# Music Player

## A PROJECT REPORT

**for**

**Mini Project (KCA353)**

**Session (2023-24)**

Submitted by

Utsav Chakraborty

University Roll No 22002901400170

Submitted in partial fulfilment of the Requirements for the Degree of

# MASTER OF COMPUTER APPLICATION

**Under the Supervision of**

**Dr. Akash Rajak**

**Professor**



**Submitted to**

### Department Of Computer Applications

**KIET Group of Institutions, Ghaziabad Uttar Pradesh-201206**

**(FEBRUARY 2024)**

# CERTIFICATE

Certified that **Utsav Chakraborty (2200290140170)** has carried out the project work having “**Music Player**” (**Mini Project KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU**)** (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

**Date: Utsav Chakraborty**

**(2200290140170)**

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Akash Rajak Dr. Arun Kumar Tripathi

Professor Professor & Head

Department of Computer Applications Department of Computer Applications KIET Group of Institutions, Ghaziabad KIET Group of Institutions, Ghaziabad

# ABSTRACT

The Music Player clone is a music streaming application that replicates the core features of the original platform. It allows users to discover, play and create playlists of their favorite songs.

The clone employs a user-friendly interface, seamless audio playback and a recommendation system based on user preferences. While mirroring the essence of Music Player, the clone may incorporate additional features or improvements tailored to specific needs.

Key Features:

Vast Music Library: The Music Player Clone offers an extensive collection of songs, spanning various genres, artists, and languages. Users can explore and discover a diverse range of music to suit their preferences.

Personalized Playlists: Users can create and manage personalized playlists, curating their favourite tracks for different moods, occasions, or themes. The application intelligently suggests songs based on user preferences, creating a dynamic and tailored listening experience.

User-Friendly Interface: The user interface is designed to be intuitive and visually appealing, ensuring a smooth and enjoyable navigation experience. Users can effortlessly search for songs, artists, or albums and access their music library with ease.

Offline Mode: The Music Player Clone allows users to download their favourite songs for offline playback. This feature ensures that users can enjoy their music even when they are not connected to the internet, enhancing the overall accessibility of the application.

Social Integration: Users can share their favourite songs, playlists, and music discoveries with friends through seamless social media integration. This feature promotes a collaborative and social aspect to the music listening experience.

High-Quality Audio Streaming: The application supports high-quality audio streaming to provide users with an immersive and crystal-clear listening experience. The music player clone prioritizes audio quality to ensure that users can enjoy their favourite tracks in the best possible sound.

Cross-Platform Compatibility: The Music Player Clone is designed to be compatible with various devices and operating systems, allowing users to access their music seamlessly across smartphones, tablets, and desktops.

# ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr Akash Rajak** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to **Dr. Arun Kumar Tripathi**, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Utsav Chakraborty

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| Certificate | | | i |
|  | Abstract |  | ii |
|  | Acknowledgement | | iii |
|  | Table of Contents | | iv-v |
|  | List of Tables | | vi |
|  | List of Figures | | vii |
| 1 | Introduction | | 8-10 |
|  | 1.1 | Background | 8 |
|  | 1.2 | Project overview | 8 |
|  | 1.3 | Objective | 9 |
|  | 1.4 | Key features | 10 |
|  | 1.5 | Scope of the project | 10 |
| 2 | Problem identification & feasibility study | | 11-12 |
|  | 2.1 | Problem Identification | 11 |
|  | 2.2 | Feasibility Study | 11 |
|  |  | 2.2.1 Technical Feasibility | 11 |
|  |  | 2.2.2 Operational Feasibility | 12 |
|  |  | 2.2.3 Economic Feasibility | 12 |
| 3 | Requirement Analysis | | 13-14 |
|  | 3.1 Introduction | | 13 |
|  | 3.2 Functional Requirements | | 13 |
|  | 3.3 Non-Functional Requirements | | 14 |
|  | 3.3.1 Performance | | 14 |
|  | 3.3.2 Security | | 14 |
|  | 3.3.3 Scalability | | 14 |
|  | 3.3.4 Usability | | 14 |
| 4 | Project Planning and Scheduling | | 15 |
|  | 4.1 Pert Chart | | 15 |
| 5 | Hardware and Software Specification | | 16-17 |
|  | 5.1 Hardware Specification | | 16 |
|  | 5.2 Software Specification | | 17 |
| 6 | Choice of Tools & Technology | | 18-22 |

