Definition of Done by User Story -

Sprint 1 -

User story 1
As a user, I want to be able to sign up and log in so I can access my own profile

DoD

- Code checked into GitHub and reviewed by teammates
- Signup API route (POST /api/auth/signup) implemented
- Credentials login via NextAuth Credentials provider works without errors
- Google OAuth provider integrated with correct env vars
- Passwords hashed using bcrypt before storage
- User object stored correctly in MongoDB
- External API dependencies (NextAuth, Google OAuth) documented
- Basic manual test cases executed (correct login, incorrect login, empty fields)
- No crashes, console errors, or unhandled exceptions
- UI for login/signup renders correctly across pages

Acceptance Criteria

- A new user can sign up using email, username, and password
- Credentials login works: users can enter username/email + password and authenticate
- Google OAuth login works, creating or linking an account
- Successful signup/login redirects user to homepage
- User session persists across refresh
- User data is stored in MongoDB with the required fields
- Invalid credentials trigger an error message

User story 2 -
As a user, I want to see a landing page to navigate the web page

DoD

- /login, /signup, and / routes implemented
- Header component created, using consistent styling
- Navigation buttons link to correct pages
- Header code reviewed and pushed to GitHub

- Static build compiles with no routing errors
- No console warnings in navigation components
- Manual navigation tests passed

Acceptance Criteria

- User can navigate between Home, Login, and Signup pages
- Header appears on all relevant pages
- Clicking "Home" takes the user back to /
- Clicking “Login” or “Signup” opens their correct respective forms

Sprint 2 -

User story 1 -
As a user, I want to search for music albums so that I can find and track the music I listen to

DoD

- Search bar component implemented and reviewed
- Spotify API integration functional using /api/spotify proxy route
- Debouncing implemented to prevent excessive API calls
- Album and artist hits correctly typed (AlbumHit, ArtistHit)
- No unhandled API failures (loading + error states present)
- Unit tests / manual tests for search flow completed

Acceptance Criteria

- User can type a search query
- Live search results appear (albums/artists)
- Clicking a result opens its album or artist page
- Handles empty state and no-results state
- Search feels fast, responsive, and reliable

User story 2 -

As a user, I want to view my profile page displaying saved albums and reviews so that I can look back at them later when I forget which albums that I listened to

DoD

- /profile dynamic route implemented
- Profile page loads user document from MongoDB
- UI shows username, profile picture, and bio (or placeholders)
- Saved albums and reviews fetched and displayed
- Code reviewed and pushed to GitHub
- Page renders with no console errors

Acceptance Criteria

- User can open their profile page
- Profile correctly displays:
    - Username

    - Profile picture (or default avatar)
    - Bio (or placeholder text “No bio yet”)
- Saved albums and reviews appear (or show placeholders if none)
- Profile loads consistently even for users missing optional fields

User Story 3 -

As a user, I want to be able to write a bio and change my name so that I can customize my profile and express myself.

DoD

- PUT route for updating bio implemented (/api/me/bio)
- MongoDB updated with new bio field
- Profile UI includes “Edit Bio” button + editing form/modal
- Response handling includes success/error states
- Code merged into main and tested
- No validation or type errors
- Bio persists on refresh

Acceptance Criteria

- User can click “Edit Bio”
- A text box or modal appears for input
- Updating bio shows success message
- Updated bio stored in MongoDB and re-rendered on the profile page
- Empty or invalid bio is handled cleanly

User story 4 -

As a user, I want to be able to see individual pages for each album so that I can see more details on the album I am reviewing

DoD

- Dynamic /album/[id] route implemented
- Spotify album details fetched server-side or client-side
- Review submission UI created
- POST /api/reviews stores reviews in MongoDB
- Reviews are displayed in chronological order

- Error handling for album fetch failures
- Back button navigates safely
- No console errors
- Code reviewed and committed to GitHub

