DJ Application

Final Report

29-April 2025

ISSUED BY

The Big 3

REPRESENTATIVE

Umer Farooq	22I-0891
Muhammad Usman	22I-0900
Muhammad Irtaza Khan	22I-0911

Problem Statement

The problem of	The lack of an efficient and feature-rich DJ mixing platform for both professional and enthusiasts
affects	DJ Artists
the impact of which is	Limited creative control, difficulty in synchronizing tracks and inefficiencies in live mixing, leading to suboptimal performances.
a successful solution would be	A comprehensive DJ software that offers core functionalities like track loading, playing, crossfading and looping along with advanced features such as tempo/BGM adjustment, pitch control, real-time effects, sample triggering, mashup creation, auto-mixing and live recording. The software should also provide intuitive library and playlist management, microphone input, and manual beat matching, ensuring a seamless and professional DJ experience.

Roles and Responsibilities

Name	**Umer Farooq**	**Muhammad Usman**	**Muhammad Irtaza Khan**
Scrum Role	Product Owner	Scrum Master	Scrum Team
Duties	Define requirements, prioritize backlog, approve features. Test app features, report bugs, and verify acceptance criteria.	Facilitate Scrum events, remove blockers, ensure smooth process. Design an intuitive DJ interface and user experience.	Implement core DJ functionalities (audio playback, BPM analysis, mixing). Build UI for track controls, playlist management, and effects.

Team Agreement

1. **Methods of Communication:**
    - Primary: Trello (for real-time discussions and quick updates)
    - Secondary: Email (for formal communications and documentation)
    - Phone/Discord: For urgent matters
2. **Communication Response Times:**
    - Trello: Within 2 hours during working hours
    - Email: Within 24 hours
    - Phone/Discord: Immediate response for urgent issues
3. **Meeting Attendance:**
    - Daily Stand-ups: Mandatory (15 minutes, everyday in university)
    - Sprint Planning: Mandatory (before each sprint)

    - Sprint Review & Retrospective: Mandatory (end of each sprint)
4. **Running Meetings:**
    - Meetings will be held online via Discord.
    - Scrum Master will facilitate meetings.
    - A rotating team member will take minutes and document key decisions.
5. **Meeting Preparation:**
    - All members must review the agenda beforehand.
    - Developers must update progress on their assigned tasks.
    - The Product Owner must prepare backlog updates.
6. **Version Control:**
    - Repository: GitHub
    - Branching Strategy: Feature branches merged into the main branch after code review.
    - What to Commit: Source code, configuration files, documentation
    - What Not to Commit: Sensitive credentials, personal files, large binary files
7. **Division of Work:**
    - Work will be divided based on expertise and workload balance.
    - Scrum Master and Scrum Team will collaboratively assign tasks.
    - Stakeholders include the Product Owner, Customers, and Senior Management.
8. **Submitting Assignments:**
    - All tasks must be completed before the sprint deadline.
    - Peer review is required before submission.
    - The Scrum Master will review and finalize submissions.
9. **Contingency Planning:**
    - If a team member drops out: Tasks will be redistributed among remaining members.
    - If a member misses meetings consistently: Scrum Master will discuss attendance issues and escalate if necessary.
    - If a member is academically dishonest: The issue will be reported, and corrective action will be taken as per institutional policies.

Introduction

Purpose

Flux DJ is a desktop-based DJ application that allows users to mix and manipulate audio tracks in real time. The system provides an intuitive interface for DJs to create seamless transitions, apply effects, and manage playlists efficiently.

Document Conventions

- **Bold** is used for section headings.
- Code formatting is used for technical terms.
- Bulleted lists are used for enumerating items.

Intended Audience and Reading Suggestions

This document is intended for developers, project managers, testers, and stakeholders involved in the development of Flux DJ. It provides a detailed overview of the system's functional and non-functional requirements.

Product Scope

Flux DJ aims to provide a powerful tool for both amateur and professional DJs to mix and manipulate music tracks with ease. The application will have the following features:

- Track Loading and Mixing
- Audio Effects and Filters
- Real-time BPM Detection and Syncing
- Waveform Visualization
- Playlist Management

Overall Description

Product Perspective

Flux DJ is a standalone desktop application designed to work with external audio interfaces and MIDI controllers. It will support Windows, macOS, and Linux operating systems.

Product Functions

- Load and play multiple audio tracks simultaneously
- Apply real-time effects and filters
- Adjust tempo and pitch with BPM sync
- Visualize audio waveforms

User Classes and Characteristics

- **Beginner DJs:** Users who are learning the basics of DJing.
- **Professional DJs:** Experienced users who require advanced mixing capabilities.
- **Music Enthusiasts:** Users who enjoy experimenting with audio manipulation.

Operating Environment

The system will be a desktop application running on Windows 10+, macOS 11+, and Linux distributions. It requires a minimum of 4GB RAM, a dual-core processor, and an audio output device.

Design and Implementation Constraints

- The application should support **low-latency audio processing**.
- **Real-time BPM detection and synchronization** should be accurate.

User Documentation

The system will include the following documentation:

- **User Guide:** Instructions on how to use Flux DJ.
- **API Documentation:** For developers integrating with external hardware.
- **FAQ Section:** Common troubleshooting steps.

Assumptions and Dependencies

- Users must have an audio output device for playback.
- The application requires external audio libraries such as **PortAudio or JUCE**.
- MIDI controllers must be **class-compliant** for plug-and-play functionality.

Specific Requirements

External Interface Requirements

User Interfaces

- Main interface with waveform display
- Track library and playlist manager
- Effect control panel
- BPM sync and pitch control interface

Hardware Interfaces

- USB connectivity for MIDI controllers
- Support for external sound cards and DJ controllers

Software Interfaces

- Integration with **ASIO and Core Audio** for low-latency playback
- Support for **MP3, WAV, FLAC, and AAC audio formats**
- MIDI protocol support for **external controller mapping**

Communication Interfaces

- USB and Bluetooth support for MIDI controllers
- REST API for third-party extensions (if applicable)

System and User Requirements

Load Track

1. Functional Requirements

System Requirements

1. The system shall allow users to browse files from both **local storage** and **cloud storage**.
2. The system shall support **common audio file formats** (e.g., MP3, WAV, FLAC, AAC).
3. The system shall provide a **file selection interface** for users to choose tracks.
4. The system shall **validate the file format** before loading the track.
5. The system shall load the selected track into a **designated deck for playback**.
6. The system shall display a **success message** when a track is loaded successfully.
7. The system shall provide an **error message** if the file is unsupported or fails to load.
8. The system shall allow users to **cancel the loading process** before completion.
9. The system shall ensure that **multiple tracks can be loaded into different decks**.
10. The system shall store recently loaded tracks in a **history or cache for quick access**.

User Requirements

1. The user shall be able to **select tracks from local storage or cloud services** (e.g., Google Drive, Dropbox).
2. The user shall be able to **see a list of available audio files** when browsing.
3. The user shall receive **real-time feedback** (loading indicator, success/error messages).
4. The user shall be able to **filter tracks based on file type, size, or metadata**.
5. The user shall be able to **drag and drop tracks** into the deck for faster loading.
6. The user shall be able to **load tracks with minimal delay** (optimized performance).
7. The user shall be able to **see track metadata** (title, artist, duration) after loading.
8. The user shall be able to **replace an already loaded track** with a new one.

2. Non-Functional Requirements

Performance Requirements

1. The track loading process shall take **no more than 2 seconds** under normal conditions.
2. The system shall support **simultaneous track loading in multiple decks** without lag.
3. The system shall use **efficient memory management** to avoid excessive resource consumption.

Usability Requirements

1. The UI shall be **intuitive and visually clear**, allowing users to load tracks without confusion.
2. Error messages shall be **user-friendly and instructive**, guiding users on how to resolve issues.
3. The system shall provide **keyboard shortcuts** for quick track loading (e.g., Ctrl + L).

