Requirements v1.0

A. Functional Requirements (Core System Behavior)

1. **(Minimal template)**
   When a user selects two Spotify tracks,
   the system should display each track’s BPM and musical key,
   so that the user can evaluate tempo and harmonic compatibility.

2. **(Minimal template)**
   When a track is playing,
   the system should allow the user to scrub forward or backward in time,
   so that transition points can be manually aligned.

3. **(Constraint-based)**
   The system shall support simultaneous playback of at least two tracks for comparison and alignment purposes.

4. **(Minimal template)**
   When the user adjusts tempo or pitch controls,
   the system should update audio playback in near real time,
   so that the user can hear the effect immediately.

5. **(Use-case style)**
   A user can pause, resume, and restart individual tracks independently during mashup preparation.

6. **(Minimal template)**
   When a user modifies EQ or volume settings,
   the system should apply those changes only to the selected track,
   so that layered mixing remains controllable.

7. **(Constraint-based)**
   The system shall operate entirely within a web browser without requiring specialized DJ hardware.

B. AI Recommendation Requirements

8. **(Minimal template)**
   When the user requests song recommendations,
   the system should suggest tracks with similar BPM and compatible musical keys,
   so that transitions are more likely to sound coherent.

9. **(Explanation requirement)**
   When an AI-generated recommendation is shown,
   the system should display the primary factors that influenced the recommendation
   (e.g., BPM difference, key compatibility, genre tags).

10. **(Negative requirement)**
    The system should not automatically replace or reorder user-selected tracks based solely on AI recommendations.

11. **(Minimal template)**
    When AI recommendations are unavailable or fail,
    the system should continue to support full manual mashup functionality,
    so that creative work is not blocked by AI components.

C. Trust & Transparency Requirements (Non-Functional)

12. **(Behavioral constraint)**
    The system shall treat AI-generated recommendations as optional suggestions rather than authoritative decisions.

13. **(Minimal template)**
    When automated pitch or tempo adjustments are applied,
    the system should allow the user to disable or revert those changes,
    so that users retain creative control.

14. **(Trust hypothesis style)**
    The system is expected to produce audible artifacts during extreme pitch or tempo changes,
    and this behavior is considered acceptable within the scope of the project.

15. **(Disclosure requirement)**
    The system should inform users that AI recommendations may reflect bias

toward popular or well-represented artists.

D. Performance & Scope Constraints (Semester-Aware)

16. **(Constraint-based)**
 The system shall prioritize responsiveness and usability over professional-grade audio fidelity.

17. **(Minimal template)**
 When network latency or API limits affect playback or recommendations,
 the system should notify the user,
 so that unexpected behavior is not silently hidden.

18. **(Explicit limitation)**
 The system does not guarantee copyright compliance or fair-use validation for user-created mashups.

Requirements v1.1

A. Functional Requirements (Core System Behavior)

1. (MUST HOLD) When a user selects two Spotify tracks, the system should display each track's BPM and musical key, so that the user can evaluate tempo and harmonic compatibility.
2. (MUST HOLD) When a track is playing, the system should allow the user to scrub forward or backward in time, so that transition points can be manually aligned.
3. (MUST HOLD) When comparing tracks, the system should support simultaneous playback of at least two tracks, so that alignment and compatibility can be evaluated in real time.
4. (MAY FAIL) When the user adjusts tempo or pitch controls, the system should update audio playback in near real time, so that the user can hear the effect immediately.
5. (MUST HOLD) When working on a mashup, the system should allow the user to pause, resume, and restart individual tracks independently, so that the preparation workflow remains flexible.
6. (MUST HOLD) When a user modifies EQ or volume settings, the system should apply those changes only to the selected track, so that layered mixing remains controllable.
7. (MUST HOLD) When accessing the system, the user should be able to operate entirely within a web browser without requiring specialized DJ hardware, so that the tool remains accessible to all users.
8. (MUST HOLD) When a user completes a mashup or set, the system should allow them to save their work including track selections, alignment points, and audio parameter settings, so that projects can be resumed later.
9. (MUST HOLD) When storing mashup data, the system should persist saved configurations (track IDs, timestamps, EQ/volume/pitch settings) to user storage without storing actual audio files, so that storage requirements remain minimal and copyright is respected.
10. (REPEAT) When a user returns to the system, the system should allow them to load a previously saved mashup or set and resume editing with all settings intact, so that creative work can continue across sessions.
11. (MUST HOLD) When a user saves a mashup, the system should allow them to provide a custom name or label, so that multiple projects can be easily distinguished.

B. AI Recommendation Requirements

1. (MUST HOLD) When the user requests song recommendations, the system should suggest tracks with similar BPM and compatible musical keys, so that transitions are more likely to sound coherent.
2. (REPEAT) When an AI-generated recommendation is shown, the system should display the primary factors that influenced the recommendation (e.g., BPM difference, key compatibility, genre tags), so that users understand the reasoning behind suggestions.
3. (MUST HOLD) When AI provides recommendations, the system should not automatically replace or reorder user-selected tracks based solely on those recommendations, so that user creative control is preserved.
4. (MUST HOLD) When AI recommendations are unavailable or fail, the system should continue to support full manual mashup functionality, so that creative work is not blocked by AI components.
5. (MAY FAIL) When displaying recommendations, the system should limit results to 5-10 suggestions, so that users are not overwhelmed and can explore options thoughtfully.