Acceptance Criteria

- User can open any album page
- Album page shows:
    - Album cover
    - Title
    - Artists
    - Tracks (optional depending on implementation)
- User can click “Review” and submit text + rating
- Review appears immediately after submission
- Reviews persist in MongoDB and appear on refresh

Sprint 3 -

User story 1 -

As a user, I want to add albums/etc to my profile or library so that I have my own diary of music

DoD

- Add/Remove buttons implemented and connected
- LibraryContext with optimistic updates added
- POST/DELETE /api/library working
- Library page /library implemented
- Hydration from backend and localStorage is verified
- Rollback behavior implemented on failure
- Code reviewed and merged
- No duplicates or missing albums

Acceptance Criteria

- User can save an album with one click
- Album appears instantly in Library page
- Removing album updates UI and database
- Library loads correctly for logged-in and logged-out users
- Saved albums show title, cover, artist list

User Story 2 -

As a user, I want to be able to view reviews from other users in a dedicated page when I click on a review in the social activity page.

DoD

- Dynamic route /review/[id] implemented
- API route fetches full review details
- Page displays album snapshot + reviewer info
- Code reviewed and integrated

Acceptance Criteria

- User can click a review and open full review page
- Page shows rating, text, album cover, artist, user info
- Page loads correctly for any review ID

User Story 4 -

As a user, I want to follow other users and see their activity in the dedicated feed page so that I can keep up to date with their reviews and activity.

DoD

- /feed page implemented with responsive UI
- Backend API route returns activity sorted by the “createdAt” timestamps
- Activity items correctly typed and mapped (review, rating, album snapshot)
- Clicking a feed item opens the correct review detail page
- Each feed card includes album cover, album title, artist names, reviewer username, and the review text snippet
- Code is reviewed and merged with no errors
- Tested the follow/unfollow + review creation
- No console errors when the feed is getting updated

Acceptance Criteria

The Social Feed is considered “done” when:

- A user can navigate to /feed and see activity from accounts they follow
- Feed shows:
    - Reviewer username
    - Album title + artists
    - Album cover image
    - Review rating & snippet
    - Timestamp (“2 hours ago”, etc.)
- Users can see if another user has reviewed an album, and this appears as a feed item
- Clicking a feed item takes the user to the full review page

User Story 5 -

As a user, I want to be able to see individual pages for each artist so that I can get more information about the artist

DoD

- The dynamic route /artist/[id] is created
- Spotify artist data is fetched and parsed
- Albums or top tracks rendered and displayed withTypeScript
- Code integrated into the repository
- Reviewed by the teammates

- Handle API request errors and display fallback UI if album data is unavailable

Acceptance Criteria

- User can click an artist name and open the artist page
- Page shows artist name, picture, albums
- Error + loading states work
- Navigation consistent with album flow

Sprint 4 -

User Story 1 -

As a user, I would like to share my profile and reviews with a link so that I can share it to anyone I want to outside of the website

DoD

- Share buttons are added to a user's profile as as well as their review cards
- Clipboard API integrated to be able to copy the URL
- There is a confirmation popup implemented
- Code reviewed and merged
- No console errors

Acceptance Criteria

- User can click the button "Share Profile", meaning their link is copied
- User can click the button "Share Review" and the review link is copied
- Users can click the share buttons in different views like the profile page, the feed page, etc.
- Confirmation message "Link Copied!" pops up
- Links open and work correctly when pasted

User Story 2 -

As a user, I would like to see an album of the week in my homepage so that I can get a recommendation.

DoD

- Billboard scraper implemented being able to retrieve the top 5 albums and label them from #1 to #5.
- Spotify enrichment pipeline has been tested manually
- The weekly albums that are fetched are stored in MongoDB
- /api/top-albums-week endpoint created
- Homepage section is built to display results

Acceptance Criteria

- Homepage shows "Top Albums This Week"

- The user can see the 5 Billboard-enriched albums
- The data updates consistently
- Album cards show the correct cover, title, artist, ranking