ASSIGNMENT 2 FRONT SHEET

Qualification	**BTEC Level 4 HND Diploma in Computing**		
Unit number and title	Unit 7: Software Development Life Cycle		
Submission date		**Date Received 1st submission**	
Re-submission Date	14/12/2023	**Date Received 2nd submission**	
Student Name	Pham A Quan	**Student ID**	BH01135
Class	IT0604	**Assessor name**	Dinh Van Dong
Student declaration I certify that the assignment submission is entirely my own work and I fully understand the consequences of plagiarism. I understand that making a false declaration is a form of malpractice.			
		Student's signature	P. Quan

Grading grid

P5	P6	P7	M3	M4	M5	M6	D3	D4

❐ **Summative Feedback:**		❐ **Resubmission Feedback:**
Grade:	**Assessor Signature:**	**Date:**
Internal Verifier's Comments:		
Signature & Date:		

Tables Of Contents

Table Of Figuers

INTRODUCTION

The Software Development Life Cycle (SDLC) is a structured approach to software development that guides the process from planning to deployment and maintenance. For the Tune Source project, following the SDLC ensures a systematic development process. The phases include requirements gathering and analysis, system design, implementation and coding, testing, deployment, and maintenance. Each phase contributes to the development of the Tune Source application, from understanding project requirements to delivering a high-quality and user-friendly product. Effective project management and collaboration are essential throughout the SDLC to ensure project success.

P5: Undertake a software investigation to meet a business need.

a. Reqirement

The Tune Source company has approved the project of building a new website for this job. Some of the final steps in bringing the project to market are outlined in this study. To begin, I must define the stakeholders, their roles, and their interests in the case study. Then, go over some of the methods that will be used to meet the requirements. Provide a requirements traceability matrix to track all requirements during the website development process. Use a combination of structural and behavioral modeling tools to analyze the requirements. After that, create a mockup of the interfaces to demonstrate the website's functions. Finally, manage the website according to some guidelines management attributes.

b. Distinguish functional requirements and non-functional requirements

1. functional requirements definition.

Pic 1: functional requirements

Functional requirements define the specific behaviors, features, and capabilities that a system or software application should have. They describe what the system should do, how it should behave, and what actions users should be able to perform. Functional requirements should be clear, concise, and written in non-technical language. They should identify the actors interacting with the system, specify system features, define input and output, include business rules, handle errors and exceptions, consider performance and scalability, prioritize requirements, validate them with stakeholders, and maintain traceability. Functional requirements serve as a basis for system design, development, testing, and validation, ensuring a common understanding among stakeholders.

2. non-functional requirements definition.

Pic 2: non-functional requirements

Non-functional requirements describe the qualities and constraints that define how a system should perform or behave. They focus on aspects such as performance, reliability, security, usability, scalability, and maintainability. Non-functional requirements specify criteria such as response times, system availability, security measures, user experience, scalability options, ease of maintenance, compatibility, compliance with regulations, performance efficiency, data management, interoperability, disaster recovery, and legal/ethical considerations. These requirements ensure the overall quality and performance of the system, complementing the functional requirements. They provide guidelines for evaluating and assessing the system's capabilities and behavior.

3. Advantages and Type of FR and N-FR

	Advantages	Type
functional requirements	• They make that the software program complies with all applicable laws and regulations. • They detail the software's quality characteristics. • They ensure the software system is reliable, available, performant, and scalable • They contribute to the development of the software system's security policy. • They guarantee a positive user experience, make it simple to use the program and keep costs down	Transaction Handling, Business Rules, Certification Requirements, Reporting Requirements, Administrative functions, Authorization levels, Audit Tracking, External Interfaces, Historical Data management, Legal and Regulatory Requirements
non-functional requirements	• They make that the software program complies with all applicable laws and regulations. • They detail the software's quality characteristics. • They ensure the software system is reliable, available, performant, and scalable. • They contribute to the development of the software system's security policy.	Usability requirement, Serviceability requirement, Manageability requirement, Recoverability requirement, Security requirement, Data Integrity requirement, Capacity requirement, Availability requirement, Scalability requirement, Interoperability requirement, Reliability requirement, Maintainability requirement, Regulatory requirement

	- They guarantee a positive user experience, make it simple to use the program and keep costs down	

4. Distinguish functional requirements and non-functional requirements

Pic 3: Distinguish functional requirements and non-functional requirements

Functional requirements	Non-functional requirements
• Describes the actions and functions that a system or application software needs to perform to meet user needs. • Defines what the system or software application does, how it works, and the activities users can perform. • For example, user registration, product search, shopping cart management, payment processing, etc. • Focus on what the system or software application needs to do	• Describes the elements, characteristics, and constraints of how the system works, rather than the core content of the functionality. • Focus on aspects such as performance, reliability, security, user experience, scalability, and maintenance. • For example, response time, fault tolerance, data security, user experience, scalability, etc. • focus on how the system works and elements that are not related to the main content of the function

c. Functional requirements and non-functional requirements in the Tune Source project

1. Functional requirements in the Tune Source project

User Registration:	• Users should be able to create an account by providing their email address, username, and password. • The system should validate the uniqueness of the email address and username to prevent duplicate accounts. • Users should receive a confirmation email to verify their account and activate it.

Song Upload:	• Authenticated users should have the ability to upload songs to the platform. • The system should support various audio file formats (e.g., MP3, WAV, FLAC). • Uploaded songs should be stored securely on the server and associated with the user who uploaded them. • The system should validate uploaded songs to ensure they meet specified file size and format requirements.
User Profile:	• Users should have customizable profiles where they can add a profile picture, bio, and other personal information. • The system should provide privacy settings to control the visibility of user profiles and personal information. • Users should be able to view and edit their profile information, including the option to change their profile picture.

2. Non-functional requirements in the Tune Source project

Performance:	• The platform should be highly responsive, providing quick loading times for song playback, search results, and other interactive features. • The system should be able to handle a large number of concurrent users without significant performance degradation. • Streaming should be smooth and uninterrupted, minimizing buffering and latency.
Security:	• User authentication and authorization should be implemented to ensure secure access to user accounts and prevent unauthorized access. • The system should protect user data, including personal information and uploaded songs, through encryption and secure storage practices. • Measures should be in place to prevent common security threats such as cross-site scripting (XSS) and SQL injection attacks.
Scalability:	• The platform should be designed to handle increased user traffic and growing song libraries. • The system architecture should support horizontal scaling, allowing for the addition of more servers or cloud resources to accommodate increased demand.
Compatibility:	• The platform should be compatible with a wide range of web browsers, ensuring consistent functionality and user experience across different browser versions.

	• The system should be responsive and optimized for various device types, including desktops, laptops, tablets, and mobile devices.
Usability:	• The user interface should be intuitive, easy to navigate, and visually appealing. • Clear and concise instructions should be provided to guide users through various actions and features. • The system should have informative error messages and user-friendly validation to assist users in providing correct inputs.
Data Backup and Recovery:	• Regular backups of user data, including user profiles and playlists, should be performed to prevent data loss. • The system should have mechanisms in place to recover data in the event of system failures or data corruption.

