Abstract geometric lines in the top-left corner of the slide, consisting of several thin white lines forming a complex, overlapping pattern of polygons and triangles.

# **ENHANCING AUTHENTICATION SECURITY: A PYTHON-BASED SYSTEM FOR BRUTE FORCE ATTACK PREVENTION**

**BEATRICE MUTEGI**

**UNIVERSITY OF ESSEX: MASTER OF SCIENCE IN CYBERSECURITY**

**SUPERVISORS: DR. OLIVER BUCKLEY AND DR. BAKHTIYAR AHMED**

# 1. PROBLEM & GAP

Brute force attacks are still a major cybersecurity threat, where attackers guess credentials repeatedly to break into systems (Abdulkader , et al., 2015), (Deep, et al., 2019).

Despite existing tools like CAPTCHA and MFA, many **Python/Django systems** still lack integrated, user-friendly defenses (Jimmy, 2024).

My research addresses this gap by developing a Django-based multi-layered authentication system with a framework that is robust, cost-effective, and user-conscious.

## 2. AIM & RESEARCH QUESTIONS

The goal was to create a secure, usable Python-Django login system that blocks brute force attempts without frustrating users.

It focused on:

- Effective mitigation methods,
- Key vulnerabilities in Django login systems,
- Usability vs. security trade-offs,
- Leveraging Django's native tools for protection.

# 3. LITERATURE REVIEW HIGHLIGHTS



Reviewed authentication methods whereby:

- Traditional methods like password-only/SFA, are vulnerable,
- Modern methods like 2FA, biometrics, and blockchain offer strong security but face usability and complexity challenges.

Key insight:

Single defenses are insufficient and multi-layered security is essential, with careful balance between protection and usability.

# 4. METHODOLOGY

The study used the **Design Science Research approach** (Hevner et al., 2004) with **Agile sprints** across 12 development cycles.

It combined qualitative and quantitative methods, including literature review, **STRIDE** and **OWASP Top 10** threat modelling, simulated attacks, and heuristic evaluations (Creswell, 2017; Nielsen & Molich, 1990).

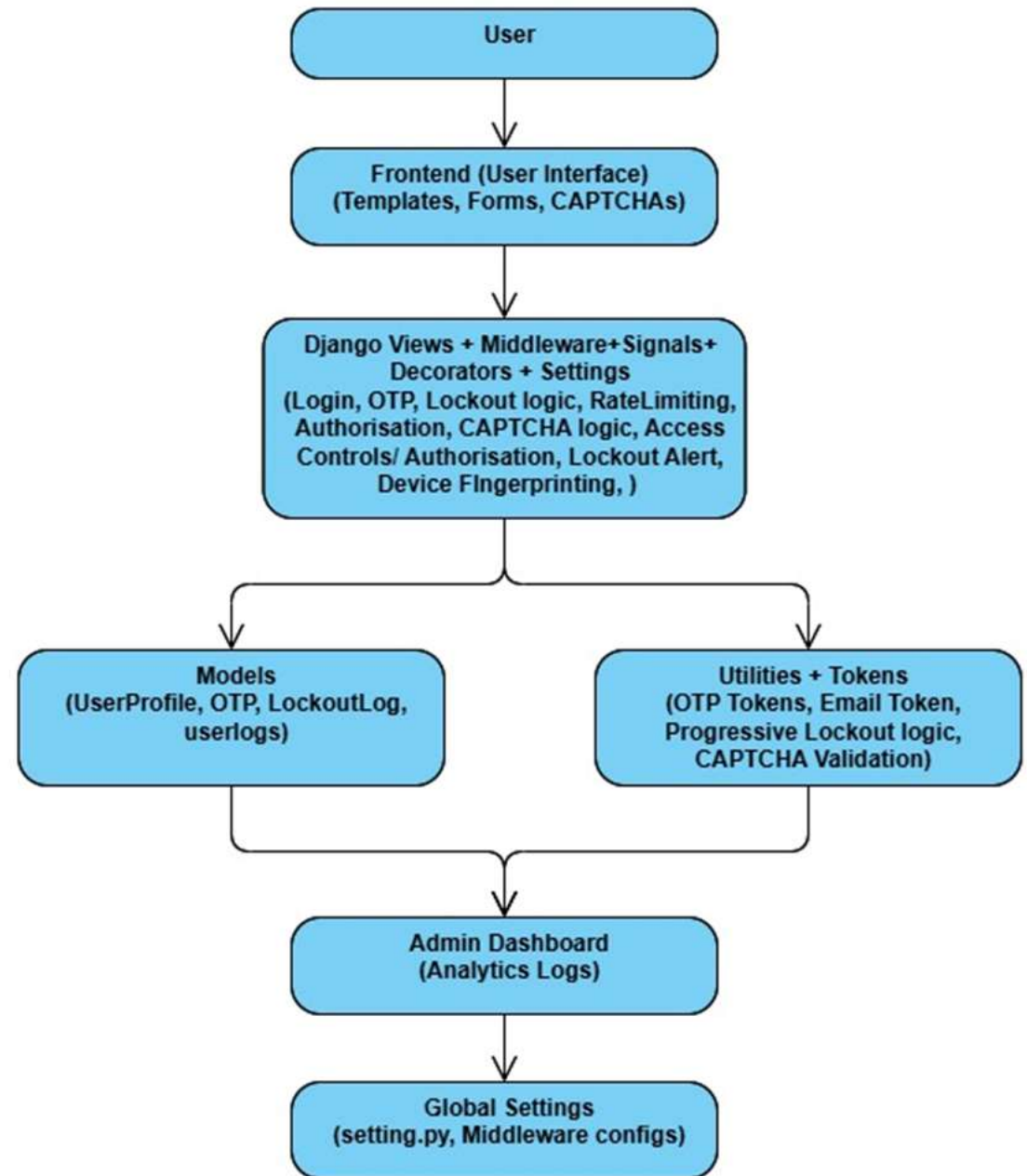
Since no humans were involved, the ethical considerations focused on using synthetic data and ensuring secure, responsible testing in isolated environments.

