KASRAT

1. APP.JS

```
import React from 'react';
import { Route, Routes } from 'react-router-dom';
import { Box } from '@mui/material';

import './App.css';
import ExerciseDetail from './pages/ExerciseDetail';
import Home from './pages/Home';
import Navbar from './components/Navbar';
import Footer from './components/Footer';
import AboutUs from './components/AboutUs';
const App = () => (
  <Box width="400px" sx={{ width: { xl: '1488px' } }} m="auto">
    <Navbar />
    <Routes>
      <Route path="/" element={<Home />} />
      <Route path="/exercise/:id" element={<ExerciseDetail />} />
    </Routes>
    <AboutUs/>
    <Footer />
  </Box>
);

export default App;
```

USING Route,Routes:
Adding React Router Components: The main Components of React Router are:

BrowserRouter: BrowserRouter is a router implementation that uses the HTML5 history API(pushState,
replaceState and the popstate event) to keep your UI in sync with the URL. It is the parent component that is used to store all of the other components.
Routes: It's a new component introduced in the v6 and a upgrade of the component. The main advantages of Routes over Switch are:
Relative s and s
Routes are chosen based on the best match instead of being traversed in order.
Route: Route is the conditionally shown component that renders some UI when its path matches the current URL.
Link: Link component is used to create links to different routes and implement navigation around the application. It works like HTML anchor tag.

The Box component serves as a wrapper component for most of the CSS utility needs.

All system properties are available via the sx prop. In addition, the sx prop allows you to specify any
other CSS rules you may need. Here's an example of how you can use it:

2.Utils
FetchData.js

```
export const exerciseOptions = {
    method: 'GET',
    headers: {
      'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com',
      'X-RapidAPI-Key': process.env.REACT_APP_RAPID_API_KEY,
    },
  };

  export const youtubeOptions = {
    method: 'GET',
    headers: {
      'X-RapidAPI-Host': 'youtube-search-and-download.p.rapidapi.com',
      'X-RapidAPI-Key': '69ded617damsh502c4b28756d143p1e7235jsn5781a1b85fbf',
    },
  };

  export const fetchData = async (url, options) => {
    const res = await fetch(url, options);
    const data = await res.json();

    return data;
  };
```

To put it simply, the Fetch API lets you talk with other APIs. It is a Web API that uses promises to make network requests over the HTTP/1.1 protocol.
You can make both same or cross-origin requests using the Fetch API.

Fetch API is included in all modern browsers, and you do not need to import any third-party library through yarn or npm.
You can use the fetch API using the fetch method. It takes multiple arguments, including the API endpoint's URL, i.e., the path of the resource you are interested in fetching.

Fetch API uses two objects, Request and Response. This Response object holds the data sent by the API.

Fetch sends the Request and returns a promise, which is resolved to the Response object when the request completes.
If the request fails, the promise is rejected.

A better and cleaner way of handling the promise is through the async/await keywords. You start by specifying the caller function as async and then use await to handle the promise.

3.

Home.js

```
import React, { useState } from 'react';
import { Box } from '@mui/material';

import Exercises from '../components/Exercises';
import SearchExercises from '../components/SearchExercises';
import Banner from '../components/Banner';

const Home = () => {
  const [exercises, setExercises] = useState([]);
// prilimary null
  const [bodyPart, setBodyPart] = useState('all');
// using this beacuse it will reflect all over application
  return (
    <Box>
      <Banner />
      <SearchExercises setExercises={setExercises} bodyPart={bodyPart}
setBodyPart={setBodyPart} />
      <Exercises setExercises={setExercises} exercises={exercises} bodyPart={bodyPart} />
    </Box>
  );
};

export default Home;
```