3. Relationship

The relationship between functional and non-functional requirements is that functional requirements define the specific features and actions the system should provide, while non-functional requirements define the qualities, characteristics, and constraints that govern how the system should perform and behave. Both types of requirements are necessary to ensure the overall success, usability, and quality of the Tune Source project. They work in tandem to guide the development, implementation, and evaluation of the system, ensuring that it meets the needs and expectations of its users while delivering a reliable, efficient, and user-friendly experience.

d. Discuss the technique(s) you would use to obtain the requirements.

1. Joint Application Development (JAD):

Pic 4: Joint Application Development

JAD is a collaborative approach that involves bringing together key stakeholders, end-users, and the development team in a facilitated workshop setting. The objective of JAD is to gather requirements, resolve conflicts, and make decisions collectively. During a JAD session, participants engage in brainstorming, discussions, and interactive exercises to elicit requirements and reach a consensus.

2. **Interviews:**

Pic 5: Interviews

Interviews involve one-on-one or small group discussions with stakeholders, subject matter experts, and end-users to gather information about their needs, expectations, and requirements. Interviews provide an opportunity to delve deeper into specific topics, ask follow-up questions, and explore individual perspectives. They are particularly useful for obtaining detailed and personalized information. Interviews can be structured, semi-structured, or unstructured, depending on the level of formality and the desired outcomes.

3. **Observation:**

Pic 6: Observation

Observation involves directly observing users or stakeholders in their natural environment or while performing tasks relevant to the project. By watching and documenting their actions, behaviors, and challenges, you can gain insights into their needs, preferences, and pain points. Observation can be passive, where you simply observe without interfering, or active, where you actively engage with the users and ask questions during the process. This technique is often used in user-centered design and usability studies to understand how people interact with systems or processes.

4. **Prototyping and User Feedback:**

Pic 7: Prototyping and User Feedback

Prototyping involves creating a simplified representation of the final product or system. By developing prototypes, you can provide stakeholders with something tangible to interact with and gather feedback. User feedback obtained through prototype testing helps uncover requirements, validate design decisions, and identify areas for improvement. Prototyping allows for iterative refinement of requirements based on user input, promoting user-centered design and enhancing the final product's usability.

5. **Surveys and Questionnaires:**

Pic 8: Surveys and Questionnaires

Surveys and questionnaires are structured data collection methods that involve asking stakeholders a set of predefined questions. They can be administered electronically or in person and are useful for gathering quantitative and qualitative data from a large number of respondents. Surveys and questionnaires are scalable and allow for statistical analysis of the collected data. They can be used to gain a broad understanding of stakeholders' opinions, preferences, and requirements.

6. **Advantages and Disavantages the technique(s) you would use to obtain the requirements.**

The techniques	Advantages	Disavantages
Joint Application Development (JAD)	• **Collaboration**: JAD promotes active participation and collaboration among stakeholders, leading to a better understanding of requirements and increased stakeholder buy-in. • **Efficiency**: JAD sessions allow for real-time discussions, leading to faster decision-making and requirement elicitation compared to individual interviews. • **Conflict resolution**: JAD sessions provide a platform for resolving conflicts and reaching consensus among stakeholders, reducing misunderstandings and potential issues during the project.	• **Time and resource-intensive**: JAD sessions require scheduling and coordinating the availability of multiple stakeholders, which can be challenging and time-consuming. • **Dominant voices**: Certain stakeholders may dominate the discussions, potentially overshadowing the perspectives of others and leading to biased requirements. • **Group dynamics**: Conflicting opinions and power struggles among stakeholders can hinder progress and compromise the effectiveness of JAD sessions.
Interviews	• **Detailed insights**: Interviews provide an opportunity to delve deep into stakeholders' individual perspectives, allowing for in-depth understanding of their needs, expectations, and requirements. • **Flexibility**: Interviews can be tailored to each stakeholder, allowing for personalized discussions and the exploration of specific topics or concerns. • **Clarification and follow-up**: Interviews enable immediate clarification of ambiguous or unclear requirements and allow for follow-up questions to gather additional information.	• **Time-consuming**: Conducting individual interviews with multiple stakeholders can be time-consuming, particularly in projects with a large number of stakeholders. • **Bias and subjectivity**: Interview results can be influenced by the interviewer's interpretation and biases, potentially leading to skewed or incomplete requirements. • **Limited scalability**: Interviews are most effective for a smaller group of stakeholders, making them less practical for projects with a large number of participants.

Observation	• **Real-world insights**: Observation allows for firsthand understanding of how users or stakeholders interact with systems, processes, or environments, providing authentic insights into their needs and behaviors. • **Unbiased information**: Observing stakeholders directly reduces the potential for biased or distorted information that may occur through self-reporting. • **Identification of implicit requirements**: Observations can uncover requirements that stakeholders may not explicitly communicate, leading to a more comprehensive understanding of their needs.	• **Limited context**: Observing stakeholders in controlled or simulated environments may not fully capture the complexities and nuances of their real-world experiences. • **Interpretation challenges**: Interpreting and documenting observational data accurately can be subjective and dependent on the observer's perspective and biases. • **Ethical considerations**: Observational studies must be conducted ethically, respecting privacy and obtaining appropriate consent from participants.
Prototyping and User Feedback	• **Tangible representation**: Prototypes provide stakeholders with a tangible representation of the final product, enabling clearer communication and more accurate feedback. • **Iterative refinement**: User feedback obtained through prototyping allows for iterative improvement of the requirements, resulting in a solution that better meets stakeholders' needs. • **User-centered design**: Prototyping and user feedback foster a user-centered approach, ensuring that the final product aligns with users' preferences and expectations.	• **Resource-intensive**: Developing prototypes and gathering user feedback can require significant time, effort, and resources. • **Limited representation**: Prototypes may not fully capture all aspects of the final product, potentially leading to incomplete or skewed feedback. • **User bias**: User feedback can be influenced by individual preferences and biases, which may not always align with the broader stakeholder group.
Surveys and Questionnaires	• **Scalability**: Surveys and questionnaires can be administered to a large number	• **Superficial insights**: Surveys may not capture the depth of stakeholders' perspectives and

	of stakeholders simultaneously, allowing for efficient data collection and analysis. • **Standardization**: Surveys enable consistent and standardized data collection, facilitating quantitative analysis and comparison across respondents. • **Anonymity**: Respondents can provide feedback anonymously, potentially encouraging more honest and unbiased responses.	may miss important context or nuanced requirements. • **Limited flexibility**: Surveys have predefined questions and response options, limiting the ability to explore topics in detail or gather unanticipated information. • **Low response rates**: Surveys often face challenges with low response rates, which can introduce response bias and limit the representativeness of the collected data.

e. Techniques for getting requirements in Tune Source

1. Business project automation (BPA)

Pic 9: Business project automation (BPA)

Business project automation in the Tune Source project involves utilizing various techniques to streamline and optimize different aspects of the business processes.

Content Management System (CMS): Implementing a customized CMS for efficient content creation, curation, and publishing.

Digital Rights Management (DRM): Deploying DRM technologies to manage and protect copyrighted content.

Metadata Management: Using automated tools to organize and maintain accurate song metadata.

Recommendation Engines: Implementing machine learning algorithms for personalized song recommendations.

Payment Processing Automation: Integrating automated payment systems for streamlined billing and payment collection.

