Project Management Document

for

Music Streaming System

Version 1.0 approved

Prepared by Group 5

2324II INT2208E 23, VNU-UET

April 9, 2024

Revision History

Date	Version	Description	Author
May 18th 2024	1.0	complete details on project scope	Trần Thế Mạnh

Table of Contents

1. Introduction

1.1. Purpose

The digital revolution has transformed the way we experience music, bringing the vast universe of songs from around the globe to our fingertips. MusicStreaming stands at the forefront of this transformation, offering a seamless and personalized music streaming service to millions of users. This document delineates the requirements for MusicStreaming's software system, ensuring that it continues to meet the evolving needs of its users with efficiency, reliability, and innovation.

1.2. Document Conventions

Headings Times New Roman/15 font size/Bold

Sub Headings Times New Roman/14 font size/Bold

1.3. Intended Audience and Reading Suggestions

The intended audience for our music app includes music enthusiasts, casual listeners, and professionals alike, seeking a seamless and immersive music experience. Whether you're a dedicated audiophile looking for curated playlists or an artist interested in reaching new audiences, our app caters to your diverse needs. For those new to the platform, we recommend exploring our recommendations curated playlists to discover new music tailored to your taste. Additionally, artists can benefit from our platform's tools for promoting their music and connecting with fans. With something for everyone, our music app invites users to explore, discover, and enjoy music in all its forms.

1.4. Product Scope

The scope of this project encompasses the development and enhancement of the MusicStreaming application, a premier music streaming service. The project aims to deliver a user-friendly platform that provides access to a vast library of music, podcasts, and videos from creators all over the world.

The key objectives include:

- Delivering high-quality audio streaming to users globally.
- Providing personalized content recommendations based on user preferences.
- Ensuring seamless integration across various devices and platforms.
- Maintaining a robust and scalable infrastructure to support a growing user base.

- Incorporating social features that allow for sharing and discovering content within user communities.

2. Project integration management

2.1. Track and Control Work

To ensure the project stays on track and meets its objectives, effective tracking and control mechanisms are essential. The team will use Trello for this purpose:

- Trello Boards: Create Trello boards to track tasks, assign responsibilities, and monitor progress. Each board will represent a different aspect of the project, such as development, testing, and documentation.
- Regular Updates: Team members will update Trello cards regularly to reflect the current status of their tasks. This provides a clear, real-time view of progress for everyone involved.
- Team Meetings: Hold team meetings twice a week to review progress, address any issues, and adjust plans as needed. These meetings will facilitate communication, collaboration, and quick problem-solving.

2.2. Change Management

- Recording Changes: All proposed changes must be documented, including the nature of the change, the reason for it, and its potential impact on the project.
- Evaluation: Each change will be evaluated in terms of its feasibility, benefits, and potential risks. This assessment will help determine whether the change should be implemented.
- Approval: Changes must be approved by the team leader, Nguyen Van A, before they can be incorporated into the project. This ensures that all changes are carefully considered and aligned with the project goals.

2.3. Project Closure

- Feedback Collection: Gather feedback from users and instructors to assess the app's performance, user satisfaction, and any areas for improvement.
- Document Archiving: Archive all project documents, including the master project plan, meeting notes, change logs, and final reports. This documentation will serve as a valuable resource for future projects and audits.

3. Project Scope Management

3.1. Collect Requirements

- Interviews:
    - Users (Students): Conduct interviews with students to understand their preferences, desired features, and pain points regarding music apps.
    - Training Departments: Interview training department staff to identify any specific needs for training purposes, such as features that support educational content or instructional use cases.

3.2. Determine Project Scope

- User Authentication and Authorization:
    - Secure login system with email/password.
    - Role-based access control to manage different user permissions.

- Track with Search and Filtering:
    - Comprehensive catalog of music type.
    - Advanced search functionality with filters based on genre, difficulty level, duration, etc.
- Playback Controls:
    - Play/Pause: Implement basic playback controls including play, pause, stop, and resume functionality.
    - Skip/Previous: Allow users to skip to the next track or go back to the previous track in a playlist.
    - Seek: Provide a seek bar that enables users to jump to any point in the track.
- Audio Quality and Streaming:
    - High-Quality Audio: Ensure tracks are available in high-quality audio formats.
    - Adaptive Streaming: Implement adaptive streaming to adjust audio quality based on the user's internet connection, ensuring smooth playback with minimal buffering.
- Create/Drop playlist Functionality:
    - Easy-to-use interface for user to create or drop playlist.
    - Real-time updates and confirmations.