ExerciseDetail.js

```
import React, { useEffect, useState } from 'react';
import { useParams } from 'react-router-dom';
import { Box } from '@mui/material';

import { exerciseOptions, fetchData, youtubeOptions } from '../utils/fetchData';
```

```
import Detail from '../components/Detail';
import ExerciseVideos from '../components/ExerciseVideos';
import SimilarExercises from '../components/SimilarExercises';

const ExerciseDetail = () => {
  const [exerciseDetail, setExerciseDetail] = useState({});
  const [exerciseVideos, setExerciseVideos] = useState([]);
  const [targetMuscleExercises, setTargetMuscleExercises] = useState([]);
  const [equipmentExercises, setEquipmentExercises] = useState([]);
  const { id } = useParams();

  useEffect(() => {
    window.scrollTo({ top: 0, behavior: 'smooth' });

    const fetchExercisesData = async () => {
      const exerciseDbUrl = 'https://exercisedb.p.rapidapi.com';
      const youtubeSearchUrl = 'https://youtube-search-and-download.p.rapidapi.com';

      const exerciseDetailData = await fetchData(`${exerciseDbUrl}/exercises/exercise/${id}`,
exerciseOptions);
      setExerciseDetail(exerciseDetailData);

      const exerciseVideosData = await
fetchData(`${youtubeSearchUrl}/search?query=${exerciseDetailData.name} exercise`,
youtubeOptions);
      setExerciseVideos(exerciseVideosData.contents);

      const targetMuscleExercisesData = await
fetchData(`${exerciseDbUrl}/exercises/target/${exerciseDetailData.target}`,
exerciseOptions);
      setTargetMuscleExercises(targetMuscleExercisesData);

      const equimentExercisesData = await
fetchData(`${exerciseDbUrl}/exercises/equipment/${exerciseDetailData.equipment}`,
exerciseOptions);
      setEquipmentExercises(equimentExercisesData);
    };

    fetchExercisesData();
// calling function
  }, [id]);
//props id

  if (!exerciseDetail) return <div>No Data</div>;

  return (
    <Box sx={{ mt: { lg: '96px', xs: '60px' } }}>
      <Detail exerciseDetail={exerciseDetail} />
```

```
      <ExerciseVideos exerciseVideos={exerciseVideos} name={exerciseDetail.name} />
      <SimilarExercises targetMuscleExercises={targetMuscleExercises}
equipmentExercises={equipmentExercises} />
{/*Similar Exercises*/}
    </Box>
  );
};

export default ExerciseDetail;
```

useState:The useState() is a Hook that allows you to have state variables in functional components .
so basically useState is the ability to encapsulate local state in a functional component.
React has two types of components,
one is class components which are ES6 classes that extend from React and the other is functional components.
Class components a Component and can have state and lifecycle methods: class Message extends React.
The useState hook is a special function that takes the initial state as an argument and returns an array of two entries.
UseState encapsulate only singular value from the state, for multiple state need to have useState calls.

Syntax: The first element is the initial state and the second one is a function that is used for updating the state.

```
const [state, setState] = useState(initialstate)
```

useEffect:
The useEffect Hook allows you to perform side effects in your components.

Some examples of side effects are: fetching data, directly updating the DOM, and timers.

useEffect accepts two arguments. The second argument is optional.

useEffect(<function>, <dependency>)

Here a useEffect Hook that is dependent on a variable. If the count variable updates, the effect will run again:

```
useEffect(() => {
  //Runs on the first render
  //And any time any dependency value changes
}, [prop, state]);
```

empty array

useParams:
The react-router-dom package has useParams hooks that let you access the parameters of the current route.
```
  const { id } = useParams();
```

Components:
Banner.js:
Typography
The theme provides a set of type sizes that work well together, and also with the layout grid.

BodyPart.js
```
import React from 'react';
import { Stack, Typography } from '@mui/material';
import Icon from '../assests/images/cr1.png';

// typography to show name of bodypart
// onclick callback function to select and show

const BodyPart = ({ item, setBodyPart, bodyPart }) => (
  <Stack
    type="button"
    alignItems="center"
    justifyContent="center"
    className="bodyPart-card"
    sx={bodyPart === item ? { borderTop: '4px solid #FF2625', background: '#fff',
borderBottomLeftRadius: '20px', width: '270px', height: '282px', cursor: 'pointer', gap: '47px' }
: { background: '#fff', borderBottomLeftRadius: '25px', width: '280px', height: '282px', cursor:
'pointer', gap: '47px' }}
    onClick={() => {
      setBodyPart(item);
      window.scrollTo({ top: 1800, left: 100, behavior: 'smooth' });
    }}
  >
    <img src={Icon} alt="cardio" style={{ width: '40px', height: '40px' }} />

    <Typography fontSize="24px" fontWeight="bold" fontFamily="Alegreya" color="black"
textTransform="capitalize"> {item}</Typography>
  </Stack>
);

{/*The Stack component manages layout of immediate children along the vertical or
horizontal axis with optional spacing and/or dividers between each child.*/}
export default BodyPart;
```

