Student Name: Sudhanshu Akarshe, Kunal Bhoite, Pranita Nanekar

Student No: 20000762, 20000769, 20000809

Module Title: Project Management for Business Analytics

Assignment Title: Creating a Project Management Plan

Table of Contents

1.0 Introduction

The music industry is constantly evolving, and the way we consume music is no exception. Traditional music apps have been instrumental in providing access to a large music library, but they frequently lack community, personalization, and overall user experience. To address these limitations, we propose the development of an innovative music app that seamlessly blends cutting-edge technology with a human-centric approach.

This project aims to revolutionize the music consumption landscape by developing a forward-thinking platform in response to the limitations of traditional music apps. The app will use advanced algorithms to build a personalized music discovery engine that will curate playlists based on user preferences and moods. The app will connect users by fostering a sense of community through collaborative playlists and integration with social media. Furthermore, features such as seamless offline playback will improve accessibility, allowing users to listen to music on the go without the need for an internet connection. The app's design, which is both visually appealing and user-friendly, aims to provide an immersive experience that goes beyond typical music apps. The app's core focus is personalization, with extensive customization options for playlists, preferences, and interface appearance. Individuality extends to community-building, with user profiles, groups, and forums facilitating connections among music fans. This project envisions an immersive and transformative music app that uses technology and human connection to redefine how users engage with music, making it a vital companion for music lovers all over the world.

1.1 Project Goal and Objective

Goal:
The music application's goal is to revolutionize music consumption by creating an innovative platform that seamlessly blends advanced technology with a deep understanding of user preferences. With features like collaborative playlists and real-time interactions. This app aims to provide improved accessibility through seamless offline playback and to provide users with extensive customization options. Ultimately, the goal is to redefine how users interact with music, making the app an indispensable and transformative companion for music lovers all over the world.

Objectives

- Music Discovery Innovation: Launch a cutting-edge music discovery engine that uses advanced algorithms to provide users with personalized recommendations based on their preferences, thereby enhancing their overall music discovery experience.

- Community Building and Interaction: Encourage interaction and shared experiences among music enthusiasts by implementing features such as collaborative playlists, real-time commenting, and seamless integration with social media platforms.

- Improved Accessibility: Implement seamless offline playback functionality, allowing users from internet constraints and enabling them to listen to their favourite tunes wherever they go, regardless of connectivity.

- Customization and Personalization: Give users extensive customization options, such as the ability to create custom playlists, define listening preferences, and personalize the app's interface, to ensure a tailored and personalized user experience.

- User-Friendly Design: To improve the overall user experience, create a visually appealing and intuitive user interface that simplifies navigation, encompasses clear labels and instructions, and provides easily accessible playback controls.

- Continuous Improvement: Establish a framework for regular application updates and improvements, introducing new features, enhancements, and personalized recommendations over time to keep the user experience dynamic and engaging.

- Testing and Quality Assurance: Implement rigorous testing processes to ensure the application's dependability, performance, and security, resulting in a high-quality and seamless user experience.

- Licensing and Legal Compliance: Ensure legal compliance by establishing agreements with artists, publishers, production houses, and record labels, allowing the platform to stream music while adhering to copyright laws and avoiding legal complications.

1.2 Deliverables

1: Conceptual Design Phase

Deliverables:

- Detailed project plan defining the project's objectives, schedule, and deliverables
- To better understand user requirements and preferences, create thorough user personas and user stories.
- Prototypes and wireframes for the user interface of the app
- Documentation of the recommendations and algorithm used by the music discovery engine framework
- Proof of concept for the ability to play content offline

2: App Development

Deliverables:

- Completely functional app that has all the essential features, such as offline playback, individualized suggestions, shared playlists, music streaming, and adjustable settings.
- Social media platform integration to facilitate user communication and sharing

- User-friendly design with intuitive navigation, clear labels, and easily accessible playback controls

3: Testing and Deployment

Deliverables:

- Thorough testing of the app's performance, security, and functionality on a variety of platforms and devices
- Fixing bugs and finding solutions to any problems found

4: Continuous Improvement

Deliverables:

- Continual improvements and updates to the app based on input from users and usage statistics
- Adding new features, making tailored suggestions, and organizing community-building events
- Continuous testing and quality control are needed to keep the app stable and functional.
- Diversity of a vast range of artists, genres, and historical periods in the music library expansion

Additional Deliverables:

- User documentation and support: In-depth instructions and tutorials to assist users in navigating the app and resolving problems
- Management of communities: Actively interacting with users through forums, in order to create a lively community on forums, social media, and events
- Collaborations between artists and labels: To maintain legal compliance and increase the amount of music available on the app, secure licensing agreements
- Data analytics and insights: Gathering and examining user information to learn about user preferences and behaviour and to guide the creation of new apps.

1.3 Milestones

Our project uses an agile methodology, breaking down the development process into smaller, more manageable tasks that are grouped into each of the three main phases using a structured six-sprint approach. This approach allows for more frequent and incremental delivery of working software, enabling the team to adapt to changes and feedback more effectively.

Iterative Testing for Quality Assurance: Each phase of the plan, which begins with user registration and authentication and ends with a beta test with a small user base, emphasizes testing activities. By using an iterative process, any problems can be found and fixed early on, and the application is tested extensively before being deployed.

Phase 1: Requirements Gathering and Planning (6 sprints)

Sprint	Target Date	Features Covered	Testing Activities
Sprint 1	Week 1-2	User Registration and Authentication (Create account, log in, third-party authentication)	Develop test cases and conduct testing for user registration and authentication features
Sprint 2	Week 3-4	Edit Personal Details (Edit/update personal details (username, password, email))	Develop test cases and conduct testing for editing personal details features.
Sprint 3	Week 5-6	Music Streaming (Seamless and uninterrupted streaming of music tracks, play/pause, skip, forward, backward, volume control)	Develop test cases and conduct testing for music streaming features
Sprint 4	Week 7-8	Content Discovery (Personalized recommendations, curated playlists, genre-based browsing, popular charts)	Develop test cases and conduct testing for content discovery features
Sprint 5	Week 9-10	Playlist Management (Create, edit, manage playlists, add/remove songs, reorder tracks, share playlists)	Develop test cases and conduct testing for playlist management features
Sprint 6	Week 11-12	Like/Dislike (Like/dislike songs, albums, podcasts, playlists)	Develop test cases and conduct testing to like/dislike features

Phase 2: Development and Testing (6 sprints)