Security Requirements

1. The system shall **prevent unauthorized access** to private cloud storage.
2. The system shall not modify or delete user files **without explicit permission**.
3. The system shall handle **corrupted or unsupported files gracefully**, avoiding crashes.

Scalability Requirements

1. The system shall support **large audio file sizes** without performance degradation.
2. The system shall be able to **integrate additional cloud storage providers** in future updates.

3. Constraints

1. The system shall not modify track audio during loading.
2. The system shall not allow **loading of DRM-protected files** unless permitted.
3. The feature shall be compatible with **React.js frontend and Node.js backend**.
4. The system shall maintain **consistent performance across different devices** (desktop, tablet).

Play Track

1. Functional Requirements

System Requirements

1. The system shall provide a **play button** to start the track.
2. The system shall ensure that the **play button remains disabled if no track is loaded**.
3. The system shall **resume playback from the current position** when the play button is pressed.
4. The system shall allow **playback control via keyboard shortcuts** (e.g., Spacebar to play/pause).
5. The system shall **visually indicate playback status** (e.g., play/pause icon changes dynamically).
6. The system shall provide **real-time waveform visualization** while the track is playing.
7. The system shall allow **audio output routing** to different sound devices (e.g., headphones, external speakers).
8. The system shall provide **low-latency playback** to ensure responsiveness.
9. The system shall allow users to **configure audio buffer settings** for optimized performance.
10. The system shall support **track playback in multiple decks independently**.

User Requirements

1. The user shall be able to **start a track by pressing the play button**.
2. The user shall be able to **see the current playback position** (e.g., progress bar or timestamp).
3. The user shall be able to **use keyboard shortcuts** to control playback.
4. The user shall be able to **see a visual indicator** when the track is playing.
5. The user shall be able to **route audio output** to different sound devices.
6. The user shall experience **immediate playback with minimal delay** after pressing play.
7. The user shall be able to **enable or disable waveform visualization** during playback.
8. The user shall be able to **toggle between different playback modes** (e.g., loop, single play).

2. Non-Functional Requirements

Performance Requirements

1. The track shall start playing within **100 milliseconds** after pressing the play button.
2. The system shall support **real-time playback with minimal latency**.
3. The system shall ensure that **multiple tracks in different decks** play without lag.

Usability Requirements

1. The play button shall have **a clear and intuitive design** to indicate its function.
2. The system shall provide **visual feedback** (e.g., animation, color change) when playback starts.
3. The system shall ensure that **all playback controls are accessible via both mouse and keyboard**.
4. The system shall provide **error messages** if playback fails due to unsupported formats or missing files.

Security Requirements

1. The system shall **prevent unauthorized file modifications** during playback.
2. The system shall ensure that **maliciously modified or corrupted files do not crash playback**.
3. The system shall **sandbox the playback process** to prevent unauthorized system access.

Scalability Requirements

1. The system shall support **multiple playback decks running simultaneously**.
2. The system shall be able to **integrate additional audio processing modules** for future enhancements.

3. The system shall **scale with higher-quality audio files** without performance degradation.

3. Constraints

1. The system shall **not modify the original track audio** during playback.
2. The system shall **not allow playback of DRM-protected files** unless explicitly permitted.
3. The system shall be compatible with **React.js frontend and Node.js backend**.
4. The system shall maintain **consistent performance across different devices** (desktop, tablet).

Pause Track

1. Functional Requirements

System Requirements

1. The system shall provide a **pause button** to halt track playback.
2. The system shall **pause playback instantly** when the pause button is pressed.
3. The system shall **retain the current playback position** when paused.
4. The system shall **resume playback from the exact paused position** when play is pressed.
5. The system shall **visually indicate the paused state** (e.g., pause icon, waveform freeze).
6. The system shall allow **pausing playback via keyboard shortcuts** (e.g., Spacebar).
7. The system shall ensure that **pause functionality works across multiple decks independently**.
8. The system shall provide **low latency pause/resume operations** for a seamless DJ experience.

User Requirements

1. The user shall be able to **pause playback at any time** using the pause button.
2. The user shall be able to **resume playback from the paused position** when play is pressed.
3. The user shall receive **visual feedback when a track is paused** (button change, waveform freeze).
4. The user shall be able to **pause playback using a keyboard shortcut**.
5. The user shall experience **no noticeable delay** when pausing or resuming playback.

2. Non-Functional Requirements

Performance Requirements

1. The pause action shall **take effect within 50 milliseconds** of user input.
2. The system shall handle **simultaneous pausing across multiple decks** without lag.

Usability Requirements

1. The pause button shall be **intuitive and visually distinct** from play and stop buttons.
2. The system shall provide **clear visual indicators** when a track is paused.

Security Requirements

1. The system shall **not allow external interference** in pausing/resuming playback.

3. Constraints

1. The system shall **not alter the track's audio or metadata** when paused.
2. The system shall be compatible with **React.js frontend and Node.js backend**.

Stop Track

1. Functional Requirements

System Requirements

1. The system shall provide a **stop button** to halt track playback completely.
2. The system shall **reset the track position to the beginning** when stopped.
3. The system shall ensure that **pressing play after stop restarts the track from the beginning**.
4. The system shall **disable the stop button when no track is loaded**.
5. The system shall **visually indicate that the track has stopped**.
6. The system shall allow users to **stop playback using a keyboard shortcut**.
7. The system shall ensure that **stop functionality works across multiple decks independently**.

User Requirements

1. The user shall be able to **stop a track at any time** using the stop button.
2. The user shall be able to **restart the track from the beginning** after stopping.
3. The user shall receive **visual feedback when a track is stopped**.
4. The user shall be able to **stop playback using a keyboard shortcut**.

2. Non-Functional Requirements

Performance Requirements

1. The stop action shall **take effect within 50 milliseconds** of user input.
2. The system shall allow **multiple tracks to be stopped independently without delay**.

Usability Requirements

1. The stop button shall be **clearly distinct** from the play and pause buttons.
2. The system shall provide **clear visual feedback** when a track is stopped.

Security Requirements

1. The system shall **prevent unauthorized scripts from forcing a track stop**.

3. Constraints

1. The system shall **not modify the audio file** when stopping playback.
2. The system shall be compatible with **React.js frontend and Node.js backend**.

Crossfade Between Tracks

1. Functional Requirements

System Requirements

1. The system shall provide a **crossfade slider** to transition between two tracks.
2. The system shall **blend audio smoothly from one track to another** based on the slider position.
3. The system shall allow users to **configure the crossfade transition time**.
4. The system shall ensure that **crossfading does not introduce noticeable gaps or artifacts**.
5. The system shall **visually represent the crossfade effect** on waveforms or volume levels.
6. The system shall provide **predefined crossfade presets** (e.g., fast, slow, manual).
7. The system shall allow users to **disable crossfade for instant transitions**.

User Requirements

1. The user shall be able to **adjust a crossfade slider** to control the transition between two tracks.
2. The user shall hear a **smooth transition** without abrupt audio cuts.
3. The user shall be able to **set a custom crossfade duration** for their mixing preference.
4. The user shall be able to **see a visual representation of the crossfade**.
5. The user shall be able to **turn off crossfade for instant switching**.

2. Non-Functional Requirements

Performance Requirements

1. The crossfade effect shall **apply in real time** with no more than **20ms processing delay**.
2. The system shall handle **simultaneous crossfades across multiple decks** efficiently.

Usability Requirements

1. The crossfade slider shall be **easy to access and adjust** during live performance.
2. The system shall provide **intuitive visual feedback** when crossfade is active.

Security Requirements

1. The system shall **prevent crossfade glitches caused by corrupted files**.

Constraints

1. The system shall **not modify track audio permanently** during crossfade.
2. The system shall be compatible with **React.js frontend and Node.js backend**.

Performance Requirements

- The application should process **audio with less than 10ms latency**.
- It should support playback of **at least two simultaneous tracks**.
- The BPM sync feature should have **accuracy of ±0.1 BPM**.

