DJ Application

Work Breakdown Structure

27-April 2025

ISSUED BY

The Big 3

REPRESENTATIVE

Umer Farooq	221-0891
Muhammad Usman	221-0900
Muhammad Irtaza Khan	221-0911

1. File Breakdown Structure (WBS)	2
Flux DJ Application	2
1.1 User Interface Development	2
1.2 Audio Processing	2
1.3 Real-time Controls	2
1.4 Synchronization & Recording	3
1.5 Backend Integration	3
1.6 Microphone Input Handling (MicrophoneInput.js)	3
1.7 Utilities & Hooks	3
1.8 Testing & Quality Assurance	4
1.9 Documentation & Deployment	4
2. Workflow Explanation: Flux DJ Application	4
Track Loading & Initialization	4
2. Real-Time Audio Adjustments	4
3. Deck Synchronization	5
4. Mixing & Crossfading	5
5. Effects & Scrubbing	5
6. Microphone Input & Mixing	6
7. Recording & Saving Sessions	6
8. Backend Interaction	6
9. State Management & Reactivity	6
10. Visual Feedback	7
Project Gantt Chart	7
ISSUED BY	0
REPRESENTATIVE	0

1. File Breakdown Structure (WBS)

1. Flux DJ Application

1.1 User Interface Development

- 1.1.1 Main Controller Interface (DJController.js)
 - Layout design for dual decks (Deck A & B)
 - Integration of sub-components: TrackDisplay, JogWheel, Knob, VolumeSlider
 - Master/Slave toggle functionality for deck synchronization
 - Real-time visual feedback (e.g., master deck indicator, tempo sliders)
- 1.1.2 Track Management & Display (TrackDisplay.js, SongLibrary.js, AddSongModal.js)
 - File upload and track metadata display (BPM, title, waveform)
 - Song library UI for browsing and selecting tracks
 - Modal window for adding songs (AddSongModal.js)

1.2 Audio Processing

- 1.2.1 Equalizer Controls (useEQControls.js, Knob.js)
 - High/Mid/Low EQ knobs with dB adjustment (-12 to +12)
 - State management for EQ values per deck
- 1.2.2 Audio Effects (AudioEffects.js, effectCreators.js, effectsData.js)
 - Implementation of effects (e.g., reverb, delay, flanger)
 - Preset management and UI styling (effectsStyles.js)
- 1.2.3 Crossfade Functionality (useCrossfadeAudio.js, CrossfadeControls.js)
 - Crossfader UI with adjustable duration (0.1s-3s)
 - Smooth audio transition between decks

1.3 Real-time Controls

• 1.3.1 Jog Wheels (JogWheel.js)

- Scrubbing/track navigation with touch or mouse input
- Pitch bend and tempo adjustment integration

1.3.2 Knobs & Sliders (Knob.js, VolumeSlider.js, TempoSlider.js)

- Volume sliders with real-time audio adjustments
- Tempo sliders with BPM synchronization

• 1.3.3 Playback Controls (ControlButtons.js, TrackPlayback.js)

- Play, pause, cue, and loop buttons
- Track progress visualization

1.4 Synchronization & Recording

• 1.4.1 Deck Synchronization (SyncButton.js)

- Master/Slave tempo matching between decks
- Automatic BPM alignment using tempoRef

• 1.4.2 Session Recording (RecordingButton.js)

- Capture audio output from both decks
- Save recordings to local storage or backend

• 1.4.3 Cue Points Management (CuePointsManager.js)

Set, delete, and jump to cue points during playback

1.5 Backend Integration

• 1.5.1 Database Management (Postgres)

- Schema design for storing tracks, user preferences, and recordings
- SQL queries for CRUD operations

• 1.5.2 API Development (Express)

- RESTful endpoints for track upload, library access, and recording storage
- o Integration with frontend via Axios/fetch

1.6 Microphone Input Handling (MicrophoneInput.js)

- Microphone audio capture and mixing with deck output
- Noise reduction and gain control

1.7 Utilities & Hooks

- 1.7.1 Audio Context Management (useAudioContext.js)
 - o Web Audio API initialization and resource cleanup
- 1.7.2 State Management (body.js, useEQControls.js)
 - Global state for track loading, effects, and deck status

1.8 Testing & Quality Assurance

- Unit testing for audio processing logic (Jest)
- Cross-browser compatibility checks
- User testing for UI responsiveness and audio latency

1.9 Documentation & Deployment

- Technical documentation for component interactions
- Deployment pipeline (Docker, AWS/Heroku)
- User manual for DJ workflows

2. Workflow Explanation: Flux DJ Application

The Flux DJ application is designed to emulate a professional DJ setup with two decks, real-time audio processing, and seamless integration between hardware-like controls and software. Below is a detailed breakdown of the user and system workflow:

1. Track Loading & Initialization

- User Action:
 - Upload tracks to Deck A or Deck B via AddSongModal.js or select from the SongLibrary.js
- System Flow:
 - TrackDisplay.js handles file uploads, extracts metadata (BPM, duration),
 and initializes the audio buffer using useAudioContext.js
 - Audio files are stored in the Postgres database via Express API endpoints
 - The loaded track is displayed with a waveform preview and BPM data

2. Real-Time Audio Adjustments

User Action:

 Adjust EQ (High/Mid/Low) using Knob.js, modify volume with VolumeSlider.js, or tweak tempo via TempoSlider.js

System Flow:

- The useEQControls.js hook processes EQ changes and applies them to the Web Audio API nodes
- Volume and tempo sliders update the audioRef state in DJController.js, altering gain or playback rate in real time
- Changes are visually reflected in the UI (e.g., knob positions, slider values)

3. Deck Synchronization

User Action:

 Press the SyncButton.js to match the tempo of the slave deck to the master deck

System Flow:

- The master deck's BPM (from TempoSlider.js) is read and applied to the slave deck's audioRef.playbackRate
- The masterDeck state in DJController.js determines which deck controls synchronization

4. Mixing & Crossfading

User Action:

 Use the CrossfadeControls.js slider to transition audio between Deck A and Deck B

System Flow:

- The useCrossfadeAudio.js hook dynamically adjusts the gain of both decks over a user-defined fadeDuration
- The crossfader's position determines the volume balance (e.g., fully left = Deck A plays, fully right = Deck B plays)

5. Effects & Scrubbing

User Action:

 Apply audio effects (e.g., reverb, delay) via AudioEffects.js or scrub through tracks using JogWheel.js

System Flow:

- effectCreators.js generates Web Audio nodes (e.g., ConvolverNode for reverb) and connects them to the deck's audio chain
- The JogWheel.js adjusts the audioRef.currentTime property to simulate vinyl-like scratching

6. Microphone Input & Mixing

- User Action:
 - Enable MicrophoneInput.js to overlay voice commentary or live vocals
- System Flow:
 - The navigator.mediaDevices.getUserMedia API captures microphone input
 - The mic audio is routed through the same Web Audio context as the decks, allowing real-time mixing

7. Recording & Saving Sessions

- User Action:
 - Click RecordingButton.js to capture the current mix
- System Flow:
 - The MediaRecorder API records the final output of the audio context
 - Recordings are saved to Postgres via Express endpoints or locally as WAV/MP3 files

8. Backend Interaction

- System Flow:
 - Express API endpoints handle:
 - Uploading/downloading tracks (/api/tracks)
 - Saving/loading user preferences (/api/settings)
 - Storing recordings (/api/recordings)
 - Postgres manages relational data (tracks, BPM metadata, user accounts)

9. State Management & Reactivity

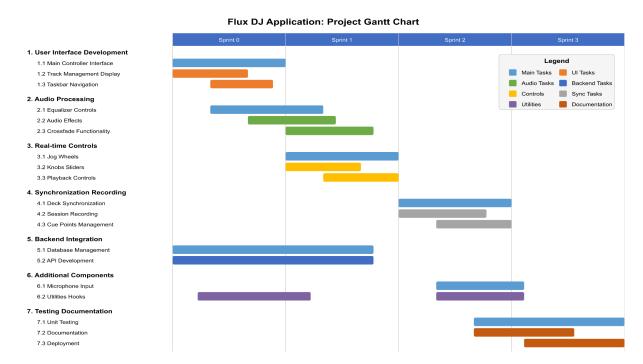
- Key Mechanisms:
 - React Hooks: useState and useRef in DJController.js track deck states (e.g., deckA, deckB, fadeDuration)

- Web Audio API: Manages real-time audio processing, effects, and mixing
- Event Listeners: UI components (sliders, buttons) trigger audio graph updates

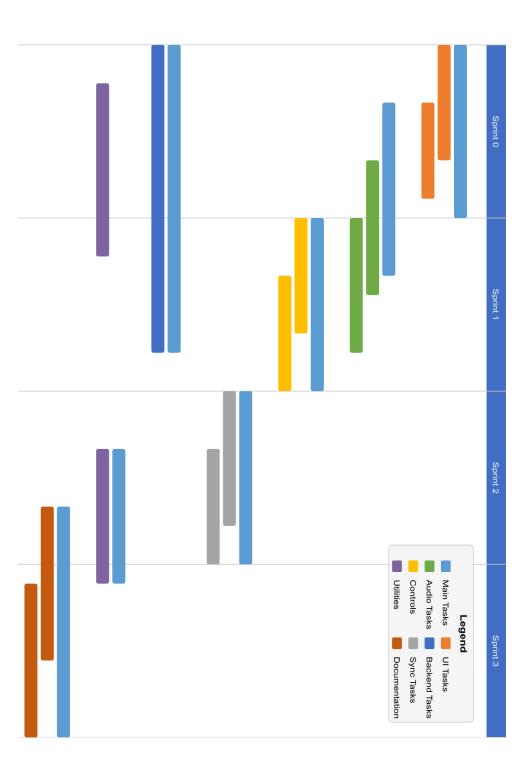
10. Visual Feedback

- The UI updates in real time using:
 - o TrackDisplay.js: Shows waveform progress and playback position
 - o CuePointsManager.js: Highlights saved cue points on the waveform
 - Taskbar.js: Displays global controls (play/pause, library access)

Project Gantt Chart



Flux DJ Application: Project Gantt Chart



7. Testing Documentation

7.1 Unit Testing

7.2 Documentation

7.3 Deployment

6. Additional Components

5.1 Database Management5.2 API Development

6.1 Microphone Input

6.2 Utilities Hooks

5. Backend Integration

4.1 Deck Synchronization4.2 Session Recording4.3 Cue Points Management

4. Synchronization Recording

3.1 Jog Wheels3.2 Knobs Sliders3.3 Playback Controls

3. Real-time Controls

2.3 Crossfade Functionality

2. Audio Processing

1.3 Taskbar Navigation

2.1 Equalizer Controls

2.2 Audio Effects

1. User Interface Development

1.1 Main Controller Interface

1.2 Track Management Display