**CEBU INSTITUTE OF TECHNOLOGY**

**UNIVERSITY**

COLLEGE OF COMPUTER STUDIES

Software Requirements Specifications

for

AstroGlow

Fernandez, Cg M.

Mangoroban, Allen Luis S.

Cañal, John Gabriel

Change History

Table of Contents

Change History 2

Table of Contents 3

1. Introduction 4

1.1. Purpose 4

1.2. Scope 4

1.3. Definitions, Acronyms and Abbreviations 4

1.4. References 4

2. Overall Description 5

2.1. Product perspective 5

2.2. User characteristics 5

2.4. Constraints 5

2.5. Assumptions and dependencies 6

3. Specific Requirements 7

3.1. External interface requirements 7

3.1.1. Hardware interfaces 7

3.1.2. Software interfaces 7

3.1.3. Communications interfaces 7

3.2. Functional requirements 7

3.2.1 Web Application Features 7

3.2.2 Mobile Application Features 8

3.4 Non-functional requirements 8

Performance 8

Security 8

Reliability 8

# Introduction

AstroGlow is a music library application. It serves as a guide for stakeholders and developers, highlighting the app's objectives, features, and the overall process for its creation. The aim is to provide clarity on the project’s goals, timeline, and the value it will offer to users.

The project aims to develop a feature-rich music application available as both a web and mobile platform, enabling users to consolidate, manage, and enjoy their music collections seamlessly. Key features include music importation from personal devices, automatic library updates, song deletion, and offline listening for uninterrupted playback. Users can sign in effortlessly using Google Easy Sign-In, with the mobile app also offering biometric authentication for enhanced security. The platform will support favorite song management, an advanced search function, and a simple play button for quick access to music. Additionally, users will have the ability to log in or sign up with minimal effort, ensuring a streamlined experience across both web and mobile versions.

## Purpose

### Purpose of the SRS (Software Requirements Specification)

The purpose of this Software Requirements Specification (SRS) is to provide a detailed and clear description of the features, functionalities, and constraints of the AstroGlow music library application. This document outlines the requirements for both the web and mobile applications to ensure that the development team, stakeholders, and other parties have a shared understanding of what is to be built. It serves as a reference throughout the software development lifecycle to guide design, development, and testing processes. The SRS also helps in defining the scope, user needs, and project goals to ensure the final product aligns with stakeholder expectations.

### Intended Audience for the SRS

The intended audience for this SRS includes:

1. **Development Team:** Programmers and software developers who will be responsible for implementing the features and functionalities described in the SRS.
2. **End Users:** While not directly involved in the creation of the SRS, this document will indirectly affect users by ensuring the application meets their expectations for usability, functionality, and performance.
3. **Teachers:** Teachers who will use the application to manage student attendance and other tasks. They help ensure the app fits their needs in the classroom.

## Scope

AstroGlow will allow users to import, organize, and manage their music collections efficiently. The application will support music uploads, automatic library updates, song deletion, and offline listening for an uninterrupted experience. Users can mark favorite songs, search for tracks easily, and play music instantly using a dedicated play button. Authentication will be simplified through Google Easy Sign-In, with biometric login available on mobile for enhanced security.

The software will not function as a music streaming service or facilitate the purchase of songs. Instead, it focuses on managing locally stored music for personal use.

#### **Application, Benefits, and Objectives:**

AstroGlow is designed for music enthusiasts looking for a centralized, intuitive, and secure platform to organize their personal music libraries. The application will enhance the user experience by offering seamless access, offline playback, and effortless authentication. The goal is to provide a feature-rich, efficient, and user-friendly solution that simplifies music management across web and mobile devices.

This scope serves as a guiding framework for stakeholders and developers, ensuring clarity on project objectives, development process, and expected value to users.