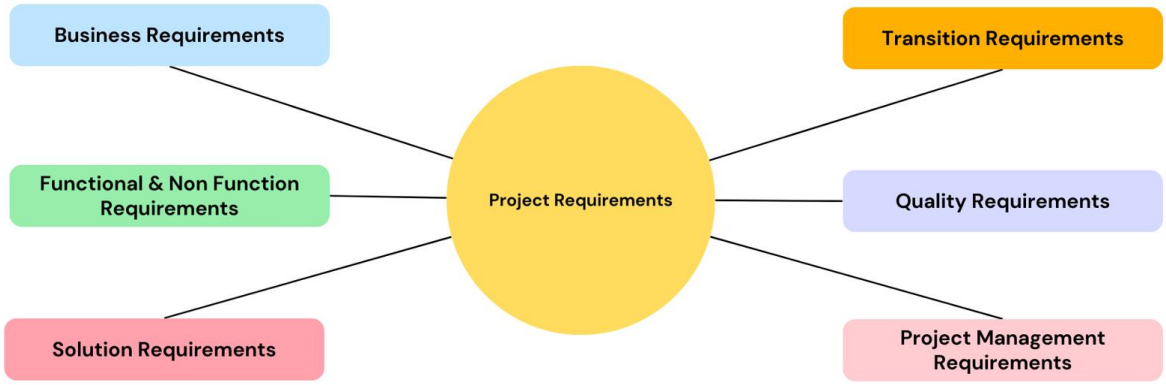
Sprint	Target Date	Features Covered	Testing Activities
Sprint 7	Week 13-14	Search Functionality (Resilient search capability using name, keywords, advanced options)	Develop test cases and conduct testing for search functionality
Sprint 8	Week 15-16	Social Integration (Sharing links, cross-platform sharing, opening links within the app)	Develop test cases and conduct testing for social integration
Sprint 9	Week 17-18	Premium User (Download songs, offline listening, ad-free access, exclusive podcasts)	Develop test cases and conduct testing for premium user features
Sprint 10	Week 19-20	Subscription (Monthly, quarterly, annual subscriptions, outsourced payment)	Develop test cases and conduct testing for subscription features
Sprint 11	Week 21-22	Customer Support (24/7 customer support, sign-in assistance, troubleshooting)	Develop test cases and conduct testing for customer support
Sprint 12	Week 23-24	Rigorous Testing Across Components	Conduct comprehensive testing across all components to ensure quality

Phase 3: Overall Testing and Deployment (2 sprints)

Sprint	Target Date	Features Covered	Testing Activities
Sprint 13	Week 25-26	Beta Testing with Select User Group	Gather feedback from a select group of users to identify and fix any remaining issues
Overall Deployment	Week 27-28	Deploy the application to a scalable and reliable cloud platform	Conduct performance testing and security testing, load testing to ensure the application is ready for production

2.0 Project Requirements

The project requirements for our music application development lay a solid and well-organized foundation, encompassing both functional and non-functional aspects to comprehensively guide the entire development process. Having formulated Business Requirements, Functional Requirements, Non-Functional Requirements, Solution Requirements, Transition Requirements, and Quality Requirements.



2.1 Business Requirements

Target Audience:

- A broad group of people, including music lovers, music supporters, and casual listeners.
- Suitable for users of all technical skill levels and ages.

Business Goals:

- Increasing user engagement and satisfaction.
- Generate revenue by selling premium subscriptions.
- Achieve market dominance as a leading music streaming platform.

2.2 Functional & Non Functional Requirements

Functional-

- User registration and authentication are secure.
- Profile management made simple.
- A large music library with streaming capability.
- Personalized recommendations and playlist curation.
- Browse by genre and view popular charts.
- Playlist administration with sharing options.
- For personalized recommendations, use the like/dislike functionality.
- Premium subscription for ad-free listening and offline listening.

Non-Functional-

- Fast streaming with minimal buffering.
- Scalability to support an expanding user base.
- Strong security measures are in place to protect user data.
- Data privacy regulations must be followed.
- High uptime and availability.
- Accessibility for a wide range of users.

2.3 Solution Requirements

The successful implementation of our music application is dependent on a carefully selected technology stack that ensures a consistent user experience. Our proposed solution includes the following key components:

Technology Stack:

- In crafting an engaging user interface, we will leverage industry-standard technologies:
- Front-end: Utilize HTML5, CSS3, and JavaScript to provide a visually appealing and interactive platform.
- Back-end: Employ (Specify back-end technology, e.g., Node.js, Django) for robust server-side functionality.
- Database: Implement (Specify database technology, e.g., MySQL, MongoDB) to efficiently manage user data, music metadata, and playlist information.

Content Acquisition: Identify and establish relationships with reputable music labels or aggregators to secure music content licensing agreements.

Content Delivery: Implement a CDN, such as Cloudflare, to distribute music content globally and ensure seamless streaming, handling fluctuations in user traffic without affecting performance or reliability.

Personalization and Recommendations: Develop machine learning algorithms using TensorFlow to analyze user preferences, listening history, and genre interests, providing personalized music recommendations.

Playlist Management: Design an intuitive interface using React for easy playlist creation, editing, and management. Utilize MongoDB for efficient storage and retrieval of playlist information.

Social Integration: Integrate with social media platforms using Facebook Graph API to enable music sharing, friend connections, and social music discovery.

Premium Subscription: Design and implement a premium subscription plan with Stripe for secure and seamless transactions. Implement offline playback technology, such as Spotify's offline mode, to allow premium users to enjoy offline listening and an ad-free experience.

2.4 Transition Requirements

The transition of our music application from development to deployment is a critical phase that requires careful planning and execution to ensure a smooth and successful rollout. It's crucial to outline the requirements that will facilitate a smooth shift from the development stage to deployment. The transition requirements include:

Deployment Plan

1. Create a detailed deployment plan that outlines the sequence of activities for releasing the application.
2. Define the deployment timeline, including timelines for various components and features.

Training

1. Provide comprehensive training sessions for end-users to familiarize them with the new application's features, interface, and navigation.
2. Conduct thorough training for the support team to equip them with the knowledge and skills to handle user inquiries and issues effectively.

Data Migration

1. Implement a rigorous data migration process to transfer user data, music metadata, and playlists from the existing system to the new application.
2. Prioritize data integrity and minimize the risk of data loss or corruption during the migration process.

User Communication

1. Maintain open communication with users throughout the transition phase.
2. Inform users about the progress of the migration process, potential downtime or disruptions, and new features.
3. Utilize multiple communication channels to reach a wide audience of users.

Support System

1. Establish a robust support system, including a helpdesk, user guides, and FAQs, to assist users during the transition and beyond.

2. Ensure the support team is readily available to address user queries and provide timely solutions.

Performance Monitoring and Feedback

1. Implement monitoring tools to track the application's performance and user interactions post-deployment.
2. Actively collect user feedback through surveys, reviews, and direct communication channels.
3. Use performance monitoring data and user feedback to identify areas for improvement.