|  |  |  |
| --- | --- | --- |
|  | 6.1 HTML, CSS | 18 |
|  | 6.2 JavaScript | 18 |
|  | 6.3 Data Flow Diagram | 18-19 |
|  | 6.4 Context Level Diagram | 19-22 |
| 7 | ER-Diagram | 23-24 |
|  | 7.1 Entity-relationship model | 23 |
|  | 7.2 Class Diagram | 24 |
| 8 | Database | 25 |
| 9 | Form Design | 26-28 |
| 10 | Testing | 29 |
|  | 10.1 Introduction | 29 |
|  | 10.2 Types of Testing | 29 |
|  | Future Scope and Further Enhancement of the Project 30 | |
|  | Conclusion & References | 31 |

**LIST OF TABLES**

|  |  |
| --- | --- |
| **Name of Table** | **Page** |
| Playlist | 16 |
| Songs | 17 |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| **Name of Figure** | **Page No.** |
| Pert Chart | 15 |
| 0 Level DFD | 20 |
| 1 Level DFD | 21 |
| 2nd Level DFD | 22 |
| Entity-relationship Model | 23 |

**CHAPTER 1**

**INTRODUCTION**

* 1. **Background**

The music Player Clone project aims to replicate the core functionalities and users experience of the popular music streaming platform Spotify. This application will allow users to search, play and organize their music library in a user friendly and visually appealing interface, mirroring the key features of Spotify.

* 1. **Project Overview**

The project aims to replicatebasic functionality of popular music player, allowing users to play, pause, skip tracks, and adjust volume. Utilize HTML, for structuring elements such as buttons, progress bars, and song lists. CSS will style the player, providing visual appeal and responsiveness. Ensure compatibility across devices and browser. Focus on clean code structure and intuitive user interface design. Incorporate and design consistency. This project serves as a practical exercise in front-end web development, honing skills in HTML and CSS.

* 1. **Objective**

The primary objective pf a Music Player include:

* Efficiency: Optimizing the search.
* User Convenience: Searching of the songs without any interruption.
  1. **Key Features**

1. **User Authentication:**
2. Allow users to create login securely
3. Implement password encryption
4. **Music Catalogue:**
5. Integrate vast music catalogue with a diverse range of artists, albums, and genres
6. Utilize an external API to fetch and deploy music metadata
7. **User Profile Settings:**
8. Customization of user profiles, including display names, profile picture and notification preference
9. **Responsive Design:**
10. Ensure the application is visually appealing and functionally on various devices and screen sizes
11. **Offline mode:**
12. Implementation of an offline mode for users to listen to download music without an internet connection

**CHAPTER 2**

**PROBLEM IDENTIFICATION & FEASIBILITY STUDY**

### Problem Identification

Identifying potential problems in a traveling website is crucial for ensuring a smooth user experience. Common issues include poor user interface and experience, limited search and filtering options, incomplete or inaccurate information, booking and payment issues, ineffective customer support, poor mobile responsiveness, limited personalization, and security concerns. Addressing these issues through regular user testing, feedback gathering, and continuous improvement efforts can enhance the overall user satisfaction and usability of the website.

