Stage 1 Report

CS 4366: Senior Capstone Project

The Clever Coders

Robert Wells – Team Leader

Sang Nguyen – Documentation Specialist

Rocco Swaney – Backend Developer

Gage Flores – Frontend Developer

Logan LiVigni – Website Developer

Group Website

Report

Project Description:

The project we decided to tackle this semester is personally identifiable information (PII) redaction from text documents. IBM says, “Personally identifiable information (PII) is any information connected to a specific individual that can be used to uncover or steal that individual’s identity.” These include information such as social security numbers, full name, email address or phone number. The security benefits of redacting PII include preventing identity theft and fraud, ensuring regulatory compliance, mitigates insider threats, protects against data leaks, enhances document security in cloud storage, and boosts consumer and client trust. Our objective during this project is to develop a program that will automatically detect and redact PII to protect an individual’s privacy.

Tasks/Objectives:

Our goal was to identify the problems and intended solutions.

- <u>Interviews:</u>
    - First, we must analyze potential target users and identify any function or performance requirement for our system by interviewing at least four or five users who are not taking CS 4366. The interview will consist of several questions, and we will log answers and create a summary of them.
- <u>Create an SRS:</u>
    - After interviewing potential target users, we will create an SRS (Software Requirements Specification) that will specify and describe the requirements of the software that need to be fulfilled for successful deployment of the software system.
- <u>Creating reports/slides:</u>
    - To convey and show our proposed project, we are tasked with creating a report consisting of project description, tasks with explanations, interview logs with summary, and an SRS (Software Requirements Specification).
    - Next, we must prepare presentation slides based on our report and present them to the class and instructor.

Interview log:

We conducted interviews to analyze potential users and identify any function or performance requirements for our system. We asked the interviewees the following questions:

1. What is your level of experience with computers in general?

2. What is your background? You can put your current/future job, student, etc.?
3. Does your job, or future job, require redaction of information?
4. Do you ever redact information from any forms of documents or files?
5. What is your level of trust in automation in general?
6. What is your level of trust in automation for redacting information?
7. What type of documents or text format do you usually, or would like to, redact information from?
8. What type of file format do you most often, or would likely to, redact from?

- <u>**Interviewee #1**</u>
  1. **What is your level of experience with computers in general?**
     - 4/10
  2. **What is your background? You can put your current/future job, student, etc.?**
     - Doctor of Occupational Therapy student
  3. **Does your job, or future job, require redaction of information?**
     - Yes
  4. **Do you ever redact information from any forms of documents or files?**
     - Yes
  5. **What is your level of trust in automation in general?**
     - 8/10
  6. **What is your level of trust in automation for redacting information?**
     - 7/10
  7. **What type of documents or text format do you usually, or would like to, redact information from?**
     - Tax records, medical records, legal documents, financial statements, academic records
  8. **What type of file format do you most often, or would likely to, redact from?**
     - PDF files
- <u>**Interviewee #2**</u>
  1. **What is your level of experience with computers in general?**
     - 6/10
  2. **What is your background? You can put your current/future job, student, etc.?**
     - Graduate student
  3. **Does your job, or future job, require redaction of information?**
     - No
  4. **Do you ever redact information from any forms of documents or files?**
     - No
  5. **What is your level of trust in automation in general?**
     - 4/10
  6. **What is your level of trust in automation for redacting information?**
     - 4/10

7. **What type of documents or text format do you usually, or would like to, redact information from?**
    - None
8. **What type of file format do you most often, or would likely to, redact from?**
    - None

