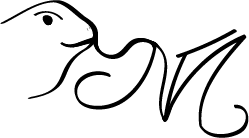
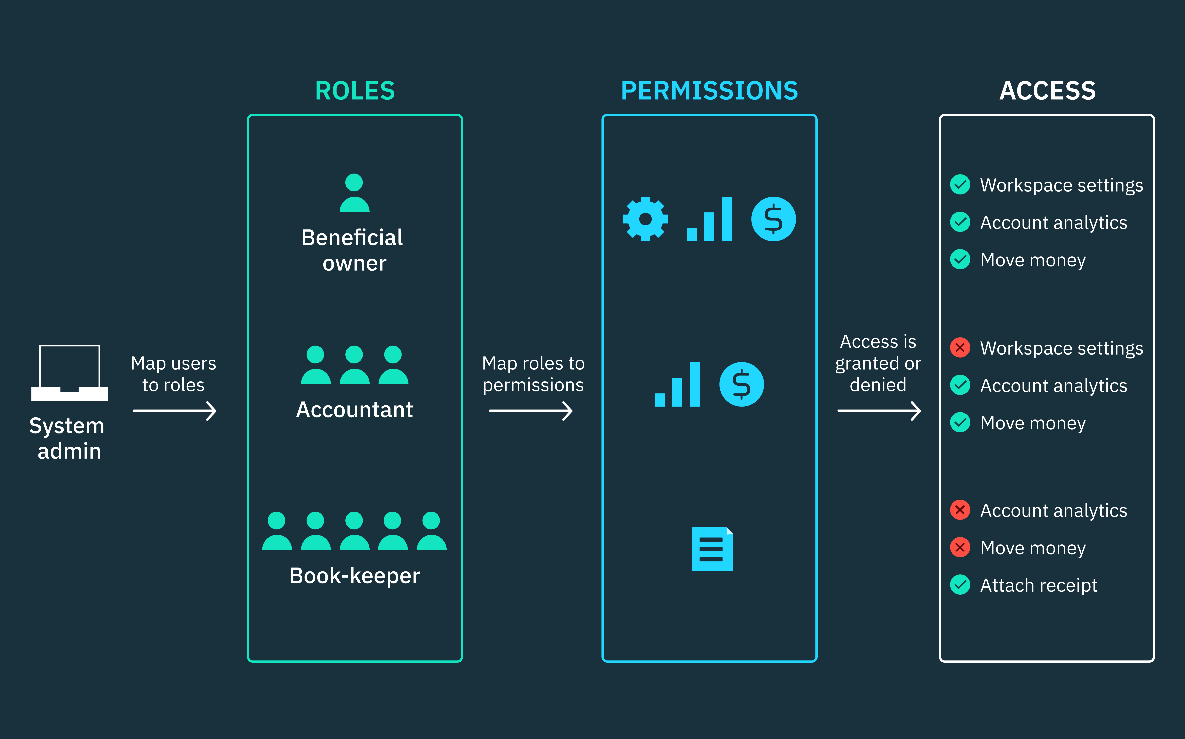
****

IMY 120 – Logo – Adobe Course

**RBAC + MFA - LizardLock**

Practical 2  
Michael Todd  
U23540223



**My PC Specifications**

* Windows 11 Home with DirectX 12
* Intel Core i7-10700, 8 cores and 16 threads, base clock 2.90GHz
* 16GB RAM
* 1TB SSD
* NVIDIA GeForce GTX 1650 SUPER with 1280 CUDA cores, 4GB GDDR6 memory
* There are some overclocks present on the RAM and GPU
* Python 3.13.5

**Commands**

These are just all general commands I either made or used during development…

My README file stipulates the most pain free way to set this up locally in order :D