- Visual and Interactive Elements:
    - Album Art Display: Show album art and track details during playback.
    - Interactive Playback: Allow users to like, share, and add tracks to playlists directly from the playback screen.

3.3. UI/UX Design:

- Design Prototyping:
    - Wireframes: Develop wireframes to outline the basic structure and layout of the app's user interface.
    - High-Fidelity Prototypes: Create detailed prototypes with visual design elements, interactions, and animations.
    - Design Tools: Use tools Figma for designing and prototyping.

- User Testing:
    - Usability Testing: Conduct usability testing sessions with real users to gather feedback on the design and identify areas for improvement.
    - Iterative Refinement: Refine the design based on user feedback, ensuring the interface is intuitive and user-friendly.
    - Accessibility Testing: Ensure the app meets accessibility standards and is usable by people with disabilities.

3.4. Frontend Development:

- Framework Selection: React
- Component Development:
    - UI Components: Develop reusable UI components based on the design specifications, such as buttons, forms, modals, and navigation elements.
    - State Management: Implement state management using tools like Redux or Context API to handle application state effectively.
    - Responsive Design: Ensure all components are responsive and provide a seamless experience across different devices and screen sizes.
- Integration:
    - API Integration: Integrate the frontend with backend services using RESTful APIs for functionalities like authentication, track retrieval, playlist management, and notifications.

    - Continuous Deployment: Set up a CI/CD pipeline to automate testing and deployment of the frontend code, ensuring quick and reliable updates.

3.5. Backend Development:

- Database Design: Design the database schema to support app functionalities.
- API Development: Develop RESTful APIs for communication between frontend and backend.
- Authentication: Implement user authentication and authorization mechanisms.
- Testing:
    - Unit Testing: Write and execute unit tests for individual components.
    - Integration Testing: Test interactions between different modules.
- Implementation:
    - Deployment Planning: Plan the deployment process, including staging and production environments.
    - User Training: Provide training and documentation for users.

3.6. Verify Scope

3.6.1. Regular User Feedback Sessions

- Scheduled Reviews:
    - Conduct regular review sessions with users and stakeholders throughout the development process.
    - Schedule these sessions at key milestones to gather feedback on completed features and functionalities.

- Feedback Collection:
    - Use structured feedback collection methods such as surveys, interviews, and focus groups.
    - Document feedback comprehensively to identify common themes and areas for improvement.

3.7. Prototype Reviews

- Design Prototypes:
    - Present design prototypes, including wireframes and high-fidelity mockups, to users and stakeholders.

    - Use tools like Figma, Sketch, or Adobe XD to create interactive prototypes that simulate the user experience.

- Validation Sessions:
    - Conduct validation sessions where users can interact with the prototypes and provide immediate feedback.
    - Focus on usability, aesthetics, and overall user experience during these sessions.

- Adjustments Based on Validation:
    - Make necessary adjustments to the design and functionality based on validation feedback.
    - Ensure that all changes are documented and communicated to the development team.
    - Requirement Traceability Matrix

- Mapping Requirements:
    - Create a requirement traceability matrix (RTM) to map each requirement to corresponding deliverables, test cases, and user stories.
    - Use project management tools JIRA to maintain and update the RTM.

- Continuous Tracking:
    - Continuously track the progress of each requirement throughout the development lifecycle.
    - Ensure that all requirements are being addressed and no critical aspects are overlooked.

- Verification and Validation:
    - Regularly verify that each requirement is met by conducting formal reviews and inspections.
    - Validate the functionality against user stories and acceptance criteria to ensure compliance.

3.8. Control Scope

3.8.1. Scope Change Management

- Change Control Process:

    - Documentation: All change requests must be formally documented, detailing the nature of the change, the reason for the change, and the impact on the project.
    - Evaluation: Assess the impact of the proposed changes on the project scope, schedule, budget, and resources. This includes evaluating the benefits, risks, and feasibility of the changes.
    - Approval: Changes must be reviewed and approved by the project leader.
    - Communication: Communicate approved changes to all stakeholders and update project documentation accordingly.

- Impact Analysis:
    - Conduct a thorough impact analysis for each change request to understand how it will affect the overall project scope and objectives.
    - Use tools like impact matrices and decision trees to facilitate the analysis and decision-making process.

