# PRESENTATION

Team members:

Sara Atallah

Shadwa Ahmed

Yehia Tarek

Marawan Farouk

Mohammed Tamer

# Core Technology And Setup

```python
- Framework: Flask (Python web framework)
- Database: MySQL with flask_mysqldb
- Authentication: Flask-Bcrypt for password hashing
- Session Management: Flask-Session
- OAuth Integration: Flask-Dance for Google and GitHub login
- Security: SSL/HTTPS enabled
```

## Framework: Flask

- Flask is a lightweight Python web framework.
- It’s used to build web applications and APIs quickly and easily.

## Database: MySQL with flask_mysqldb

- The project uses a MySQL database to store data.
- flask_mysqldb is a Flask extension that helps you connect to and interact with the MySQL database.

## Authentication: Flask-Bcrypt

- Used for securely hashing passwords.

It ensures that stored passwords are encrypted and safe from attackers.

## Session Management: Flask-Session

- Manages user sessions (like login state) across pages.
- Stores session data on the server side instead of browser cookies, which is more secure.

## OAuth Integration: Flask-Dance

Allows users to log in using their Google or GitHub accounts.

Simplifies third-party login by handling the OAuth process.

## Security: SSL/HTTPS Enabled

- Your website uses HTTPS to encrypt the connection between the user and the server.
- This prevents eavesdropping or tampering with data in transit.

# SECURITY FEATURES

```python
a) Authentication:
- Multiple authentication methods:
  * Traditional username/password
  * Two-factor authentication (2FA) using TOTP
  * OAuth (Google and GitHub)
- Password hashing using Bcrypt
- Session management with secure configurations

b) File Security:
- File encryption/decryption
- Digital signatures
- File integrity verification through hashing
- Secure file upload handling

c) Access Control:
- Role-based access (admin vs regular users)
- Login required decorator
- Admin required decorator
```

## a) Authentication

Your system uses strong, layered authentication:

- ✅ Username/Password: Traditional login method.
- ✅ Two-Factor Authentication (2FA): Adds an extra layer of security using TOTP (Time-based One-Time Passwords) — for example, codes from Google Authenticator.
- ✅ OAuth: Allows users to log in via Google or GitHub safely.
- ✅ Bcrypt: Hashes passwords before storing them, making them secure even if the database is exposed.
- ✅ Secure Sessions: Manages user sessions with settings that prevent hijacking or tampering.

## b) File Security

You're protecting files at multiple levels:

- 🔐 Encryption/Decryption: Files are encrypted before storing and decrypted when needed.
- ✍️ Digital Signatures: Ensures files are authentic and haven't been tampered with.
- ✅ Hashing for Integrity: Verifies that a file hasn't changed (detects corruption or malicious edits).
- 📤 Secure Uploads: Only allows safe file types and checks content before saving.

## c) Access Control

Controls what users can and can't do based on roles:

- 👤 Role-Based Access: Different features for admin and regular users.
- 🔐 Login Required Decorator: Blocks access to certain pages unless the user is logged in.
- 👮 Admin Required Decorator: Restricts access to admin-only pages.

# KEY FEATURES

## A)user managment

```python
- User registration with 2FA setup
- Profile management
- Profile photo handling
- Password change functionality
- OAuth account linking
```

User accounts support registration with 2FA, profile and photo management, password changes, and OAuth account linking.

## B)Document Management

```python
- Secure file upload
- File encryption
- Document signing
- File integrity verification
- Download functionality
- Edit document properties
```

The system offers secure file uploads, encryption, digital signing, integrity checks, download support, and document property editing

# c)Admin features

```python
- User management (add/edit/delete)
- Document management
- System logs viewing
- Administrative dashboard
```

**Admins can manage users and documents, view system logs, and access a centralized administrative dashboard.**

# Security Best Practices Implemented

```python
- HTTPS/SSL enforcement
- Secure session configuration
- CSRF protection
- XSS prevention
- Secure file handling
- Input validation
- Password policy enforcement
- Rate limiting (implicit through session management)
```

The system enforces HTTPS/SSL and uses secure session configurations to protect user data.

It includes CSRF protection, XSS prevention, secure file handling, and strong input validation.

Password policies and rate limiting are applied to enhance security and reduce abuse

# Logging and Monitoring

```python
- Comprehensive logging system
- Both file and database logging
- Action tracking
- Error logging with stack traces
- Rotating file handler implementation
```

The system features comprehensive logging, storing logs in both files and the database. It tracks user actions and logs errors with stack traces for easier debugging.

A rotating file handler is used to manage log file size and ensure long-term storage.