Rollback Plan

1. Develop a contingency plan outlining the steps for rolling back to the previous version in case of unforeseen issues or critical problems.
2. Ensure the rollback plan is well-documented and easily executable to minimize downtime and disruption.

Security Measures

1. Implement robust security measures to safeguard user data and protect against potential vulnerabilities.
2. Conduct regular security audits to identify and address any potential security risks.
3. Maintain consistency and accuracy in all security documentation.
4. Continuously monitor and address any potential security vulnerabilities identified in the application codebase or infrastructure.

2.5 Quality Requirements

- Quality Assurance: Implement a comprehensive quality assurance (QA) plan to ensure the music application meets all requirements and standards.
- User Acceptance Testing: Conduct user acceptance testing (UAT) with representative users to gather feedback and identify any usability issues.
- Performance Testing: Perform performance testing to ensure the application can handle expected user load and maintain responsiveness.
- Security Testing: Conduct security testing to identify and fix any potential security vulnerabilities.
- Scalability Testing: Ensure the application can scale to accommodate future growth in user base and usage.
- Reliability Testing: Implement reliability testing to guarantee the application remains functional and available under various conditions.

2.6 Project management requirements

- To guarantee continuous communication and progress assessment, conduct weekly updates and monthly reviews on a regular basis.
- Organize sprints every two weeks using an Agile methodology to promote an iterative and adaptive approach to development.
- Keep thorough records of every procedure to improve openness and give a thorough grasp of the project's operations.

3.0 Acceptance Criteria

1. User Registration and Authentication- The user should be able to create an account and log in to it. A third-party system performs user authentication.

2. Edit Personal Details- The user should be able to edit or update his personal information such as his user's name, password, and email address.

3. Music Streaming- Users should be able to listen to their favourite songs, albums, playlists, and podcasts via a music application that provides seamless and uninterrupted streaming of music tracks. The music app should also allow the user to play, pause, skip, navigate forward and backward, increase and decrease volume of a song, playlist, or album.

4. Content Discovery- To assist users in discovering new music, the platform should provide a variety of features such as personalized recommendations based on their listening history, curated playlists, genre-based browsing, and popular charts.

5. Playlist Management- Users should be able to create, edit, and manage their playlists, including adding and removing songs, reordering tracks, and sharing playlists with others.

6. Like/Dislike- A song, album, podcast, or playlist should be able to be liked or disliked by the user.

7. Search Functionality- The music app should have a robust search capability for finding songs and artists by name, keywords, and advanced search options.

8. Social Integration- The platform should allow users to connect with friends by sharing song, album, podcast, or playlist links. This link can be shared on a variety of platforms. When the user clicks on the link, our music app will open the corresponding songs, albums, podcast, or playlist.

9. Premium Users- Premium users should be able to download the song and listen to it offline, as well as have access to exclusive podcasts.

10. Subscription- The user should be able to choose between monthly, quarterly, and annual subscriptions. The subscription payment is handled by a third-party system.

11. Customer support- The user should be able to contact customer support at any time if he is having trouble signing in or with any other issue. Customer service should be available 24 hours a day, seven days a week.

4.0 Project Assumptions & Project Constraints

Project Assumptions:

Reliability of Third-Party Authentication:
Assumption: The third-party authentication system (e.g., Google, Facebook) will be dependable and ready for integration with minimal downtime.

Continuous Internet Connectivity:
Assumption: Users will have constant internet access for streaming music and using online features. Premium users will have access to offline functionality.

Data Security Compliance:
Assumption: Third-party authentication and payment systems comply with applicable data security standards and regulations.

User Interface Device Compatibility:
Assumption: The application is intended to be compatible with a wide range of devices and operating systems, ensuring a seamless user experience across different platforms

Licensing of Content:
Assumption: All necessary licenses for streaming music content have been obtained, and the application complies with all applicable copyright laws and regulations.

User Engagement:
Assumption: Users will actively engage with the application's social features, such as playlist sharing and liking/disliking songs.

Customer Service is available:
Assumption: Customer service is available and responsive to user inquiries and issues 24 hours a day, seven days a week.

Project Constraints:

Budget Constraints
Navigating project finances efficiently is crucial for the success of our music application development. The project must adhere to the defined budget, ensuring that every expenditure aligns with the project's objectives. While we strive to optimize resource allocation, the limited availability of additional funds necessitates a cautious approach to resource management.

Timeline Constraints
Meeting the project's specified deadline is paramount to maintaining stakeholder satisfaction and market competitiveness. We must meticulously plan and execute each stage of the project, from development to testing and deployment, to ensure timely completion. Regular project tracking will enable us to monitor progress, identify potential delays, and take corrective action promptly.

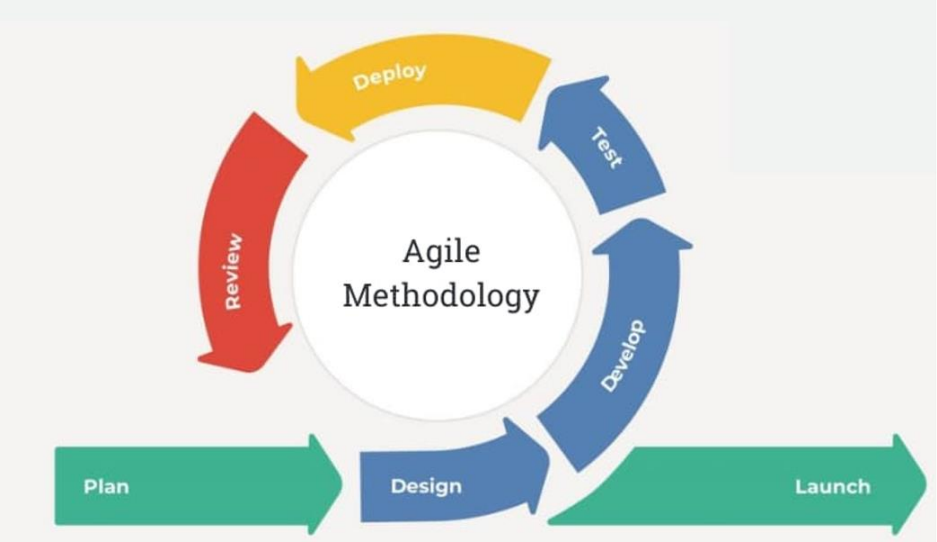
Resource Constraints

Assembling a competent team with the necessary expertise in development, testing, and marketing is essential for project success. Limited availability of skilled resources, both in terms of personnel and technology, could hinder the project's capabilities and affect its ability to meet user expectations.

Scope Creep Constraints
The project must remain focused on the core requirements and avoid unnecessary expansion beyond the initial scope. Expanding the scope could strain resources, compromise the project's overall feasibility, and lead to delays or quality issues.

