Esther Kim
HCI 584 – Revised project spec

General description of the project

This project aims to create a lightweight AI-assisted screening tool to support researchers in conducting systematic reviews. Specifically, it focuses on automating the initial title and abstract screening process, which is typically one of the most time-consuming parts of the review workflow. Researchers will upload a `.txt` file containing article metadata (title, author, abstract, year of publication, DOI) exported from bibliographic databases such as Clarivate Endnote. The program will parse the file, extract relevant fields, and use ChatGPT to generate a relevance score (1–10) for each article based on predefined inclusion criteria. These scores will help researchers prioritize which articles to review in depth.

Originally, the plan was to build a fully functional web application with GUI, filtering, and visualization features. This has been revised so that the first version focuses on getting the core parsing and scoring functionality working in a command-line environment, with GUI and filtering moved to later development stages. The application currently uses Python with pandas for data handling, re for text parsing, and the OpenAI API for relevance scoring. GUI and visualization features will be implemented later.

<u>Task vignettes (User activity flow)</u>

Upload and parse article file

The user starts by opening the app and selecting an exported `.txt` file containing article metadata. The file uses field codes such as %A for author, %T for title, %X for abstract, %D for year, and %R for DOI. The app parses the file using regular expressions (rather than the originally planned CSV import) to handle inconsistent file formats. The extracted information is structured into a table with the following columns: Authors, Title, Abstract, Year Published, and DOI. This table is then saved as parsed_articles.csv.

Automated relevance scoring

The user then runs the scoring script. The application reads the structured parsed_articles.csv file, extracts the title and abstract of each article, and sends it to the ChatGPT API with a well-defined rubric to score the article's relevance from 1 to 10. The

score is appended as a new column called relevancy_score in the dataset. This is a change from the original plan, which initially envisioned generating both a numeric score and textual notes. For simplicity and reliability, only a numeric score is returned.

Output and review

Once scoring is complete, the app generates a new file called parsed_articles_scored.csv. The user can open this file in Excel or another spreadsheet program to view the scores, sort articles by relevance, and filter as needed. Future versions will include GUI elements for filtering and visualization, such as histograms or tables within the app interface, to make the review process more interactive.

Technical flow

The original planned flow included multiple components such as deduplication, caching, interactive filtering, and visualization.

The current flow is simplified:
1. Upload .txt file
2. Regex-based parsing to DataFrame
3. Relevance scoring using GPT
4. CSV export with scores

This simplification was made to ensure the core functionality works reliably before building more advanced features. User interaction is currently limited to file selection and command-line execution. GUI integration will come in the next development phase.

Revisions compared to the original spec

• Changed: GUI features (upload interface, filtering, visualization) were moved to a later milestone to prioritize core functionality.
• Changed: Parsing approach was revised from structured CSV/TSV imports to regex-based parsing of raw text.
• Removed (for now): Deduplication, logging, caching, and “AI notes” features.
• Added: Retry logic and stricter prompt structure to ensure consistent GPT integer outputs.
• Added: Secure key management using a keys.py file and sys.path modification for clean imports.

Proposed additional features (maybe?)

• Tkinter GUI for uploading files and running scoring without command line interaction.
• Interactive visualization of score distribution (histogram or bar chart).
• Filtering based on score thresholds (e.g., only display articles with a score of 8 or higher).
• Export of filtered results into a new CSV or Excel file.

Milestones (completed & to come)

1. Core Parsing & Scoring: Implemented regex parsing, GPT scoring, and CSV export: Completed.
2. GUI Implementation: Build Tkinter-based interface for file upload and scoring trigger.
3. Visualization: Add graphs to display score distribution and simple filters.
4. Export & Filtering: Implement filtered result exports.
5. Error Handling & Enhancements: Improve robustness with error handling, logging, and optional notes.

Self-assessment and feasibility

The revised plan focuses on a strong functional core first and defers the more time-intensive interface and visualization components. This approach ensures that the app remains feasible within the timeline while retaining flexibility to add more advanced features later. The main challenges during implementation so far were file parsing inconsistencies and API key import issues, both of which have been resolved. The remaining tasks are well-defined and can be built on top of the existing structure.