Detail.js:

```
import React from 'react';
import { Typography, Stack, Button } from '@mui/material';

import BodyPartImage from '../assests/images/Logo.png';
import TargetImage from '../assests/icons/right-arrow.png';
import EquipmentImage from '../assests/images/Lgo1.png';


const Detail = ({ exerciseDetail }) => {
  const { bodyPart, gifUrl, name, target, equipment } = exerciseDetail;

//Array
  const extraDetail = [
    {
      icon: BodyPartImage,
      name: bodyPart,
    },
    {
      icon: TargetImage,
      name: target,
    },
    {
      icon: EquipmentImage,
      name: equipment,
    },
  ];

  return (
    <Stack gap="60px" sx={{ flexDirection: { lg: 'row' }, p: '20px', alignItems: 'center' }}>
      <img src={gifUrl} alt={name} loading="lazy" className="detail-image" />
      <Stack sx={{ gap: { lg: '35px', xs: '20px' } }}>
        <Typography sx={{ fontSize: { lg: '64px', xs: '30px' } }} fontWeight={700}
textTransform="capitalize">
          {name}
        </Typography>
        <Typography sx={{ fontSize: { lg: '24px', xs: '18px' } }} color="white">
         This  Exercises keep you mentally fit and  Physically strong .{' '}
          <span style={{ textTransform: 'capitalize' }}>{name}</span> but is one
          of the best <br /> exercises to target your {target} body-part. It will help you to improve
your{' '}
          <br /> mood and energy level . It Keeps You Healthy .
        </Typography>


        {extraDetail?.map((item) => (
```

```
          <Stack key={item.name} direction="row" gap="24px" alignItems="center">
            <Button sx={{ background: '#FFF2DB', borderRadius: '50%', width: '100px', height:
'100px' }}>
              <img src={item.icon} alt={bodyPart} style={{ width: '50px', height: '50px' }} />
            </Button>

            <Typography textTransform="capitalize" sx={{ fontSize: { lg: '30px', xs: '20px' } }}>
              {item.name}
            </Typography>
          </Stack>
        ))}
      </Stack>
    </Stack>
  );
};

export default Detail;
```

In React, the map () function is most commonly used for rendering a list of data to the DOM.
To use the map () function,
attach it to an array you want to iterate over.
The map () function expects a callback as the argument and executes it once for each element in the array.

ExerciseVideo.js:

```
import React from 'react';
import { Typography, Box, Stack } from '@mui/material';
import Loader from './Loader';
// props as exercisevideos name
  // slice to using to no. of videos show

const ExerciseVideos = ({ exerciseVideos, name }) => {
  if (!exerciseVideos.length) return <Loader />;

  return (
    <Box sx={{ marginTop: { lg: '203px', xs: '20px' } }} p="20px">
      <Typography sx={{ fontSize: { lg: '44px', xs: '25px' } }} fontWeight={700} color="#000"
mb="33px">
        Watch <span style={{ color: '#FF2625', textTransform: 'capitalize' }}>{name}</span>
exercise videos
      </Typography>
      <Stack sx={{ flexDirection: { lg: 'row' }, gap: { lg: '110px', xs: '0px' } }}
justifyContent="flex-start" flexWrap="wrap" alignItems="center">
        {exerciseVideos?.slice(0, 3)?.map((item, index) => (
          <a
```

```
          key={index}
          className="exercise-video"
          href={`https://www.youtube.com/watch?v=${item.video.videoId}`}
          target="_blank"
          rel="noreferrer"
        >
          <img style={{ borderTopLeftRadius: '20px' }} src={item.video.thumbnails[0].url}
alt={item.video.title} />
          <Box>
            <Typography sx={{ fontSize: { lg: '28px', xs: '18px' } }} fontWeight={600}
color="#000">
              {item.video.title}
            </Typography>
            <Typography fontSize="14px" color="#000">
              {item.video.channelName}
            </Typography>
          </Box>
        </a>
      ))}
    </Stack>
  </Box>
 );
};
// last box abut video details showing
export default ExerciseVideos;
```