Analytics and Reporting Automation: Utilizing automated tools for data analysis and reporting on user engagement, song popularity, and revenue generation.

Customer Relationship Management (CRM): Implementing a CRM system to automate customer interactions and support processes.

API Integration: Integrating with external systems through APIs for seamless data exchange and service integration.

2. **Business project improvement (BPI)**

Pic 10: Business project improvement (BPI)

Business project improvement in the Tune Source project involves implementing various techniques to enhance processes, optimize performance, and achieve better results:

Continuous Process Improvement: Using methodologies like Lean or Six Sigma to identify and eliminate inefficiencies and waste.

Agile Project Management: Implementing agile methodologies such as Scrum or Kanban to enhance project planning, execution, and delivery.

Stakeholder Engagement: Actively involving stakeholders in decision-making and feedback processes.

Key Performance Indicators (KPIs) and Metrics: Establishing relevant metrics to measure progress and track achievements.

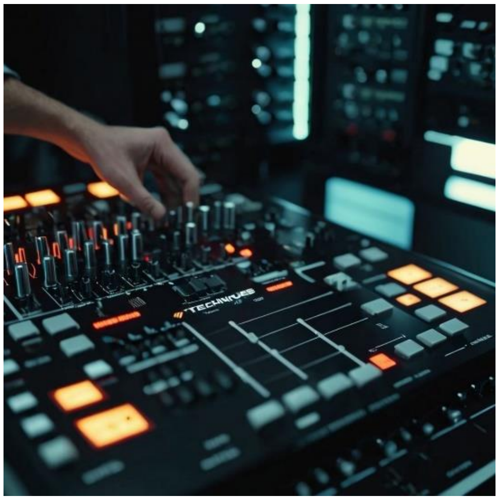
User Experience (UX) Design: Applying UX principles to enhance the usability and user experience of the Tune Source platform.

Quality Assurance and Testing: Implementing robust testing and quality assurance processes throughout the project.

Training and Skill Development: Providing training opportunities to enhance team members' expertise and capabilities.

Feedback and Lessons Learned: Gathering feedback and documenting lessons learned for continuous learning and improvement.

3. Business project reengineering (BPR)



Pic 11: Business project reengineering (BPR)

Business project reengineering in the Tune Source project involves implementing various techniques to fundamentally redesign and improve existing business processes.

Process Mapping and Analysis: Documenting and analyzing existing processes to identify inefficiencies and areas for improvement.

Value Stream Mapping: Visualizing the flow of value through the project to identify waste and opportunities for streamlining.

Business Process Redesign: Rethinking and redesigning processes to eliminate unnecessary steps and optimize resource utilization.

Technology Integration: Leveraging technology solutions to automate and streamline processes.

Organizational Restructuring: Making necessary changes to the organizational structure to improve collaboration and decision-making.

Change Management: Managing the transition and acceptance of reengineered processes.

Performance Measurement and Monitoring: Establishing metrics and mechanisms to track the effectiveness of reengineered processes.

Customer-Centric Approach: Focusing on understanding and meeting customer needs to enhance satisfaction and loyalty.

P6 Use appropriate software analysis tools/techniques to carry out a software investigation and create supporting documentation.

a. Use case diagram

1. Definition

A case diagram, also known as a use case diagram, is a visual representation of the interactions and relationships between actors and use cases in a system. It illustrates how the system is used by different actors to accomplish specific tasks or goals. Actors are represented as stick figures, and use cases are depicted as ovals or rectangles. The diagram helps in understanding the system's functionality, identifying user interactions, and capturing the system's requirements. It serves as a communication tool among stakeholders and aids in defining the system's scope and boundaries.

2. Define actors and use case for the project

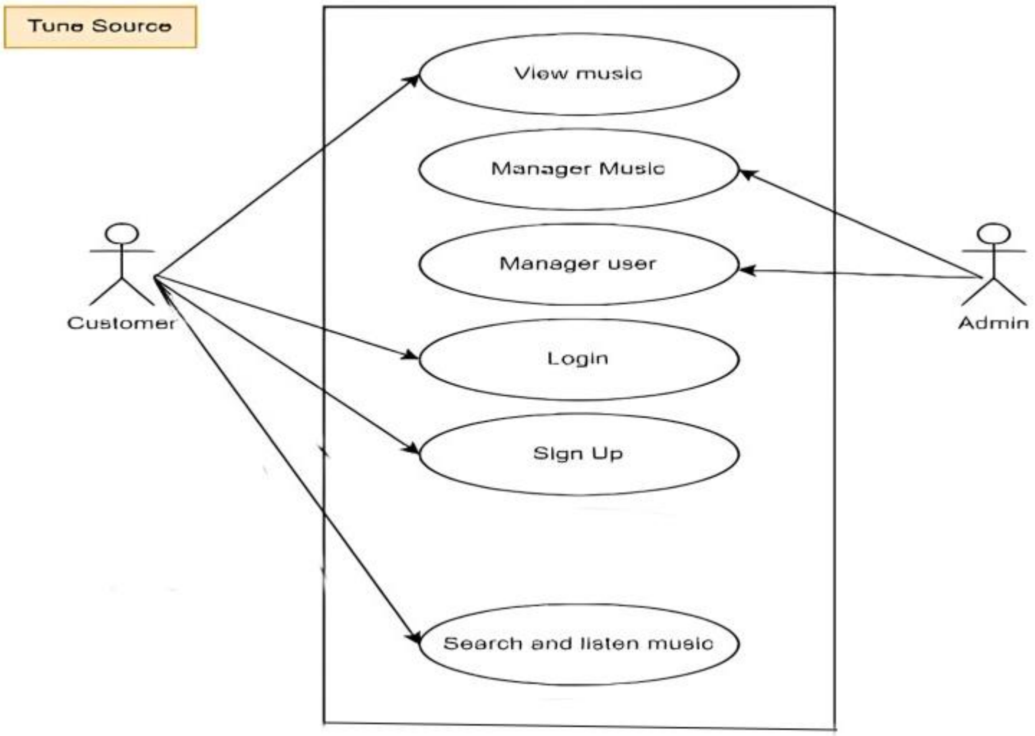
Actors:

Users	Individuals who interact with the Tune Source platform, including music enthusiasts, listeners, artists, and music industry professionals.
Artists	Musicians and music creators who upload and showcase their music on Tune Source.
Administrators	Individuals responsible for managing and maintaining the Tune Source platform.