- <u>**Interviewee #3**</u>
    1. **What is your level of experience with computers in general?**
        - 8/10
    2. **What is your background? You can put your current/future job, student, etc.?**
        - Nursing student
    3. **Does your job, or future job, require redaction of information?**
        - Yes
    4. **Do you ever redact information from any forms of documents or files?**
        - Yes
    5. **What is your level of trust in automation in general?**
        - 7/10
    6. **What is your level of trust in automation for redacting information?**
        - 8/10
    7. **What type of documents or text format do you usually, or would like to, redact information from?**
        - Medical records
    8. **What type of file format do you most often, or would likely to, redact from?**
        - PDF files
- <u>**Interviewee #4**</u>
    1. **What is your level of experience with computers in general?**
        - 8/10
    2. **What is your background? You can put your current/future job, student, etc.?**
        - Attorney
    3. **Does your job, or future job, require redaction of information?**
        - Yes
    4. **Do you ever redact information from any forms of documents or files?**
        - Yes
    5. **What is your level of trust in automation in general?**
        - 5/10
    6. **What is your level of trust in automation for redacting information?**
        - 3/10
    7. **What type of documents or text format do you usually, or would like to, redact information from?**
        - Tax records, medical records, legal documents, financial statements, emails
    8. **What type of file format do you most often, or would likely to, redact from?**
        - PDF files

- <u>**Interviewee #5**</u>
  1. **What is your level of experience with computers in general?**
     - 10/10
  2. **What is your background? You can put your current/future job, student, etc.?**
     - Deputy CISO
  3. **Does your job, or future job, require redaction of information?**
     - Yes
  4. **Do you ever redact information from any forms of documents or files?**
     - Yes
  5. **What is your level of trust in automation in general?**
     - 9/10
  6. **What is your level of trust in automation for redacting information?**
     - 9/10
  7. **What type of documents or text format do you usually, or would like to, redact information from?**
     - Medical records, legal documents, and employment records
  8. **What type of file format do you most often, or would likely to, redact from?**
     - Word documents, PDF files, Images

Interview summary:

We conducted five interviews to gather information about potential users and to identify functional or performance requirements for our system. Two interviewees have a moderate level of computer experience (4-6/10) and three have a high level (8-10/10). Two interviewees going into the medical field will redact PII, two interviewees currently redact PII, and one will not redact PII in their future job. Four of our interviewees said they have already redacted PII, or would like to, from tax records, medical records, legal documents, financial statements, academic records, employment records, and emails. The most common file format that interviewees redacted PII from, or would like to redact PII from, PDF, with one saying they also redact from word documents and images. However, our project will focus on text files, which users can easily convert to a PDF if needed.

Three interviewees reported a high level of trust in automation (7/10, 8/10, 9/10), while the other two have a moderate/low level of trust (4/10, 3/10). In comparison to automation in general, one interviewee reported a greater level of trust in automation for redaction (8/10), two the same level (4/10, 9/10), and the other two with a lower level of trust (7/10, 3/10).

Since we had interviewees going into the medical field, we decided to focus on redaction methods that ensure HIPAA compliance.

SRS

1. Introduction

1.1. Purpose

The purpose of this document is to present a detailed description of the personally identifiable information redactor. It will explain the purpose, features, and constraints. This document is intended for the development team, users, and professor.

1.2. Scope of Project

This program will be called PII redactor. The PII redactor will find all personally identifiable information and then redact it from the document. The benefits of the PII redactor include preventing identity theft and fraud, ensuring regulatory compliance, mitigates insider threats, protects against data leaks, enhances document security in cloud storage, and boosts consumer and client trust.

1.3. Definitions, acronyms, and abbreviations

- Masking – Replacing a word or phrase with random or single characters to censor the word or phrase.
- Personally Identifiable Information - IBM says, “Personally identifiable information (PII) is any information connected to a specific individual that can be used to uncover or steal that individual’s identity.”
- PII – Personally Identifiable Information

- Program – Refers to the software this software requirements specification was written for. It is the software that identifies and redacts PII.
- Redact – Edit pieces of text to remove sensitive information.
- Redactor – something that redacts.
- Regex – Regular expression
- Regular expression – Patterns used to match character combinations in texts
- SRS – Software Requirements Specification
- System – Used synonymously with “program” to refer to the software that identifies and redacts PII.

1.4. References

[1] Ibm, “What is personally identifiable information (PII)?,” IBM, https://www.ibm.com/think/topics/pii (accessed Sep. 11, 2025).
[2] “PivotAi Ainon Tool,” Ainon, https://ainon.ai/blog/what-is-pii-and-why-do-i-need-to-protect-it/ (accessed Sep. 11, 2025).
[3] IEEE, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830-1998, 1998. [Online]. Available: https://ieeexplore.ieee.org/document/720574. [Accessed: Sep. 23, 2025]
[4] MozDevNet, “Regular expressions,” MDN Blog RSS, https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_expressions (accessed Sep. 25, 2025).