# code structure analysis

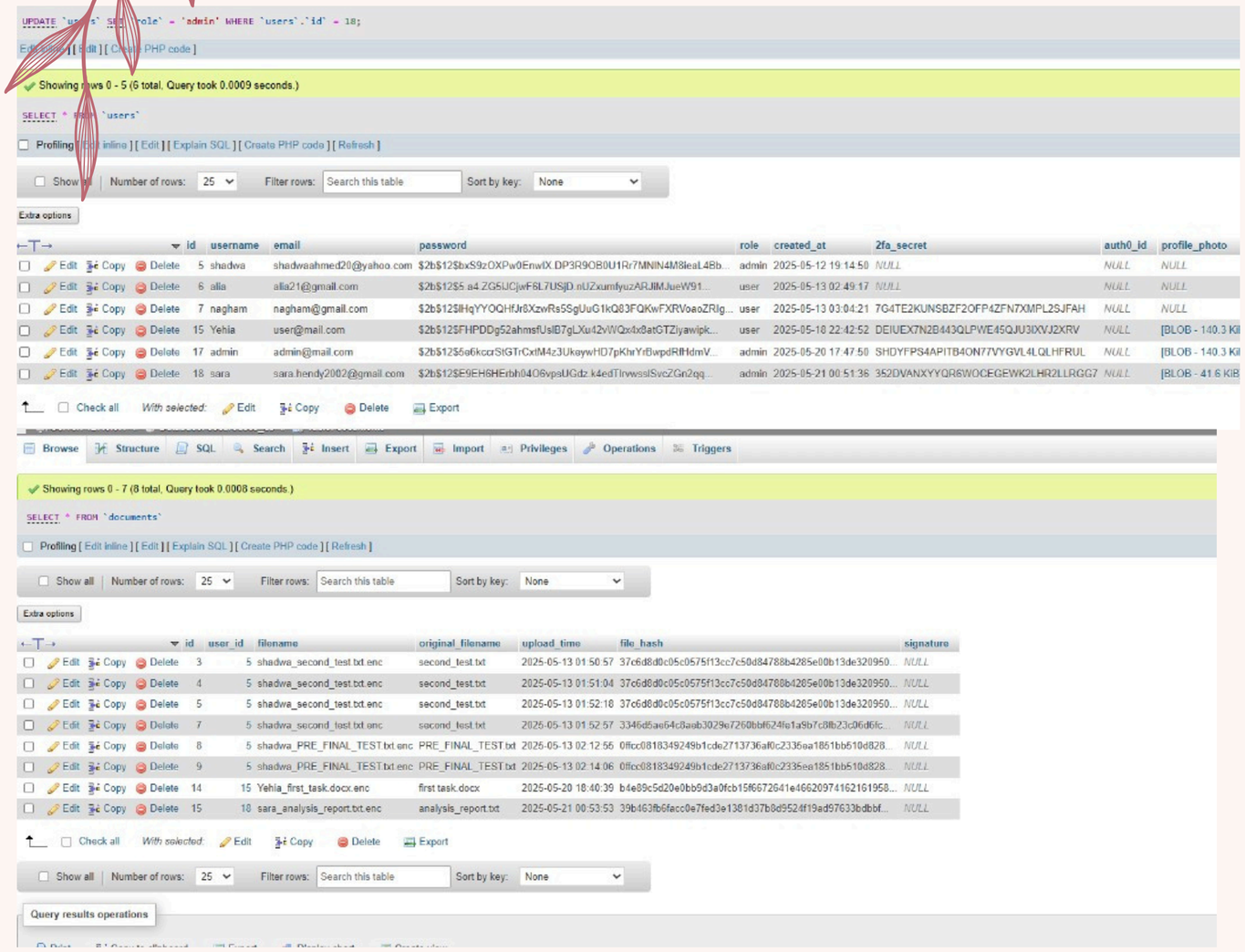

**The system follows best practices with well-structured route handlers, proper error handling, and a clear separation of concerns.**
**It maintains consistent logging, strong security measures, thorough input validation, and uses database transactions appropriately.**

# Areas For Improvments

```
1. Could benefit from:
   - Code modularization (splitting into multiple files)
   - Implementation of a service layer
   - Database model abstraction
   - Configuration management improvements
   - API documentation
   - Unit tests
```

**The project would be improved by modularizing the code and introducing a service layer for better structure. Abstracting database models and enhancing configuration management will improve maintainability. Adding API documentation and unit tests will support scalability and long-term reliability**

# Data Base Tables



```
UPDATE `users` SET `role` = 'admin' WHERE `users`.`id` = 18;
```

[ Edit ] [ Edit ] [ Create PHP code ]

