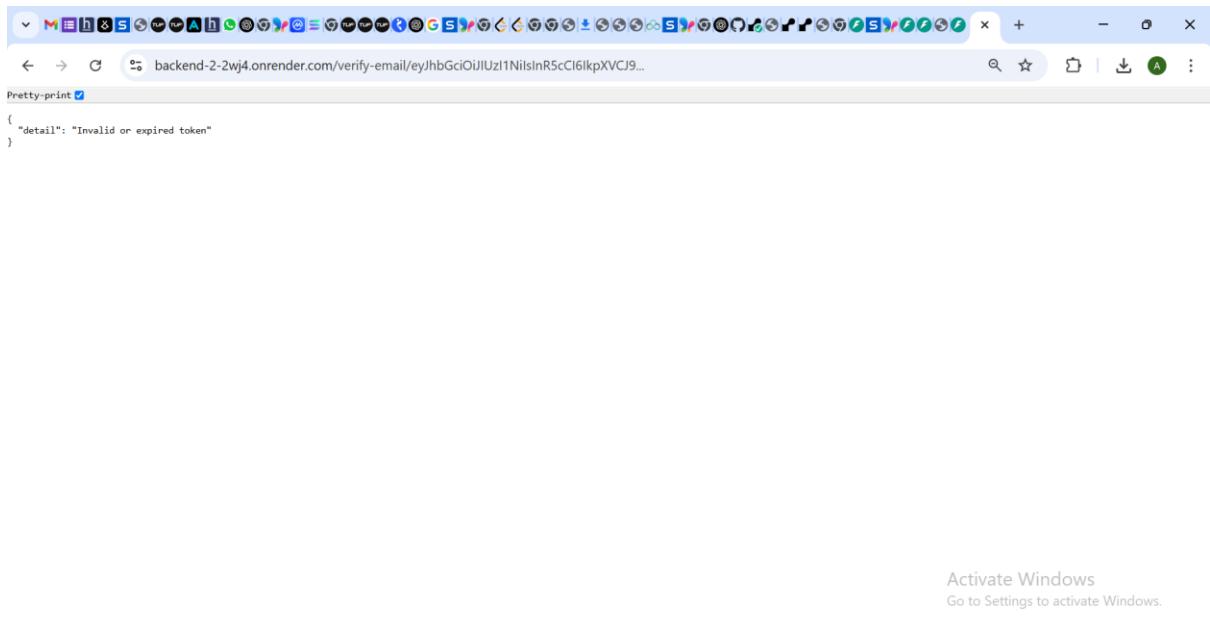
```
{
 "email": "client@example.com",
 "name": "Client User",
 "password": "client123"
}
```

🔄 Output:

json      ⚒ Copy ⚒ Edit

✓ 9. List Uploaded Files

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



A screenshot of a web browser window. The address bar shows a URL starting with "backend-2-2wj4.onrender.com/verify-email/eyJhbGciOiJIUzI1NilsInR5cCl6IkpxVCJ9...". The main content area displays a JSON object with the following content:

```
Pretty-print
{
 "detail": "Invalid or expired token"
}
```

Activate Windows  
Go to Settings to activate Windows.



## 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:

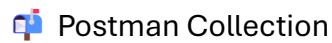
- ◆ Render
- ◆ Railway
- ◆ Docker on AWS EC2 / DigitalOcean / Azure

 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>



Nishu Kaushik



nishukaushik166@gmail.com



GitHub Profile