System Design Document v1.0

Discovering Early Hollywood

Samuel Backer

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

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.

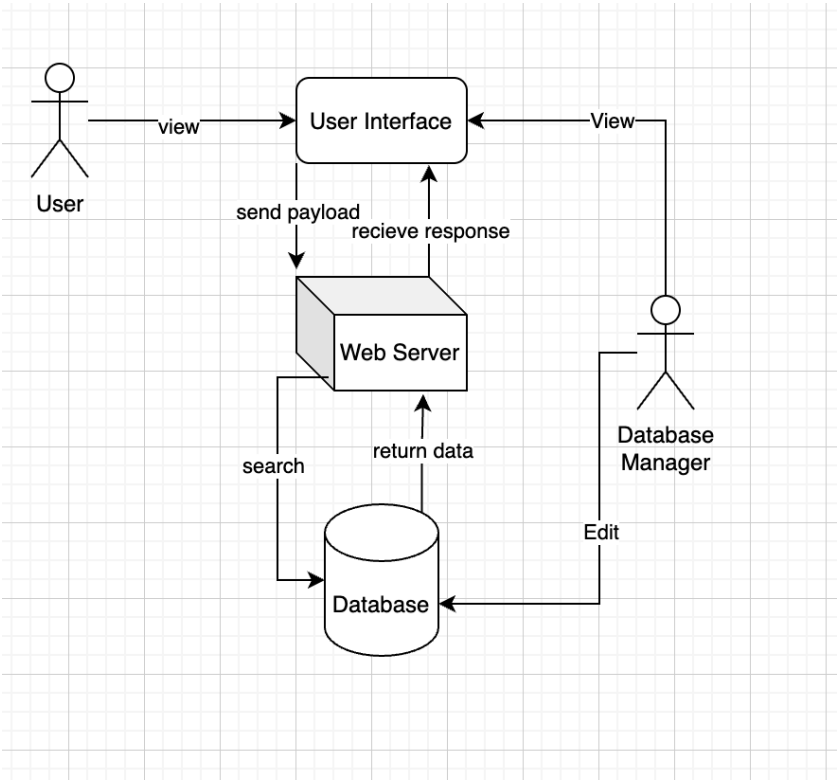
IBM. (n.d.). *IBM devops model architect*. https://www.ibm.com/docs/en/dma

