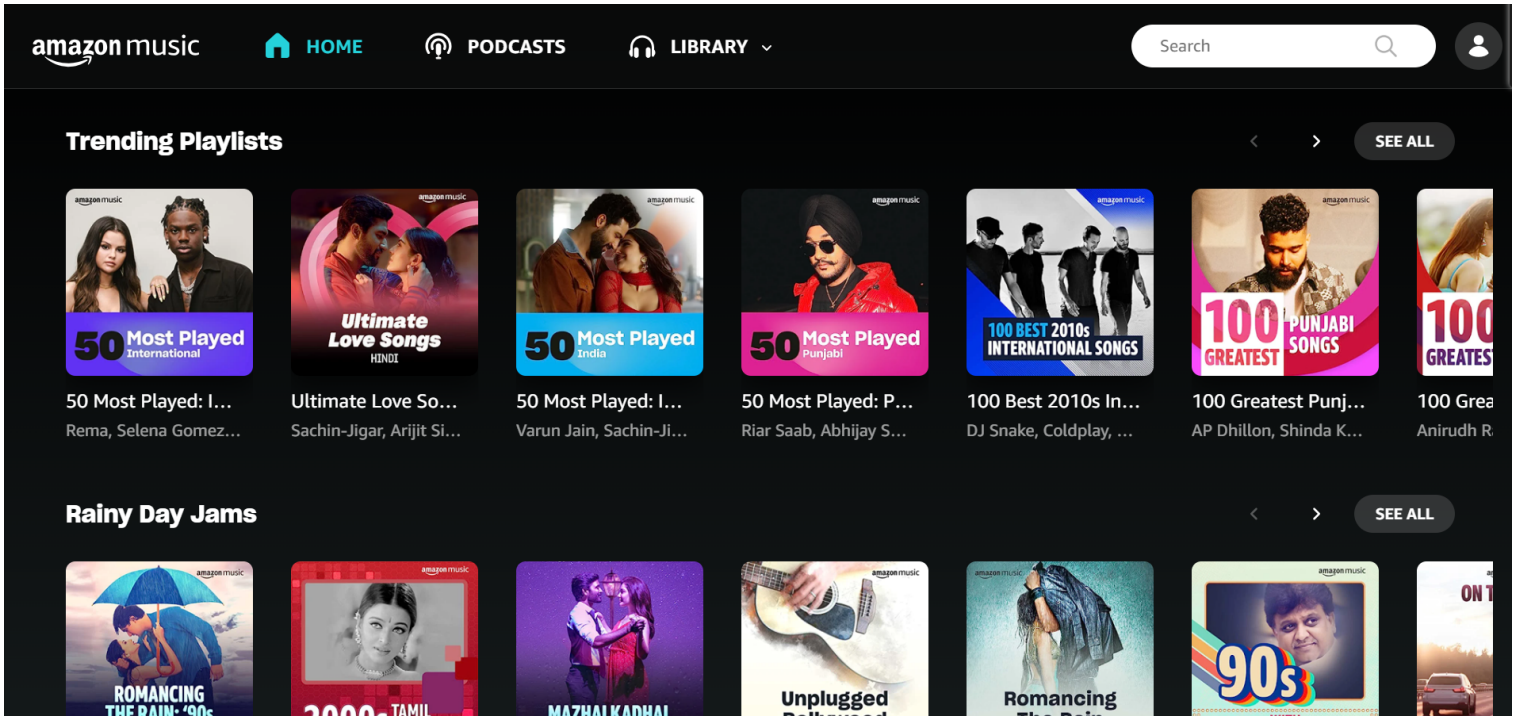
Project Objective

The project aims to create an ***Amazon Music (https://music.amazon.in/)*** clone application using HTML, CSS, JavaScript, and React. The app will replicate the core functionalities of Amazon Music, a music streaming platform, allowing users to browse, listen to, and discover their favorite songs and artists.



Project Context

👉 Amazon music is a popular music streaming platform that offers a vast library of songs, Albums, and personalized recommendations. The Amazon Music clone will offer users a similar experience, enabling them to enjoy their favorite music and discover new tracks and artists.