## Definitions, Acronyms and Abbreviations

### **Terms, Acronyms, and Abbreviations**

1. **SRS (Software Requirements Specification)**: A complete specification and description of requirements of the software that need to be fulfilled for the successful development of the software system [1].
2. **API (Application Programming Interface)**: Stands for Application Programming Interface. In the context of APIs, the word "Application" refers to any software with a distinct function [2].
3. **UI (User Interface)**: The user interface (UI) is the point of human-computer interaction and communication in a device. This can include display screens, keyboards, a mouse, and the appearance of a desktop [3].
4. **UX (User Experience)**: The overall experience a user has while interacting with the software, focusing on ease of use, accessibility, and satisfaction [4].
5. **CRUD (Create, Read, Update, Delete)**: The basic operations used in databases and applications to manage data. These operations refer to adding, retrieving, modifying, and removing data [5].
6. **End Users**: The individuals who will directly interact with the software once it is deployed [6].
7. **Login**: The process of entering credentials (like a username and password) to access an application [7].
8. **Sign Up**: The process of creating a new user account by providing necessary information, such as a name, email, and password [8].
9. **Biometrics**: Technology that uses physical characteristics (like fingerprints or facial recognition) to identify users and provide access to the system [9].
10. **Offline Listening**: The ability to access and listen to music or content without needing an active internet connection [10].
11. **Import Music**: The feature that allows users to upload or add music files from their local device into the application [11].
12. **Favorite Songs**: A feature that lets users mark and store their preferred songs for quick access later [12].
13. **Search**: A feature that allows users to find songs, artists, or albums within the application by entering specific keywords or using filters [13].
14. **Profile**: A user-specific area in the application where personal information, preferences, and settings are stored [14].
15. **Play Button**: A control in the application that starts or resumes playback of a song or playlist [15].
16. **Offline Playback**: The ability to play music without an internet connection after it has been downloaded [16].
17. **Google Easy Sign-In**: A login feature that allows users to sign in to the application using their Google account with a single tap [17].
18. **Touch ID**: A biometric authentication method on mobile devices that uses a fingerprint to unlock apps or access secure content [18].
19. **Android Biometrics**: A system used on Android devices to authenticate users using fingerprints, facial recognition, or other biometric data [19].

## References

[1] GeeksforGeeks, “Software Requirement Specification (SRS) Format,” GeeksforGeeks, Jun. 18, 2020. <https://www.geeksforgeeks.org/software-requirement-specification-srs-format/>

[2] Amazon Web Services, “What is an API? - API Beginner’s Guide - AWS,” Amazon Web Services, Inc., 2024. <https://aws.amazon.com/what-is/api/>

[3] F. Churchville, “What is user interface (UI)? Definition from SearchAppArchitecture,” SearchAppArchitecture, 2021. <https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI>

[4] Interaction Design Foundation, “What is User Experience (UX) Design?,” Interaction Design Foundation, 2021. <https://www.interaction-design.org/literature/topics/ux-design>

[5] S. M. Sinha, “CRUD Operations in SQL,” GeeksforGeeks, Oct. 10, 2018. <https://www.geeksforgeeks.org/crud-operations-in-sql/>

[6] Stack Overflow, “What is an End User in Software Development?” Stack Overflow, 2020. <https://stackoverflow.com/questions/41071532/what-is-an-end-user-in-software-development>

[7] W. F. Stenning, “What is Login Authentication in Web Development?” DigitalOcean, 2021. <https://www.digitalocean.com/community/tutorials>

[8] “What is the Process of Sign-Up in Software?” Techopedia, 2022. <https://www.techopedia.com/definition/34217/sign-up>

[9] TechCrunch, “How Biometric Authentication Works,” TechCrunch, 2022. <https://techcrunch.com/biometrics>

[10] “Offline Music Playback on Android,” Android Developers, 2021. <https://developer.android.com/guide/topics/media>

[11] “How to Import Music into Android Devices,” Android Authority, 2021. <https://www.androidauthority.com/import-music>

[12] “How to Add Songs to Favorite on Android,” Android Central, 2021. <https://www.androidcentral.com/favorite-songs>

[13] “Search Functionality in Mobile Apps,” Stack Overflow, 2021. <https://stackoverflow.com/questions/17879761/creating-a-search-feature>

[14] "How to Create a User Profile in Android," Android Developers, 2020. <https://developer.android.com/training/data-storage/shared-preferences>

[15] G. Miller, “Play Button and UI Components in Android,” Android Developers, 2022. <https://developer.android.com/guide/topics/ui>

[16] "Offline Music Playback in Apps," TechRadar, 2021. <https://www.techradar.com/news/offline-music-apps>

[17] Google, “Google Easy Sign-In for Android,” Google Developers, 2021. <https://developers.google.com/identity/sign-in/android>

[18] Apple Support, “Touch ID Setup for iPhone and iPad,” Apple, 2021. <https://support.apple.com/en-us/HT201371>

[19] Android Developers, “Biometric Authentication on Android,” Android Developers, 2021. <https://developer.android.com/training/sign-in/biometric>

‌

‌

# Overall Description

AstroGlow is a feature-packed music library application designed to give you full control over your music collection while providing an immersive listening experience. At its core, AstroGlow revolves around a beautifully organized music library, where users can store, manage, and play their favorite music effortlessly. Whether you're a casual listener or a dedicated audiophile, AstroGlow ensures that your music is always at your fingertips.