### Feasibility Study

A feasibility study for a Spotify clone, named Music Player Clone, involves assessing the viability and practicality of the project. The study will cover various aspects, including technical, economic, operational, and legal considerations. Here's a breakdown of key components:

### Technical Feasibility

* **Technology Stack**: Assess the availability and appropriateness of technologies required development (e.g., frontend frameworks, backend technologies, databases).
* **API Integration:** Check the feasibility of integrating with a music catalog API (e.g, Spotify API) and other third-party services.
* **Scalability:** Evaluate the scalability of the chosen architecture to handle potential increases in user base and content.

### Operational Feasibility

* **Resource Availability:** Assess the availability of skilled developers, designers, and other resources needed for the project.
* **Timeline:** Define a realistic development timeline, considering potential challenges and resource constraints.
* **Operational Impact:** Analyse the impact of the Music Player Clone on existing operations and workflows.

### Economic Feasibility

* **Cost Estimation:** Estimate the costs associated with development, hosting.
* **Revenue Model:** Explore potential revenue streams, such as subscription plans,

advertisements, or partnerships.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

## Functional Requirements

**Playback Controls**:

* + Implement basic controls such as play, pause, stop, and skip forward/backward.
  + Allow users to start playing a song from a specified time (seek functionality).

**Volume Control**:

* + Provide a slider or buttons to adjust the volume level.
  + Mute/unmute option for quick volume control.

**Playlist Management**:

* + Allow users to create playlists.
  + Enable adding/removing songs from playlists.
  + Support rearranging the order of songs in a playlist.

**Time Display**:

* + Display the current playback time and total duration of the song.
  + Optionally, provide a progress bar to visualize the playback progress.

**Repeat and Shuffle**:

* + Include options to repeat the current song or the entire playlist.
  + Enable shuffling of the playlist for randomized playback.

**Song Information**:

* + Display metadata such as song title, artist, album, and album art.
  + Allow users to view detailed information about the currently playing song.

**Playback State Management**:

* + Remember the playback state (e.g., current song, position, volume) when the user navigates away from the player and returns.

**Keyboard Controls**:

* + Support keyboard shortcuts for controlling playback (e.g., spacebar for play/pause, arrow keys for skipping tracks).

**Mobile Compatibility**:

* + Ensure the player is mobile-friendly with touch-friendly controls and responsive design.

**Audio Format Support**:

* Support common audio file formats (e.g., MP3, AAC, FLAC) for playback.

## Non-Functional Requirements

**Performance:**

* Ensure fast loading times and smooth playback performance even with large playlists.
* Minimize latency in response to user interactions for a seamless experience.

**Scalability:**

* Design the player to handle a growing library of songs and playlists efficiently.
* Ensure scalability to accommodate increasing user traffic and usage patterns.

**Reliability**:

* Ensure high availability with minimal downtime for uninterrupted playback.
* Implement error handling mechanisms to gracefully recover from failures.

**Usability**:

* Design an intuitive user interface with clear navigation and easy-to-understand controls.
* Prioritize user experience to make the player accessible and enjoyable for all users.

**Security**:

* Implement measures to protect user data and prevent unauthorized access to the player.
* Ensure secure communication channels for transferring songs and user interactions.

**CHAPTER 4**

**PROJECT PLANNING AND SCHEDULING**

### Pert Chart:

A PERT chart is a project management tools used to schedule, organize, and coordinate tasks within a project. PERT stands for Program Evaluation Review Technique. A PERT chart presents a graphic illustration of a project as network diagram consisting of numbered nodes (either circles or rectangles) representing events, or milestones in the project linked by labelled vectors (directional lines) representing tasks in the project.

The direction of the arrows on the lines indicates the sequence of tasks.

