TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

AN OVERVIEW OF OUR PROJECT

Security Project Overview: Strengthening Our Future

Done By :

Farah Toumi

Molka Braham

Thouraya Harrabi

Introduction

Why FortiPass?

- Passwords are the first line of defense—but often weak.
- FortiPass is a web-based password strength checker.
- Built to help users create safer passwords.
- Uses OWASP standards, real-time feedback, and breach checking.
- Educates users, not just validates.

Main Concepts

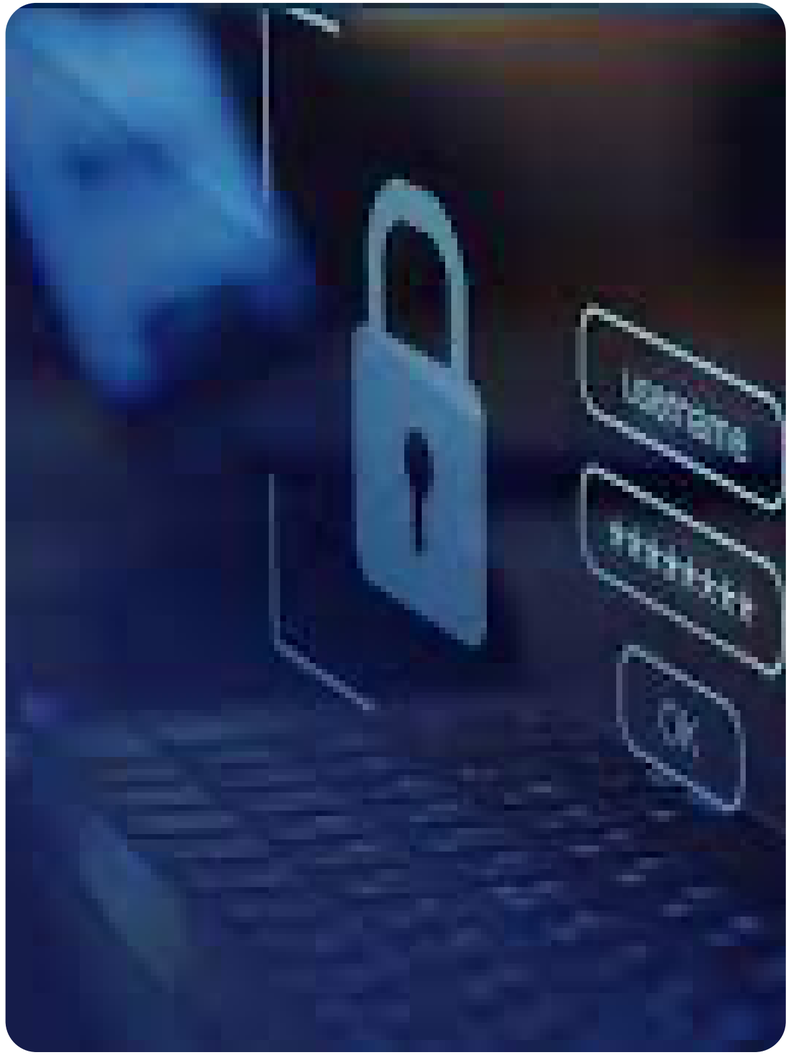
Core Security Concepts Behind FortiPass :

- OWASP Password Policy Requirements: Enforces rules on length,
- character variety, common passwords, personal info.
- Password Entropy: Measures unpredictability
- Levenshtein Distance: Prevents passwords similar to
- usernames/emails.
- zxcvbn Algorithm: ML-based estimator using real-world data.
- Breach Intelligence: Uses “Have I Been Pwned” API with k-anonymity.
- Passphrase Encouragement: Promotes strong, memorable phrases

Theoretical Background

- FortiPass follows OWASP’s Password Policy Project.
- Focus on length over complexity (e.g., 12+ characters).
- Avoids outdated rules like forced symbols or expirations.
- Checks for breaches using secure queries.
- Prioritizes usability + security.

Functional Flow:



How FortiPass Works:

1. User enters password.
2. Validation Engine checks length, complexity, and entropy.
3. Password hashed → partial hash sent to breach API.
4. API responds if password has been exposed.
5. Aggregated result + real-time feedback shown to user.
6. Suggestions provided if needed

Unique Value Proposition

What Makes FortiPass Stand Out?

- Goes beyond basic checks (symbols, digits).
- Detects use of personal info.
- Real-time breach alerts.
- Privacy-preserving with k-anonymity.
- Encourages passphrases (easy + secure).
- Clear, adaptive feedback.
- Developer-friendly and scalable for teams or enterprises.

System Design Overview :

- Web-based application
- Uses OWASP’s Password Storage Cheat Sheet as core logic
- Gives actionable feedback to improve passwords
- Goal: Improve digital hygiene through education

Architecture Diagram

System Components:

- Frontend: HTML/CSS/JavaScript or React
- Backend: Python (Flask)
- OWASP Rules: Integrated into backend logic
- Data Flow:
- User → Frontend → Backend → Password evaluated → Score +
- suggestions → Shown in real-time

Roles & Users

User:

- Enters password
- Receives score and suggestions

Admin:

- Updates policies and rules
- Manages evaluation settings

Core Functionalities & Workflows

1. **Password Strength Check:**
Evaluates based on OWASP rules
2. **Real-Time Feedback:**
JS listener updates UI live
3. **Suggestions Engine:**
Recommends specific fixes (e.g.,
“Add a symbol”)
4. **OWASP Compliance:**
No forced symbol rules, promotes memorability

Tools Used :

Development Tools & Tech Stack:

- HTML/CSS/JavaScript or React – Frontend
- Python / Flask – Backend
- OWASP Guidelines – Rules logic
- GitHub – Version control
- Canva – Diagram creation
- VS Code – Code editor

Development Phases

- **Phase 1:** Requirement Analysis

Studied OWASP password standards

- **Phase 2**: Frontend Development

Input field, meter bar, live JS updates

- **Phase 3:** Backend Development

API route /evaluate, applies rules

- **Phase 4**: OWASP Integration

Enforces proper password rules

- **Phase 5**: Testing

Used Postman to test strong vs weak passwords

- **Phase 6**: Final Touches

Visual strength bar, tooltips, deploy on GitHub Pages/Vercel

Conclusion

FortiPass: Empowering Secure Passwords

- OWASP-compliant, real-time password evaluator
- Combines theory (entropy, ML) with real-world data (breaches)
- Educates users and encourages safe password habits
- Simple yet powerful solution to a widespread problem

Thank you