Architectural Spike

Authors:

Tianyang Wang

Kate Moreland

DJ Oakman

Haoran Ding

Submitted in partial fulfillment of the requirements for COMP 4710 Senior Design to the Department of Computer Science and Software Engineering, Samuel Ginn College of Engineering, Auburn University

Auburn, Alabama

September 2, 2025

Comp 4710 Senior Design Project

Fall 2025

[System Metaphor: 2](#_Toc1991046789)

[Cycle Intent: 3](#_Toc403161492)

[Design Decision: 3](#_Toc1788612654)

[User Stories: 3](#_Toc1860205896)

[Design Documentation: 9](#_Toc798598088)

[Lessons Learned: 11](#_Toc1283054773)

[Test Results: 11](#_Toc1158213032)

[Management Plan: 11](#_Toc798512359)

[Memoranda: 11](#_Toc1890376079)

[Contributions: 12](#_Toc810013722)

[Source Code: 13](#_Toc789898019)

[Sponsor’s Approval: 13](#_Toc1588020682)

# System Metaphor:

To design and create a system which can fix and identify accessibility issues from an uploaded PDF, Word Document or PowerPoint.

# Cycle Intent:

Our team’s intent for this spike is to develop a design for a device that will upload documents to a server for a user, process the document for accessibility issues, and return a display a list of things that it finds wrong with the assignment. We also want to get a start to begin our backend service which will allow a user to fix and identify issues from a PDF file of their selection.

# Design Decision:

We designed our system to process PDFs, PowerPoints, and Word documents in a secure and consistent way. Files are uploaded through the front end and sent to the backend, where all processing takes place so that sensitive details, like API keys, remain protected. Because each format is different, we handle them separately. PDFs are sent to Adobe’s PDF Accessibility Checker and Tagger for reliable accessibility checks and tagging. Word documents are not fully supported yet, but for now we either convert them to PDFs for Adobe processing or use Python to check for accessibility issues. PowerPoints also return an error, but we run partial checks with Python to provide as much feedback as possible. Once processing is complete, results are returned to the front end for display, with an option to download the updated file. This design ensures that we can provide accurate results now while building a foundation to expand support for Word and PowerPoint files in the future.

# User Stories:

Functionality  
**Name:** Upload documents from a file selection

* **Summary:** Allow users to upload documents (PDF, Word, PowerPoint) through a file selection component so that the files can be programmatically stored and accessed later.
* **Description:**

A user should be able to select and upload a document from their local device using a web-based file selection component. Supported formats include PDF, DOCX, and PPTX. Once uploaded, the file will be stored in a designated system location or database and made accessible to backend code for further processing, retrieval, or integration with other features.

* **Planned Hours:** 3
* **Planned Hours this cycle:** 2
* **Actual Hours:** 2
* **Hours this cycle:** 2
* **Coder:** DJ Oakman
* **Tester:** Kate Moreland
* **Reviewer:** Kate Moreland
* **Status:** Completed

**Name:** Send PDF to Back-End for Accessibility Processing

* **Summary:** After a user uploads a PDF and presses “Submit” on the front end, the document will be sent to the back end to be processed by the Adobe PDF Accessibility API. The result will include an accessibility report and optionally a remediated PDF.
* **Description:**  
   Users will be able to upload PDFs through the front-end interface. When the user submits the document, it will be transmitted securely to the back end. The back end will call the Adobe PDF Accessibility API to:
  + Check the PDF for WCAG accessibility compliance (tags, alt text, reading order, headings).
  + Generate an accessibility report.
  + Optionally provide a remediated version of the PDF. The system should handle errors gracefully (e.g., invalid PDFs, API failures) and return clear feedback to the user.
* **Planned Hours:** 4
* **Planned Hours this cycle:** 3
* **Actual Hours:** 1.5
* **Hours this cycle:** 1.5
* **Coder:** DJ Oakman
* **Tester:** Hoaran Ding
* **Reviewer:** Tianyang Wang
* **Status:** Completed

**Name:** Display Uploaded PDF and Accessibility Results

* **Summary:** After a PDF is processed by the back end via the Adobe PDF Accessibility API, the front end will display the uploaded PDF, the accessibility report, and optionally a remediated PDF for the user to review.
* **Description:**  
   Once the back-end finishes processing a PDF, the system will return:
  + **Original uploaded PDF** – displayed or downloadable.
  + **Accessibility report** – showing WCAG compliance results, highlighting missing tags, alt text issues, reading order problems, etc.
  + **Remediated PDF (if available)** – downloadable so the user can replace the original document with an accessible version.
  + The front end should present this information clearly, with sections for the report and file downloads, and handle any errors if processing fails.
* **Planned Hours:** 8
* **Planned Hours this cycle:** 4
* **Actual Hours:** 4
* **Hours this cycle:** 4
* **Coder:** DJ Oakman
* **Tester:** Hoaran Ding
* **Reviewer:** Tiangyang Wang
* **Status:** Completed

