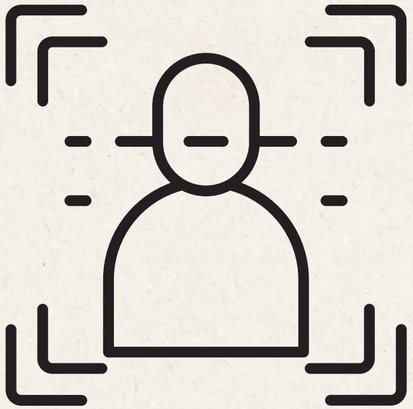
SECURE AUTHENTICATION SYSTEM

PRESENTED BY: ADIL, AIDOS, ARSEN

Problem Statement

Why  matters?

Goal # 1

Make a working secure login system

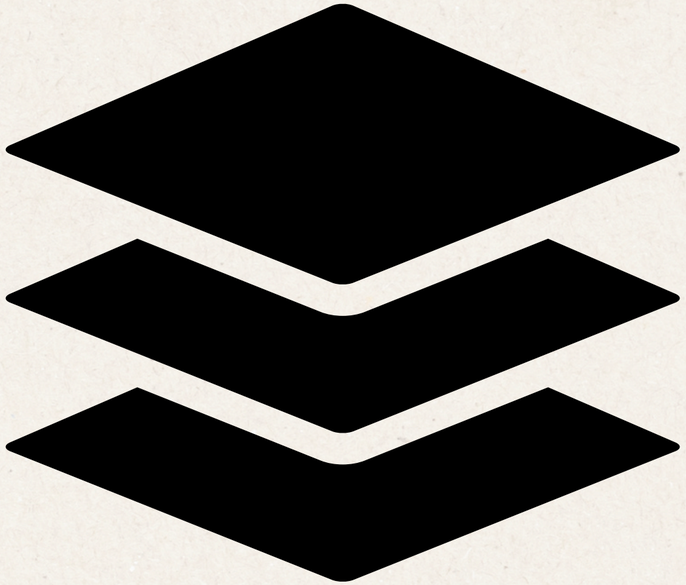
Goal # 2

Implement safe password reset

Goal # 3

Show cryptography in action

Architecture Overview



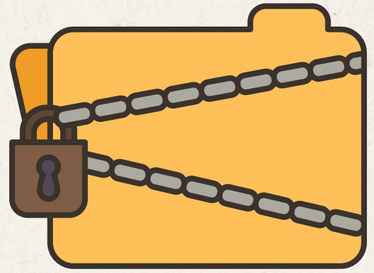
5-layer architecture

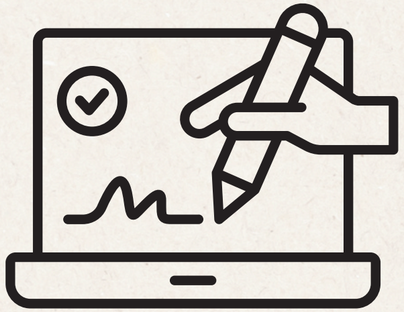
- Authentication Layer
- Cryptographic Layer
- Session Management
- Data Persistence
- Email Service

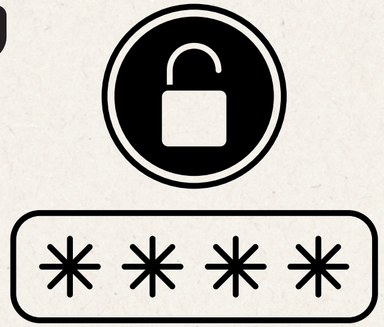
Cryptographic Components

Required Elements:

- ✅ Symmetric encryption (AES-128)
- ✅ Asymmetric encryption (RSA-OAEP)
- ✅ Hash functions (SHA-256)
- ✅ Digital signatures (RSA-PSS)
- ✅ Key exchange protocol (Diffie-Hellman)
- ✅ Password hashing (bcrypt)







Cryptographic Components

Additional requirements met:

