IT 360 : INFORMATION ASSURANCE & SECURITY

TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

FortiPass (2nd Part)

Intelligent Password Strength & Security Tester

FortiPass helps users create safer passwords using OWASP rules, breach checks, and real-time feedback.

Presented By :

Farah Toumi

Thouraya Harrabi

Molka Braham

Presented To :

DR. Manel Abdelkader

Table of Contents

System Design Overview

1.Goal

FortiPass is a web-based application that checks the strength of passwords using rules from the OWASP Password Storage Cheat Sheet. It gives real-time feedback and actionable suggestions to help users improve their passwords

Architecture Diagram

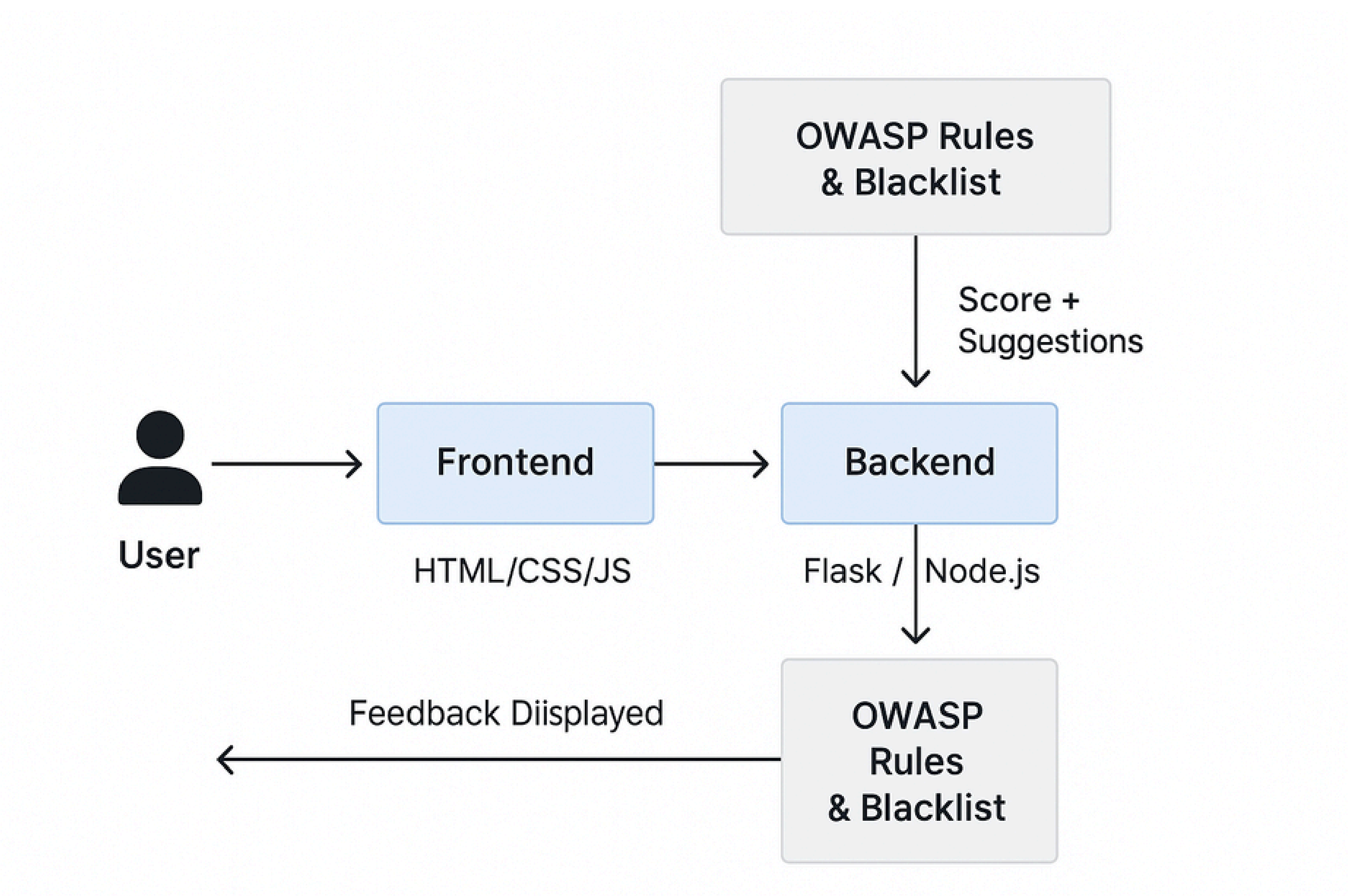
Components:

1. **Frontend** (User Interface) : built with HTML/CSS/JavaScript or React.
2. **Backend** (Password Evaluation Engine): built with Node.js or Python (Flask).
3. **Database** (Optional):to log results or keep password policies (use SQLite or Firebase).
4. **OWASP Rules Library** : integrated as logic into backend for password scoring.

Architecture Diagram

Data Flow:

- User types password → Sent to backend → Evaluated using OWASP rules → Returns score + suggestions → Shown to user in real-time.



