Music Player

A PROJECT REPORT
for
Mini Project (KCA353)
Session (2023-24)

Submitted by

Ayushi Singh
University Roll No
2200290140047

Submitted in partial fulfilment of the Requirements for the Degree of

MASTER OF COMPUTER APPLICATION

Under the Supervision of
Dr Vipin Kumar
(Associate Professor)

Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206 (FEBRUARY 2024)

CERTIFICATE

Certified that **Ayushi Singh (2200290140047)** have 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: **Ayushi Singh(2200290140047)**

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

Date:

Dr Vipin Kumar
Associate Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head
Department of Computer Applications
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 favourite 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.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr Vipin Kumar** for her 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.

Ayushi Singh

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background

The Music Player Clone project aims to replicate the core functionalities and user 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.

1.2 Project Overview

The project aims to replicate basic functionality of popular music players, 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 browsers. Focus on clean code structure and intuitive user interface design. Incorporate icons and animations to enhance user experience. Finally, test thoroughly for functionality and design consistency. This project serves as a practical exercise in front-end web development, honing skills in HTML and CSS.

1.3 Objective

The primary objectives of a Music Player I include:

Efficiency: Optimizing the search.
User Convenience: Searching of the songs without any interuption.

1.4 Key Features

1-User Authentication-

- Allow users to create account login securely.
- Implement password encryption.

2-Music Catalogue-

- Integrate a vast music catalogue with a diverse range of artists, albums, and genres.

- Utilize an external API (such as Spotify API) to fetch and display music metadata.

3- Search Functionality-

- Implement a robust search feature allowing users to find songs, artists, and albums quickly.
- Include autocomplete suggestions for a seamless user experience.

4-User Playlist Management-

- Enable users to create, edit, and delete playlists.
- Allow users to add or remove songs from their playlists.

5-Audio Player-

- Develop a responsive audio player with play, pause, skip, and volume control functionalities.
- Display song progress, duration, and album artwork during playback

6- User Recommendations-

- Implement a recommendation system ad on user preferences and listening history.
- Provide personalized playlists and suggestions for users.

7-Responsive Design-

- Ensure the application is accessible and visually appealing across various devices and screen sizes.

8-User Interaction and Feedback-

- Incorporate features like liking songs, following artists, and commenting on tracks.
- Provide feedback to users through notifications and visual cues.

9-User Settings-

- Allow users to customize their profile settings, including display name, profile picture, and notification preferences.

1.5 Scope of the project

Here is the project scope Music Player Clone

1.Playlist Management:

- Users can create, edit, and delete playlists.
- Ability to add or remove songs from playlists.

2.Audio Player Features:

- Responsive audio player with play, pause, skip, and volume control.
- Display song progress, duration, and album artwork during playback.

3.User Recommendations:

- Implementation of a recommendation system based on user preferences and listening history.
- Personalized playlists and song suggestions for users.

4.Social Features:

- Users can follow other users.
- Social sharing options for sharing favourite songs or playlists.

5.User Interaction and Engagement:

- Users can like, comment, and share songs.

- Notifications for new releases, followers, and user interactions.

6.User Profile Settings:

- Customization of user profiles, including display names, profile pictures, and notification preferences.

7.Responsive Design:

- Ensure the application is visually appealing and functional on various devices and screen sizes.

8.Offline Mode (Optional):

- Implementation of an offline mode for users to listen to downloaded music without an internet connection.

9.Advanced Search and Filters (Optional):

- Advanced search options and filters for users to refine their music discovery.

10.Analytics Dashboard (Optional):

CHAPTER 2

PROBLEM IDENTIFICATION & FEASIBILITY STUDY

2.1 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.

2.2 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

A **feasibility study** is an assessment of the practicality of a project or system. A feasibility study aims to objectively and rationally uncover the strengths and weaknesses of an existing business or proposed venture, opportunities and threats present in the natural environment, the resources required to carry through, and ultimately the prospects for success. In its simplest terms, the two criteria to judge feasibility are cost required and value to be attained.

A well-designed feasibility study should provide a historical background of the business or project, a description of the product or service, accounting statements, details of the operations and management, marketing research and policies, financial data, legal requirements and tax obligations. Generally, feasibility studies precede technical development and project implementation. A feasibility study evaluates the project's potential for success; therefore, perceived objectivity is an important factor in the credibility of the study for potential investors and lending institutions. It must therefore be conducted with an objective, unbiased approach to provide information upon which decisions can be based

2.2.1 Technical Feasibility

This assessment is based on an outline design of system requirements, to determine whether the company has the technical expertise to handle completion of the project. When writing a feasibility report, the following should be taken to consideration:

- A brief description of the business to assess more possible factors which could affect the study
- The part of the business being examined
- The human and economic factor
- The possible solutions to the problem

At this level, the concern is whether the proposal is both *technically* and legally feasible (assuming moderate cost)

The technical feasibility assessment is focused on gaining an understanding of the present technical resources of the organization and their applicability to the expected needs of the proposed system. It is an evaluation of the hardware and software and how it meets the need of the proposed system

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.

2.2.2 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.

The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives about the development schedule, delivery date, and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability, etc. These parameters are required to be considered at the early stages of the design if desired operational behaviours are to be realised. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that must be integral to the early design phases.

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.

2.2.3 Economic Feasibility

Cost Estimation: Estimate the costs associated with development, hosting, maintenance, and any licensing fees for third-party services.

Revenue Model: Explore potential revenue streams, such as subscription plans, advertisements, or partnerships.

2.2.4 Financial feasibility

In case of a new project, financial viability can be judged on the following parameters:

- Total estimated cost of the project
- Financing of the project in terms of its capital structure, debt to equity ratio and promoter's share of total cost
- Existing investment by the promoter in any other business
- Projected cash flow and profitability

The financial viability of a project should provide the following information

- Full details of the assets to be financed and how liquid those assets are.
- Rate of conversion to cash-liquidity (i.e., how easily the various assets can be converted to cash).
- Project's funding potential and repayment terms.
- Sensitivity in the repayments capability to the following factors:

- Mild slowing of sales.
- Acute reduction/slowing of sales.
- Small increase in cost.
- Large increase in cost.
- Adverse economic conditions.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 Functional Requirements

Playback Controls:

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

Volume Control:

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

Playlist Management:

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

Time Display:

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

Repeat and Shuffle:

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

Song Information:

1. Display metadata such as song title, artist, album, and album art.

2. Allow users to view detailed information about the currently playing song.

Playback State Management:

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

Keyboard Controls:

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

Mobile Compatibility:

1. 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.

3.2 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.

