# SOFTWARE REQUIREMENT SPECIFICATION OF MUSIC PLAYER SYSTEM

Prepared by:

Roll No :- IT029 (H2)

Name :- Trupal Godhat

Roll no :- IT026 (H2)

Name :- Krunal Dudhatra

# Table of contents

3.6 Music Listening and Control

**3.7** Follow Artists and Share Content

3.8 Subscription

## 4.Non-Functional Requirements

4.1 Performance Requirements

4.2 Availability Requirements

4.3 Data Integrity  Requirements

4.4 Security  Requirements

4.5 Usability Requirements

# 1.Introduction

## 1.1 Purpose of the Project

The purpose of this Music Player application is to provide a platform where artists can upload their songs and users can listen to these songs. This software will be deployed in a cloud environment and will be accessible through web and mobile applications. Artists will have accounts where they can manage and upload their content, while users will have accounts to create playlists, follow artists, and listen to music. This platform aims to support both amateur and professional artists in sharing their music and reaching a broad audience.

## 1.1.1 The software will be used in various environments, including:

Web browsers (Google Chrome, Mozilla Firefox, Safari, etc.)

Mobile devices (Android, iOS, etc.)

Desktop applications (Windows, macOS, Linux, etc.)

## 1.2 Project Scope

The Music Player application is designed to automate the distribution and listening of music by enabling artist registration and song uploads, alongside user registration, streaming, and playlist management. Users can search for songs and artists to enhance their music discovery. Future developments will include personalized recommendation algorithms, detailed analytics for artists, social sharing features, offline listening, and integration with third-party services to expand the music library. This project aims to create a comprehensive and engaging platform for both artists and music enthusiasts

## 1.3 Environmental Characteristics

The Music Player application will interact with the following environments:

- **Hardware:** The application will primarily run on servers hosted in the cloud, which will handle the storage and streaming of music files. End users will access the

application on personal computers, smartphones, and tablets.

- **Software:** The backend will be developed using a scalable and secure framework (e.g., Node.js, Django), and the frontend will be developed using modern web and mobile development frameworks (e.g., React, Flutter). The application will interact with databases (e.g., PostgreSQL, MongoDB) for storing user and music data, and with cloud storage services (e.g., AWS S3) for storing the music files.

- **Network:** The application will require a stable internet connection for both uploading and streaming music. It will leverage Content Delivery Networks (CDNs) to ensure fast and reliable streaming performance globally.

# 2.Overall Description

## 2.1 Product Perspective

The Music Player application is a new software solution designed to provide a platform for artists to upload their music and for users to listen to it. It is not a replacement for an existing system but rather an innovative product aimed at enhancing music distribution and consumption. The software will function as a standalone system but will integrate with various external services such as cloud storage for music files and social media platforms for sharing content.

The Music Player application is a new platform designed for artists to upload music and for users to stream it, integrating with cloud storage and social media. It is a standalone system, not replacing any existing system.

## 2.2 Product Features

The Music Player application will offer a range of features to enhance user experience and facilitate artist content management. Key features include:

- Artist registration and song upload.
- User registration and login.
- Music streaming and playlist creation.
- Search functionality for songs and artists.
- Future enhancements: recommendations, social sharing, and offline listening.

## 2.3 User Classes

The software is designed to cater to various user classes:

- **Artists**: Users who will upload and manage their music content.
- **Listeners**: Users who will stream music, create playlists, and follow artists.
- **Administrators**: Users with elevated privileges to manage the platform, oversee user activity, and maintain system integrity.

## 2.4 Operating Environment

The Music Player application will run on the following platforms:

- **Hardware**: Cloud servers for hosting the backend and music files, personal computers, smartphones, and tablets for end-users.
- **Operating Systems**: The backend will be hosted on cloud-based servers running Linux, while the frontend will be accessible on various operating systems including Windows, macOS, Android, and iOS.
- **Software**: The application will interact with web browsers, mobile operating systems, cloud storage services, and databases.

## 2.5 Design and Implementation Constraints

The design and implementation of the Music Player application will adhere to the following constraints:

- **Corporate or Regulatory Policies**: Compliance with data privacy laws and music licensing regulations.
- **Hardware Limitations**: Ensuring the application is optimized for performance and can handle large numbers of concurrent users.

- **Interfaces to Other Applications**: Integration with cloud storage solutions and social media platforms.
- **Technologies and Tools**: Utilization of specific frameworks (e.g., React, Flutter for frontend; Node.js, Django for backend), databases (e.g., PostgreSQL, MongoDB), and cloud services (e.g., AWS, Google Cloud).
- **Programming Languages**: Primarily JavaScript, Python, and Dart.
- **Communication Protocols**: HTTPS for secure communication, RESTful APIs for backend services.
- **Security Considerations**: Implementation of robust security measures to protect user data and content.

