**Отчет**

Подготовил Тілесбек Бейбарыс

2 курс, русское отделение

Специальность: Software Engineering

**Step-1**

import logo from './logo.svg';

import './App.css';

import { Route, Routes } from 'react-router-dom';

import { Box } from '@mui/material'

import ExerciseDetail from './pages/ExercisDetail';

import Home from './pages/Home';

import Navbar from './components/Navbar';

import Footer from './components/Footer';

const App = () => {

  return (

    <Box width="400px">

      Navbar

      <Routes>

        <Route path="/" element={Home} />

        <Route path="/exercise/:id" element={ExerciseDetail} />

      </Routes>

      <Footer />

    </Box>

  )

}

export default App;

Это основной компонент приложения, который определяет структуру интерфейса с помощью компонентов MUI (Material-UI), React Router для управления маршрутами и собственных компонентов Navbar и Footer. При переходе по различным маршрутам отображаются соответствующие компоненты (например, Home или ExerciseDetail).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Также мы создали страницы и компоненты

import React from "react";

const Home = () => {

    return (

        <div>Home</div>

    )

}

export default Home

Также добавили простые функциональные компоненты ‘Home’, ‘ExerciseDetail’, ‘Navbar’, ‘Footer’

Изображение выглядит как снимок экрана, Прямоугольник

Автоматически созданное описание

**Step-2**

import React, { useState} from "react";

import { Box } from "@mui/material";

import HeroBanner from "../components/HeroBanner";

import SearchExercises from "../components/SearchExercises";

import Exercises from "../components/Exercises";

const Home = () => {

    return (

        <Box>

            <HeroBanner/>

            <SearchExercises />

            <Exercises />

        </Box>

    )

}

export default Home

Здесь мы добавили новые компоненты **HeroBanner**, **SearchExercises** и **Exercises** внутри контейнера **Box**

import logo from './logo.svg';

import './App.css';

import { Route, Routes } from 'react-router-dom';

import { Box } from '@mui/material'

import ExerciseDetail from './pages/ExercisDetail';

import Home from './pages/Home';

import Navbar from './components/Navbar';

import Footer from './components/Footer';

const App = () => {

  return (

    <Box width="400px" sx={{ width: { xl: '1488p' }}} m="auto">

      <Navbar />

      <Routes>

        <Route path="/" element={Home} />

        <Route path="/exercise/:id" element={ExerciseDetail} />

      </Routes>

      <Footer />

    </Box>

  )

}

export default App;

Импортировали компонента Box из библиотеки Material-UI (MUI). Box - это контейнер, который обеспечивает гибкую систему стилей. Использование компонента Box с указанием ширины 400 пикселей (width="400px") и стилей для больших экранов (используя MUI's sx prop). m="auto" центрирует Box по горизонтали. Определение маршрутов с использованием компонента Routes. Внутри него определены маршруты для главной страницы (/) и страницы упражнения (/exercise/:id).

import React from "react";

import { Link } from "react-router-dom";

import { Stack } from "@mui/material";

import Logo from '../assets/images/Logo.png'

const Navbar = () => {

    <Stack direction="row" justifyContent="space-around" sx={{ gap: { sm: '123px', xs: '40px' }, mt: { sm: '32px', xs: '20px' }, justifyContent: 'none' }} px="20px">

        <Link to="/">

          <img src={Logo} alt="logo" style={{ width: '48px', height: '48px', margin: '0px 20px' }} />

        </Link>

        <Stack

        direction="row"

        gap="40px"

        fontFamily="Alegreya"

        fontSize="24px"

        alignItems="flex-end"

        >

            <Link to="/" style={{ textDecoration: 'none', color: '#3A1212', borderBottom: '3px solid #FF2625' }}>Home</Link>

            <a href="#exercises" style={{ textDecoration: 'none', color: '#3A1212' }}>Exercises</a>

        </Stack>

    </Stack>

}

export default Navbar

Наш компонент использует Material-UI компоненты Stack для создания разметки в виде горизонтального стека, и Link из react-router-dom для создания навигационных ссылок.

Link to="/">: Это компонент Link, который создает ссылку на главную страницу вашего приложения. При нажатии на эту ссылку пользователь будет перенаправлен на корневой путь (/).

<a href="#exercises">: Это обычная HTML-ссылка с якорем (#exercises). Вероятно, она предназначена для прокрутки к секции с упражнениями (если у вас есть секция с id "exercises" в вашем коде).