# 5. SYSTEM ARCHITECTURE AND TOOLS

The system was built with Python (3.10), Django (4.x).

On the right is the interaction it had with its components.

## System Architecture and Component Interaction



The key **tools and libraries** used were(Django Software Foundation, 2023):

- **django.contrib.auth:** Authentication
- **Google reCAPTCHA v2:** CAPTCHA protection
- **django-ratelimit:** Rate-limiting
- **django\_axes / axes.signals.user\_locked\_out:** Account lockouts on repeated login failures
- **Pyotp:** Time-based OTPs via email
- **default\_token\_generator:** Secure email verification tokens

- **django-lockout:** Lockout mechanism
- **django.core.cache:** Store failed login attempts
- **Logging:** Audit login attempts and lockouts
- **http.client:** IP geolocation (ip-api.com)
- **user\_agents:** Parse device/user-agent info
- **Messages:** Frontend user feedback
- And many others.



# 6. SYSTEM FEATURES



The final system integrates the following features.

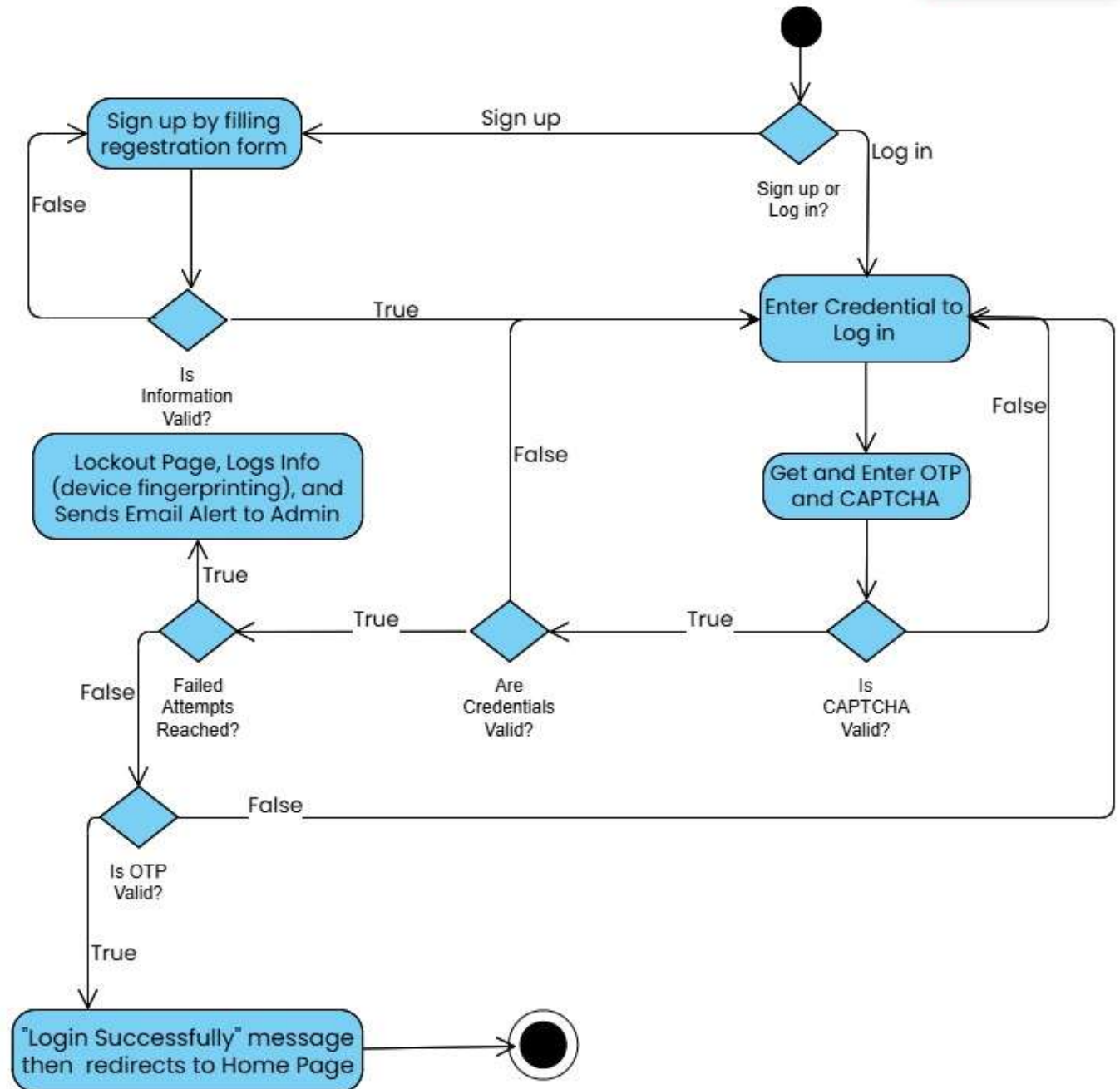
- **Progressive Account Lockouts:** Triggered after 3 to 10 failed logins with increasing lockout delays.
- **IP-Based Rate Limiting:** Mitigates distributed attacks (Nurhaida & Bisht, 2022).
- **Google reCAPTCHA v2:** Triggered after the 3rd failed login to filter out bots (Google, 2025).
- **Time-Based OTP (2FA):** Email OTPs expire in 10 minutes and are required for high-privilege logins (NIST, 2025).
- **Time-based Email Token Verification:** 10-min expiry window to prevent<sub>9</sub> fake or unintended signups.

- **Device Fingerprinting and GeoIP:** Logs OS, browser, and location during lockouts (Yonkeu, 2020).
- **Session Management:** Auto logout after 30 minutes of inactivity and concurrent logins prevention using last activity tracking (Django, n.d.; Fluid Attacks, 2024).
- **Password Management:** Used PBKDF2 hashing, CSRF token and custom validators to enforce secure passwords and prevent reuse (Crudu & Team, 2024), (Django Software Foundation, 2023).
- **Forced Password Expiry:** Users are forced password reset after every 90 days.
- **User Enumeration Prevention:** Error messages generalized to “Invalid credentials” to prevent attackers from distinguishing valid usernames (Macsinouiu, 2024; Agghey et al., 2021).

