**Software Requirements**

**Specification**

**for**

**MelodyHub – Music Education Platform**

**Version 1.0 approved**

**Prepared by:**

Amrik Bhadra (202201040021)

Essak Dasari (202201040013)

Raviraj Tekle (202301040010)

Harish Chavankde (0120200177)

Omkar Shinde (202201040190)

Ankur Dome (202201040191)

**30-05-2025**

***Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.***

# Table of Contents

[Table of Contents ii](#_TOC_250033)

[Revision History ii](#_TOC_250032)

1. [Introduction 1](#_TOC_250031)
   1. [Purpose 1](#_TOC_250030)
   2. [Document Conventions 1](#_TOC_250029)
   3. [Intended Audience and Reading Suggestions 1](#_TOC_250028)
   4. [Product Scope 1](#_TOC_250027)
   5. [References 1](#_TOC_250026)
2. [Overall Description 2](#_TOC_250025)
   1. [Product Perspective 2](#_TOC_250024)
   2. [Product Functions 2](#_TOC_250023)
   3. [User Classes and Characteristics 2](#_TOC_250022)
   4. [Operating Environment 2](#_TOC_250021)
   5. [Design and Implementation Constraints 2](#_TOC_250020)
   6. [User Documentation 2](#_TOC_250019)
   7. [Assumptions and Dependencies 3](#_TOC_250018)
3. [External Interface Requirements 3](#_TOC_250017)
   1. [User Interfaces 3](#_TOC_250016)
   2. [Hardware Interfaces 3](#_TOC_250015)
   3. [Software Interfaces 3](#_TOC_250014)
   4. [Communications Interfaces 3](#_TOC_250013)
4. [System Features 4](#_TOC_250012)
   1. [System Feature 1 4](#_TOC_250011)
   2. [System Feature 2 (and so on) 4](#_TOC_250010)
5. [Other Nonfunctional Requirements 4](#_TOC_250009)
   1. [Performance Requirements 4](#_TOC_250008)
   2. [Safety Requirements 5](#_TOC_250007)
   3. [Security Requirements 5](#_TOC_250006)
   4. [Software Quality Attributes 5](#_TOC_250005)
   5. [Business Rules 5](#_TOC_250004)
6. [Other Requirements 5](#_TOC_250003)

[Appendix A: Glossary 5](#_TOC_250002)

[Appendix B: Analysis Models 5](#_TOC_250001)

[Appendix C: To Be Determined List 6](#_TOC_250000)

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

This document outlines the Software Requirements Specification (SRS) for **MusicVerse v1.0**, a full-featured music education platform. The goal of this platform is to provide structured, interactive, and personalized music learning experiences. This SRS covers the entire software system, including backend services, frontend interfaces, and integrations with third-party APIs (e.g., video streaming, payment gateways). It is intended for use throughout the project lifecycle, from planning and design to development and testing.

## Document Conventions

* **Bold text** is used to highlight section titles, sub-headings & important keywords/information.
* Each functional requirement is uniquely labeled using the format **REQ-#** (e.g., REQ-1, REQ-2).
* **Each requirement is assigned an explicit priority level** (High, Medium, Low).
* Technical terms and abbreviations are defined in Appendix A: Glossary

## Intended Audience and Reading Suggestions

This document is intended for the following stakeholders:

* **Developers** – to understand functional and non-functional system requirements.
* **Testers/QA Engineers** – to develop test cases and test plans based on defined requirements.
* **Project Managers and Product Owners** – to monitor progress and ensure deliverables align with objectives.
* **UX/UI Designers** – to align designs with functional flows and use cases.
* **Client and Investors** – to get a clear overview of product capabilities and expected outcomes.

## Product Scope

**MelodyHub** is a web-based music education platform aimed at democratizing access to high-quality music learning. The platform delivers structured courses in music theory and practice, supports live sessions, tracks progress, and integrates AI-based feedback. It enables global collaboration between students and instructors and supports monetization via subscription and course-based payments. MelodyHub aligns with modern educational trends by offering an engaging, scalable, and personalized learning environment.

Key goals include:

* Enhancing music learning through interactive, engaging content.
* Supporting instructors with tools to create and manage courses.
* Empowering learners with tailored learning paths and progress tracking.
* Supporting platform monetization through subscriptions and certifications.