pip install -r requirements.txt

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

python manage.py runserver

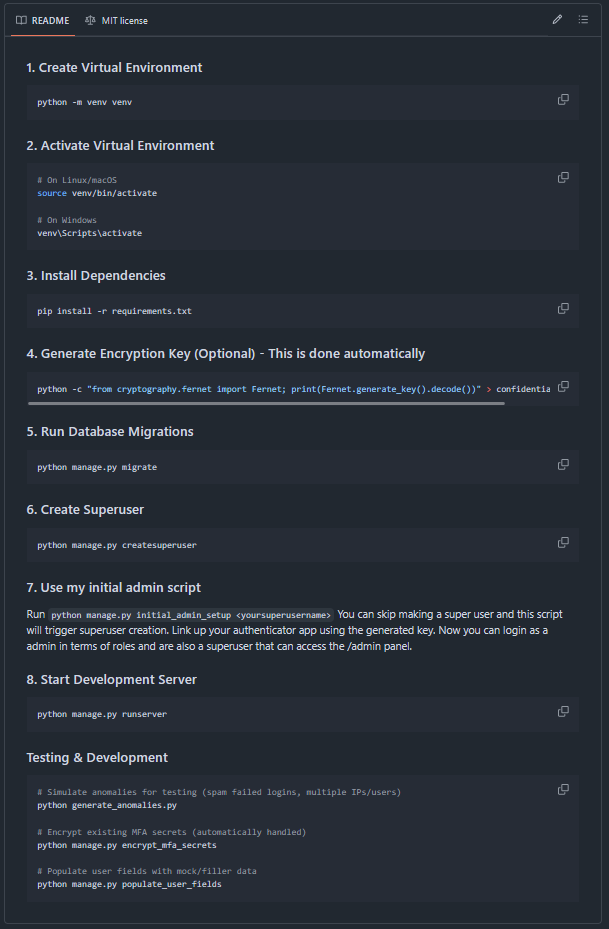
python manage.py encrypt\_mfa\_secrets (I forgot that plaintext MFA secrets are vulnerable – so I just made a quick script to encrypt existing users – “lizards” mfa\_secret – the codebase now does this automatically)

python manage.py populate\_user\_fields (Same thing here I was just using usernames – as that’s Djangos default form – made a custom one – this just populated my existing users with some rudimentary filler stuff)

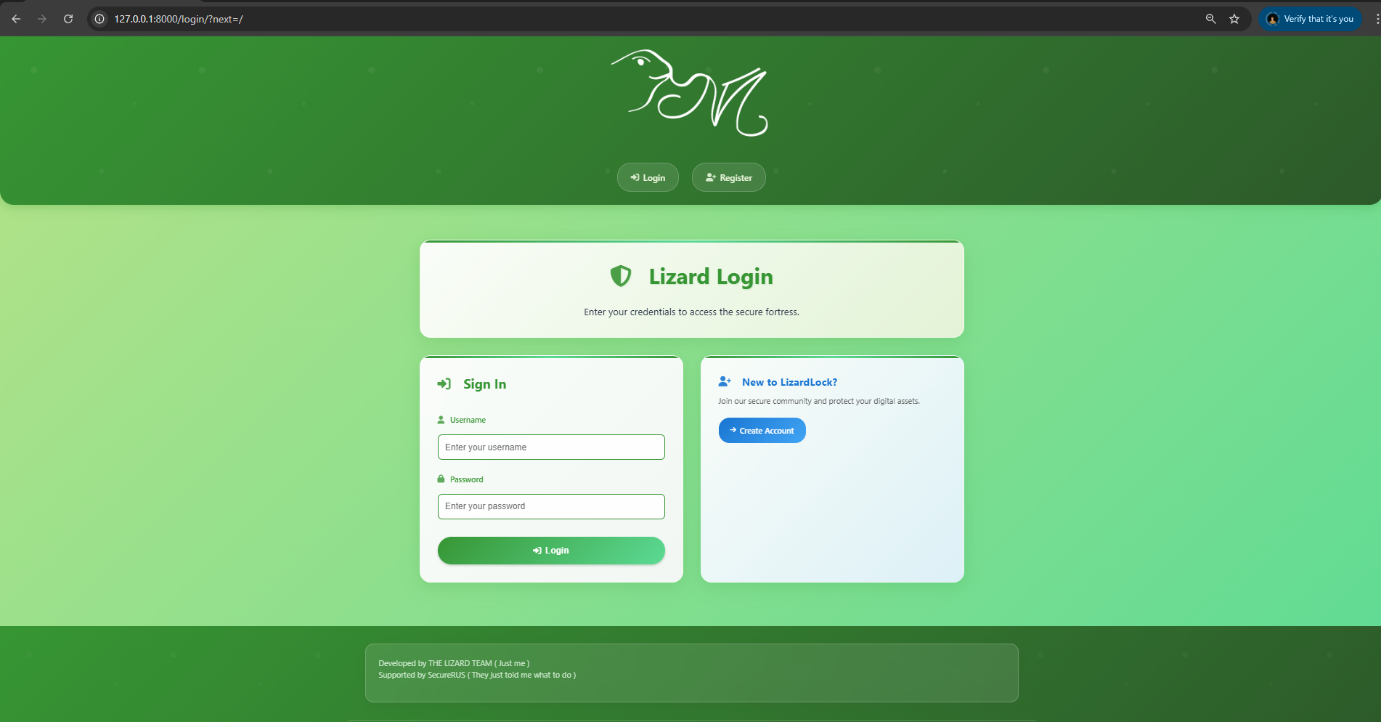
python generate\_anomalies.py (This script essentially spams my application to trigger anomalies – Failed attempts – Multiple IPs – Multiple Users same IP – High Failure rates)