1.5. Overview

The next section of this document is the overall description, which gives an overview of the functionality of the program. It will tell us about the program, its functions, user characteristics, constraints, assumptions and dependencies, and dividing requirements.

The third section will tell us the specific requirements of the program. It should have all software requirements described in detail so that a development team can design and implement those requirements. In this section, every stated requirement should include at minimum a description of every input, every output, and all functions performed by the program in response to input.

2. Overall Description

2.1. Product Perspective

PII Redactor is a standalone system that detects and removes personal information from text documents. The system will perform processes locally on the user’s computer independently of other users within the same organization, regardless of if the end directory is shared.

2.1.1 System interfaces

File System: The program will read input (TXT) documents from the user’s computer, process them to find and redact PII, and then write on a new file that will be saved to the user’s computer.

NLP and Regular Expressions (Regex): The program uses NLP techniques and regex patterns to identify PII including names, emails, phone numbers, and addresses, for redaction.

2.1.2. User interfaces

Our system will use a GUI (if not console run) which will prompt the user to specify the input and output files, where our system will process the input file, redact PII, and write the newly redacted text to the output file.

2.1.3. Hardware interfaces

This section is not applicable because the system is a standalone software application. It runs on a laptop or standard desktop and does not require any specialized hardware. It only requires basic computer components (CPU, memory, disk space).

2.1.4. Software interfaces

The PII Redactor will have a graphical interface (if not console run), that will require the user to specify the input and output files. The only software necessary is an operating system (Windows, macOS, or Linux) to run the system.

2.1.5 Communications interfaces

This section is not applicable because the system is a standalone system that does not communicate over a network but runs on a user’s local computer.

2.1.6. Memory constraints

The PII Redactor has no memory requirements beyond the normal resources in a standard computer. The system uses RAM or primary memory for processing the files and disk storage or secondary memory for the reading of the input file and storage of the output file.

2.1.7. Operations

a) The various modes of operations in the user organization:

The user will launch the program (via GUI or console) and provide the input/output files. The program then processes the input document, finding and redacting PII, then writes that text to the output file. Success or error messages are displayed in the console.

b) Periods of interactive operations and periods of unattended operations:

For interactive operations, the user will provide the input/output files in the GUI or console. Unattended operations are not applicable in this system because the program cannot run on its own but must be started by the user.

c) Data processing support functions:

The program processes an input file (TXT file), detects and redacts PII, then writes the redacted file to the output file. Error handling is provided for missing/unreadable text files.

d) Backup and recovery operations:

The program does not change the input file, instead it processes it and writes to an output file. So, if the user needs the original, unredacted file back for any reason, they can obtain it.

2.1.8. Site adaptation requirements

a) Data requirements:

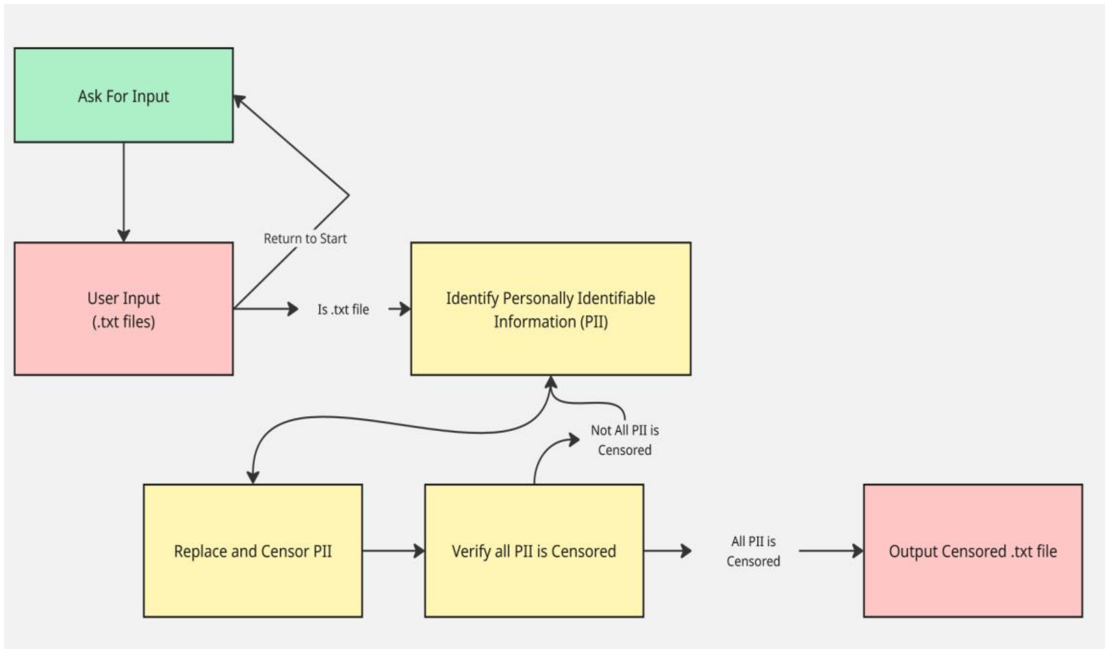
The program requires that the data must be in the form of a text file (txt). No additional requirements are needed for the system.

