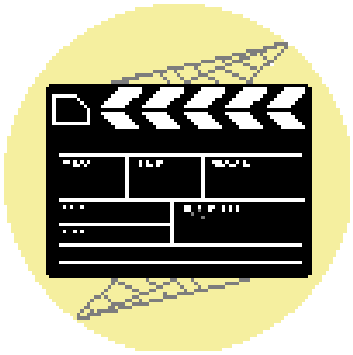
Critical Design Review Document v1.0

Discovering Early Hollywood

Samuel Backer



by Willy N' Gang: Xander Dufour, Liam Hillery, Vincent Lin, Patrick Storer, Caleb Thurston

Abstract

The Discovering Early Hollywood project focuses on saving and sharing film copyright descriptions from the Library of Congress. These records, created before 1912, are one of the few ways early movies were documented, since filmmakers often did not preserve their work. Our team, working with Professor Samuel Backer at the University of Maine, is building a website and database to organize this material so that researchers, film fans, and historians can easily explore it. Tools like PostgreSQL, GitHub Actions, Selenium, and Docker help us manage data, test features, and make a reliable system. So far, we have delivered requirements and design documents, created a prototype, and set up testing environments. The project shows strong progress toward a platform that makes these film records accessible and easily accessible. The system is on track to meet its goals, with future work focused on improving metadata, refining the user interface, and improving the search functionality.

Preface

This document is created by the team Willy n' Gang (consisting of Xander Dufour, Liam Hillery, Vincent Lin, Patrick Storer, and Caleb Thurston) as part of the Discovering Early Hollywood project and as a requirement for University of Maine class COS 397: Capstone 1. It provides an overview of our design process, decisions, challenges, and progress to date. The goal of this Preface is to introduce the context of the Critical Design Review Document (CDRD) and prepare the reader for the reflections and evaluations that follow.

Table of Contents

Symbols (Nomenclature)

Optical Character Recognition (OCR) refers to a technology that converts images of text from scans, photos, or PDFs into digital text that is readable to the machine.

PostgreSQL is a program that manages and hosts a relational database compatible with Structured Query Language (SQL) with a focus on complex queries.

Summary

This Critical Design Review Document (CDRD) provides an overview of the system's current requirements, design, testing and system structure. It examines the decisions/methods made behind key design choices and documents how well the proposed system aligns with the needs of the client, Samuel Backer. The system currently is only in the simple prototyping stage, but the project has established a solid foundation based on the requirement specifications. The work completed so far positions the team to move confidently into the full implementation and testing phase for next semester. Overall, this document serves as a checkpoint to assess readiness and clarity of the proposed system.

Introduction

Samuel Backer is a Professor of History at the University of Maine. He teaches about the mass culture movements that existed before the physical media that we have recorded. Backer teaches a course that concludes with the advent of film in America. In the early days of film, very few people thought to save their work, and in order to copyright a movie, filmmakers would have to submit a still image of each frame of the movie. In 1912, that law changed, allowing filmmakers to submit a description of their movie in order to have it copyrighted. These records form the bulk of our software database. We aim to digitize and organize the thousands of film descriptions released from the Library of Congress (LoC) into the public domain. Our work aims to provide a hub where researchers, film buffs, and historians can easily search and access these records, which have rarely had light shed on them.

Purpose

The purpose of this Critical Design Review Document (CDRD) is to evaluate, reflect on, and critique the entire Discover Early Hollywood project to date. This includes every aspect of design,

documentation, communication, feedback, and other processes carried out by our team, Willy n' Gang, in collaboration with our client, Samuel Backer, over the course of the semester.

Method

The Method section outlines the processes that our team used to understand the project requirements and develop the solutions presented in this review. It explains how we collaborate with our client to analyze film records and shape the direction of the Discover Early Hollywood project.

To address our client Samuel Backer's need for a website to organize his cleaned OCR data of early film copyright descriptions, our team followed a V-model systems development lifecycle (SDLC) to guide our work throughout the semester. We began by conducting initial client interviews to understand the historical context of the Library of Congress film records and how he wanted this year's requirements for this project to be different from last year's. Based on these discussions, we identified key requirements for the website. To review the requirements please go to Appendix A. Software Requirements Specification (SRS) Section 1.3: Purpose of the Product.

We reviewed several of the client's available cleaned OCR data descriptions and analyzed their structure, inconsistencies, and thought about the metadata needs. This research guided our decisions about categorization and data cleaning. We then shared our findings with the client and drew on his expertise as a film historian to refine and validate our categorization approach.

To explore possible design direction we looked at our requirements compared with previous year's project design and added much needed design changes to match this year's requirements. Based on these insights, we created prototypes, which we refined through internal critique sessions and some feedback meetings with the client.

Throughout the project, we used collaborative documentation tools (e.g. Jira, figma) to track decisions, and make consistent communication within the team. Semi-regular check-ins with Samuel Backer allowed us to refine our approach, verify historical accuracy, and ensure that our design aligned with the expectations of researchers, historians, and general film fans. These methods collectively help shape the evolving design of the Discover Early Hollywood platform.

Design

Design Parameters

The system is designed to support organization, search, viewing, and preservation of early film copyright descriptions. Core parameters include usability for non-technical users, and search features. More Detailed parameters and functional goals are documented in the Appendix A. System Requirements Specification (SRS) and further refined in the Appendix B. System Design Document (SDD)

Design Constraints

The design is constrained by the quality of historical source data given by our client, Samuel Backer and performance limitations of web-based systems. Additional constraints include, but not limited to limited development time and the current project scope.

Design Approach

The design follows a modular, layered web architecture separating the user interface, application logic, and database layers. Design decisions are based on requirements given in Appendix A. System Requirements Specification (SRS). This approach, along with architectural diagrams and system flow descriptions, is detailed in Appendix B. System Design Document (SDD) Section 2.

Design Problems

At the current stage, key challenges include, but are not limited to inconsistencies in client provided data, and limitations of the simple prototype. These issues are listed as risks and considerations rather than as implementation failures.

Design Details

For a more detailed specification such as the system architecture, database schema, file structures and data formats go to:

Appendix A. System Requirements Specification (SRS) for functional and nonfunctional requirements

Appendix B. System Design Document (SDD) for architecture, decomposition diagram and persistent data design

Appendix C. User Interface Design Document (UIDD) for User interface layouts and interactions.

Continuous Integration & Continuous Deployment

The Continuous Integration & Continuous Deployment (abbreviated CI and CD respectively) section details the tools and methods we will use to maintain clean, functional code throughout the development process. CI involves tools for testing code and documentation frequently to ensure that new contributions are not destructive to older ones, and CD involves providing up-to-date software to users automatically, such as an application that generates executables automatically when the code is altered. For Discovering Early Hollywood, we are implementing tests and static analysis (style checking) through GitHub Actions and deploying the code on a machine owned by the History Department at the University of Maine.

Testing Environment

The testing environment of the project is a set of software or hardware used exclusively to ensure the delivered code functions as expected, without modifying or using sensitive data. This restriction ensures that the software may be tested by any user without risk of modifying or removing data. Since this project involves a webpage that interfaces with a PostgreSQL database, we are using Selenium and Docker. Selenium is a tool for simulating user interactions with a website, such as filling out a text field or clicking a button, and Docker is a program that allows for code to be tested in a clean, minimal, and temporary environment. Docker allows the program to interface with a small, curated database during testing, without needing to access an external server.

Equipment

The Equipment section describes all hardware required for the deployment of the Discovering Early Hollywood project. In general, the only operations that must be deployed on permanent hardware are PostgreSQL server and the Flask backend. In practice, we are deploying both on a single machine located in the History Department at the University of Maine over the course of development, and the client will move one or both of these operations to a public-facing server if the project is deemed ready for release.

Deliverables, Milestones, and Timeline

Deliverables List

(Include explanations for each deliverable.)

- **Prototype delivery dates**
- **First tested version milestone**
- **Beta test milestone**

As part of the process in working with our client we plan on delivering a set of hard copy and digital deliverables. These deliverables are documentation and planning documents to catalog and document our plans, expectations, and implementations of our client's product. These deliverables also will include source code for the client's product, as well as any extra software needed to operate it.

Document	Due Date	Description	Delivery Method
System Requirements Specification Document	October 29th, 2025	Document detailing the requirements and specifications to go along with the project	Paper Copy, Digital Document uploaded
System Design Document	November 17th, 2025	Document detailing how the system will be designed and developed	Paper Copy, Digital Document uploaded
User Interface Design Document	December 3rd, 2025	Document detailing how the User Interface will be designed	Paper Copy, Digital Document uploaded
User Manual	TBA	A guide for a consumer or user's use, including how to use the project, and information they might need	Paper Copy, Digital Document uploaded
Administrator	TBA	Guide detailing policies,	Paper Copy,

Manual		procedures, and rules for operating and managing this project	Digital Document uploaded
Biweekly Status Reports	Biweekly (As a whole, end of semester 2)	Biweekly reports of our thoughts and current progress in the project	N/A (Only for in-class use)
Source Code	TBA	Source code of the project including the github repository	Uploaded to Git repository
Executable Program	TBA	Ready to use program with the finished product of the project	Compiled by team, given as executable to client
Other Required Software	TBA	Any other software necessary or heavily needed for the project to function correctly	Given to client in guide format for product setup
Critical Design Review Document	TBA	Document encompassing all of the work and documentation we've done regarding this project	Paper Copy, Digital Document uploaded
Configuration Item Record	TBA	Detailed Database entry to describe the configuration of different components of the project	Paper Copy, Digital Document uploaded
Final Project Report	TBA	A report detailing information about the project and reporting out our progress	Paper Copy, Digital Document uploaded

Milestones / Timeline Table

1. SRS Oct 29
2. SDD Nov 17
3. UIDD Dec 5
4. CDRD Dec 8
5. UI prototype Dec 19

Month	January (little into Feb if needed)	February	March	April	May
Document	CIR	AM	UM	FPR	Release
Testing	Unit Testing	Integration Testing	System Testing	Acceptance Testing	Final QA

Conclusions

The project is on track and meeting its goals. We've defined requirements, built prototypes, and tested core features with our client's input. The system already shows improvement by making the records easier to search and study, with UI improvements to improve usability. The main areas we still have to improve are metadata handling, search tools, and updates to allow users to flag incorrect information. With steady progress and helpful feedback, the final product will be a useful platform for both historians and film fans.

References

- Docker. (2024). Enterprise Application Container Platform | Docker. Docker. https://www.docker.com/
- Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025). *GitHub - Discovering Early Hollywood.* GitHub. https://github.com/BoppleOpple/DiscoveringEarlyHollywood
- Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025). *System Requirements Specification Document* [Unpublished manuscript]. Discovering Early Hollywood.
- Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025). *System Design Document* [Unpublished manuscript]. Discovering Early Hollywood.
- Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025). *User Interface Design Document* [Unpublished manuscript]. Discovering Early Hollywood.
- Fitzpatrick, G., Call, A., Ugbuaja, C., Ocaya, J., Wilkinson, M. (2025). *GitHub - Recovering Early Hollywood*. GitHub. https://github.com/gabrielfitzpatrickcs/CrashTestDummies-RecoveringEarlyHollywood
- Flask. (2010). *Welcome to Flask — Flask Documentation (3.0.x).* Palletsprojects.com. https://flask.palletsprojects.com/en/stable/
- Pytest. (2015). *pytest: helps you write better programs — pytest documentation.* Docs.pytest.org. https://docs.pytest.org/en/stable/

- Selenium. (2023). SeleniumHQ Browser Automation. Www.selenium.dev. https://www.selenium.dev/
- The PostgreSQL Global Development Group. (2019). PostgreSQL: The world's most advanced open source database. Postgresql.org. https://www.postgresql.org/

Acknowledgement

We would like to thank Anthony Veilleux, Arius Ahmad, Ben Yandell, Robert Kulow, and Vasu Patel for reviewing our documents over the course of this project. Their extra eyes and perspectives have been crucial in helping us catch errors and consider additional approaches. We would also like to thank Professor Yoo, Jonny Slyvain our Teaching Assistant , and our client Samuel Backer for guiding us through this project.