python manage.py initial\_admin\_setup <superusername> (This script helps make a user with admin roles and is also a superuser that can access the admin panel)  
I plan on changing the codebase around quite a bit because I want to host this on my own Linode server, this is also why I spent quite some time styling it exactly how I want it and adding IP tracing – because if Felix (PewDiePie) is degoogling his life then so can I – We don’t need em! #HomeLabbing

**PROJECT SETUP – STEP BY STEP**

****

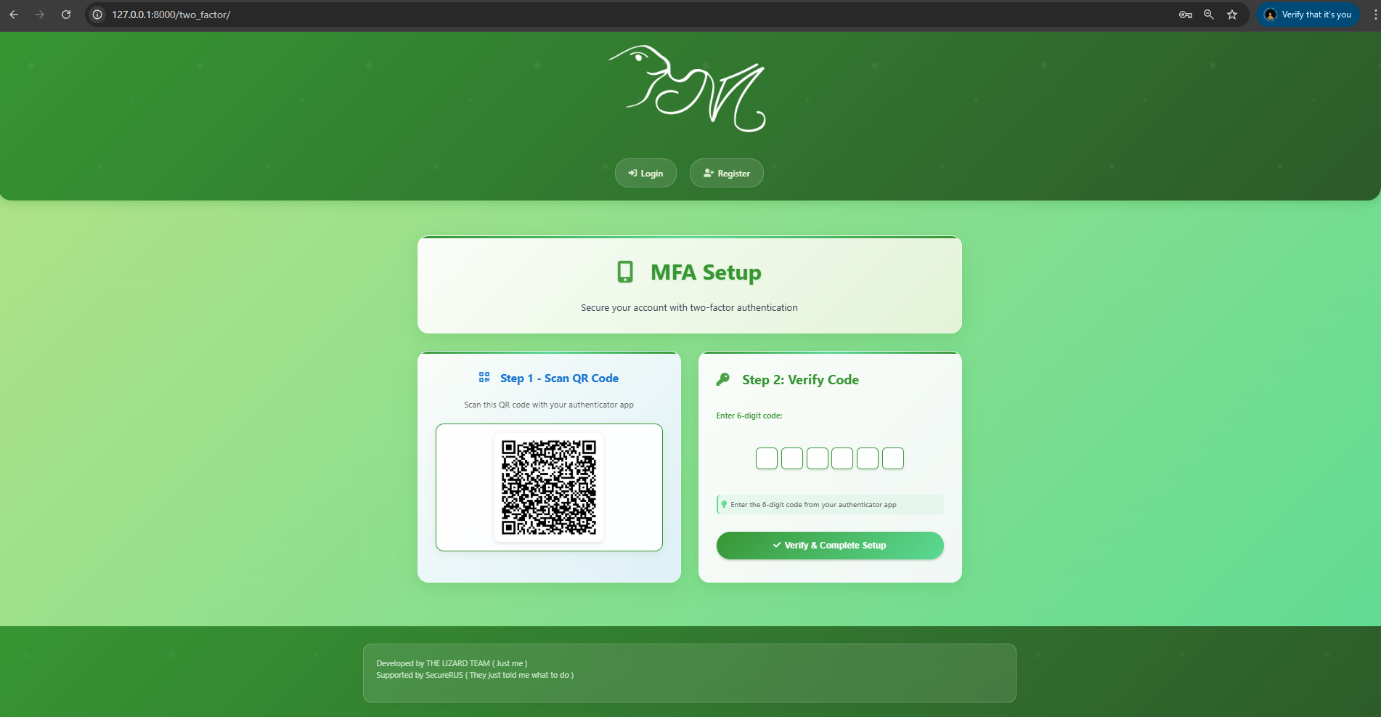
**UNIVERAL PAGES**

**LOGIN**

**REGISTER**

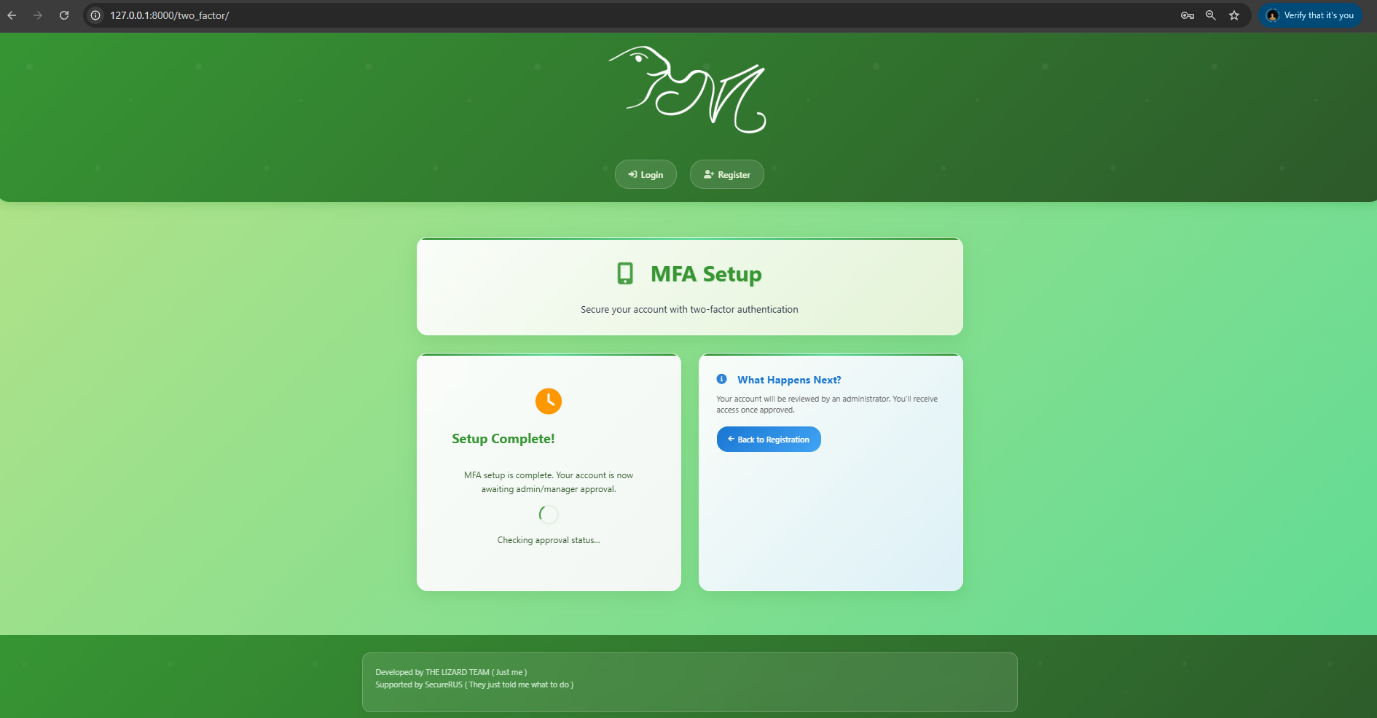
**A screenshot of a computer

AI-generated content may be incorrect.**

**TWO FACTOR – SETUP  
  
  
  
  
  
  
  
  
  
