COSC 499
Project Proposal Group 9

Sami Jaffri, TianXing Chen, Evan Pasenau, Jinxi Hu, Kevin Zhang, Ryan Eveson

Table of Contents

Tech Stack

Backend

- Python (Flask)

Frontend

- React, Node.JS, Typescript (potentially)

Database

- Docker SQL (undetermined where it is hosted)

Functional Requirements

File Collection

- The system should allow users to collect files from a single folder on their computer
- The system should allow users to collect multiple files from multiple folders at once if required.
- The system should allow users to collect files from many devices.
- The system should only collect files when prompted by the user. (This is conditional based on whether or not the system needs to be running at all times for live feedback)
- The system should request permission before it attempts to scan any files.
- The system should allow users to search and filter the files that are collected
- The system should generate a report summarizing the collected files
- The system should analyze the contents of each file collected.

Organizing Data and Metrics

- The system should organize files by either: time of day, day of week, subject, content type, or file size
- The system should be able to identify similarities and trends across data.
- The system should visualize metrics either using graphs, or other plotting methods.

Front End Dashboard

- The system should allow users to register, sign in, and log in.
- The system should allow users to filter their projects by metric type.
- The system should generate and display graphs based on user-selected metrics
- The system should provide a dashboard which showcases analyzed, and collected data.
- The system should be flexible for administrative control. (i.e. Potential Admin Page using token access)

Summarization and Analysis

- The system should summarize files based on data, type, and size
- The system should generate summaries that are easy and simple to understand for all users.

Non-Functional Requirements

Performance

- The system should process and summarize files within a reasonable time.
- The dashboard should load within < 3 seconds on a wireless connection, and < 2 seconds on a wired connection.

Scalability

- The system should be able to support simultaneous analysis from various devices up to a maximum of 3.

Reliability

- The system should be able to handle empty or inaccessible folders without crashing.
- The system should maintain >99% uptime during runtime.

Usability

- The system should be easily accessible for at minimum 80% of new clients. (i.e. it should be usable with minimum assistance from support)

Security

- The system should be secured via user authentication at log in.
- The system should restrict file access to authorized users ***ONLY***

Portability

- The system should be able to run on many different operating systems, and should be lightweight enough to run on 90% of modern hardware

Maintainability

- The system should follow a strict code design to make debugging, and extension easy.

User Scenarios

Target User Groups

- Recent graduates, early/late professionals (including but not limited to: engineers, architects, software engineers, etc.).

Usage Scenarios

- Preparing for an interview
    - The system showcases progress via collected artifacts.
- Building a resume
    - Summarizes key contributions
- Maintaining a project portfolio
    - Organizes by subject, and timeline

- Receiving mentor feedback
    - Summarizes trends used for evaluation

Testing

(sorted by functional and non functional headings aforementioned)

File Collection (functional)

- Verify that files from unsupported formats are flagged, and or skipped without a hitch.
- Verify that user permissions are requested ***before*** any scan begins.
- Verify that simultaneous collection from multiple devices does not overwrite/duplicate files. If this is the case, verify with the user that it's acceptable.
- Verify that search and filtering return only relevant results.
- Verify that generated reports accurately reflect the collected files.

Data Organization (functional)

- Verify that all files are correctly categorized.
- Verify that similarity and trend detection produces accurate outputs.
- Verify that graphs update dynamically when files are added.

Front-End Dashboard (functional)

- Verify that users can register, sign in, and log in successfully.
- Verify that invalid credentials are rejected with appropriate error messaging.
- Verify that users can filter projects by metric type and that the results are accurate.
- Verify that the graphs are generated correctly and showcase accurate, relevant data.
- (Conditional) Verify that the management/admin page is accessible only to authorized users.

Project Summarization and Analysis (functional)

- Verify that summarization includes file type, size, and key data.
- Verify that text summaries are easy to understand and accurate.
- Verify that reports include all ***selected*** files.

Performance, and Scalability (non-functional)

- Verify that system can process an upwards of 100 files in under 5 seconds (benchmark can be adjusted)
- Verify that the dashboard loads within 3 seconds on a wireless connection, and 2 seconds on a wired connection.
- Verify that the system scales up to multiple devices with minimal performance impact.

Reliability and Error Handling (non-functional)

- Verify that inaccessible folders produce an error message without crashing the entire system.
- Verify that corrupted files are handled properly. (logged, skipped, and user is notified)
- Verify that uptime is maintained at >= 99% during testing.

Usability (non-functional)

- Verify that at least 80% of new users can complete a task without major support.
- Verify that graphs and summaries are easily interpreted without technical knowledge.

Security (non-functional)

- Verify that ***only*** authenticated users are able to access their files and dashboards.
- Verify that user sessions expire after inactivity.
- Verify that unauthorized access attempts are logged. (either via admin page, or user notification)

Portability (non-functional)

- Verify that the system will run correctly on Windows, macOS, and Linux with no visual glitches.
- Verify that docker deployment works across various environments.

Maintainability (non-functional)

- Verify that code structure allows one module to be updated without impacting others