Стили: Используются стили Material-UI (sx prop), а также встроенные стили в виде объектов в атрибутах style. Например, style={{ width: '48px', height: '48px', margin: '0px 20px' }} устанавливает ширину, высоту и отступы для изображения логотипа.

fontFamily="Alegreya" fontSize="24px": Установка шрифта и размера шрифта для текста внутри Stack.

**Step-3**

import React from 'react';

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

import HeroBannerImage from '../assets/images/banner.png';

const HeroBanner = () => (

  <Box sx={{ mt: { lg: '212px', xs: '70px' }, ml: { sm: '50px' } }} position="relative" p="20px">

    <Typography color="#FF2625" fontWeight="600" fontSize="26px">Fitness Club</Typography>

    <Typography fontWeight={700} sx={{ fontSize: { lg: '44px', xs: '40px' } }} mb="23px" mt="30px">

      Sweat, Smile <br />

      And Repeat

    </Typography>

    <Typography fontSize="22px" fontFamily="Alegreya" lineHeight="35px">

      Check out the most effective exercises personalized to you

    </Typography>

    <Stack>

      <a href="#exercises" style={{ marginTop: '45px', textDecoration: 'none', width: '200px', textAlign: 'center', background: '#FF2625', padding: '14px', fontSize: '22px', textTransform: 'none', color: 'white', borderRadius: '4px' }}>Explore Exercises</a>

    </Stack>

    <Typography fontWeight={600} color="#FF2625" sx={{ opacity: '0.1', display: { lg: 'block', xs: 'none' }, fontSize: '200px' }}>

      Exercise

    </Typography>

    <img src={HeroBannerImage} alt="hero-banner" className="hero-banner-img" />

  </Box>

);

export default HeroBanner;

Этот структурирован для создания эффективного героического баннера.

<Box sx={{ mt: { lg: '212px', xs: '70px' }, ml: { sm: '50px' } }} position="relative" p="20px">: Используется компонент Box из Material-UI для создания контейнера с определенными стилями. Заданы отступы сверху (mt), слева (ml), позиция (position), и отступы внутри (p).

<TypographyЮ: Используется компонент Typography для отображения текста. Задан цвет, жирность шрифта и размер шрифта.

<Typography>: Опять используется Typography для заголовка. Заданы жирность шрифта, размер шрифта, отступы снизу и сверху, а также использование HTML-тега <br /> для переноса строки.

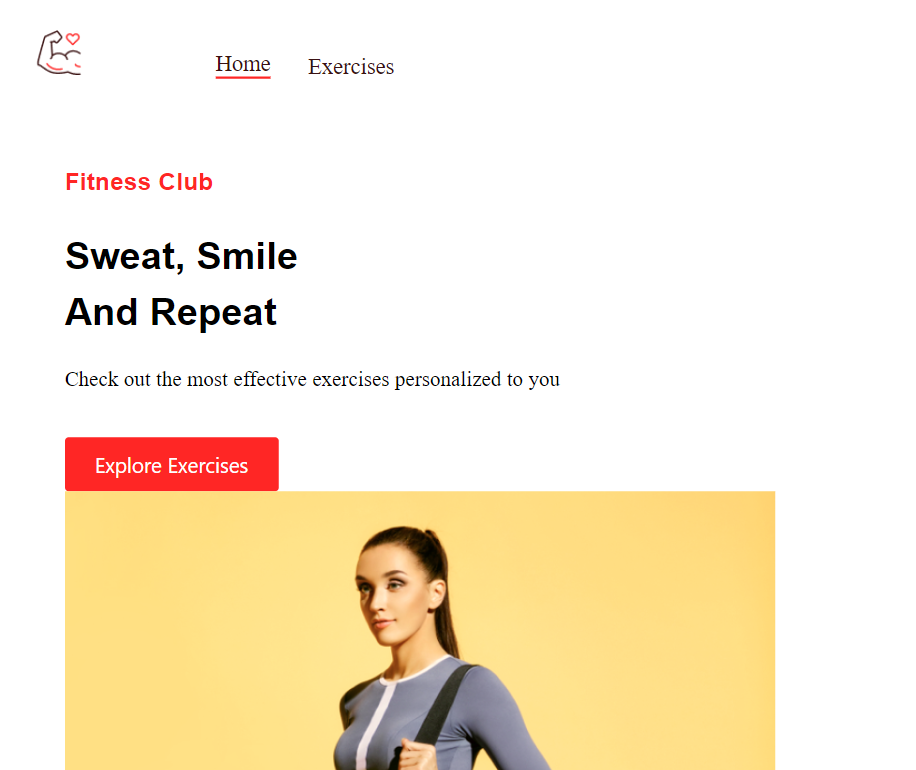
<Typography fontSize="22px" fontFamily="Alegreya" lineHeight="35px">Check out the most effective exercises personalized to you</Typography>: Еще один Typography для текста с заданными размером шрифта, семейством шрифта и высотой строки.