  
VERIFY**

**A screenshot of a computer

AI-generated content may be incorrect.**

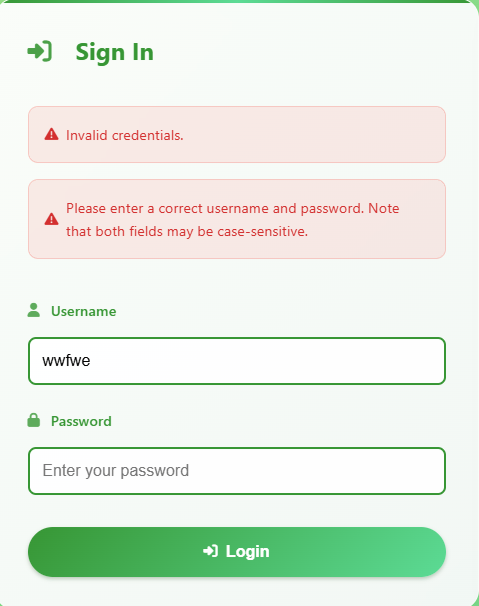
**TWO FACTOR – AWAITING APPROVAL**

**VALIDATIONS**

**A screenshot of a login screen

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a phone

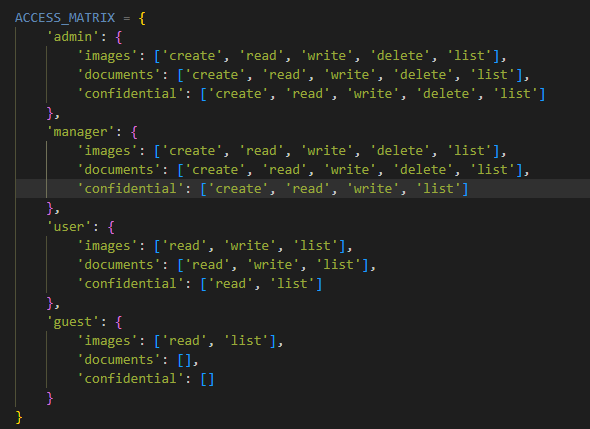
AI-generated content may be incorrect.A green rectangle with red text

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.**

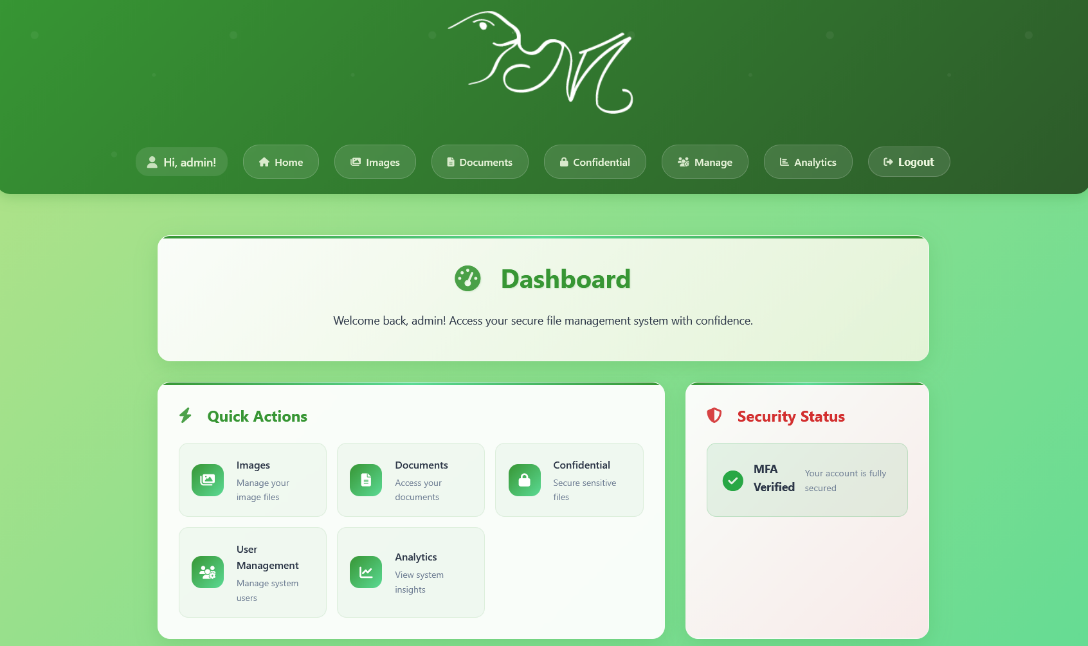
**SECURITY FEATURES**

* I implemented TOTP with a ±30 second tolerance to handle clock synchronization issues between server and client devices. It uses the HMAC-SHA1 algorithm compatible with Google Authenticator.
* MFA secrets are encrypted at rest using Fernet (AES-128) to prevent exposure if the database is compromised.
* To prevent brute force attacks, I track login attempts per IP. This will be useful when deployed to a server like my Linode. I also separate the checking logic from incrementing attempts to avoid false positives during page loads.
* Confidential files are encrypted before storage. Even with file system access, contents remain protected. It uses AES-128 encryption.
* I prevent file tampering and ensure data integrity using SHA256 for strong cryptographic validation.
* Files are served directly, bypassing the web server to maintain access control. Security headers are applied to prevent MIME sniffing and enforce correct content handling.
* I prevent directory traversal and file system exploitation by sanitizing file names before storage.
* Confidential operations require recent MFA verification (30-minute timeout) to prevent session hijacking.
* I create a forensic trail for security incidents by tracking user actions, IP addresses, timestamps, and success/failure statuses.
* I improve user experience while maintaining security by preventing unauthorized actions during approval wait periods.
* To prevent malicious file uploads, I validate the actual content type, not just the file extension.
* MFA verification expires over time, requiring re-authentication for extended sessions to balance security with usability.