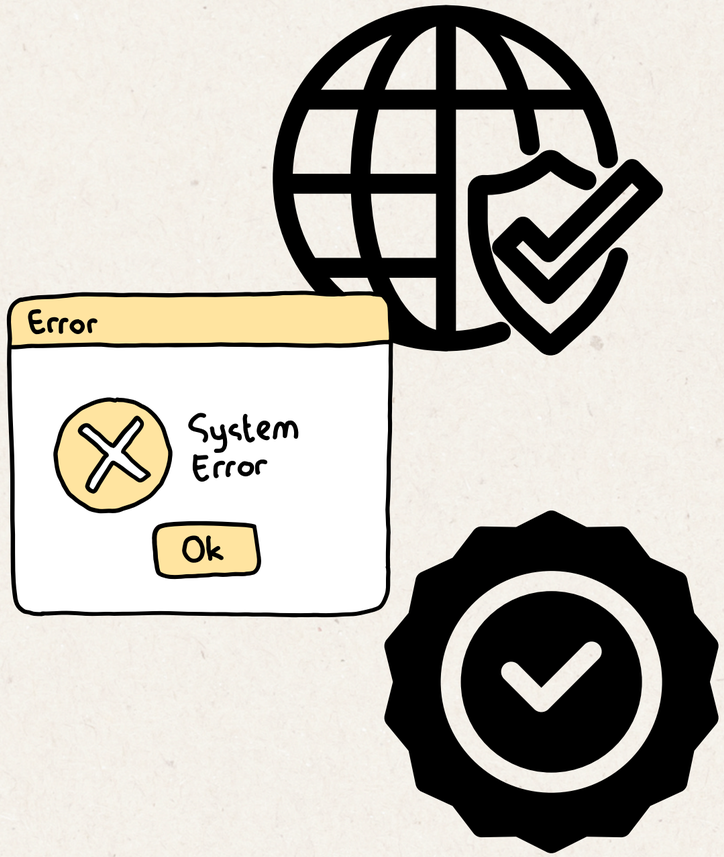
✅ Secure random generation using Python's `secrets` module

✅ Key Management

✅ Error Handling

✅ Input Validation

✅ Mitigations from Threat Model

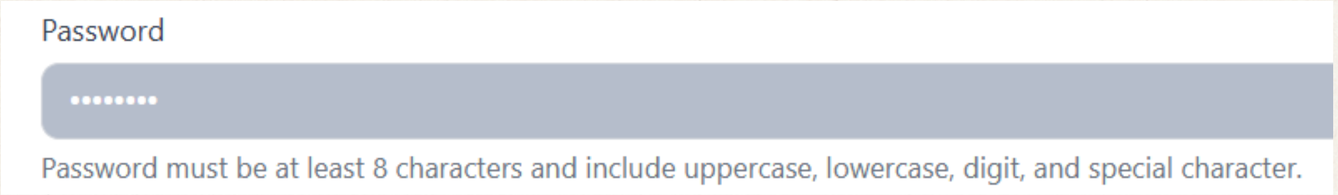


Security Analysis

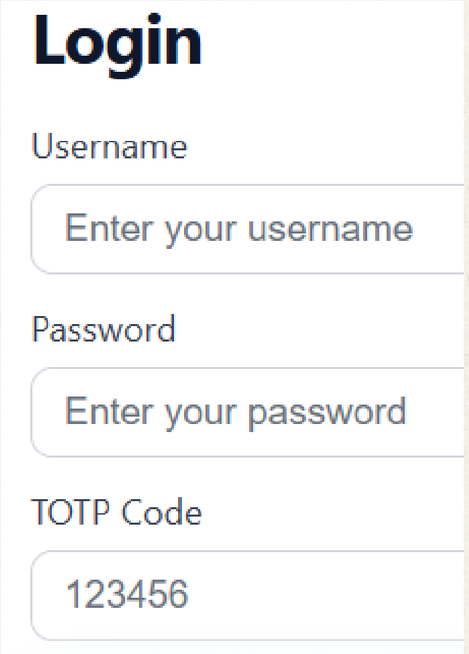
Brute force

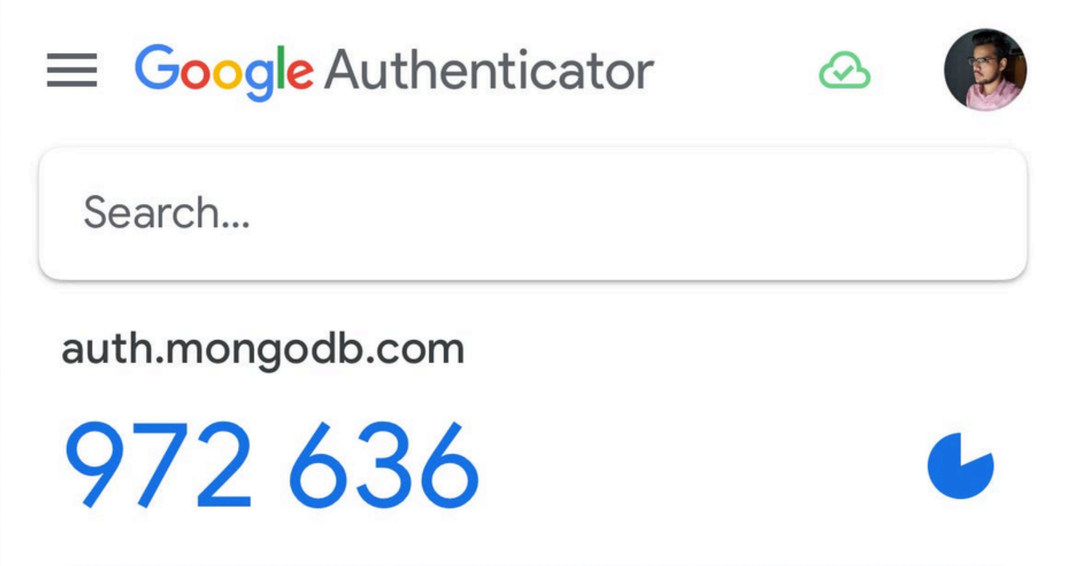
- Bcrypt hashing used. We don’t store passwords