Appendix

Appendix A. Software Requirements Specification (SRS)

Table of Contents

1. Introduction

This is a capstone project for Samuel Backer, the historian who codes, in partial fulfillment of the Computer Science BS degree for the University of Maine. The project will explore the early years of Hollywood films which have long been lost to time. We will create a database and companion website to access the descriptions of movies which have been parsed into text format by the previous years team.

1.1 Purpose of This Document

This document will formally describe the need for the software we are creating and outline its function. It will detail the ultimate product we will be delivering along with all of its accompanying documents. It will clarify to any stakeholders, the process and design of the software. It will be a living document which will grow and shift as the project progresses.

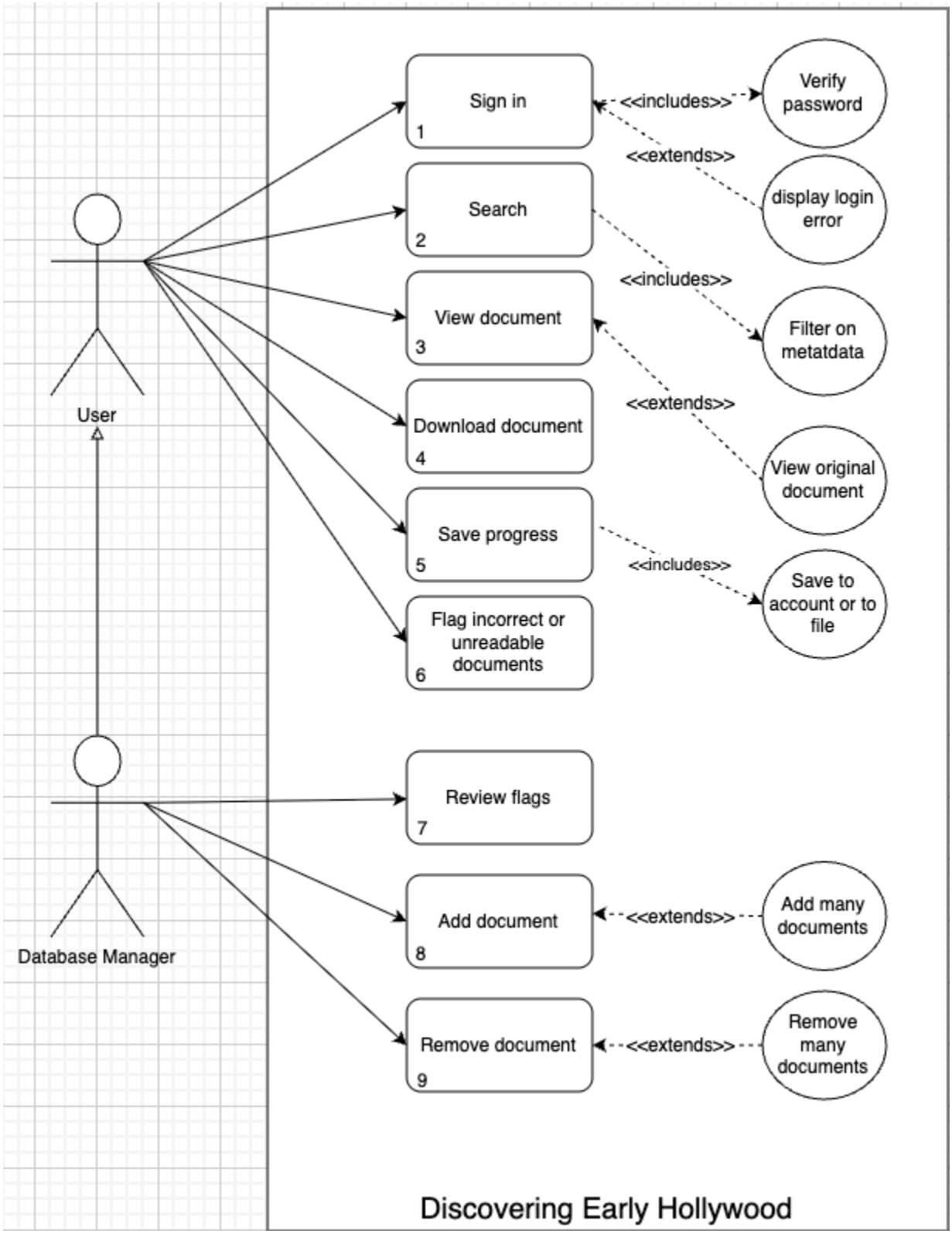
1.2. References

UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition, Martin Fowler, Addison-Wesley, 2003

1.3. Purpose of the Product

Samuel Backer is a Professor of History at the University of Maine. He teaches about the mass culture movements which existed before the physical media that we have recorded. His course ends with the beginning of the film. In the early days of film, very few people thought to save their work, and in order to copyright a movie, filmmakers would have to submit a still image of each frame of the movie. In 1912 that law changed, allowing filmmakers to submit a description of their movie in order to have it copyrighted. These records form the bulk of our software database. We aim to digitize and organize the thousands of film descriptions released from the Library of Congress into the public domain. Our work aims to provide a hub where researchers, film buffs, and historians can easily search and access these records which have rarely had light shed on them.

1.4. Product Scope



The above figure is the high-level Use Case Diagram for the Discovering Early Hollywood system.

2. Functional Requirements

The Functional Requirements Section outlines the specific functions and behaviors that the system needs to have to meet the client's needs. This section will describe how the system responds to user interaction, data handling and other core functionality (e.g. Search, save progress etc.). These requirements will serve as the foundation for design, implementation and testing throughout the development process.
Note: The priority for each Functional Requirement is designated a number from 1 to 5, with increasing numbers representing higher priority. .

Number	1	
Name	Sign in	
Summary	User signs into their user account	
Priority	3	
Preconditions	User has a user account	
Postconditions	User is logged into account	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	User clicks the Sign In Button	
Main Scenario	**Step**	**Action**
	1	User clicks Sign In Button
	2	System displays username & password Text fields
	3	User enters username
	4	User enters password
	5	User clicks Log In button
	6	System verifies user log in info
	7	System displays user log in verification success window
	8	System redirects user to user home page
Extensions	**Step**	**Branching Action**
	6a	User's username fails to verify**:** System Displays "Incorrect Username" dialog
	6b	User's password fails to verify: System displays "Incorrect Password" dialog
Open Issues	6	
Tests	1. Attempt to log in to user account with correct username but incorrect password 2. Attempt to log in to user account with incorrect username but correct password	

	3. Attempt to sign in with correct username and password but case insensitive

Number	2	
Name	Search feature	
Summary	User searches on parameters for entries related to their query.	
Priority	5	
Preconditions	User has the query parameters on which they will search	
Postconditions	User has returned list of entries which match the parameters of their search	
Primary Actor	User	
Secondary Actors	Database? Otherwise N/A	
Trigger	User enters a parameters into the search bar and presses enter	
Main Scenario	**Step**	**Action**
	1	The user enters parameter(s) into the search bar and presses enter
	2	The system queries the database and returns a formatted list of entries with thumbnail and title which match the parameters
Extensions	**Step**	**Branching Action**
	1a	User clicks on an article and then clicks back: They are returned to their previous search
Open Issues	5	
Test	1. Given a set of static query parameters search results on the first and second page of results remain the same 2. When searching on a year, or a range of years, no entries outside that specification is returned 3. When returning to a search from an article, viewed articles are denoted and the list of entries remains the same. 4. An article which a user wants to find is easily findable if they know basic information about the film. 5. Searching with no parameters does nothing.	

Number	3	
Name	View Document	
Summary	User views a document	
Priority	4	
Preconditions	User has searched/found a document	
Postconditions	User views the document	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	User clicks the specific snub for a document	
Main Scenario	**Step**	**Action**
	1	User clicks the specific snub for a document

	2	The system requests document information from database
	3	System receives document information from database
	4	System validates document information from database
	5	System displays document information on webpage
Extensions	**Step**	**Branching Action**
	4a	System unable to verify document information **:** Display 404 error message
Open Issues	6	
Tests	1. Go through steps 1 through 5, stress testing system to see if any part of system breaks 2. View a document that's known to be missing one type of information to confirm placeholder shows 3. Go to document where multiple pieces of similar information should be displayed to confirm it is	

Number	4	
Name	Download data	
Summary	The user downloads the data of a desired document to local machine	
Priority	3	
Preconditions	User is on a document's webpage	
Postconditions	The user now has a zip folder name {title}.zip inside it will contain the transcript ({title}_transcript.txt"), metadata ("{title}_metadata.json") and image("{title}_image.{type}").	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	The user hits the download button of a document.	
Main Scenario	**Step**	**Action**
	1	The user clicks on the download button.
	2	The system packages the related image, transcript and metadata in a zip folder from the database and displays it in a pop-up.
	3	The user clicks on the confirmation.
	4	The system begins a HTTPS transfer of a zip file containing the transcript ({title}_transcript.txt"), metadata ("{title}_metadata.json") and image("{title}_image.{type}") to users' local machine.
Extensions	**Step**	**Branching Action**
	2a	The system is missing one of the files**:** A missing file message with what's missing.
	3a	User clicks "Cancel": The system cancels the transfer process.
	4a	The zip file fails to download: A pop up message explaining why it failed(internet error, insufficient space, and etc.) then gives an option to retry.
Open Issues	3,4,6	

Test	1. Go through steps 1-4 using selenium for ~ 100 random complete documents and trying to verify content are same with database 2. Go through steps 1-4 using selenium for ~100 fake/real error filled documents to see if everything is working correctly. 3. Simulate different types error like internet to see if there is any corrupt file is downloaded 4. Go through step 1-3a using selenium for ~10 documents to check if system cancels transfer process

Number	5	
Name	Save progress(Bookmark)	
Summary	User saves the results of a query they have done to their account or to file, somewhat like returning to a filing cabinet.	
Priority	4	
Preconditions	N/A	
Postconditions	Users are able to return to the search they have previously searched with previously clicked entries denoted.	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	User saves progress on a search	
Main Scenario	**Step**	**Action**
	1	The user enters parameter(s) into the search bar and presses enter
	2	The system queries the database and returns a formatted list of entries with thumbnail and title which match the parameters
	3	The user clicks the save progress button
	4	The system asks the user if they would like to save to account or to file
	5	The user clicks account or file
	6	The system if account: verifies the user is logged in, asks them to log in if they are not, and saves to account or downloads to file
	7	The user returns to search in the accounts tab, importing it if it is a file
Extensions	**Step**	**Branching Action**
	N/A	N/A
Open Issues	5	
Test	1) A saved search which is saved as a file produces the same search as one saved to account 2) Search is the same on original and on return. 3) User is able to save many search progresses 4) Previously viewed entries are stored and denoted within save progress	

	5) Any search file from our system is able to be imported in any instance of our system with no to light modifications.