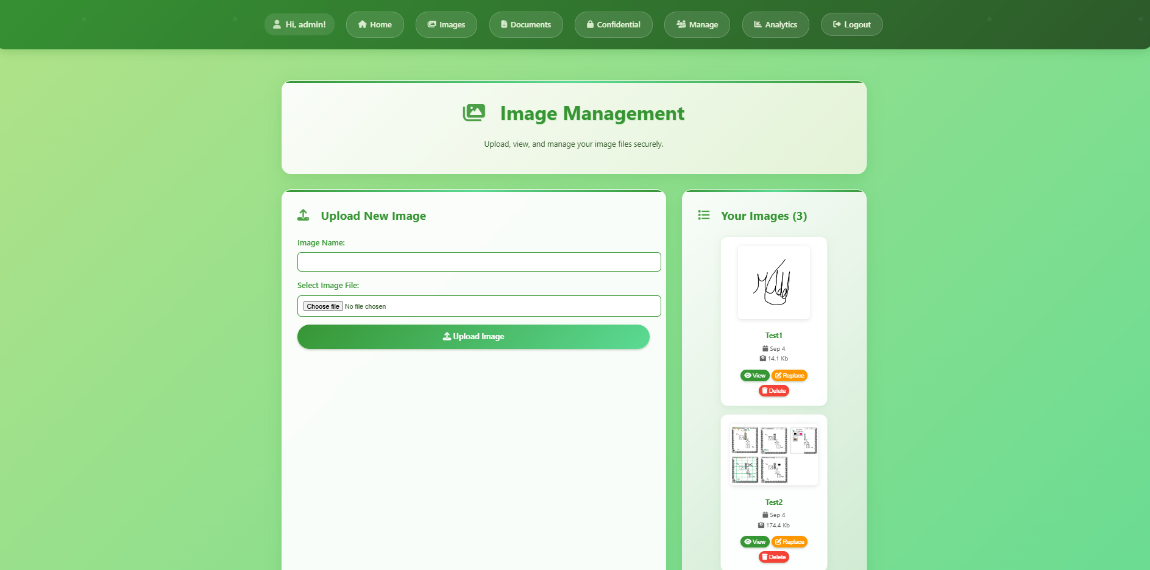
**POSITIVE AND NEGATIVE SCENARIOS**

Just take note of the navbar and possible actions on the images/documents/confidentials  
( I don’t want to screen shot everything xD )  
(You will need to zoom in a lot)

