System Requirements Specification Index

For

Music Playlist Management System

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Music Playlist Management System

System Requirements Specification

1 PROJECT ABSTRACT

Melodia Music Streaming requires a playlist management system to organize their expanding music collection. The system will catalog songs, track playlist details, and generate reports using Python's tuple data structures and related operations. This console application demonstrates tuple immutability principles and effective use cases while allowing users to efficiently manage music collections. The system performs critical functions including organizing songs by attributes, creating immutable playlist records, manipulating music data through tuple operations, and generating listening statistics. By implementing these operations with Python tuples, the system provides an efficient way for users to organize and analyze their music collection while ensuring data integrity.

2 BUSINESS REQUIREMENTS:

Screen Name	Console input screen
Problem Statement	1. System needs to store and categorize different song attributes (artist, title, genre, duration, release year, album) 2. System must support filtering songs by genre, artist, duration range, or release decade 3. Console should handle different tuple operations like Creating immutable song records, Tuple packing and unpacking, Tuple as dictionary keys, Named tuples for improved readability, Tuple comparison and sorting

3 CONSTRAINTS

3.1 INPUT REQUIREMENTS

1. Song Records):
    - Must be stored as tuples with fields for id, title, artist, genre, duration, release_year, album
    - Must be stored in list variable `songs`
    - Example: `("S001", "Bohemian Rhapsody", "Queen", "rock", 354, 1975, "A Night at the Opera")`
2. Song Genres:
    - Must be one of: "rock", "pop", "jazz", "classical", "electronic", "hip-hop"
    - Must be stored as string in song's genre field (index 3)
    - Example: "rock"
3. Duration:
    - Must be stored as integer in seconds at index 4
    - Example: 354 (for 5:54)
4. Release Year:
    - Must be stored as integer at index 5
    - Example: 1975
5. Predefined Songs:
    - Must use these exact predefined songs in the initial song list:
    - `("S001", "Bohemian Rhapsody", "Queen", "rock", 354, 1975, "A Night at the Opera")`
    - `("S002", "Billie Jean", "Michael Jackson", "pop", 294, 1982, "Thriller")`
    - `("S003", "Take Five", "Dave Brubeck", "jazz", 324, 1959, "Time Out")`
    - `("S004", "Moonlight Sonata", "Ludwig van Beethoven", "classical", 363, 1801, "Piano Sonatas")`
    - `("S005", "Strobe", "Deadmau5", "electronic", 601, 2009, "For Lack of a Better Name")`
6. New Releases:
    - Must use these exact predefined items in the new releases list:
    - `("N001", "Blinding Lights", "The Weeknd", "pop", 200, 2020, "After Hours")`
    - `("N002", "Bamboo", "J Balvin", "hip-hop", 190, 2023, "Colores")`

3.2 OPERATIONS CONSTRAINTS

1. Song Record Creation:
    - Must use tuple constructor or literal syntax
    - Example: `song = ("S001", "Bohemian Rhapsody", "Queen", "rock", 354, 1975, "A Night at the Opera")`
    - Must use variable name `song` for individual songs
2. Song Filtering by Genre:
    - Must use tuple unpacking in list comprehension or filter
    - Example: `[song for song in songs if song[3] == "rock"]`
    - Must use variable name `filtered_songs` for the result
3. Duration Formatting:
    - Must use tuple indexing to access duration
    - Example: `f"{song[4] // 60}:{song[4] % 60:02d}"`
    - Must use variable name `formatted_duration` for the result
4. Tuple Packing and unpacking:
    - Must demonstrate both packing and unpacking operations
    - Example: `song_id, title, artist, *rest = song`
    - Must use variable names `song_id`, `title`, `artist`, and `rest`
5. Creating named Tuples:
    - Must use `collections.namedtuple` to create a Song type
    - Example: `Song = namedtuple('Song', ['id', 'title', 'artist', 'genre', 'duration', 'release_year', 'album'])`
    - Must use variable name `Song` for the named tuple type
    - Must use variable name `named_songs` for a list of named tuple instances
6. Tuple as Dictionary Keys:
    - Must use tuples as dictionary keys for counting or grouping
    - Example: `plays = {(song[2], song[1]): 0 for song in songs}`
    - Must use variable name `plays_dict` for the result
7. Multiple Playlist Creation:

- Must create immutable playlist records that include name, created_date, and song_ids
- Example: `playlist = ("Rock Classics", "2023-03-04", tuple(song[0] for song in rock_songs))`
- Must use variable name `playlist` for individual playlists
- Must use variable name `playlists` for a list of playlists

8. Song Sorting:

- Must use tuple comparison for sorting songs
- Example: `sorted(songs, key=lambda x: (x[2], x[5]))` (by artist then year)
- Must use variable name `sorted_songs` for the result

9. Decade Filtering:

- Must use tuple data for filtering by decade
- Example: `[song for song in songs if song[5] // 10 == 197]` (for 1970s)
- Must use variable name `decade_songs` for the result

10. Genre Distribution Calculation:

- Must use tuples in dict creation to count genre distribution
- Example: `{genre: len([s for s in songs if s[3] == genre]) for genre in genres}`
- Must use variable name `genre_counts` for the result

3.3 OUTPUT CONSTRAINTS

1. Display Format:

- Show song ID, title, artist, genre, duration, release year, album
- Format duration as MM:SS
- Each song must be displayed on a new line

2. Output:

- Show "===== MUSIC PLAYLIST MANAGEMENT SYSTEM ====="
- Show "Total Songs: {count}"
- Show "Total Duration: {hours}h {minutes}m {seconds}s"
- Show "Current Song Collection:"
- Show songs with format: "{id} | {title} | {artist} | {genre} | {duration} | {year} | {album}"
- Show "Playlist Information:" when displaying playlist data
- Show "Song Distribution:" when showing genre counts

4. TEMPLATE CODE STRUCTURE:

1. Data Management Functions:

- `initialize_data()` - creates the initial song and new releases lists

2. Tuple Operation Functions:

- `create_song_record(id, title, artist, genre, duration, release_year, album)` creates immutable song record
- `filter_by_genre(songs, genre)` - filters songs by genre using tuple data
- `filter_by_artist(songs, artist)` - filters songs by artist
- `filter_by_duration(songs, min_duration, max_duration)` - filters songs by duration range
- `filter_by_decade(songs, decade)` - filters songs by release decade
- `format_duration(seconds)` - formats duration from seconds to MM:SS
- `create_named_tuple_songs(songs)` - converts regular tuples to named tuples
- `create_playlist(name, song_ids, songs)` - creates immutable playlist record
- `sort_songs(songs, sort_key)` - sorts songs by specified attribute
- `calculate_genre_distribution(songs)` - calculates song count by genre
- `integrate_new_releases(songs, new_releases)` - combines song lists

3. Display Functions:

- `get_formatted_song(song)` - formats a song for display
- `display_data(data, data_type)` - displays songs or other data types

4. Program Control Functions:

- `main()` - main program function

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. View the main menu
3. Select operations:
   - Option 1: View Songs
   - Option 2: Filter Songs
   - Option 3: Create Playlist
   - Option 4: Convert to Named Tuples
   - Option 5: Calculate Statistics
   - Option 6: Integrate New Releases
   - Option 0: Exit
4. Perform operations on the song list
5. View results after each operation
6. Exit program when finished