SCHOOL OF ENGINEERING AND PHYSICAL SCIENCES

Department of Computer Science

Software Requirement Specification (SRS)

For Media Search Web Application – part of CMP9134 Software Engineering Assessment

---

Author

Name: Do Minh Giang Vu

Student ID: 29100950

Date: 05/03/2025

TABLE OF CONTENTS

DEFINITIONS AND ACRONYMS

Term/Acronym	Definition
API	Application Programming Interface – a set of rules that allows software components to communicate.
CI/CD	Continuous Integration / Continuous Deployment – development practices that automate testing and deployment of code.
CRUD	Create, Read, Update, Delete – basic operations for persistent storage.
Docker	A containerisation platform used to package software and dependencies for consistent environments.
GDPR	General Data Protection Regulation – a European Union regulation for data protection and privacy.
HTTP/HTTPS	Hypertext Transfer Protocol / Secure – protocols used for communication between web servers and clients.
JSON	JavaScript Object Notation – a lightweight data-interchange format used in APIs.
Laravel	A PHP web application framework with expressive syntax, used to build the backend of the system.
MVC	Model-View-Controller – a software design pattern used to separate logic, UI, and data models.
MySQL/MariaDB	Relational database systems used to store structured data.
OAuth 2.0	A standard authorisation protocol used to allow secure delegated access.
Openverse API	A public API used to retrieve openly licensed media such as images and audio.
REST	Representational State Transfer – an architectural style for designing networked applications.
SPA	Single Page Application – a web application that loads a single HTML page and dynamically updates content.
UI/UX	User Interface / User Experience – design aspects that influence how users interact with the system.
Vue.js	A JavaScript framework used for building interactive frontend user interfaces.
JWT	JSON Web Token – a compact and self-contained way for securely transmitting information between parties.
XSS	Cross-site Scripting – a type of security vulnerability that allows attackers to inject malicious scripts.
SQL Injection	A code injection technique that could destroy a database if user inputs are not properly handled.
Version Control	A system for managing changes to source code, typically using Git.

INTRODUCTION

Product Scope

The media search web application aims to provide users with a streamlined platform to discover, view, and interact with openly-licensed media content. Built around the integration of the Openverse API, the system will allow authenticated users to search, filter, and manage multimedia resources effectively. The project supports the broader goal of increasing access to open media, especially for content creators, educators, and researchers who require legally shareable content.

The web application will feature user registration and login, secure data handling, and a user-friendly interface for search and media interaction. Technically speaking, the system will be built using Laravel (back-end) and Vue.js or Blade (front-end), ensuring modularity and scalability through clean architecture and modern development practices.

Product Value

The application addresses the growing need for reliable sources of open-license media. By providing a focused and efficient search interface, the platform enhances how users discover media content for various use such as creativity or education. Unlike general-purpose search engines, this system ensures that results are legally safe to use, and tailored filters help users quickly find suitable resources. Developers and researchers benefit from a clean, documented API integration, and the system's scalable design makes it a solid foundation for future enhancements.

Intended Audience

This system is intended for stakeholders involved in the development, deployment, and evaluation of the application, including:

- Developers and software engineers contributing to or extending the system.
- Project managers overseeing development progress.
- Academic supervisors or assessors evaluating the software artefact.
- Web users interested in using open-license media.

Intended Use

Users will interact with the system to perform the following core tasks:

- Register and authenticate securely.
- Search for open-license media using basic and advanced filters.
- View, play, and save selected media for later reference.
- Manage their search history.

Developers will use the specification to guide implementation and ensure alignment with project goals and software engineering principles. Academic assessors will refer to this document to verify that the artefact meets the defined requirements.

SYSTEM AND FUNCTIONAL REQUIREMENTS

System Overview

The system will provide:

- A secure and intuitive login and registration system.
- Integration with Openverse for fetching openly-licensed media.
- A user interface for searching, filtering, and displaying media content.
- Search history management for authenticated users.
- A scalable architecture supporting future enhancements.

Functional Requirements

User Authentication and Account Management

- **FR1.1**: The system shall allow new users to register an account.
- **FR1.2**: The system shall allow users to log in securely using email and password.
- **FR1.3**: The system shall store passwords securely using hashing algorithms.
- **FR1.4**: The system shall allow users to log out of their accounts.
- **FR1.5**: The system shall provide secure handling of personal data in compliance with GDPR.

Search and Media Interaction

- **FR2.1**: The system shall allow users to search for media using a single keyword.
- **FR2.2**: The system shall support advanced filtering, such as by type, licence, and source.
- **FR2.3**: The system shall allow users to sort search results by relevance or recency.
- **FR2.4**: The system shall display media content (images, audio, etc.) directly within the application.
- **FR2.5**: The system shall allow playback of audio files and viewing of image previews.