- Strong password requirements (8+ chars,complexity)



- TOTP as second factor





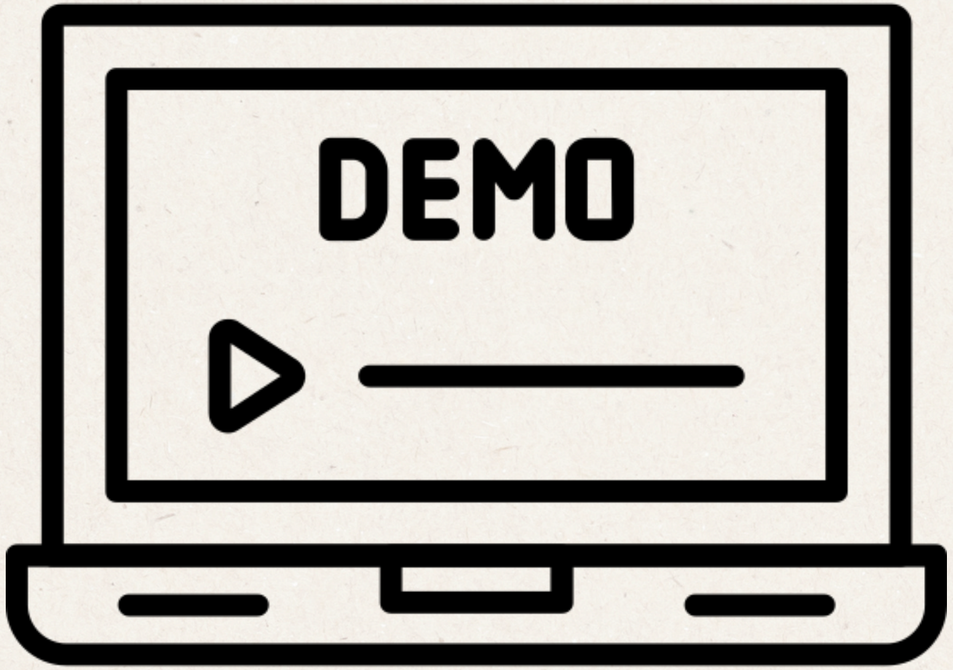
Security Analysis

- CSRF attacks
  Constant-time comparison(secrets.compare_digest)
- JWT validation on each request
- XSS attacks

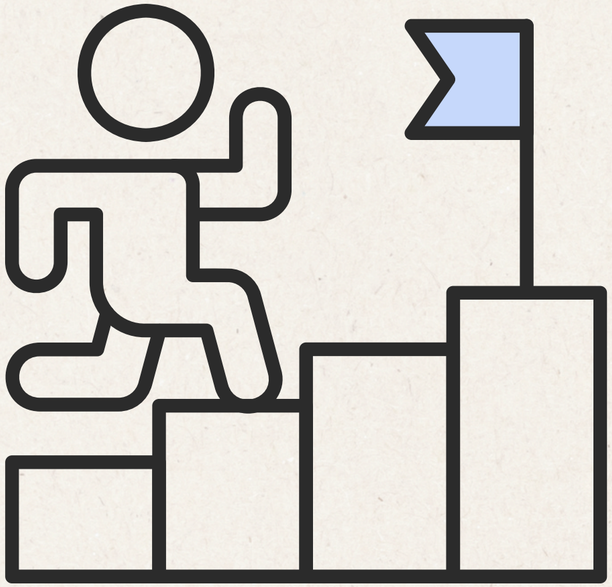
Data encryption

- AES-GCM (symmetric encryption)
- RSA-OAEP (asymmetric encryption)
- RSA-PSS (digital signatures)
- Diffie-Hellman from scratch

