Sprint 3 Plan

Product Name: Groovo

Team Name: Groovo

Sprint Completion Date: 11/17/2025

Revision Number: Rev. 1.0

Revision Date: 11/03/2025

Sprint Goal

Finish all Sprint 3 user stories by expanding Groovo's social and discovery features. This includes implementing the album library system, the Top 5 favorites section, individual review pages, the follow/unfollow system, a dynamic social activity feed, and artist detail pages. The sprint also includes reviewing prior progress, carrying forward any essential unfinished tasks, and integrating frontend–backend functionality to improve navigation, interactivity, and user engagement across the app.

Task Listing (Organized by User Story)

User Story 1:

[8 pts] As a user, I want to add albums/etc to my profile or library so that I have my own diary of music (debating if we want to make it reviews only instead of saving albums to profile without review)

1. Task: create "Add to library" button (e.g, like button on Letterboxd) — (2pts)
    a. Add an "add to library" button or icon to album cards and album detail pages
    b. Use a reusable react component (e.g., <addToLibraryButton/>) that visually updates once an album is added (e.g, filled heart or checkmark).
    c. Create a route.ts that implements the GET and DELETE requests to albums from the library, and connect it to the button.
    d. Use consistent color and hover styles to match the existing theme
2. Task: Manage library state on the frontend— (1 pts)
    a. Use react state to temporarily store which albums the user has added to their library (when not logged in)
    b. Update the UI so the album appears immediately before waiting for a server response
    c. Display feedback response(e.g("Review added to your profile", "Added to your library!" message)
3. Task: Integrate with backend API for saving albums — (2 pts)

    a. Connect the frontend to the backend route(e.g post /api/) to save the albums data to the users profile
    b. Send the necessary album info(album ID, title, cover URL) and user ID with the request
    c. Handle success and error responses and update the UI based on the responses
4. Task: Create mongodb table referencing users and favorited albums — (1 pts)
    a. New table with user id, favorited album id
5. Task: Create a Library Page storing all the saved albums (2 pts)
    a. Create a UI to be able to display and see the saved albums in a library page.

Total for User Story 1: 8 pts

User Story 2:

[8 pts] As a user, I want a section in my profile showing my top 5 all-time favorite albums.

1. Task: Design and implement "Top 5 Albums" section on profile page (2 pts)
    a. Create a new UI section on the profile page titles "Top 5 Favorites"
    b. Display up to 5 album cards in a horizontal grid or carousel layout
    c. Each card should show the album cover, title, and artists(maybe use same design as the album card in the reviews album card?)
    d. Code the same styling design(consistent spacing, border radius, and hover styling from other album displays
2. Task: Owner-only edit mode(1 pts)
    a. Add Edit / Save / Cancel controls (visible only to the profile owner)
    b. Disable Save when no changes are pending
3. Task: Add albums via your reviewed albums (2 pts)
    a. Select albums to fill remaining slots (max 5) and prevent duplicates
4. Task: Fetch and display data (2 pts)
    a. GET /api/users/:userId/favorites/top5 and render in rank order
5. Task: Persist changes to backend (1 pts)
    a. PUT /api/me/favorites/top5 with array of { rank, albumId }
    b. Do optimistic UI update; rollback on error; show success/error toast

Total for User Story 2: 3 pts

User Story 3:

[4 pts] 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.

1. Task: Create Review Details Page Layout (1 pts)
    a. Set up a new route (e.g., /review/[id]) in next.js for displaying individual review pages

   b. Design a consistent layout with header/navigation, main content, and optional sidebar
   c. Include a placeholder sections for review content, album info, reviewer info, and coments/likes section
2. Task: Display Review and album information (2 pts)
   a. Fetch the selected review's data from the backend(using the review ID in the URL)
   b. Show the album cover, title, and artist name along with the review text rating, and review date.
   c. Add a small reviewer section with their username, profile picture, and link back to their profile
   d. Implement loading and error states while fetching the data
3. Task: Enable navigation from social activity page(1 pts)
   a. Make each review card in the social activity page clickable, routing the user to the corresponding /review/[id] page
   b. Pass the selected reviews id though the route parameter
   c. Add simple transition animations or visual feedback on a click

Total for User Story 3: 4 pts

User Story 4:

[11 pts] 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.

1. Task: Implement backend follow/unfollow API endpoints (1 pts)
   a. Implement a POST api follow and a DELETE api follow.
2. Task: Implement "Get User Activity" & "Get User Likes" Endpoints (1 pts)
   a. Use a GET request to return a list of another user's actions on Groovo, such as posting a review on an album.
   b. Use a GET request to and display a list of 5 albums they've liked. Might be randomized or the most recent likes they've posted.
3. Task: Create the Social Activity(feed) page layout (2 pts)
   a. Set up a new route(e.g., /feed) in next.js to serve as the social activity page
   b. Implement a consistent layout with the nav bar, feed content section
   c. Include placeholder sections for user activity cards(reviews, likes, new follows, etc.)
   d. Apply consistent design elements to match the rest of the app
   e. Handle logged-out state with a placeholder message
   f. Handle empty feed state(no followed users)
   g. Implement conditional rendering logic for tasks 3e and 3f
4. Task: Implement follow/unfollow functionality (UI side) (1 pts)