Showing rows 0 - 5 (6 total, Query took 0.0009 seconds.)

```
SELECT * FROM `users`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

| | id | username | email | password | role | created_at | 2fa_secret | auth0_id | profile_photo |
|---|---|---|---|---|---|---|---|---|---|
| Edit Copy Delete | 5 | shadwa | shadwaahmed20@yahoo.com | $2b$12$bxS9zOXPw0EnwIX.DP3R9OB0U1Rr7MNlN4M8leaL4Bb... | admin | 2025-05-12 19:14:50 | NULL | NULL | NULL |
| Edit Copy Delete | 6 | alia | alia21@gmail.com | $2b$12$5.a4.ZG5IJCjwF8L7USjD.nUZxumfyuzARJIMJueW91... | user | 2025-05-13 02:49:17 | NULL | NULL | NULL |
| Edit Copy Delete | 7 | nagham | nagham@gmail.com | $2b$12$IlHqYYOQHfJr8XzwRs5SgUuG1kQ83FQKwFXRVoaoZRIg... | user | 2025-05-13 03:04:21 | 7G4TE2KUNSBZF2OFP4ZFN7XMPL2SJFAH | NULL | NULL |
| Edit Copy Delete | 15 | Yehia | user@mail.com | $2b$12$FHPDDg52ahmsfUsIB7gLXu42vWQx4x8atGTZiyawipk... | user | 2025-05-18 22:42:52 | DEIUEX7N2B443QLPWE45QJU3IXVJ2XRV | NULL | [BLOB - 140.3 KiB] |
| Edit Copy Delete | 17 | admin | admin@mail.com | $2b$12$5a6kccrStGTrCxtM4z3UkeywHD7pKhrYrBwpdRfHdmV... | admin | 2025-05-20 17:47:50 | SHDYFPS4APITB4ON77VYGVL4LQLHFRUL | NULL | [BLOB - 140.3 KiB] |
| Edit Copy Delete | 18 | sara | sara.hendy2002@gmail.com | $2b$12$E9EH6HErbh04O6vpsUGdz.k4edTlrvwsslSvcZGn2qq... | admin | 2025-05-21 00:51:36 | 352DVANXYYQR6WOCEGEWK2LHR2LLRGG7 | NULL | [BLOB - 41.6 KiB] |

Check all | With selected: Edit Copy Delete Export

Browse | Structure | SQL | Search | Insert | Export | Import | Privileges | Operations | Triggers

Showing rows 0 - 7 (8 total, Query took 0.0008 seconds.)

```
SELECT * FROM `documents`
```

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

| | id | user_id | filename | original_filename | upload_time | file_hash | signature |
|---|---|---|---|---|---|---|---|
| Edit Copy Delete | 3 | 5 | shadwa_second_test.txt.enc | second_test.txt | 2025-05-13 01:50:57 | 37c6d8d0c05c0575f13cc7c50d84788b4285e00b13de320950... | NULL |
| Edit Copy Delete | 4 | 5 | shadwa_second_test.txt.enc | second_test.txt | 2025-05-13 01:51:04 | 37c6d8d0c05c0575f13cc7c50d84788b4285e00b13de320950... | NULL |
| Edit Copy Delete | 5 | 5 | shadwa_second_test.txt.enc | second_test.txt | 2025-05-13 01:52:18 | 37c6d8d0c05c0575f13cc7c50d84788b4285e00b13de320950... | NULL |
| Edit Copy Delete | 7 | 5 | shadwa_second_test.txt.enc | second_test.txt | 2025-05-13 01:52:57 | 3346d5ae64c8aab3029e7260bbf624fe1a9b7c8fb23c06d6fc... | NULL |
| Edit Copy Delete | 8 | 5 | shadwa_PRE_FINAL_TEST.txt.enc | PRE_FINAL_TEST.txt | 2025-05-13 02:12:55 | 0ffcc0818349249b1cde2713736af0c2335ea1851bb510d828... | NULL |
| Edit Copy Delete | 9 | 5 | shadwa_PRE_FINAL_TEST.txt.enc | PRE_FINAL_TEST.txt | 2025-05-13 02:14:06 | 0ffcc0818349249b1cde2713736af0c2335ea1851bb510d828... | NULL |
| Edit Copy Delete | 14 | 15 | Yehia_first_task.docx.enc | first task.docx | 2025-05-20 18:40:39 | b4e89c5d20e0bb9d3a0fcb15f66726 41e46620974162161958... | NULL |
| Edit Copy Delete | 15 | 18 | sara_analysis_report.txt.enc | analysis_report.txt | 2025-05-21 00:53:53 | 39b463fb6facc0e7fed3e1381d37b8d9524f19ad97633bdbbf... | NULL |

Check all | With selected: Edit Copy Delete Export

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Query results operations

# Environmental Configurations

```
- Database settings
- Upload folder configuration
- Session settings
- OAuth credentials
- SSL certificates
- Logging configurations
```

**Centralized configuration includes database settings, file upload paths, session management, and OAuth credentials.**
**It also manages SSL certificate paths and logging settings for consistent and secure behavior.**

**Using environment variables or .env files helps keep sensitive data secure and manageable.**

# Error Handling