Tech stack Prerequisite: HTML, CSS, JavaScript, React

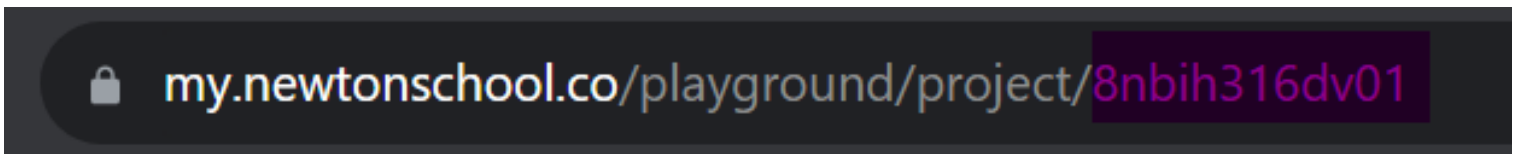
Note on Protected Routes

👉 Auth related APIs are protected (update user data, update password), So to use them, follow the instruction properly.
👉 Remember, for any protected routes, you'll need to include the JWT token in the header:

```
headers: {
  'Authorization': 'Bearer YOUR_JWT_TOKEN',
  'projectId': 'YOUR_PROJECT_ID'
}
```

👉 To know, what is JWT and how JWT works? Watch: What Is JWT and Why Should You Use JWT
👉 How to Use JWT Authorization using Postman? Watch: How to Use JWT Authorization

👉 Every API request must include a '**projectId**' in the headers. If it doesn't, the request will be rejected with an error.
--->Project ID here is the hash of the Playground URL
--->Example: *https://my.newtonschool.co/playground/project/**f104bi07c490***
--->Your Corresponding Project ID would be: **f104bi07c490**
---> Note: *These projectId will be unique for each question.*



Project Steps

Step 01: Set up the project and UI design

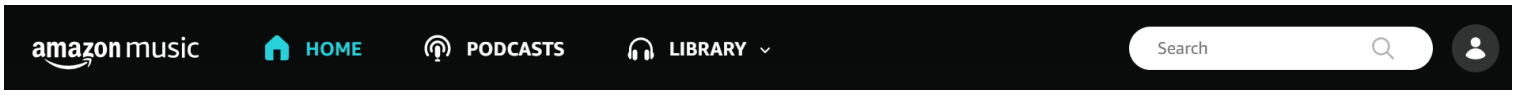
👉 Create a new React project using either **create-react-app** or **vite**.
👉 Organize the project directory structure, including subdirectories for components, styles, and other necessary files.
👉 Install required dependencies, such as **react-router**.
👉 The UI of this project requires huge amount of CSS to make the application look alike the real one.
👉 It is better to use external CSS libraries like **Material UI** (https://mui.com/material-ui/) which is a popular React component library that follows Google's Material Design guidelines. It provides a set of ready-made components for various UI elements which can help us to create complex UI in easier way by using the predefined CSS classes.
👉 The UI of the application must be responsive in nature.

Step 02: Navbar

👉 The website should have a navigation section.
👉 The navigation bar should have a **logo of the website**, which should point toward the home page of your website.
👉 Add **Home**, **Podcasts**, and **Library** nav links as well. Home with point towards the home page of the website, Podcasts will point towards the podcasts section and Library will point towards library section, which should display Favorite Song List.
👉 Add a **search bar**. This will allows users to easily search for songs, artists, or album based on keywords.
👉 There should be a **user icon** with the user's name, once clicked the button, it should display Login/ Logout, Subscriptions Options as a dropdown
--> If the user is not logged in point toward the Login page once clicked on the "Login".
--> If the user is logged in, point towards the Logout page once clicked on the

"Logout"

--> Once clicked on the Subscription, it redirect to the Subscription page details are mentioned below



Step 03: Home Page and Featured Music

👉 Create a home page displaying featured songs, albums, and artists.

👉 Fetch and display featured music from the database or API.

👉 Display albums by **artists**, **mood**, and **activity**.