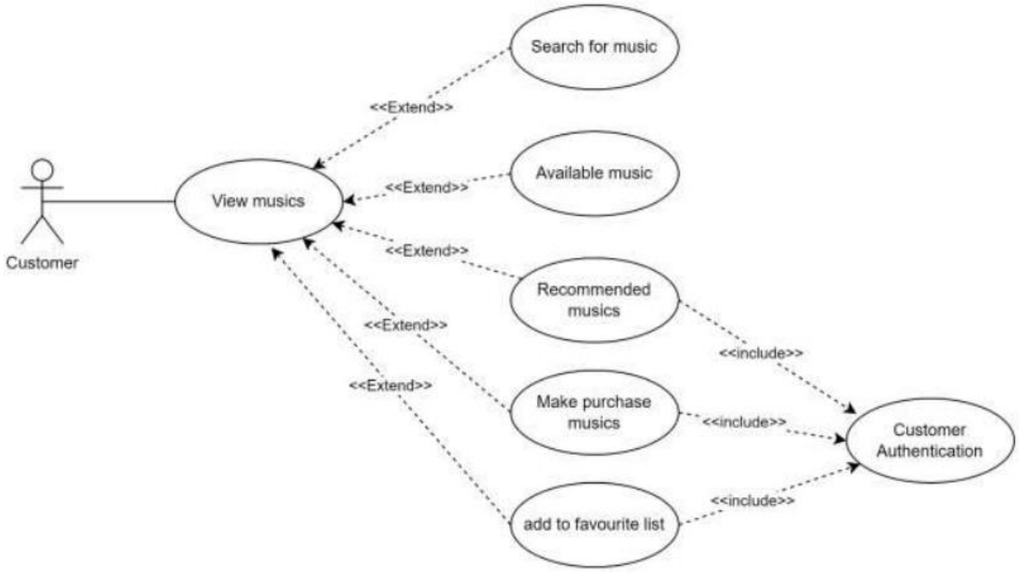
Use Cases:

User Registration and Authentication	The process of users creating an account and authenticating themselves on Tune Source.
Song Discovery and Playback	Users searching, browsing, and playing songs on Tune Source.
Playlist Creation and Management	Users creating and managing personalized playlists on the platform.
Artist Profile	Artists updating and managing their profiles on Tune Source.
Song Upload and	Artists uploading their original songs and managing them on the platform.
Licensing and Royalty	Managing licensing agreements and royalty payments for artists.
Reporting and Analytics	Administrators accessing reporting and analytics features to monitor platform performance.

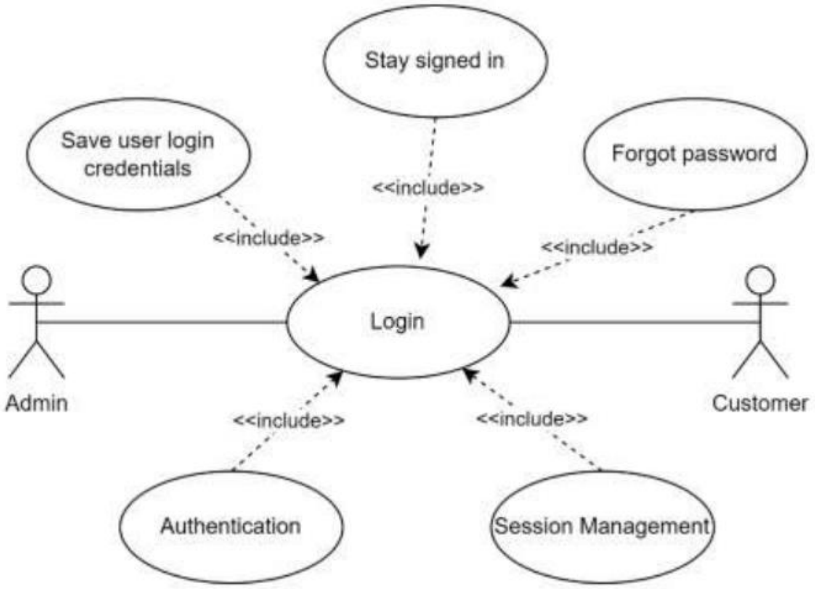
3. Use case diagram for the project



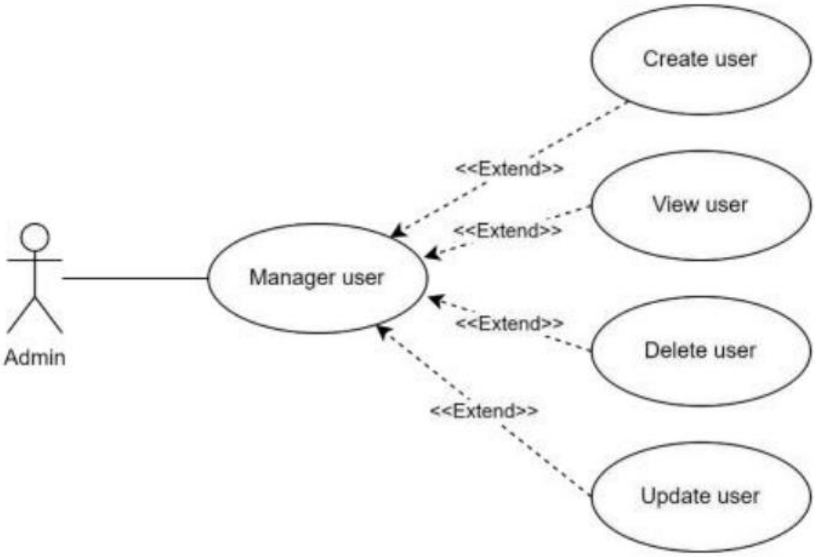
Pic 12: Use Case diagram for the Tune Source project



Pic 13: Customer's Use case diagram



Pic 14: Login Use case diagram



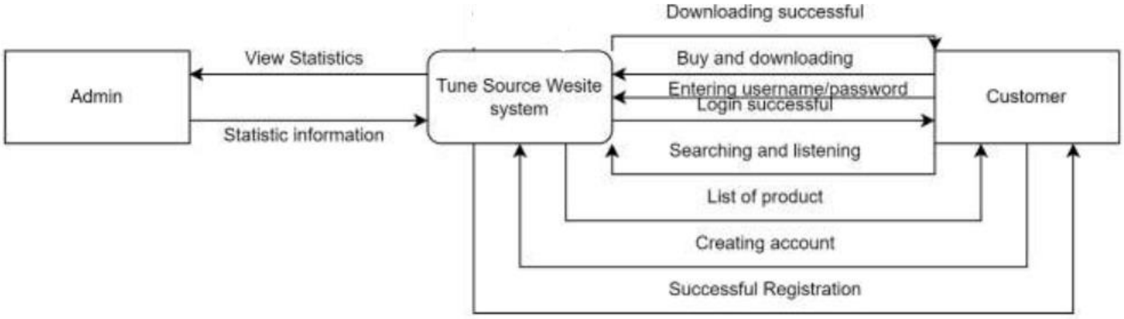
Pic 15: Admin's Use case diagram

b. Data flow diagram.

1. Definition

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within a system or process. It shows how data is input, processed, transformed, and output by different components of the system. DFDs consist of processes, data flows, external entities, and data stores. Processes represent activities or transformations, data flows depict the movement of data, external entities represent sources or destinations of data, and data stores represent storage locations. DFDs help in understanding the data flow, identifying dependencies, and visualizing the system's structure. They are useful in system analysis, requirements specification, and stakeholder communication during system development.

2. **Use Data flow diagram for the project**



Pic 16: Data flow diagram for the Tune Source project

A data flow diagram (DFD) outlines the information flow for this process and the system. It uses defined symbols such as rectangles, circles, and arrows, along with short text labels, to display input data, output data, save points, and routes between each destination. Customers can download their music and they can view that information in detail.

c. **Entity relation diagram.**

1. **Definition**

An Entity-Relationship Diagram (ERD) is a visual representation used in database design to illustrate the relationships between entities. Entities represent real-world objects or concepts, attributes represent their characteristics, and relationships represent associations between entities. Cardinality describes the number of instances associated with entities, and primary and foreign keys establish unique identifiers and relationships between entities. ERDs aid in understanding the structure of a database, facilitating database design and communication among stakeholders involved in the database system.