3.8.2. Continuous Monitoring and Reporting

- Progress Tracking:
    - Use project management tools JIRA to continuously monitor the progress of project tasks against the scope baseline.
    - Track key performance indicators (KPIs) and milestones to ensure the project remains within scope.

- Regular Reporting:
    - Generate regular status reports that provide insights into scope adherence, progress, and any deviations.
    - Share these reports with the project team and stakeholders to maintain transparency and facilitate informed decision-making.

- Scope Audits:
    - Conduct periodic scope audits to compare actual project deliverables with the scope baseline.
    - Identify any discrepancies and take corrective actions to realign the project with the defined scope.

3.8.3. Scope Verification and Validation

- Regular Reviews:

    - Conduct regular scope verification meetings with the project team and stakeholders to review completed deliverables.
    - Use checklists and acceptance criteria to ensure that each deliverable meets the defined requirements.

- Quality Assurance:
    - Implement quality assurance processes, including automated and manual testing, to ensure that deliverables meet the required standards and specifications.
    - Conduct code reviews, peer reviews, and usability testing to validate the quality and functionality of the app.

4. Project Schedule

Conceptualization phase (1 week):

Define App's Purpose(2 days): Gather the team and brainstorm the core purpose of the music app. Determine whether it will focus on streaming, discovery, social features, or a combination.

Identify Target Audience(3 days): Conduct market research to identify demographics most likely to engage with the app. This could include age groups, music preferences, geographical location, etc.

Key Features Identification(3 days): Prioritize features that align with the app's value proposition and are feasible within the initial development timeline and budget.

Planning phase (1 weeks): Create a detailed project plan, including timelines, resources, and budget. Identify key milestones and potential risks.

Project Scope Definition(1 days): Review the outcomes of the conceptualization phase and define the scope of the project, including the features to be included in the initial release (MVP). Determine any constraints or limitations, such as budget, time, or technological requirements.

Create a Detailed Project Plan(2 days): Break down the project into smaller tasks and create a timeline for each, considering dependencies and

resource availability. Use project management tools like Gantt charts or Kanban boards to visualize the project timeline and allocate tasks to team members.

Resource Allocation(3 days): Identify the resources required for each task, including human resources (developers, designers, testers, etc.) and technological resources (software, hardware, etc.).

Budget Estimation(1 days): Estimate the costs associated with each phase of the project, including development, design, testing, marketing, and ongoing maintenance.

Risk Assessment and Mitigation(3 days): Identify potential risks and challenges that may arise during the project, such as technical issues, resource constraints, scope creep, or market fluctuations.

Design phase(1 week): Develop wireframes and prototypes of the app. Finalize the user interface and user experience design.

User Experience (UX) Design(1 day): Conduct user research to understand user preferences, behaviors, and pain points related to music app usage.

User Interface (UI) Design(3 day): Translate wireframes into high-fidelity mockups, incorporating branding elements, color schemes, typography, and visual assets.

Visual Design Refinement(1 days): Refine the UI design based on feedback received during usability testing and stakeholder reviews. Pay attention to detail, optimizing visual elements for different screen sizes and resolutions to ensure a seamless experience across devices.

Interaction Design(1 days): Define interactive elements such as buttons, gestures, animations, and transitions to enhance user engagement and usability.

Development phase(3 weeks): Code the app's functionality. This stage often includes several sub-stages, such as setting up the database, developing the front-end and back-end, integrating APIs, and testing each feature.

Environment Setup(1 day): Set up development environments, version control systems, and project management tools.

Backend Development(3 days): Develop backend functionalities, including user authentication, database management, and API integrations.

Frontend Framework Selection(1 day): Choose a suitable frontend framework or technology stack based on project requirements and team expertise.

UI Implementation(2 days): Translate UI designs into code, using HTML, CSS, and JavaScript to build responsive and visually appealing user interfaces.

Core Feature Development(3 days):Develop core app features such as music discovery, playlist creation, search functionality, and user preferences.

Integration with Third-Party Services(3 days): Integrate with third-party services such as music streaming platforms, payment gateways, and analytics tools.

Unit and Integration Testing(2 days): Conduct unit tests to validate individual components and functions, ensuring they meet specifications and perform as expected.

Performance Optimization(2 days): Optimize code for performance, scalability, and efficiency, addressing bottlenecks and improving app responsiveness. Conduct load testing to simulate heavy user traffic and identify areas for optimization.