Design Constraints

- The system should be developed using **React for frontend, Node.js with Express.js for backend, and MongoDB for database**.
- It should be optimized for **real-time audio processing**.

Software System Attributes

Security

- Encrypted storage for user preferences and playlists.
- Secure API calls for any cloud-based integrations.

Maintainability

- Modular architecture for easy updates and expansions.

Usability

- Intuitive UI/UX optimized for live performance scenarios.

Availability

- The system should work **offline without internet dependency**.

Other Requirements

- Compliance with **audio licensing and copyright regulations**.
- Automatic backup of **user settings and playlists**.

Feature List

1. Core DJ Features

These essential features allow users to load, play, and manipulate tracks for live DJing.

- **Load Track** – Import tracks from the device library or cloud storage.
- **Play Track** – Start playback of a selected track.
- **Pause Track** – Temporarily stop playback without resetting the position.
- **Stop Track** – Completely stop the track and reset its position.
- **Adjust Volume** – Modify the track's volume level.
- **Adjust Tempo/BPM** – Speed up or slow down the track's BPM without changing pitch.
- **Cue Point Setting** – Set a specific point in the track for instant playback.
- **Crossfade Between Cues** – Smoothly transition from one cue to another.

2. Effects and Mixing

These features provide tools for enhancing and customizing audio output.

- **Apply Sound Effects** – Add effects such as echo, reverb, flanger, distortion, etc.
- **Equalizer Adjustment** – Fine-tune bass, mid, and treble frequencies.
- **Filter Effects** – Apply high-pass, low-pass, or band-pass filters.
- **Beat Sync** – Automatically synchronize BPM between two tracks.
- **Record Live Mix** – Capture the live DJ session for playback.

3. Library and Playlist Management

These features help DJs organize and manage their music collections efficiently.

- **Search Tracks** – Find tracks using keywords.
- **Save and Export Track**– Export the recorded mixed track in various formats (MP3, WAV, etc.).

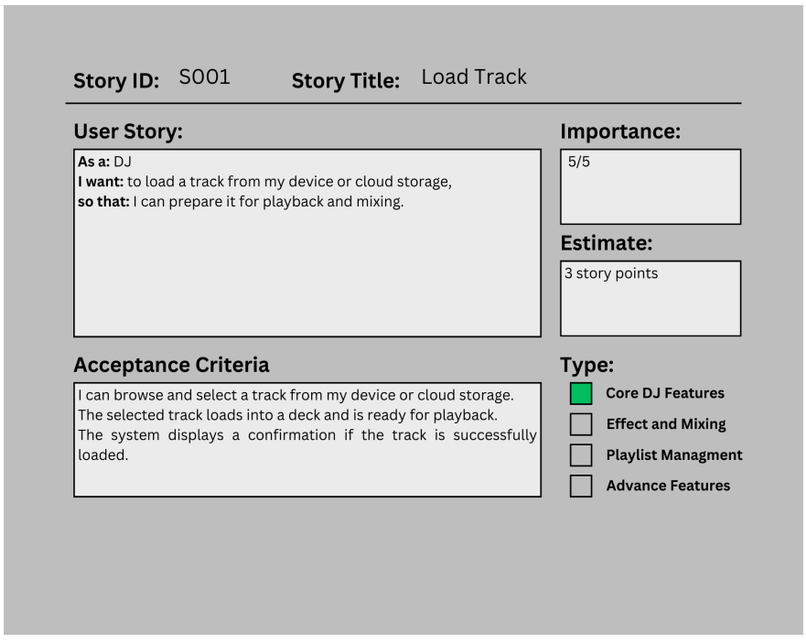
4. Advanced DJ Features

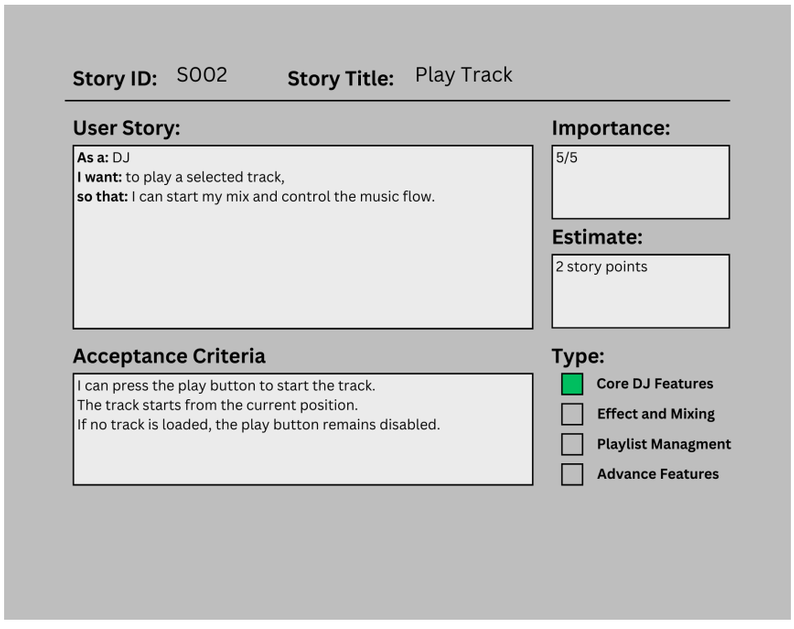
Additional functionalities that improve performance and automation.

- **Auto-Mix** – Enable automatic transitions between tracks.
- **Analyze BPM and Key** – Display BPM and musical key for tracks.
- **Preview Track** – Listen to a short preview of a track before playing.
- **Manual Beat Matching** – Manually sync beats by adjusting tempo.

- **Microphone Input** – Integrate live voice input during a set.

User Stories





Story ID: S003 **Story Title:** Pause Track

User Story:

As a: DJ
I want: to pause a playing track,
so that: I can stop the music temporarily and resume later.

Importance:

5/5

Estimate:

2 story points

Acceptance Criteria

I can press the pause button to halt playback.
When I press play again, the track resumes from the paused position.

Type:

- [x] Core DJ Features
- [ ] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S004 **Story Title:** Stop Track

User Story:

As a: DJ
I want: to stop a track completely,
so that: I can reset playback or switch to another song.

Importance:

4/5

Estimate:

2 story points

Acceptance Criteria

I can press the stop button to halt playback.
The track resets to the beginning.
Pressing play after stopping restarts the track from the beginning.

Type:

- [x] Core DJ Features
- [ ] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S006 **Story Title:** Adjust Volume

User Story:

As a: DJ
I want: to control the volume of a track,
so that: I can balance audio levels while mixing.

Importance:

5/5

Estimate:

2 story points

Acceptance Criteria

I can increase or decrease the track volume.
Volume changes apply in real-time.
A visual indicator shows the current volume level.

Type:

- [x] Core DJ Features
- [ ] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S007 **Story Title:** Adjust Tempo/BPM

User Story:

As a: DJ
I want: to adjust the tempo of a track,
so that: I can match the beats of different songs without changing pitch.

Importance:

5/5

Estimate:

5 story points

Acceptance Criteria

I can increase or decrease the track's BPM.
The track speeds up or slows down accordingly.
The pitch remains unchanged when adjusting tempo.

Type:

- [x] **Core DJ Features**
- [ ] **Effect and Mixing**
- [ ] **Playlist Managment**
- [ ] **Advance Features**

Story ID: S010 **Story Title:** Cue PointTrack

User Story:

As a: DJ
I want: to mark and jump to specific points in a track,
so that: I can quickly access key moments during a performance.

Importance:

5/5

Estimate:

4 story points

Acceptance Criteria

I can set multiple cue points in a track.
I can start playback from any saved cue point.
Cue points persist across sessions.

Type:

- [x] **Core DJ Features**
- [ ] **Effect and Mixing**
- [ ] **Playlist Managment**
- [ ] **Advance Features**

Story ID: S011 **Story Title:** Apply Sound Effects

User Story:

As a: DJ
I want: to apply sound effects like echo, reverb, and flanger to my tracks
so that: I can enhance my mix and create a unique sound.