👉 Display the albums according to the **Popular, Trending, Featured this Week, Recommended, and Podcasts.**

👉 Use the following APIs to fetch music data:

👉 ***Get list of music:***

```
fetch('https://academics.newtonschool.co/api/v1/music/song', {
  headers:
    'projectId': 'YOUR_PROJECT_ID'
  }
})
```

👉 ***Get list of albums:***

```
fetch('https://academics.newtonschool.co/api/v1/music/album', {
  headers: {
    'projectId': 'YOUR_PROJECT_ID'
  }
})
```

👉***Example with features (Filtering):***

This can be done by passing key-value pairs in the query string, with each key corresponding to a field in the database. For instance, you might make a request to GET /api/v1/music/song?filter={"field1":"value1","field2":"value2"} to get all data of type field1. You can use operators like $lte,$gte,$eq,$lt,$gt.

```
fetch('https://academics.newtonschool.co/api/v1/music/song?filter={"mood":"romantic"}',
{
  headers: {
    'projectId': 'YOUR_PROJECT_ID'
  }
})
--> https://academics.newtonschool.co/api/v1/music/song?filter={"mood":"romantic"}
```

👉***Sorting:***

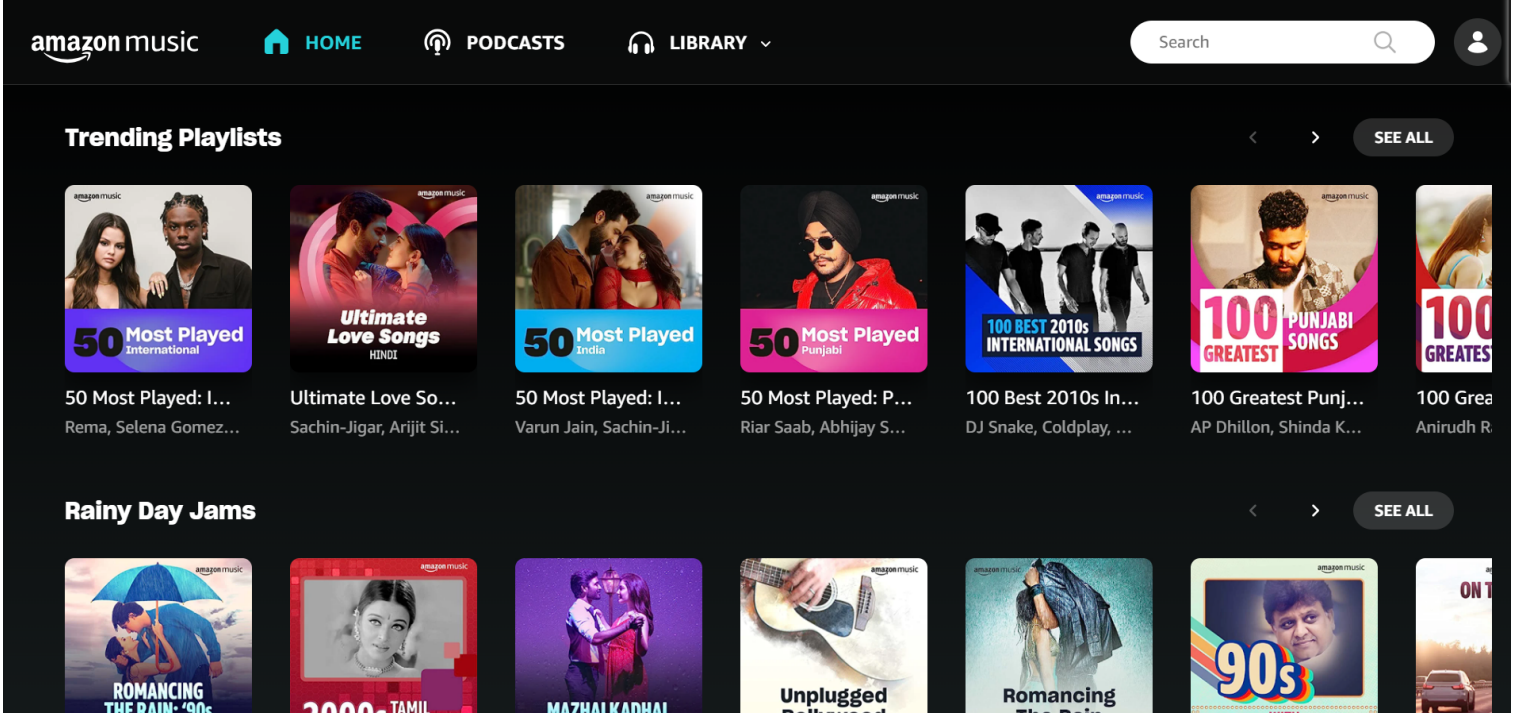
You can sort the results by one or more fields by specifying a sort query parameter with the fields to sort by. For example, GET /api/v1/music/song?sort={"field1":1,"field2":-1} would sort the results by field1 in ascending order (since it's value is 1) and field2 in descending order (since it's value is -1).

```
--> https://academics.newtonschool.co/api/v1/music/song?sort={"release":1}
```

