# System Requirements Specification 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

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.

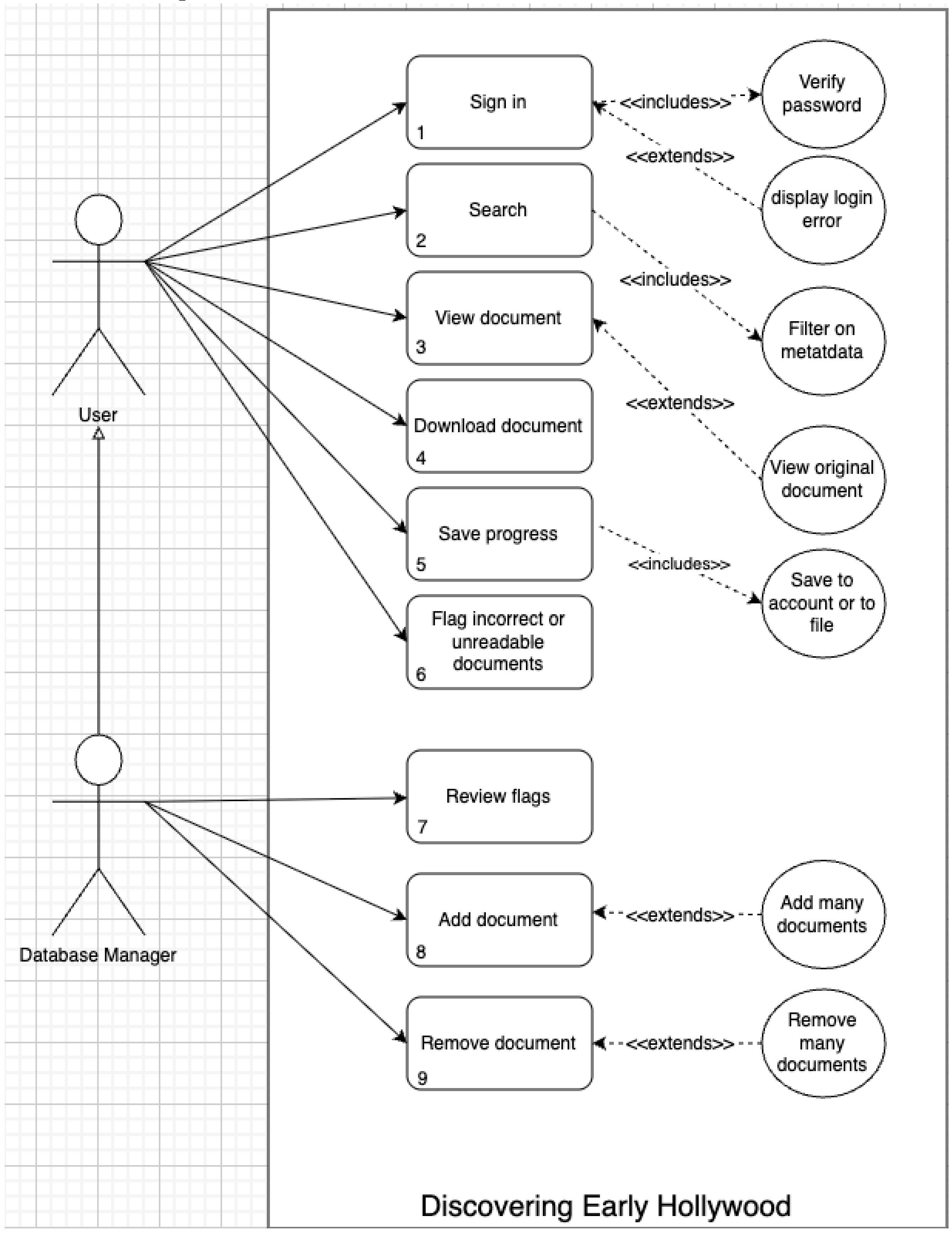
## 1.2. References

*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**:**<br>System Displays "Incorrect Username" dialog |
| | 6b | User's password fails to verify:<br>System displays "Incorrect Password" dialog |
| **Open Issues** | 6 | |
| **Tests** | 1. Attempt to log in to user account with correct username but incorrect password<br>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<br>2. When searching on a year, or a range of years, no entries outside that specification is returned<br>3. When returning to a search from an article, viewed articles are denoted and the list of entries remains the same.<br>4. An article which a user wants to find is easily findable if they know basic information about the film.<br>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 **:**<br>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<br>2. View a document that's known to be missing one type of information to confirm placeholder shows<br>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**:**<br>A missing file message with what's missing. |
|  | 3a | User clicks "Cancel":<br>The system cancels the transfer process. |
|  | 4a | The zip file fails to download:<br>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<br>2. Go through steps 1-4 using selenium for ~100 fake/real error filled documents to see if everything is working correctly.<br>3. Simulate different types error like internet to see if there is any corrupt file is downloaded<br>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<br>2) Search is the same on original and on return.<br>3) User is able to save many search progresses<br>4) Previously viewed entries are stored and denoted within save progress<br>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:<br>1. Create a new entry for testing<br>2. Use a webdriver to navigate to its webpage and click "Flag Incorrect Information"<br>3. Select a checkbox<br>4. Add a comment<br>5. Click "Submit"<br>6. Check that the correction is stored in the database correctly<br><br>And similarly, check for the error case:<br>1. Create a new entry for testing<br>2. Use a webdriver to navigate to its webpage and click "Flag Incorrect Information"<br>3. Click "Submit"<br>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<br>2. Create a mock entry with 2 flags using the same mock copyright holder<br>3. Use a webdriver to navigate to the “Management” page and click “Review Flagged Entries”<br>4. Search for a document not present in the database<br>5. Ensure “No results matched your search” is displayed<br>6. Search for the copyright holder<br>7. Ensure that the mock with 2 flags appears above the mock with 1 flag<br>8. Visit the webpage for the mock with 1 flag<br>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<br>2. Use a webdriver to navigate to the “Management” page<br>3. Click “Add Document(s)”<br>4. Fill out the text boxes using the mock data<br>5. Click “Submit”<br>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<br>2. Search for that entry using a webdriver<br>3. Select the entry<br>4. Click "Delete Selected Entries"<br>5. Click "Confirm"<br>6. Check for the entry in the "Recently Deleted" table<br>7. After 24 hours, check for it again<br><br>1. Add a unique entry to the database<br>2. Search for that entry using a webdriver<br>3. Select the entry<br>4. Click "Delete Selected Entries"<br>5. Click "Confirm"<br>6. Check for the entry in the "Recently Deleted" table<br>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 98% of the time. | Simulate a valid login for 50 different users. Ensure 49 of them are authenticated within 2 seconds. |