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

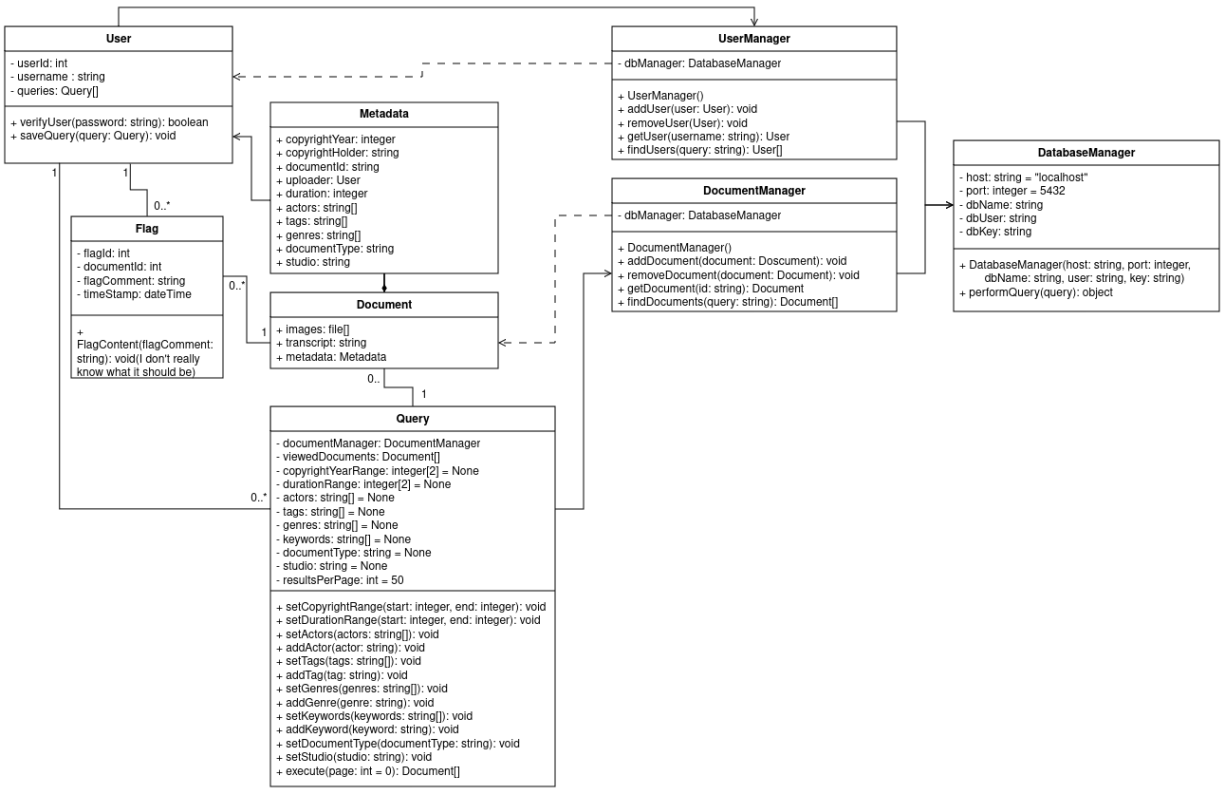


The users will access our website through a web browser, where they will interact with the User Interface. The User Interface will consist of several pages providing an ensemble of functions defined in our SRS. User inputs will be used to activate functions on the frontend, which will either resolve 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.

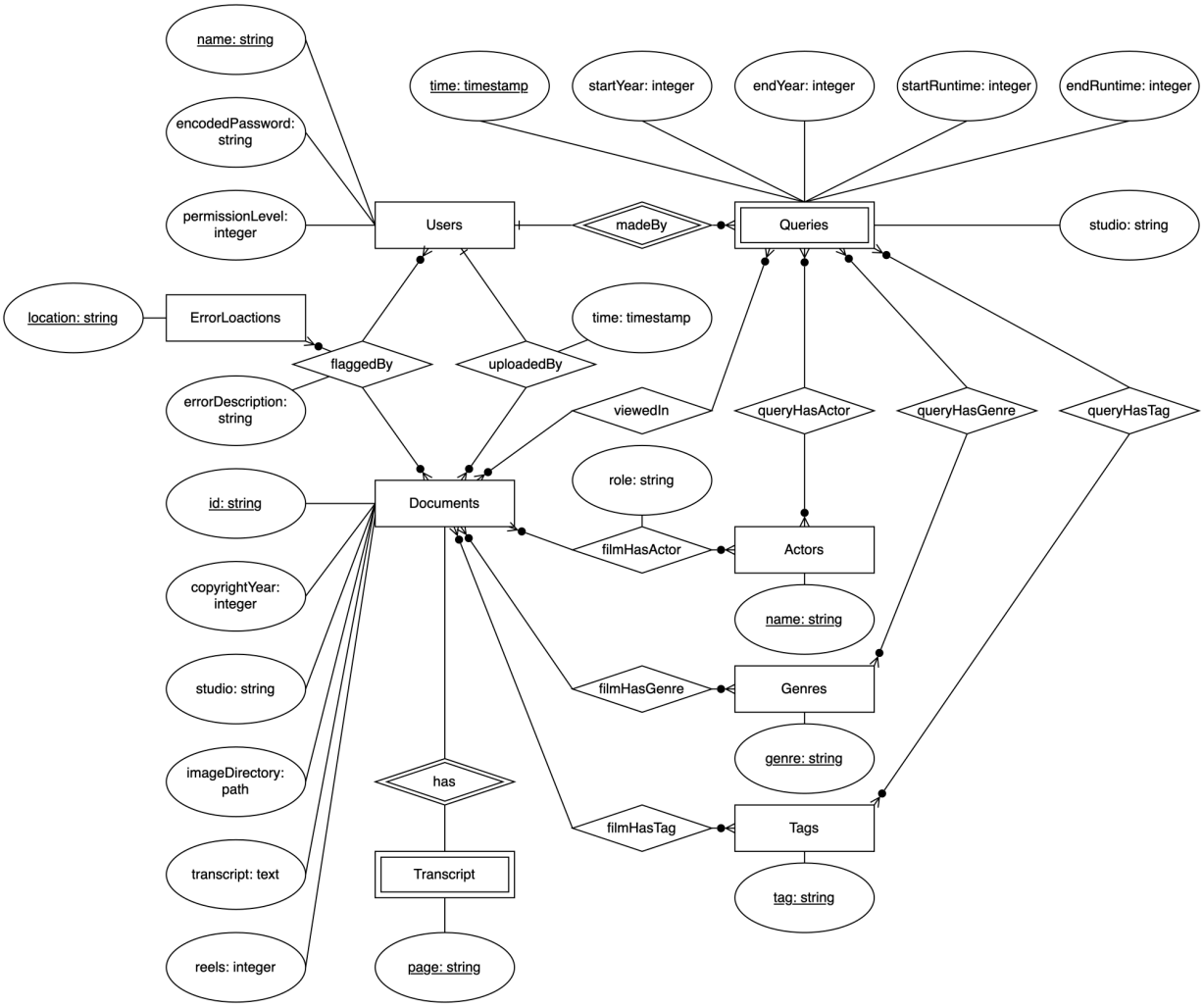
The DocumentManager calls the DatabaseManager to perform the retrieval of information.