Risk Tolerance Constraints
The project team's risk tolerance will influence the approach to innovation and the adoption of new technologies. While we must carefully consider risks and implement mitigation strategies, we should also strive to strike a balance between innovation and risk management to deliver a cutting-edge application while ensuring its stability and success.

5.0 Scope Plan



Scope Planning Approach

To manage the scope of the Music App effectively, an Agile scope planning approach will be employed. This involves breaking down the project into smaller, more manageable increments called sprints. Each sprint focuses on a specific set of tasks, enabling the team to deliver working software incrementally and gather feedback from stakeholders throughout the development process.

The iterative nature of sprints allows for continuous adaptation and improvement. As the team progresses through each sprint, they receive feedback from stakeholders and users, which can be used to refine the product and address any emerging requirements. This

iterative approach ensures that the project remains aligned with the evolving needs of users and stakeholders.

Agile Scope Management

A key aspect of agile scope management is the establishment of a clear and transparent backlog. This backlog serves as a prioritized list of tasks and features that need to be addressed in each sprint. The backlog is continuously refined and updated based on stakeholder feedback and the team's assessment of priorities.

Regular stakeholder meetings and daily stand-up sessions are essential for maintaining open communication and transparency. These meetings allow the team to discuss progress, address any roadblocks, and ensure that the project remains aligned with the overall goals and objectives.

Sprint reviews provide an opportunity to showcase the work completed during each sprint and gather feedback from stakeholders. This feedback is invaluable in evaluating the product's progress and identifying areas for improvement.

Azure DevOps Integration

Azure DevOps will serve as the central platform for managing the Music App project's lifecycle. This integration provides a comprehensive set of tools for backlog management, task tracking, version control, and continuous integration/continuous delivery (CI/CD) pipelines.

Backlog management enables the team to organize and prioritize tasks, ensuring that all important work is accounted for and tracked. Task tracking provides visibility into the progress of individual tasks, allowing the team to identify potential bottlenecks and manage resource allocation effectively.

Version control facilitates collaboration and ensures that changes to the codebase are tracked and retrievable. This enables the team to revert to previous versions if necessary and maintain a consistent codebase throughout the development process.

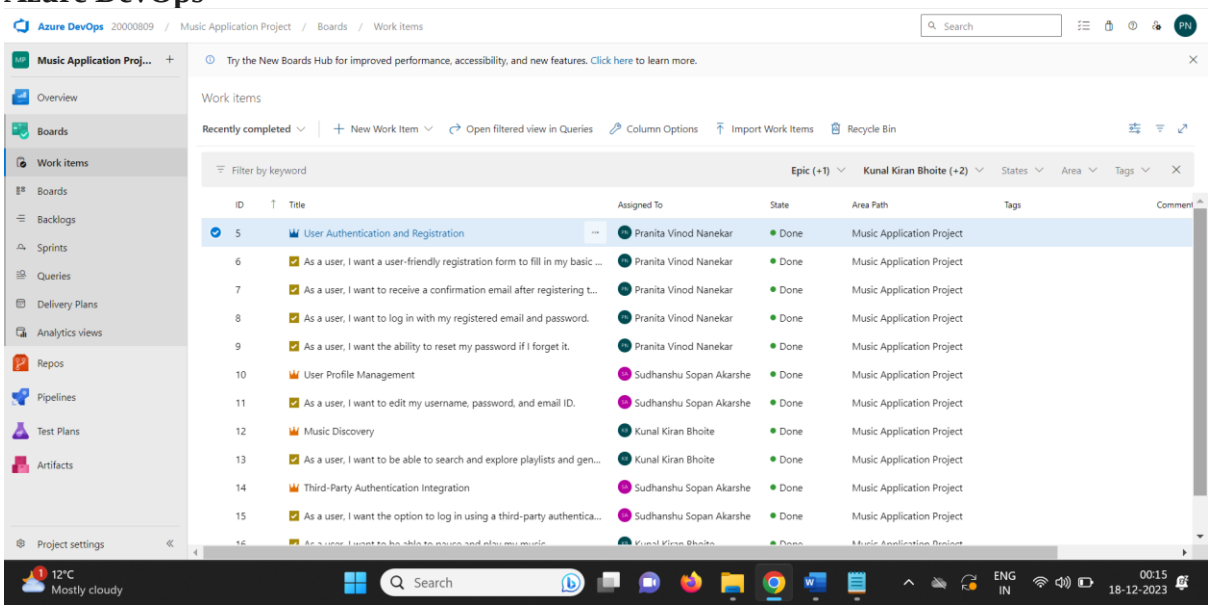
CI/CD pipelines automate the process of building, testing, and deploying the application. This automated approach streamlines the development workflow, reduces errors, and ensures that new features and bug fixes are deployed quickly and efficiently.

Integrating Azure DevOps into the Music App project will provide a centralized platform for managing the project's lifecycle and enhancing collaboration among team members. This integration will also improve productivity, ensure efficient project delivery, and reduce the risk of errors.

Assignment on User Stories

Assigning user stories to team members according to their qualifications and experience is an essential next step in the agile development process. By strategically allocating tasks, each team member's strengths are maximized, resulting in increased productivity and efficiency.

Azure DevOps-



Developer Responsibilities:
"User can play music" is an example of a user story.
The person best suited to take on tasks related to implementing the functionality that allows users to play music seamlessly is a developer who has experience with both coding and system functionality.

Designer Responsibilities:
The user can make a new playlist, is an example of a user story.
The task of designing and implementing the playlist creation user interface might fall to a designer who specializes in making user interfaces that are both visually appealing and intuitive.

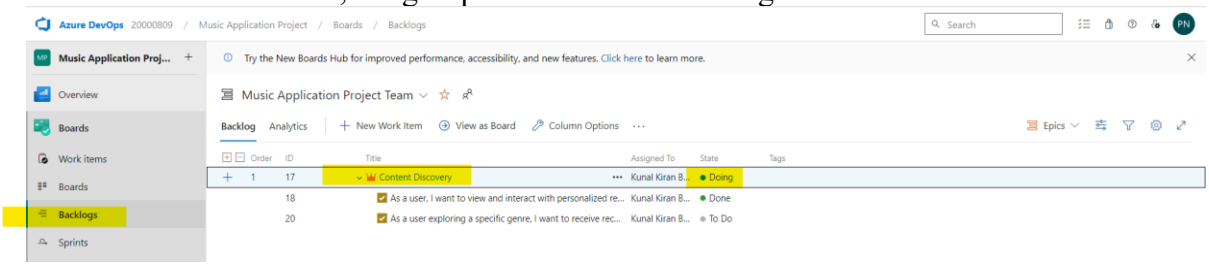
QA Engineer Duties:
Example "Ensure music playback is bug-free" is the user story.
The responsibility of validating and confirming that the music playback feature operates without any problems or glitches falls to quality assurance engineers.