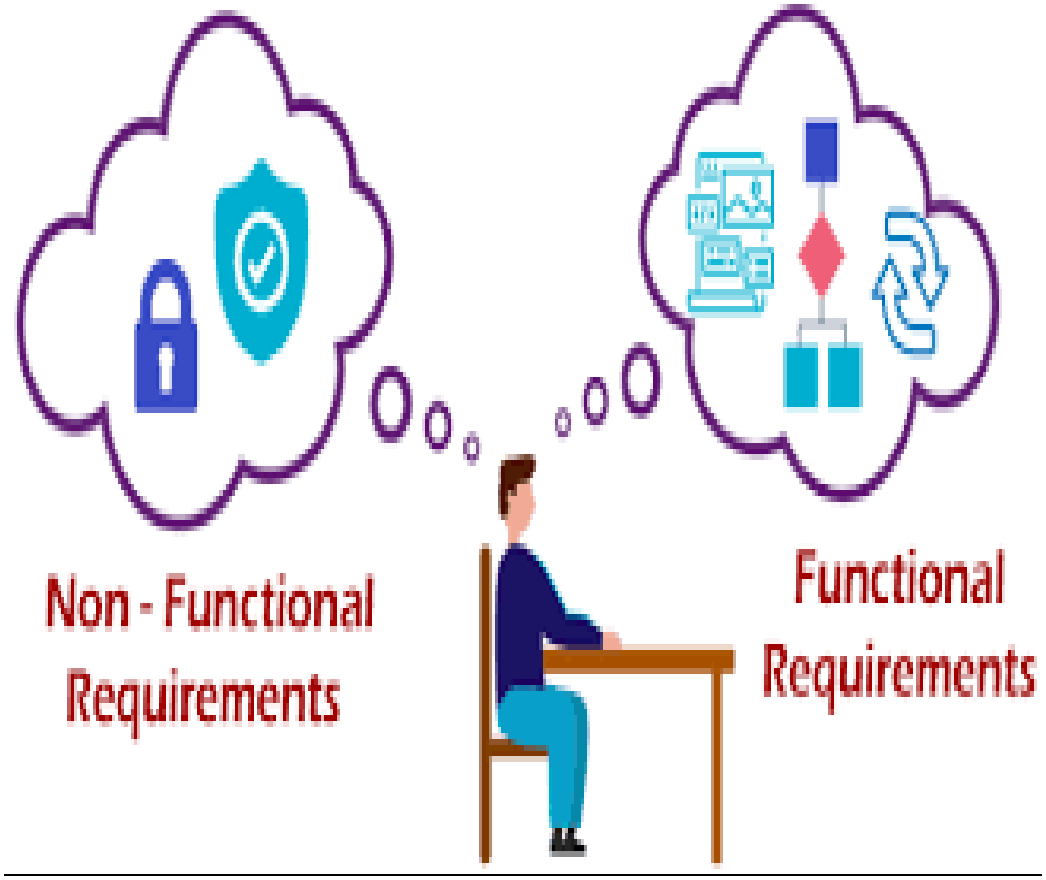


FIG 3.0

FUNCTIONAL AND NON FUNCTIONAL

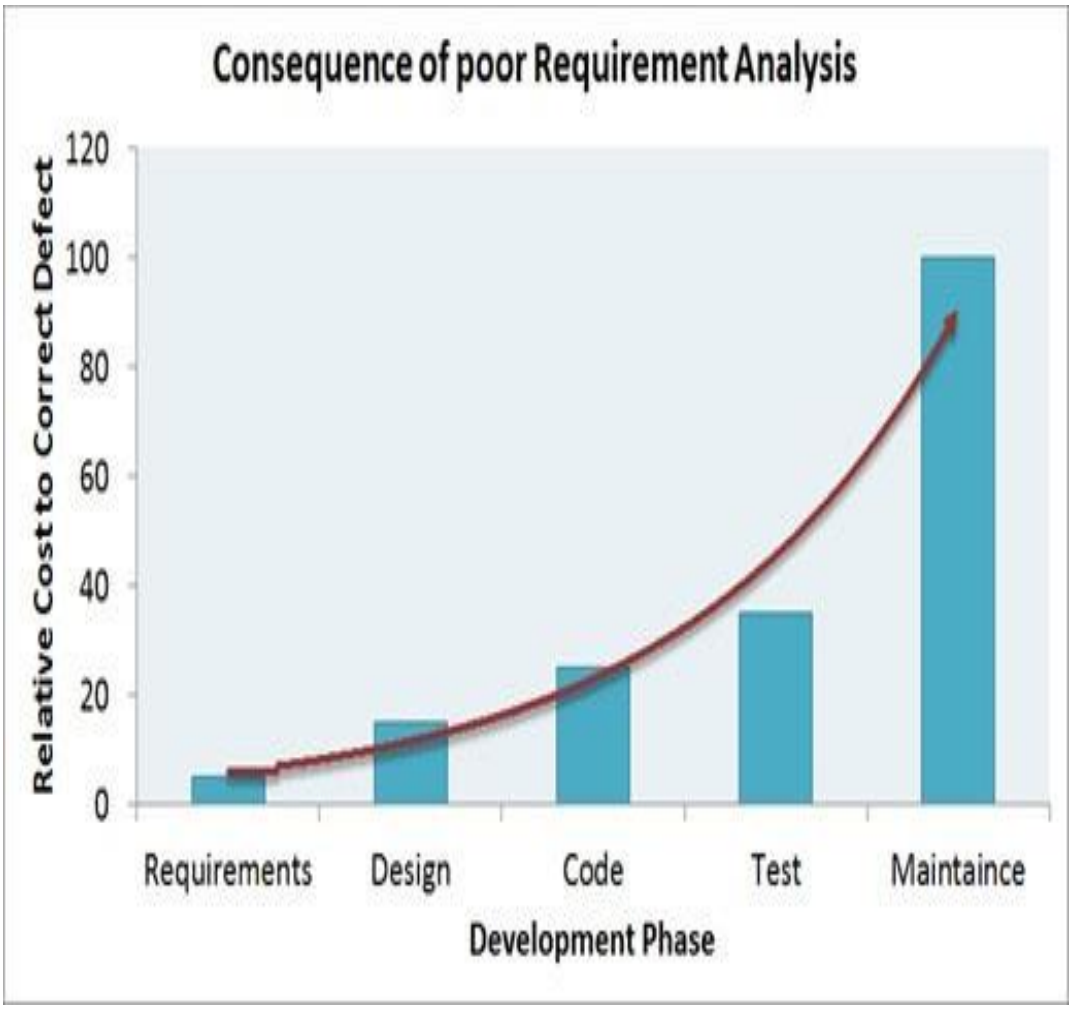


FIG 3.1

Requirement Analysis

CHAPTER 4

PROJECT PLANNING AND SCHEDULING

Pert Chart:

PERT (Program Evaluation Review Technique) charts are indispensable tools in project management, offering a graphical depiction of project tasks, milestones, and interdependencies. These charts employ numbered nodes to represent project events or milestones and directional arrows to illustrate task sequences and dependencies. PERT charts integrate three time estimates for each task: optimistic, pessimistic, and most likely, enabling project managers to calculate expected durations and conduct probabilistic analysis for deadline assessment.

Critical path analysis, a core aspect of PERT charts, identifies the longest path of dependent tasks, determining the project's minimum duration. Tasks along this critical path are crucial, as any delays directly impact the overall project timeline. Understanding task dependencies is paramount for effective scheduling and resource allocation, both of which are facilitated by PERT charts.

Additionally, PERT charts serve as powerful communication tools, providing stakeholders with a visual representation of the project's timeline, milestones, and dependencies. This visual clarity fosters better understanding and alignment among team members, clients, and other stakeholders.

Moreover, PERT charts support iterative planning by allowing for updates and adjustments as the project progresses. Project managers can track progress, monitor deviations from the planned schedule, and make informed decisions to mitigate risks and ensure project success.

In summary, PERT charts play a vital role in project management by aiding in planning, scheduling, and coordination. They empower project managers to visualize project workflows, identify critical tasks, manage dependencies, communicate effectively, and adapt to changing project conditions, ultimately leading to efficient project execution and delivery.