Exercises.js:

```
import React, { useEffect, useState } from 'react';
import Pagination from '@mui/material/Pagination';
import { Box, Stack, Typography } from '@mui/material';

import { exerciseOptions, fetchData } from '../utils/fetchData';
import ExerciseCard from './ExerciseCard';
import Loader from './Loader';

const Exercises = ({ exercises, setExercises, bodyPart }) => {
  const [currentPage, setCurrentPage] = useState(1);
  const [exercisesPerPage] = useState(6);
// call back function
  useEffect(() => {
    const fetchExercisesData = async () => {
      let exercisesData = [];

      if (bodyPart === 'all') {
```

```
      exercisesData = await fetchData('https://exercisedb.p.rapidapi.com/exercises',
exerciseOptions);
    } else {
      exercisesData = await
fetchData(`https://exercisedb.p.rapidapi.com/exercises/bodyPart/${bodyPart}`,
exerciseOptions);
    }

    setExercises(exercisesData);
  };

  fetchExercisesData();
 }, [bodyPart]);

//dependency array
 // Pagination
 const indexOfLastExercise = currentPage * exercisesPerPage;
 const indexOfFirstExercise = indexOfLastExercise - exercisesPerPage;
 const currentExercises = exercises.slice(indexOfFirstExercise, indexOfLastExercise);

 const paginate = (event, value) => {
  setCurrentPage(value);

  window.scrollTo({ top: 1800, behavior: 'smooth' });
 };

 if (!currentExercises.length) return <Loader />;
// using Loader here
//Length greater 8 then pagination
 return (
  <Box id="exercises" sx={{ mt: { lg: '109px' } }} mt="50px" p="20px">
    <Typography variant="h4" fontWeight="bold" sx={{ fontSize: { lg: '44px', xs: '30px' } }}
mb="46px">Showing Results</Typography>
    <Stack direction="row" sx={{ gap: { lg: '107px', xs: '50px' } }} flexWrap="wrap"
justifyContent="center">
      {currentExercises.map((exercise, idx) => (
       <ExerciseCard key={idx} exercise={exercise} />
      ))}
    </Stack>

    <Stack sx={{ mt: { lg: '114px', xs: '70px' } }} alignItems="center">
      {exercises.length > 8 && (
       <Pagination
        color="standard"
        shape="rounded"
        defaultPage={1}
        count={Math.ceil(exercises.length / exercisesPerPage)}
        page={currentPage}
```

```
            onChange={paginate}
            size="large"
          />
        )}
      </Stack>
    </Box>
  );
};


export default Exercises;
```

Pagination is a technique using the data that is fetched from the server in a form of groups. Such groups are created according to a particular value and each group has that number of records.

A callback is a function passed as an argument to another function. This technique allows a function to call another function.
A callback function can run after another function has finished

HorizontalScrollbar.js:

```
import React, { useContext } from 'react';
import { ScrollMenu, VisibilityContext } from 'react-horizontal-scrolling-menu';
import { Box, Typography } from '@mui/material';

import ExerciseCard from './ExerciseCard';
import BodyPart from './BodyPart';
import RightArrowIcon from '../assests/icons/ar.png';
import LeftArrowIcon from '../assests/icons/al.png';
//Looping through exercise card like (pic)
const LeftArrow = () => {
  const { scrollPrev } = useContext(VisibilityContext);

  return (
    <Typography onClick={() => scrollPrev()} className="right-arrow">
      <img src={LeftArrowIcon} alt="right-arrow" />
    </Typography>
  );
};

const RightArrow = () => {
  const { scrollNext } = useContext(VisibilityContext);

  return (
```

```
    <Typography onClick={() => scrollNext()} className="left-arrow">
      <img src={RightArrowIcon} alt="right-arrow" />
    </Typography>
  );
};

const HorizontalScrollbar = ({ data, bodyParts, setBodyPart, bodyPart }) => (
  <ScrollMenu  RightArrow={RightArrow}  LeftArrow={LeftArrow} >
    {data.map((item) => (
      <Box
        key={item.id || item}
        itemId={item.id || item}
        title={item.id || item}
        m="0 10px"
      >
        {bodyParts ? <BodyPart item={item} setBodyPart={setBodyPart} bodyPart={bodyPart}
/> : <ExerciseCard exercise={item} /> }
      </Box>
    ))}

  </ScrollMenu>
);

export default HorizontalScrollbar;
```

React Context is a way to manage state globally.

It can be used together with the useState Hook to share state between deeply nested components more easily than with useState alone.

Loader.js:

```
const Loader = () => (
  <Stack direction="row" justifyContent="center" alignItems="center" width="100%">
    <InfinitySpin color="blue" />
  </Stack>
);
```