Importance:

4/5

Estimate:

3

Acceptance Criteria

I can select different sound effects from a list.
I can adjust effect intensity in real time.
The effect is applied without audio lag.

Type:

- [ ] **Core DJ Features**
- [x] **Effect and Mixing**
- [ ] **Playlist Managment**
- [ ] **Advance Features**

Story ID: S012 **Story Title:** Equalizer Adjustment

User Story:

As a: DJ
I want: to adjust the bass, mid, and treble frequencies of my track
so that: I can fine-tune the sound output to match the environment.

Importance: 4/5

Estimate: 2

Acceptance Criteria

User can adjust bass, mid, and treble separately.
Changes reflect in real-time.
Reset option is available to revert to default settings.

Type:

- [ ] Core DJ Features
- [x] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S013 **Story Title:** Apply Filter Effects

User Story:

As a: DJ
I want: to apply high-pass, low-pass, and band-pass filters to my tracks
so that: I can control the frequency range and create smoother transitions.

Importance: 3/5

Estimate: 3

Acceptance Criteria

User can apply filters individually or in combination.
Filters should update in real-time.
Sliders are available for fine-tuning filter frequency and resonance..

Type:

- [ ] Core DJ Features
- [x] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S016 **Story Title:** Automatic Beat Synchronization

User Story:

As a: DJ
I want: to synchronize the BPM of two tracks automatically
so that: my transitions sound seamless and professional.

Importance: 5/5

Estimate: 4

Acceptance Criteria

System detects BPM of both tracks.
Option to sync one track to another.
Tracks remain in sync when playing simultaneously.

Type:

- [ ] Core DJ Features
- [x] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S019 **Story Title:** Record Live Mix

User Story:

As a: DJ
I want: to record my live DJ session
so that: I can review or share it later.

Importance:

4/5

Estimate:

3

Acceptance Criteria

User can start/stop recording anytime.
Recorded mix is stored with a timestamp.
Playback quality remains high without distortion.

Type:

- [ ] Core DJ Features
- [x] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S020 **Story Title:** Save and Export Mix

User Story:

As a: DJ
I want: to save and export my recorded mix in various audio formats
so that: I can share it on different platforms.

Importance:

5/5

Estimate:

3

Acceptance Criteria

User can choose export formats (MP3, WAV, etc.).
File is saved with proper metadata (artist, title, etc.).
Export process is fast and does not affect sound quality.

Type:

- [ ] Core DJ Features
- [x] Effect and Mixing
- [ ] Playlist Managment
- [ ] Advance Features

Story ID: S027 **Story Title:** Analyze BPM and Key

User Story:

As a: DJ
I want: the app to analyze and display the BPM and key of tracks
So that: I can match tracks harmonically

Importance:

5/5

Estimate:

3 Story Points

Acceptance Criteria

- The app detects and displays BPM and key for all tracks.
- User can view this information in the library and playlist.
- Suggested compatible tracks appear based on BPM and key.
- Analysis runs automatically when a track is added.

Type:

- [ ] Core DJ Features
- [ ] Effect and Mixing
- [ ] Playlist Managment
- [x] Advance Features

Story ID: S028 **Story Title:** Preview Track

User Story:

As a: DJ
I want: to preview a few seconds of a track
So that: I can check if it fits my mix before playing

Importance:

4/5

Estimate:

2 Story Points

Acceptance Criteria

- User can preview the track before adding it to a playlist or queue.
- The preview starts at a key section (drop, chorus, or intro).
- Volume control is available during preview.
- Preview does not interfere with the currently playing track.

Type:

- [ ] Core DJ Features
- [ ] Effect and Mixing
- [ ] Playlist Managment
- [x] Advance Features

Story ID: S029 **Story Title:** Manual Beat Matching

User Story:

As a: DJ
I want: to manually adjust the tempo to sync beats
So that: I can create perfect transitions between tracks

Importance:

4/5

Estimate:

2 Story Points

Acceptance Criteria

- User can adjust the BPM slider for fine-tuned control.
- Beat grid is displayed for visual alignment.
- Tap tempo option is available for manual BPM detection.
- No sound distortion when adjusting BPM.

Type:

- [ ] **Core DJ Features**
- [ ] **Effect and Mixing**
- [ ] **Playlist Managment**
- [x] **Advance Features**

Story ID: S030 **Story Title:** Microphone Input

User Story:

As a: DJ
I want: to add live microphone input
So that: I can interact with my audience during a set

Importance:

4/5

Estimate:

3 Story Points

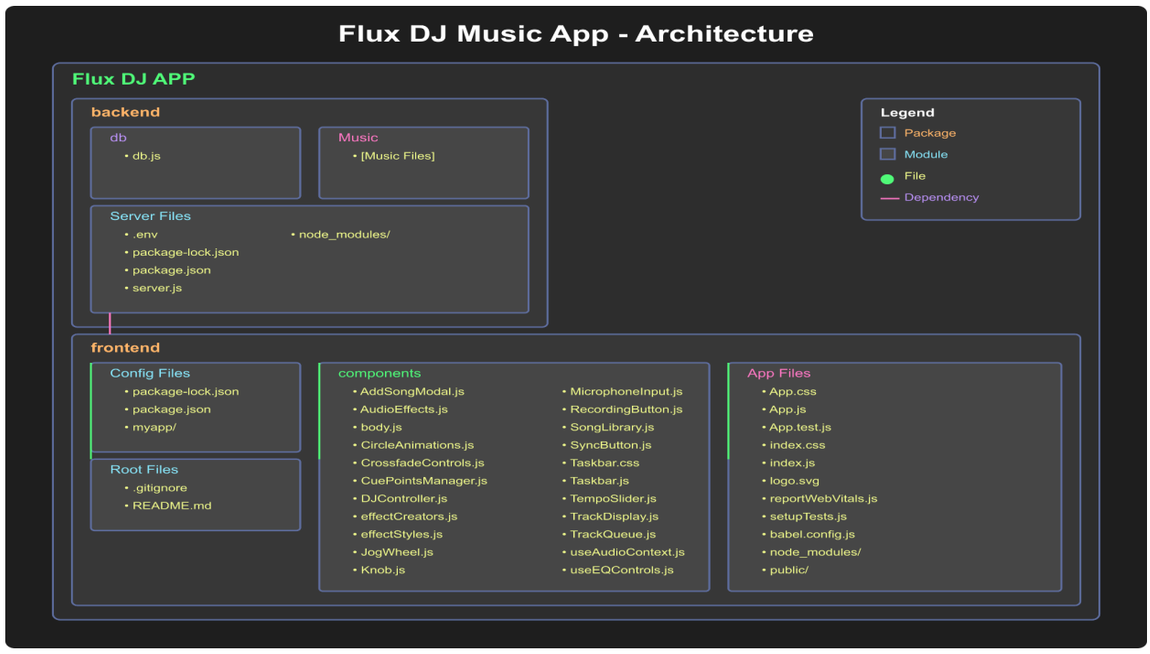
Acceptance Criteria

- User can enable or disable microphone input.
- Mic volume control is independent of track volume.
- Echo and reverb effects are available for mic input.
- Mic input does not affect Auto-Mix mode.