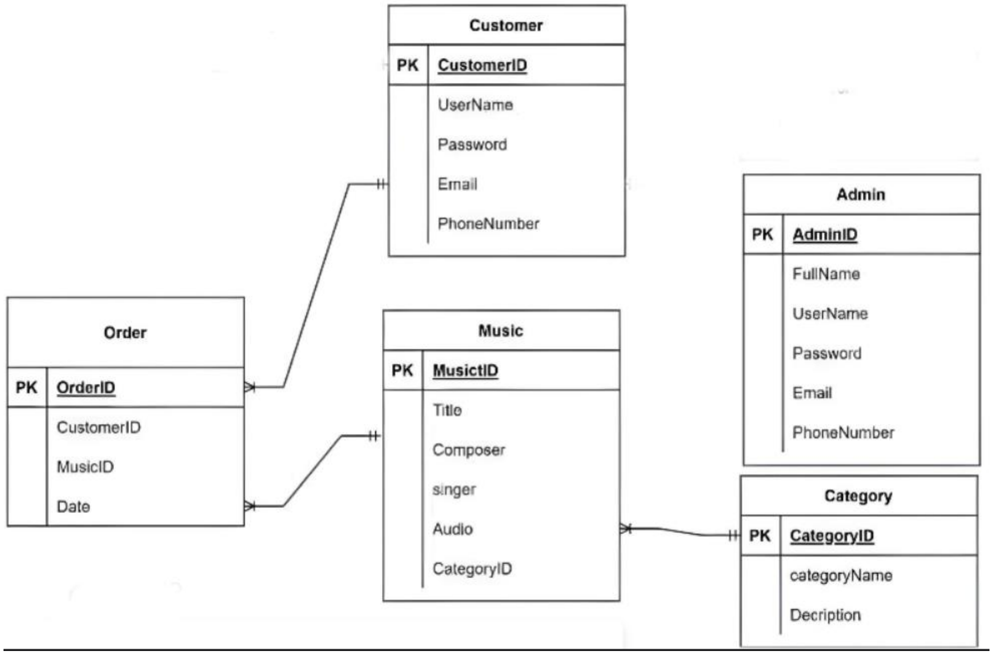
2. **Entity description table for project.**

An entity description table for the Tune Source project could include the following entities and their corresponding descriptions:

Entity	Description
User	Represents users who interact with the Tune Source platform. They can include music enthusiasts, listeners, artists, and music industry professionals.

Artist	Represents musicians, bands, and music creators who upload and showcase their music on the Tune Source platform.
Song	Represents individual songs available on the Tune Source platform
Playlist	Represents personalized playlists created by users on Tune Source.
Admin	Represents administrators responsible for managing and maintaining the Tune Source platform.
Licensing	Represents licensing agreements for songs on Tune Source.
Genre	Represents music genres available on Tune Source.

3. Entity relation diagram for project.



Pic 17: Entity relation diagram for the Tune Source project.

Entity	Description
Admin	The manager of the whole system
Customer	Information of use
Category	Category of song
Song	Song information
Order	Search and selection of songs

Entity Order

No	Name	Datatype	Relationship
1	Order_ID	Varchar (20)	PK
2	Customer_ID	Varchar (20)	FK
3	Music_ID	Varchar (20)	FK
4	Date	Date	

Entity Customer

No	Name	Datatype	Relationship
1	Customer_ID	Varchar (20)	PK
2	Username	Varchar (20)	FK
3	Password	Varchar (20)	FK
4	Email	Varchar (20)	
5	Phone_number	Int(10)	

Entity Admin

No	Name	Datatype	Relationship
1	Customer_ID	Varchar (20)	PK
2	Username	Varchar (20)	FK
3	Password	Varchar (20)	FK
4	Email	Varchar (20)	
5	Phone_number	Int (10)	
6	Full_name	Varchar (20)	

Entity Category

No	Name	Datatype	Relationship
1	Category ID	Varchar (20)	PK
2	Category_name	Varchar (20)	
3	Description	Varchar (20)	

Entity Song

No	Name	Datatype	Relationship
1	Music_ID	Varchar (20)	PK
2	Composer	Varchar (20)	
3	singer	Varchar (20)	
4	Audio	Varchar (20)	
5	Category_ID	Varchar (20)	FK

P7 Discuss, using examples, the suitability of software behavioral design techniques.

User and software requirements are addressed during the design phase of a software project to guarantee that the final product fits the needs and expectations of the users. This is accomplished through a variety of strategies, including the creation of mock-ups and wireframes, the selection of an adequate architecture, and the selection of a suitable technological solution stack. Let us go over each of these points in more detail:

1. Defintion

Mock-ups and wireframes are visual representations used in the design and development process of software or websites. Wireframes are simplified and focus on the structure and layout without detailed design elements. They provide a skeletal framework to outline the user interface. Mock-ups, on the other hand, are more detailed and resemble the final product with colors, typography, and imagery. They simulate the visual appearance and help stakeholders visualize the aesthetics and branding. Wireframes are used for initial planning and feedback on structure, while mock-ups refine the design and gather feedback on visual aspects. Both serve important purposes in the design process.

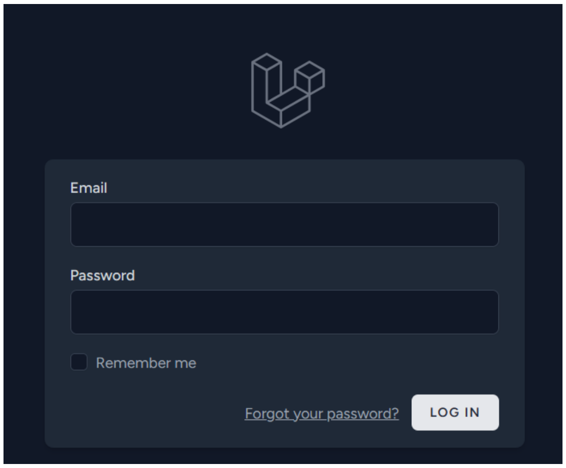
2. Mock-up, and Wireframe are used in the project

a. Design Phase: Addressing User and Software Requirements:

In the design phase, the goal is to translate user and software requirements into a tangible and visually appealing system. This involves creating mock-ups and wireframes to ensure that the user interface aligns with expectations and facilitates efficient collaboration between stakeholders.

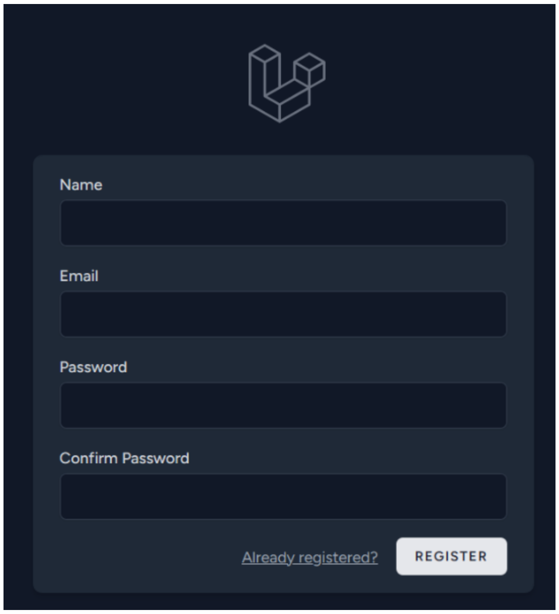
b. Mock-up - Customer login and Homepage

The visual presentation of the Tune Source homepage displays branding, search functionality, featured songs and navigational elements. Besides, enables customers to provide feedback on visual layout, color scheme and location of login elements.