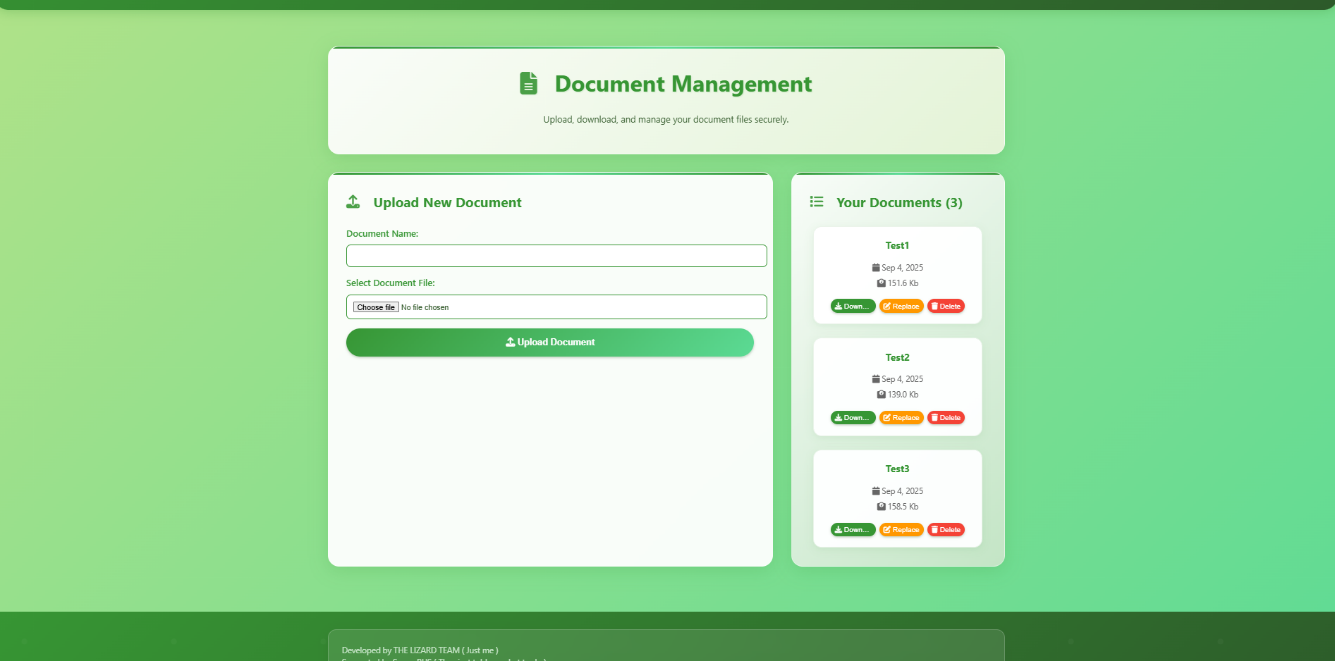
**ADMIN – LOGGED IN**

****

**ADMIN – IMAGES**

****

**ADMIN – DOCUMENTS**

****

**A screenshot of a computer

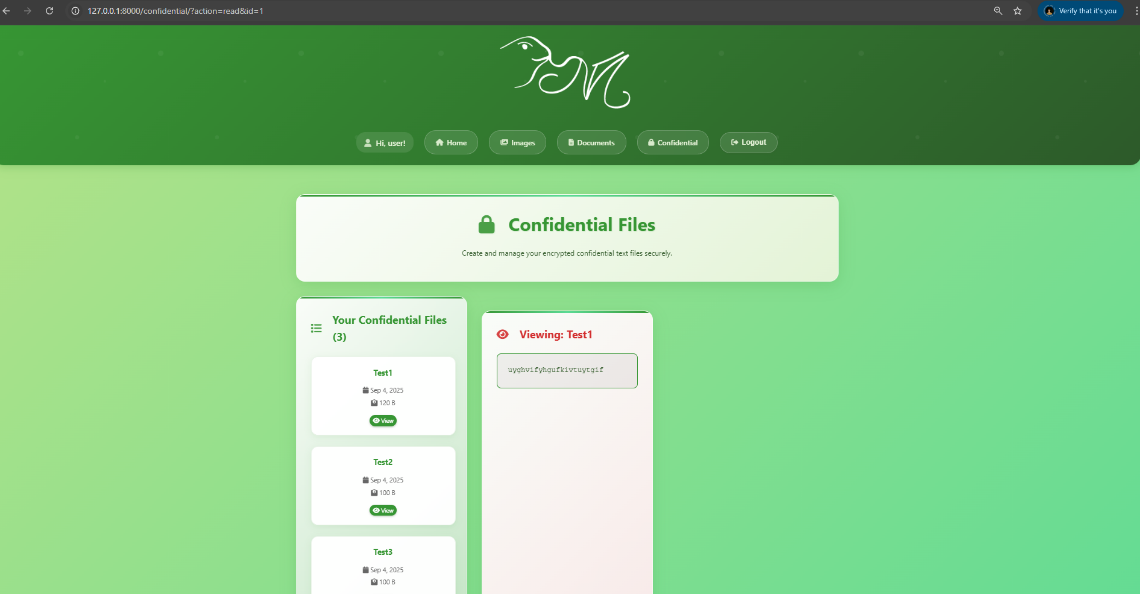
AI-generated content may be incorrect.ADMIN – CONFIDENTIAL**