C. Trust & Transparency Requirements (Non-Functional)

1. (REPEAT) When presenting AI recommendations, the system should treat them as optional suggestions rather than authoritative decisions, so that users maintain agency over creative choices.
2. (MUST HOLD) When automated pitch or tempo adjustments are applied, the system should allow the user to disable or revert those changes, so that users retain creative control.
3. (MAY FAIL) When pitch shift exceeds ±3 semitones or tempo change exceeds ±15%, the system should warn users that audio quality degradation may occur, so that users can make informed decisions about extreme adjustments.
4. (NOT GUARANTEED) When providing AI recommendations, the system should inform users that suggestions may reflect bias toward popular or well-represented artists, so that users understand potential limitations.
5. (MAY FAIL) When storing user data, the system should not share saved mashup configurations or user listening patterns with third parties beyond what is required for Spotify API functionality, so that user privacy is protected.
6. (MAY FAIL) When a user selects a track, the system should authenticate with Spotify API and retrieve track metadata (BPM, key, genre tags) within 5 seconds, so that users experience minimal latency during track exploration.

D. Performance & Scope Constraints (Semester-Aware)

1. (MAY FAIL) When processing audio, the system should prioritize responsiveness and usability over professional-grade audio fidelity, so that the project remains achievable within semester constraints.
2. (MAY FAIL) When network latency or API limits affect playback or recommendations, the system should notify the user, so that unexpected behavior is not silently hidden.
3. (MAY FAIL) When users create mashups, the system does not guarantee copyright compliance or fair-use validation for user-created content, so that users understand their legal responsibilities.
4. (MAY FAIL) When displaying audio changes, the system should provide visual feedback (waveforms, level meters), so that hearing-impaired users can engage with core features.

Requirements v1.2

A. Functional Requirements (Core System Behavior)

Must Have

1. When a user selects two Spotify tracks, the system should display each track's BPM and musical key, so that the user can evaluate tempo and harmonic compatibility.
2. When a track is playing, the system should allow the user to scrub forward or backward in time, so that transition points can be manually aligned.
3. When comparing tracks, the system should support simultaneous playback of at least two tracks, so that alignment and compatibility can be evaluated in real time.
4. When working on a mashup, the system should allow the user to pause, resume, and restart individual tracks independently, so that the preparation workflow remains flexible.
5. When a user modifies EQ or volume settings, the system should apply those changes only to the selected track, so that layered mixing remains controllable.
6. When accessing the system, the user should be able to operate entirely within a web browser without requiring specialized DJ hardware, so that the tool remains accessible to all users.
7. When a user completes a mashup or set, the system should allow them to save their work including track selections, alignment points, and audio parameter settings, and load previously saved projects with all settings intact, so that creative work can continue across sessions.
8. When storing mashup data, the system should persist saved configurations (track IDs, timestamps, EQ/volume/pitch settings) to user storage without storing actual audio files, so that storage requirements remain minimal and copyright is respected.
9. When a user saves a mashup, the system should allow them to provide a custom name or label, so that multiple projects can be easily distinguished.

May Fail

10. When the user adjusts tempo or pitch controls, the system should update audio playback in near real time, so that the user can hear the effect immediately.

11. When a user selects a track, the system should authenticate with Spotify API and retrieve track metadata (BPM, key, genre tags) within 5 seconds, so that users experience minimal latency during track exploration.

B. AI Recommendation Requirements

Must Have

1. When the user requests song recommendations, the system should suggest tracks with similar BPM and compatible musical keys, so that transitions are more likely to sound coherent.
2. When an AI-generated recommendation is shown, the system should display the primary factors that influenced the recommendation (e.g., BPM difference, key compatibility, genre tags) and treat them as optional suggestions rather than authoritative decisions, so that users understand the reasoning behind suggestions and maintain agency over creative choices.
3. When AI provides recommendations, the system should not automatically replace or reorder user-selected tracks based solely on those recommendations, so that user creative control is preserved.
4. When AI recommendations are unavailable or fail, the system should continue to support full manual mashup functionality, so that creative work is not blocked by AI components.

May Fail

5. When displaying recommendations, the system should limit results to 5-10 suggestions, so that users are not overwhelmed and can explore options thoughtfully.

Not Guaranteed

6. When providing AI recommendations, the system should inform users that suggestions may reflect bias toward popular or well-represented artists, so that users understand potential limitations.

C. Trust, Transparency & User Control Requirements (Non-Functional)

Must Have

1. When automated pitch or tempo adjustments are applied, the system should allow the user to disable or revert those changes, so that users retain creative control.

May Fail

2. When pitch shift exceeds ±3 semitones or tempo change exceeds ±15%, the system should warn users that audio quality degradation may occur, so that users can make informed decisions about extreme adjustments.
3. When storing user data, the system should not share saved mashup configurations or user listening patterns with third parties beyond what is required for Spotify API functionality, so that user privacy is protected.