**Name:** Alternative text for images and figures

* **Summary:** If there are pictures and/or figures in the file, then they will have to be substituted with accessible text. This will make sure that it is compliant with the WCAG standards and allows for screen readings of these files.
* **Description:**

Whenever a user uploads a file that contains an image or figure, the system will flag it and then provide a suitable substitute for accessible text. The interface will allow for the following:

* + **Display** - Present the uploaded file with the image and the text that has been generated as well.
  + **Review** - The user should be able to check the information and ensure that it is correct.
  + **Warning** - If an image is of low quality, corrupted, or cannot be processed for any other reason, it will flag it so the user is aware.
  + The back end should be able to scan the file, extract the images or figures, send them to a service for accessible text generation, and re-integrate that inside the new document.
* **Planned Hours:** 8
* **Planned Hours this cycle:** 0
* **Actual Hours:** 0
* **Hours this cycle:** 0
* **Coder:** N/A
* **Tester:** N/A
* **Reviewer:** N/A
* **Status:** New

**Name:** Accessible headings and navigation

* **Summary:** The system needs to automatically analyze the structure of the heading in the files to make sure there is a logical hierarchy. This will create an accessible navigation for screen readers to correctly analyze the file.
* **Description:**

Whenever a user loads a file, the system will need to automatically scan the content and analyze all of the headings and their types. Then the system will:

* + **Display a document outline** – Show a clear outline of the document’s structure with the headings included
  + **Identify issues** – Flag any issues that are discovered, like H1 followed by H3
  + **Automatic remediation** – The headings will automatically be resolved to fit with the WCAG rules
  + The back end has to be able to scan and analyze the file’s structure, apply the WCAG set of rules for heading hierarchy, and generate a new accessible file.
* **Planned Hours:** 6
* **Planned Hours this cycle:** 0
* **Actual Hours:** 0
* **Hours this cycle:** 0
* **Coder:** N/A
* **Tester:** N/A
* **Reviewer:** N/A
* **Status:** New

Usability

**Name:** File upload and format handling

* **Summary:** The app must allow users to upload PDFs, PowerPoints, or Word documents, and provide clear feedback on what happens with each format.
* **Description:**

Whenever a user uploads a file, then the system will:

* + Display accepted formats
  + Show upload progress
  + Provide clear message about what formats are supported
  + Notify the user if an unsupported file is uploaded

|  |  |
| --- | --- |
| Planned Hours | 3 |
| Planned Hours this Cycle | 1.5 |
| Actual Hours | 1.5 |
| Hours this Cycle | 1.5 |
| Code | Kate Moreland |
| Tester | DJ Oakman |
| Reviewer | DJ Oakman |
| Status | In Development |

**Name:** Secure backend processing

* **Summary:** The app must securely send files to the backend for processing without exposing sensitive information like API keys.
* **Description:**

When a file is uploaded, the system will:

* + Route the file to backend services via secure connections
  + Ensure API keys and processing logic remain hidden from the frontend
  + Return only results and processed files to the frontend
* **Planned Hours:** 3
* **Planned Hours this cycle:** 0
* **Actual Hours:** 0
* **Hours this cycle:** 0
* **Coder:** N/A
* **Tester:** N/A
* **Reviewer:** N/A
* **Status:** Not Started

**Name:** Results display on frontend

* **Summary:** The app must clearly show results of accessibility checks in a simple, user-friendly format.
* **Description:**

After processing is complete, the system will:

* + Present results in a structured list or table
  + Highlight errors and warnings with clear labels
  + Offer tooltips or short explanations for each issue
* **Planned Hours:** 3
* **Planned Hours this cycle:** 0
* **Actual Hours:** 0
* **Hours this cycle:** 0
* **Coder:** N/A
* **Tester:** N/A
* **Reviewer:** N/A
* **Status:** Not Started

**Name:** Error handling and messaging

* **Summary:** The app must give clear, helpful error messages when something goes wrong.
* **Description:**

If an error occurs, the system will:

* + Show a user-friendly error message (e.g., “This file type is not fully supported yet”)
  + Suggest next steps where possible (e.g., “Try converting to PDF first”)
  + Log technical details for developers while hiding them from end-users
* **Planned Hours:** 3
* **Planned Hours this cycle:** 0
* **Actual Hours:** 0
* **Hours this cycle:** 0
* **Coder:** N/A
* **Tester:** N/A
* **Reviewer:** N/A
* **Status:** Not Started