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

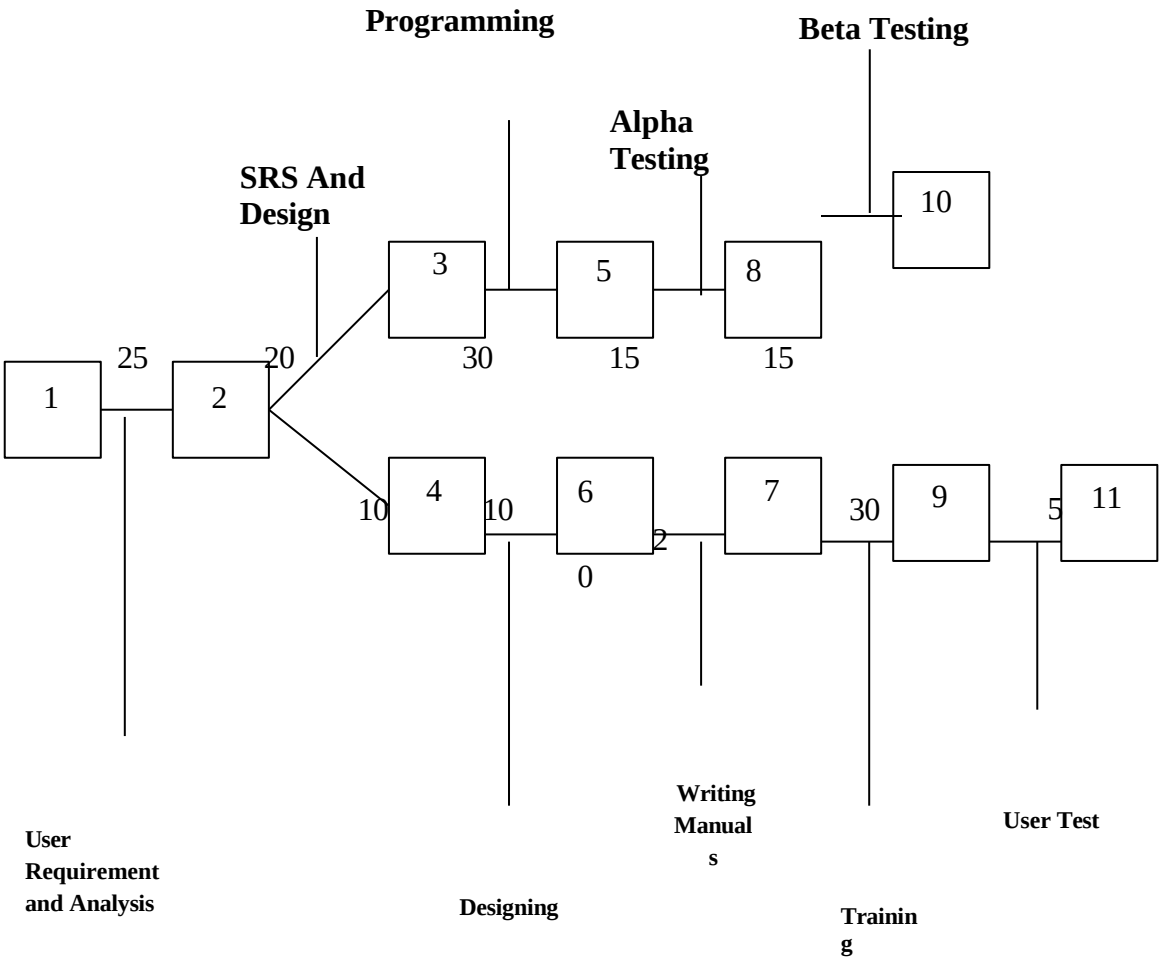


fig 4.0
PERT CHART

CHAPTER 5

HARDWARE & SOFTWARE SPECIFICATION

5.1 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

5.2 Software Specification

It is develped with the Help of HTML,CSS,JS

Server-Side Technologies:

Operating System: Windows Server 2016 or later

Web Server

Server-Side Scripting Language: JavaScript

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.

6.1 Data Flow Diagram

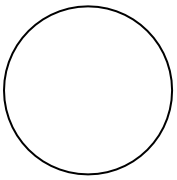
The data flow diagram shows the flow of data within any system. It is an important tool for designing phase of software engineering. Larry Constantine first developed it. It represents graphical view of flow of data. It's also known as BUBBLE CHART. The purpose of DFD is major transformation that will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

An arrow identifies data flow-data in motion. It is 2a pipeline through which information flows.

2. A circle or a “bubble “(Some people use an oval bubble) represents a process that transfers informing data flows into outgoing data flows.



3.An open rectangle is a data store-data at rest, or a temporary repository of data.

6.2 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.