Roles-Users of the System

Role	Description
User	Anyone using the tool to check a password
Admin	Maintains rules or updates policies

Core Functionalities & Workflows

Function	Steps
1. Password Strength Check	User enters password → Sent to backend → Score calculated → Feedback shown
2. Real-Time Feedback	JS event listener detects input → Sends it → Updates UI live
3. Suggestions Engine	Based on score, backend returns suggestions (e.g., “Add symbol”)
4. Compliance with OWASP	Backend rules follow OWASP Password Policy recommendations

Tools used

Tool/Tech	Purpose
HTML/CSS/JS or React	Frontend User Interface
Node.js	Backend logic (password evaluation)
Express	API routes
OWASP guidelines	Rules for strength evaluation
GitHub	Version control
Postman or Curl	Testing API calls
canva	for diagrams
VS Code	Code editor

Development Phases

Phase 1: Requirement Analysis

- Goal: Understand OWASP rules.
- Action: Researched OWASP Password Storage Cheat Sheet.
- Outcome: Identified evaluation criteria (length, variety, blacklist)

Phase 2: Frontend Development

- Simple UI: Password input field, strength meter bar.
- JS to capture keystrokes in real time.
- Display feedback from backend.

Phase 3: Backend Development

- API with /evaluate route.
- Takes password as input.
- Checks:
  - Length
  - Uppercase, lowercase, numbers, symbols
  - Blacklist common passwords

Phase 4: OWASP Integration

- Write functions that follow OWASP rules:
  - No composition rules
  - Password length is the most important
  - Avoid hints or password expiration

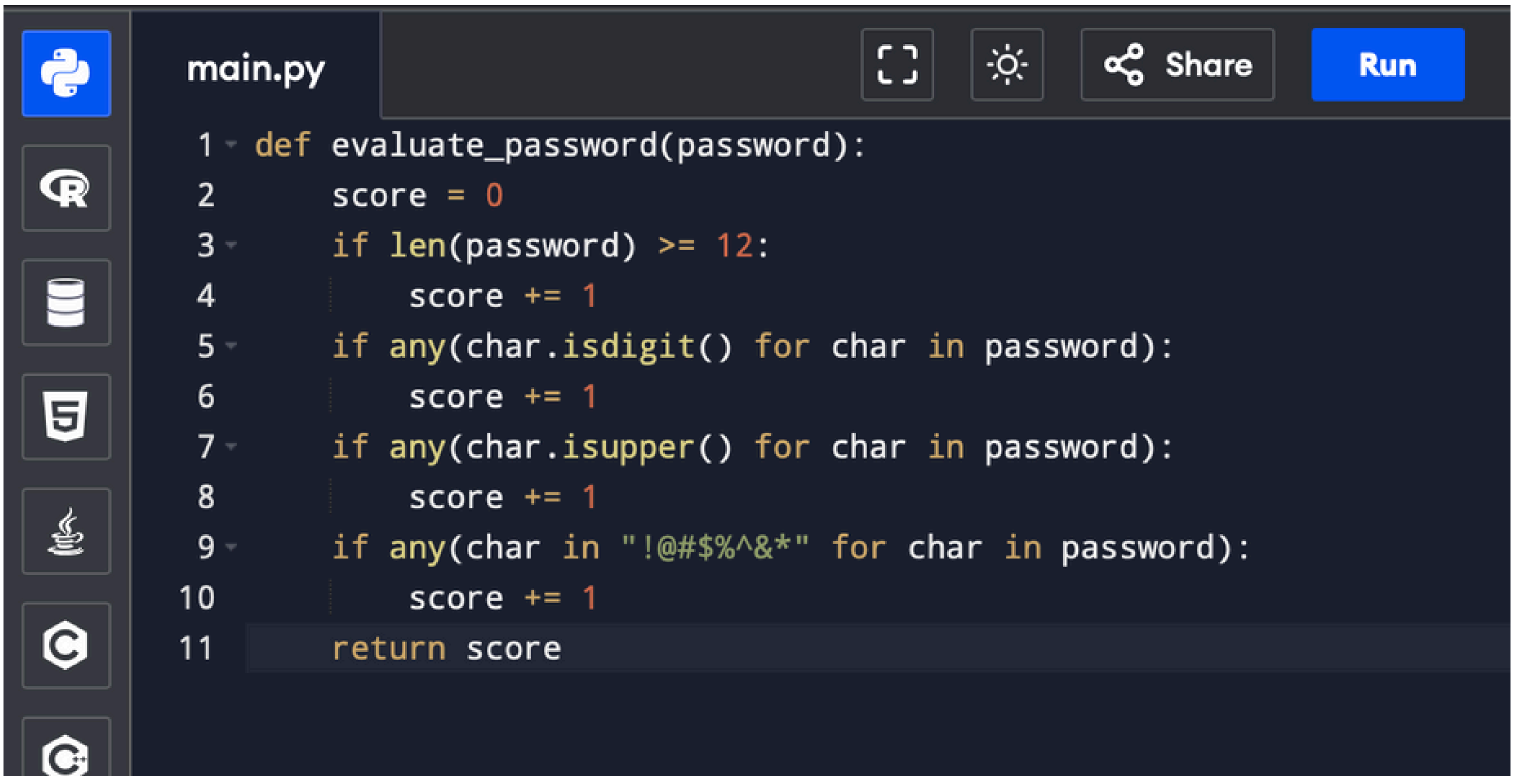
Phase 5: Testing

- Use Postman to test API endpoints
- Try good vs bad passwords and check results

Phase 6: Final Touches

- Add visual bar to show strength
- Add tooltip with suggestions
- Deploy on GitHub Pages or Vercel

Example: Backend password evaluation logic



```python
def evaluate_password(password):
    score = 0
    if len(password) >= 12:
        score += 1
    if any(char.isdigit() for char in password):
        score += 1
    if any(char.isupper() for char in password):
        score += 1
    if any(char in "!@#$%^&*" for char in password):
        score += 1
    return score
```

Conclusion

FortiPass offers a simple yet powerful solution to one of the most common security problems: weak passwords. By following OWASP's best practices and offering real time feedback, it empowers users to adopt better security habits.