# Password Strength Analyzer with Custom Wordlist Generator

## Internship Project Report

### 1. Introduction

Weak passwords continue to be a major vulnerability in personal and organizational cybersecurity. Despite increased awareness, users often select predictable passwords that can be cracked using brute-force or dictionary attacks. To address this issue, we developed a command-line based tool that performs two critical tasks: it evaluates the strength of user-provided passwords and generates custom wordlists based on personal information for penetration testing or audit simulations.  
  
This tool aims to raise awareness about password security while offering a hands-on demonstration of how attackers may exploit predictable patterns using customized dictionaries.

### 2. Abstract

The project combines two components: a password strength analyzer and a wordlist generator. The password analyzer uses Dropbox’s zxcvbn library to provide a real-time strength score and feedback on a given password, including crack-time estimations. The wordlist generator accepts personal inputs such as name, date of birth, and pet name, and produces a comprehensive list of variants using common password patterns (e.g., leetspeak, suffixes like "123", "2025", etc.).  
  
This project is built entirely using Python, keeping it lightweight, portable, and cross-platform compatible. The generated wordlist can be used for cybersecurity training, red teaming, or research on password vulnerabilities.

### 3. Tools Used

- Programming Language: Python 3.11  
- Libraries:  
 - zxcvbn (for password strength analysis)  
 - argparse (for CLI argument handling)  
 - nltk (optional, for advanced linguistic processing – not used in final version)  
- Environment: Visual Studio Code (VS Code)  
- Platform: Windows 10

### 4. Steps Involved in Building the Project

- Step 1: Password Analyzer  
 - Integrated zxcvbn library to evaluate password strength.  
 - Used argparse to accept passwords via the command line.  
 - Displayed score (0 to 4), estimated crack times, and suggestions for improvement.  
  
- Step 2: Wordlist Generator  
 - Accepted inputs: name, year of birth, and pet name.  
 - Generated common variants: leetspeak substitutions, casing patterns, and numeric suffixes.  
 - Exported results into a .txt wordlist file (sample\_wordlist.txt), formatted for compatibility with password auditing tools.  
  
- Step 3: Output & Testing  
 - Verified analyzer feedback with weak and strong password samples.  
 - Confirmed wordlist file creation and reviewed output for logical coverage.

### 5. Conclusion

The project successfully simulates a real-world approach to both evaluating and attacking password security. It demonstrates how weak passwords can be identified and how attackers might exploit predictable inputs using generated wordlists.  
  
While basic in implementation, the project reflects practical knowledge in cybersecurity fundamentals, password entropy, and social engineering-based attacks. The tool also serves as a learning resource to reinforce best practices in password creation and defense mechanisms.