❖ **Data Flow Diagram**

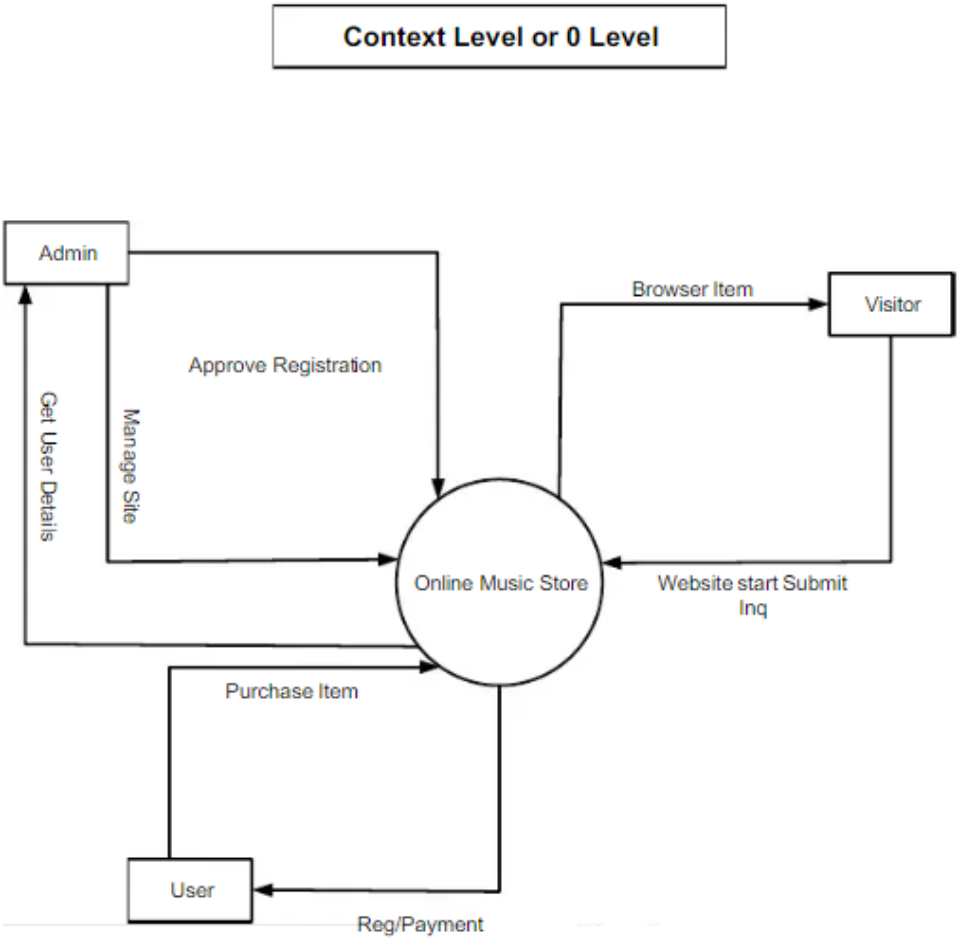


Fig 6.1

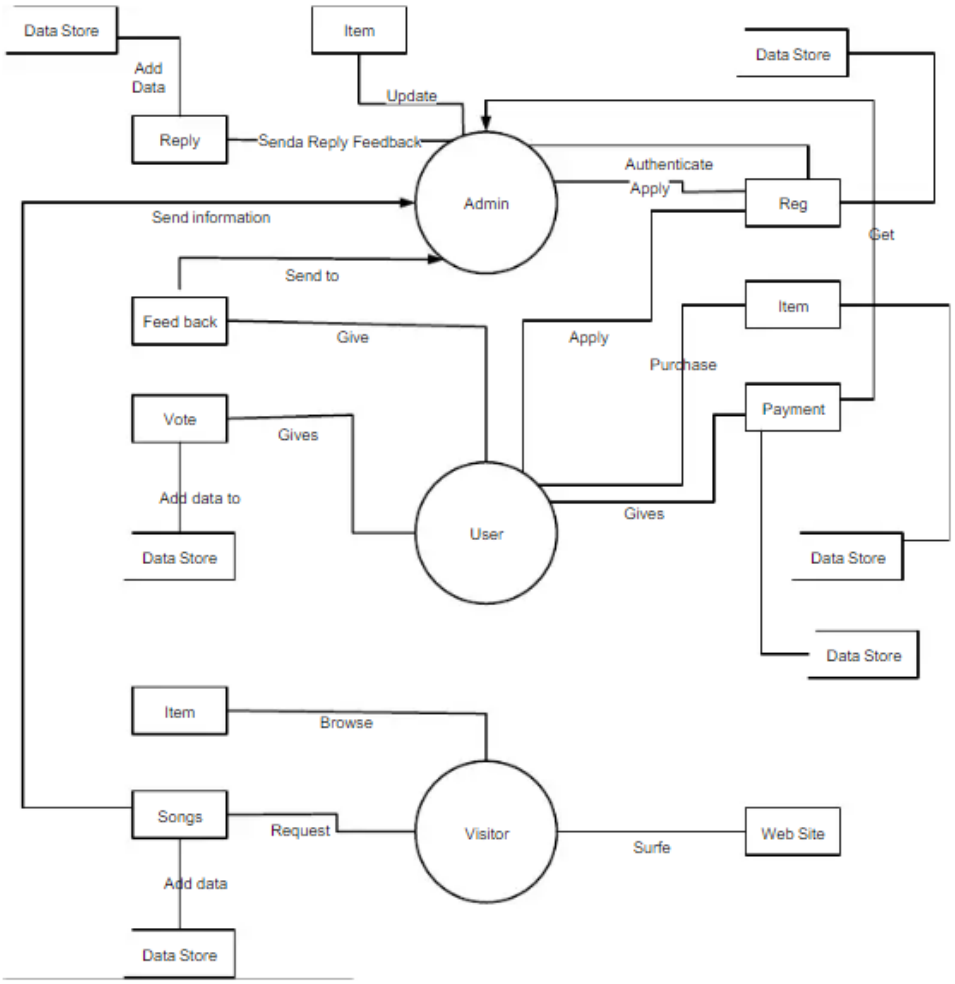
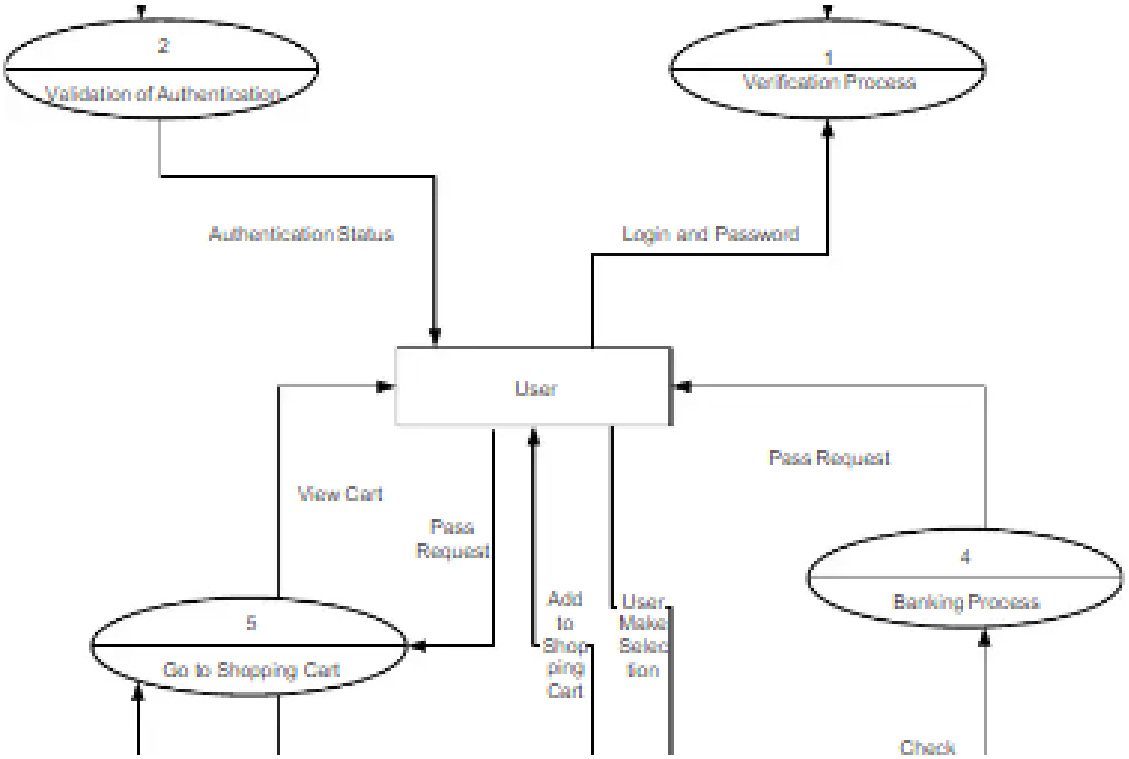


Fig .6.2 DFD Level 1

2nd Level DFD

Fig 6.3



CHAPTER 7

ER-DIAGRAM

7.1 Entity-relationship model: -

Entity-Relationship (ER) modeling is a widely used technique in software engineering and database design for modeling the logical structure of a database. It represents the relationships between different entities within a system and helps in understanding the data requirements and relationships in a clear and organized manner.

Here are the key components of entity-relationship modeling:

Entity: An entity is a real-world object, concept, or thing that has attributes describing its properties. In a database context, entities are represented as tables. Each row in the table represents an instance of the entity, and each column represents an attribute of the entity.

Attribute: An attribute is a property or characteristic of an entity that describes it. For example, in a "Student" entity, attributes could include "student ID," "name," "age," and "grade." Attributes are represented as columns in the entity's table.

Relationship: A relationship defines the connection or association between two or more entities. Relationships are represented as lines connecting entities in an ER diagram. They indicate how entities are related to each other and the cardinality of the relationship (e.g., one-to-one, one-to-many, many-to-many).

Cardinality: Cardinality specifies the number of instances of one entity that are associated with the number of instances of another entity. It helps define the nature of the relationship between entities. Cardinality constraints include one-to-one, one-to-many, and many-to-many relationships.