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).

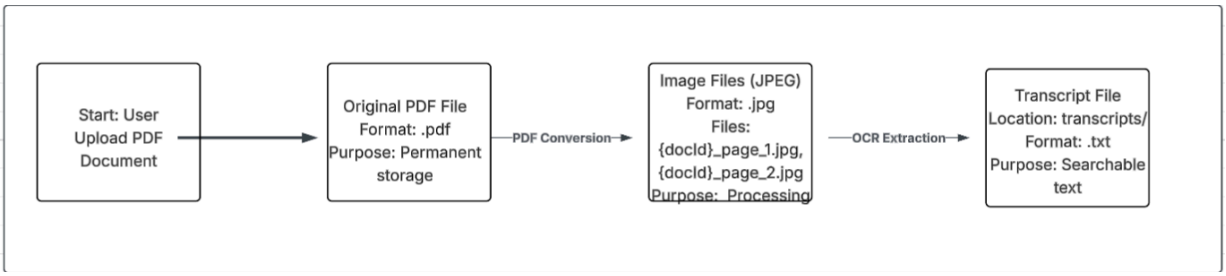
3.1 Database Descriptions

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**
documentId	string	20 bytes	Stores the identifier for the document
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
FR 7	Review Flags

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

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

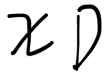
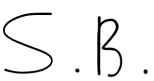
User Interface

Functional Requirement	Name
FR 1	Sign In
FR 3	View Document

Appendix A – 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.

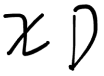
In the event that any addendums to the document, which redefine the scope of the project, the client will be notified of any changes and asked to confirm the changes to the document still align with their goal for the project before the changes can be added to the system. For minor changes to the document 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	Nov. 17, 2025	
Caleb Thurston	Nov. 17, 2025	
Patrick Storer	Nov. 17, 2025	
Liam Hillery	Nov. 17, 2025	
Xander Dufour	Nov. 17, 2025	
Samuel Backer	Nov. 17, 2025	

Appendix B – Team Review Sign-off

All team members have reviewed this Software Requirements Specification(SRS) 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	Nov. 17, 2025	
Caleb Thurston	Nov. 17, 2025	
Patrick Storer	Nov. 17, 2025	
Liam Hillery	Nov. 17, 2025	
Xander Dufour	Nov. 17, 2025	

Appendix C – 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: Section 2.1: Architectural Design diagram and explanation
- Percentage: 20%

Liam Hillery:

- Contributions: Section 2.2: Decomposition Description diagram and explanation, Section 3.1: Database Descriptions diagram and explanation
- Percentage: 20%

Vincent Lin:

- Contributions: Section 1: Introduction, Helped Liam on section 2.2 Decomposition Description (Very little of the diagram) diagram and explanation
- Percentage: 20%

Patrick Storer:

- Contributions: Section 3.2: File Descriptions diagram and explanation
- Percentage: 20%

Caleb Thurston:

- Contributions: Section 4: Requirements Matrix diagram and explanation
- Percentage: 20%