Number	6	
Name	Flag Incorrect or Unreadable Documents	
Summary	A user flags some aspect of an entry as incorrect	
Priority	2	
Preconditions	The user is signed in	
	The user is on an entry's webpage	
Postconditions	The correction is stored in the database	
Primary Actor	User	
Secondary Actors	N/A	
Trigger	The user clicks the "Flag Incorrect Information" button	
Main Scenario	**Step**	**Action**
	1	The user clicks the "Flag Incorrect Information" button
	2	The system provides the "Submit Correction" pop-up
	3	The user checks any of the checkboxes in the "Error Location section of the pop-up"
	4	The user adds a description of the error to the pop-up's text input
	5	The user clicks the "Submit" button
	6	The system appends the correction to the database, including the user's uid, selected error locations, and comment
Extensions	**Step**	**Branching Action**
	6a	No checkboxes are checked: An error entitled "No Location Specified" is displayed, but the "Submit Correction" pop-up remains
Open Issues	2, 6, 7	
Test	Repeat the following for each checkbox: 1. Create a new entry for testing 2. Use a webdriver to navigate to its webpage and click "Flag Incorrect Information" 3. Select a checkbox 4. Add a comment 5. Click "Submit" 6. Check that the correction is stored in the database correctly And similarly, check for the error case: 1. Create a new entry for testing 2. Use a webdriver to navigate to its webpage and click "Flag Incorrect Information" 3. Click "Submit" 4. Check that an error is displayed the correction is not stored in the database	

Number	7	
Name	Review Flags	
Summary	A database manager reviews the documents that have been flagged by users	
Priority	1	
Preconditions	The database manager is signed in	
	The database manager is on the “Management” tab	
Postconditions	The recent flags of a document are shown	
Primary Actor	Database manager	
Secondary Actors	N/A	
Trigger	The database manager clicks the “Review Flagged Entries” button	
Main Scenario	**Step**	**Action**
	1	The database manager clicks the “Review Flagged Entries” button
	2	The system redirects the database manager to the “Search” page, sorted by the number of flags
	3	The database manager selects a result and views its webpage
	4	The system displays a list of the entry’s most recent flags.
Extensions	**Step**	**Branching Action**
	2a	No results are found: The system displays “No results matched your search”
Open Issues	2, 6, 7	
Test	1. Create a mock entry with a single flag 2. Create a mock entry with 2 flags using the same mock copyright holder 3. Use a webdriver to navigate to the “Management” page and click “Review Flagged Entries” 4. Search for a document not present in the database 5. Ensure “No results matched your search” is displayed 6. Search for the copyright holder 7. Ensure that the mock with 2 flags appears above the mock with 1 flag 8. Visit the webpage for the mock with 1 flag 9. Ensure the correction matches the original flag	

Number	8	
Name	Add entry	
Summary	A database manager adds an entry to the database	
Priority	3	
Preconditions	The database manager is signed in	
	The database manager is on the “Management” tab	
Postconditions	The data is sent to the Data Formatting System	
Primary Actor	Database Manager	
Secondary Actors	N/A	
Trigger	The database manager clicks the “Add Document(s)” button	
Main Scenario	**Step**	**Action**
	1	The database manager clicks the “Add Document(s)” button
	2	The system provides the “Data Entry” pop-up
	3	The database manager selects the “Upload Document” field
	4	The system requests a file
	5	The database manager selects a file
	6	The system downloads the file temporarily
	7	The database manager inputs the title of the document
	8	The database manager inputs the copyright holder of the document
	9	The database manager inputs the copyright year of the document
	10	The database manager clicks the “Submit” button
	11	The document, transcript, and copyright information are added to the database
Extensions	**Step**	**Branching Action**
	6a	The document fails to download: The file upload box is highlighted and an error entitled “Upload failed” is displayed, but the “Data Entry” pop-up remains
Open Issues	3, 4, 6, 7	
Test	1. Create a mock document, transcript, and metadata file 2. Use a webdriver to navigate to the “Management” page 3. Click “Add Document(s)” 4. Fill out the text boxes using the mock data 5. Click “Submit” 6. Check the database for the new mock entry	

Number	9	
Name	Remove entry	
Summary	A database manager selects and removes an entry from the database	
Priority	2	
Preconditions	The database manager is signed in	
	The database manager is on the “Search” tab	
Postconditions	The selected entry is removed from the database	
Primary Actor	Database manager	
Secondary Actors	N/A	
Trigger	The database manager makes a search	
Main Scenario	**Step**	**Action**
	1	The database manager makes a search
	2	The system provides relevant entries
	3	The database manager selects any number of individual entries
	4	The database manager clicks the “Delete Selected Entries” button
	5	The system provides a confirmation pop-up containing the number of documents that will be removed
	6	The database manager clicks the “Confirm” button
	7	The system moves all entries to the “Recently Deleted” table
	8	After 24 hours, the System removes the entries from the database
Extensions	**Step**	**Branching Action**
	4a	The database manager clicks the “Delete All Matching Entries” button: The system includes all entries matching the search filters in the deletion
	8a	The database manager restores an entry before it is removed from the database: The entry is moved back out of the “Recently Deleted” table
Open Issues	5, 6, 7	
Test	1. Add a unique entry to the database 2. Search for that entry using a webdriver 3. Select the entry 4. Click “Delete Selected Entries” 5. Click “Confirm” 6. Check for the entry in the “Recently Deleted” table 7. After 24 hours, check for it again 1. Add a unique entry to the database 2. Search for that entry using a webdriver 3. Select the entry 4. Click “Delete Selected Entries” 5. Click “Confirm” 6. Check for the entry in the “Recently Deleted” table 7. Restore the entry	

	8. Check for the entry in the database and the “Recently Deleted” table

3. Non-Functional Requirements

These requirements describe various thresholds and procedures that must be carried out for the project to be considered operational. Each Non-Functional Requirement includes an associated priority, a field specifying what type of requirement it is (Product, Security, Organization), a description of the requirement, and the tests that must be performed to confirm the requirement has been met.

List of Non-Functional Requirements				
Number	**Priority (1-5)**	**Type**	**Description**	**Tests**
NFR1	3	Product Requirement	When a user searches through the database, it will return the result within 5 seconds.	Simulate 50 different users making a search request. Check that all requests are responded to in 5 seconds or less in 98% of tests.
NFR2	5	Security Requirement	The database will prevent the upload and deletion of files for non-administrators.	Attempt to add and delete files to the database as a regular user. Check that the database is not modified.
NFR3	3	Organization Requirement	Every pull request will be reviewed by at least one other team member.	Using GitHub actions, require both an automatic and manual review before any changes are committed to the main branch.
NFR4	4	Product Requirement	The web interface will display all elements properly for the newest versions of Chrome, Firefox, and Safari web browsers.	Manually log into the web interface on the three browsers, and ensure no UI elements are improperly displayed.
NFR5	2	Product Requirement	The system will successfully authenticate (log in) users within 2 seconds	Simulate a valid login for 50 different users. Ensure 49 of them are authenticated within 2 seconds.

List of Non-Functional Requirements				
			98% of the time.	
NFR6	3	Organization Requirement	The web interface will comply with WCAG 2 accessibility standards for contrast and text readability.	Test if all UI elements have both alt text for screen readers and at least a 4.5:1 contrast ratio.
NFR7	2	Product Requirement	When an upload or deletion action fails, the system will display an error message within 3 seconds of the failed action 98% of the time.	Simulate 50 invalid file upload and deletion attempts. Ensure that at least 49 attempts display the error message within 3 seconds.
NFR8	4	Product (Usability) Requirement	A first-time user will be able to create an account and enter a search query within 5 minutes, 90% of the time.	Usability Test: Have 10 people attempt to create an account and complete a valid search. Ensure at least 9 can complete in 5 minutes or less.
NFR9	5	Security Requirement	The system will not store passwords as plaintext.	Simulate the creation of 100 users. Ensure 0 of their passwords are stored plainly in the database (without hashing).
NFR10	5	Security Requirement	The system will sanitize user inputs for login/signup to prevent SQL injection.	Inject common malicious queries into the user input fields; ensure that there are no modifications to the database.

Appendix B. System Design Document (SDD)

Table of Contents

1 Introduction

Samuel Backer is a Professor of History at the University of Maine. He teaches about the mass culture movements that existed before the physical media that we have recorded. Backer teaches a course that concludes with the advent of film in America. In the early days of film, very few people thought to save their work, and in order to copyright a movie, filmmakers would have to submit a still image of each frame of the movie. In 1912, that law changed, allowing filmmakers to submit a description of their movie in order to have it copyrighted. These records form the bulk of our software database. We aim to digitize and organize the thousands of film descriptions released from the Library of Congress (LoC) into the public domain. Our work aims to provide a hub where researchers, film buffs, and historians can easily search and access these records, which have rarely had light shed on them.

1.1 Purpose of This Document

The purpose of this System Design Document (SDD) is to describe the technical and architecture design for the Discovering Early Hollywood System. This document describes how the system will be structured and implemented by turning the requirements mentioned in the System Requirements Specification (SRS) Document into design specifications. The SDD is intended to be a design guide for the project team members. It also serves as a reference document for the client and capstone instructors to make sure all project stakeholders have a shared understanding of the system's architecture, data design, and requirements matrix.

1.2 References

Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025) *System Requirements Specification Discovering Early Hollywood.*

Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025) *UI Design Document Discovering Early Hollywood.*

Fowler, M. (2003). *UML Distiller: A brief guide to the standard object modeling languages*. Addison-Wesley.

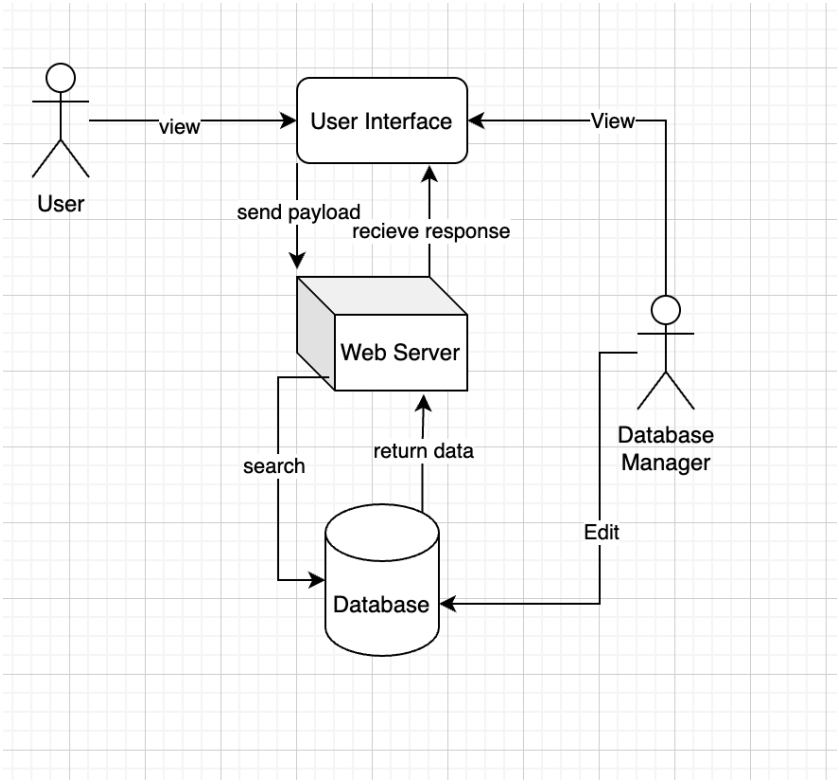
IBM. (n.d.). *IBM devops model architect*. https://www.ibm.com/docs/en/dma

Williams, L. (2013). *An introduction to software engineering*. Williams Publishing.

2 System Architecture

This section will outline the flow of data through the system on the scale of the external actors and systems it interfaces with (Section 2.1), as well as its components (Section 2.2). The diagrams in this section (especially in Section 2.2) are abstractions of this data flow, not necessarily accurate depictions of how and where data are stored while the system is executing.