👉***Pagination:***

Pagination can be handled with page and limit parameters. For instance, GET /api/v1/music/song?page=2&limit=10 would get the second page of results, with 10 results per page.

```
-->https://academics.newtonschool.co/api/v1/music/song?page=2&limit=10
```



Step 04: Music Player Component

👉 Design a reusable music player component displaying song details, album art, and playback controls (play, pause, skip, volume).

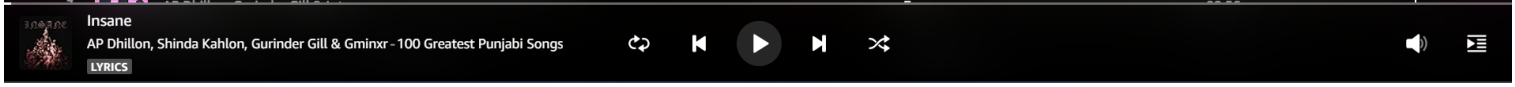
👉 Allow users to control the music playback and view progress.

👉 Position the music player controller at a fixed location on the screen, so it's easily accessible to users while they navigate the app. This will allow users to control their music playback without needing to scroll or navigate to a specific page.

👉 Display the Title, Album and Artist's Name in this component.

👉 While designing the component, you'll need details about the song that's currently playing. Use the Get music using id endpoint.

```
fetch('https://academics.newtonschool.co/api/v1/music/album/:id', {
  headers: {
    'projectId': 'YOUR_PROJECT_ID'
  }
})
```