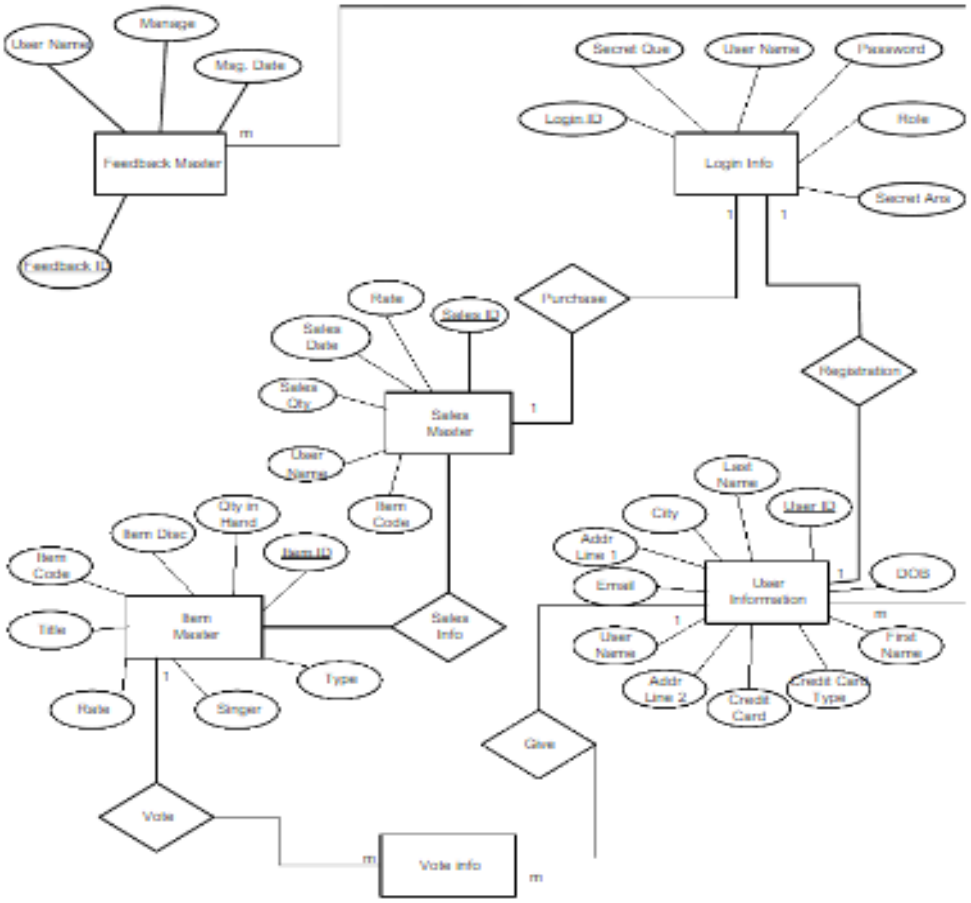
Primary Key: A primary key is a unique identifier for each record (instance) in an entity. It uniquely identifies each row in the entity's table and ensures data integrity and consistency. Primary keys are typically represented by underlining the attribute in an ER diagram.

Foreign Key: A foreign key is an attribute or set of attributes in one entity that refers to the primary key of another entity. It establishes a link between two tables and enforces referential integrity. Foreign keys are represented in ER diagrams as lines connecting attributes in different entities.

ER Diagram: An ER diagram is a graphical representation of the entities, attributes, relationships, and constraints in a database. It provides a visual overview of the database structure and helps in communicating the design to stakeholders.

Entity-relationship modeling is an essential part of the database design process, enabling developers and designers to create well-organized, efficient, and maintainable databases that accurately represent the underlying data requirements and relationships.

E-R Diagram


User Name
Manage
Msg. Date
Feedback Master
m
Feedback ID
Secret Que
User Name
Password
Login ID
Role
Login Info
Secret Ans
1
1
Purchase
Registration
Rate
Sales ID
Sales Date
Sales Qty
Sales Master
1
User Name
Item Code
City in Hand
Item Desc
Item ID
Item Code
Item Master
Title
Rate
Singer
Type
1
Sales Info
Vote
m
Vote Info
m
Give
Last Name
City
User ID
Addr Line 1
Email
User Information
1
DOB
m
User Name
Addr Line 2
Credit Card
Credit Card Type
First Name

7.2 Class Diagram: -

Authentication:

- Classification: Weak Class
- Description: Represents user authentication details, including username and password. This class is responsible for user login functionality.

User:

- Classification: Strong Class
- Description: Represents the users of the system, including administrators and employees.

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 behaviour to stakeholders in a visual and understandable way.

- Capturing and documenting high-level user requirements.
- Identifying system boundaries and external interactions.
- modelling 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.

Use case Diagram

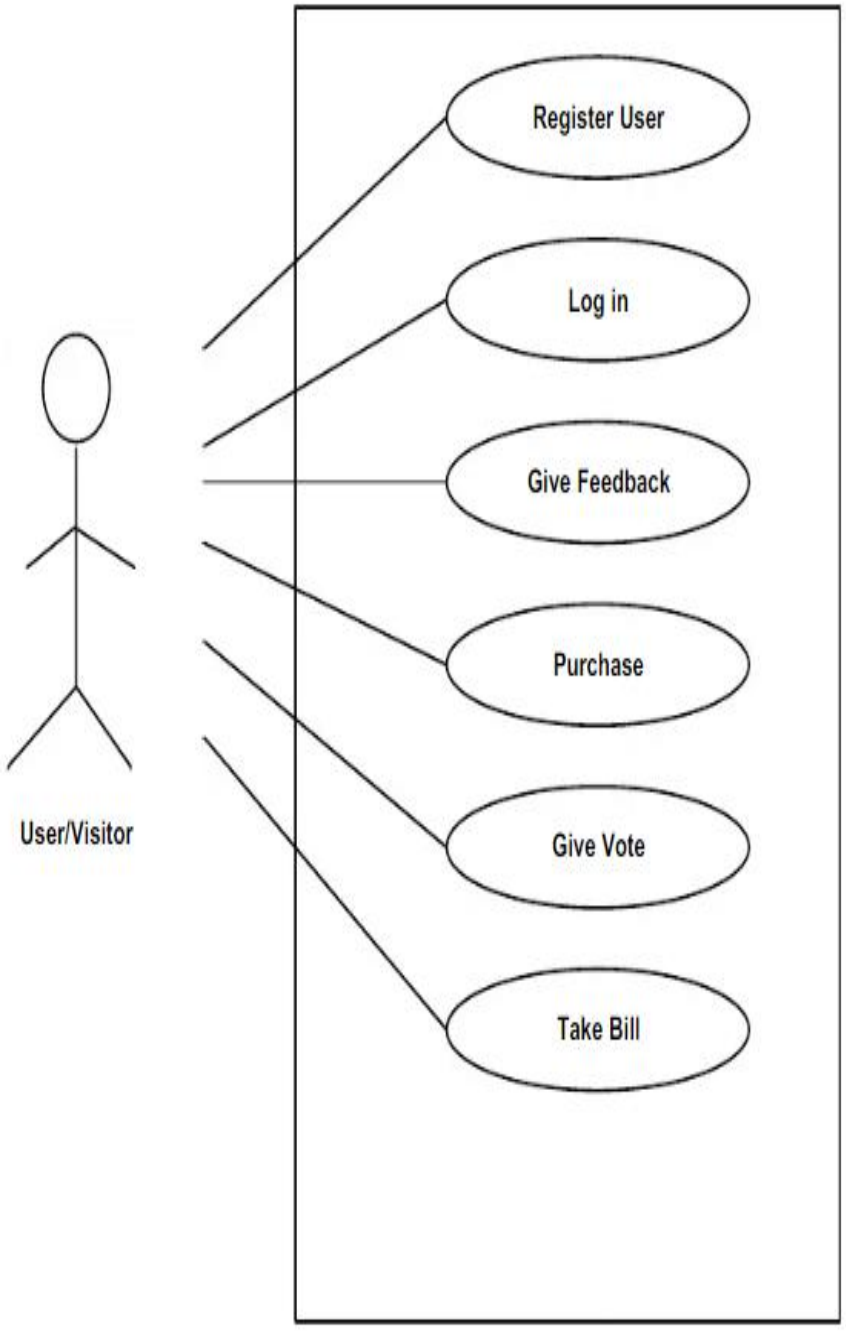


Fig 8.0 Use Case diagram

.