| List of Non-Functional Requirements | | | | |
|---|---|---|---|---|
| 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. |

## 4. User Interface

See “User Interface Design Document for Discovering Early Hollywood”.

## 5. Deliverables

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 | Delivery Method |
|---|---|---|
| System Requirements Specification Document | October 29th, 2025 | Paper Copy, Digital Document uploaded |
| System Design Document | November 17th, 2025 | Paper Copy, Digital Document uploaded |
| User Interface Design Document | December 3rd, 2025 | Paper Copy, Digital Document uploaded |
| User Manual | TBA | Paper Copy, Digital Document uploaded |
| Administrator Manual | TBA | Paper Copy, Digital Document uploaded |
| Biweekly Status Reports | Biweekly (As a whole, end of semester 2) | N/A (Only for in-class use) |
| Source Code | TBA | Uploaded to Git repository |
| Executable Program | TBA | Compiled by team, given as executable to client |
| Other Required Software | TBA | Given to client in guide format for product setup |

## 6. Open Issues

The open issues section represents areas where decision and specification are still pending. All issues will be resolved as the project progresses throughout the later development stages in collaboration with the client and project team.

1. **Copyright restrictions:**
   The client and the team have discussed copyright concerns, but a more detailed legal review is required. The team will have to confirm which materials can be legally stored in the database such as digitized versions of sheet music, synopses, and other early film content.
   Projection date: 10/29/25

2. **Genre classification:**
   The system requires a standardized genre classification framework for the early film content. The team and the clients must confirm the labeling rules and vocabulary for each genre tag.
   Projection date: TBA(third meeting with samuel backer) before 11/17/2025

3. **Automate metadata approach:**
   The team has to decide on which type AI model or algorithm to use for automation of genre tagging, actor/director extraction, and extraction of other important information. These choices would likely have to go through tests of model accuracy and processing time to see what fits best.
   Projection date: TBA(third meeting with Samuel Backer) before 11/17/2025

4. **Metadata schema:**
   A consistent metadata schema for scholarly and public use has to be defined. The client and the team needs to decide on whether to use existing schema for labeling metadata (e.g. dublin core) or have a custom one. This will heavily affect the database and search query design.
   Projection date: TBA(second meeting with Samuel Backer) before 11/7/2025

5. **Scope of search function:**
   The client has specified some basic search parameters (e.g. title, genre, time, publicist, actor, and director), but the full search features are still undefined. Some of the potential features may include: most viewed, most saved, and some more unique features that would depend on the metadata schema talked about in 6.4. This will have to be finalized with the team and client.
   Projection date: TBA(second meeting with Samuel Backer) before 11/7/2025

6. **User interface design:**
   The team will have to design UI/UX framework, layout, styling and themes(e.g. Historically inspired archival style or a more modern sleek look) in collaboration with clients to satisfy their need.
   Projection date: TBA(third/fourth meeting with Samuel Backer) before 12/3/2025

7. **Database**
   The team still will have to determine the specific database management system and data architecture. This includes which models to use(sql vs no sql), strategies for efficient search and establishing standards for scalability.
   Projection date: 11/17/2025

## 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. 7, 2025 | [illegible] |
| Caleb Thurston | Nov. 7, 2025 | [illegible] |
| Patrick Storer | Nov. 7, 2025 | [illegible] |
| Liam Hillery | Nov. 7, 2025 | William Hillery |
| Xander Dufour | Nov. 7, 2025 | X D |
| Samuel Backer | Nov. 7, 2025 | S. B. |

## 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. 7, 2025 | Vincent Lin |
| Caleb Thurston | Nov. 7, 2025 | [illegible] |
| Patrick Storer | Nov. 7, 2025 | [illegible] |
| Liam Hillery | Nov. 7, 2025 | William Hillery |
| Xander Dufour | Nov. 7, 2025 | X D |

## 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 1 - Introduction, Appendix A, Functional Requirement 2 & 5
- Percentage: 20%

Liam Hillery:
- Contributions: Section 2 - Functional Requirements, Functional Requirement 6-9
- Percentage: 20%

Vincent Lin:
- Contributions: Section 6 - Open Issues, Appendix B, Functional Requirement 4
- Percentage: 20%

Patrick Storer:
- Contributions: Section 3 - Non-Functional Requirements
- Percentage: 20%

Caleb Thurston:
- Contributions: Section 5 - Deliverables, Appendix C, Functional Requirement 1 & 3
- Percentage: 20%