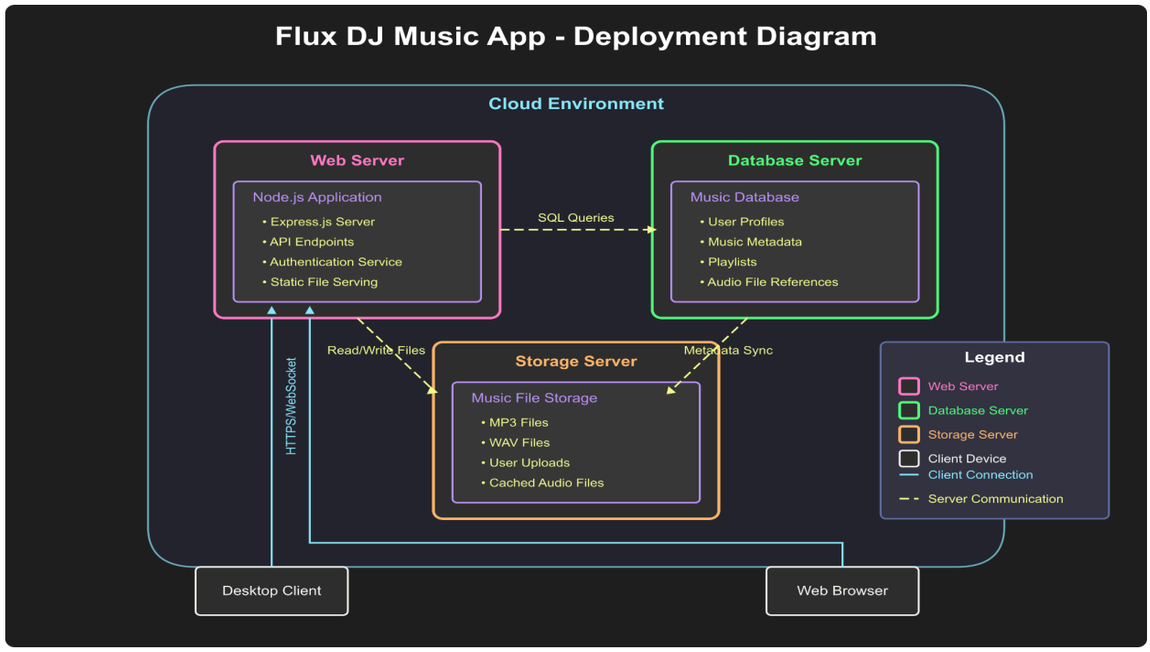
Type:

- [ ] **Core DJ Features**
- [ ] **Effect and Mixing**
- [ ] **Playlist Managment**
- [x] **Advance Features**