- **Error Handling & Feedback:** Provides clear user messages and robust logging.
- **Email Alert:** Both Admin and user are notified by email after a logout event.
- **Role-Based Access Control (RBAC) & Authorization:** View access is role-based using decorators (Django, n.d.; Nurhaida & Bisht, 2022; Yonkeu, 2020).
- **A dedicated admin dashboard:** Built with Chart.js, to show failed login heatmaps, logout trends, CAPTCHA stats, and OTP usage analytics.
- **Password Guidelines Modal:** Displays password strength guidelines on forms (Das et al., 2014).

The activity diagram of the login and signup logic is shown on the diagram.

## Activity Diagram: Login View



# 7. TESTS, RESULTS AND EVALUATIONS

Test Type	Result	Evaluation
Brute-force Attack Simulation	Lockout at 5 attempts; 98% blocked	Highly effective
Brute-force Distributed Attack Simulation	IP lockouts triggered	Effective;  Global throttling could enhance
CAPTCHA Logic	Triggered after 3 failures	Blocks bots with minimal UX impact
OTP Expiry Simulation	Rejected expired OTPs	Enforces timely access
Email Token Expiry Simulation	Blocked expired signups	Prevents fake/inactive accounts
Session Expiry Test	Auto logout after 30 mins	Secure session lifecycle
Password Expiry Simulation	Forced reset after 90 days	Enforces strong password hygiene
Concurrent Sessions Simulation	Detection validated	Supports session control enforcement

Additionally, the **admin dashboard** effectively visualizes logout logs, providing actionable security insights.


Login, signup, and password reset workflows function correctly with proper validation and redirection.

Usability testing using **Nielsen's heuristics** showed a simple User Interface with clear feedback (e.g., expired OTPs, invalid credentials) that protects system details; thus, following OWASP guidelines.

Overall, the system blocked over 98% of brute-force attempts while delivering valuable admin insights.

# 8. CHALLENGES AND PROPOSED SOLUTIONS

Challenge / Limitation	Description	Proposed Solution
<b>hCAPTCHA Validation Failures</b>	Integration issues led to replacement	Switched to Google reCAPTCHA v2
<b>CAPTCHA v2 AI Bot Bypass</b>	Potential vulnerability to advanced bots	Upgrade to reCAPTCHA v3 + behavioural analytics
<b>Third-Party Dependency</b>	Reliance on Google services risks downtime	Add fallback local bot detection mechanisms
<b>Limited Usability Testing</b>	No formal accessibility audits conducted	Conduct usability studies & accessibility audits
<b>Dashboard lacks AI intelligence</b>	No automated threat alerts or scoring	Integrate ML-based anomaly detection
<b>Controlled Simulations Limitations</b>	Simulated attacks lack real-world diversity	Use threat intelligence & chaos engineering
<b>Geolocation Accuracy</b>	IP-based geo-location error prone	Use HTML5 geolocation & enhanced device fingerprinting
<b>Scalability Constraints</b>	Only tested locally	Perform load testing, add Redis caching, async tasks
<b>Weak Passwords Creation</b>	No password generator provided	Add client-side password generation tool
<b>User Accessibility Constraints</b>	Not fully inclusive design	Implement audio CAPTCHA, WCAG-compliant UI



Moreover, there were also technical challenges in integrating multiple security features without introducing new vulnerabilities or excessive complexity.

To address emerging threats, the system also needs ongoing maintenance and advanced attack simulations and testing.

Future work could focus on further automation, adaptive security measures, and even deeper integration with biometric or passwordless authentication.



As well as more human involvement in usability testing.



# 9. CONCLUSION AND RECOMMENDATION

This Django-based solution provides **multi-layered authentication** tailored for small to mid-size organizations, greatly improving protection against brute force attacks while preserving usability and cost-effectiveness.

It's **modular, easy to deploy**, and compatible with emerging technologies like biometrics or blockchain.

Furthermore, the admin dashboard offers real-time threat visibility and logs, enabling faster response; an advantage often missing in smaller solutions.

It also highlights the importance of continuous research and adaptation in the face of evolving cyber threats.

My recommendation is to adopt **multi-layered security** that is both usable and scalable; especially for sensitive data and systems.

An abstract graphic on the left side of the slide, consisting of several overlapping, tilted rectangular outlines in white. These lines create a complex, layered geometric pattern that extends from the top-left towards the center of the frame.