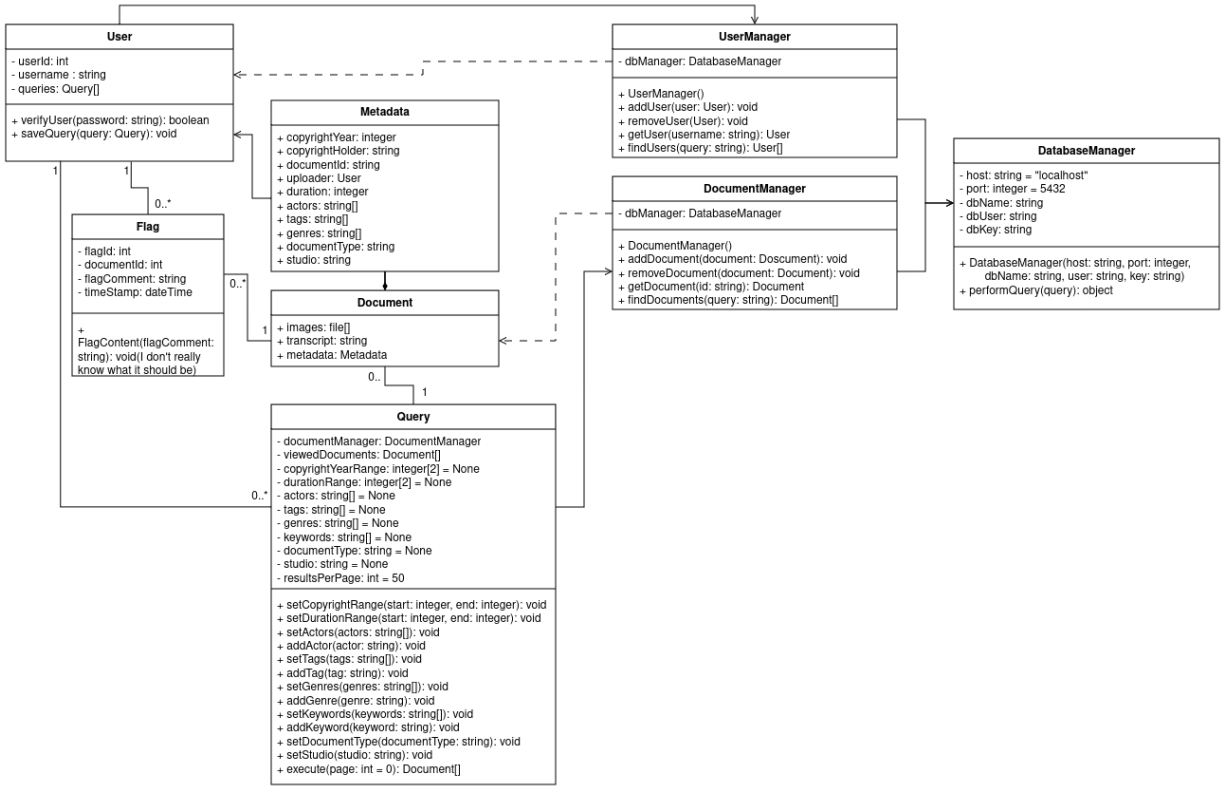
2.1 Architectural Design



The users will access our website through a web browser, where they will interact with the User Interface. The User Interface will consist of several pages providing an ensemble of functions defined in our SRS. User inputs will be used to activate functions on the frontend, which will either resolve immediately and change the state of the webpage, or collect data about the state of the webpage and then send payloads to the Python Flask backend using JavaScript. The webserver will either immediately perform some function on the data and return new data to update the User Interface, or use the data to make a query to the database, depending on the payload and the route by which it was accessed. This query will mainly be a search to find matching or related data. These data will then be returned to the backend, where they will be cleaned up, formatted, and returned to the frontend, which will update the User Interface to reflect the user's command.

The *database manager* is a separate, administrator type of user, who will be able to perform all actions that a user can, but will also be able to access and review flags, and will be able to use the User Interface delete or add entries to the database. The database manager will also be able to edit the database directly using the server management service we will be leveraging, however that falls outside the scope of the System Design Document.

2.2 Decomposition Description



The UML class diagram above shows the overall architecture of the Discovering Early Hollywood System; it contains the major components (e.g Users, Documents, Queries, Managers and all supporting classes) and the relationship between them. The goal of this diagram is to clarify how data flows through the system and which class is responsible for which part of the system's functionality.

Each component of the diagram has a specific responsibility. The User class represents the end user and stores authentication information and recently saved queries. The User class is associated with the Query class to represent ownership, though this is an abstraction since the database is responsible for storing these relationships. Document objects have transcript text, metadata, and optional images. Each Document will have 1 Metadata class that contains all descriptive information of a document (e.g copyright year, uploader, date uploaded, and other relevant information). The Flag class records user submitted errors in documents. The service layer is handled by the UserManager and DocumentManager. The UserManager handles adding,

removing, and querying users. Similarly, the DocumentManager handles adding, removing, and querying Documents. Both the UserManager and DocumentManager will interact with the DatabaseManager for storing and retrieving information. Lastly, the Query class connects to the DocumentManager to retrieve information based on its local variables. The DocumentManager calls the DatabaseManager to perform the retrieval of information.

3 Persistent Data Design

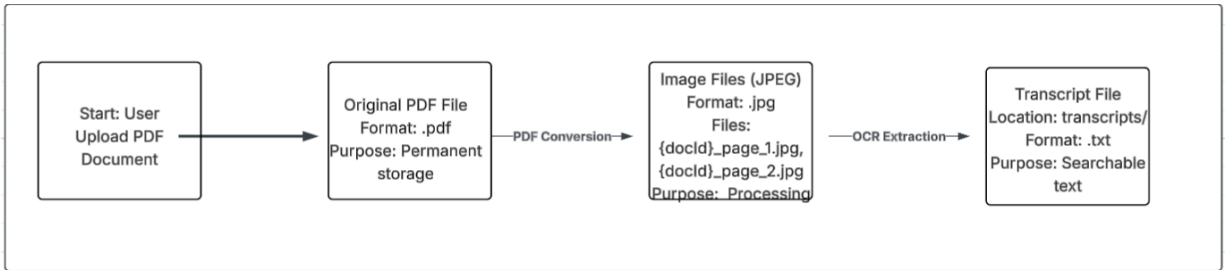
The purpose of this section is to outline the storage and schema of data and files used by the system. These data are contained within both a relational database (Section 3.1) and a filesystem (Section 3.2).

3.1 Database Descriptions

The system will use the PostgreSQL relational database management system to accommodate the strictly typed and formatted data provided by the LoC as well as the user, query, and flag information. All these data will be contained within the same database since they must reference each other. Below is an entity-relationship (ER) diagram of the database, in which rectangles represent entities (objects), ellipses represent attributes (the values within each object), and diamonds represent relationships (references between objects). Attributes which are underlined are called “primary keys” of their respective entity set or relationship, and are a unique identifier within that table. Plainly, a single primary key must map to at most one item in its associated table.

3.2 File Descriptions

The purpose of this section is to describe the files used by the system, including their structure, contents, and purpose. Each file type in the system is described to give an idea of how the system processes data. The descriptions specify the file formats, the fields in these files that are used, and how the files relate to each other.



PDF File

Name	**Data Type**	**Size**	**Description**
filename	string	Variable	Stores the name of the pdf file being uploaded
fileSize	int	8 bytes	Stores the size of the file
fileLocation	string	Variable	Stores the path to the pdf that the user is trying to upload to the server.

Plaintext File

Name	**Data Type**	**Size**	**Description**
filename	string	Variable	Stores the name of the text file
fileSize	int	8 bytes	Stores the size of the file
content	string	Variable	Stores the plaintext transcription of the file

JSON File

Name	**Data Type**	**Size**	**Description**
documen	string	20 bytes	Stores the identifier for the document

tId			
title	string	200 bytes	Stores the name of the movie
copyrighTYear	int	4 bytes	Stores the year that the copyright was filed for the movie
copyrighTHolder	string	50 bytes	Stores the name of the company/individuals who own the movie's copyright license.
duration	int	4 bytes	Stores the length of the movie (in minutes)
documenTType	string	25 bytes	Stores the type of media, specifying what information the document is displaying.
studio	string	50 bytes	Stores the name of the studio that produced the film.
actors	string array	Variable	Stores the names of the actors starring in the film.
genres	string array	Variable	Stores the names of the genres for the film, making it easy to filter.
tags	string array	Variable	Stores any additional information about the movie to increase search potential.

JSON File Example

```
{
 "documentId": "ahs72378274",
 "title": "Example Title",
 "copyrightYear": 1930,
 "copyrightHolder": "Holder",
 "duration": 110,
 "documentType": "short_film",
 "studio": "Movie_Studio",
 "actors": [
  "John Smith",
  "Jane Doe"
 ],
 "genres": [
  "comedy",
  "drama"
 ],
 "tags": [
  "award_winner",
  "silent_era"
 ]
}
```

4 Requirements Matrix

This section will outline and define how the different system components satisfy our functional requirements. This section is broken down into a tabular formatted table detailing each system component and which functional requirement it satisfies and/or helps satisfy.

Copyright/Document Database

Functional Requirement	Name
FR 2	Search Feature
FR 3	View Document
FR 4	Download Data
FR 6	Flag Incorrect/Unreadable Documents

Web Server

Functional Requirement	Name
FR 1	Sign In
FR 2	Search Feature
FR 4	Download Data
FR 5	Save Progress
FR 6	Flag Incorrect/Unreadable Documents

FR 7	Review Flags
FR 8	Add Entry

FR 7	Review Flags

Database Manager

Functional Requirement	Name
FR 2	Search Feature
FR 3	View Document
FR 4	Download Data
FR 6	Flag Incorrect/Unreadable Documents
FR 7	Review Flags
FR 8	Add Entry

Metadata schema

Functional Requirement	Name
FR 2	Search Feature
FR 4	Download Data
FR 8	Add Entry

User Account Database

Functional Requirement	Name
FR 1	Sign In
FR 5	Save Progress

Account System

Functional Requirement	Name
FR 1	Sign In
FR 5	Save Progress

Search Function

Functional Requirement	Name
FR 2	Search Feature

User Interface

Functional Requirement	Name
FR 1	Sign In
FR 3	View Document

Appendix C. User Interface Design Document (UIDD)

Table of Contents

1. Introduction

Samuel Backer is a Professor of History at the University of Maine. He teaches about the mass culture movements that existed before the physical media that we have recorded. Backer teaches a course that concludes with the advent of film in America. In the early days of film, very few people thought to save their work, and in order to copyright a movie, filmmakers would have to submit a still image of each frame of the movie. In 1912, that law changed, allowing filmmakers to submit a description of their movie in order to have it copyrighted. These records form the bulk of our software database. We aim to digitize and organize the thousands of film descriptions released from the Library of Congress (LoC) into the public domain. Our work aims to provide a hub where researchers, film buffs, and historians can easily search and access these records, which have rarely had light shed on them.

1.1 Purpose of This Document

The User Interface Design Document (UIDD) describes the format and layout of the User-facing portion of the codebase. It outlines the format of standard elements (buttons, input fields, text) as well as an overview of the content covered in each screen of the UI. The UIDD will make up the guidelines for the UI of the system throughout development, and should be used by both the team and the client to clarify the desired user experience.