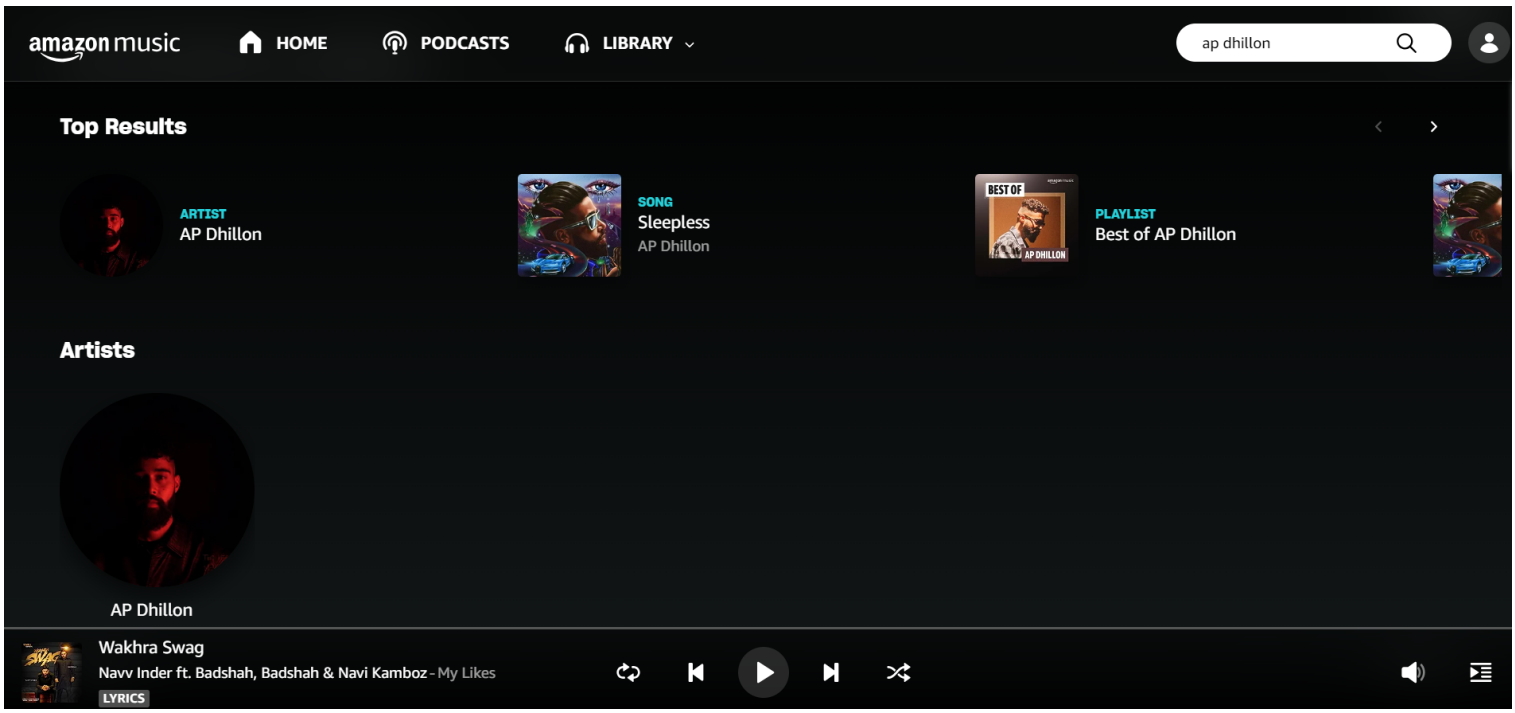
Step 05: Browse and Search Music

👉 Implement browsing functionality to explore songs, albums, and artists.

👉 Include a search bar for users to search for specific songs, albums, or artists.

👉 You can use the filtering feature from the API for this:

```
fetch('https://academics.newtonschool.co/api/v1/music/song?filter=
{"title":"search_term_here"}', {
  headers: {
    'projectId': 'YOUR_PROJECT_ID'
  }
})
```



Step 06: Song and Album Details Page

👉 Ensure that the detailed album page is displayed only when a user clicks on a specific album displayed on the home page. This way, the user can choose to explore more details about an album they're interested in without cluttering the main interface.

👉 Create pages to display detailed information about selected songs and albums.

👉 Show track lists, album art, and options to play songs from the selected album.

👉 The Album information should contain: **Title, Album, Artists, Number of Songs,** and **Duration of all the songs**.

👉 Album will have the songs of the corresponding album where the title and Artists will be displayed.

👉 Replace these "**+ ...**" buttons with Heart Symbol, and once the user clicks it should Add to Favorites.

👉 ***Get album using id:***

```
fetch('https://academics.newtonschool.co/api/v1/music/album/:id', {
  headers: {
    'projectId': 'YOUR_PROJECT_ID'
  }
})
```

👉 ***Get artist using id:***

```
fetch('https://academics.newtonschool.co/api/v1/music/artist/:id', {
  headers: {
    'projectId': 'YOUR_PROJECT_ID'
  }
})
```

Step 07: Register and Login

👉 Before the User can create a favorite songs list or personal album, check if the user is registered and login into the application. Since we are working with the Frontend only, use ***this auth route*** to Register the user, and then login the user. Ask the user for it's **first name, last name, email,** and **password**.

👉 If the user is not login, redirect the user toward the login page.

👉 Also, make sure users are logged in to access the music player and listen to songs. If a user attempts to play a song without logging in, display a modal prompting them to either log in if they have an account or register if they're new to the platform. This helps in encouraging user engagement and capturing potential new users. **[As mentioned in the below second image]**

👉 Add **reset password** functionality.

👉 **Login:**

```
fetch('https://academics.newtonschool.co/api/v1/user/login', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
```