# THANK YOU

# REFERENCES

Abdulkader , S., Atia, A. & Mostafa , M.-S., 2015. Authentication systems: principles and threats. *Computer and Information Science*, 8(3).

Agghey , A. Z. et al., 2021. Detection of Username Enumeration Attack on SSH Protocol: Machine Learning Approach. *Symmetry*, 13(11), p. 2192.

Creswell, J. W., 2017. *Research design: Qualitative, quantitative, and mixed methods approaches*. 3rd ed. Lincoln: Sage Publications.

Crudu, V. & Team, M. R., 2024. *Best Practices for Django User Authentication*. [Online]  
Available at: <https://moldstud.com/articles/p-best-practices-for-django-user-authentication>  
[Accessed 30 April 2025].

Das, A., Bonneau, J., Caesar, M. & Borisov, N., 2014. *The Tangled Web of Password Reuse*. [Online]  
Available at: [https://www.researchgate.net/publication/269197028\\_The\\_Tangled\\_Web\\_of\\_Password\\_Reuse](https://www.researchgate.net/publication/269197028_The_Tangled_Web_of_Password_Reuse)  
[Accessed 30 April 2025].

Deep, G. et al., 2019. Authentication Protocol for Cloud Databases Using Blockchain Mechanism. *Sensors*, 19(20), p. 4444.

Django Software Foundation, 2023. *Security in Django*. [Online]  
Available at: <https://docs.djangoproject.com/en/5.2/topics/security/>  
[Accessed 30 April 2025].

Nielsen, J. & Molich, R., 1990. *Heuristic Evaluation of User Interfaces*. Denmark, s.n.

NIST, 2025. *NIST Special Publication 800-63B*. [Online]

Available at: <https://pages.nist.gov/800-63-3/sp800-63b.html>

[Accessed 30 April 2025].

Nurhaida , I. & Bisht , R. K., 2022. *Python for Cyber Security*. [Online]

Available at: [chrome-](#)

<extension://efaidnbmnnnibpcajpcglclefindmkaj/https://cs4all.studentscenter.in/assets/Python%20CS/Python%20for%20Cyber%20Security%20Manual.pdf>

[Accessed 30 April 2025].

OWASP, 2021. *OWASP Top Ten*. [Online]

Available at: <https://owasp.org/www-project-top-ten/>

[Accessed 30 April 2025].

Yonkeu, S., 2020. *Role-Based Access Control in Django*. [Online]

Available at: <https://dev.to/yokwejuste/role-based-access-control-in-django-4j1d>

[Accessed 30 April 2025].

Zhang, X. J., Li, Z. & Deng, H., 2017. Information security behaviors of smartphone users in China: an empirical analysis. *The Electronic Library*, 35(6), pp. 1177-1190.

Django Software Foundation, 2025. *Using Sessions*. [Online]

Available at: <https://docs.djangoproject.com/fr/5.2/topics/http/sessions/>

[Accessed 01 May 2025].

Fluid attacks: help center, 2024. *Concurrent sessions - Python*. [Online]

Available at: <https://help.fluidattacks.com/portal/en/kb/articles/criteria-fixes-python-062>

[Accessed 01 April 2025].

Google, 2025. *recaptcha how it works*. [Online]

Available at: <https://cloud.google.com/security/products/recaptcha#how-it-works>

[Accessed 30 April 2025].

Hevner, A. R., March, S. . T., Ram, S. & Park, J., 2004. Design Science in Information Systems Research. *MIS Quarterly*, 28(1), pp. 75-105.

Jimmy, F., 2024. Cybersecurity Threats and Vulnerabilities in Online Banking Systems. *International Journal of Scientific Research and Management (IJSRM)*, 12(10), pp. 1631-1646.

Macsinoiu, V. E., 2024. Unveiling User Enumeration Attacks: Methods, Impacts and Mitigation Strategies. *International Journal of Information Security and Cybercrime (IJISC)*, 26(2), pp. 59-64.

National Cyber Security Centre, 2018. *GDPR security outcomes*. [Online]

Available at: <https://www.ncsc.gov.uk/guidance/gdpr-security-outcomes>

[Accessed 20 July 2022].