Integration Specialist Responsibilities:
Example User Story: "Integrate music streaming API"
An integration specialist, familiar with connecting external services, could be assigned to integrate a third-party music streaming API into the application.

Progress tracking with Azure DevOps
The team uses Azure DevOps or a comparable tool for progress tracking in order to keep project visibility and guarantee on-time, within-budget completion.

User Story Backlog
Every user story is arranged in a backlog and given a priority according to dependencies and

business value. As a result, the group can concentrate on urgent tasks first.



Assigning Work Items
Using their skills and knowledge, team members select tasks or user stories from the backlog. Transparency in assignment creation helps to maintain a balanced workload and prevent people from being overworked.

Task Boards and Visualization Boards
Task boards or boards are provided by Azure DevOps to help visualize the work flow. The team can monitor each user story's progress on these boards and determine whether it needs more work or is finished.

Burndown Charts
In Azure DevOps, burndown charts offer a visual depiction of the project's advancement. This aids the team in determining whether they are on schedule to finish the user stories in the allotted amount of time.

5.1 Change Management Process

Activity	Description	Responsible
Submit Change Request	The change request should be submitted in writing and include the following information: a description of the proposed change, the justification for the change, the impact of the change on the project scope, schedule, and budget, and the proposed mitigation strategies for any negative impacts.	Change Requestor
Review Change Request	The change request will be reviewed by the project manager and stakeholders.	Project Manager, Stakeholders
Approve/Reject Change Request	The project manager and stakeholders will determine whether to approve or reject the change request.	Project Manager, Stakeholders
Estimate Testing Effort	The project manager will estimate the testing effort required for the approved change.	Project Manager
Schedule Testing	The project manager will schedule the testing for the approved change.	Project Manager
Conduct Testing	The approved change will be tested by the testing team.	Testing Team
Analyze Test Results	The testing team will analyze the test results and provide feedback to the project manager.	Testing Team

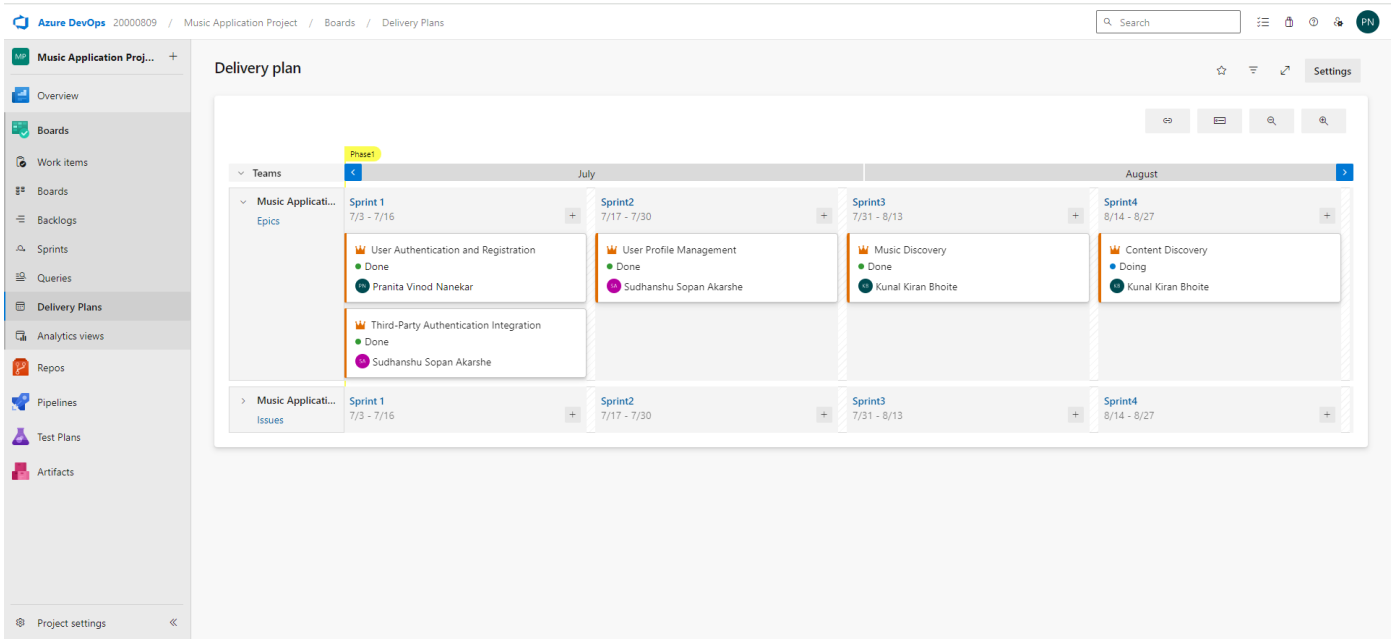
Update Project Scope, Schedule, and Budget	If the testing results indicate that the change requires additional development or testing, the project scope, schedule, and budget will be updated accordingly.	Project Manager
Implement Approved Change	The approved change will be implemented by the developers.	Developers
Monitor and Update Change Log	The change log will be updated to reflect the approved change and any associated testing activities.	Project Manager
Communicate Change to Stakeholders	The change request, approval/rejection decision, and updated project scope, schedule, and budget will be communicated to all stakeholders.	Project Manager

Change Management Roles and Responsibilities

- Project Manager: Responsible for managing the change request process.
- Stakeholders: Responsible for reviewing and approving change requests.
- Developers: Responsible for implementing approved changes.