1.2 References.

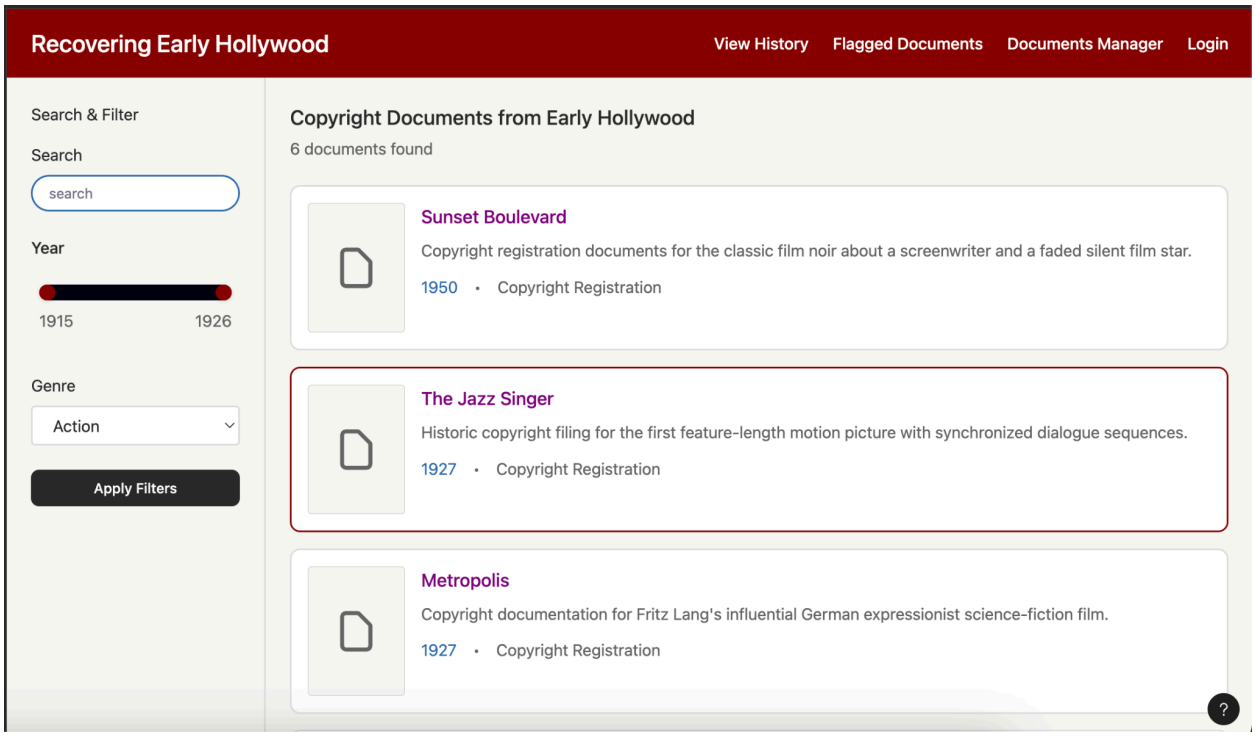
Figma. (2025). *Figma: the Collaborative Interface Design tool*. Figma. https://www.figma.com

Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025) *System Requirements Specification Discovering Early Hollywood.*

Dufour, X., Hillery, L., Lin, V., Storer, P., Thurston, C. (2025) *System Design Document Discovering Early Hollywood.*

2. User Interface Standards

The purpose of Section 2: User Interface Standards is to outline the design standards to be used across all the system's user interface. By defining the common screen layouts, shared interface components, navigation patterns, and general error handling, the interface will remain consistent and intuitive to follow. These standards will make sure that the users can easily understand how to interact with the system, regardless of which page they are on.



Screen 1: Home page

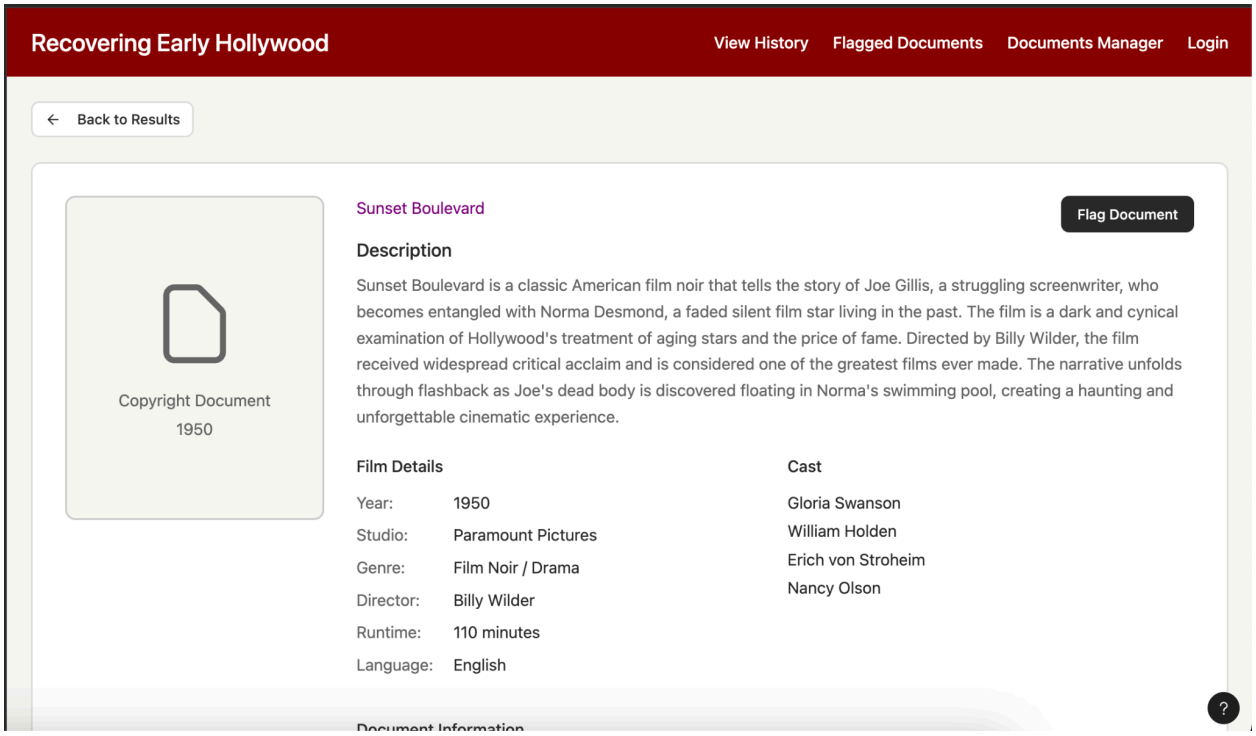
2.1 Overall screen layout

The screen will follow a layout of the header then the main content of the page. The main colors should be close these certain colors ; reddish nav bar: #7c170d, background: #f8f4f1, and white content background: #ffffff, main content text: #838181, and buttons: #2d2c2c, all other colors will be secondary and all could be changed later if needed. This layout consistency will allow users to easily navigate through this website knowing what each thing does. Now if there's a search bar it'll be different locations for different occasions, but the search bar should be easily noticeable and accessible.

The three major design principles we follow will be contrast, emphasis and repetition. For contrast one example is buttons, they will be different colors then the rest of the document preferably black (#2d2c2c). One example of emphasis is the important things like titles by the size and maybe the color. For repetition there will be repeating elements (e.g. colors, shapes) will be the same for consistency and unity.

2.2 Header

Header (*i.e nav bar; all screens*) - contains primary navigation links, and top level actions. The nav bar will be persistent across all pages and will be the main navigation for users to go from page to page. If the user is on a specific page that page will be highlighted on the nav bar.



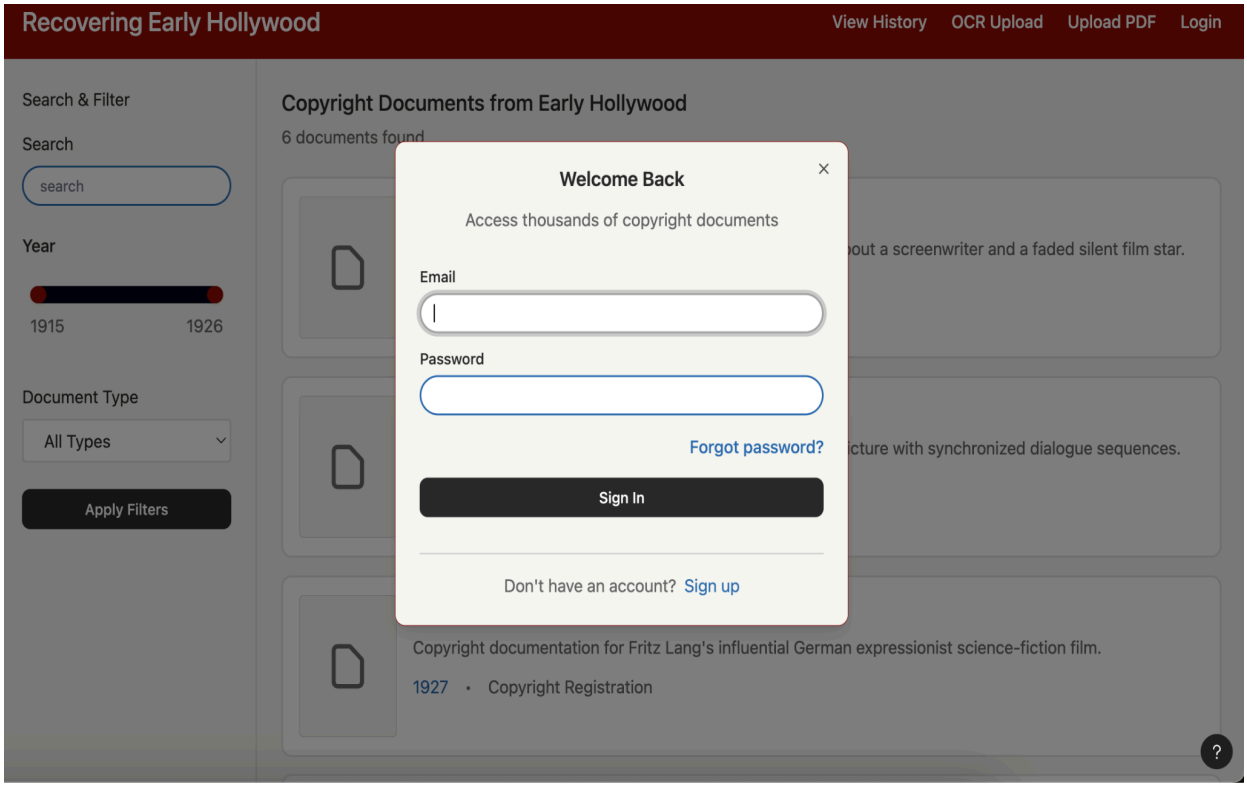
Screen 4: Document viewer

2.3 Content

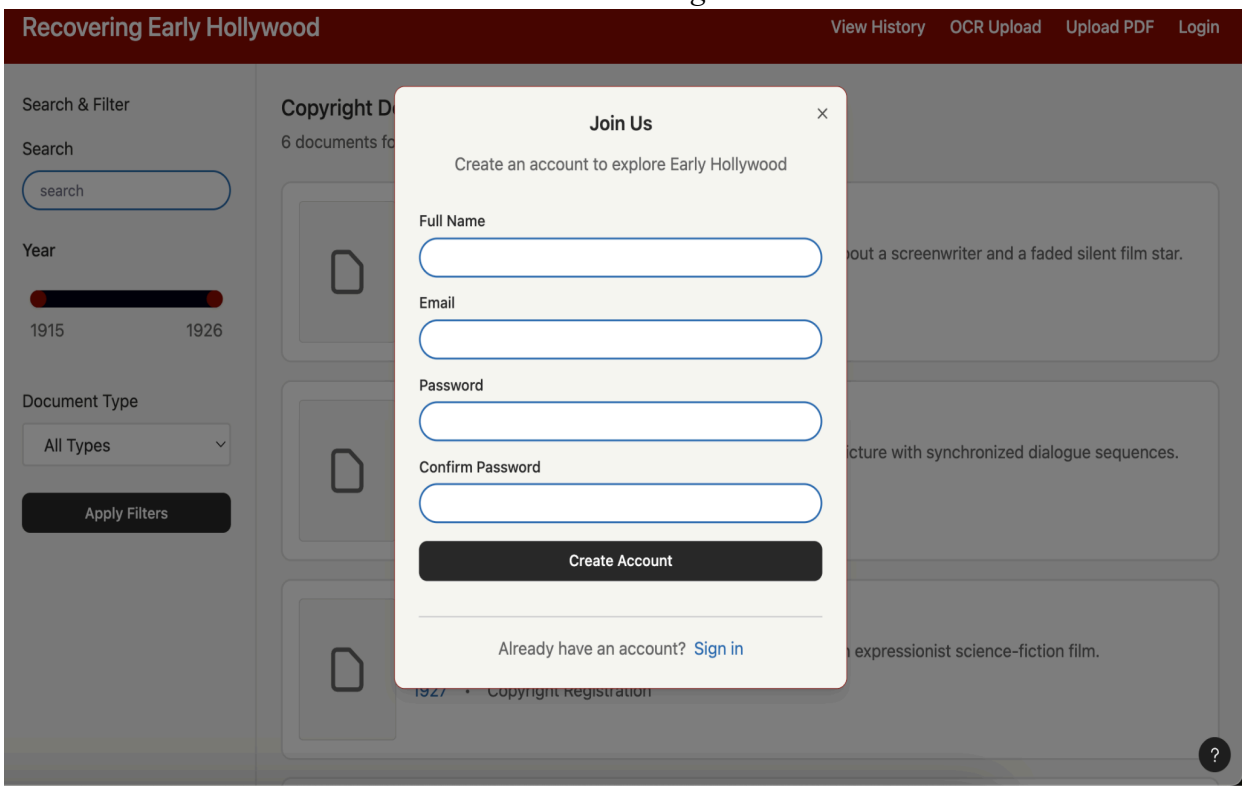
Document Cards (*Screen 1*)- The result cards will show an image on the left of the original pdf. It'll show the title of the card bolded, the Snippet of clean optical character recognition (OCR) content below the title and meta data underneath. When clicked, it'll bring you to the card's result content.