```
- Custom error handlers
- Comprehensive exception catching
- User-friendly error messages
- Detailed logging of errors
- Transaction rollbacks on failures
```

**Robust error handling includes custom error pages and comprehensive exception catching. It ensures user-friendly feedback while logging detailed error info for debugging.**

**Transaction rollbacks maintain data integrity during failures.**

# Recommendations for Enhancement:

## 1.Technical Improvements

```python
- Implement rate limiting
- Add request validation middleware
- Implement API versioning
- Add caching mechanisms
- Implement background tasks for file processing
```

**Enhance performance and security by adding rate limiting and request validation middleware.**

**Introduce API versioning for maintainability and use caching for efficiency.**

**Offload heavy tasks like file processing to background jobs for smoother operation.**

# 2.Security Enhancements

```python
- Add password reset functionality
- Implement IP-based blocking
- Add session timeout
- Implement file type validation
- Add virus scanning for uploads
```

Apply to app.py

**Improve security by adding password reset, session timeout, and IP-based blocking features.**

**Validate file types and integrate virus scanning to ensure safe file uploads.**

# 3.Feature Additions

```python
- Document sharing functionality
- Version control for documents
- Batch upload/download
- Document categories/tags
- Search functionality
- Export/import functionality
```

**The system provides secure document sharing, version control, and batch upload/download capabilities. It supports organizing documents using categories and tags for easier management. Powerful search and export/import functions enhance accessibility and data portability.**

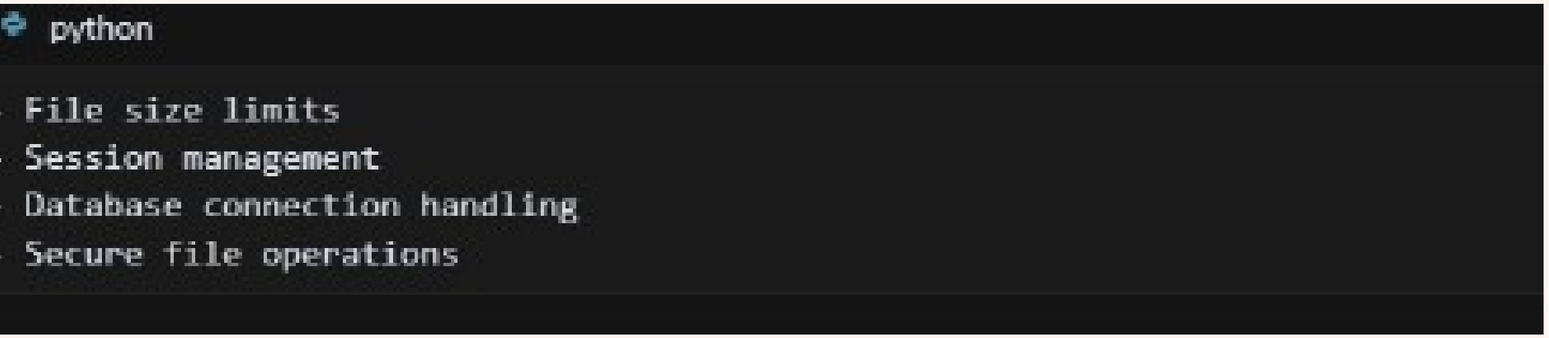
# 4.Code Organization

```
- Split into modules
- Create service layer
- Implement repository pattern
- Add configuration management
- Add comprehensive testing
```

**The system architecture is modularized for better maintainability and scalability. A service layer and repository pattern are implemented to separate concerns and manage data access cleanly. Configuration management and comprehensive testing ensure flexibility, reliability, and robustness.**

# Performance Considerations:

## Current Implementations




**The system enforces file size limits to optimize performance and storage usage. It includes robust session management and reliable database connection handling. Secure file operations ensure data integrity and protection against unauthorized access.**

# Potential Improvements

```
- Implement caching
- Add pagination
- Optimize database queries
- Implement async operations
- Add load balancing support
```

**The system enhances performance by implementing caching, pagination, and optimized database queries. Asynchronous operations improve responsiveness and scalability. Load balancing support ensures high availability and efficient resource utilization.**

# Conclusion

This codebase represents a secure document management system with robust authentication and authorization mechanisms. While it implements core security features well, there's room for improvement in terms of code organization, scalability, and additional features. The system appears to be production-ready but could benefit from additional modularity and testing infrastructure.

The application follows security best practices and provides a solid foundation for handling sensitive documents securely. Its comprehensive logging and error handling make it maintainable and debuggable in production environments.

# THANK YOU