**STD - Software Test Description**

TABLE OF CONTENTS

[**Introduction:** 2](#_Toc128245357)

[**Test Items:** 2](#_Toc128245358)

[**Test Deliverables:** 2](#_Toc128245359)

[**Test summary report:** 2](#_Toc128245360)

[**Test Environment:** 2](#_Toc128245361)

[**Test Procedure:** 3](#_Toc128245362)

[**Test Cases:** 3](#_Toc128245363)

[**Test Summary Report:** 4](#_Toc128245364)

[**Conclusion:** 4](#_Toc128245365)

STD - Software Test Description:

# **Introduction:**

The purpose of this document is to provide a Software Test Description (STD) for a Flask application that manages a movie database. The application is built using Flask, SQLAlchemy, and Flask-migrate, and it allows users to view, add, and delete movies and reviews, as well as connect with the website owner.

# **Test Items:**

The following items will be tested in the Flask application:

- Search function: search for movies by part of the title

- Home logo icon: This icon takes the user to the movie's homepage

- Alter function: alter name of movie

- Display function: display an uploaded image

- Add function: add a new movie

- Delete function: delete a movie and its associated reviews

- Connect function: Users can send the website owner a message

- Social media icon: The Google icon leads to the correct Google page

- Movie info function: the user can view and play trailers

- About us: The About Us page appears when the user clicks on the icon

# **Test Deliverables:**

The following deliverables will be produced during the testing process:

* Test plan: a document outlining the testing approach, including the testing scope, objectives, and resources required.
* Test cases: a set of detailed test cases covering all testable requirements of the application.
* Test logs: a record of all test results, including pass/fail status, test case ID, and any relevant comments or observations.

# **Test summary report:**

A document summarizing the test results, including any issues or defects identified and their severity.

# **Test Environment:**

The Flask application will be tested in the following environment:

Operating system: Windows 10

Browser: Google Chrome, version 88.0.4324.190

Python version: 3.8.5

SQLAlchemy version: 1.2.7

Flask-migrate version: 3.1.0

Testing tool: Selenium 4.9.1.0

# **Test Procedure:**

The following procedures will be followed during the testing process:

Review the functional requirements of the application and identify all testable requirements.

Develop a set of test cases for each testable requirement.

Execute each test case and record the results.

Report any defects or issues identified during testing.

Retest any issues or defects after they have been resolved.

Record the final test results in a test summary report.

# **Test Cases:**

The following test cases will be executed during the testing process:

# **Test Summary Report:**

The following summary report will be produced at the end of the testing process:

Total number of test cases executed: 10.

Total number of test cases passed: 10.

Total number of test cases failed: 0.

Issues/defects identified: None.

# **Conclusion:**

The Flask application has been thoroughly tested and all test cases have passed, indicating that the application is functioning as intended. The test logs and summary report provide a comprehensive record of the testing process and its outcomes, which can be used to evaluate the application's performance and identify any areas for improvement.

**SRS - Software Requirement Specification**

TABLE OF CONTENTS

[**Introduction:** 6](#_Toc128244971)

[**User Characteristics:** 6](#_Toc128244972)

[**System Architecture:** 7](#_Toc128244973)

[**Test Cases** 8](#_Toc128244974)

[**Appendices** 8](#_Toc128244975)

[**Glossary of Terms** 8](#_Toc128244976)

[**Reference Documents** 8](#_Toc128244977)

[**Conclusion:** 8](#_Toc128244978)

SRS - Software Requirement Specification:

# **Introduction:**

The purpose of this document is to outline the requirements for a web application that enables users to manage their movie collections, search for movies, and add reviews. This document includes:

* Overview of the system
* Stakeholder requirements
* User requirements
* Technology stack.

# **User Characteristics:**

The target audience for this application is movie enthusiasts and collectors who want to manage their movie collections effectively. Users of the application should have basic computer literacy and internet browsing skills.

2.1 Functional Requirements

The application shall allow users to:

* View a list of movies in their collection.
* Add new movies to the collection.
* Search for movies in the collection.
* View details of a specific movie.
* Add reviews for a specific movie.
* Connect with the application team.
* Check if the uploaded file is an image or video file before saving.
* Limit the size of uploaded files to 16 MB.
* Delete a movie from the collection and all reviews associated with it.

2.2 Non-Functional Requirements

The application shall:

* Be responsive and provide a consistent user experience across different devices and screen sizes.
* Be accessible to users with disabilities.
* Have a high level of availability and reliability.
* Be secure and protect user data from unauthorized access or modification.
* Be easy to install and deploy on a server.

2.3 User Interface Requirements

The application will have the following pages:

* Home Page
* Add Movie Page
* Movie Info Page
* Add Review Page
* Connect Page

The user interface should be easy to use, intuitive, and aesthetically pleasing.

2.4 User Interaction Requirements

The application shall allow users to interact with the system through a web browser. Users should be able to perform all required actions with minimal effort and without the need for advanced technical skills.