b) Adaptation requirements:

The program is easily distributed to users based on any OS (Windows, macOS, Linux) by visiting the URL.

2.2. Product Functions

1. File Input: Take user input in the form of .txt files
2. PII Detection: Detect personally identifiable information (PII)
3. PII Redaction: Censor the personally identifiable information
4. File Output: Return the censored version of the .txt file to the user as output
5. Accuracy: Perform the task with 100% accuracy



2.3. User Classes and Characteristics

User Classes will be those in professional positions who need to redact and censor PII on various documentation. This could be, but is not limited to, Health Care professionals, Government personnel, Finance professionals and Executive personnel.

The necessary education and experience would be limited, as the user would need to understand how to run a pre-installed executable. Simple documentation of the necessary steps would be sufficient to correctly run the program. Technical experience would be the basic understanding(s) of their computer.

2.4. Constraints

The PII Redactor must operate on a PC running either Windows, Linux, or macOS operating system. PII Redactor must redact information in a way that is compliant with PHI guidelines set by HIPAA and US Privacy Act of 1974 definitions of PII. The PII Redactor must be able to complete the redaction and output the censored file in under 1 minute.

2.5. Assumptions and dependencies

Assumptions

- Having access to 1 of 3 different operating systems, those being Windows, MacOS or Linux.
- Use of a .txt file as input for the executable, that is readable and not corrupted.

- Either use of pre-defined PII(s) or substitution of a custom PII list, to identify the information to be censored.
- Censoring, via masking
- Verification that all PII is censored, and that upon failing verification the censoring pass is repeated.
- Output is a .txt file with all PII censored, with the file being retitled to Redacted_(Previous name).txt.

Dependencies

- Python Libraries
    - Mandatory: re, os, sys, argparse, logging, spacy
- SpaCy language model

2.6. Apportioning of requirements

File Handling

- Functional Requirements
    - Accept .txt files
    - Ensure readability and no corruption
    - Rename and save redacted files
    - Preserve original file structure
- Non-Functional Requirements
    - Handle large file efficiently

PII Detection

- Functional Requirements
    - Predefined PII types
- Non-Functional Requirements
    - High detection accuracy
    - Customizable

Censoring

- Functional Requirements
    - Censorship via Masking
    - Configuration for different PII types
- Non-Functional Requirements
    - Irreversible
    - Consistent formatting

Verification

- Functional Requirements
    - Re-scan censored files
    - Re-masking triggered after identifying missed PII
    - Record failures for future correction
- Non-Functional Requirements
    - False negative handling
    - Support for multiple verification passes

Re-censoring

- Functional Requirements
    - Re-masking with censor logic on failed sections
    - Record number of censor passes
    - Error handling if loop occurs
- Non-Functional Requirements
    - Log actionable errors

Output Generation

- Functional Requirements
    - New .txt file with masked content
    - Formatting Compliance with inputted file
    - Save to user-specified directory
- Non-Functional Requirements
    - Unique output with set naming convention
    - Metadata file for censored entities

