

# MeloFi - Software Requirements Specification

---

## 1. Introduction

### 1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define the objectives, scope, and requirements for MeloFi, a music-sharing and discovery platform. This document ensures a clear understanding between developers, designers, and stakeholders about what the system will do and how it will function.

### 1.2 Scope

MeloFi enables users to upload, stream, and explore music from various artists and genres. It focuses on personalized playlists, social interaction, and community-driven discovery, providing an engaging environment for both creators and listeners.

### 1.3 Objectives

- Provide a smooth, user-friendly platform for sharing and discovering music.
- Empower independent artists to reach a wider audience.
- Enable listeners to curate personalized playlists based on mood or genre.
- Build an interactive music community with social features like likes and comments.

### 1.4 Intended Audience

This document is intended for:

- **Developers:** To understand technical and functional requirements.
- **Designers:** To create a visually appealing and intuitive interface.
- **Project Managers:** To track progress and align with goals.
- **Stakeholders:** To review and validate project direction.

### 1.5 Acronyms

- **SRS** -Software Requirements Specification
- **UI/UX** -User Interface / User Experience
- **API** - Application Programming Interface

## 2. Project Description

### 2.1 Project Overview

MeloFi is a digital platform where users can upload, stream, and share music. It aims to create a vibrant ecosystem connecting music lovers and artists, offering personalized playlists, discovery tools, and a social layer that enhances engagement.

### 2.2 Target Users

- **Listeners:** Users who explore, stream, and share music.
- **Artists:** Creators who upload and promote their songs.
- **Admins:** Moderators ensuring content quality and platform safety.

### 2.3 Core Features

- **User Accounts:** Signup, login, and profile management.
- **Music Upload & Playback:** Artists upload; listeners stream seamlessly.
- **Playlist Creation:** Create and share custom playlists.
- **Explore Section:** Discover trending and recommended tracks.
- **Social Interaction:** Like, comment, and follow other users.
- **Admin Dashboard:** Manage uploads, monitor users, and track analytics.

### 2.4 Operating Environment

- **Platform:** Web and mobile (responsive).
- **Devices:** Desktop, tablet, and smartphone.
- **Network:** Requires stable internet connection.

### 2.5 Constraints

- Audio upload size limit for performance optimization.
- Compliance with copyright and content regulations.
- Scalable backend for growing user base.

### 2.6 Assumptions

- Users have stable internet access.
- Uploaded audio files are in supported formats (e.g., MP3, WAV).
- Third-party tools (e.g., Firebase, AWS) remain available and reliable.

### **3. GOALS**

- Simplify music sharing and streaming.
- Support emerging artists with visibility tools.
- Deliver personalized recommendations based on user interests.
- Promote interaction between listeners and creators.

### **4. Technology Stack**

Layer	Tools	Purpose
Frontend	React.js, Tailwind CSS, Redux/context API	Build responsive and dynamic user interfaces
Backend	Flask	APIs, route, and secure authentication
Database	Mongo DB/Firebase Firestore	Store users, tracks, and playlist data
Storage	AWS S3/Firebase storage	Host and serve music files
Deployment	Vercel, Render, AWS EC2	Continuous deployment and hosting
Version Control	Git & Github	Manage source code and collaboration
Testing	Postman	Test and verify API endpoints

### **5. Conclusion**

Through this document, the functionalities , technical specifications , interactions of the user are clearly specified inorder to guide developers and designers towards creating a user-friendly application. This enables a smooth progress in its development.