6.0 Schedule Plan

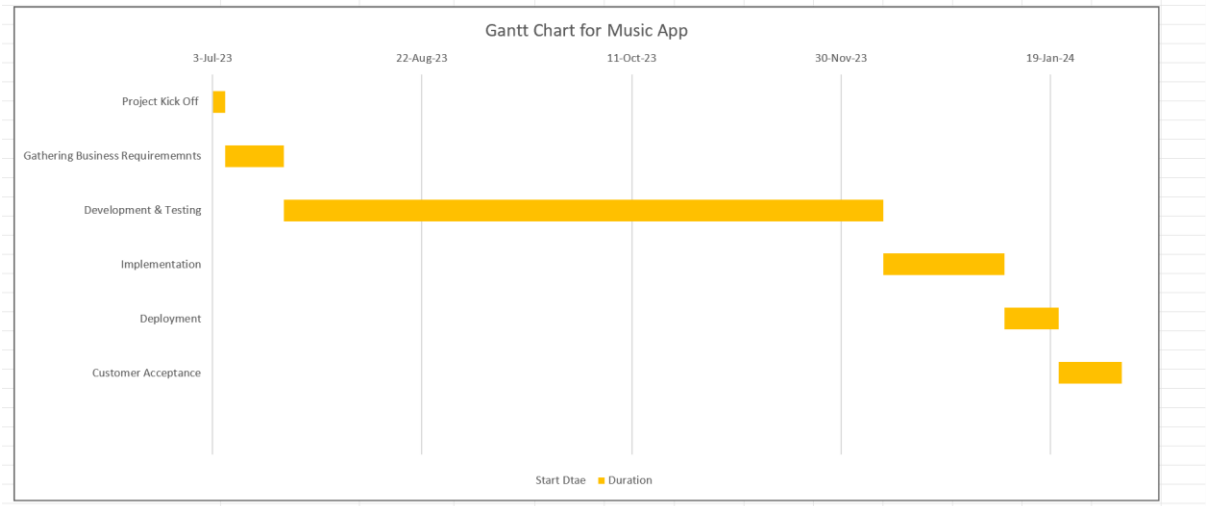
Azure Devops - Delivery Plan



The carefully thought-out Azure DevOps Delivery Plan is depicted in the diagram, in line with the objectives and scope of the project. It describes the different project phases and gives a visual depiction of sprint details. This Azure DevOps delivery plan is a tactical schedule that takes into account both the longer and shorter sprint durations, as well as the overall phases, to guarantee a coordinated and goal-oriented project lifecycle. The tool makes thorough planning, monitoring, and execution easier, which encourages a smooth and effective development process.

6.1 Gantt Chart

Task	Start Dtae	End Date	Duration
Project Kick Off	3-Jul-23	6-Jul-23	3
Gathering Business Requirememnts	6-Jul-23	20-Jul-23	14
Development & Testing	20-Jul-23	10-Dec-23	143
Implementation	10-Dec-23	8-Jan-24	29
Deployment	8-Jan-24	21-Jan-24	13
Customer Acceptance	21-Jan-24	5-Feb-24	15



7.0 Project Budget and Earned Value Management Analysis

Project Budget and Earned Value Management Analysis for Music App Development. It is the financial aspects associated with the development of application. It includes the budget, actual cost to access the project progress and efficiency. Below are few steps involved in estimating project budget.

Scope of Your Project:

- This includes all the fundamental features expected in this music application such as music streaming, music sharing, social integration.
- Mention all the platform on which application is expected to be available such as iOS, Android.

Timeline:

- Mention a timeline for development, it should be set based on the complexity of the features and releases.

Estimate Resource Costs:

- This should include all the cost of personnel involved, including developers, designers, testers, and project managers, based on their hourly/monthly rates and estimated time commitment.

- Cost of technology should include any software licenses, cloud hosting fees, API integrations, etc.
- Budget for marketing materials, app store fees, and legal compliance.

Earned value management (EVM) –

Planned value – This is value estimated based on the completion status of the task associated with the project development.
Earned value – This value is represented for the actual progress made in the project.
Cost performance index- Ratio of earned value to actual cost gives cost performance index, it is used to assess the cost efficiency of project.
Schedule Performance Index – Ratio of earned value to planned value gives schedule performance index, it is used to assess the schedule efficiency of the project.

7.1 Project cost budget

We have made few assumptions related to team size, technologies used and app features to provide cost estimates of the project.

Project resources	**Value**	**Cost ($/Annual)**
Development team	5 developers 2 UI/UX designer 2 QA engineers 1 Project manager	80,000 per developer and 100,000 project manager.
Marketing and support	2 members	100,000
Tools and license	IDE, React, MongoDB, OpenAPI, Jira, Adobe XD, Jenkins, Selemium	50,000
Infrastructure	Cloud hosting cost	20,000
	Total Estimated cost:	**990,000**

Planned value – here assuming an even distribution over the year i.e. $990,000/12 = **$82,500 per month**

Earned value – Consider 20% of development work is done in first 3 months.

Hence, earned value will be $990,000 * 20% = **$198,000**

8.0 Project communication plan

A Project Communications Plan is an important part of project management as it outlines how information will be communicated to various stakeholders throughout the project. Stakeholder Analysis is a part of this plan and involves identifying, analysing, and prioritizing individuals or groups with an interest in the project. Project communication plan includes various analysis such as stakeholder analysis, below is the sample framework for project communication plan for app development project.

Project Communications Plan

1. Objectives

Communication is key, hence it the primary objective of communication plan. Goal is to maintain efficient and effective communication among project team members and stakeholders, fostering a collaborative and transparent environment throughout the initiation stage of the music app development project.

2. Stakeholder Analysis

Stakeholder analysis is crucial part of this plan. Primary goal here is to Identify and analyse key stakeholders, considering their interests, influence, and impact on the project.

Executive Team – This team may include Chief Executive Officer (CEO), Chief Operations Officer (COO), Chief Financial Officer (CFO) and further executives. Interest and influence of executive team in music application development will be high. Also, the communication of project progress, grievances discussion and strategic decisions will be discussed on regular basis.

Development Team – Interest of development team will be high, as this team will be responsible to perform the development of application. Communication will on regular time period which may include development plan, sprints, stories, task and progress reports.

Marketing Team – Interest and influence of marketing team will moderate in this project. Roles such as market research for the music application, brand development, identifying target audience for the application, user acquisition and retention strategies will be assigned to this this. Communication may include the progress report, updates on features for promotional activities.

Users/Clients – They include the audience to whom music application will be presented. They will play most important part and will high interest in the application. Users will download the application on their devices, they will pay and subscribe to premium plan. Communication to users may include updates on new features, feedback on sessions, extraordinary request and support.

Regulatory Bodies – They include government agencies or organization who will be responsible for overseeing and regulating aspects of the project. Their involvement ensure the

music app complies the laws and regulation and hence their influence will be high. Communication may include the compliance updates and legal requirements.

3. Communication Channels

Communication channels mainly include the appropriate channels assigned for communicating with each stakeholder group.

- Regular Team Meetings - Weekly meetings for the development team to discuss and plan progress.
- Email Updates – This helps to keep the executive and marketing teams on the same page about the progress of application.
- User Feedback Sessions – This may include Bi-monthly sessions to gather input such as review and comments from users.
- Compliance Reports - Regular reports to regulatory bodies as required.

4. Communication Methods

Specify the methods of communication for different types of information.

- Formal Reports - For executive team and regulatory communications.
- Informal Updates – Updates over E-mail for team members and marketing team.
- Surveys and Feedback Forms - For collecting user opinions and preferences.

5. Frequency and Timing

Plan should also define the frequency and timing of communication events.

- Team Meetings- Meeting can be held on weekly basis, particularly on Mondays at 10 am.
- Email Updates – Emails about instructions or grievances could be done bi-weekly on Wednesdays.
- User Feedback Sessions – These feedbacks can be communicated bi-monthly on Fridays.
- Compliance Reports - Monthly before regulatory deadlines.

6. Escalation Procedures

Set of steps and procedures needs to be established for handling issues or concerns that may arise during the project development.