Testing and Logging

- Functional Requirements
    - Testing for file handling, detection accuracy and correct censorship
    - Logging for PII types, censored entities and errors
- Non-Functional Requirements
    - Logs must be written in structured format
    - Logging system should not affect performance

Configuration and Extensibility

- Functional Requirements
  - Configuration of PII types to detect (email, name, etc.)
  - Customization of redaction replacement styles (*** or *[REDACT]*)
- Non-Functional Requirements
  - The program configuration should be easily readable and editable via a text editor

Command Line Interface

- Functional Requirements
  - The program must accept an input path, an output directory, a help command and verbose mode
- Non-Functional Requirements
  - Command Line Interface (CLI) should be minimalist and straight forward

3. Specific requirements

3.1. External interfaces

a) Name of item:
  - Input Text File (.txt)
  - Redacted Text File (.txt)

b) Description of Purpose:
  - The input file is the unredacted text file that contains PII
  - The output file is the redacted text file that does not contain PII

c) Source of Input or Destination of Output:
  1. Source of Input:
     - The user provides the input file via GUI (if not console run)
  2. Destination of Output:
     - The redacted text is generated and saved to a specified file via GUI (if not console run)

d) Valid Range, Accuracy, and/or Tolerance:
  1. Valid Range:
     - Input files specified by user must be text files (.txt) and English text
  2. Accuracy:
     - The system must detect and redact all PII accurately
  3. Tolerance:
     - If the input file is missing or in an incorrect format, the program returns an error

e) Units of Measure
  - File Size: There is no strict limit on file size because standard desktop storage size is assumed

- Text Length: The number of words or characters

f) Timing
   - Processing occurs immediately after the user executes the command and usually takes a few seconds

g) Relationships to other Inputs/Outputs
   - Each input file will map to one and only one redacted output file

h) Screen Formats/Organization
   - The system will provide screens for uploading files for PII redaction
   - If console run, screen formats are not applicable

i) Window Formats/Organization
   - The system will provide a window for file uploading
   - If console run, window formats are not applicable

j) Data Formats
   i. Input File:
      - Plain text file (.txt) in regular English

   ii. Output File:
      - Plain text file (.txt) with PII redacted

k) Command Formats
   i. Standard usage:
      - In GUI mode, the system will show a window for input and output file specification
      - If console run: pii_redactor <input_file.txt> <output_file.txt>

   ii. Example:
      - In GUI mode, the user will click to open file explorer for selecting input and output files
      - If console run: pii_redactor.exe report.txt report_redacted.txt

l) End Messages
   i. Success:
      - “Redacting Complete. Output saved to <filename>.”

   ii. Error:
      - “Error: Missing input file.”
      - “Error: Invalid file format. Only .txt files are supported.”
      - “Error: Unable to read input file.”

3.2. Functions

a) Validity checks on the inputs
   - The system shall accept input files of text document format (.txt)
   - The system shall verify input files for correct format and to make sure it’s not empty
   - The system shall reject unsupported file formats and display an error message

b) Exact sequence of operations
   - The system shall accept text files, detect all PII, redact the PII from the file, and generate the output file without the PII.

c) Responses to abnormal situations
   - Overflow – The system shall report an error if the file is too large to be processed.
   - Communication facilities – The system shall notify the user if the input file cannot be read/processed or if the output file cannot be written.
   - Error handling and recovery – The system shall stop processing the input file if an error occurs and reports it to the user, allowing them to retry.

d) Effect of parameters
   a. The system shall get the input file from the user which is the file to be processed
   b. The system shall redact all PII from the input file and write the result to the output file, which is specified by the user

e) Relationship of outputs to inputs
   a. The system shall keep the same order of words and lines in the output file as the input file.
   b. The system shall replace all redacted PII with a generic placeholder according to its type.

3.3. Performance requirements

Response Time

- The system should respond immediately to user input.
- File size checks should prevent abnormalities in response time which would otherwise be dependent on file size.

Capacity

- System shall be able to handle multiple files per request.
- The system shall run independently on the user's device and therefore not limited by use of the system on another device.
- The size of an individual file to be processed shall be limited in accordance with file size checks.

