Software Requirements Specification (SRS)

Project Title: PDF Decryption Tool

Developer: Gonzalo Patino

Objective

Develop a Python application using Tkinter that allows users to decrypt password-protected PDF files by inputting the password once. The output will be a copy of the original PDF without the encryption.

1. Introduction

Purpose

This document outlines the software requirements for a PDF Decryption Tool, which is a desktop application built with Python's Tkinter GUI. The program will allow users to upload a password-protected PDF file, input the password, and save the decrypted version of the PDF without the password.

Scope

The tool will accept password-protected PDFs, request the password to unlock the file, and generate an unencrypted PDF as output. This tool will be intended for personal use to manage password-protected PDF documents.

2. System Requirements

2.1 Hardware Requirements:

- CPU: 1.0 GHz or higher
- RAM: 512 MB or higher
- Hard Disk: Minimum 100 MB free space
- Platform: Cross-platform (Windows, macOS, Linux)

2.2 Software Requirements:

- Python 3.x
- Tkinter (for GUI)
- PyPDF2 (for PDF decryption and manipulation)

• ReportLab (optional, for additional PDF operations)
• Operating system: Windows, macOS, Linux

3. Functional Requirements

3.1 User Interface

Main Window:
The application will have a simple Tkinter-based graphical user interface (GUI). The window should contain the following components:
• A title label "PDF Decryption Tool"
• A button labeled "Select PDF" that opens a file explorer for the user to select a PDF file.
• A text box for users to input the password required to unlock the PDF.
• A button labeled "Decrypt PDF" that triggers the decryption process.
• A status label (initially empty) that shows messages like "PDF successfully decrypted" or "Invalid password".
• A save file dialog to specify the output location and file name for the decrypted PDF.

3.2 PDF Upload

The user must be able to upload a password-protected PDF file via a file explorer dialog that opens when the "Select PDF" button is clicked. Only PDF files (.pdf) should be allowed.

3.3 Password Input

The application should provide an input field for the user to enter the password for the selected PDF. This should be a masked input (e.g., * or • characters should appear instead of the password text).

3.4 Decryption Process

Upon clicking the "Decrypt PDF" button, the application should attempt to unlock the selected PDF using the provided password. If the password is correct, the application should remove the encryption and save the file as an unencrypted PDF.

Steps:
1. Load the selected PDF.
2. Attempt to unlock it using the input password.
3. If successful, remove the encryption and save a new unencrypted copy.
4. If unsuccessful, display an error message (e.g., "Incorrect password").

3.5 File Save Location

After decryption, the user should be able to select the destination and file name for the unencrypted PDF using a save file dialog. The output PDF should be saved with the same structure and content but without the password.

3.6 Error Handling

The application must handle the following error scenarios:
• No PDF selected: Display a message if the user tries to decrypt without selecting a file.
• Invalid password: If the password is incorrect, show an error message and allow the user to try again.
• PDF not encrypted: Display an error if the selected PDF is not encrypted.
• PDF decryption failure: If the PDF cannot be decrypted for any other reason (e.g., corrupted file), display a generic error message.

4. Non-Functional Requirements

4.1 Usability

The user interface must be intuitive and require minimal effort from the user to decrypt a PDF.

4.2 Performance

The decryption process should be efficient, completing in a reasonable amount of time based on the size of the PDF file.

4.3 Security

The password entered by the user must not be stored anywhere in the system.
The application should not keep any temporary files of decrypted PDFs once the decryption is complete and the file is saved.

4.4 Portability

The application should be cross-platform and run smoothly on Windows, macOS, and Linux.

5. Dependencies

• Tkinter: Used for building the graphical user interface.
• PyPDF2: Used for handling the PDF decryption and manipulation process.
• filedialog: To implement file selection and saving.
• os: To handle system-related tasks such as file management.

6. Future Enhancements (Optional)

• Batch Processing: Support for selecting multiple PDF files and decrypting them at once.
• PDF Metadata Editing: The option to modify the metadata of the decrypted PDF file (e.g., title, author).
• Password Management: Save frequently used passwords for easier decryption in future sessions (securely stored).
• PDF Preview: Display a preview of the first page of the PDF after decryption.

7. Assumptions

• Users have the correct password to unlock the PDF.
• The input files are valid, non-corrupted PDFs.