Search History Management

- **FR3.1**: The system shall allow logged-in users to save a search query.
- **FR3.2**: The system shall allow users to view their saved searches.
- **FR3.3**: The system shall allow users to delete saved searches.

System Administration and Development Features

- **FR4.1**: The system shall integrate the Openverse API to retrieve media content.
- **FR4.2**: The system shall follow a modular architecture using design patterns for scalability.
- **FR4.3**: The system shall support containerisation using Docker.
- **FR4.4**: The system shall implement automated tests covering core features.
- **FR4.5**: The system shall include an automated build and deployment workflow.

EXTERNAL INTERFACE REQUIREMENTS

User Interface Requirements

The application will offer a clean, intuitive, and responsive user interface, optimised for both desktop and mobile use. Key UI features include:

- A registration and login page with validation and feedback messages.
- A search interface with:
  - A text input field for keyword-based searches.
  - Filter dropdowns (e.g. licence type, media type, source).
  - A sort option (e.g. by relevance or recency).
- A results page displaying media in a grid or list view, with image/audio previews.
- A user dashboard with options to view and manage saved searches.
- Consistent design using a CSS framework (e.g. Bootstrap).
- Accessibility considerations following WCAG 2.1 where possible.

Hardware Interface Requirements

The application is designed to be accessed through a web browser on standard user devices. No specific hardware interface is required beyond:

- A modern web browser (Chrome, Firefox, Safari, or Edge).
- An internet connection.
- A server with Docker support for deployment and testing environments.

Software Interface Requirements

The software will interface with the following systems and libraries:

- **Openverse API**: External API for searching and retrieving open-license media. Communication is via HTTPS with RESTful endpoints (https://api.openverse.org/v1/).
- **Laravel Framework**: Handles back-end logic, API communication, user authentication, and data management.
- **Vue.js or Laravel Blade**: Renders the frontend user interface.
- **MySQL/MariaDB**: Stores user data, saved searches, and search history.
- **Docker**: Containerises the application for deployment.
- **PHPUnit / Pest**: For automated testing of backend functionality.
- **Git and GitHub**: For version control, collaborative development, and project management.

Communication Interface Requirements

The system will support the following communication interfaces:

- **RESTful communication with Openverse API** using HTTP(S).
- **OAuth 2.0 (optional)** for third-party authentication (if implemented).
- **Web-based form submission** for user interactions such as login, search, and saving/deleting history.
- **Internal HTTP requests** (AJAX via Axios or Fetch) for frontend-backend communication in a SPA-style layout (if Vue.js is used).

NON-FUNCTIONAL REQUIREMENTS (NFRS)

Security

- The system shall implement secure user authentication using hashed passwords (e.g. bcrypt or Argon2).
- All communication shall occur over HTTPS to prevent data interception.
- User input shall be validated and sanitised to prevent injection attacks (SQL, XSS).
- The system shall follow GDPR principles for storing and handling personal data.
- Access to user-specific data shall be restricted by authentication and authorisation mechanisms.

Capacity

- The system shall support at least 50 concurrent users in the development phase.
- The database shall be able to store user data and up to 1,000 saved searches initially.
- The system shall be scalable to support growth in user base and data volume.

Compatibility

- The application shall be compatible with the latest versions of Chrome, Firefox, Safari, and Microsoft Edge.
- The system shall support mobile responsiveness on iOS and Android platforms.
- The backend shall run on any Docker-supported Linux environment.

Reliability

- The system shall have an availability rate of 99% during standard operation.
- Failures (e.g. failed API responses) shall be handled gracefully with appropriate error messages.
- All critical operations (e.g. login, saving searches) shall be logged for monitoring and debugging.

Scalability

- The system architecture shall support horizontal scaling through containerisation with Docker.
- The backend shall use modular services and clean code practices to facilitate the addition of new features.
- The database structure shall allow indexing and optimisation for performance under higher loads.

Maintainability

- The system shall follow MVC principles and separation of concerns.
- Code shall be clean, well-documented, and structured into logical components.
- GitHub shall be used for version control with branches for features and bug fixes.
- Development shall follow Agile iterations, enabling continuous delivery and updates.

Usability

- The user interface shall be intuitive and accessible to non-technical users.
- The system shall use icons, tooltips, and placeholder text to guide user actions.
- Response time for search and data retrieval shall not exceed 3 seconds under normal load.

Other Non-Functional Requirements

- **Performance**: The system shall handle media preview without buffering for files under 5MB.
- **Environmental**: The project shall rely on cloud or local development environments without requiring physical infrastructure.
- **Regulatory**: The application shall comply with GDPR and the Openverse API's terms of use.