Pic 18: Login screen

Explain*:* Allows customers to provide feedback on the visual layout, color scheme, and placement of login elements.



Name

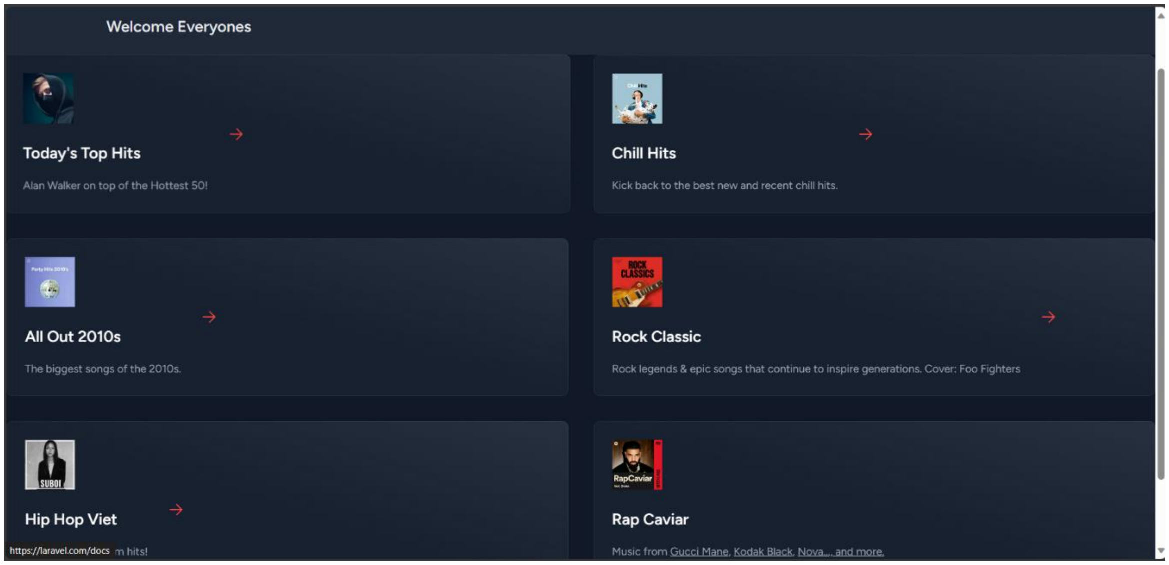
Email

Password

Confirm Password

Pic 19: Regiter screen

Explain: *Visualizes the design of the customer registration interface, allowing stakeholders*



Welcome Everyones

Today's Top Hits

Alan Walker on top of the Hottest 50!

Chill Hits

Kick back to the best new and recent chill hits.

All Out 2010s

The biggest songs of the 2010s.

Rock Classic

Rock legends & epic songs that continue to inspire generations. Cover: Foo Fighters

Hip Hop Viet

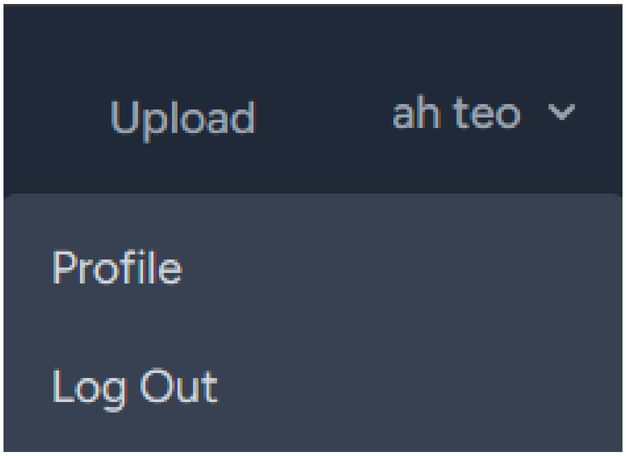
https://laravel.com/docs m hits!

Rap Caviar

Music from Gucci Mane, Kodak Black, Nova..., and more.

Pic 20: HomePage screen

Explain: Greet new users after signing up for an account and logging into the app

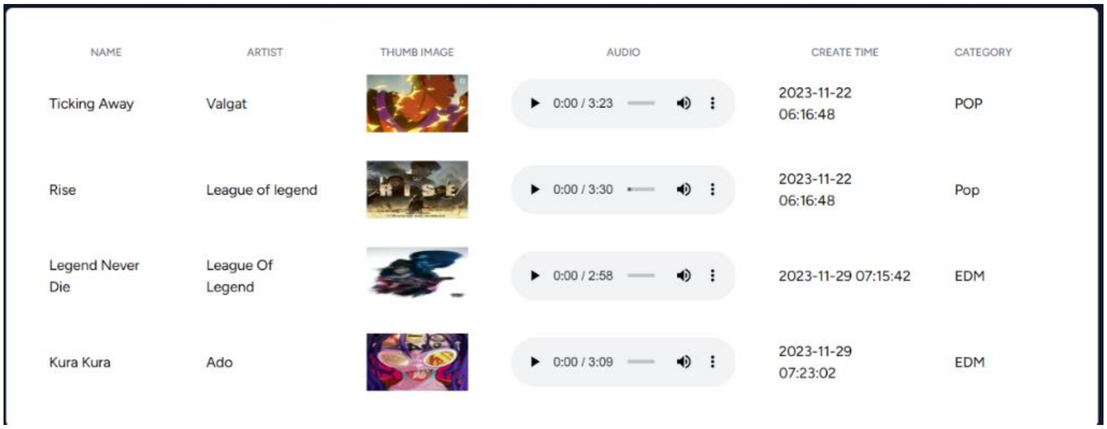


Pic 21: Logout screen

Explain: Provide feedback on the layout, information fields, and user-friendly registration process. to enable users to end their login session inside the application if they do not use it

c. Wireframe - Song Details Interface:

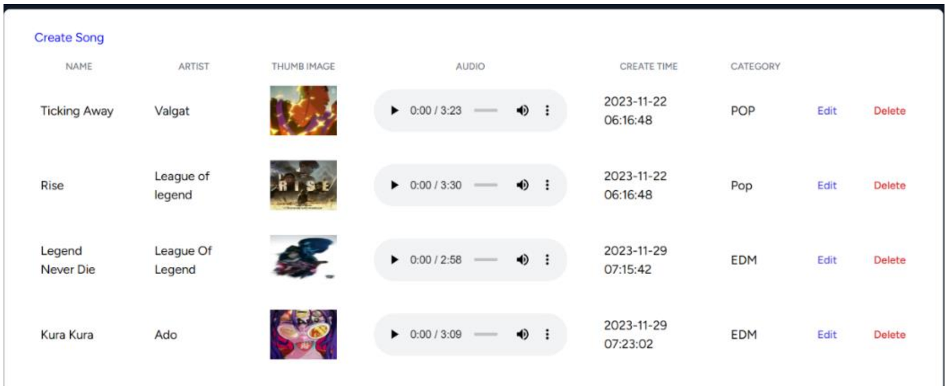
Provides a skeletal representation of the playlists management interface, focusing on the layout and essential interactive elements. Creation, editing, and organization. components of the song details page, including song title, artist name, album cover, play controls, lyrics, and related songs



