PlayWise Project Structure Guide

Quick Start for New Developers

Welcome to PlayWise! This guide will help you understand the project structure and get started quickly.

Folder Structure Overview

```
PlayWise/
├── include/          # Header files (.h) - Class definitions
├── src/              # Source files (.cpp) - Implementation
├── tests/            # Test files and framework
├── docs/             # Documentation
└── README.md         # Detailed project documentation
```

What Each Folder Contains

`include/` - Header Files

All class definitions and interfaces are here. These files tell you **what** each component does:

- `song.h` - Song class with metadata (title, artist, duration, rating)
- `playlist.h` - Doubly linked list for managing song playlists
- `history.h` - Stack-based system for tracking played songs
- `rating_tree.h` - Binary search tree for rating-based organization
- `song_database.h` - HashMap for fast song lookup
- `sorting.h` - Various sorting algorithms (merge, quick, heap)
- `dashboard.h` - System monitoring and analytics
- `playwise_app.h` - Main application interface

`src/` - Source Files

The actual implementation of all classes. These files show you **how** everything works:

- `song.cpp` - Song class implementation
- `playlist.cpp` - Playlist management with doubly linked list

- `history.cpp` - History tracking with stack operations
- `rating_tree.cpp` - Binary search tree implementation
- `song_database.cpp` - HashMap database implementation
- `sorting.cpp` - Sorting algorithms implementation
- `dashboard.cpp` - Analytics and monitoring
- `playwise_app.cpp` - Main application logic
- `main.cpp` - Program entry point

`tests/` - Testing Framework

Comprehensive test suite for all components:

- `test_framework.h` - Custom testing framework
- `test_*.cpp` - Individual component tests
- `test_integration.cpp` - System integration tests
- `test_runner.cpp` - Test execution engine
- `README_TESTS.md` - Testing documentation

`docs/` - Documentation

Detailed documentation for users and developers:

- `user_guide.md` - Complete user manual
- `playwise.md` - Technical documentation
- `playwise.pdf` - PDF version of documentation

Key Files to Start With

For Understanding the Project:

1. `README.md` - Complete project overview
2. `include/playwise_app.h` - Main application interface
3. `src/main.cpp` - How the program starts

For Learning Data Structures:

1. `include/song.h` - Basic data structure
2. `include/playlist.h` - Doubly linked list
3. `include/rating_tree.h` - Binary search tree
4. `include/song_database.h` - HashMap

For Understanding Algorithms:

1. `include/sorting.h` - Sorting algorithms
2. `src/sorting.cpp` - Algorithm implementations

How to Build and Run

Quick Build (Windows):

```
g++ -std=c++17 -Wall -Wextra -Iinclude -o PlayWise.exe src/*.cpp
```

Quick Build (Linux/macOS):

```
g++ -std=c++17 -Wall -Wextra -Iinclude -o PlayWise src/*.cpp
```

Run the Application:

```
./PlayWise.exe  # Windows
./PlayWise       # Linux/macOS
```

Run Tests:

```
cd tests
g++ -std=c++17 -Wall -Wextra -I../include -o test_runner *.cpp ../src/*.cpp
./test_runner
```

Data Structures Implemented

Structure	Location	Purpose
Song Class	`include/song.h`	Basic song data with metadata
Doubly Linked List	`include/playlist.h`	Playlist management
Stack	`include/history.h`	Playback history tracking
Binary Search Tree	`include/rating_tree.h`	Rating-based organization

Structure	Location	Purpose
HashMap	`include/song_database.h`	Fast song lookup
Vector	Used throughout	Dynamic arrays

Algorithms Implemented

Algorithm	Location	Time Complexity
Merge Sort	`src/sorting.cpp`	O(n log n)
Quick Sort	`src/sorting.cpp`	O(n log n) average
Heap Sort	`src/sorting.cpp`	O(n log n)
Binary Search	`src/rating_tree.cpp`	O(log n)
Linear Search	Various files	O(n)

Main Features

1. **Playlist Management** - Add, remove, reorder songs
2. **History Tracking** - Undo/redo playback history
3. **Rating System** - Organize songs by ratings
4. **Fast Search** - O(1) song lookup
5. **Multiple Sorting** - Sort by title, duration, rating, artist
6. **Analytics Dashboard** - System monitoring and statistics
7. **Data Export** - Save/load functionality

Important Notes

- **C++17 Required** - Uses modern C++ features
- **No External Dependencies** - Pure C++ implementation
- **Console Application** - Text-based interface
- **Comprehensive Testing** - 56 test cases included
- **Memory Safe** - Proper destructors and memory management

Next Steps

1. **Read the README.md** for detailed project information
2. **Check docs/user_guide.md** for usage instructions

3. **Run the application** to see it in action
4. **Explore the source code** starting with `main.cpp`
5. **Run the tests** to verify everything works

Need Help?

- Check the **README.md** for detailed documentation
- Look at **docs/user_guide.md** for usage examples
- Run the **tests** to understand expected behavior
- Examine the **source code** for implementation details

Happy Coding!