CHAPTER 9 FORM DESIGN

Fig 9.1

Main page

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

Time duration and Song Icon is also there.

Fig 9.2

Playlist

List of songs which a user can choose from the list

Songs can be Play , Pause , Replay etc

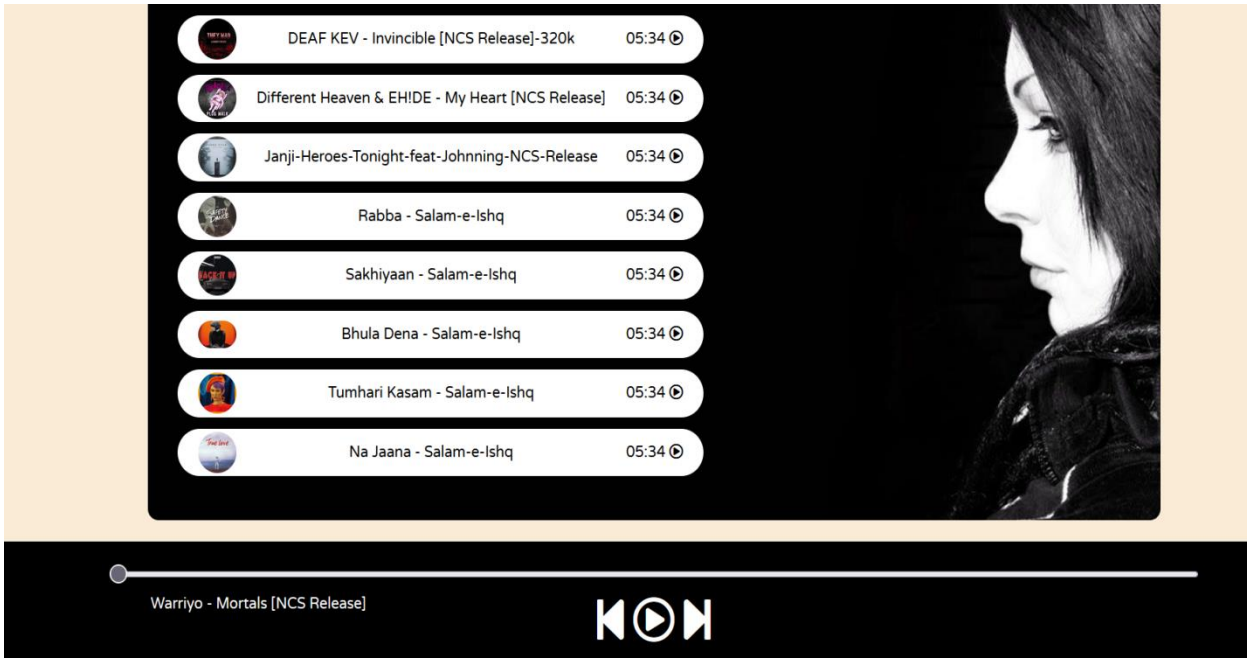


Fig 9.3

Play and Pause

CHAPTER 10

CODING

HTML:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="UTF-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
      <title>Spotify - Your favourite music is here</title>
      <link rel="stylesheet" href="style.css">
      </head>
<body>
   <nav>
      <ul>
         <li class="brand"><img src="logo.png" alt="Spotify"> Spotify</li>
         <li>Home</li>
         <li>About</li>
      </ul>
   </nav>

   <div class="container">
```

```
<div class="songList">
    <h1>Best of NCS - No Copyright Sounds</h1>
    <div class="songItemContainer">
        <div class="songItem">
            <img alt="1">
            <span class="songName">Let me Love You</span>
            <span class="songlistplay"><span class="timestamp">05:34 <i id="0" class="far songItemPlay fa-play-circle"></i> </span></span>
        </div>
        <div class="songItem">
            <img alt="1">
            <span class="songName">Let me Love You</span>
            <span class="songlistplay"><span class="timestamp">05:34 <i id="1" class="far songItemPlay fa-play-circle"></i> </span></span>
        </div>
        <div class="songItem">
            <img alt="1">
            <span class="songName">Let me Love You</span>
            <span class="songlistplay"><span class="timestamp">05:34 <i id="2" class="far songItemPlay fa-play-circle"></i> </span></span>
        </div>
        <div class="songItem">
            <img alt="1">
            <span class="songName">Let me Love You</span>
            <span class="songlistplay"><span class="timestamp">05:34 <i id="3" class="far songItemPlay fa-play-circle"></i> </span></span>
```

```
            </div>
            <div class="songItem">
                <img alt="1">
                <span class="songName">Let me Love You</span>
                <span class="songlistplay"><span class="timestamp">05:34 <i id="4" class="far
songItemPlay fa-play-circle"></i> </span></span>



            </div>
            <div class="songItem">
                <img alt="1">
                <span class="songName">Let me Love You</span>
                <span class="songlistplay"><span class="timestamp">05:34 <i id="5" class="far
songItemPlay fa-play-circle"></i> </span></span>
            </div>


            <div class="songItem">
                <img alt="1">
                <span class="songName">Let me Love You</span>
                <span class="songlistplay"><span class="timestamp">05:34 <i id="6" class="far
songItemPlay fa-play-circle"></i> </span></span>
            </div>
```

```
<div class="songItem">
    <img alt="1">
    <span class="songName">Let me Love You</span>
    <span class="songlistplay"><span class="timestamp">05:34 <i id="7" class="far
songItemPlay fa-play-circle"></i> </span></span>
</div>

<div class="songItem">
    <img alt="1">
    <span class="songName">Let me Love You</span>
    <span class="songlistplay"><span class="timestamp">05:34 <i id="8" class="far
songItemPlay fa-play-circle"></i> </span></span>
</div>

<div class="songItem">

    <img alt="1">
    <span class="songName">Let me Love You</span>

    <span class="songlistplay"><span class="timestamp">05:34 <i id="9"
class="far songItemPlay fa-play-circle"></i> </span></span>
```

```
            </div>

        </div>

    </div>

    <div class="songBanner"></div>

</div>

 <div class="bottom">

    <input type="range" name="range" id="myProgressBar" min="0" value="0" max="100">

    <div class="icons">

        <!-- fontawesome icons -->

        <i class="fas fa-3x fa-step-backward" id="previous"></i>

        <i class="far fa-3x fa-play-circle" id="masterPlay"></i>

        <i class="fas fa-3x fa-step-forward" id="next"></i>

    </div>
```

```
        <div class="songInfo">
            <img src="playing.gif" width="42px" alt="" id="gif"> <span
id="masterSongName">Warriyo - Mortals [NCS Release]</span>

        </div>

    </div>

    <script src="script.js"></script>

    <script src="https://kit.fontawesome.com/26504e4a1f.js" crossorigin="anonymous">

    </script>

</body>

</html>
```

CSS:

```
@import url('https://fonts.googleapis.com/css2?family=Ubuntu&display=swap');
@import url('https://fonts.googleapis.com/css2?family=Varela+Round&display=swap');
body{
background-color: antiquewhite;

}
*{
margin: 0;

    padding: 0;
}
nav{

    font-family: 'Ubuntu', sans-serif;

}

nav ul{

    display: flex;
    align-items: center;

    list-style-type: none;
```

```
    height: 65px;

    background-color: black;

    color: white;

}




nav ul li{

    padding: 0 12px;
}
.brand img{

    width: 44px;

    padding: 0 8px;

}
.brand {

    display: flex;

    align-items: center;
```

```
    font-weight: bolder;
    font-size: 1.3rem;