Document content (*Screen* 4) - This Result content page will consist of a title, more cleaned OCR content, underneath with meta data(e.g. Producer, actor, studio, year, etc.), and if you click the image all images related to that content will show. A flag error button and download content button will be on this page.

Search History Cards(*Screen 6*) - This card is for your past queries and will show what you searched for previously. When clicked, it'll take you to the past query.



Screen 2.1: Log In



Screen 2.2: Sign up

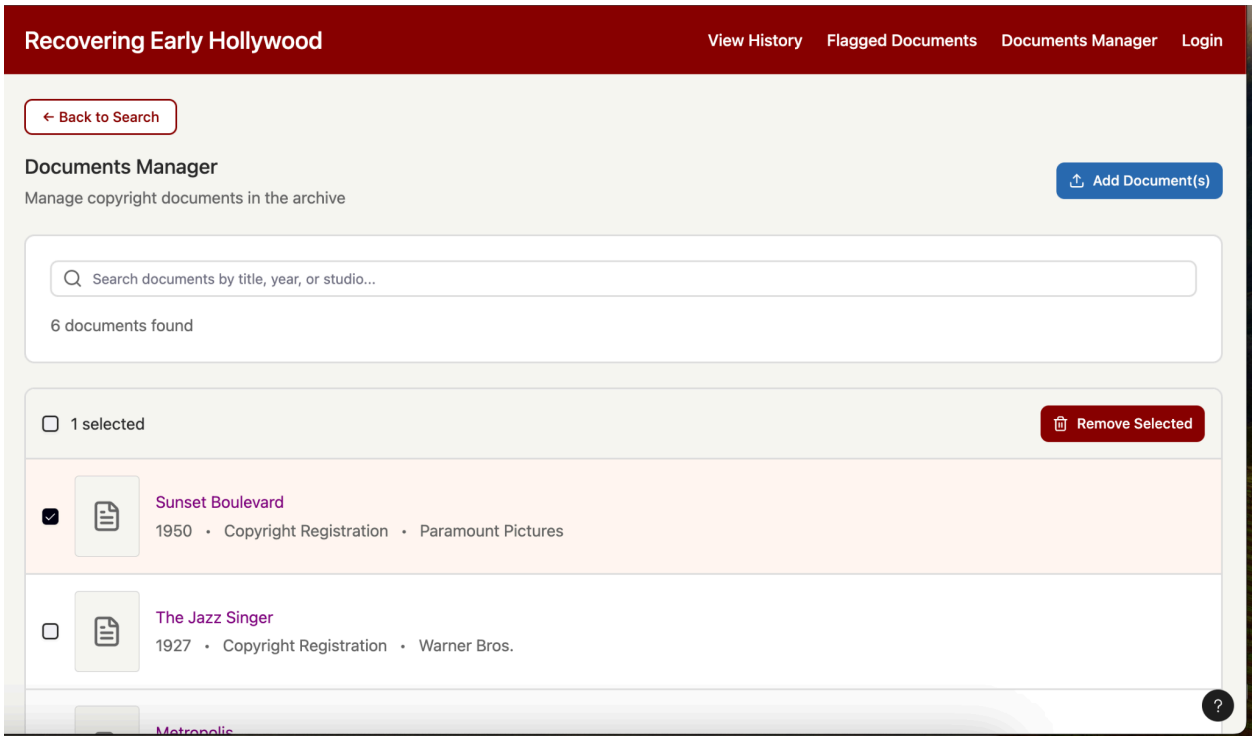
2.4 Forms

Search Bar (*Screen 1*) - Standard text input, search inputs for content, actors, director, producer, and studios. This form will be in the search section.

Sign in/sign up (*Screen 2*) - Standard text input, inputs for email and password. This will form in the sign in/sign up pop up.

Years filter (*Screen 1*) - Slider to select between years. This form will be in the search section.

Genre filter (*Screen 1*) - Check box selection to select multiple genres. This form will be in the search section. (The UI will be updated)



Screen 6: document manager

2.5 Buttons/clickables

Navigation (*All screens*) - Once clicked, It'll bring the user to the corresponding section.

Download button (*Screen 4*) - Once clicked, the download content button will show a pop up of what you are about to download.

Flag error button (*Screen 4*) - Once clicked, the Flag error button will show a pop up of a form for users to explain what error there is in the document content.

Confirmation button (*Screen 2 and more*) - A confirmation button to confirm your action such as log in, sign up, but could be used anywhere that needs confirmation.

Clear button(*Screen 6 and more*) - A destructive button that allows users to clear their search history, search terms or for any destructive purposes.

Meta data clickable in document content(*Screen 4*) - By clicking one of the meta data it'll filter by the genre, year, actor, director, or studio that was clicked on.

2.6 Error handling

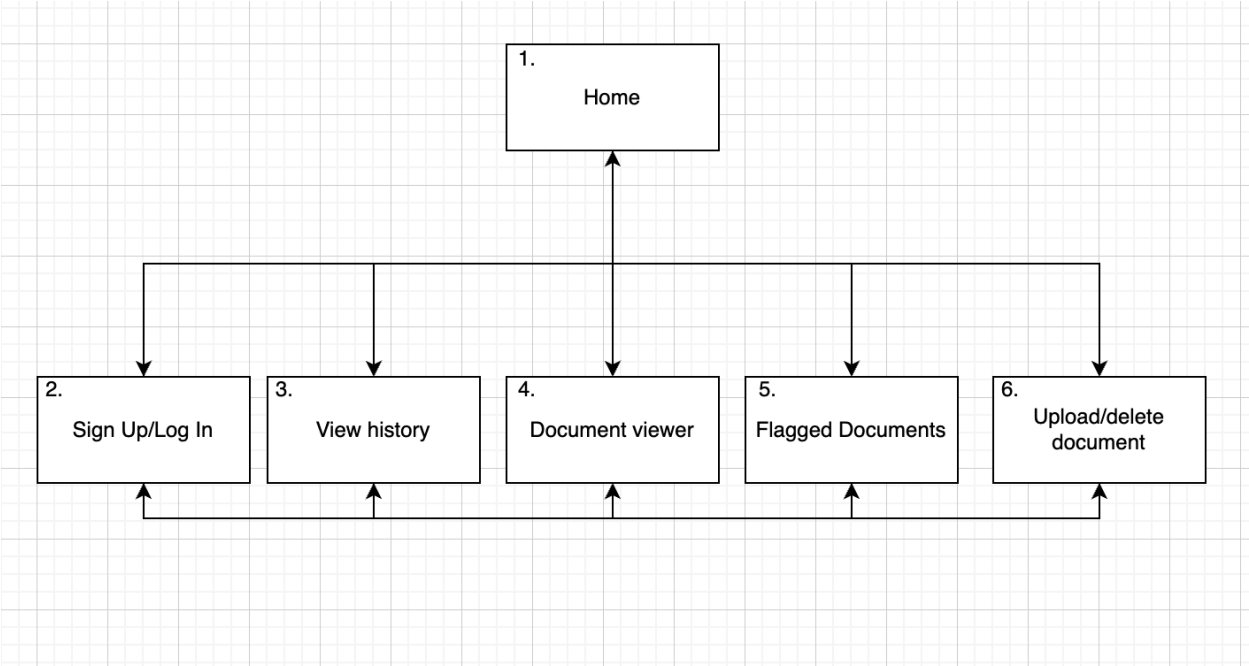
Document result error - If there was no results for search it'll show an error "No content found try a new search"

Pop up error - If the pop up has an error for any reason the pop up will show an error message.

General response error - If any other error shows the error message will be near the error site. For example, if a user has a sign up error the error message will be near the sign up area explaining why.

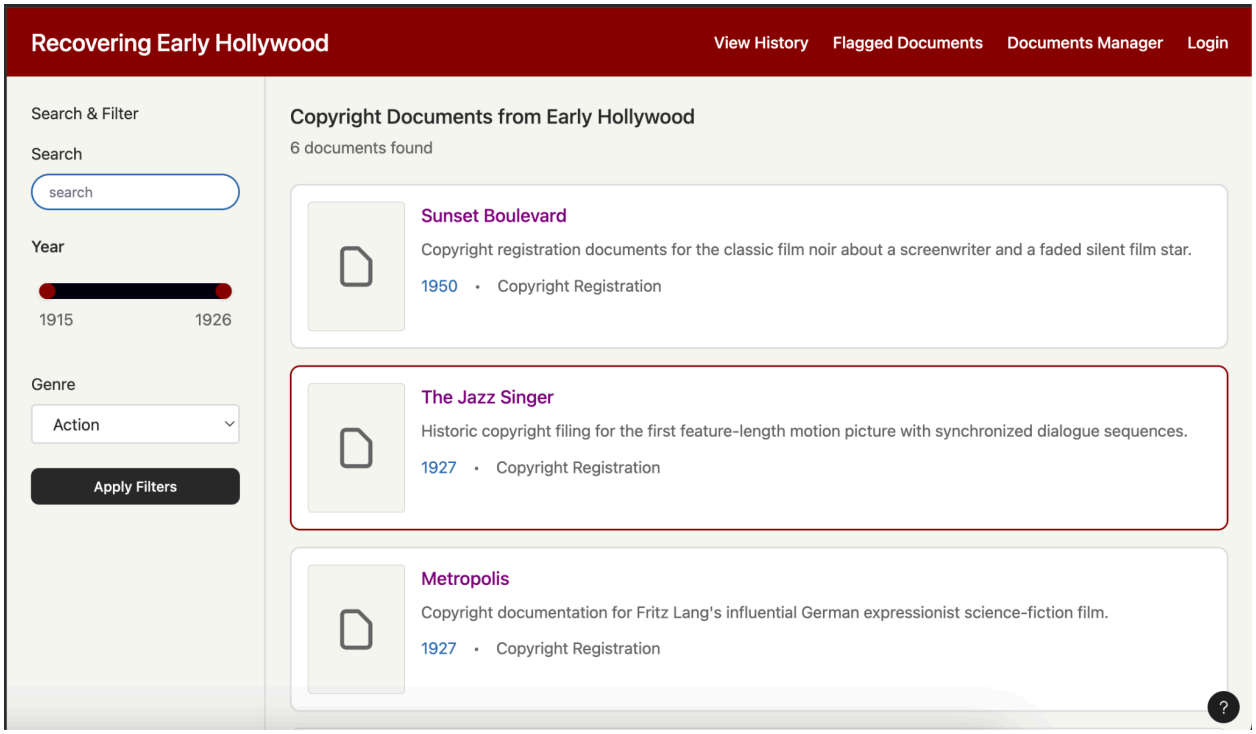
3. User Interface Walkthrough

The purpose of Section 3: User Interface Walkthrough is to incorporate the elements of Section 2 into the screens that make up the system. This section also includes a brief example of a user's behavior as they step through these screens. Our system consists of 6 main screens: a "Home" screen, a "Sign Up/Log In" screen, a "View History" screen, a "Document Viewer" screen, a "Flagged Documents" screen, and a "Upload/Delete Document" screen.



