Sprint 2 Plan

Product Name: Groovo

Team Name: Groovo

Sprint Completion Date: 11/03/2025

Revision Number: Rev. 1.0

Revision Date: 10/20/2025

Sprint Goal

Deliver core profile functionality, album search features, and the foundational review system by completing remaining Sprint 1 work (User Stories 2 & 3) while beginning major Sprint 2 features. This includes implementing album search, building the profile page and editing tools, developing dynamic album routes, and laying the groundwork for reviews, ratings, and user interactions.

Task Listing (Organized by User Story)

User Story 1:

[5 pts] As a user, I want to search for music albums so that I can find and track the music I listen to

1. Task: Implement search bar component — (1 pts)
    a. Build a working search input field so that the text will appear in the search bar with input handling
2. Task: Integrate external music API — (2 pts)
    a. Connect to Spotify API to fetch albums/artists based on search queries. Handle authentication, rate limiting, and data parsing.
3. Task: Display search results — (2 pts)
    a. Create a results list that displays album/artist names, cover images, and release years.
    b. Add a button that allows the user to track/add to favorites for each item.

Total for User Story 1: 5pts

User Story 2:

[5 pts] 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

1. Task : Create the profile page layout: — (1 pt)
    a. Set up a new route (eg. /profile) and corresponding React page
    b. Add a consistent layout(eg, header, navigation bar, main content area) from the previous pages we have created(eg. color palette, spacing.
2. Task : Implement the User profile Information section — (1 pts)
    a. Show basic profile information such as the username, profile picture, and short bio
    b. Create a reusable UserHeader component for this section
    c. Handle cases where no profile picture or bio is available
3. Task: Fetch and display saved album/reviews — (2 pts)
    a. Integrate the frontend with the backend api endpoint
    b. Fetch and render a grid/list of saved albums and reviews(or mockup data for now?)
    c. Display album artwork, title, artist, and a short review snippet
    d. Implement loading and error states for data fetching
4. Task: Add Interactivity/polish— (1 pts)
    a. Ensure that clicking on an album navigates to its specific page
5. Task: Configure MongoDB collections to store collections for user reviews and comments— (2 hours)

Total for User Story 2: 5 pts

User Story 3:

[6 pts] 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.

1. Task: Implement a user bio field in MongoDB — (1 pts)
    a. Update the users collection to include a bio field.
2. Task: Build API profile update route — (1 pts)
    a. Create an authenticated API route that allows users to update their bio and basic profile information (e.g., bio, display name, etc.).
3. Task: Implement bio input and edit functionality— (2 pts)
    a. Add an “edit profile” button to the user’s profile page
    b. Create a form or modal allowing users to input or update their profile
    c. Validate input length(e.g., 0-200 characters) and prevent empty submissions
    d. Manage form state with TypeScript types(Controlled components in react/next.js?)

4. Task: Connect bio editing to backend API— (2 pts)
    a. Integrate frontend form with backend endpoints(eg, Put command)
    b. Send updated bio via JSON body and update the MongoDB user doc through Express
    c. Display Success/error messages to the user
    d. Ensure changes reflect immediately on the profile page (maybe react state or refetch user data)

Total for User Story 3: 6 pts

User Story 4:

[6 pts] 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

1. Task: Set up dynamic album routes — (1 pts)
    a. Configure a dynamic Next.js route (e.g., /albums/[id]) to render a unique page for each album based on its database _id
    b. Create a new React component (e.g., Albumpage.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 album data from spotify api — (2 pts)
    a. Create an API route to fetch album details (title, artist, cover art, genre, release year, etc.) from Spotify API.
    b. Display essential information from spotify api(e.g, title artists, release date, image,Tracktitle?)
    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 album links from the albums/review page navigate to the correct album route
    b. Add a back to the previous page button to return to the previous page
    c. Implement a loading indicator during data fetch

4. Task: Add /review interaction — (1 pts)
    a. Include a button that lets users review an album to their profile (no functionality yet)

Total for User Story 4: pts

User Story 5:

[10 pts] As a User, I want to write a review and rate an album/etc so that I can share my opinions and thoughts

1. Task: Create api review endpoint — (1 pts)
    a. Build a secure API route that allows authenticated users to submit a new review, including album ID, rating (1–5), and text content.
    b. Validate input and store in MongoDB.
2. Task: Create review input and rating UI — (2 pts)
    a. Add a review input section to the album detail page or a dedicated “write review” component
    b. Include a text area for the user to write their review and a star rating system
    c. Use controlled components in React (with TypeScript) to manage form state
3. Task: Implement form validation and state management — (1 pts)
    a. Validate that the review text is not empty and the rating is selected before submission
    b. Use React state or a global store to handle form data
    c. Display error or success messages below the form for user feedback
4. Task: Integrate frontend with backend API endpoint — (2 pts)
    a. Connect to the backend API endpoint(e.g, POST /api/…) to submit the user's review and rating
    b. Send the album ID, review text, rating value, and user ID
    c. Handle API responses to show success, loading, or error states
5. Task: Display user’s existing review — (2 pts)
    a. Show the user’s own review (if it exists) on the album page
    b. Fetch existing reviews from the backend and render them under the review form
    c. Show reviewer name, rating stars, and review text on an individual component/page