}

.container{

    min-height: 72vh;

    background-color: black;

    color: white;

   font-family: 'Varela Round', sans-serif;

   display: flex;
   margin: 23px auto;
   width: 70%;
   border-radius: 12px;
   padding: 34px;
   background-image: url('bg.jpg');
}

.bottom{
```

```
    position: sticky;
    bottom: 0;
    height: 130px;
    background-color: black;
    color: white;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
}

.icons{
    margin-top: 14px;
}
.icons i{
    cursor: pointer;
}

#myProgressBar{
    width: 80vw;
    cursor: pointer;
}

.songItemContainer{
    margin-top: 74px;
}
```

```
.songItem{
    height: 50px;
    display: flex;
    background-color: white;
    color: black;
    margin: 12px 0;
    justify-content: space-between;
    align-items: center;
     border-radius: 34px;

}
.songItem img{

    width: 43px;
    margin: 0 23px;
    border-radius: 34px;
}

.timestamp{

     margin: 0 23px;

}

.timestamp i{
```

```
    cursor: pointer;


}
.songInfo{
   position: absolute;
   left: 10vw;
   font-family: 'Varela Round', sans-serif;
}


.songInfo img{
   opacity: 0;
   transition: opacity 0.4s ease-in;
}




@media only screen and (max-width: 1100px) {
   body {
     background-color: red;
   }
  }
```

Java Script:

```
console.log("Welcome to Spotify");


// Initialize the Variables
let songIndex = 0;
let audioElement = new Audio('songs/1.mp3');
let masterPlay = document.getElementById('masterPlay');
let myProgressBar = document.getElementById('myProgressBar');
let gif = document.getElementById('gif');
let masterSongName = document.getElementById('masterSongName');
let songItems = Array.from(document.getElementsByClassName('songItem'));


let songs = [

    {songName: "Warriyo - Mortals [NCS Release]", filePath: "songs/1.mp3", coverPath:
"covers/1.jpg"},
    {songName: "Cielo - Huma-Huma", filePath: "songs/2.mp3", coverPath: "covers/2.jpg"},
    {songName: "DEAF KEV - Invincible [NCS Release]-320k", filePath: "songs/3.mp3", coverPath:
"covers/3.jpg"},
    {songName: "Different Heaven & EH!DE - My Heart [NCS Release]", filePath: "songs/4.mp3",
coverPath: "covers/4.jpg"},
    {songName: "Janji-Heroes-Tonight-feat-Johnning-NCS-Release", filePath: "songs/5.mp3",
coverPath: "covers/5.jpg"},
    {songName: "Rabba - Salam-e-Ishq", filePath: "songs/2.mp3", coverPath: "covers/6.jpg"},
    {songName: "Sakhiyaan - Salam-e-Ishq", filePath: "songs/2.mp3", coverPath: "covers/7.jpg"},
```

```
    {songName: "Bhula Dena - Salam-e-Ishq", filePath: "songs/2.mp3", coverPath: "covers/8.jpg"},
    {songName: "Tumhari Kasam - Salam-e-Ishq", filePath: "songs/2.mp3", coverPath: "covers/9.jpg"},
    {songName: "Na Jaana - Salam-e-Ishq", filePath: "songs/4.mp3", coverPath: "covers/10.jpg"},
]

songItems.forEach((element, i)=>{
    element.getElementsByTagName("img")[0].src = songs[i].coverPath;
    element.getElementsByClassName("songName")[0].innerText = songs[i].songName;
})


// Handle play/pause click
masterPlay.addEventListener('click', ()=>{

    if(audioElement.paused || audioElement.currentTime<=0){
        audioElement.play();
        masterPlay.classList.remove('fa-play-circle');
        masterPlay.classList.add('fa-pause-circle');
        gif.style.opacity = 1;
    }
    else{
        audioElement.pause();
        masterPlay.classList.remove('fa-pause-circle');
        masterPlay.classList.add('fa-play-circle');
        gif.style.opacity = 0;
    }
```

```
})
// Listen to Events
audioElement.addEventListener('timeupdate', ()=>{

    // Update Seekbar
    progress = parseInt((audioElement.currentTime/audioElement.duration)* 100);
    myProgressBar.value = progress;
})

myProgressBar.addEventListener('change', ()=>{

    audioElement.currentTime = myProgressBar.value * audioElement.duration/100;
})

const makeAllPlays = ()=>{

    Array.from(document.getElementsByClassName('songItemPlay')).forEach((element)=>{
        element.classList.remove('fa-pause-circle');
        element.classList.add('fa-play-circle');
    })
}

Array.from(document.getElementsByClassName('songItemPlay')).forEach((element)=>{
    element.addEventListener('click', (e)=>{
        makeAllPlays();
        songIndex = parseInt(e.target.id);
```

```
        e.target.classList.remove('fa-play-circle');
        e.target.classList.add('fa-pause-circle');
        audioElement.src = `songs/${songIndex+1}.mp3`;
        masterSongName.innerText = songs[songIndex].songName;
        audioElement.currentTime = 0;
        audioElement.play();
        gif.style.opacity = 1;
        masterPlay.classList.remove('fa-play-circle');
        masterPlay.classList.add('fa-pause-circle');
    })

})

document.getElementById('next').addEventListener('click', ()=>{
    if(songIndex>=9){
        songIndex = 0
    }

    else{

        songIndex += 1;
    }

    audioElement.src = `songs/${songIndex+1}.mp3`;
    masterSongName.innerText = songs[songIndex].songName;
    audioElement.currentTime = 0;
```

```
    audioElement.play();
    masterPlay.classList.remove('fa-play-circle');
    masterPlay.classList.add('fa-pause-circle');

})

document.getElementById('previous').addEventListener('click', ()=>{
    if(songIndex<=0){
        songIndex = 0
    }