- Team Issues – Issues with a team should be escalated to Project Manager.
- Technical Challenges – These concerns should be escalated to the Technical Lead.
- Regulatory Concerns - Escalate it to Legal Advisor.

7. Review and Update

Reviews and updates on regular intervals for the communication plan to ensure its effectiveness and make necessary updates based on project progress and changes.

- Review Interval – Project development review could be setup every two weeks during team meetings.
- Update Criteria - Any significant project changes or milestones achieved should be communicated immediately as confirmed.

This Project Communications Plan serves as a comprehensive guide for ensuring effective communication within the music app development project, helping to keep all stakeholders informed and engaged throughout the initiation stage.

9.0 Comprehensive Risk Plan

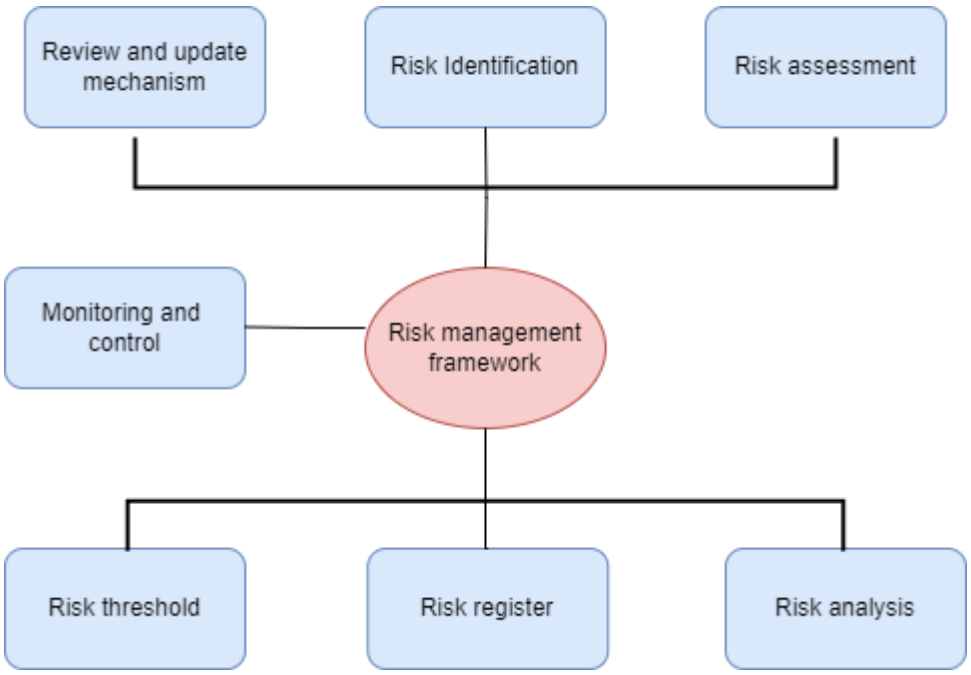
A Comprehensive Risk Plan is crucial in managing uncertainties and potential challenges during the development of a music application. Comprehensive risk plan includes multiple aspects such as risk analysis, a risk register, and guidelines for quantitative and qualitative risk assessment, risk tolerances, and risk thresholds. Primary goal of comprehensive risk plan lies in identifying, analysing, and managing risks to ensure application development success.

Risk Management Framework - This framework is used to define the overall methods used for managing risks in the application development. This includes the processes, tools, and techniques that will be utilized to identify, assess, respond to, and monitor risks throughout the project lifecycle.

- **Risk Identification-** Identification and listing out all the potential risks that may occur while development of music application. These risks may relate to technology, resources, stakeholders, regulatory compliance, and external factors. Brainstorming, interviews and reviews are few of the techniques that can used to identify risk associated to project
- **Risk Analysis-** Assessment of the risk can be done by two factor – quantitative assessment and qualitative assessment.
    1. Quantitative Risk Assessment-
       In this technique, numerical values are assigned to risk where ever possible (e.g., estimating the cost impact of potential delay in application launch). Tools used for quantitative risk assessment are decision trees, simulations, or sensitivity analysis.
    2. Qualitative Risk Assessment-
       In this technique, qualitative scale is used to assess risks based on probability. Scale such as low, medium or high considering factors like user adoption

challenges. Consider factors such as project complexity, stakeholder influence, and historical data.

- **Risk Register-** Maintaining consolidated records of potential risks helps to completely assess the risk factors. Records can be maintained by assigning a unique ID, description, impact and risk owner to it throughout the music app development.



- **Risk Tolerances and Risk Thresholds-** This includes the certain levels of risk that the project team is willing to tolerate.

  Risk Tolerances - The acceptable level of variation in project objectives.

  Risk Thresholds – A threshold level at which a risk becomes unacceptable, triggering predefined responses.

- **Monitoring and Control -** Manage identified risks in the music application development project. This involves continuous tracking, updating the risk register, implementing mitigation strategies, and ensuring that risk responses align with project objectives and timelines**.**
- **Review and Update Mechanism –** With regular plan, reviewing and update the risk plan is also necessary. Regular assessment should be done for the effectiveness risk management strategies of app development. If changes required, update the plan in response to changes in project scope, objectives.

This plan outlines a structured framework for identifying, evaluating, and mitigating potential risks throughout the music application development project. Drawing upon best practices in

project management, this approach empowers the team to proactively address challenges, thereby ensuring the project's ultimate success.

9.1 Risk Register

Risk register plays important role in comprehensive risk plan, it is a consolidated register of potential risk and its mitigations. It is helpful during the development of music app. It includes various aspects of risk as below-

Risk Type – Broad type to which risk can be associated.

Risk ID – A unique ID to identify the risk.

Description – Brief description of the risk.

Probability – Likelihood of the risk occurring (1- Low and 5- High).

Impact – Potential consequences if the risk occurred in future (1- Low and 5- High).

Mitigation – Strategies to lower or eliminate the risk.