Testing phase(1 week): Conduct thorough testing to identify and fix bugs. This includes unit testing, integration testing, and user acceptance testing.

Test Case Creation(3 days): Develop comprehensive test cases based on functional requirements, user stories, and use cases.

Test Execution(3 days): Execute test cases according to the test plan, following predefined test procedures and documenting test results.

Performance and security Testing(3 days): Perform load testing, stress testing, and scalability testing to assess the app's performance under various conditions, including peak usage and high traffic volumes.security assessments and penetration testing to identify vulnerabilities, such as authentication flaws, data leaks, and injection attacks.

Launch phase(1 week): Prepare for launch by setting up app store accounts, creating promotional materials, and planning a marketing strategy.

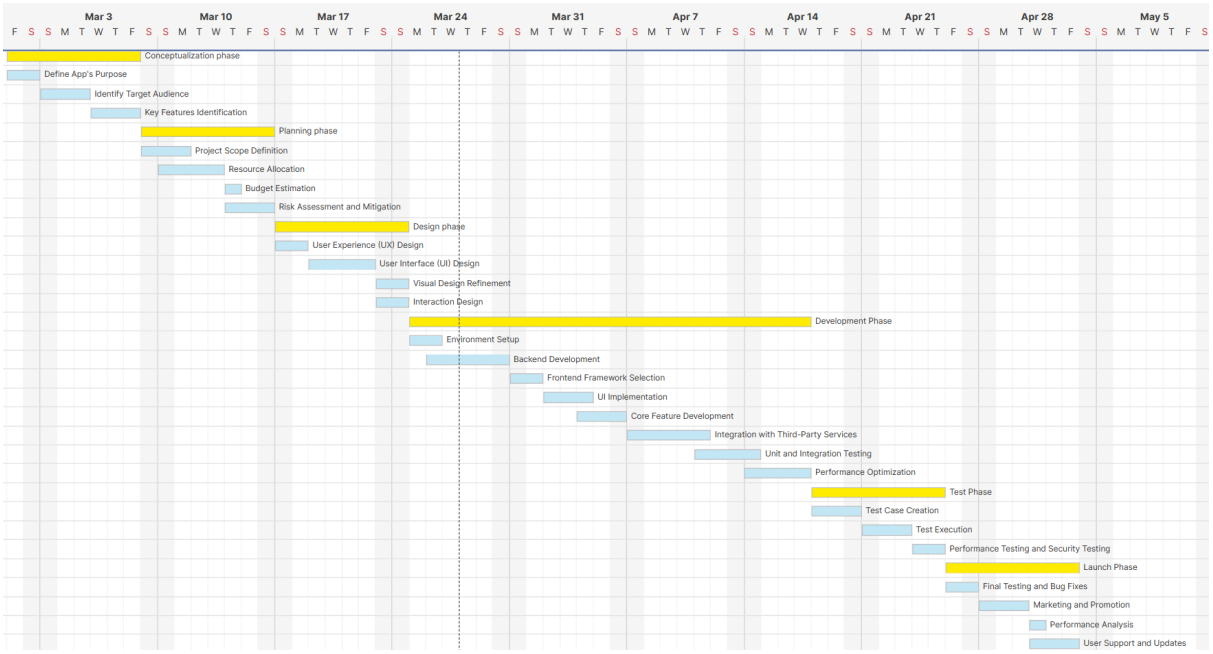
Final Testing and Bug Fixes(2 days): Conduct thorough testing of the app to identify and address any remaining bugs or issues.

Marketing and Promotion(3 days): Develop a marketing strategy to generate buzz and attract users to the app.

Performance Analysis(1 day): Analyze app performance metrics, including downloads, user engagement, retention rates, and revenue generation.

User Support and Updates(3 days): Provide ongoing customer support to address user inquiries, issues, and feature requests in a timely manner.

Post-Launch (Ongoing): Monitor the app’s performance, gather user feedback, and make necessary updates or improvements.