D. Performance & Scope Constraints (Non-Functional, Semester-Aware)

May Fail

1. When processing audio, the system should prioritize responsiveness and usability over professional-grade audio fidelity, so that the project remains achievable within semester constraints.
2. When network latency or API limits affect playback or recommendations, the system should notify the user, so that unexpected behavior is not silently hidden.
3. When users create mashups, the system does not guarantee copyright compliance or fair-use validation for user-created content, so that users understand their legal responsibilities.
4. When displaying audio changes, the system should provide visual feedback (waveforms, level meters), so that hearing-impaired users can engage with core features.

Requirements v1.3

A. Functional Requirements

Must Have

1. When a user selects two Spotify tracks, the system should display each track's BPM and musical key, so that the user can evaluate tempo and harmonic compatibility.
2. When a track is playing, the system should allow the user to scrub forward or backward in time, so that transition points can be manually aligned.
3. When comparing tracks, the system should support simultaneous playback of at least two tracks, so that alignment and compatibility can be evaluated in real time.
4. When working on a mashup, the system should allow the user to pause, resume, and restart individual tracks independently, so that the preparation workflow remains flexible.
5. When a user modifies EQ or volume settings, the system should apply those changes only to the selected track, so that layered mixing remains controllable.
6. When accessing the system, the user should be able to operate entirely within a web browser without requiring specialized DJ hardware, so that the tool remains accessible to all users.
7. When a user completes a mashup or set, the system should allow them to save their work including track selections, alignment points, and audio parameter settings, and load previously saved projects with all settings intact, so that creative work can continue across sessions.
8. When storing mashup data, the system should persist saved configurations (track IDs, timestamps, EQ/volume/pitch settings) to user storage without storing actual audio files, so that storage requirements remain minimal and copyright is respected.
9. When a user saves a mashup, the system should allow them to provide a custom name or label, so that multiple projects can be easily distinguished.
10. When the user requests song recommendations, the system should suggest tracks with similar BPM and compatible musical keys, so that transitions are more likely to sound coherent.
11. When an AI-generated recommendation is shown, the system should display the primary factors that influenced the recommendation (e.g., BPM difference, key compatibility, genre tags), so that users understand the reasoning behind suggestions.

12. When AI provides recommendations, the system should not automatically replace or reorder user-selected tracks based solely on those recommendations, so that user creative control is preserved.
13. When AI recommendations are unavailable or fail, the system should continue to support full manual mashup functionality, so that creative work is not blocked by AI components.
14. When automated pitch or tempo adjustments are applied, the system should allow the user to disable or revert those changes, so that users retain creative control.

May Fail

15. When the user adjusts tempo or pitch controls, the system should update audio playback in near real time, so that the user can hear the effect immediately.
16. When a user selects a track, the system should authenticate with Spotify API and retrieve track metadata (BPM, key, genre tags) within 5 seconds, so that users experience minimal latency during track exploration.
17. When displaying recommendations, the system should limit results to 5-10 suggestions, so that users are not overwhelmed and can explore options thoughtfully.
18. When displaying audio changes, the system should provide visual feedback (waveforms, level meters), so that hearing-impaired users can engage with core features.

Not Guaranteed

19. When providing AI recommendations, the system should inform users that suggestions may reflect bias toward popular or well-represented artists, so that users understand potential limitations.

B. Non-Functional Requirements

Must Have

1. When presenting AI recommendations, the system should treat them as optional suggestions rather than authoritative decisions, so that users maintain agency over creative choices.
    a. **Trust Hypothesis:** The system does not automatically apply AI-recommended tracks without explicit user confirmation.

May Fail

2. When pitch shift exceeds ±3 semitones or tempo change exceeds ±15%, the system should warn users that audio quality degradation may occur, so that users can make informed decisions about extreme adjustments.
    a. **Trust Hypothesis:** The system displays a warning before applying pitch shifts beyond ±3 semitones or tempo changes beyond ±15%.
3. When storing user data, the system should not share saved mashup configurations or user listening patterns with third parties beyond what is required for Spotify API functionality, so that user privacy is protected.
    a. **Trust Hypothesis:** The system does not transmit user mashup data or listening patterns to any third-party services except Spotify API endpoints.
4. When processing audio, the system should prioritize responsiveness and usability over professional-grade audio fidelity, so that the project remains achievable within semester constraints.
    a. **Trust Hypothesis:** The system may produce audible artifacts or reduced audio quality in exchange for maintaining responsive playback controls.
5. When network latency or API limits affect playback or recommendations, the system should notify the user, so that unexpected behavior is not silently hidden.
    a. **Trust Hypothesis:** The system displays error notifications when Spotify API rate limits or network failures prevent successful operations.
6. When users create mashups, the system does not guarantee copyright compliance or fair-use validation for user-created content, so that users understand their legal responsibilities.
    a. **Trust Hypothesis:** The system does not validate copyright status or fair-use compliance of user-created mashups.