Live Demo



Challenges Faced



1)Researching attack types, their mitigation and implementation

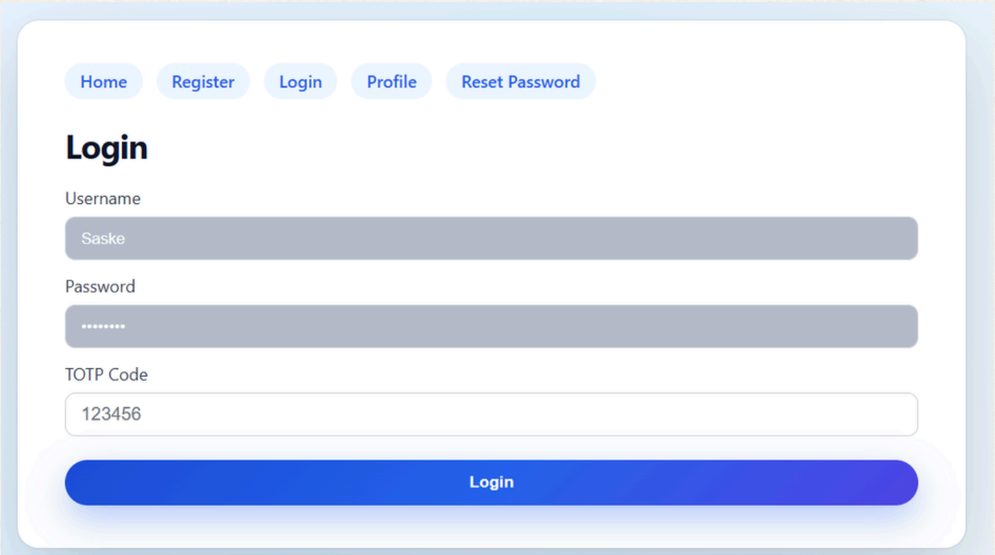
```
# BAD CODE (vulnerable)
username = "<script>alert('XSS!')</script>"
page_content = f"<p>Welcome, {username}!</p>"
# This would execute the script! 💥
```

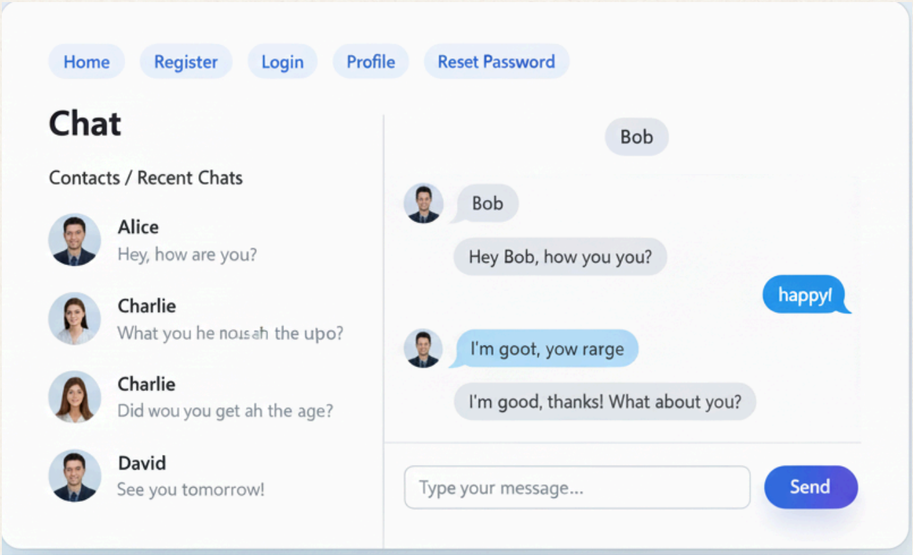
2)Implementing Diffie-Hellman from scratch

3)CSRF protection

- Validating on all POST requests
- Timing attack prevention

Future Improvements





Currently it is just authorization system. In future we could evolve it into chatting app. or other system

Future Improvements

Rate limiting

- Current: Unlimited login attempts
- Needed: 5 attempts per 15 minutes per IP
- Impact: Prevents brute force attacks

Deploy app

- Usage of HTTPS and reliable hosting

Database migration

- Current: JSON file storage
- Needed: PostgreSQL/MySQL with connection pooling
- Impact: Production scalability and concurrency

Account lockout

- Current: No account lockout
- Needed: Lock after N failed attempts, temporary lockout
- Impact: Reduces persistent brute force risk

Thank you!