    a. Update the UI in real time when a user follows/unfollows someone(maybe using state management)
    b. Show a toast or alert confirming the action
    c. Handle disabled states and errors gracefully(eg, failed network requests)
5. Task: Display the social feed(following Users Activity) (3 pts)
    a. Integrate with the backend endpoint to fetch the activity feed (e.g. recent reviews, likes, or follows by followed users)
    b. Render a feed of activity cards, showing the user's profile picture and username, type of activity(e.g., “X reviewed Album name”), a short snippet of the review or link to the full post,and the timestamp
    c. Implement loading and error states for data fetching
6. Task: Add Navigation and Interactivity (1 pts)
    a. Make usernames clickable, routing to their perspective /profile/[id] pages
    b. Allow clicking a review snippet to open the full review page(/review/[id]
    c. Add hover effects and interactive icons(like/comment indicators)
7. Task: polishing UX (1 pts)
    a. Add subtle animations for appearing posts or button interactions
    b. Maintain consistent spacing, image sizes, and alignment
8. Task: Configure mongodb schema to support feed (1 pts)
    a. Create feed table referencing reviews based off user pulling in: creation time, and review details (like count, body, message thread)
    b. Need to create table or append table in MongoDB to support follower functionality.

Total for User Story 4:11 pts

User Story 5:

[5 pts] 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

1. Task: Set up dynamic artist routes(2 pts)
    a. Configure a dynamic Next.js route (e.g., /artist/[id]) to render a unique page for each album based on its database _id
    b. Create a new React component (e.g., ArtistPage.tsx) responsible for displaying album details
    c. Add typescript interfaces for album data(e.g, album, artist, track to ensure type safety
2. Task: Fetch and render artist data from spotify api (2 pts)
    a. Create an API route to fetch artist details from Spotify API.
    b. Display essential information from spotify api
    c. Use next.js fetch caching for loading or server-side rendering
    d. Handle API request errors and display fallback UI if album data is unavailable
3. Task: Add navigation and interactivity — (1 pts)

a. Ensure artist links from the albums/review page navigate to the correct artist route
b. Add a back to the previous page button to return to the previous page
c. Implement a loading indicator during data fetch

Total for User Story 5: 5 pts

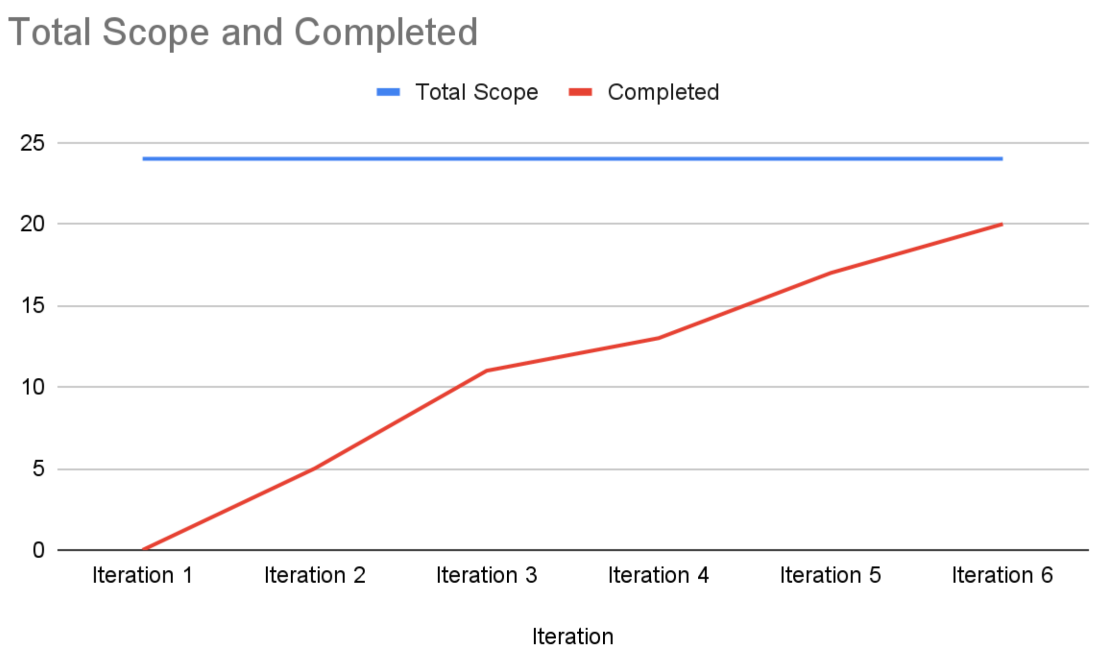
Team Roles

Team Member	Role(s)
Adam Gonzales	[Developer]
Christopher Bocanegra	[Developer/Product Owner]
Carter Wong	[Developer]
Srikar Chunduri	[Developer/ Scrum Master]
Shawn Dhillon	[Developer]

Initial Task Assignments

Team Member	User Story	Initial Task
Adam Gonzales	User Story # 2	Create a section in my profile showing my top 5 all-time favorite albums
[Carter Wong]	User Story #5	[Task Description]
[Christopher Bocanegra]	[User Story 3 t1 and t3/ User Story 4 task 4 / Title]	[working on the frontend side of user story 3 and 4 and will be leaving user 3 task 2 to someone else. Will be using mock data for now]
[Srikar Chunduri]	[User Story 1, and User Story 4 Tasks 2, User Story 5 Task 4/ Title]	[Task Description]
[Shawn Dhillon]	[User Story #4 task 5, 8/ User Story 1 task 3,4,]	[Backend and Backend integration]

Initial Burnup Chart



Scrum Meeting Schedule

Day	Time	Type / Attendees
[Monday]	[2:00 pm - 2:45 pm]	[TA/Tutor Visit]
[Monday]	[9 pm]	[Team Scrum]
[Wednesday]	[9 pm]	[Team Scrum or TA/Tutor Visit]