Performance

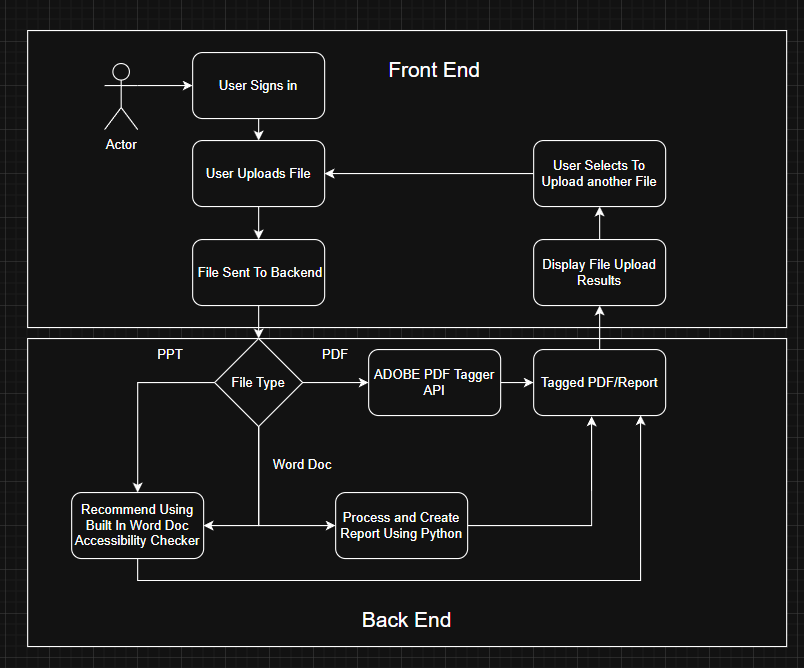
**Name:** App performance and responsiveness

* **Summary:** The app must process uploads and display results quickly so that users have a smooth experience and don’t feel delayed.
* **Description:**

When a user uploads a file, the system will:

* + Ensure uploads complete within a reasonable time (under 10 seconds for average files)
  + Optimize backend requests so that large files or multiple users do not slow down performance
  + Provide clear feedback if processing takes longer than expected
* **Planned Hours:** 3
* **Planned Hours this cycle:** 0
* **Actual Hours:** 0
* **Hours this cycle:** 0
* **Coder:** N/A
* **Tester:** N/A
* **Reviewer:** N/A
* **Status:** Not Started

# Design Documentation:



High-Level System Architecture

* Frontend
* Backend
* Third-Party Services

Frontend Design

* UI Components

Backend Microservices

* File upload service
* Document processing service
* Accessibility reporting service

Third-Party Integration

* Adobe PDF Accessibility API
* Generative AI to convert from images to text and other cases

# Lessons Learned:

We have had the opportunity to learn more about WCAG standards and what elements they encompass in the higher education world. We learned that all of the documents that are made available to students are equally accessible to all students, so there are some modifications that must be made to them. We looked through all of the standards here: <https://www.w3.org/TR/WCAG21/> and saw that some standards are required earlier than others, so those should be a higher priority on our list. The specific standards that are higher on our list are WCAG 2.1 Level AA, which is required by April 24, 2026.

# Test Results:

Since we are starting on a new project, we do not have any test results yet.

Management Plan:

We are planning to meet with our sponsor every 2 weeks to update them on our progress. Once we understand the accessibility requirements then we can split up our tasks and get started on the project.

# Memoranda:

8 – 26 – 25:

* Got together with our team and created a team Discord
* Kate emailed our sponsor, Asim Ali, to find a time to meet and get resources for our project

Total Hours: 1

9 – 2 – 25:

* Got together with our team and scheduled a good time for us to meet our sponsor
* Tianyang emailed our sponsor to schedule our Zoom meeting
* Kate created the document for the architectural spike

Total Hours: 1.5

9 – 3 – 25:

* Had our first meeting with the whole project group
* We took these notes:
  + Dive in and understand the WCAG guidelines so we can see what we are facing.
  + Want to check the accessibility requirements for Word documents, PowerPoint presentations, and PDF files.
  + Want to use SSO as a login feature.
* Figured out that we are working on a digital accessibility project
* DJ created a Teams chat with all our sponsors and team members
* Tianyang created a GitHub for our project that we can pass on to future team members
* We all worked on the architectural spike

Total Hours: 2

9 – 4 – 25:

* Created backend code for processing a file using the Adobe PDF tagger api
* Worked on architectural spike.
* Worked on presentations
* Created starter code for frontend project

9 – 8 – 25:

* Created frontend code for processing a file and signing in
* Worked on architectural spike formatting and small details.
* Worked on presentation

Total Hours: 4

# Contributions:

* **Kate Moreland:** created presentation, created memoranda, lessons learned, communicated plans
* **DJ Oakman:** created frontend starter code, and backend starter code. Created most user stories, cycle intent, design decision, system metaphor.
* **Horan Ding:** Was highly involved in discussion. Got assigned to our team very late so he hasn’t had a fair chance to contribute yet.
* **Tianyang Wang:** Created GitHub for code sharing, figured out how to grant access as collaborators through code sharing, helped plan design/processes through meetings with sponsors.

# Source Code:

[https://github.com/TiantangWangAUBURN/Architectural-Spikelin](https://github.com/TiantangWangAUBURN/Architectural-Spike)

# Sponsor’s Approval:

A screenshot of a chat

AI-generated content may be incorrect.