NAME	ARTIST	THUMB IMAGE	AUDIO	CREATE TIME	CATEGORY
Ticking Away	Valgat		▶ 0:00 / 3:23	2023-11-22 06:16:48	POP
Rise	League of legend		▶ 0:00 / 3:30	2023-11-22 06:16:48	Pop
Legend Never Die	League Of Legend		▶ 0:00 / 2:58	2023-11-29 07:15:42	EDM
Kura Kura	Ado		▶ 0:00 / 3:09	2023-11-29 07:23:02	EDM

Pic 22: The operation process of the system for the role: User

Explain: The system has eliminated tasks that affect the system so that users can use the service without affecting the system

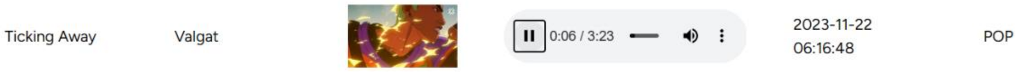


Create Song

NAME	ARTIST	THUMB IMAGE	AUDIO	CREATE TIME	CATEGORY		
Ticking Away	Valgat		0:00 / 3:23	2023-11-22 06:16:48	POP	Edit	Delete
Rise	League of legend		0:00 / 3:30	2023-11-22 06:16:48	Pop	Edit	Delete
Legend Never Die	League Of Legend		0:00 / 2:58	2023-11-29 07:15:42	EDM	Edit	Delete
Kura Kura	Ado		0:00 / 3:09	2023-11-29 07:23:02	EDM	Edit	Delete

Pic 23:The operation process of the system for the role: admin

Explain: The system has added tasks that affect the system so that administrators can control and repair the system in case of necessity such as encountering bugs or hackers stealing information, etc.



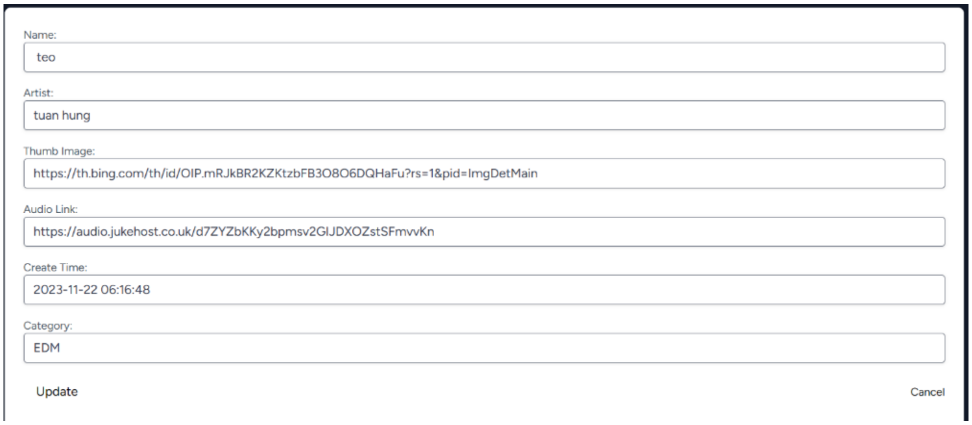
Ticking Away	Valgat		0:06 / 3:23	2023-11-22 06:16:48	POP

Pic 24: Play music

Explain: You can enjoy music quickly, easily and conveniently when using the system with a playlist we offer you to share by adding songs for everyone using the application to enjoy it.

d. Mock-up - Music system management Interface:

Simulation model demonstrates the process of creating a new playlist, includes a user interface for adding songs, setting titles, and saving playlists. illustrates the music upload interface design, enabling stakeholders provides feedback on the process, file upload options and metadata import steps, and visualizes the process for artists and users to contribute to the music archive



Name:
teo

Artist:
tuan hung

Thumb Image:
https://th.bing.com/th/id/OIP.mRJkBR2KZKtzbFB3O8O6DQHaFu?rs=1&pid=ImgDetMain

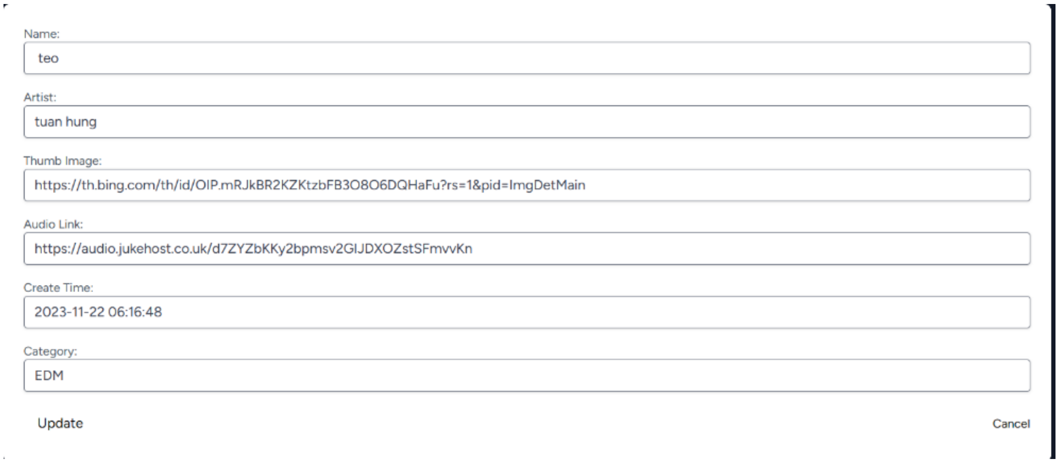
Audio Link:
https://audio.jukehost.co.uk/d7ZYZbKKy2bpmsv2GlJDXOZstSFmvvKn

Create Time:
2023-11-22 06:16:48

Category:
EDM

Pic 25: Upload song

Explain: Illustrates the design of the music upload interface, enabling stakeholders to provide feedback on the flow, file upload options, and metadata entry steps. This mock-up visualizes the process for artists and users to contribute to the music archive.