Accuracy

- The system shall perform at 100% accuracy to ensure it complies with data privacy regulations.
- System shall not rely on any human interference to assist in reaching accuracy standard.

3.4. Logical database requirements

This system will not use a database. Input and output files will be stored locally on a user's computer, which will be specified at runtime. The system will process all data in memory.

3.5. Design constraints

Files being input into the program must adhere to – but not limited to – the government, medical, or academic record standards. Although the program is designed to sparsely identify and redact PII, there are edge cases for certain record formats set by standards.

3.5.1. Standards compliance

Since the program accepts a variety of records, ranging from personal documentation to government records, the input files must adhere to their respective standards set by the field it is coming from. For an example, an input file containing a tax record in the United States must adhere to the format set by the Internal Revenue Service. Another example would be an input file containing medical records where they must adhere to the legal medical record standards.

3.6. Software system attributes

3.6.1. Reliability

At the time of delivery, the program will have all required Python libraries compiled into the executable file such that users are not required to install Python and Python libraries.

3.6.2. Availability

The program remains inactive until the user runs it, at which point the program will become active to complete its tasks. Once the tasks are completed, the program returns to inactivity until it is run once again. If the program crashes, the file input retains its original data and format because the program reads the data into memory and modifies it in memory.

3.6.3. Accuracy

The system correctly identifies and redacts only the specified PII types (names, emails, phone numbers, and SSNs). It ensures no PII elements outside the defined scope are altered while maintaining a low rate of false positives and false negatives. Accuracy directly impacts trust in the redacted outputs since missed detections leave sensitive information exposed, while over-reduction can reduce document usability. Testing with labeled data sets helps measure precision, recall, and overall F1 score to verify this requirement.

3.6.4. Security

Security ensures that all sensitive information handled by the system is fully protected during processing and storage. No raw PII will ever be stored in logs, caches, or crash dumps, and any temporary files created during processing will be securely encrypted or deleted after use. This prevents unauthorized access and minimizes the risk of data leaks or breaches. Following secure coding practices and conducting regular checks helps maintain compliance with privacy and security standards.

3.6.5. Maintainability

The program's source code will be modulated to allow for quick identification and modification of certain regex patterns or tokenization methods that are causing issues. Since the source code is small, it will be easy to maintain with it being modulated.

3.6.6. Portability

100% of the program's source code will be written with Python on a Windows machine. Since Python is an interpreted programming language, it has proven to be portable across different operating systems. The GUI, if implemented, for the program will be built each for Windows, macOS, and Linux operating systems.

3.6.7. Irreversibility of Redaction

Once PII is removed from a document, it cannot be recovered or reconstructed by any means. Redacted outputs must replace or completely obscure sensitive data, so the original text is unrecoverable, even with technical tools or forensic methods. This guarantees compliance with privacy regulations and builds trust that personal information is permanently protected. Testing should confirm that no traces of the original PII remain in any redacted file.

3.6.8. Performance/Scalability

The system can handle multiple files and large data volumes quickly without slowing down or crashing. The program should process batch inputs efficiently while keeping memory and CPU usage within reasonable limits. As workloads grow, the system must maintain consistent speed and responsiveness. Benchmark testing helps verify that processing time and resource usage stay stable as the number or size of files increases.

3.6.9. Resilience

The system can continue operating even when encountering errors such as malformed files, unexpected inputs, or timeouts. Instead of crashing, the program should handle these issues gracefully by skipping problematic files, logging the errors, and continuing with the remaining tasks. This prevents single points of failure and allows batch processing to finish successfully despite individual file errors. Error handling and fallback mechanisms help maintain overall system stability and reliability.

3.6.10. Backup/Recovery

The system prevents data loss or corruption if a crash or unexpected shutdown occurs during processing. A safe writing strategy, such as writing to a temporary file before finalizing output, helps protect both original and redacted files. If the process is interrupted, the system can recover gracefully without leaving incomplete or corrupted output behind. This guarantees data integrity and reliability for every run.

3.6.11. Documentation

Documentation includes a clear user guide and a section explaining the system’s known limitations. This helps users quickly learn how to use the program and understand any boundaries or potential issues.