Gantt chart for project schedule

5. Project Cost Management

Software and Tools:

The costs associated with purchasing or licensing software tools required for development, testing, and project management: 0 VND

Costs for development environments, version control systems, project management platforms, and communication tools: 0 VND

Servers and Infrastructure:

The costs of acquiring and maintaining server infrastructure needed for hosting the music app: 0 VND

Support Tools:

Include expenses for bug tracking systems, customer support platforms, and analytics tools: 0 VND

6. Project Quality Management

6.1. Quality Planning:

- Determine Quality Standards:
    - Features: Define standards for the functionality and features of the music app, ensuring that it meets user requirements and expectations.
    - Performance: Establish performance benchmarks for the app, including response times, loading times, and resource utilization, to ensure optimal performance.
    - Security: Set rigorous security standards to protect user data, implement encryption protocols, and adhere to industry best practices for cybersecurity.
    - Usability: Define usability standards to ensure the app is intuitive, easy to navigate, and accessible to users of all abilities.
    - Evaluation Criteria: Develop clear evaluation criteria to assess the quality of the software at various stages of development.

6.2. Quality Assurance

- Automated Testing:
    - Implement automated testing processes using Selenium to validate the functionality of the app.
    - Conduct unit tests, integration tests, and end-to-end tests to identify and fix defects early in the development process.

- Manual Testing:
    - Perform manual testing by skilled testers to evaluate the app's usability, accessibility, and overall user experience.
    - Conduct exploratory testing to uncover any hidden defects or usability issues that may not be captured through automated testing.
    - Code Review:Facilitate regular code review sessions among development team members to ensure code quality, maintainability, and adherence to coding standards.

6.3. Quality Control

- Continuous Monitoring:
    - Continuously monitor the software quality throughout the development lifecycle using continuous integration (CI) and continuous deployment (CD) pipelines.

    - Implement automated monitoring solutions to track performance metrics, detect anomalies, and identify potential issues in real-time.

- Post-Deployment Monitoring:
    - Monitor software quality even after deployment to production environments.
    - Collect and analyze user feedback, error logs, and performance metrics to identify areas for improvement and address any emerging issues promptly.
    - Feedback Loop: Establish a feedback loop with stakeholders and users to gather insights and address concerns related to software quality.

7. Project human resource management

7.1. Define Roles and Responsibilities

Development Team:
Lead Developer: Ngô Lê Hoàng
Frontend Developer: Nguyễn Đức Khánh, Đỗ Minh Quang
Backend Developer: Trần Thế Mạnh, Ngô Lê Hoàng
UI/UX Designer: Đỗ Minh Quang
Quality Assurance (QA) Team: Nguyễn Đức Khánh
QA Lead: Ngô Lê Hoàng
QA Tester: Trần Thế Mạnh

7.2. Motivate Team Members

- Recognition and Rewards:
    - Regularly recognize and reward team members for their contributions and achievements.
    - Use both formal (awards, bonuses) and informal (praise, public recognition) methods to acknowledge good work.

- Professional Development:
    - Provide opportunities for team members to develop their skills and advance their careers.
    - Organize training sessions, workshops, and access to online courses.

- Positive Work Environment:
    - Foster a supportive and collaborative team culture.

- Encourage open communication, creativity, and innovation.

- Empowerment:
  - Give team members autonomy and ownership over their work.
  - Encourage them to take initiative and make decisions within their areas of responsibility.

7.3. Resolve Conflicts

- Open Communication:
  - Promote a culture of open and honest communication where team members feel comfortable sharing their concerns.
  - Address issues promptly before they escalate.

- Mediation:
  - Act as a mediator to help resolve conflicts between team members.
  - Listen to all parties involved and work towards a fair and amicable solution.

- Clear Policies:
  - Establish clear policies and procedures for conflict resolution.
  - Ensure that all team members are aware of these policies and understand the process.

8. Manage project communication

- Frequency: Weekly meetings will be held every Monday to review project progress, discuss any challenges or roadblocks, and plan tasks for the upcoming week.
- Format: Meetings will be conducted virtually via video conferencing tools such as Zoom to accommodate remote team members. An agenda will be circulated prior to each meeting to ensure discussions remain focused and productive.
- Communication Tools:

Email: Used for official communication, sharing important updates, and circulating meeting agendas and minutes.

- Decision-making Process:

Consensus Building: Decisions will be made collaboratively through consensus-building discussions during weekly meetings or via Slack channels. Input from all team members will be considered, and decisions will be based on a collective understanding of project requirements, goals, and constraints.

Project Manager Authority: In cases where consensus cannot be reached or immediate decisions are required, the project manager will have the authority to make final decisions, ensuring the project stays on track and deadlines are met.