Используется Stack для создания стека элементов. В вашем случае, это контейнер, в котором расположена ссылка (кнопка).

HTML-ссылка (<a>) с определенными стилями, такими как отступ сверху (marginTop), цвет фона (background), отступы и размер текста.

<Typography Еще один Typography для текста "Exercise" с определенными стилями, такими как жирность шрифта, цвет и прозрачность. С помощью свойства display управляется видимость текста на разных экранах.

<img src={HeroBannerImage} alt="hero-banner" className="hero-banner-img" />: Используется тег <img> для отображения изображения героического баннера. Устанавливается атрибут src для указания пути к изображению, alt для текста, который будет отображаться, если изображение не может быть загружено, и класс стиля для применения дополнительных стилей через CSS.



**Step-4**

import React, { useEffect, useState } from "react";

import { Box, Button, Stack, TextField, Typography } from "@mui/material";

const SearchExercises = () => {

    const [search, setSearch ]= useState('')

    const handleSearch = async () => {

        if(search) {

            // const exercisesData = await fetchData()

        }

    }

    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: '#FF2625', 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>

        </Stack>

    )

}

export default SearchExercises;

const [search, setSearch ]= useState(''): Используется хук useState для создания состояния search и функции setSearch. По умолчанию устанавливается значение пустой строки.

const handleSearch = async () => { ... }: Функция handleSearch, которая будет вызываться при нажатии на кнопку поиска. В настоящее время она проверяет, что значение search не является пустой строкой, и, если это так, может выполнить какие-то дополнительные действия (например, вызывать функцию fetchData, которая, видимо, закомментирована).

<Stack>: Используется Stack для создания стека элементов. Устанавливаются отступы сверху (mt), выравнивание по центру (alignItems), распределение по центру (justifyContent) и внутренние отступы (p).

<Typography> Используется Typography для отображения заголовка с определенными стилями, такими как жирность шрифта, размер шрифта, отступ снизу (mb) и выравнивание по центру (textAlign).

<Box position="relative" mb="72px">...</Box>: Используется Box для создания контейнера с определенными стилями, такими как позиция и отступ снизу (mb). Внутри этого контейнера располагаются элементы для ввода текста (TextField) и кнопка (Button).

<TextField ... />: Используется компонент TextField для создания поля ввода. Устанавливаются различные стили, такие как высота, ширина, цвет фона, и дополнительные стили через свойство sx. Состояние search используется в качестве значения поля ввода, а функция setSearch вызывается при его изменении.

<Button ...>Search</Button>: Используется кнопка (Button) для запуска поиска. Устанавливаются различные стили, такие как цвет фона, цвет текста, ширина, высота и дополнительные стили через свойство sx. Функция handleSearch вызывается при нажатии на кнопку.

Компонент экспортируется по умолчанию.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

**Step-5**

import React, { useEffect, useState } from "react";

import { Box, Button, Stack, TextField, Typography } from "@mui/material";

import { exerciseOptions, fetchData } from "../utils/fetchData";

const SearchExercises = () => {

    const [search, setSearch ]= useState('')

    const [exercises, setExercises] = 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();

    }, []);

    const handleSearch = async () => {

        if(search) {

            const exercisesData = await fetchData('https://exercisedb.p.rapidapi.com/exercises', exerciseOptions)

            const searchedExercises = exercisesData.filter(

                (exercise) => exercise.name.toLowerCase().includes(search)

                    || exercise.target.toLowerCase().includes(search)

                    || exercise.equipment.toLowerCase().includes(search)

                    || exercise.bodyPart.toLowerCase().includes(search),

            );

            setSearch('');

            setExercises(searchedExercises);

        }

    }

    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: '#FF2625', 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>

