**3rd Sem Mini Project Report on**

**Playlist Management**

**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name-**Sarthak Singh **University Roll No.-**2319524

***Under the Guidance of***

**Guide Name**

**Designation**

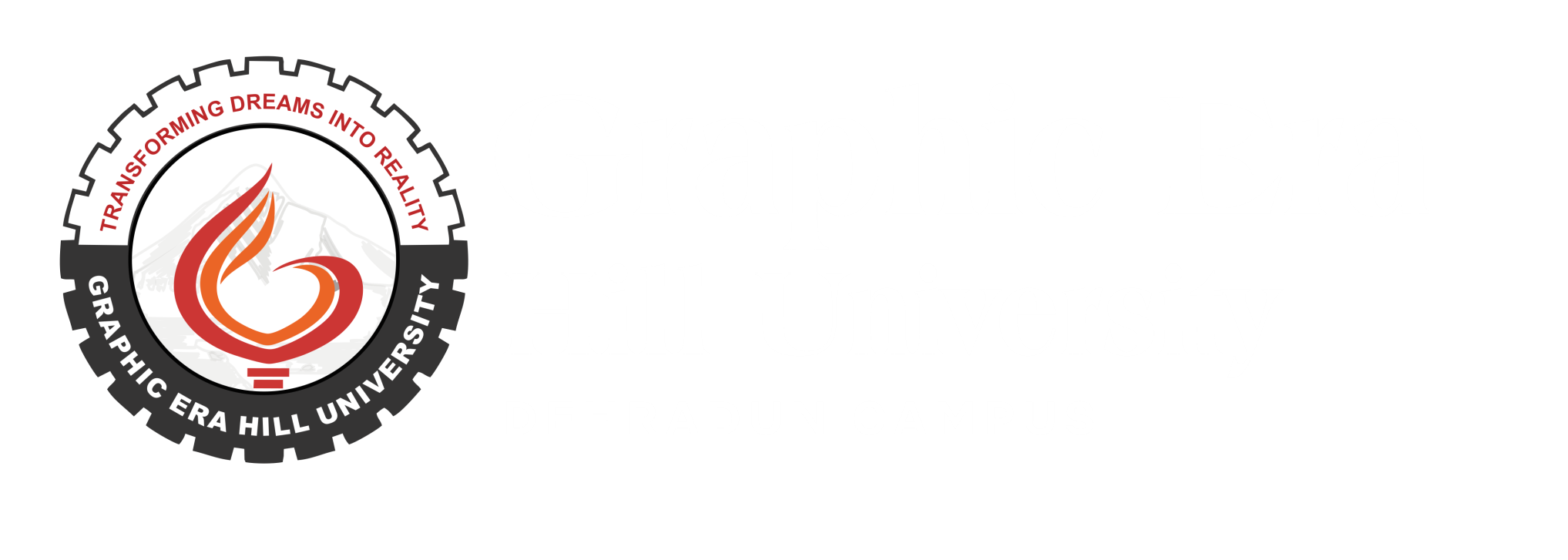


**Department of Computer Science and Engineering**

**Graphic Era Hill University**

**Dehradun, Uttarakhand**

**2024-25**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled**“Playlist Management”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering of the Graphic Era Hill University, Dehradun shall be carried out by the undersigned under the supervision of **Guide Name, Designation**, Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun.

Name-Sarthak Singh University Roll no-2319524

The above mentioned student shall be working under the supervision of the undersigned on the

**“Playlist Management”**

**Supervisor** **Head of the Department**

**Examination**

**Name of the Examiners: Signature with Date**

1. Sonal Malhotra Madam

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction and Problem Statement | **4** |
| Chapter 2 | Methodology | **5** |
| Chapter 3 | Project Work Carried Out | **6** |
| Chapter 4 | Results and Discussion | **7** |
| Chapter 5 | Conclusion and Future Work | **8** |
|  | References | **9** |
|  |  |  |

**Chapter 1**

**Introduction and Problem Statement**

Managing playlists is a common requirement for music enthusiasts and professionals. The given project implements a Playlist Management System in C++ that allows users to create, manage, and modify playlists effectively. The problem addressed here is organizing songs into playlists with capabilities to add, update, sort, save, and remove songs while ensuring data consistency through unique song IDs.