    else{

        songIndex -= 1;
    }


    audioElement.src = `songs/${songIndex+1}.mp3`;
    masterSongName.innerText = songs[songIndex].songName;
    audioElement.currentTime = 0;
    audioElement.play();
    masterPlay.classList.remove('fa-play-circle');
    masterPlay.classList.add('fa-pause-circle');
})
```

CHAPTER 11

TESTING

10.1 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.

10.2 Types of Testing

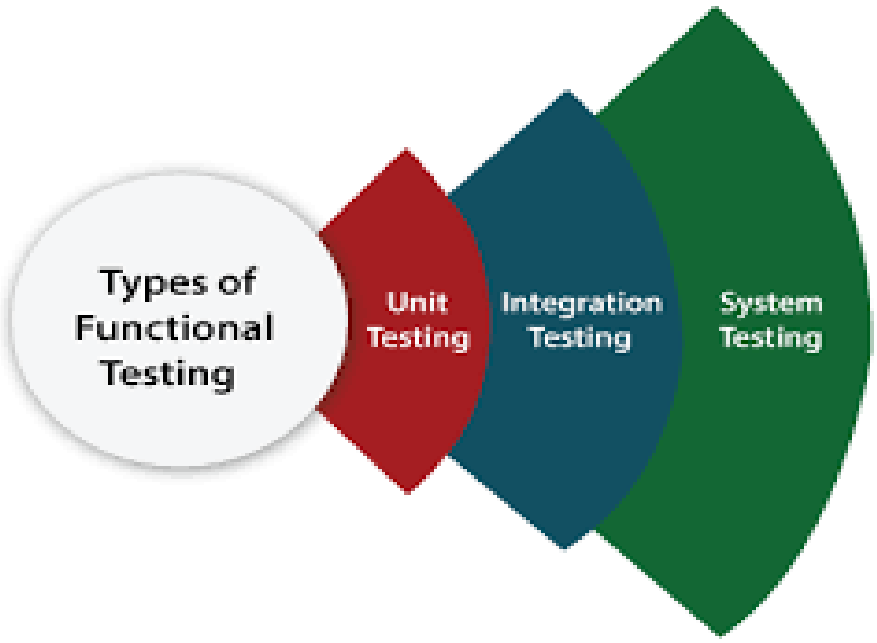


Fig 10.2 Types of Testing

10.2.1 Unit Testing

Unit testing is a fundamental practice in software development aimed at verifying the correctness of individual units or components of a software application. These units are typically the smallest testable parts of the code, such as functions, methods, or classes. The primary goal of unit testing is to ensure that each unit of the software performs as expected in isolation, independent of other parts of the system.

Unit testing is typically performed by developers during the development process, often using testing frameworks and libraries specific to the programming language or platform being used. These frameworks provide tools for writing test cases, executing tests, and reporting results.

Here's how unit testing typically works:

Writing Test Cases: Developers create test cases for each unit of code to be tested. A test case typically includes the inputs to the unit under test and the expected outputs or behaviors.

Executing Tests: The test cases are executed using automated testing tools or frameworks. Each unit is tested in isolation from the rest of the system.

Assertions: Within each test case, assertions are used to compare the actual output of the unit with the expected output. If the actual output matches the expected output, the test passes; otherwise, it fails.

Reporting Results: The results of the unit tests are reported, usually in a format that indicates which tests passed and which ones failed. Developers can use this feedback to identify and fix defects in the code.

Unit testing offers several benefits:

- **Early Detection of Defects**: By testing individual units as they are developed, defects can be detected and fixed early in the development process, reducing the cost of fixing bugs later.

- **Improved Code Quality**: Unit testing encourages developers to write modular, well-structured code that is easier to test, understand, and maintain.

- **Regression Testing**: Unit tests serve as a safety net against regressions. When code is modified or refactored, unit tests can quickly identify any unintended side effects.

- **Facilitates Refactoring**: Unit tests provide confidence that code changes haven't introduced new defects, allowing developers to refactor code with greater ease and confidence.

In summary, unit testing is a critical practice in software development that helps ensure the reliability, maintainability, and quality of software applications. It forms the foundation of a robust testing strategy, complementing other testing techniques such as integration testing and end-to-end testing.

.

10.2.2 Integration Testing

Integration testing is a software testing technique that focuses on verifying the interactions between different modules or components of a software application. Unlike unit testing, which tests individual units of code in isolation, integration testing evaluates how these units work together as a group or subsystem.

The primary goal of integration testing is to uncover defects in the interfaces and interactions between components, ensuring that they integrate correctly and function as intended when combined. This testing phase typically occurs after unit testing and before system testing in the software development lifecycle.

Here's how integration testing typically works:

Identifying Integration Points: Developers identify the integration points between different modules or components of the software application. These integration points are where data or control is exchanged between components.

Defining Test Cases: Test cases are created to verify the interactions between components at the integration points. These test cases may include scenarios to test different data flows, error handling, and boundary conditions.

Testing Strategies: There are several strategies for conducting integration testing, including top-down integration testing, bottom-up integration testing, and incremental integration testing. Each strategy focuses on integrating components in a specific order to gradually build and test larger parts of the system.

Executing Tests: The integration tests are executed using automated testing tools or frameworks. The software application is tested as a whole, with its various components integrated and interacting with each other.

Validation: During test execution, the behavior of the integrated components is validated against the expected behavior. This includes verifying that data is passed correctly between components, that interfaces are working as expected, and that error handling mechanisms are effective.

Reporting Results: The results of the integration tests are reported, indicating which tests passed and which ones failed. Failed tests may indicate defects in the integration logic or incompatibilities between components.

Integration testing offers several benefits:

- **Early Detection of Integration Issues**: Integration testing helps uncover integration issues early in the development process, reducing the risk of larger-scale problems during system testing or production.

- **Improved System Reliability**: By validating the interactions between components, integration testing helps ensure the overall reliability and stability of the software application

.

- **Better Understanding of System Behavior**: Integration testing provides insights into how different parts of the system interact and behave when integrated, helping developers understand the system as a whole.

- **Facilitates Collaboration**: Integration testing encourages collaboration between developers working on different components, fostering communication and coordination to ensure smooth integration.

In summary, integration testing plays a crucial role in validating the interactions between components of a software application, ensuring that they integrate correctly and function as intended. By detecting integration issues early, integration testing contributes to the overall quality and reliability of the software product.

10.2.3 System Testing

System testing is a critical phase in the software testing process that evaluates the overall functionality, performance, and behavior of a software application as a complete and integrated system. Unlike unit testing and integration testing, which focus on individual units or components, system testing examines the entire system as a whole, including its interactions with external systems or dependencies.

The primary objectives of system testing are to ensure that the software meets its specified requirements, functions correctly in its intended environment, and satisfies the needs of its stakeholders. System testing typically occurs after integration testing and before user acceptance testing (UAT) in the software development lifecycle.

Here's how system testing typically works:

Test Environment Setup: The test environment is set up to closely resemble the production environment in which the software will operate. This includes hardware, software, network configurations, and any other relevant components.

Test Case Development: Test cases are developed based on the system requirements, functional specifications, and use cases. These test cases cover various aspects of the system, including functionality, usability, performance, security, and reliability.

Test Execution: The system tests are executed using the developed test cases. This involves running the software application under various scenarios and conditions to validate its behavior and performance.

Functional Testing: Functional testing verifies that the software performs its intended functions correctly and meets the specified requirements. This includes testing individual features, user interfaces, workflows, and system integrations.

Non-Functional Testing: Non-functional testing evaluates aspects of the system beyond its core functionality, such as performance, scalability, reliability, security, and compatibility. This ensures that the software meets quality attributes and user expectations.

Regression Testing: Regression testing is performed to ensure that new changes or fixes haven't introduced any unintended side effects or regressions in the system. This involves retesting previously tested functionalities and verifying that they still work as expected.

Error Handling and Recovery Testing: Error handling and recovery testing assess how the system handles errors, exceptions, and failures. This includes testing error messages, recovery mechanisms, and system resilience under adverse conditions.

User Acceptance Criteria Verification: System testing may also involve verifying that the software meets the acceptance criteria defined by stakeholders, including end users, customers, and business owners.

Documentation and Reporting: Test results, including any defects or issues found during testing, are documented and reported to relevant stakeholders. This information helps stakeholders make informed decisions about the readiness of the software for deployment.

System testing offers several benefits:

- **Comprehensive Validation**: System testing provides a comprehensive evaluation of the entire software system, ensuring that it meets all specified requirements and functions correctly in its intended environment.

- **Risk Mitigation**: By identifying defects, errors, and vulnerabilities in the system, system testing helps mitigate risks associated with software failures, security breaches, and performance issues.

- **Quality Assurance**: System testing contributes to the overall quality assurance process by verifying that the software meets quality standards, regulatory requirements, and user expectations.

- **Confidence Building**: Successful system testing builds confidence among stakeholders, including project sponsors, customers, and end users, by demonstrating that the software is reliable, stable, and ready for deployment.

In summary, system testing is a critical phase in the software testing lifecycle that evaluates the overall functionality, performance, and behavior of a software application as a complete and integrated system. By validating the software against its requirements and use cases, system testing ensures that it meets quality standards and satisfies the needs of its stakeholders.

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

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.tutorialspoint.com
- https://www.javatpoint.com
- https://www.w3school.com
- https://www.youtube.com
- https://www.google.com
- https://www.wikepedia.com
- https://www.geeksforgeeks.com
- https://www.studocu.com