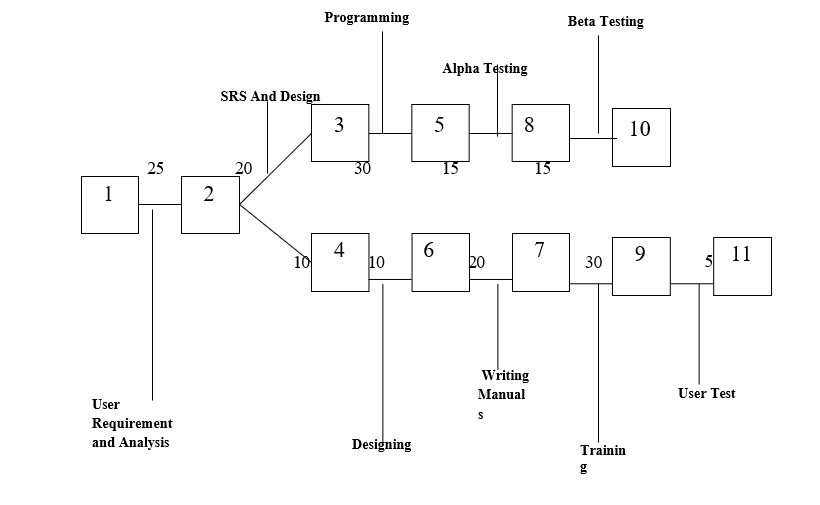


Figure 4.1 Pert Chart

**CHAPTER 5**

**HARDWARE & SOFTWARE SPECIFICATION**

### Hardware Specification

### Server:

Processor: Intel Core i5 or equivalent RAM: 8 GB or higher

Storage: 256 GB SSD or higher

### Database Server:

Processor: Intel Core i5 or equivalent RAM: 8 GB or higher

Storage: 256 GB SSD or higher Network Interface: Gigabit Ethernet

**Client Machines:**

Processor: Intel Core i3 or equivalent

RAM: 4 GB or higher

Storage: 128 GB SSD or higher

Network Interface: 100 Mbps Ethernet or Wi-Fi

### Software Specification

### It is developed with the Help of HTML, CSS, JS

### Server-Side Technologies:

### Operating System: Windows Server 2016 or later

### Client-Side Technologies:

Web Browser: Latest versions of Chrome, Firefox, Safari, or Edge Client-Side Scripting: JavaScript

### Development Tool:

Integrated Development Environment (IDE): Visual Studio Code

### Version Control:

Git: Version control for collaborative development

**Security:**

SSL/TLS: Ensure secure data transmission over the network Firewall: Implement firewall rules to restrict unauthorized access

Anti-malware Software: Regularly updated anti-malware software on server and client machines

**CHAPTER 6**

**CHOICE OF TOOLS & TECHNOLOGY**

**HTML, CSS**

HTML (Hypertext Markup Language) is a standard language used for creating web pages. It defines the structure of content using elements like headings, paragraphs, and links, formatted with tags.

CSS (Cascading Style Sheets) is a language used for styling web pages. It controls the presentation of HTML elements, defining attributes such as color, layout, and typography. CSS enhances the visual appearance and layout of websites.

**JavaScript**

JavaScript is a versatile programming language primarily used for web development. It enables interactive features, dynamic content updates, and behavior changes on web pages. JavaScript runs on the client side, executing scripts within a web browser to enhance user experience, validate forms, and interact with server-side data asynchronously.

### Data Flow Diagram

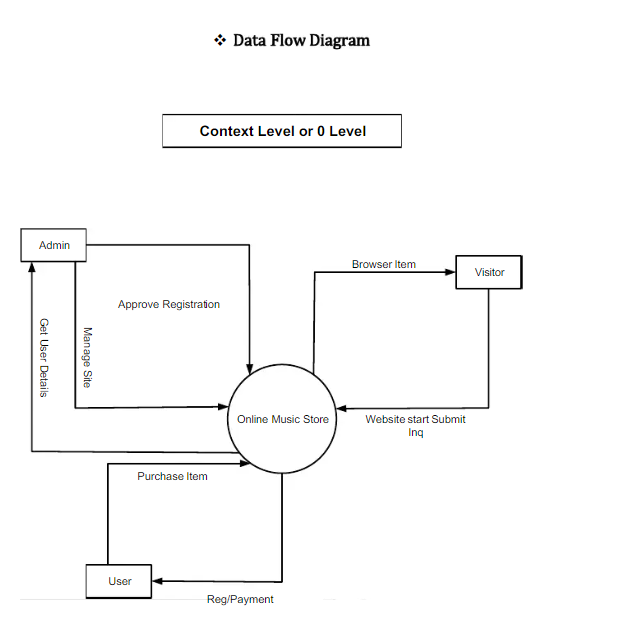
The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It’s also known as BUBBLE CHART. The purpose of DFD is major transformation that will become in system design symbols used in DFD: -