The system should utilize an object-oriented approach, leveraging classes and data structures such as map, list, and string. Each song in the playlist is represented by a Song class, which stores details like the artist's name, song title, song duration, and unique song ID. Each playlist is represented by a Playlist class, which contains a list of songs and includes operations to manage the songs. The PlaylistManager class manages multiple playlists and facilitates user interaction. The Playlist Management System provides a strong foundation for playlist management and offers a range of functionalities to handle songs efficiently.

**Chapter 2**

**Methodology**

The project follows an **object-oriented programming (OOP)** approach with the following main components:

* **Class Song:** Represents a song with attributes like ID, artist, title, and duration. It includes methods for updating details, displaying information, and converting to/from a file-friendly format.
* **Class Playlist:** Represents a collection of songs. It supports adding, removing, displaying, sorting songs by duration, and saving to a file.
* **Class PlaylistManager:** Provides an interface for managing multiple playlists. It allows the creation of playlists, addition and modification of songs, and persistence through file storage.

**Data structures used include:**

* **list:** For storing songs within playlists.
* **map:** For managing multiple playlists by their names, ensuring efficient retrieval.

**Chapter 3**

**Project Work Carried Out**

**The project implements the following features:**

1. **Object-Oriented Programming (OOP):**

* The project uses OOP principles well by encapsulating the Song and Playlist as separate classes.
* **Encapsulation:** The attributes of the song and Playlist classes are private, ensuring that they are only accessible through member functions.
* **Methods and Constructors**: The Song class has methods to update, display, and handle file operations, demonstrating class behaviour and interaction.
* **Inheritance** (Could be Added): Although the project doesn't require inheritance, you could introduce it in future iterations, such as creating a SpecialPlaylist class that extends Playlist.

1. **Playlist Management:**
   * Create new playlists with unique names.
   * Display existing playlists.
2. **Song Management:**
   * Add songs with unique IDs to playlists.
   * Remove songs from playlists.
   * Update song details.
   * Sort songs in a playlist by duration.
3. **File Handling:**
   * Save playlists to text files for persistence.
   * Load playlists from files (commented out in the provided code).
4. **Error Handling:**
   * Input validation for song IDs, durations, and playlist names.
   * Prevent duplicate playlist or song IDs.

**The program includes a menu-driven interface that provides users with a set of options to interact with the system.**

**Chapter 4**

**Results and Discussion**

The project achieves its primary objective of managing playlists efficiently. Key observations include:

**Strengths:**

* Clear separation of concerns using OOP principles.
* Robust input validation and error handling.
* Easy-to-use interface for basic playlist operations.
* Support for file persistence ensures data longevity.

**Areas for Improvement:**

* The load\_playlist method is implemented but commented out, limiting the usability of file operations.
* Adding a feature to display cumulative duration or total songs in a playlist could enhance user insights.
* A more interactive error display system could improve user experience.

Using a vector instead of list for song storage might simplify indexing and operations like finding songs.

**Chapter 5**

**Conclusion and Future Work**

The Playlist Management System provides a strong foundation for playlist management and offers a range of functionalities to handle songs efficiently.

**Future Enhancements:**

* Enable loading playlists from files for complete persistence functionality.
* Provide additional sorting options (e.g., by artist or title).
* Integrate a search feature for songs within playlists.
* Enhance the UI with graphical or web-based interfaces.
* Use a database for managing playlists and songs instead of file storage for scalability.

**References**

1. C++ Documentation: [cplusplus.com](http://www.cplusplus.com/)
2. STL Overview: cppreference.com
3. File Handling in C++: TutorialsPoint and GeeksForGeeks articles on file I/O.