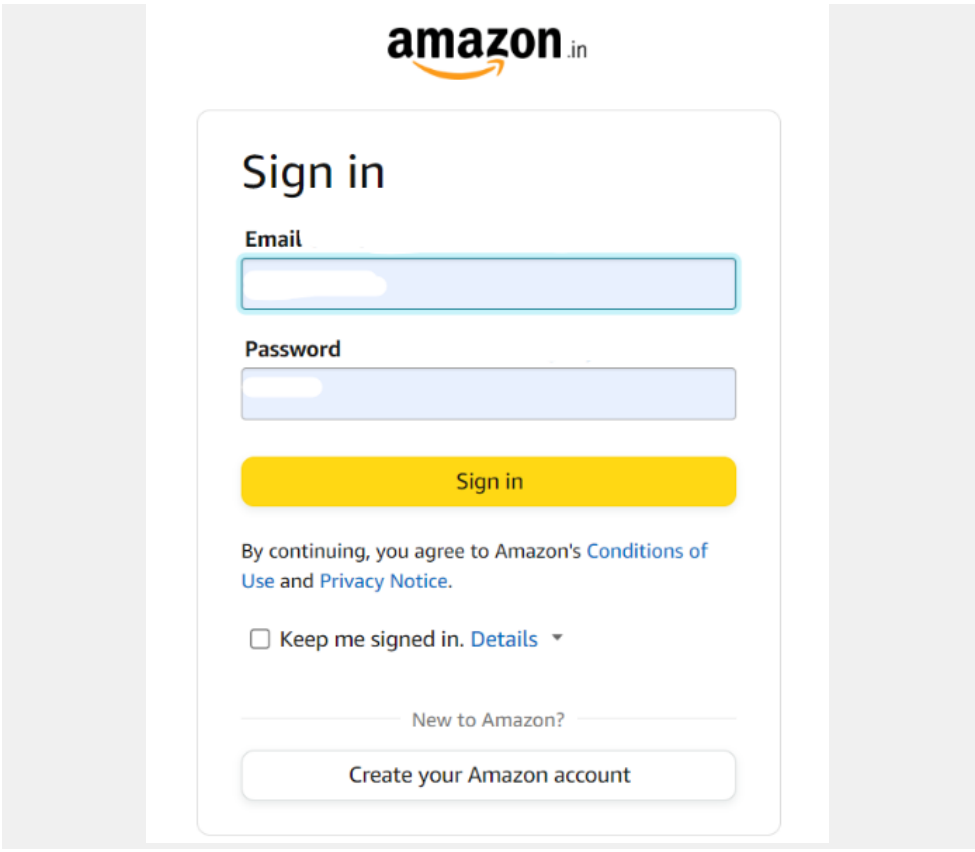
```
  },
  body: JSON.stringify({
    email: 'user_email',
    password: 'user_password',
    appType: 'music',
  })
})
```