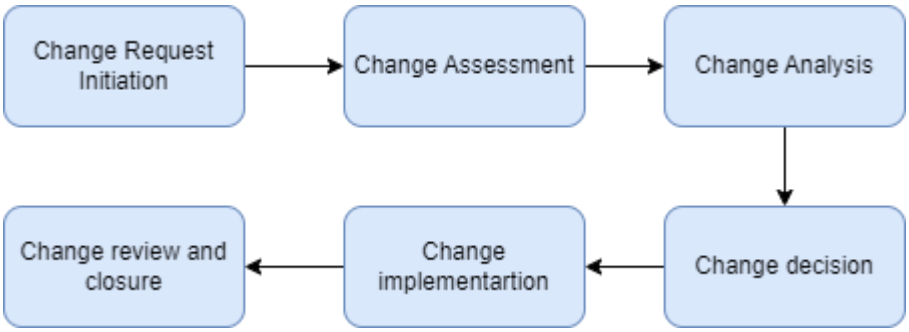
Below are few risks that can occur during development of music application -

Risk Type	**Risk ID (Unique ID)**	**Description (Summary of risk)**	**Probability Rate 1(LOW) to 5(HIGH)**	**Impact Rate 1(LOW) to 5(HIGH)**	**Mitigation (Measures to lower or eliminate risk)**
Technical	1001	Incompatibility with certain operating systems	4	5	Perform complete testing and compatibility test during development phase
Market	1002	Changing UI as per user preferences for music	5	2	Keep close note of customers feedback and market trends
Resource	1003	Executive/ senior technical person leaving the project	1	5	Cross train team members and maintain documentation if development work

Regulatory	1004	Changes in law of customer data gathering	4	5	Regularly review data protection laws
Integration	1005	Application facing issue for third party API integration	2	2	Establish strong partnership and develop reliable application
Security	1006	Security breach	1	5	Review and update security system on regular intervals

9.2 Change process

Since the beginning of music application development till the application is live and present in operating state, there are many factors that demands to change the planned process. This change request can be implemented smoothly if a series of steps are performed. These steps include change initiation, assessment, analysis, decision, implement and review and closure.



In music application development, change process may be used to implement an additional feature in the app based on stakeholder's request. The change process will be implemented as follows-

- **Change request initiation –** As per the approval of stakeholder, the change request is initiated. In this step, reason and effect after implementing this change are noted
- **Change assessment –** In this step, the request is assessed end to end by technical team to confirm its feasibility to implement this feature in the app.
- **Change analysis –** In this step, change is analysis and factors such as impact on various other features, cost estimation and risk assessment are done.
- **Change decision –** After the assessment and analysis phase, final decision is made to approve, reject or defer the change request.

- **Change implementation –** If the request is approved in earlier step, technical team proceeds with the implementation of this feature, monitoring and adjustments as per requirement.
- **Change Review and closure –** If the change is implemented successfully and no issues occurred under monitoring phase, the change is well documented with all necessary adjustments and change is formally closed.

Change Request ID	Date of request	Requester	Description	Justification	Impact	Budget	Decision	Approval Date	Approver
CR3001	11-12-2023	Project manager	Add new feature for collaborative playlist creation	Competitive analysis shows a growing demand for collaborative playlist features in music apps, and it aligns with our user engagement strategy.	Low	Additional development cost	Approved	13-12-2023	Project manager, product owner

10.0 Stakeholder Definition of Done Template

Project Name: Music Application Development

Version: 1.0.0

Date: 15-12-2023

Objective: To establish a clear and agreed-upon definition of done for stakeholders, ensuring alignment and understanding of project deliverables at the initiation stage.

Stakeholder Involvement:

- Executives
- Chief Executive Officer (CEO)
- Chief Operations Officer (COO)
- Chief Financial Officer

Functional requirements: Functional requirements for application development will be the essential features that should meet customers requirement. Below is the list of few functional requirements-

- User Registration and Authentication - Users should be able securely create accounts and log in
- Music Library -Application should provide organized music library including songs, albums, podcasts.
- Users should be able to search, browse, and discover music easily.

- Playlist Creation and Management -Users should be able to create, edit, manage and share playlists.
- Recommendation Engine - Implement a recommendation system for personalized song suggestions based on user preferences and listening history.

Non-functional requirements: Non-functional requirements define the characteristics of a music app that are not directly related to its functionality but are crucial for its overall performance, usability, and reliability. Below is the list of few nonfunctional requirements-

- Performance - The app should respond quickly to user actions, providing a seamless and responsive experience. Also, app should be scalable to handle a growing number of users and an expanding music library without compromising performance.
- Reliability - App should have high availability, minimizing downtime for maintenance or updates. Also, it should be able to handle system failures gracefully without disrupting user experience.
- Security - Data encryption should be used to secure user data during transmission and storage. And secure authentication mechanisms to protect user accounts.

Documentation: Documentation is another key aspect while development of the application. Various points need to be well documented to keep the development process smooth. Below are some the points that should be documented.

- Requirements Document
- System Architecture Documentation
- Database Design Document
- UI Design Document
- Technical Specifications
- API Documentation
- Testing Documentation
- Deployment Documentation
- Configuration Management
- User Documentation
- Security Documentation
- Maintenance Documentation
- Compliance and Legal Documentation
- Change Management Documentation
- Project Management Documentation
- Communication Plan
- Training Documentation

Testing and Validation: This Definition of Done template ensures that testing and validation processes are aligned with Agile principles, emphasizing continuous integration, measurable acceptance criteria, and a systematic approach to defect management for maintaining the music app's overall quality and performance.

- Types of Testing - Various testing methods can be used during different phases of development of this app.

- Unit Testing - It is mainly used to verify individual components (e.g., playback feature, user authentication). Acceptance Criteria - All units pass predefined test cases, ensuring functionality and correctness.

- Integration Testing -It is mainly used to validate the interaction between integrated components (e.g., seamless transition between playlist and playback). Acceptance Criteria - Integration points function as expected, with no disruptions to overall app flow.

- Defect Tracking and Resolution-

- Defect Logging - Use a dedicated issue tracking tool such as Jira to log and categorize defects. Include details such as the severity, steps to reproduce, and current status.

- Defect Resolution - Define priority levels for defects (e.g., critical, major, minor) to determine resolution urgency. Establish a clear process for developers to address and resolve defects promptly.

Deployment and release: This Definition of Done template ensures that the deployment and release processes for the music app align with best practices, including clear communication, effective monitoring, and continuous improvement for a successful and user-friendly release.

During deployment series of steps needs to be followed in order to complete the process smoothly. Below are the brief steps -

- Complete code review and testing phases.
- Generate production-ready builds and perform final verification.
- Deploy the app to staging for a final validation environment.
- Conduct user acceptance testing (UAT) on the staging environment.
- Finalize the deployment package for production.
- Execute the production deployment in a phased manner, minimizing downtime.

- Communication Protocols for Stakeholders-Notify stakeholders (internal teams, users) of the upcoming release well in advance. Clearly communicate release notes, highlighting new features, improvements, and any potential impacts. Different channels such as email, in-app notifications should be used for communication. Also, feedback mechanism on each new release should be implemented.
- Overall Acceptance Criteria - Upon successful deployment without major disruptions to app functionality as per the outlined process and timeline the deployment can be marked under acceptance. Stakeholders receive clear and timely communication regarding the release and critical issues are addressed promptly.

After highlighting all the above points with stakeholder and accepting the criteria for successful project initiation, we take signatures of stakeholder.

Stakeholder Signatures:

[Signature] [Date]

[Signature] [Date]