## Product perspective

### 2.1.1. User Authentication & Security

* **Transaction 1.1:** User sign-up via email or Google Easy Sign-In
* **Transaction 1.2:** User login with email, Google, or biometric authentication (mobile)
* **Transaction 1.3:** Password recovery and account management

### 2.1.2. Music Library Management

* **Transaction 2.1:** Import music from the user’s device
* **Transaction 2.2:** Upload new music files to the library
* **Transaction 2.3:** Automatic updates to reflect file changes
* **Transaction 2.4:** Delete unwanted songs from the library

### 2.1.3. Playback & Offline Listening

* **Transaction 3.1:** Play music using the in-app player
* **Transaction 3.2:** Download songs for offline playback
* **Transaction 3.3:** Adjust playback controls (pause, skip, shuffle, repeat)

### 2.1.4. Favorite Songs & Search

* **Transaction 4.1:** Mark songs as favorites
* **Transaction 4.2:** Remove songs from the favorites list
* **Transaction 4.3:** Use the advanced search bar to locate songs

### 2.1.5. User Interface & Navigation

* **Transaction 5.1:** Display and manage the music library interface
* **Transaction 5.2:** Provide an intuitive and responsive design for web and mobile
* **Transaction 5.3:** Ensure seamless navigation between modules

## User characteristics

### **2.2.1.** **User Types and Roles in the System**

**User**

1. **Role**: The users who access the system to sign in, view their music library, and interact with the system’s features.

### 2.2.2. Privileges

1. **Import Music**: Users can import music from their devices, putting their entire music library into one convenient space. They can upload new music files to the platform from their devices, and the system will automatically update the music files in their library when changes occur. Users can also delete songs they no longer wish to keep in their library.
2. **Offline Listening**: Users can download their favorite songs for offline playback, ensuring uninterrupted enjoyment even without an internet connection.
3. **Google Easy Sign-In**: Users can sign in quickly and effortlessly with a single tap using Google Easy Sign-In, avoiding the need for complex account setups.
4. **Favorite Songs**: Users can mark and organize their favorite songs by clicking the heart icon next to any track. They can also use an advanced search bar to quickly locate songs. Additionally, users can remove songs from their favorites if they no longer wish to keep them.
5. **Log In and Sign Up**: Users can create an account or log in, with the option to use Google for a simplified process.
6. **Play Button**: Users have access to a simple and accessible play button for quick playback of songs or playlists directly from the browser (web) or app (mobile).
7. **Biometrics (Mobile)**: Users can unlock their accounts instantly using their fingerprint for secure and convenient access (on mobile).

## 2.4. Constraints

### **Limitations on Developer's Options**

1. **Regulatory Policies**: The app must comply with data protection laws like GDPR and CCPA, ensuring proper handling, encryption, and storage of user data.
2. **Hardware Limitations**: The app must support a wide range of devices, optimizing storage and performance for offline listening and music playback, especially on lower-end devices.
3. **Interfaces to Other Applications**: External services like Google for sign-ins must be reliable and secure. Changes to third-party APIs may affect functionality.
4. **Parallel Operation**: The app must handle multiple simultaneous users and processes, ensuring smooth music updates, logins, and playback.
5. **Audit Functions**: The app must track user activities (e.g., sign-ups, music imports) securely and efficiently, without affecting performance.
6. **Control Functions**: The app should have effective account and library management functions to handle user access, updates, and data integrity.
7. **Reliability Requirements**: The app must be highly reliable, especially for core features like playback and song imports, ensuring minimal service interruptions.
8. **Criticality of the Application**: The app’s functionality is critical to users, and failure in key features (e.g., playback) can lead to dissatisfaction and loss of users.
9. **Safety and Security Considerations**: The app must secure user data with encryption, strong authentication, and regular security updates to prevent unauthorized access.

## 2.5. Assumptions and dependencies

This section lists the factors that influence the requirements outlined in the SRS. These are not design constraints but rather assumptions or dependencies that may change and could require updates to the SRS.