Name:
teo

Artist:
tuan hung

Thumb Image:
https://th.bing.com/th/id/OIP.mRJkBR2KZKtzbFB3O8O6DQHaFu?rs=1&pid=ImgDetMain

Audio Link:
https://audio.jukehost.co.uk/d7ZYZbKKy2bpmsv2GIJDXOZstSFmvvKn

Create Time:
2023-11-22 06:16:48

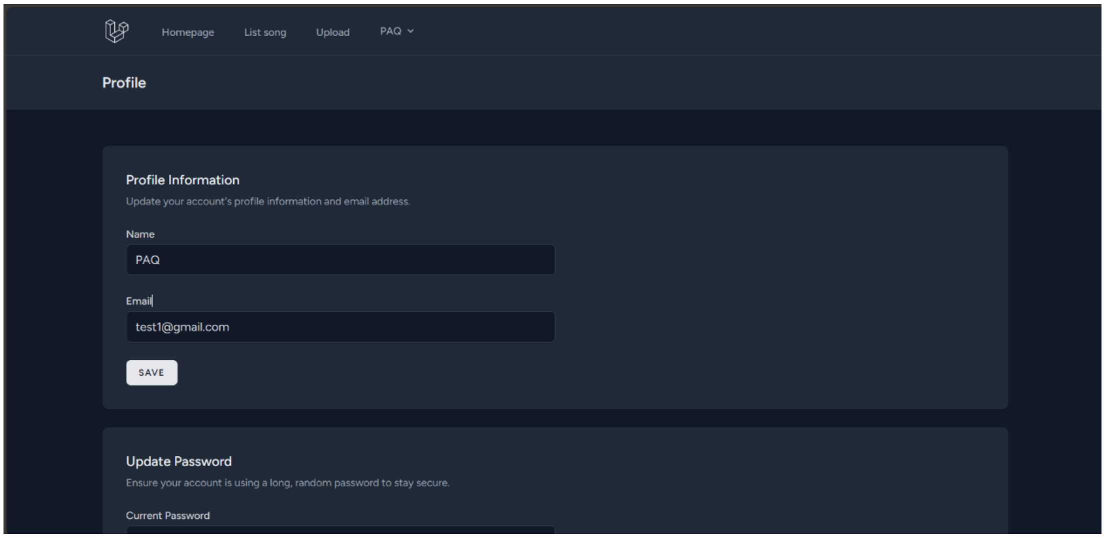
Category:
EDM

Pic 26: Edit song

Explain: to edit or change the song's information to suit the app space

e. Mock-up - Account Settings

A visual representation of the account settings page, showcasing options for changing personal information, subscription preferences, and password management.



Profile

Profile Information

Update your account's profile information and email address.

Name
PAQ

Email
test1@gmail.com

Update Password

Ensure your account is using a long, random password to stay secure.

Current Password

Pic 27: Account Settings

Explain: The Profile page shows rights such as the option to change personal information, registration options and password management.

3. Appropriate Architecture

The architecture selected is determined by criteria like as scalability, performance, maintainability, and the specific needs of the Tune Source project. The client-server architecture may be appropriate for such a project. The system is divided into two primary components in the client-server architecture: the client, which is responsible for user interface and presentation, and the server, which is responsible for data storage, processing, and business logic. This architecture allows for centralized data control and makes maintenance and upgrades easy. In the case of Tune Source, the client-side application can be designed as a web or mobile application, allowing users to search for music, listen to it, manage playlists, and make purchases through an intuitive interface. The server-side would handle duties such as song data storage and retrieval, user account management, payment processing, and other business logic

4. Suitable technical solution stack

Frontend: React.js

- **Modular Design**: React.js provides a modular and efficient way to build user interfaces, enhancing the overall user experience.
- **Responsive UI**: Enables the creation of a dynamic and responsive user interface, crucial for a seamless music discovery and download platform.

Backend: XAMPP with PHP

- **Fast and Scalable**: XAMPP, with its integration of Apache HTTP Server and PHP, allows for fast and scalable server-side scripting, aligning perfectly with the requirements of a dynamic music platform.
- **Efficient Request Handling**: PHP, in combination with XAMPP, simplifies the handling of HTTP requests, ensuring a smooth and efficient backend operation.

Database: XAMPP MySQL

- **Flexibility**: MySQL, included in the XAMPP stack, offers flexibility in managing music data and customer information, adapting well to the dynamic nature of Tune Source's content.
- **Efficient Storage and Retrieval**: Provides efficient storage and retrieval of data, crucial for managing a vast archive of music tracks and customer records.

Payment Integration: Stripe API

- **Security**: Stripe API ensures secure and seamless payment transactions, meeting the software's financial requirements.
- **Industry Standard**: Widely recognized and used in the industry, offering reliability and a comprehensive set of features for payment processing**.**

Hosting: AWS (Amazon Web Services)

- **Reliability**: AWS provides reliable and scalable hosting services, crucial for accommodating the potential growth of Tune Source.
- **Scalability: Ensures the platform can scale with increased demand, offering a robust hosting** solution for a dynamic music download platform.

CONCLUSION

In conclusion, the Software Development Life Cycle (SDLC) provides a structured framework for the development of the Tune Source project. By following the SDLC phases, including requirements gathering, system design, implementation, testing, deployment, and maintenance, the project can progress in a systematic and efficient manner.

The SDLC ensures that the Tune Source application meets the desired requirements and functionalities by thoroughly analyzing and prioritizing project goals. It enables the development team to design a robust system architecture, implement the necessary features, and conduct comprehensive testing to ensure the application's quality and performance.

The deployment phase ensures a smooth transition of the Tune Source application to the production environment, while the maintenance phase enables ongoing support, bug fixing, and updates to keep the application secure and up-to-date.

By adhering to the SDLC, the Tune Source project can benefit from improved project management, effective collaboration, and enhanced communication among team members. This structured approach minimizes risks, improves efficiency, and increases the likelihood of delivering a successful and user-friendly application.

REFERENCES

- All you need to know about business analysis in software development, explained (no date) Softwarehut.com. Available at: https://softwarehut.com/blog/business/business-analysis-in-software-development (Accessed: November 26, 2023)
- Gillis, O. (2020) “How to create a website wireframe (3 simple steps),” Elementa, 8 June. Available at: https://elementor.com/blog/wireframe-website/ (Accessed: November 26, 2023).
- Investigation Management Software (2023) ComplianceQuest QHSE Solutions. Available at: https://www.compliancequest.com/lab-investigations/investigation-management-software/ (Accessed: November 26, 2023).
- Is documenting a big project with UML Diagrams needed, good to have or even not possible? (no date) Software Engineering Stack Exchange. Available at: https://softwareengineering.stackexchange.com/questions/441543/is-documenting-a-big-project-with-uml-diagrams-needed-good-to-have-or-even-not (Accessed: November 26, 2023).
- Music library management system use case diagram (no date) Freeprojectz.com. Available at: https://www.freeprojectz.com/use-case/music-library-management-system-use-case-diagram (Accessed: November 26, 2023).
- Oragui, D. (2020) Software documentation best practices [with examples], Helpjuice.com. Available at: https://helpjuice.com/blog/software-documentation (Accessed: November 26, 2023).

- Replogle, N. (2018) The 7 best wireframe tools in 2023, Zapier.com. Zapier. Available at: https://zapier.com/blog/best-wireframe-tools/ (Accessed: November 26, 2023).
- Software Analysis & Design Tools (no date) Tutorialspoint.com. Available at: https://www.tutorialspoint.com/software_engineering/software_analysis_design_tools.htm (Accessed: November 26, 2023).
- “Technical documentation in software development: Types, best practices, and tools” (2023) Altex Soft, 29 March. Available at: https://www.altexsoft.com/blog/technical-documentation-in-software-development-types-best-practices-and-tools/ (Accessed: November 26, 2023).
- Verner, J. M. and Evan co, W. M. (2003) “An investigation into software development process knowledge,” in Managing Software Engineering Knowledge. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 29–47.