KrySpotify App -> Spotify 2.0. My replica to one of the world's most popular music streaming applications, Spotify. This is a modern web development of an improved version of Spotify, with a modern homepage, fully-edged music player, searching function, lyrics, song exploration features, popular music around, worldwide top charts, and much more. The development of KrySpotify App contains abundant technologies and skills of React functional components and their reusability, React file and folder structure, Redux Toolkit to manage the state of the application, Tailwind to make the app responsive on all devices, and RapidAPI to fetch data from various unlimited sources.

Functionalities/Demo:

- The music player application with a modern responsive homepage, with an exceptional design where the user can choose a genre and get the top songs you can see top charts, and top artists on the right side a fully fledged Spotify, it also allows the user to go to the previous song and next song, repeat, shuffle forward and change volume.
- The App also has a fully functional search and Pages where the user can explore the most popular songs in your country, trending artists and worldwide top charts, the user can click on a song title to navigate to a song's detail page where the user can see the song's official music video and the lyrics on the artist's name which shows you the songs related to that artist.

Developing Languages, Tools, and Techniques Needed:

JavaScript https://www.javascript.com/
Vscode 1.73.1 https://code.visualstudio.com/updates/v1_73
React Native JS https://reactjs.org/
RapidAPI Shazam Core API https://rapidapi.com/tipsters/api/shazam-core/
RapidAPI Client Vscode Extension
https://marketplace.visualstudio.com/items?itemName=RapidAPI.vscode-rapidapi-client
npm https://www.npmjs.com/
Tailwind CSS IntelliSense Vscode Extension https://tailwindcss.com/docs/editor-setup
Redux JS https://redux.js.org/
IP Geolocation API https://geo.ipify.org/

Prerequisites & Setup:

Subscribe to Shazam Core API on RapidAPI.
Install RapidAPI Client Vscode extension to Vscode App.
Install npm package.json dependencies files in Vscode terminal:

```
npm install
```

Run the web server:

```
npm dev run
```

Install a declaration file for module 'react-redux' in Vscode Terminal:

```
npm install --save-dev @types/react-redux
yarn add @types/react-redux --dev
```

Install axios for axios requirement:

```
npm install axios
```

Synchronous Developing Note:

Discover Section Creation:
Map the genre titles as a list in `Discover.jsx`:

```
 <div className="w-full flex justify-between items-center sm:flex-row
flex-col mt-4 mb-10">
                <h2 className="font-bold text-3xl text-white
text-left">Discover{genreTitle}</h2>
                <select
                    onChange={() => { }}
                    value=""
                    className="bg-black text-gray-300 p-3 text-sm
rounded-lg outline-none sm:mt-0 mt-5">
                    {genres.map((genre) => <option key={genre.value}
value={genre.value}>{genre.title}</option>)}
```

genre titles showed.PNG

Rapid API Music Fetching:

Fetch world charts by sending a new request in RapidAPI extension, 200 if the API is fetched.

world chart api fetched okay.PNG

Loader Components:

In `Loader.jsx`:

```
const Loader = ({ title }) => (
  <div className = "w-full flex justify-center items-center flex-col">
    <img src = {loader} alt = "loader" className = "w-32 h-32
object-contain" />
    <h1 className = "font-bold text-2xl text-white mt-2">{title ||
"Loading..."}</h1>
  </div>
);
```

loading songs when refreshing the page.PNG

Song Card Components:

To fetch songs album covers, in `SongCard.jsx`:

```
   <div className="flex flex-col w-[250px] p-4 bg-white/5 rounded-lg
cursor-pointer">
      <div className="relative w-full h-56 group">
        <div className={`absolute inset-0 justify-center items-center
       bg-black bg-opacity-50 group-hover:flex
        ${activeSong?.title === song.title ? 'flex bg-black
bg-opacity-70' : 'hidden'}`} >
        <PlayPause/>
      </div>
      <img alt = "song_img" src = {song.images?.coverart}/>
```

song cards cover images fetched.PNG

Display titles and subtitles:

```
        <p className="font-semibold text-lg text-white truncate">
          <Link to={`/songs/${song?.key}`}>
            {song.title}
```

```
          </Link>
        </p>
        <p className="text-sm truncate text-gray-300 mt-1">
          <Link to={song.artists ?
`/artists/${song?.artists[0]?.admid}` : 'top-artists'}>
            {song.subtitle}
```

song titles and subtitles artists showed.PNG

To make play and pause buttons show, in `PlayPause.jsx`:

```
 <FaPauseCircle
    size = {35}
    className = "text-gray-300"
    onClick = {handlePause}
  />
) : (
  <FaPlayCircle
    size = {35}
    className = "text-gray-300"
    onClick = {handlePlay}
  />
));
```

play and pause hover buttons showed.PNG

Play/Pause Functionalities:

To make play/pause buttons work, use dispatch state to trigger:

```
 const SongCard = ({ song, isPlaying, activeSong, i, data }) => {
  const dispatch = useDispatch();
  const handlePauseClick = () => {
    dispatch(playPause(false));
  };
  const handlePlayClick = () => {
    dispatch(setActiveSong({ song, data, i }));
    dispatch(playPause(true));
  };
```

song playing-play/pause buttons worked!.PNG

We can also adjust the volume as the songs are playing.

Sidebar Customization:

In `SideBar.jsx`:

```
    {links.map((item) => (
      <NavLink
        key={item.mame}
        to={item.to}
        className="flex flex-row justify-start items-center my-8
text-sm font-medium text-gray-400
```

```
        hover:text-cyan-400"
         onClick={() => handleClick && handleClick()}
         <item.icon className="w-6 h-6 mr-2" />
         {item.name}
       </NavLink>
     ))}
```

sidebar sections are customized.PNG

To make the sidebar responsive and right top toggles the left sidebar sections, import HiOutlineMenu and RiOutline methods with onClick:

```
   <div className="absolute md:hidden block top-6 right-3">
        {mobileMenuOpen ? (
          <RiCloseLine className="w-6 h-6 text-white mr-2" onClick={()
=> setMobileMenuOpen(false)} />
        ) : <HiOutlineMenu className="w-6 h-6 text-white mr-2"
onClick={() => setMobileMenuOpen(true)} />}
      </div>
      <div className={`absolute top-0 h-screen w-2/3 bg-gradient-to-tl
from-white/10
        to-[#483d8b] backdrop-blur-lg z-10 p-6 md:hidden
smooth-transition ${mobileMenuOpen ? 'left-0' :
          '-left-full'}`}>
        <img src={logo} alt="logo" className="w-full h-14
object-contain" />
        <NavLinks handleClick={() => setMobileMenuOpen(false)} />
```

Now when we minimized the webpage, its responsive and clicking right-top-three lines the left side-bar sections showed:

minimized page is responsive and sidebar sections showed when clicking right-top 3 lines.PNG

Top Play Section Swiper Customization:

Customize top artists horizontal swiper in `TopPlay.jsx`:

```
 <Swiper
          slidesPerView="auto"
          spaceBetween={15}
          freeMode
          centeredSlides
          centeredSlidesBounds
          modules={[FreeMode]}
          className="mt-4"
        >
          {topPlays?.map((song, i) => (
            <SwiperSlide
              key={song?.key}
              style={{ width: '25%', height: 'auto' }}
```

```
              className="shadow-lg rounded-full animate-slideright"
            >
              <Link to={`/artists/${song?.artists[0].adamid}`}>
                <img src={song?.images.background} alt="name"
className="rounded-full w-full object-cover" />
...
```

top artists swiper showed.PNG

Customize top charts music along with their corresponding artists:

```
const TopChartCard = ({ song, i }) => (
  <div className="w-full flex flex-row items-center hover:bg-[#4c426e]
py-2 p-4 rounded-lg cursor-pointer mb-2">
    <h3 className="fot-bold text-base text-white mr-3">{i + 1}</h3>
    <div className="flex-1 flex-row justify-between items-center">
      <img className="w-20 h-20 rounded-lg"
src={song?.images?.coverart} alt={song?.title} />
      <div className="flex-1 flex-col justify-center mx-3">
        <Link to={`/songs/${song.key}`}>
          <p className="text-xl font-bold
text-white">{song?.title}</p>
        </Link>
        <Link to={`/artists/${song?.artists[0].adamid}`}>
          <p className="text-ase text-gray-300
mt-1">{song?.subtitle}</p>
        </Link>
      </div>
    </div>
  </div>
);
```

top charts music showed.PNG

Now to enable the top chart songs to also play while clicking the play buttons:

```
 <PlayPause
      isPlaying={isPlaying}
      activeSong={activeSong}
      song={song}
      handlePause={handlePauseClick}
      handlePlay={handlePlayClick}
    />
  const topPlays = data?.slice(0, 5);
  const handlePauseClick = () => {
    dispatch(playPause(false));
  };
  const handlePlayClick = (song, i) => {
```

```
    dispatch(setActiveSong({ song, data, i }));
    dispatch(playPause(true));
  };
```