# **System Architecture:**

The application will be built using the following technology stack:

* Python 3.7 or higher
* Flask 2.0.2
* Flask\_SQLAlchemy 2.6.0
* Flask\_Migrate 3.1.0
* SQLite

3.1 Technical Requirements

The application shall be developed using industry-standard programming practices and follow the principles of clean code. The system should be modular, easy to maintain, and scalable.

3.2 Performance Requirements

The application should be able to handle a large number of concurrent users and provide a fast and responsive user experience. The system should be optimized for speed and performance.

3.3 Security Requirements

The application shall follow industry-standard security practices and protocols to ensure the safety and confidentiality of user data. The system should be protected from common web application security threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

3.4 Compliance Requirements

The application shall comply with all relevant data privacy regulations, including the General Data Protection Regulation (GDPR).

3.5 Acceptance Criteria

The acceptance criteria for the application shall be defined in the test cases section.

# **Test Cases**

The test cases shall include acceptance criteria for each functional requirement. The testing process should ensure that the application meets the requirements and functions as expected.

# **Appendices**

This section will include any additional information such as diagrams, charts, or tables that support the requirements.

# **Glossary of Terms**

This section will provide definitions for any technical terms or jargon used throughout the document.

# **Reference Documents**

This document refers to the Software Requirements Specification (SRS) Document, which outlines the overall requirements for the application.

# **Future Enhancements:**

The application can be enhanced in the following ways:

* Add user authentication and authorization to protect user data from unauthorized access or modification.
* Allow users to edit their reviews.
* Add a feature to recommend movies based on user preferences.
* Integrate with external APIs to provide more movie information.
* Improve the application's search functionality.

# **Conclusion:**

This document provides a comprehensive overview of the requirements for a web application that allows users to manage their movie collections, search for movies, and add reviews. By following the outlined requirements, the application will meet the needs.

STP - Software Test Plan

TABLE OF CONTENTS

[**Introduction:** 10](#_Toc128251767)

[**Test Data:** 10](#_Toc128251768)

[**Testing Scope:** 10](#_Toc128251769)

[**Testing Setup:** 10](#_Toc128251770)

[**Testing Requirements:** 10](#_Toc128251771)

[**Testing Strategy:** 11](#_Toc128251772)

[**Testing Methodology:** 11](#_Toc128251773)

[**Risk Assessment:** 11](#_Toc128251774)

[**Conclusion:** 11](#_Toc128251775)

STP - Software Test Plan:

# **Introduction:**

The Flask Movie App is an application designed to manage movies and user reviews. This STP document provides information on testing the application.

# **Test Data:**

The following data will be used for testing the Flask Movie App:

* + Movies: title, description, director, release\_year, filename, name\_actors, video.
  + Reviews: name, review\_text, rating\_movie, movie\_id.
  + Contact: name, message, email\_user.

# **Testing Scope:**

The testing scope for the Flask Movie App includes:

* + Functional testing: testing the basic functions of the application.
  + Integration testing: testing the interaction between the different modules of the application.
  + User interface testing: testing the user interface of the application.
  + Performance testing: testing the performance of the application under different loads.
  + Security testing: testing the security of the application.

# **Testing Setup:**

The testing setup for the Flask Movie App includes:

* + Python version 3.6 or higher.
  + Flask version 1.1.2 or higher.
  + SQLAlchemy version 1.3.20 or higher.
  + SQLite version 3 or higher.

# **Testing Requirements:**

The testing requirements for the Flask Movie App include:

* + Test cases for all functions of the application.
  + Test cases for all user interface elements.
  + Test cases for different loads on the application.
  + Test cases for different user roles (admin, user, guest).
  + Test cases for different types of input (valid, invalid, malicious).

# **Testing Strategy:**

The testing strategy for the Flask Movie App includes:

* + Black box testing: testing the functionality of the application without knowledge of the internal workings.
  + White box testing: testing the functionality of the application with knowledge of the internal workings.
  + Automated testing: testing the application using automated test scripts.
  + Manual testing: testing the application manually.
  + Regression testing: retesting the application after each change.
  + Exploratory testing: testing the application by exploring its features.

# **Testing Methodology:**

The testing methodology for the Flask Movie App includes:

* + Unit testing: testing individual components of the application.
  + Integration testing: testing the interaction between different components of the application.
  + System testing: testing the entire application as a system.
  + Acceptance testing: testing the application against user requirements.

# **Risk Assessment:**

The following risks have been identified for the Flask Movie App:

* + Security risks: risk of data breach, risk of unauthorized access.
  + Performance risks: risk of slow response time, risk of server overload.
  + Usability risks: risk of confusing user interface, risk of poor user experience.
  + Compatibility risks: risk of incompatibility with different operating systems, risk of incompatibility with different browsers.

The risks will be mitigated by:

* + Implementing security measures such as encryption and authentication.
  + Optimizing the application for better performance.
  + Improving the user interface based on user feedback.
  + Testing the application on different operating systems and browsers.

# **Conclusion:**

This STP document provides information on testing the Flask Movie App. The testing strategy, methodology, and requirements have been defined. Risks have been identified and mitigation measures proposed.