Documentation: Decisions made and rationale behind them will be documented in meeting minutes and shared with the team to ensure transparency and accountability.

9. Risk Assessment and Mitigation Strategies

9.1. Process and Steps to Manage Risk

9.1.1. Risk Identification:

- SWOT Analysis: Perform a SWOT analysis (Strengths, Weaknesses, Opportunities, Threats) to uncover risks that might affect the project.
- Checklists: Use checklists based on past projects and industry standards to ensure all common risks are considered.

9.1.2. Risk Assessment:

- Qualitative Risk Analysis: Evaluate the identified risks based on their likelihood and impact. Categorize risks as low, medium, or high priority.
- Likelihood: Assess the probability of each risk occurring.
- Impact: Determine the potential impact on the project if the risk materializes.
- Quantitative Risk Analysis: For high-priority risks, perform a quantitative analysis to estimate their potential financial impact and schedule delays.

9.1.3. Risk Prioritization:

- Risk Matrix: Use a risk matrix to prioritize risks based on their likelihood and impact. This helps in focusing on the most significant risks first.

9.1.4. Risk Response Planning:

- Avoidance: Develop strategies to avoid high-impact risks by changing project plans or requirements.
- Mitigation: Implement measures to reduce the likelihood or impact of risks. For example, use automated testing to mitigate the risk of software bugs.

- Transfer: Transfer the risk to a third party, such as through insurance or outsourcing certain project activities.
- Acceptance: Acknowledge the risk and prepare to manage it if it occurs, often through contingency planning.

9.1.5. Risk Monitoring and Control:

- Continuous Monitoring: Regularly review and update the risk management plan to reflect any new risks or changes in existing risks.
- Risk Audits: Conduct periodic risk audits to ensure that risk management processes are being followed and are effective.
- Risk Reporting: Maintain transparent communication with stakeholders about risk status and management efforts. Use risk registers and risk reports to document and communicate risks.

9.2. Identify risks

9.2.1. technical risk:

Risk: Integration challenges with third-party APIs for music streaming and payment processing.

Mitigation:

Conduct thorough API compatibility testing during the development phase.

Establish direct communication channels with API providers to troubleshoot and resolve integration issues promptly.

Implement fallback mechanisms and error handling procedures to minimize service disruptions.

9.2.2. Resource Risks:

Risk: Key team members leaving the project unexpectedly.

Mitigation:

Cross-train team members to ensure knowledge sharing and redundancy in critical roles.

Maintain documentation and clear task assignments to facilitate smooth transition in case of personnel changes.

Consider hiring freelancers or contractors as backup resources to mitigate workload impact.

9.2.3. Schedule Risks:

Risk: Delays in development due to unforeseen technical challenges or scope changes.

Mitigation:

Implement Agile methodologies such as Scrum or Kanban to adapt to changing requirements and mitigate schedule risks.

Break down project tasks into smaller, manageable chunks with defined deadlines to maintain momentum and track progress.

Regularly review and adjust project timelines based on ongoing assessments of progress and potential risks.

9.2.4. Security Risks:

Risk: Data breaches or unauthorized access compromising user privacy and security.

Mitigation:

Implement robust authentication and authorization mechanisms to protect user accounts and sensitive data.

Encrypt data transmission using secure protocols (e.g., HTTPS) and encrypt stored data to prevent unauthorized access.

Conduct regular security audits and penetration testing to identify vulnerabilities and proactively address security risks.

9.2.5. Market Risks:

Risk: Changes in market trends, user preferences, or competitive landscape affecting app adoption and retention.

Mitigation:

Conduct market research and stay updated on industry trends to anticipate changes and adapt the app's features and strategies accordingly.

Monitor competitor activities and user feedback to identify emerging opportunities and areas for differentiation.

Implement agile development practices to iterate quickly, pivot when necessary, and stay responsive to market dynamics.

9.2.6. Legal and Compliance Risks:

Risk: Non-compliance with data protection regulations (e.g., GDPR, CCPA) or copyright infringement issues related to music licensing.

Mitigation:

Consult legal experts to ensure compliance with relevant laws and regulations governing data privacy, intellectual property rights, and digital content distribution.

Obtain necessary licenses and permissions for music streaming and ensure adherence to copyright laws and licensing agreements.

Implement robust terms of service, privacy policies, and user consent mechanisms to protect user rights and mitigate legal risks.