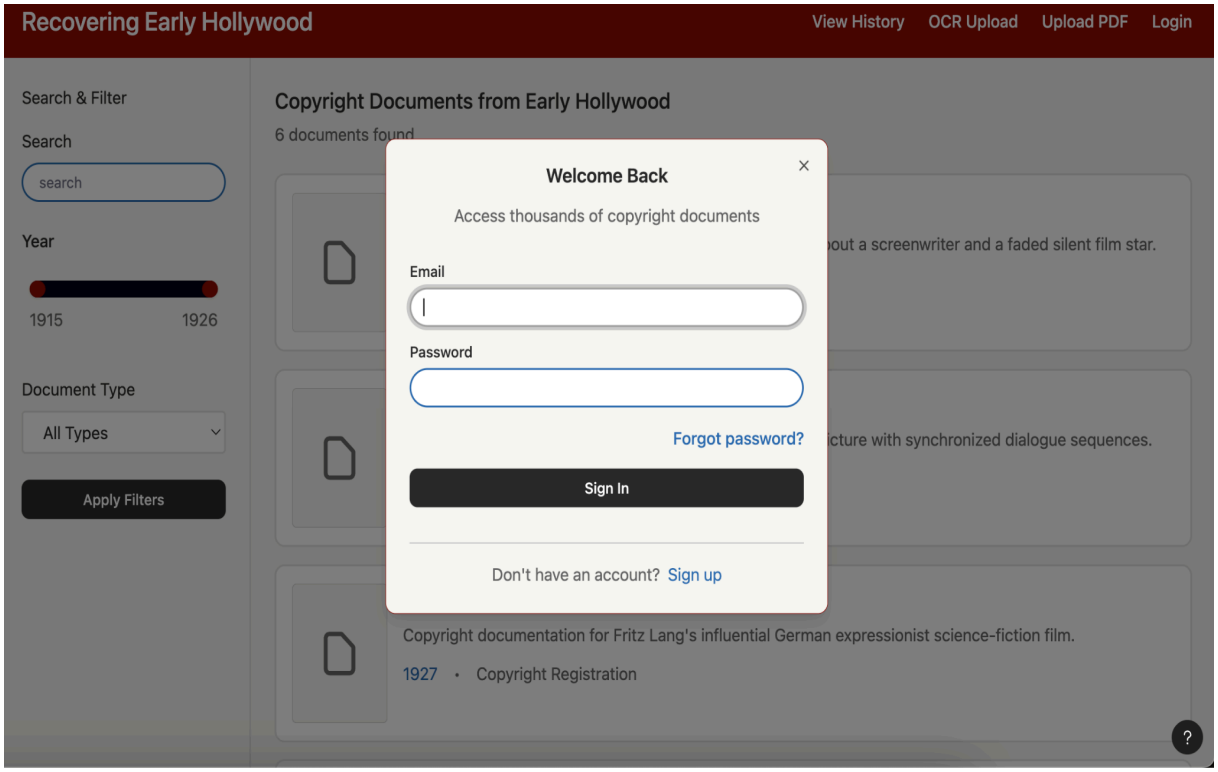
Navigation Diagram
(note: Navigation does not account for uses state some area like 6.upload and delete is only accessible only accessible by admin)

The navigation of the website will be very cyclical with each area accessible from every other area via the navigation bar at the top of the screen. The design of the website means you can travel to any area of the website from anywhere else and return.

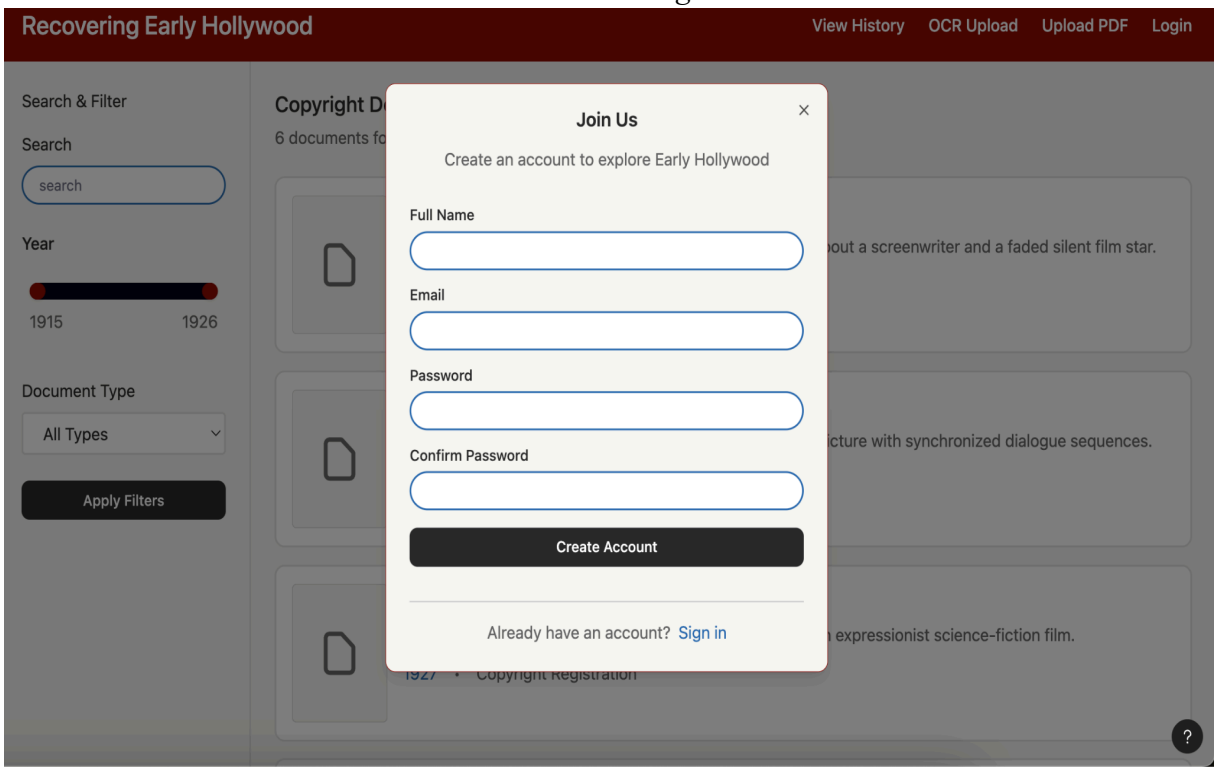


Screen 1: Home page

The user visits the site homepage. They are presented with multiple text inputs for keywords, tags, studios, and actors, sliders for selecting a date range and a duration range, and a dropdown to filter by the type of document present in the copyright. Upon filling these fields and clicking "Apply Filters" the main panel populates with relevant documents.

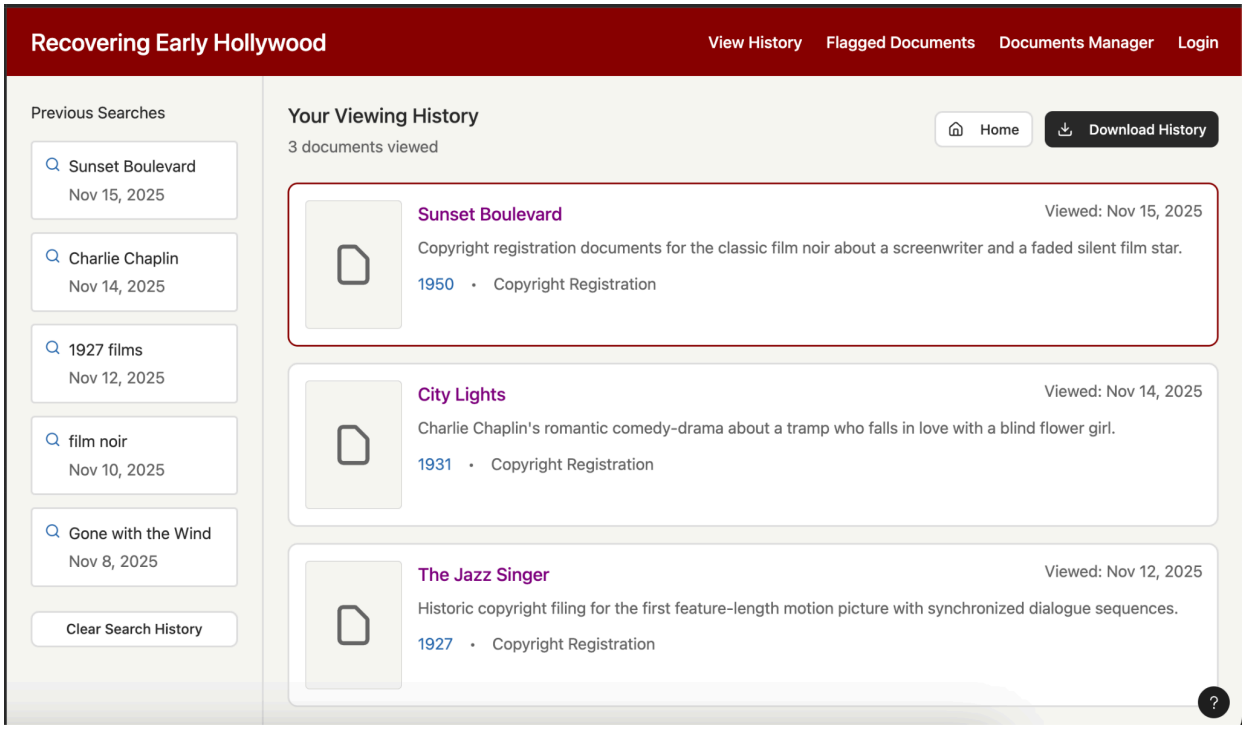


Screen 2.1: Log In



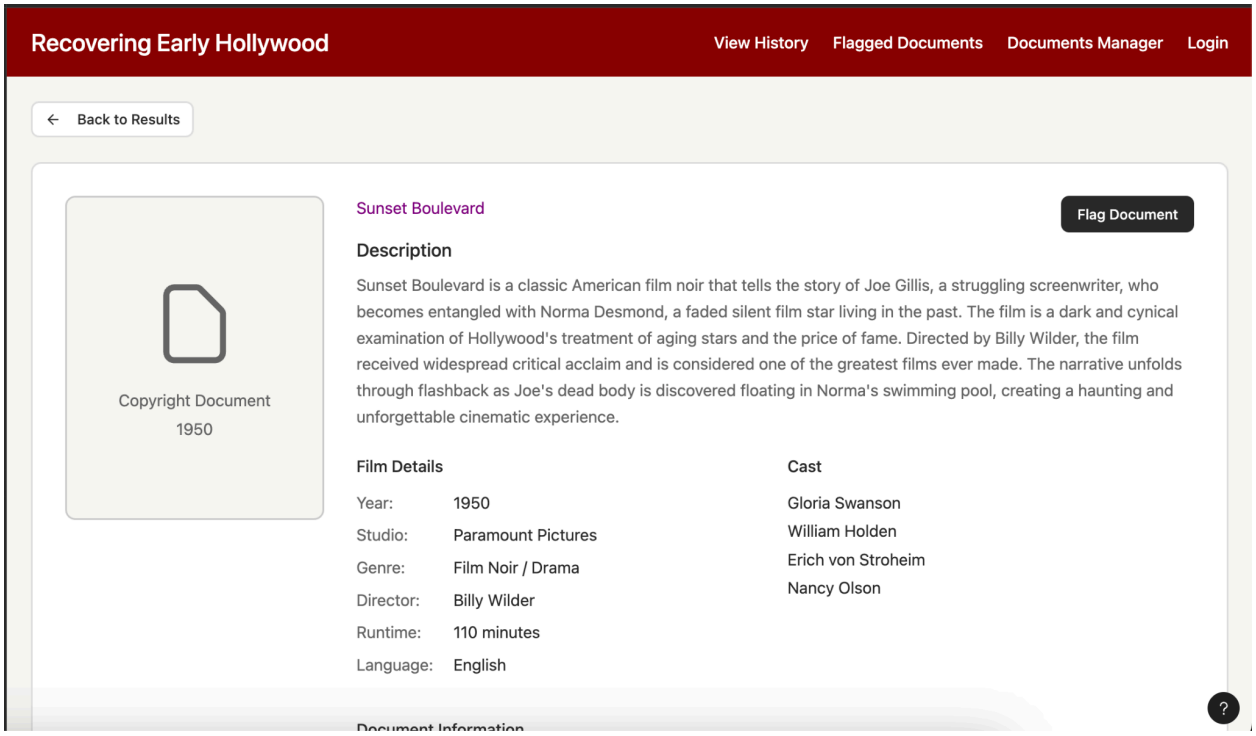
Screen 2.2: Sign up

Login/Sign up, the user will be presented with an option to Login when clicking on said button which will be prominent on view history if user is not logged in, there if they do not have an account they can click sign up in the login popup.