In the DFD, four symbols are used and they are as follows.

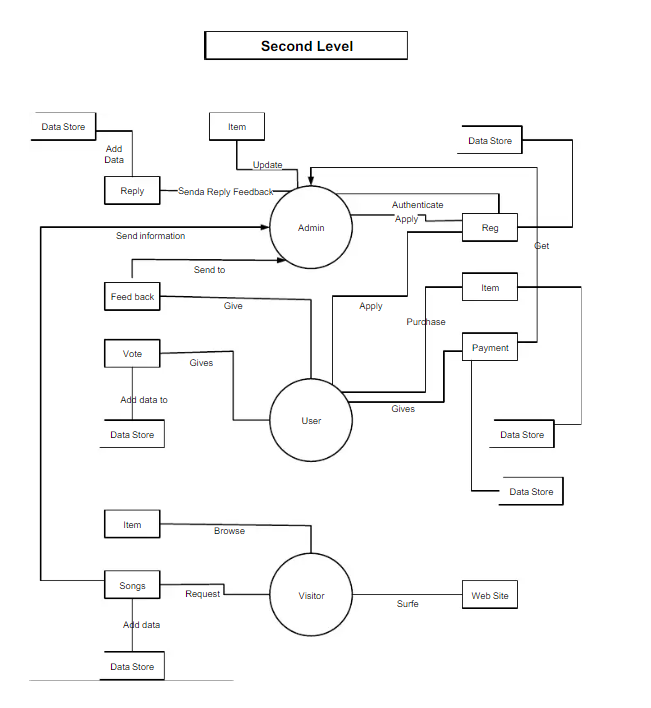
* + 1. A square defines a source (originator) or destination of system data.
    2. An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.
    3. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.
    4. An open rectangle is a data store-data at rest, or a temporary repository of data.

### Context Level Diagram

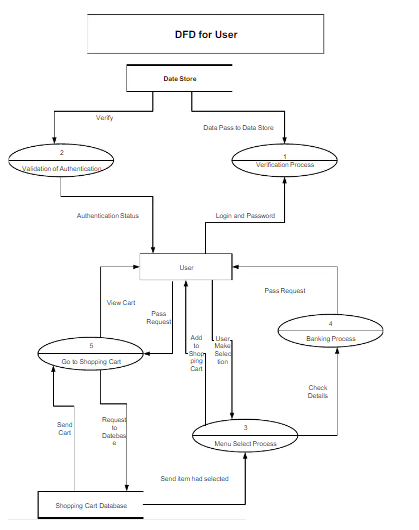
This level shows the overall context of the system and its operating environment and shows the whole system as just one process. Travels and Tales is shown as one process in the context diagram; which is also known as zero level DFD, shown below. The context diagram plays important role in understanding the system and determining the boundaries. The main process can be broken into sub-processes and system can be studied with more detail; this is where 1st level DFD comes into play.