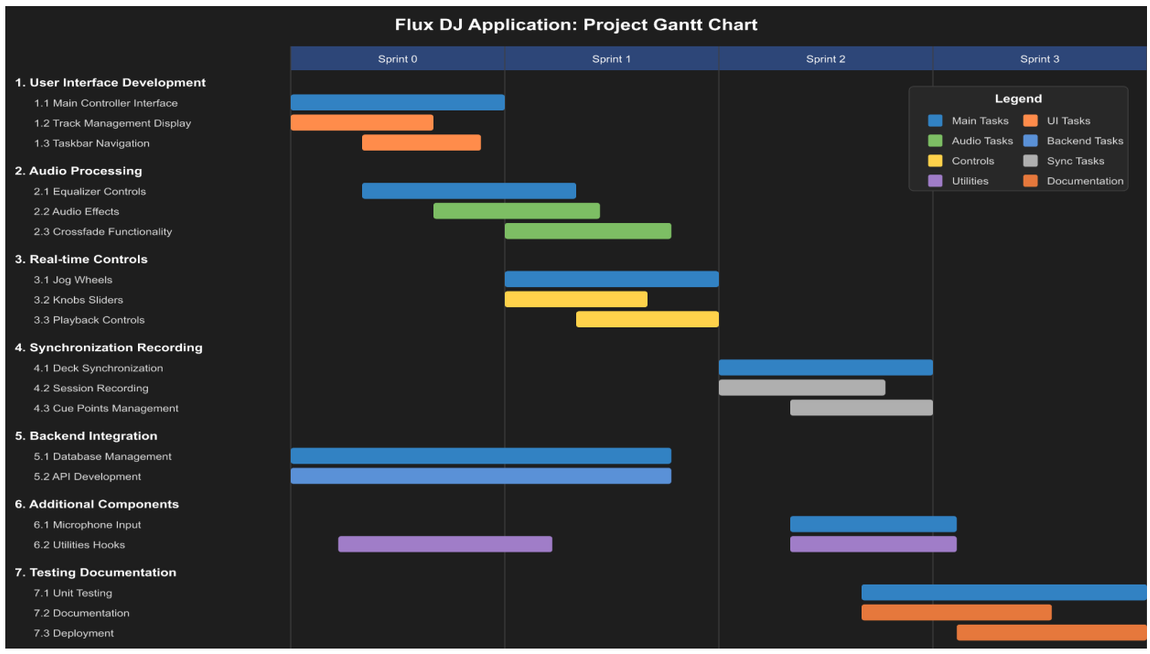
Package Diagram



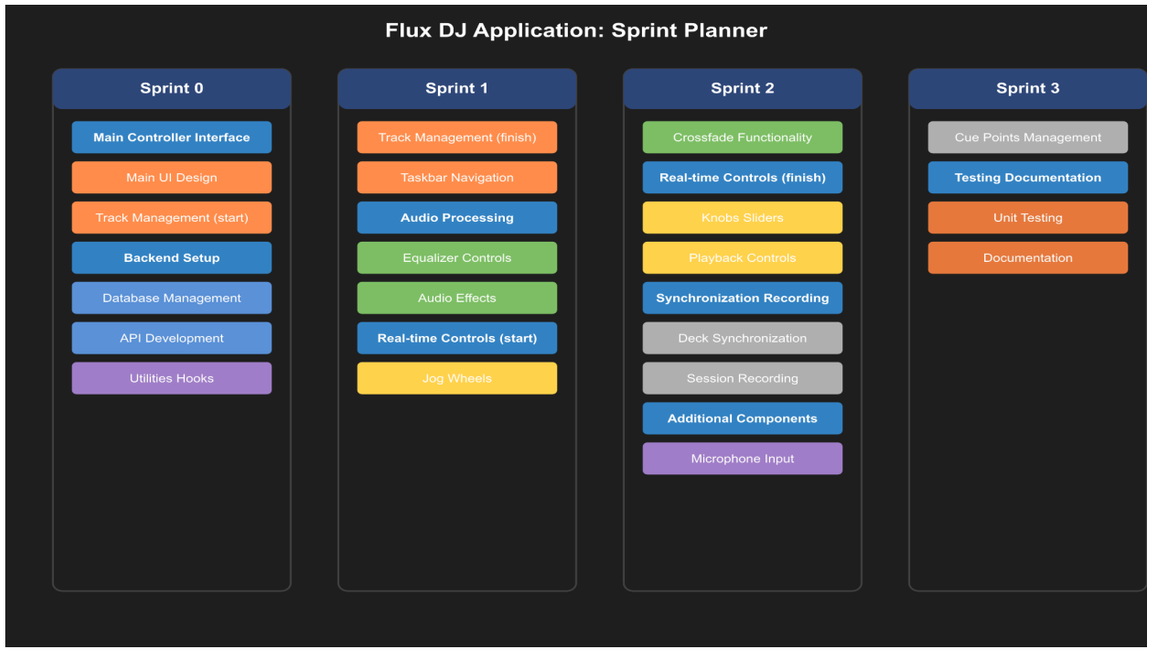
Deployment Diagram



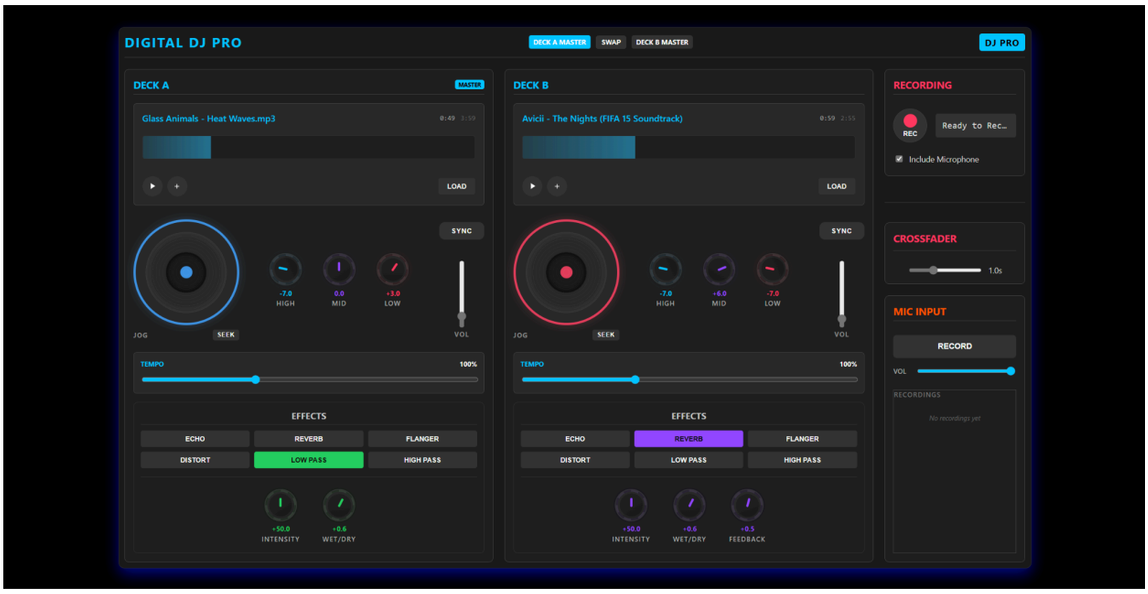
Project Gantt Chart:



Sprint Planner:



Final Output



Black Box Testing

Equivalence Class Partitioning (ECP)

Feature	Parameter	Valid Classes	Invalid Classes
Core DJ Features			
Load Track	File Type	.mp3, .wav, .flac	.txt, .exe
	File Size	≤ 100 MB	> 100 MB
Play Track	File Loaded	Loaded	Not loaded
Pause Track	Playback State	Playing	Stopped, Not started
Stop Track	Playback State	Playing, Paused	Already stopped
Crossfade Between Cues	Crossfade Duration	1–3 s	< 1 s, > 3 s
Adjust Volume	Volume %	0–100%	< 0%, > 100%
Adjust Tempo/BPM	BPM	30–300 BPM	< 30, > 300
Cue Point Setting	Cue Position	≥ 0 s, ≤ track_length	< 0 s, > track_length
Effects & Mixing			

Apply Sound Effects	Effect Type	echo, reverb, flanger, distortion	unsupported effect
	Intensity %	0–100%	< 0%, > 100%
Equalizer Adjustment	Gain (dB)	–20 to +20 dB	< –20 dB, > +20 dB
Filter Effects	Filter Type	high-pass, low-pass, band-pass	notch, all-pass
Sample Triggering	Sample Format	.mp3	.txt
Beat Sync	Tempo Difference (BPM)	–1, 0, +1	< –1, > +1
Mashup Creation	Number of Tracks	2	< 2, >2
Record Live Mix	Recording State	Started	Not started, Completed
Library & Playlist Mgmt			
Search Tracks	Keyword Length	1–100 chars	0, > 100 chars
Save and Export Mix	Format	.mp3	.txt, .exe
Advanced DJ Features			
Auto-Mix	Queue Length	≥ 2, ≤ 20	< 2, > 20
Analyze BPM and Key	File Type	.mp3, .wav, .flac	.txt
Preview Track	Preview Length	1–30 s	0 s, > 30 s
Manual Beat Matching	Tempo Offset (BPM)	–5 to +5	< –5, > +5
Microphone Input	Input Level (dB)	–60 to +10 dB	< –60 dB, > +10 dB

Boundary Value Analysis (BVA)

Feature	Parameter	Test Values
Core DJ Features		
Load Track	File Size (MB)	0 MB, 1 MB, 100 MB, 101 MB
Play Track	Playback Position (sec)	0, 1, track_length – 1, track_length
Pause Track	Playback Position (sec)	0, 1, track_length – 1, track_length
Stop Track	Playback Position (sec)	0, 1, track_length – 1, track_length
Crossfade Between Cues	Crossfade Duration (sec)	0 s, 1 s, 2 s, 3 s
Adjust Volume	Volume %	–1%, 0%, 1%, 99%, 100%, 101%
Adjust Tempo/BPM	BPM	29 BPM, 30 BPM, 150 BPM, 300 BPM, 301 BPM
Cue Point Setting	Cue Position (sec)	0 s, 1 s, track_length – 1, track_length
Effects and Mixing		
Apply Sound Effects	Effect Intensity %	–1%, 0%, 1%, 99%, 100%, 101%
Filter Effects	Cutoff Frequency (Hz)	19 Hz, 20 Hz, 1 kHz, 20 kHz, 20 001 Hz
Scratch Simulation	Scratch Speed (RPM equiv.)	0 RPM, 33 RPM, 78 RPM, 79 RPM
Beat Sync	Tempo Difference (BPM)	–2 BPM, –1 BPM, 0 BPM, +1 BPM, +2 BPM
Record Live Mix	Recording Duration (min)	0 min, 1 min, 59 min, 60 min, 61 min
Library & Playlist Mgmt		
Search Tracks	Keyword Length (chars)	0, 1, 50, 100, 101
Save and Export Mix	File Size (MB)	0 MB, 1 MB, 500 MB, 1 GB+, 1025 MB

Advanced DJ Features		
Auto-Mix	Number of Tracks in Queue	1, 2
Analyze BPM and Key	File Size (MB)	0 MB, 1 MB, 100 MB, 101 MB
Preview Track	Preview Length (sec)	0 s, 1 s, 10 s, 30 s, 31 s
Manual Beat Matching	Tempo Offset (BPM)	–5, –1, 0, +1, +5
Microphone Input	Input Level (dB)	–60 dB, –20 dB, 0 dB, +10 dB, +11 dB

Test Cases

Each test case follows the template below.

Tester: Umer Farooq

Test Type: Manual

Product: DJ Application V1.0

Documented Date: 27-Apr-2025

Suite: TS-DJ-01

T-001: Load Track

Field	Details
Test Item	Load Track
Description	Import a valid .mp3 track (< 100 MB) from device library.
Operation Procedure	1. Launch app → 2. Click “Load Track” → 3. Select valid .mp3 → 4. Click “Open”.
Pre-conditions	Application running; library view open.
Inputs	.mp3 file size = 50 MB
Expected Output	Track loads into deck; waveform displayed; duration shown.
Post-conditions	Track ready for playback.
Required Scripts	NA

Cross References	R-01, UC-LoadTrack

T-002: Play Track

Field	Details
Test Item	Play Track
Description	Start playback of loaded track.
Operation Procedure	1. Load track → 2. Click “Play” button.
Pre-conditions	Track loaded and deck selected.
Inputs	Click Play
Expected Output	Playback starts from position 0 s; play icon toggles to “Pause”.
Post-conditions	Track playing.
Required Scripts	NA
Cross References	R-02, UC-PlayTrack

T-003: Pause Track

Field	Details
Test Item	Pause Track
Description	Temporarily pause playback.
Operation Procedure	1. While playing, click “Pause” button.
Pre-conditions	Track playing.
Inputs	Click Pause
Expected Output	Playback halts; position preserved; icon toggles to “Play”.
Post-conditions	Playback paused.
Cross References	R-03, UC-PauseTrack

T-004: Stop Track

Field	Details

Test Item	Stop Track
Description	Stop playback and reset position.
Operation Procedure	1. While playing or paused, click “Stop”.
Pre-conditions	Track playing or paused.
Inputs	Click Stop
Expected Output	Playback stops; position resets to 0 s; icon shows “Play”.
Post-conditions	Deck idle.
Cross References	R-04, UC-StopTrack

T-005: Crossfade Between Tracks

Field	Details
Test Item	Crossfade
Description	Transition smoothly with 5 s crossfade.
Operation Procedure	1. Load two tracks → 2. Start both decks → 3. Set crossfade duration to 5 s → 4. Slide fader.
Pre-conditions	Two tracks loaded.
Inputs	Crossfade duration = 5 s; fader movement
Expected Output	Outgoing track fades out; incoming fades in over 5 s.
Post-conditions	Both tracks are audible at the end; correct volumes.
Cross References	R-05, UC-Crossfade

T-006: Adjust Volume

Field	Details
Test Item	Adjust Volume
Description	Increase volume from 50% to 75%.
Operation Procedure	1. Load track → 2. Set volume slider to 75%.
Pre-conditions	Track loaded and stopped or playing.
Inputs	Volume = 75%
Expected Output	Audio level increases; volume indicator shows 75%.