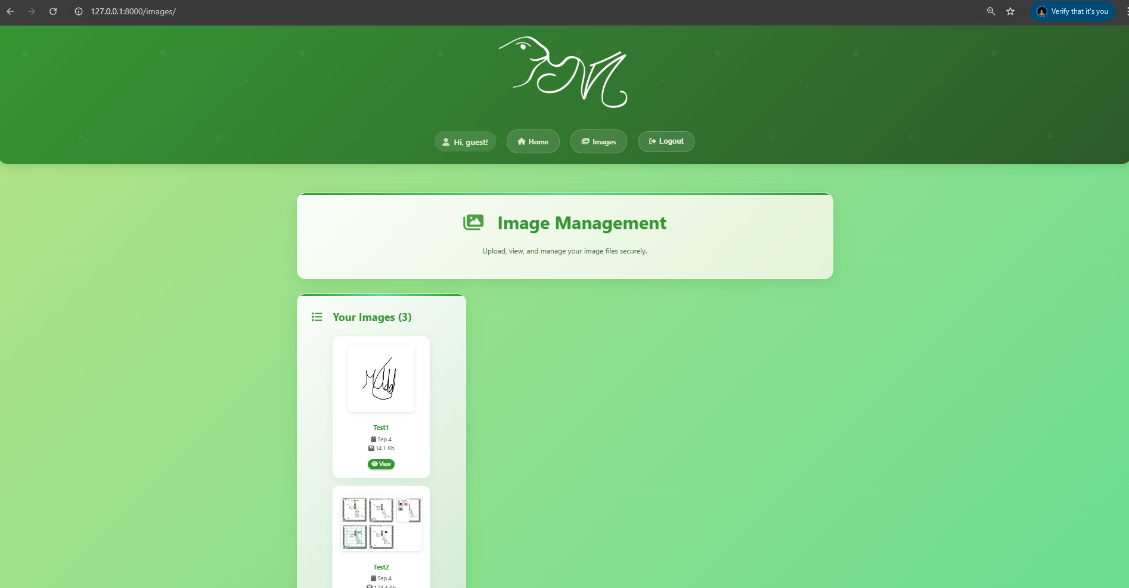
**NEGATIVE SCENARIOS PER ROLE**

For length reasons I will attach one image of a random page of lower roles…

Just to demonstrate the access restrictions and permissions quickly.

**USER – CONFIDENTIAL PAGE**



**GUEST - IMAGES**

**MANAGER - CONFIDENTIALS**



**TASK 3 – ANOMALY ALGORITHM**

****

**TASK 3 - ALGORITHM**

This is a rule-based system that analyses access logs in real time to flag suspicious behaviour and potential security threats.

The main reason I added IP based rules is for when this code base is hosted.

**Detection Rules**

* >5 failed logins per user – High severity
* >3 IPs per user – Medium severity
* >5 users on same IP – Medium severity
* >10 failed attempts from one IP – High severity

**How It Works**

* Scans the 100 most recent logs
* Applies simple thresholds to detect anomalies
* Tags each finding with a severity level
* Outputs clear, actionable alerts

**What I think this helps protect against**

Helps detect brute force attacks, credential sharing, and account compromise.

Runs in a single pass with O(n) complexity – Cause my Linode Server has one CPU and like no RAM at all XD, integrated into the /analytics dashboard for real-time use.

**VIDEO  
My video will be within the ZIP of the Practical**

**REFERENCES**

**Django Framework (3.0) Crash Course Tutorial**  
I used a lot of design principles for Django from this awesome playlist – beforehand I had never used Django, so this was quite a big “breakthrough” in terms of deeper understanding of the framework which helped me understand my errors a lot more, instead of just coding some dumb stuff in random files and hoping it works.  
  
I had also had a full time job in software development – All massive PHP Laravel projects – So the core ideas in my head helped.  
  
I did acknowledge that it was 5 years old (Django 3.0) and when something that sounded deprecated or I knew other frameworks had these features already existing I just did a google search.

[**https://youtube.com/playlist?list=PL-51WBLyFTg2vW-\_6XBoUpE7vpmoR3ztO&si=KARgeYAtrxktLVts**](https://youtube.com/playlist?list=PL-51WBLyFTg2vW-_6XBoUpE7vpmoR3ztO&si=KARgeYAtrxktLVts)

**CLOUDFARE - Website security guide: A 10-step checklist**  
This helped me keep an on-tap reminder of what a secure website needs to cover its surfaces as there are so many.

[**https://www.cloudflare.com/learning/security/glossary/website-security-checklist/**](https://www.cloudflare.com/learning/security/glossary/website-security-checklist/)