**Fig 6.1** Data Flow Diagram



**Fig 6.2** Level-1 DFD



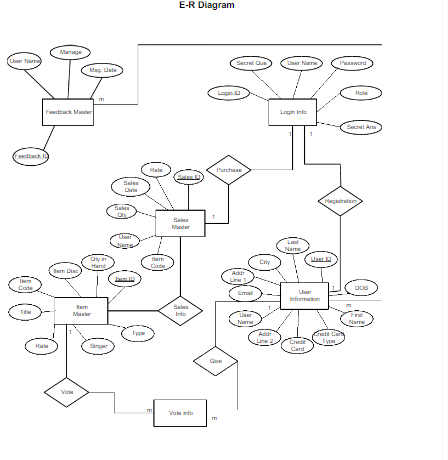
**Fig 6.3** Level-2 DFD

**CHAPTER 7**

**ER-DIAGRAM**

### Entity-relationship model

The entity-relationship model or entity-relationship diagram (ERD) is a data model or diagram for high-level descriptions of conceptual data model, and it provides a graphical notation for representing such data models in the form of entity-relationship diagrams.



**Fig 7.1** E-R Diagram

## 7.2 Class Diagram

## Authentication:

## Classification: Weak Cliet

## Description: Represents users authentication details, including username and password. This class is responsible for users login functionality.

**CHAPTER 8**

**DATABASE**

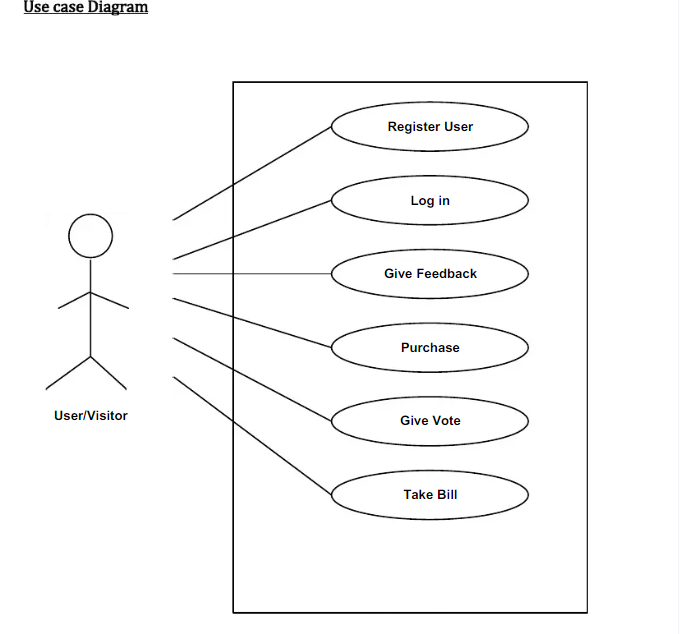
**Use Case Diagram**

A use case diagram is a type of diagram in the Unified Modelling Language (UML) that is used to visualize and describe the functional requirements of a system from an external user's perspective. It provides a high-level view of how users interact with a system and the various functionalities or use cases the system offers in response to those interactions.

Use case diagrams are particularly useful for:

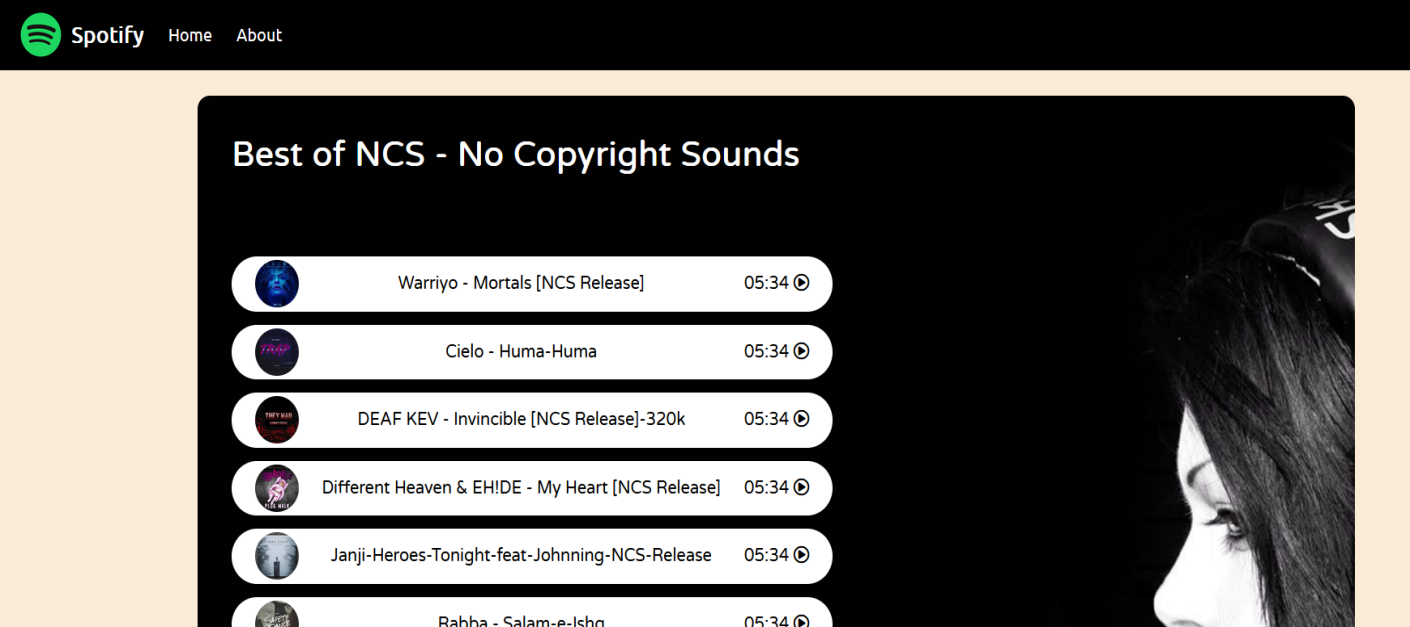
* Communicating the system's functionality and behavior to stakeholders in a visual and understandable way.
* Capturing and documenting high-level user requirements.
* Identifying system boundaries and external interactions.
* Modeling how different use cases relate to each other.

They are a valuable tool in the early stages of software development for understanding and discussing the functional aspects of a system before diving into more detailed design and implementation phases.



**CHAPTER 9**

**FORM DESIGN**



**Fig 9.1** Main page

**About:**

Playlist-Different songs are stored at the local storage of the device

Time duration and Song Icon is also there.

