System Requirements Specification Index

For

Digital Music Mixer Console Application

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

TABLE OF CONTENTS

Digital Music Mixer Console
System Requirements Specification

1 Project Abstract

SoundByte Labs requires a digital music mixing tool to generate and manipulate sound patterns for their audio production workflow. The application will create varied amplitude patterns for different sound types (bass, melody, percussion, ambient), apply transformations to these patterns, and provide visual representation of the results. This console application will help sound engineers prototype rhythmic and melodic structures through mathematical pattern manipulation before implementing them in full audio production.

2 Business Requirements:

Screen Name	Console input screen
Problem Statement	1. Application must create and manipulate numerical lists representing patterns 2. System must demonstrate basic list operations (slicing, concatenation, reversal, etc.) 3. Program must apply various transformations to lists (reverse, slice, extend, shuffle) 4. Application must visualize lists in a readable format

3 Constraints

3.1 Input Requirements

1. Patterns represented as lists of integers (0-100):
    - Each integer represents an amplitude value
    - Values must be in range 0-100

2. Four pattern types available:
    - Bass patterns: Strong beats with lower values
    - Melody patterns: Varied amplitudes
    - Percussion patterns: Sharp peaks
    - Ambient patterns: Smooth, gradually changing values
3. User input required for:
    - Pattern type selection
    - Pattern length specification
    - Transformation selection
    - Transformation parameters

3.2 Operations Constraints

1. List Creation Operations:
    - Create patterns using list comprehensions
    - Generate random values within specified ranges
    - Build lists with specific characteristics for each pattern type

2. List Manipulation Operations:
    - Apply list slicing using [start:end:step] notation
    - Perform list reversal using [::-1] syntax
    - Implement list repetition using multiplication * n
    - Execute list concatenation using + operator
    - Implement segment shuffling using list subdivision and reassembly

3.3 Output Constraints

1. Application Interface:
    - Show "List Operations Lab" as application title
    - Display clear menu options for pattern creation and transformation
    - Present transformation options with corresponding list operation syntax

2. List Visualization:
    - Represent lists using ASCII art visualization
    - Show list indices for reference
    - Display list length and type information
    - Indicate list transformation applied

4. TEMPLATE CODE STRUCTURE:

1. Pattern Creation Functions:

- create_bass_pattern(length: int) -> list
- create_melody_pattern(length: int) -> list
- create_percussion_pattern(length: int) -> list
- create_ambient_pattern(length: int) -> list

2. List Operation Functions:

- slice_list(pattern: list, start: int, end: int, step: int) -> list
- reverse_list(pattern: list) -> list
- extend_list(pattern: list, repeats: int) -> list
- combine_lists(pattern1: list, pattern2: list) -> list
- shuffle_segments(pattern: list, segment_size: int) -> list

3. Visualisation Functions:

- visualize_list(pattern: list) -> str

4. Program Control Functions:

- display_menu() -> None
- display_transformation_menu() -> None
- main()

5. EXECUTION STEPS TO FOLLOW:

1. Run the program
2. Select list creation method:
   - Option 1: Create Bass Pattern
   - Option 2: Create Melody Pattern
   - Option 3: Create Percussion Pattern
   - Option 4: Create Ambient Pattern
3. Enter pattern length (8-64)
4. View created list visualization
5. Apply transformations (Option 5):
   - Option 1: Reverse a list
   - Option 2: Slice a list
   - Option 3: Extend a list
   - Option 4: Combine lists
   - Option 5: Shuffle segments
6. View transformed list
7. Repeat or exit program (Option 0)