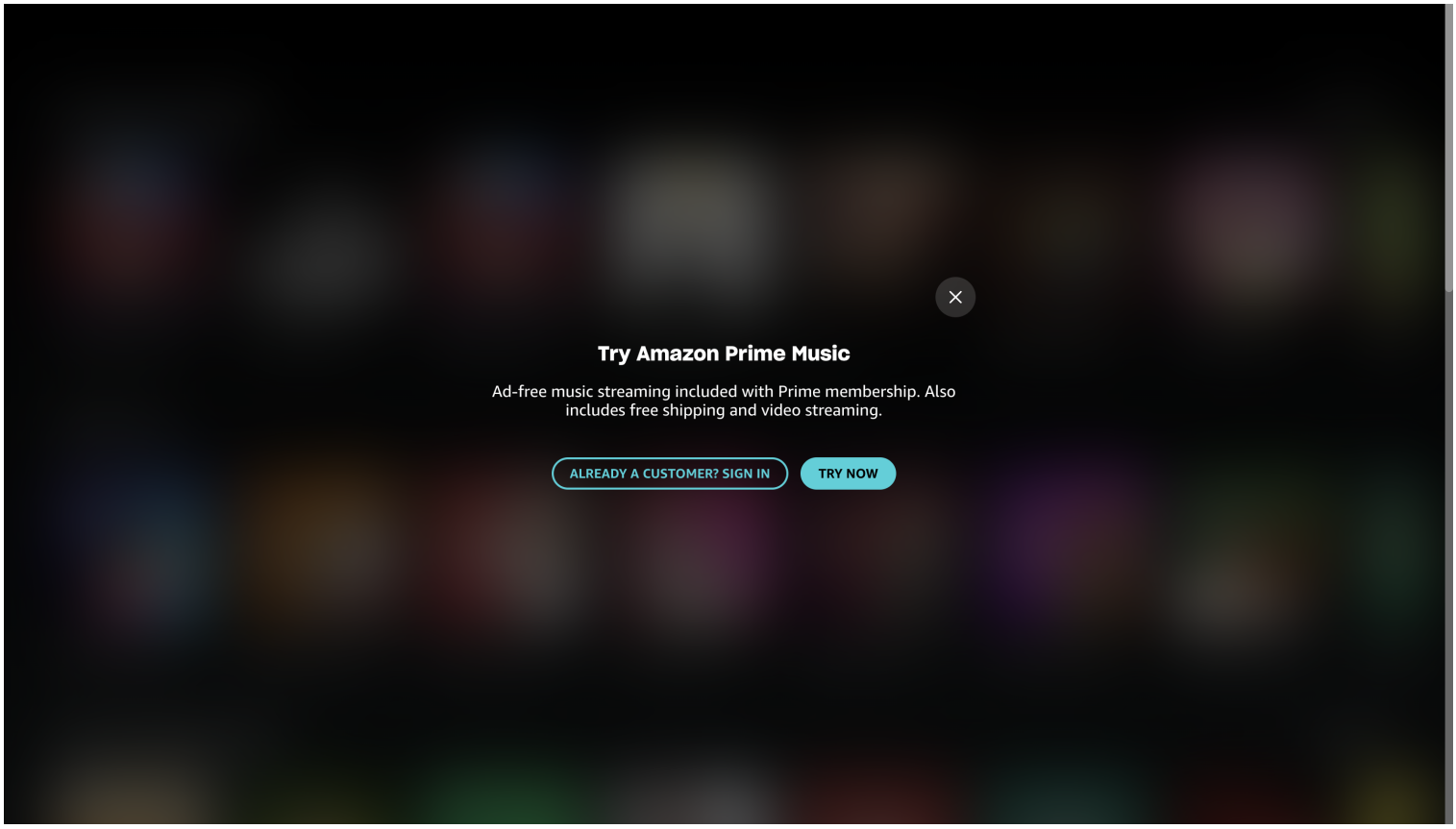
👉 **Signup:**

```
fetch('https://academics.newtonschool.co/api/v1/user/signup', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    name: 'user_name',
    email: 'user_email',
    password: 'user_password',
    appType: 'music',
  })
})
```

👉 **Update Password:**

```
fetch('https://academics.newtonschool.co/api/v1/user/updateMyPassword, {
  method: 'PATCH',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    name: 'user_name',
    email: 'user_email',
    passwordCurrent: 'current_password'
    password: 'user_new_password',
    appType: 'music',
  })
})
```





Step 08: Favorite Songs Page

👉 Once the user clicks on the Library it should redirect to this .

👉Provide an option for users to add and remove songs to their favorites list.

-->Add/Remove Show in Favorites (Protected Routes)

Method: PATCH

URL: https://academics.newtonschool.co/api/v1/music/favorites/like

headers: `{`

```
'Authorization': 'Bearer YOUR_JWT_TOKEN',
'projectID': 'Your Project ID'
}
```

BODY: { "songId" : songId }

Description: This API serves the dual purpose of adding and removing songs from favorites.

For example, when the API is triggered with an ID, it adds the corresponding show to the favorites. Conversely, if the API is subsequently invoked with the same ID, the show will be removed from the favorites.

-->Get My Favorite (Protected Routes)