6. Task: Update average rating dynamically — (1 pts)
    a. Calculate and update the album's average rating each time a new review is added, edited, or deleted.
7. Task: Add a save to library and remove from library button - (1 pts)
    a. Implement a button that allows users to save an album to a library once they're logged in

Total for User Story 5: 10 pts

User Story 6:

[6 pts] As a user, I want to like or comment on reviews so that I can interact with people on the website

1. Task: Add like and comment UI elements to reviews — (1 pts)
    a. Update the review component to include icons or buttons for "like:
    b. Implement active states for visual feedback(color change or animation when clicked)
    c. Ensure responsive layout across desktop
2. Task: Handle frontend state for likes and comments — (1 pts)
    a. Use react state or context to manage whether a user has liked a review and how many likes exists
    b. Update the like count in real-time on click before server confirmation
3. Task: Integrate with backend API endpoints — (2 pts)
    a. Connect to backend routes (e.g, post /api/reviews/…) to like or unlike a review
    b. Handle success and error responses with visual feedback
4. Task: Display interaction data — (1 pts)
    a. Display total likes for each review

Total for User Story 6: 6 pts

Product Backlog:

[3 pts] As a user, I would like to create lists of albums so that I can have a small playlist of music that defines a theme like "best 90s shoegaze"

Team Roles

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

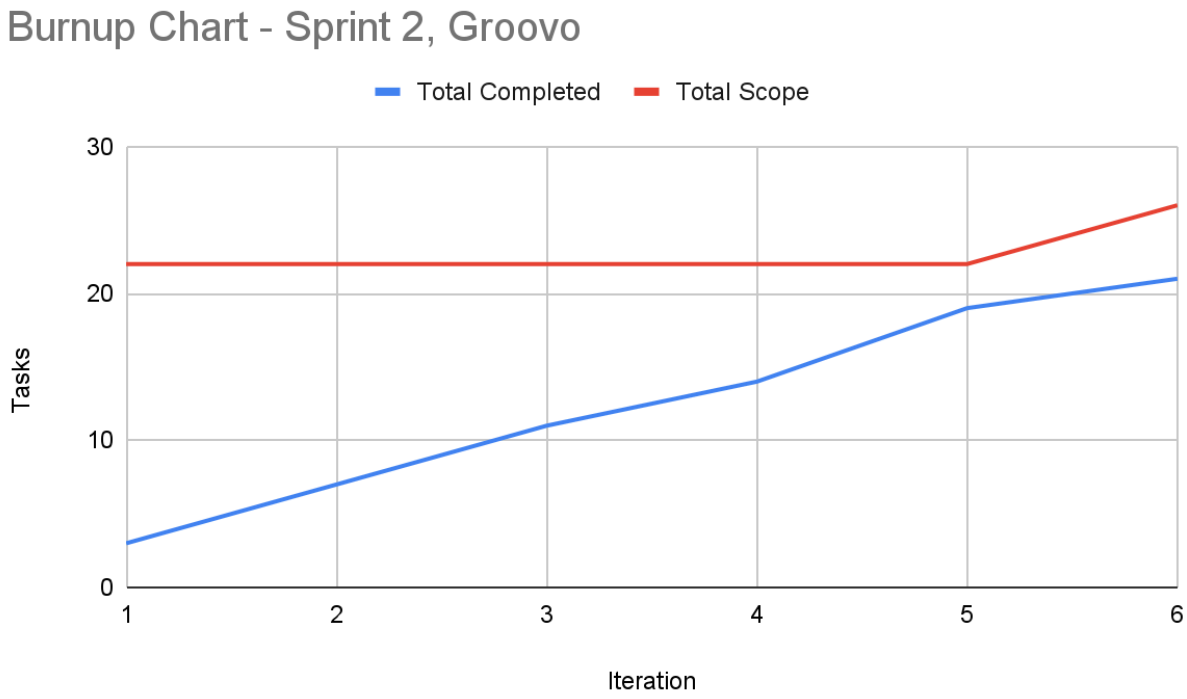
Initial Task Assignments

Team Member	User Story	Initial Task
Adam	[User Story 1 / Title]	[Task Description]
Chris	[User Story 2/3 4/ Title]	[Work on designing the UI for 2 & 3 and 4 and implementing it into typescript]
Shawn	[User Story #5 / Title]	Set up DB calls to write reviews to the albums collection
Carter	[User Story #3/ Title]	Add bio to user collection in DB; write/add bio and username
Srikar Chunduri	[User Story 4/5 / Title]	Work on navigation and interactivity, and being able to save an album.

Initial Burnup Chart

Attach or link the burnup chart here (e.g., link to chart in Google Sheets).

Label as: “Initial Burnup Chart – Sprint {Number}, {Project Name}”



Scrum Meeting Schedule

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