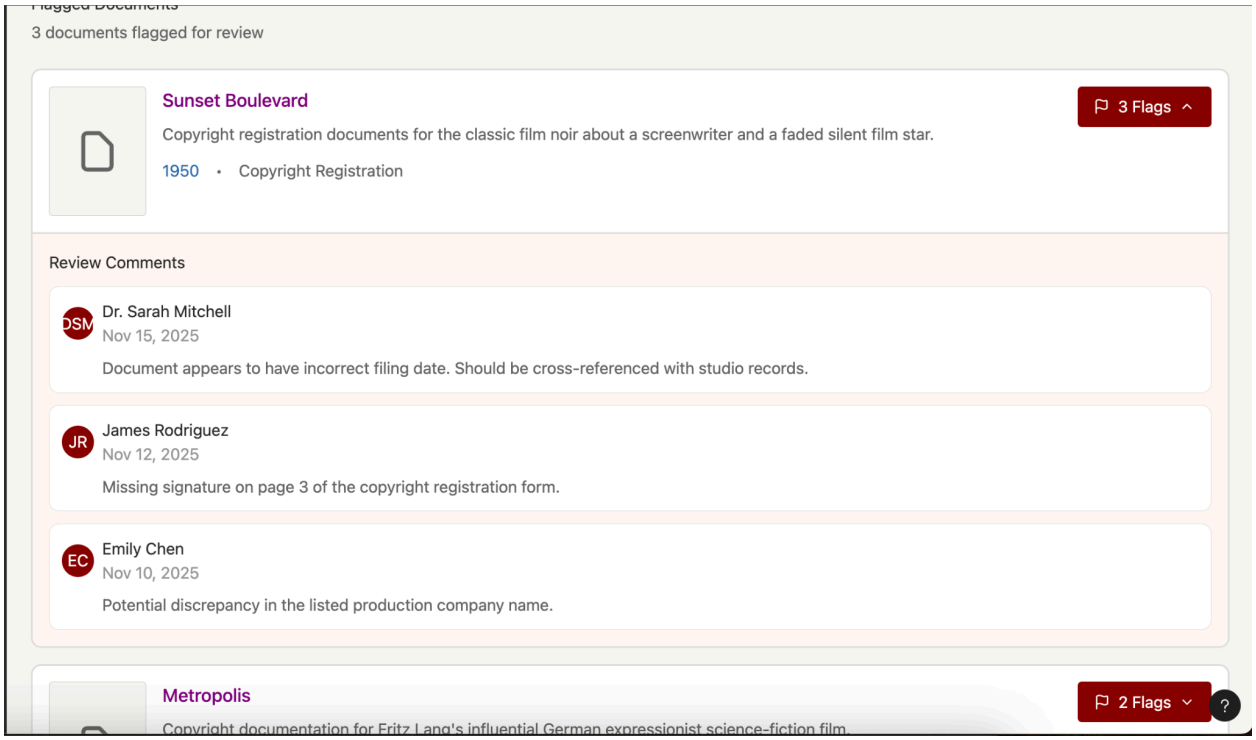
Screen 3: View history

The user selects "View History" in the navigation bar and is taken to Screen 3. They are presented with their past queries in the sidebar, with the documents viewed in each query contained in the main frame.



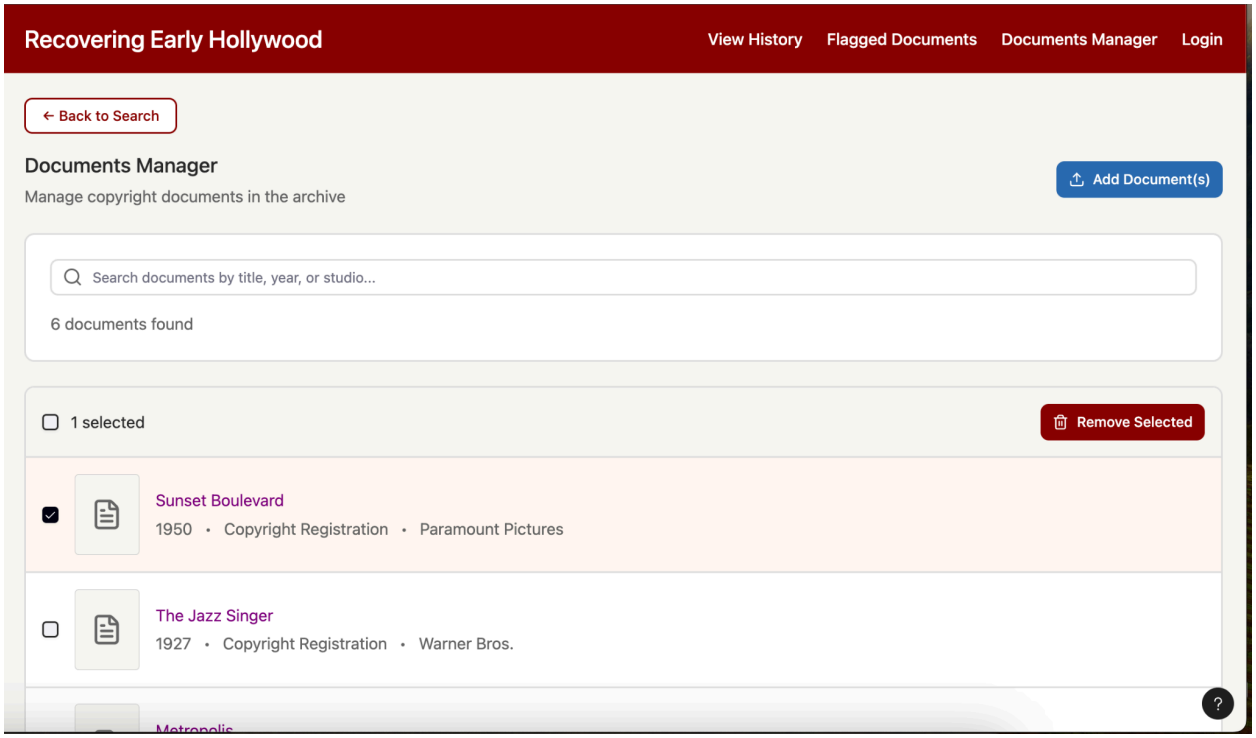
Screen 4: Document viewer

The user clicks on a document from their viewing history. They are presented with a view of the original document, the transcript of that document, and all of its metadata in a single frame. The user may click the "Download Document" button to download a ZIP file containing the document's metadata, transcript, and PDF file. The user may also click the "Flag Document" button, which triggers a pop-up requesting the error's location and any comments on the error.



Screen 5: Flagged document

The database manager clicks "Flagged Documents" in the navbar. The system provides an interface similar to Screen 1, where documents can be filtered and are ordered by the number of flags they have received.



Screen 6: document manager

The Database Manager (Admin) clicks “Documents Manager” in the navbar. The system provides a “Add Document(s)” button, which opens a pop-up window allowing them to either upload a ZIP containing documents or upload several files individually through the “Document,” “Metadata,” and “Transcript” file input fields. Additionally, the database manager may search for a document in the database using the search interface below and select documents to remove from the database. Upon clicking the “Remove Selected” button, a pop-up appears confirming the action. If a User attempts to navigate to this screen, they will receive HTTP error 403: Forbidden.

4. Data Validation

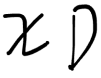
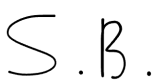
This section will detail all available datatypes for the user to enter. The table will show, in a tabular format, details about the data. Details will include its name, data type, limits, and formats.

Data Item		Data Type	Limits	Formats
Username		String	Min = 8 chars Max = 20 chars	“ExUser223”
Password		String	Min = 12 chars Max = 50 chars At least: 1 Uppercase Letter 1 Lowercase letter 1 Number 1 Special Character	“Password123!”
Query		String	Within SQL Syntax	“SELECT * FROM <table>”
Genre		String	Selected from dropdown	“Horror” “Comedy”
Year		Integer	Selected by Slider	“1921”
Actor(s)		String	Min = 5 chars Max = 20 chars	“Firstname Lastname”
Film Name		String	Min = 2 chars Max = 50 chars	“Sunset Boulevard”
Studio		String	Min = 4 chars Max = 30 chars	“Studio Ghibli”
Runtime		Float	Must include Hours, Minutes & Seconds	“2:31:13”
PDF		Hybrid Document	Below X size (size TBD)	Must be in .pdf format

Appendix E. Agreement between Customer and Contractor

All parties agree that the contents of this document are going to be held to the best of each parties ability. The requirements listed here will make up the majority of the work which will be produced by the development team in recompense for the completion of the capstone class requirement. The parties will work to the best of their ability to produce a system which testable satisfies these requirements.

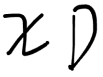
In the event that any addendums to the document, which redefine the scope of the project, the client will be notified of any changes and asked to confirm the changes to the document still align with their goal for the project before the changes can be added to the system. For minor changes to the document such as changes to the open issues section, the client will not be notified, as this is a living document which is subject to minor changes.

	Date:	Signature:
Vincent Lin	Dec. 8, 2025	
Caleb Thurston	Dec. 8, 2025	
Patrick Storer	Dec. 8, 2025	
Liam Hillery	Dec. 8, 2025	
Xander Dufour	Dec. 8, 2025	
Samuel Backer	Dec. 8, 2025	

Appendix F. Team Review Sign Off

All team members have reviewed this Critical Design Review (CDR) document and agree with all its content. The team acknowledges that the document accurately reflects the agreed-upon project scope and requirements at this development stage. Any minor issues shall be recorded among the team in the comment section.

This is a living document which is subject to changes or modifications. In the event that any addendums to the document, which redefine the scope or requirements of this project, will have to require every team member to agree and sign off.

Signatures:	Date:	Signature:
Vincent Lin	Dec. 8, 2025	
Caleb Thurston	Dec. 8, 2025	
Patrick Storer	Dec. 8, 2025	
Liam Hillery	Dec. 8, 2025	
Xander Dufour	Dec. 8, 2025	

Appendix G. Document Contributions

This section outlines the individual contribution made by each team member to the development of the document. It provides a record of who was responsible for each specific section for accountability throughout the documentation process. The following shows what they contributed and the percentage of work each member did.

Xander Dufour:
- Contributions: timeline
- Percentage: 18%

Liam Hillery:
- Contributions: CI/CD, Equipment
- Percentage: 18%

Vincent Lin:
- Contributions: Preface, Methods, Summary, Introduction, Purpose, Design, Revisions
- Percentage: 28%

Patrick Storer:
- Contributions: Abstract, Conclusion
- Percentage: 18%

Caleb Thurston:
- Contributions: All appendices, Part of the Bibliography, Deliverables
- Percentage: 18%