## References

* IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
* JWT Authentication Guide – <https://jwt.io/introduction>
* React.js Official Docs – <https://reactjs.org/docs/getting-started.html>
* Node.js Official Docs – <https://nodejs.org/en/docs>
* MongoDB Manual – <https://www.mongodb.com/docs/manual/>

# Overall Description

## Product Perspective

MelodyHub is a new, standalone web-based platform designed to deliver comprehensive music education through interactive, personalized learning tools. It is not an extension or replacement of any previous system and is being developed from scratch using the **MERN stack** (MongoDB, Express.js, React.js, Node.js). While the product itself is self-contained, it integrates with several third-party services such as **RazorPay** for payments, **WebRTC** for live sessions, and AI feedback engines for music analysis.

## Product Functions

At a high level, MelodyHub will enable the following functionalities:

* **User Authentication & Management**: Secure signup/login, role-based access (student, instructor, admin), and profile settings.
* **Course Management**: Creation, browsing, enrollment, and progress tracking of music courses.
* **Content Delivery**: Video-based lessons, quizzes, assignments, and interactive tools.
* **Live Sessions**: Schedule and attend real-time video classes and webinars.
* **Feedback & Analytics**: AI-driven performance analysis and progress reports.
* **Communication**: In-app chat, discussion forums, and community engagement.
* **Monetization**: Support for subscriptions, course-based payments, and certifications

## User Classes and Characteristics

MelodyHub will cater to multiple user types:

* **Students**
  + Primary users of the platform
  + May vary in age and musical experience
  + Use basic to intermediate features like course consumption, quizzes, and progress tracking
* **Instructors**
  + Skilled musicians or educators
  + Require advanced tools for course creation, feedback, and live session management
* **Administrators**
  + Responsible for content moderation, user management, and platform oversight
  + Require full backend access and analytics
* **Visitors/Guests**
  + Can browse available courses but need to register to access learning content

## Operating Environment

MelodyHub will be accessible on **standard web browsers** (Chrome, Firefox, Safari, Edge) and optimized **for desktop and mobile**.

* **Frontend**: React.js (HTML5, CSS3, JavaScript)
* **Backend**: Node.js with Express.js
* **Database**: MongoDB (Cloud-hosted)
* **Hosting**: Deployed via platforms like Vercel (frontend) and AWS/GCP/Render (backend)
* **Third-party Integrations**: RazorPay, WebRTC, Firebase for email services, and optional AI/ML APIs

## Design and Implementation Constraints

* Must use the **MERN stack** as per team expertise
* **JWT-based** authentication is required for secure role-based access control
* **Real-time features** (chat, live sessions) must scale with user volume
* **AI-based feedback systems** may depend on available open-source models or APIs
* Integration with **Razorpay** is mandatory for handling payments

## User Documentation

The following documentation will be provided with the platform:

* **User Manual** – for students and instructors (PDF + HTML format)
* **Admin Guide** – covering backend controls and reports
* **Onboarding Tutorials** – guided flows for new users inside the app
* **Help Center/FAQ Page** – accessible from the app interface
* **API Documentation** – for internal/external use (Swagger/OpenAPI format)

## Assumptions and Dependencies

* Assumes continuous internet connectivity for all real-time and content-heavy features
* Dependent on third-party APIs (Stripe, Zoom/WebRTC) for core functionalities
* Assumes instructors will be responsible for uploading content and managing their courses
* Depends on React and Node.js libraries being compatible across all modules
* Scalable cloud hosting will be available for deployment and data storage

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on.

Define the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

## Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

## Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

## Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

# System Features

<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>

## System Feature 1

<Don’t really say “System Feature 1.” State the feature name in just a few words.>

### Description and Priority

<Provide a short description of the feature and indicate whether it is of High, Medium, or Low priority. You could also include specific priority component ratings, such as benefit, penalty, cost, and risk (each rated on a relative scale from a low of 1 to a high of 9).>

### Stimulus/Response Sequences

<List the sequences of user actions and system responses that stimulate the behavior defined for this feature. These will correspond to the dialog elements associated with use cases.>

### Functional Requirements

<Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs.

Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.>

<Each requirement should be uniquely identified with a sequence number or a meaningful tag of some kind.>

### REQ-1:

### REQ-2:

## System Feature 2 (and so on)

# Other Nonfunctional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.>

## Safety Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied.>

## Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. Refer to any external policies or regulations containing security issues that affect the product. Define any security or privacy certifications that must be satisfied.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.>

## Business Rules

<List any operating principles about the product, such as which individuals or roles can perform which functions under specific circumstances. These are not functional requirements in themselves, but they may imply certain functional requirements to enforce the rules.>

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

# Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

# Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

# Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>