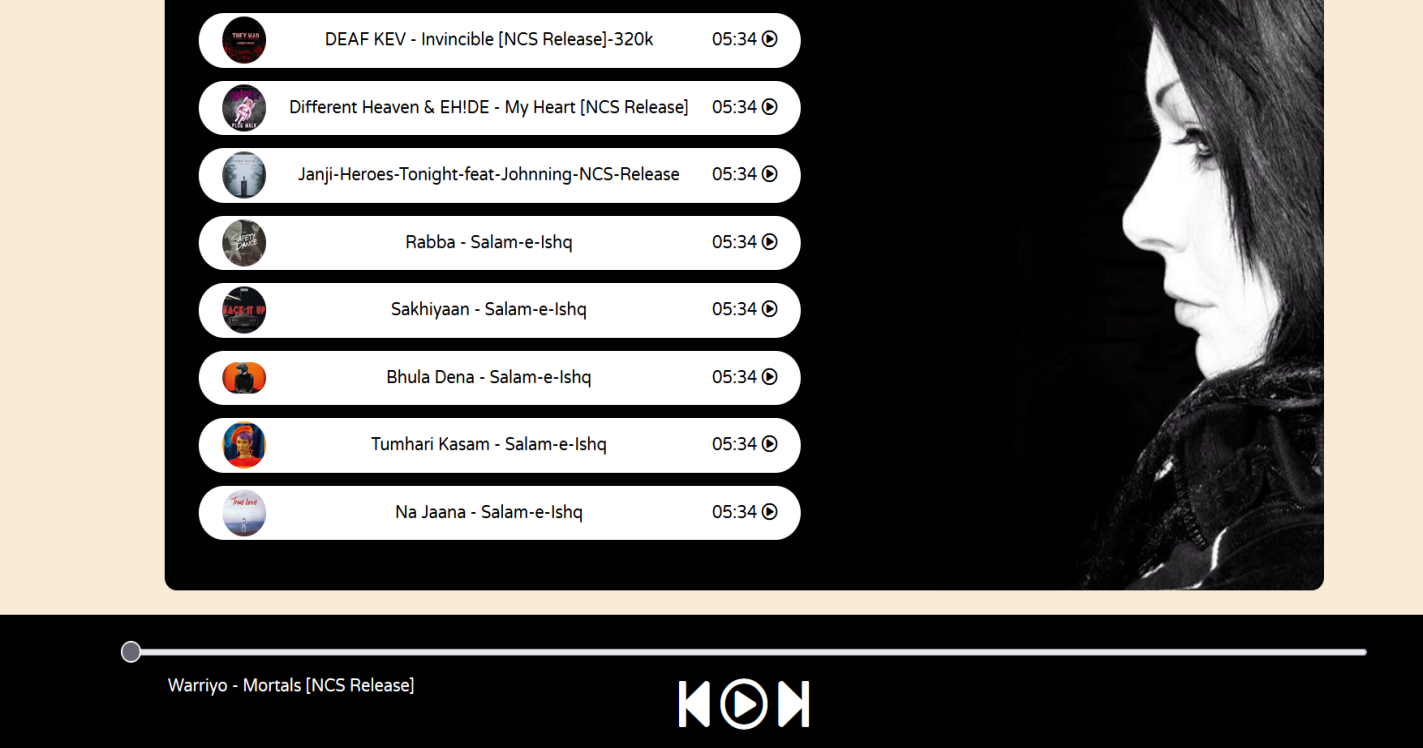


**Fig 9.2** Playlist

**About:**

List of songs which a user can choose from the list

Songs can be Play , Pause , Replay etc



**Fig 9.3** Player

**CHAPTER 10**

**TESTING**

### Introduction

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

### Types of Testing

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

### Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

### System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT**

The project aims to develop a music player clone using HTML, CSS, and JavaScript. It will feature playback controls (play, pause, stop), volume adjustment, playlist management, song information display, and repeat/shuffle options. The player will have a responsive design for compatibility across devices. Non-functional requirements include smooth performance, scalability for large libraries, reliability with robust error handling, usability with an intuitive interface, security for data protection, compatibility across browsers/devices, and maintainability through well-documented code. Additional features may include equalizer, lyrics display, offline support, social integration, and custom themes. Stretch goals may involve API integration, advanced audio effects, user authentication, collaboration features, and voice control.

**CONCLUSION & REFERNCES**

In conclusion, the Music Player clone presents an impressive replication of the popular Spotify platform, offering a seamless and enjoyable music streaming experience for users. With its user-friendly interface, extensive music library, and personalized playlists, it successfully captures the essence of the original service. The clone's responsive design ensures accessibility across various devices, enhancing user convenience.

While the Music Player clone excels in emulating key features of Spotify, it also introduces innovative elements, such as enhanced social integration or unique playlist curation algorithms, setting it apart and adding value for users. The development team's dedication to ensuring a smooth and bug-free experience contributes to the overall positive impression of the clone.

It is worth noting that continuous updates and improvements will be crucial for the Music Player clone to stay competitive in the dynamic music streaming industry. Regularly adding new features, expanding the music catalogue, and refining the user experience will be essential to keep users engaged and attract a growing user base.

Ultimately, the Music Player clone not only serves as a testament to the capabilities of modern technology in replicating successful platforms but also offers a promising alternative for music enthusiasts seeking a high-quality streaming experience.

References

<https://www.w3schools.com/>

https://www.javatpoint.com/