1. **Mobile Platform**:
   1. The system assumes that the software will only be deployed on **Android mobile devices**, and the application will be developed for **Android OS versions 5.0 (Lollipop)** and higher.
   2. The availability of the **Android operating system** on devices is assumed. If an incompatible OS is introduced or if a different OS version is required, the SRS will need modification.
2. **Device Hardware**:
   1. The app assumes that the target Android devices will include the necessary hardware components like **touchscreens** and **fingerprint sensors** (for biometric login) to ensure full functionality.
   2. The system expects that Android devices will have **adequate internal storage** for music file storage. Changes in hardware specifications, such as a lack of fingerprint sensors, smaller screen sizes, or reduced storage capacity, may require adjustments.
3. **Network Connectivity**:
   1. The app will operate without the need for **Wi-Fi** or **mobile data** for key functionalities, as users can enjoy **offline listening** by downloading their favorite songs. However, online features (e.g., Google Easy Sign-In, music imports) may require temporary internet connectivity.
   2. If there are limitations on network availability (e.g., no mobile data service or limited connection), the app’s online features may be restricted but offline functions will still be available.

# Specific Requirements

## External interface requirements

### 3.1.1. Hardware interfaces

*This should specify the logical characteristics of each interface between the software product and the hard- ware components of the system. This includes configuration characteristics (number of ports, instruction sets, etc.). It also covers such matters as what devices are to be supported, how they are to be supported, and protocols. For example, terminal support may specify full-screen support as opposed to line-by-line support.*

This section outlines the hardware interfaces of the system, focusing on Android mobile phones, as the platform is exclusively designed for mobile use.

1. **Device Support**:
   1. The application is compatible with Android smartphones, with a minimum requirement of Android version 5.0 (Lollipop).
   2. It supports smartphones with varying screen sizes and resolutions, ranging from 480x320 to 2560x1600 pixels.
2. **Input Devices**:
   1. **Touchscreen**: The system relies on **touchscreen interaction** for all user input, including taps, swipes, and gestures for navigating the app and controlling playback (e.g., play, pause, skip).
   2. **Fingerprint Sensor**: The system utilizes the **fingerprint sensor** available on supported Android devices for biometric authentication (login).
3. **Audio Output**:
   1. The application supports **internal speakers** and **Bluetooth audio devices** (e.g., wireless headphones, Bluetooth speakers) for music playback.
   2. Audio quality is dependent on the device's built-in speaker configuration or connected Bluetooth device.
4. **Storage**:
   1. The app will access the **device's internal storage** to store and manage music files. The app does not require external storage or SD cards, but it uses available space on the device for storing downloaded music.
   2. The system will also allow the user to import and delete songs stored on the device's local storage.
5. **Biometric Authentication Hardware**:
   1. The app supports **fingerprint sensors** for user authentication on Android devices that include this hardware.
   2. The system will use **Android's biometric API** to manage secure login and access to the app using biometric data.

### 3.1.2. Software interfaces

This section specifies the required software products and how the system interfaces with them, including any dependencies on external systems, software, and services.

### **Operating System**

* **Web Application**: The web application will be compatible with modern web browsers, such as Google Chrome, Mozilla Firefox, Safari, and Microsoft Edge. The backend server running Spring Boot will be hosted on Windows servers, depending on the deployment environment.
* **Mobile Application (Android)**: The mobile application will be compatible with Android OS version 10 and above.

### **Data Management System**

* **MySQL**: MySQL will serve as the primary data management system for storing user accounts, music libraries, playlists, favorite songs, and other app data. The backend (Spring Boot) will interact with MySQL through JDBC to handle database operations, including CRUD (Create, Read, Update, Delete).

### 3.1.3. Communications interfaces

*This should specify the various interfaces to communications such as local network protocols, etc.*

The system should support various communication interfaces to ensure smooth interaction with the app’s functionalities, including:

1. **Local Network Protocols**:
   1. The app should communicate over common network protocols such as HTTP/HTTPS for secure data transmission between the web or mobile application and the server.
   2. For file uploads and downloads (e.g., importing music), the system should use reliable and efficient protocols such as FTP or SFTP when applicable.
2. **API Interfaces**:
   1. The app will interact with external services, like Google for Easy Sign-In, through secure and standardized RESTful APIs.
   2. The system should handle API responses and errors gracefully, ensuring smooth data exchange and user experience.
3. **Database Communication**:
   1. The application should connect to the database using standard communication protocols like JDBC (Java Database Connectivity) for backend systems, ensuring secure and efficient data exchange.
4. **Offline Data Synchronization**:
   1. The mobile app should allow users to store data offline and sync it with the cloud once a stable network connection is available, utilizing background synchronization protocols.
5. **Authentication & Authorization Protocols**:
   1. Secure login mechanisms (such as OAuth for Google sign-in) will be used to ensure secure communication for user authentication and authorization.