Cross References	R-06, UC-AdjustVolume

T-007: Adjust Tempo/BPM

Field	Details
Test Item	Adjust Tempo/BPM
Description	Change BPM from 120 to 130 without pitch shift.
Operation Procedure	1. Load track → 2. Set BPM slider/input to 130 → 3. Verify waveform speed.
Pre-conditions	Track loaded.
Inputs	BPM = 130
Expected Output	Playback speed increases; pitch remains constant.
Cross References	R-07, UC-AdjustTempo

T-010: Cue Point Setting

Field	Details
Test Item	Cue Point Setting
Description	Set cue at 45 s and jump to it.
Operation Procedure	1. Load track → 2. Scrub to 45 s → 3. Click “Set Cue” → 4. Click “Cue”.
Pre-conditions	Track loaded.
Inputs	Cue position = 45 s
Expected Output	Playback jumps to 45 s.
Cross References	R-10, UC-CuePoint

T-011: Apply Sound Effects

Field	Details
Test Item	Apply Sound Effects
Description	Apply reverb at 60% intensity.
Operation Procedure	1. Load track → 2. Select reverb → 3. Set intensity = 60% → 4. Play.

Pre-conditions	Track loaded.
Inputs	Effect=reverb; intensity = 60%
Expected Output	Reverb audible on output.
Cross References	R-11, UC-SoundEffects

T-012: Equalizer Adjustment

Field	Details
Test Item	Equalizer Adjustment
Description	Boost bass by +5 dB, cut treble by –3 dB.
Operation Procedure	1. Load track → 2. Set bass +5 dB → 3. Set treble –3 dB → 4. Play.
Pre-conditions	Track loaded.
Inputs	Bass=+5 dB; Treble=–3 dB
Expected Output	Low frequencies louder; high frequencies softer.
Cross References	R-12, UC-EQAdjustment

T-013: Filter Effects

Field	Details
Test Item	Filter Effects
Description	Apply a high-pass filter at 500 Hz.
Operation Procedure	1. Load track → 2. Choose high-pass → 3. Set cutoff=500 Hz → 4. Play.
Pre-conditions	Track loaded.
Inputs	Filter=high-pass; cutoff=500 Hz
Expected Output	Frequencies below 500 Hz attenuated.
Cross References	R-13, UC-FilterEffects

T-016: Beat Sync

Field	Details
Test Item	Beat Sync
Description	Auto-sync two tracks with 2 BPM difference.
Operation Procedure	1. Load both tracks (98 BPM & 100 BPM) → 2. Click “Beat Sync”.
Pre-conditions	Two tracks loaded.
Inputs	BPM1=98; BPM2=100
Expected Output	Second track tempo adjusts to 98 BPM; beats aligned.
Cross References	R-16, UC-BeatSync

T-019: Record Live Mix

Field	Details
Test Item	Record Live Mix
Description	Record 2-minute live session.
Operation Procedure	1. Click “Record” → 2. Perform mix for 2 min → 3. Click “Stop Recording”.
Pre-conditions	Microphone/mixer configured.
Inputs	Recording duration=2 min
Expected Output	Audio file of 2 min saved in session folder.
Cross References	R-19, UC-RecordLiveMix

White Box Testing:

Code Snippet:

```
describe("useCrossfadeAudio hook", () => {
  // This is more complex since we need to mock the Web Audio API
  // A minimal placeholder test:

  test("should create a crossfade function", () => {
    // Mock minimal required HTMLMediaElement functionality
    const mockAudioRef = {
      current: {
        play: jest.fn().mockResolvedValue(undefined),
        pause: jest.fn(),
        currentTime: 10,
        volume: 1,
      },
    };

    // Reset the mock before using it
    useCrossfadeAudio.mockClear();

    // Call the hook
    const { crossfadeTo } = useCrossfadeAudio(mockAudioRef, 1);

    // Verify that crossfadeTo is a function
    expect(typeof crossfadeTo).toBe("function");
  });
});
```

Test Cases

```
Test Suites: 4 failed, 12 passed, 16 total
Tests:       7 failed, 150 passed, 157 total
Snapshots:   0 total
Time:        19.778 s
Ran all test suites.
```

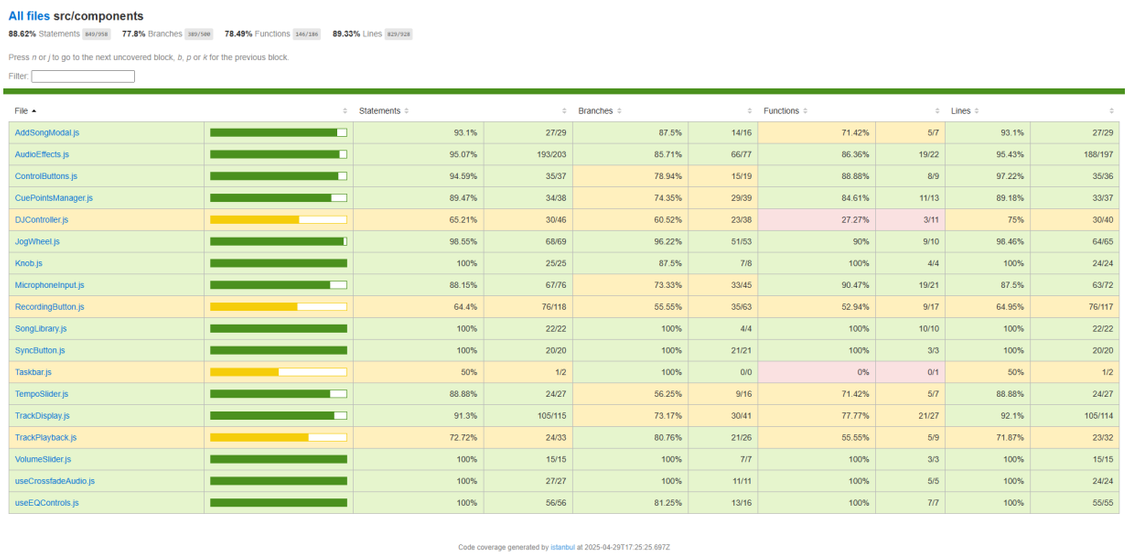
Coverage Report:

Statement Coverage: 88.62%

Branches Coverage: 77.8%

Function Coverage: 78.49%

Lines Coverage: 89.33%



All files src/components

88.62% Statements 849/958 77.8% Branches 389/500 78.49% Functions 146/186 89.33% Lines 829/928

Press n or j to go to the next uncovered block, b, p or k for the previous block.