## 2.6 User Documentation

The following types of user  documentation will be provided:

- **User Manuals**: Comprehensive guides detailing how to use the application’s features.
- **Online Help**: Context-sensitive help accessible from within the application.
- **Troubleshooting Manuals**: Guides to help users resolve common issues.
- **Video Tutorials**: Step-by-step video instructions on how to use the application.

# 3. Functional Requirements

## 3.1. User:

### R.1. User Registration

- **Description**: To use the system, users need to register themselves with their personal details such as username, email, contact information, password, and address.
- **Input**: Valid details
- **Output**: Confirmation message upon successful registration and prompted to login.

### R.1.1. Login

- **Description**: After successful registration to access, all features users required to login into the system.
- **Input**: Username and Password
- **Output**: user can access features if it successful login or receive error message if invalid Username or Password.

### R.2. Profile Management

- **Description**: After successful login Allows users to manage their profiles.
- **Input**: Profile picture, Gender, Age, Country, Language.
- **Output**: Confirmation of profile update.

**R.3. Search and Discovery**

- **Description**:  Allows users to search for songs, artists, or genres and discover new music.
- **Input**: Search queries (keywords, song/artist/genre names).
- **Output**: Search results displaying relevant songs and artists, recommendations based on search history and preferences.

**R.4. Filter**

- **Description**: Allows users to apply filter for language, favourite artist, genre.
- **Inputs**: Select language, favourite artist, genre etc.
- **Outputs**: display the filterout song or playlist.

**R.5. Manage Playlists**

- **Description**: Users can create and manage their playlists.

**R.5.1: Create Playlist**

- **Input**: Enter playlist name and add songs.
- **Output**: Playlist created with added songs.

### R.5.2 Add/Remove Songs

- **Input**: Select songs to add or remove from the playlist .
- **Output**: Playlist updated with the changes .

## R.6. Music Listening and Control

- **Description**: Enables users to listen music and control functionalities.
- **Input**: Listener can perform action like pause, resume, loop etc.
- **Output**: Listener can easily listen music .

## R.7. Follow Artists and Share Content

- **Description**: Users can follow their favourite artists and share content with others.
- **Input**: Selection of artist to follow, content to share.
- **Output**: Updated list of followed artists, shared content notifications to friends/followers.

**R.8. Subscription**

- **Description**: user can get subscription for add free music and better sound quality.
- **Inputs**: user can choose subscription type.
- **Outputs**: user can access all functions based on their subscription.

## 3.2. Artist:

### R.1. Artist Registration

- **Description**: To use the system, users need to register themselves with their personal details such as username, email, contact information, password, and address.
- **Input**: Valid details
- **Output**: Confirmation message upon successful registration and prompted to login.

### R.1.1. Login

- **Description**: After successful registration to access, all features users required to login into the system.
- **Input**: Username and Password
- **Output**: user can access features if it successful login,or receive error message if invalid Username or Password.

### R.2. Profile Management

- **Description**: After successful login Allows users to manage their profiles.
- **Input**: Profile picture, Gender, Age, Country, Language.
- **Output**: Confirmation of profile update.

### R.3. Song Upload and Management

- **Description**: This functionality enables artists to upload their songs and manage their content.
- **Input**: Song file (audio), metadata (title, genre, release date), cover art.
- **Output**: Confirmation of successful upload, song listing on artist's profile, options for editing or deleting songs.

### R.4. View and Analysis Song Performance

- **Description**: Provides artists with analytics on their songs' performance.
- **Input**: Request for song performance data.
- **Output**: Analytics dashboard showing metrics such as plays, likes, shares, and comments.

### 3.3. Administrators:

### R.1. Manage User Accounts

- **Description**: Administrators can manage user accounts, including actions like suspending or deleting accounts.
- **Input**: User account details, actions (suspend, delete).
- **Output**: Confirmation of account actions, updated user account status.

### R.2. System Maintenance and Updates

- **Description**: Administrators perform system maintenance and updates to ensure smooth operation.
- **Input**: Maintenance schedule, update details.
- **Output**: Confirmation of maintenance actions, system update logs.

## 4. Non-Functional Requirements

This section outlines the non-functional requirements of the Bus Transportation Management System (BTMS), specifically focusing on performance requirements, safety requirements, and security requirements.

### N.1 Performance

**Description**: The application must support streaming for at least 10,000 concurrent users without noticeable lag or downtime.

### N.2 Availability

**Description**: The system should be accessible 24/7, with planned maintenance scheduled during off-peak hours to minimize user impact.

### N.3 Data Integrity

**Description**: Ensure that all user and song data is accurately stored, retrieved, and maintained without corruption or loss.

### N.4 Security

**Description**: Implement robust security measures including data encryption, secure authentication, and authorization to protect user data and prevent unauthorized access.

## N.5 Usability

**Description**: The user interface should be intuitive and easy to navigate for both artists and listeners, ensuring a smooth user experience.