Now if the user click on top chart songs play buttons, the songs are also playable, and meanwhile, the clicked songs are displaying as playing status simultaneously at 3 platforms: top charts, bottom bar, and the album covers:

top charts songs playing simultaneously on 3 platforms.PNG

Song Details Page:

Fetch lyrics in `SongDetails.jsx`:

```
 {songData?.sections[1].type === 'LYRICS'
           ? songData?.sections[1].text.map((line, i) => (
         <p className="text-gray-400 text-base my-1">{line}</p>
             )) : <p className="text-gray-400 text-base my-1">Sorry,
no lyrics found!</p>}
```

Fetch song album cover images in `DetailsHeader.jsx`:

```
 const DetailsHeader = ({ artistId, artistData, songData }) => (
  <div className="relative w-full flex flex-col">
    <div className="w-full bg-gradient-to-1 from-transparent to-black
sm:h-48 h-28" />
    <div className="absolute inset-0 flex items-center">
      <img
        alt="art"
        src={artistId ? artistData?.artists[artistId].attributes?.
          artwork?.url.replace('{w}', '500').replace('{h}', '500')
          : songData?.images?.coverart}
```

song lyrics and header cover image fetched.PNG

Related Songs Swiper Section customization:

Fetch related songs to a specific song in `RelatedSong.jsx`:

```
    <div className="mt-6 w-full flex flex-col">
      {data?.map((song, i) => (
        <SongBar
          key={`${song.key}-${artistId}`}
          song={song}
          i={i}
          artistId={artistId}
          isPlaying={isPlaying}
          activeSong={activeSong}
          handlePauseClick={handlePauseClick}
          handlePlayClick={handlePlayClick}
...
```

related songs swiper displayed.PNG

Around You Section Customization:

Fetch country code and geography data from geo ipfy website, then fetch songs country codes in `AroundYou.jsx` to display Around You section:

```
  return (
        <div className="flex flex-col">
            <h2 className="font-bold text-3xl text-white text-left
mt-4 mb-10">Around You</h2>
            <div className="flex flex-wrap sm:justify-start
justify-center gap-8">
                {data?.map((song, i) => (
                    <SongCard
                        key={song.key}
                        song={song}
                        isPlaying={isPlaying}
                        activeSong={activeSong}
                        data={data}
                        i={i}
```

Around You section music are fetched.PNG

Top Charts Swiper:

Map top charts data in `TopCharts.jsx`:

```
<div className="flex flex-wrap sm:justify-start justify-center gap-8">
                {data?.map((song, i) => (
                    <SongCard
                        key={song.key}
                        song={song}
                        isPlaying={isPlaying}
                        activeSong={activeSong}
                        data={data}
                        i={i}
```

Top Charts section is implemented.PNG

Fetch top artists and their artists cards in `ArtistCards.jsx`:

```
 const ArtistCard = ({ track }) => {
  const navigate = useNavigate();
  return (
    <div className="flex flex-col w-[250px] p-4 bg-white/5
bg-opacity-80
    backdrop-blur-sm animate-slideup rounded-lg cursor-pointer">
      <img alt="artist" src={track?.images?.coverart}
        className="w-full h-56 rounded-lg" />
      <p className="mt-4 font-semibold text-lg text-white truncate">
        {track?.subtitle}</p>
```

top artists and their cards fetched.PNG

Search Functionality:

Customize Search bar in `SearchBar.jsx`:

```
 <FiSearch className="w-5 h-5 ml-4" />
      <input
        name="search-field"
        autoComplete="off"
        id="search-field"
        placeholder="Search"
        type="search"
        value=""
        onChange={() => {}}
        className = "flex-1 bg-transparent border-none outline-none
placeholder-gray-500 text-base text-white p-4"
```

search bar is customized.PNG

Fetch music by genre:

```
  const dispatch = useDispatch();
    const { activeSong, isPlaying, genreListId } = useSelector((state)
=> state.player);
    const { data, isFetching, error } =
useGetSongsByGenreQuery(genreListId || 'POP');
    if (isFetching) return <Loader title="Loading songs..." />;

    const genreTitle = genres.find(({ value }) => value ===
genreListId)?.title;
```

music of different genres displayed.PNG

Implement search by queries in `Search.jsx`:

```
 const Search = () => {
  const { searchTerm } = useParams();
  const { activeSong, isPlaying } = useSelector((state) =>
state.player);
  const { data, isFetching, error } =
useGetSongsBySearchQuery(searchTerm);
  const songs = data?.tracks?.hits?.map((song) => song.track);
  if (isFetching) return <Loader title="Loading top charts..." />;
  if (error) return <Error />;
```

search function works 1.PNG

search function works 2.PNG