Filter:

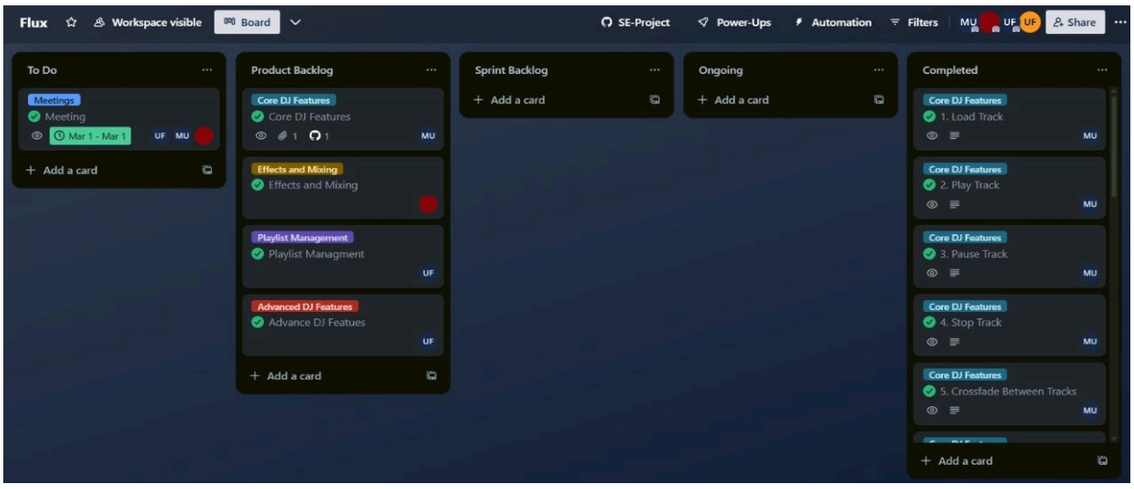
File	Statements		Branches		Functions		Lines	
AddSongModal.js	93.1%	27/29	87.5%	14/16	71.42%	5/7	93.1%	27/29
AudioEffects.js	95.07%	193/203	85.71%	66/77	86.36%	19/22	95.43%	188/197
ControlButtons.js	94.59%	35/37	78.94%	15/19	88.88%	8/9	97.22%	35/36
CuePointsManager.js	89.47%	34/38	74.35%	29/39	84.61%	11/13	89.18%	33/37
DJController.js	65.21%	30/46	60.52%	23/38	27.27%	3/11	75%	30/40
JogWheel.js	98.55%	68/69	96.22%	51/53	90%	9/10	98.46%	64/65
Knob.js	100%	25/25	87.5%	7/8	100%	4/4	100%	24/24
MicrophoneInput.js	88.15%	67/76	73.33%	33/45	90.47%	19/21	87.5%	63/72
RecordingButton.js	64.4%	76/118	55.55%	35/63	52.94%	9/17	64.95%	76/117
SongLibrary.js	100%	22/22	100%	4/4	100%	10/10	100%	22/22
SyncButton.js	100%	20/20	100%	21/21	100%	3/3	100%	20/20
Taskbar.js	50%	1/2	100%	0/0	0%	0/1	50%	1/2
TempoSlider.js	88.88%	24/27	56.25%	9/16	71.42%	5/7	88.88%	24/27
TrackDisplay.js	91.3%	105/115	73.17%	30/41	77.77%	21/27	92.1%	105/114
TrackPlayback.js	72.72%	24/33	80.76%	21/26	55.55%	5/9	71.87%	23/32
VolumeSlider.js	100%	15/15	100%	7/7	100%	3/3	100%	15/15
useCrossfadeAudio.js	100%	27/27	100%	11/11	100%	5/5	100%	24/24
useEQControls.js	100%	56/56	81.25%	13/16	100%	7/7	100%	55/55

Code coverage generated by istanbul at 2025-04-29T17:25:25.697Z

Work Distribution

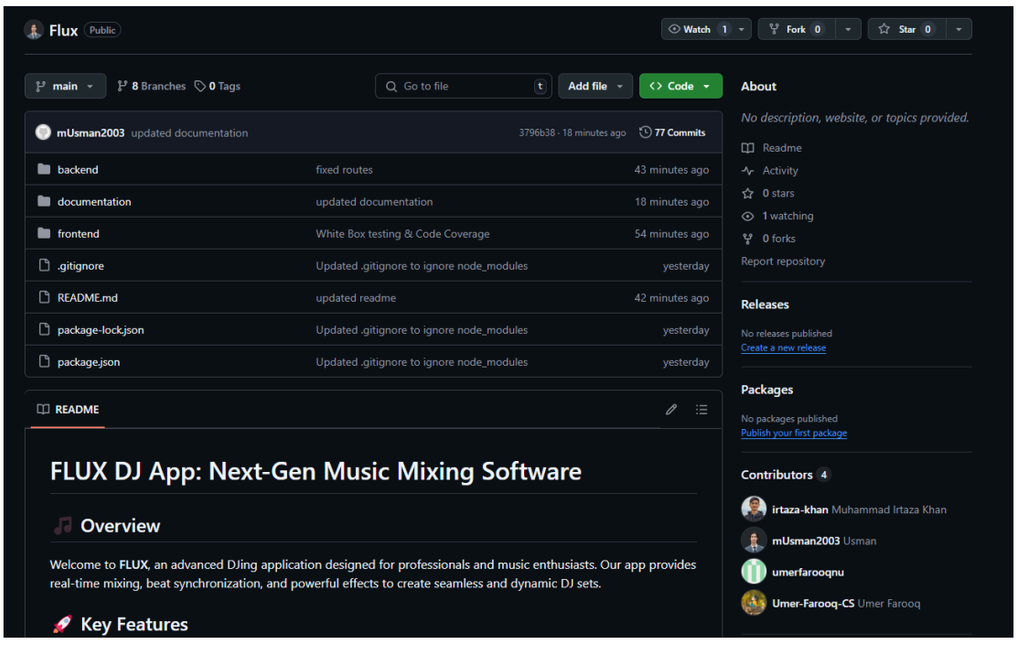
- **Umer Farooq:** UI Implementation, Documentation and BlackBox Testing
- **Muhammad Usman:** Backend**,** Database, Project Management
- **Muhammad Irtaza Khan:** Functionality Implementation, WhiteBox Testing

Trello Workspace



Github Repository

mUsman2003/Flux



Conclusion and Final Reflections

Lessons Learned

SCRUM and Daily Standups Fostered Steady Progress

Implementing SCRUM methodologies, particularly daily standup meetings, significantly

enhanced our team's productivity and cohesion. These brief, focused sessions facilitated real-time updates, quick identification of impediments, and immediate course corrections, ensuring continuous momentum throughout the development lifecycle.

Trello Streamlined Task Management and Prioritization

Utilizing Trello as our primary task management tool brought clarity and organization to our workflow. Its visual boards and card system allowed for effective tracking of tasks, deadlines, and responsibilities, enabling the team to prioritize work efficiently and adapt to changes swiftly.

Rigorous Testing Ensured Application Reliability

A strong emphasis on testing, particularly through the adoption of JEST, was pivotal in maintaining the application's reliability. Comprehensive unit and integration tests helped in early detection of bugs and regressions, leading to a more stable and robust DJ application.

Team Reflections

Clear Roles and Open Communication Enhanced Collaboration

Defining clear roles and fostering open lines of communication were instrumental in improving team dynamics. This clarity reduced overlaps, minimized misunderstandings, and promoted accountability, resulting in a more harmonious and effective collaboration.

Complex Testing Scenarios Highlighted JEST’s Capabilities

The complexity of our testing requirements underscored the value of JEST as a testing framework. Its flexibility and comprehensive feature set allowed us to simulate intricate user interactions and system behaviors, ensuring thorough validation of the application's functionality.

User Feedback Drove Rapid Iterations and Improvements

Actively seeking and incorporating user feedback was a cornerstone of our development process. This user-centric approach enabled us to make informed decisions, prioritize features that mattered most to our users, and iterate rapidly to enhance the overall user experience.

Final Thoughts

Delivery of a Reliable and Modular DJ Application

The culmination of our efforts is a reliable DJ application characterized by modular architecture and thoroughly tested code. This foundation not only ensures current performance and stability but also positions the application for future enhancements and scalability.

Agile Practices as a Catalyst for Future Scalability

The adoption of Agile methodologies proved to be a catalyst for efficient development and adaptability. The iterative nature of Agile allowed for continuous improvement, responsiveness to change, and alignment with user needs, setting a precedent for future projects and scalability.

In summary, the integration of SCRUM practices, effective task management through Trello, rigorous testing protocols, and a user-focused development approach coalesced to produce a high-quality DJ application. The lessons learned and methodologies adopted during this project will serve as valuable assets in our ongoing and future endeavors.