## Functional requirements

### 3.2.1. Web Application Features

1. **Import Music: Users should be able to import music from their devices, consolidating their entire music library into one convenient space.**
2. **Users should be able to upload new music files to the platform from their devices.**
3. **The system should automatically update the music files in the user's library when changes occur.**
4. **Users should be able to delete songs they no longer wish to keep in their library.**
5. **Offline Listening: Users should be able to download their favorite songs for offline playback, ensuring uninterrupted enjoyment even without an internet connection.**
6. **Google Easy Sign-In: Users should be able to quickly and effortlessly sign in with a single tap using Google Easy Sign-In, avoiding the need for complex account setups.**
7. **Favorite Songs: Users should be able to mark and organize their favorite songs by clicking the heart icon next to any track.**
   1. **Users should be able to use an advanced search bar to quickly locate songs.**
   2. **Users should be able to remove songs from their favorites if they no longer wish to keep them.**
8. **Log In and Sign Up: Users should be able to create an account or log in, with the option to use Google for a simplified process.**
9. **Play Button: Users should have access to a simple and accessible play button for quick playback of songs or playlists directly from the browser.**

### 3.2.2. Mobile Application Features

1. **Import Music: Users should be able to import music from their devices, consolidating their entire music library into one convenient space.**
   1. **Users should be able to upload new music files to the platform from their devices.**
   2. **The platform should update music files in the user's library.**
   3. **Users should be able to delete songs they no longer wish to keep in their library.**
2. **Offline Listening: Users should be able to download their favorite songs for offline playback, ensuring uninterrupted enjoyment even without an internet connection.**
3. **Google Easy Sign-In: Users should be able to sign in quickly and effortlessly with a single tap using Google Easy Sign-In, avoiding complicated account setups.**
4. **Biometrics: Users should be able to unlock their accounts instantly using their fingerprint for secure and convenient access.**
5. **Favorite Songs: Users should be able to mark and organize their favorite songs by clicking the heart icon next to any track.**
   1. **Users should be able to quickly locate songs using an advanced search bar.**
   2. **Users should be able to remove songs from their favorites if desired.**
6. **Log In and Sign Up: Users should be able to create an account or log in, with the option to use Google for a simplified process.**
7. **Play Button: Users should be able to access the play button for instant playback, easily accessible on any screen or while navigating the app.**

## Non-functional requirements

### Performance

##### **Music Playback:** Music must play 1 second or less when user clicks “Play”

##### **API endpoints:** must respond within 500 milliseconds or less 95% of request

##### **Offline Playbacks:** Downloaded songs must play instantly without delays

### Security

##### **Secure Storage:** Passwords must be hashed

* ***Authentication:*** *Implement secure authentication mechanisms (e.g., OAuth 2.0 for Google Sign-In).*
* ***Authorization:*** *Validate user permissions on the server-side for every request.*

### Reliability

* ***Streaming Quality:*** *Ensure smooth music streaming with minimal buffering, even during peak usage times.*
* ***Error Handling:*** *The system must handle errors without crashing*
* ***Response Time****: The system must respond to user requests within 2 seconds or less for 95% of requests.*

## ***3.5*** ***System Features***

* ***User Authentication*** – Secure login system to protect user accounts
* ***Import Music*** – Upload and manage music files seamlessly
* ***Offline Listening*** – Enjoy music without an internet connection
* ***Import Music*** – you can upload music into the platform
* ***Biometric login for mobile*** – Use fingerprints for quick access
* ***Play Button*** – Instantly play music with a simple tap

## ***3.6*** ***System Architecture***

* The project must follow a client-server architecture.
* At least one web-based interface and one mobile application must be included.

## ***3.7*** ***Technology Stack***

* The server backend must be developed using Java Spring Boot
* Frontend will be ReactJs
* The mobile application must be built using Android Kotlin.

## ***3.8*** ***Database Integration***

* MySQL (hosted on AWS RDS) - scalable and managed database hosting for reliable data storage

## ***3.9*** ***System Integration***

* Google OAuth 2.0 - Securely authenticate users using their Google accounts