Method: GET

URL: https://academics.newtonschool.co/api/v1/music/favorites/like

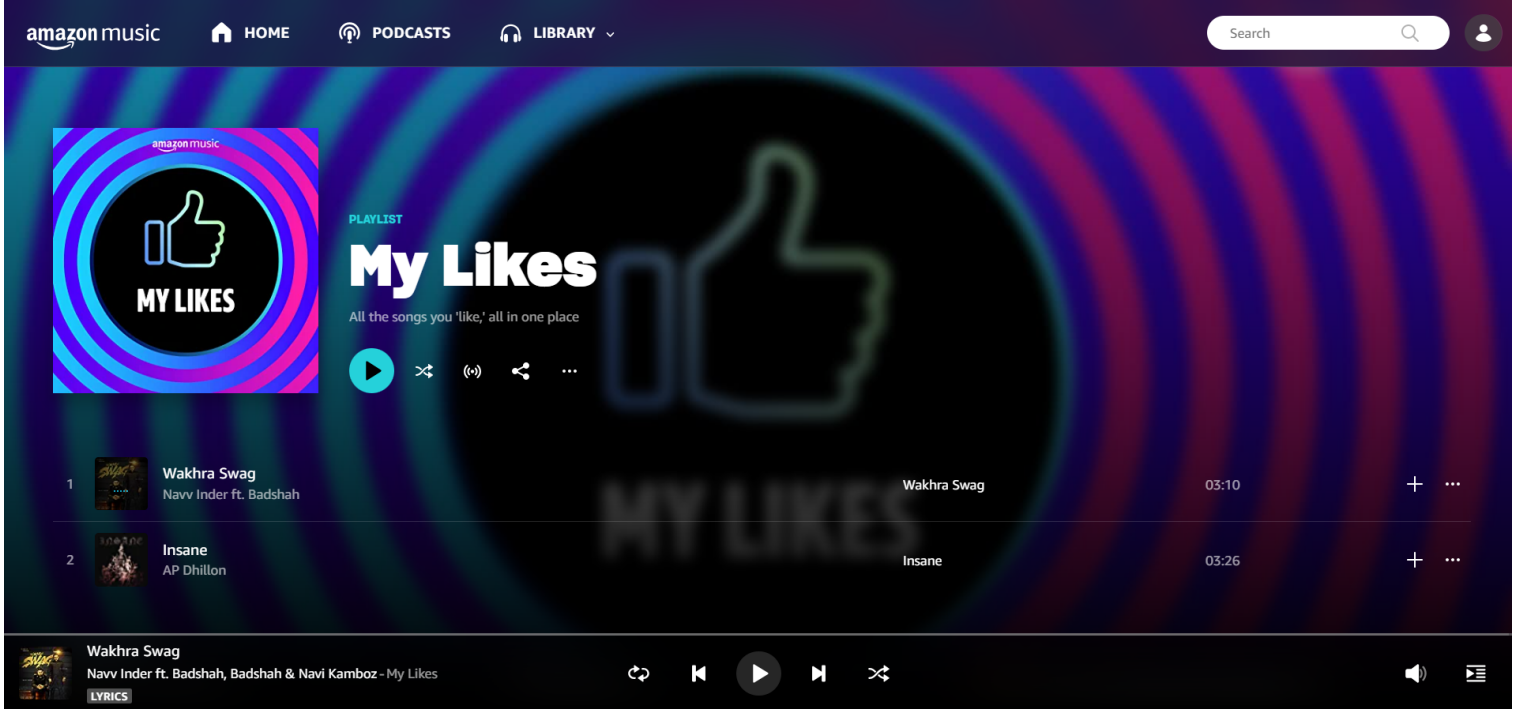
headers: `{`

```
'Authorization': 'Bearer YOUR_JWT_TOKEN',
'projectID': 'Your Project ID'
}
```

Description: This API is used to get shows added into the favorite

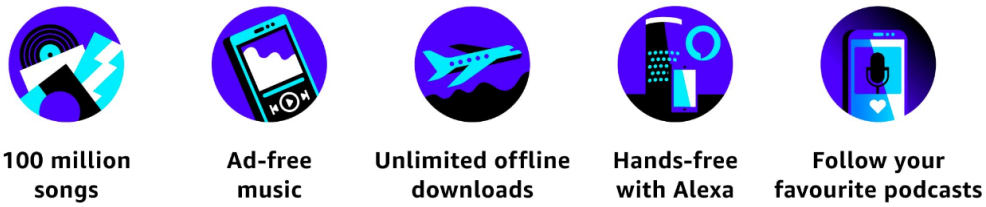
kindly refer to the link given below:

`https://drive.google.com/file/d/1dnmVLyT_zsb0ve-Z5zctILO2_n7TZICA/view?usp=sharing`



Step 09: Add Subscription Page

👉 Provide users with premium features or benefits for purchasing Amazon Music Subscription. The UI should looks like this.



Project Checkpoints

- 👉 Set up the project and UI design, Navbar
- 👉 Home Page and Featured Music, Music Player Component
- 👉 Browse and Search Music, Song and Album Details Page
- 👉 Register and Login, Favorite Songs
- 👉 Add Subscription Page

ON THIS PAGE

Project Steps