SearchExercises.js:

```
import React, { useEffect, useState } from 'react';
import { Box, Button, Stack, TextField, Typography } from '@mui/material';

import { exerciseOptions, fetchData } from '../utils/fetchData';
```

```
import HorizontalScrollbar from './HorizontalScrollbar';

const SearchExercises = ({ setExercises, bodyPart, setBodyPart }) => {
  const [search, setSearch] = useState('');
  const [bodyParts, setBodyParts] = useState([]);

  useEffect(() => {
    const fetchExercisesData = async () => {
      const bodyPartsData = await
fetchData('https://exercisedb.p.rapidapi.com/exercises/bodyPartList', exerciseOptions);

      setBodyParts(['all', ...bodyPartsData]);
    };

    fetchExercisesData();
  }, []);

  // call back-function
  const handleSearch = async () => {
    if (search) {
      const exercisesData = await fetchData('https://exercisedb.p.rapidapi.com/exercises',
exerciseOptions);

      const searchedExercises = exercisesData.filter(
        (item) => item.name.toLowerCase().includes(search)
              || item.target.toLowerCase().includes(search)
              || item.equipment.toLowerCase().includes(search)

              || item.bodyPart.toLowerCase().includes(search),
      );

      window.scrollTo({ top: 1800, left: 100, behavior: 'smooth' });

      setSearch('');
      setExercises(searchedExercises);
    }
  };

  // last box to show body-parts type for exercise
  return (
    <Stack alignItems="center" mt="37px" justifyContent="center" p="20px">
      <Typography fontWeight={700} sx={{ fontSize: { lg: '44px', xs: '30px' } }} mb="49px"
textAlign="center">
        Awesome Exercises You <br /> Should Know
      </Typography>
      <Box position="relative" mb="72px">
        <TextField
          height="76px"
```

```
          sx={{ input: { fontWeight: '700', border: 'none', borderRadius: '4px' }, width: { lg:
'1170px', xs: '350px' }, backgroundColor: '#fff', borderRadius: '40px' }}
          value={search}
          onChange={(e) => setSearch(e.target.value.toLowerCase())}
          placeholder="Search Exercises"
          type="text"
        />
        <Button className="search-btn" sx={{ bgcolor: "#b5e7a0", color: '#fff', textTransform:
'none', width: { lg: '173px', xs: '80px' }, height: '56px', position: 'absolute', right: '0px', fontSize:
{ lg: '20px', xs: '14px' } }} onClick={handleSearch}>
          Search
        </Button>
      </Box>

      <Box sx={{ position: 'relative', width: '100%', p: '20px' }}>
        <HorizontalScrollbar data={bodyParts} bodyParts setBodyPart={setBodyPart}
bodyPart={bodyPart} />
      </Box>
    </Stack>
  );
};

export default SearchExercises;
```

What is async await?
Use of async and await enables the use of ordinary try / catch blocks around asynchronous code.
Note: The await keyword is only valid inside async functions within regular JavaScript code.

What is difference between async and await?
Image result for async await
In using async and await, async is prepended when returning a promise, await is prepended when calling a promise.
try and catch are also used to get the rejection value of an async function.

SimilarExercises.js:

```
import React from 'react';
import { Typography, Box, Stack } from '@mui/material';

import HorizontalScrollbar from './HorizontalScrollbar';
import Loader from './Loader';
// using Loader
const SimilarExercises = ({ targetMuscleExercises, equipmentExercises }) => (
    // props value
  <Box sx={{ mt: { lg: '100px', xs: '0px' } }}>
```

```
    <Typography sx={{ fontSize: { lg: '44px', xs: '25px' }, ml: '20px' }} fontWeight={700}
color="#000" mb="33px">
      Similar <span style={{ color: '#FF2625', textTransform: 'capitalize' }}>Target
Muscle</span> exercises
    </Typography>
    <Stack direction="row" sx={{ p: 2, position: 'relative' }}>
      {targetMuscleExercises.length !== 0 ? <HorizontalScrollbar
data={targetMuscleExercises} /> : <Loader />}
    </Stack>
    <Typography sx={{ fontSize: { lg: '44px', xs: '25px' }, ml: '20px', mt: { lg: '100px', xs: '60px' }
}} fontWeight={700} color="#000" mb="33px">
      Similar <span style={{ color: '#FF2625', textTransform: 'capitalize' }}>Equipment</span>
exercises
    </Typography>
    <Stack direction="row" sx={{ p: 2, position: 'relative' }}>
      {equipmentExercises.length !== 0 ? <HorizontalScrollbar data={equipmentExercises} /> :
<Loader />}
    </Stack>
  </Box>
);
// using Loader
export default SimilarExercises;
```

Api End point:

```
export const exerciseOptions = {
    method: 'GET',
    headers: {
      'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com',
      'X-RapidAPI-Key': process.env.REACT_APP_RAPID_API_KEY,
    },
  };

  export const youtubeOptions = {
    method: 'GET',
    headers: {
      'X-RapidAPI-Host': 'youtube-search-and-download.p.rapidapi.com',
      'X-RapidAPI-Key': '69ded617damsh502c4b28756d143p1e7235jsn5781a1b85fbf',
    },
  };

  export const fetchData = async (url, options) => {
    const res = await fetch(url, options);
    const data = await res.json();

    return data;
  };
```

Axios is a Promise-based HTTP client for the browser and Node. break down this definition to understand what Axios does.

An HTTP GET request is used to request a specified resource from a server. These requests do not contain any payload with them, i.e., the request doesn't have any content. Axios GET is the method to make HTTP GET requests using the Axios library.