                <HorizontalScrollbar data={bodyparts} />

            </Box>

        </Stack>

    )

}

export default SearchExercises;

Hаш компонент SearchExercises теперь включает запрос к API для получения данных о частях тела (body parts) и упражнениях (exercises) с использованием функции fetchData. Также в компоненте появился компонент HorizontalScrollbar, который предположительно используется для отображения горизонтального скроллбара с частями тела.

const [search, setSearch] = useState(''): Создание состояния search с хуком useState для отслеживания значения ввода поиска.

const [exercises, setExercises] = useState([]);: Создание состояния exercises для хранения списка упражнений.

const [bodyParts, setBodyParts] = useState([]);: Создание состояния bodyParts для хранения списка частей тела.

useEffect(() => { ... }, []);: Использование хука useEffect для выполнения асинхронного запроса к API при монтировании компонента. Полученные данные о частях тела сохраняются в состоянии bodyParts.

const handleSearch = async () => { ... }: Функция обработки поиска, которая вызывается при нажатии на кнопку. Она использует функцию fetchData для получения данных об упражнениях, затем фильтрует результаты в соответствии с введенным значением поиска и обновляет состояние exercises.

<HorizontalScrollbar data={bodyparts} />: Вероятно, этот компонент ожидает свойство data и будет отображать горизонтальный скроллбар для списка частей тела. Однако, обратите внимание на то, что название свойства data должно быть точным, и вы используете bodyparts, но в коде вы сохраняете данные в состоянии bodyParts. Убедитесь, что имена совпадают.

Внутри TextField используется функция setSearch для обновления состояния search при изменении значения ввода.

<Button className="search-btn" ... onClick={handleSearch}>: Кнопка, которая запускает функцию handleSearch при нажатии.

export const exerciseOptions = {

  method: 'GET',

  headers: {

    'X-RapidAPI-Host': 'exercisedb.p.rapidapi.com',

    'X-RapidAPI-Key': 'ff33ae0779msh2be7dc5cae73e3fp193e9ajsnfe199ebf34b7'

  },

};

export const fetchData = async (url, options) => {

    const response =  await fetch(url, options);

    const data = await response.json();

    return data;

}

Hаши экспортированные функции и объект exerciseOptions предназначены для выполнения HTTP-запросов к API.

exerciseOptions: Это объект, содержащий опции для выполнения HTTP-запросов. В частности, он используется для отправки запросов к API exercisedb.p.rapidapi.com. Опции включают метод запроса (GET) и заголовки, такие как 'X-RapidAPI-Host' и 'X-RapidAPI-Key'. Заголовки содержат ключи, необходимые для аутентификации при использовании RapidAPI.

fetchData: Это асинхронная функция, принимающая URL и опции запроса. Она использует функцию fetch для выполнения HTTP-запроса и затем парсит ответ в формат JSON. В конечном итоге, данные возвращаются из функции.

**Step-6**

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();

    }, []);

    const handleSearch = async () => {

        if(search) {

            const exercisesData = await fetchData('https://exercisedb.p.rapidapi.com/exercises', exerciseOptions)

            const searchedExercises = exercisesData.filter(

                (exercise) => exercise.name.toLowerCase().includes(search)

                    || exercise.target.toLowerCase().includes(search)

                    || exercise.equipment.toLowerCase().includes(search)

                    || exercise.bodyPart.toLowerCase().includes(search),

            );

            setSearch('');

            setExercises(searchedExercises);

        }

    }

    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: '#FF2625', 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;

В нашем обновленном компоненте SearchExercises добавлены новые пропсы (setExercises, bodyPart, setBodyPart), и компонент HorizontalScrollbar теперь используется для отображения горизонтального скроллбара с частями тела.

Добавленные пропсы:

setExercises: Этот пропс предположительно используется для установки списка упражнений, полученных после выполнения поиска.

bodyPart: Пропс для отслеживания выбранной части тела.

setBodyPart: Пропс для установки выбранной части тела.

Обновленный handleSearch:

Теперь, после выполнения поиска и получения данных, список упражнений устанавливается через setExercises. Это предположительно будет использоваться в родительском компоненте для отображения результатов поиска.

Добавлен HorizontalScrollbar:

Компонент HorizontalScrollbar теперь обернут в Box с дополнительными стилями. Он использует data, setBodyPart и bodyPart из пропсов для отображения списка частей тела с возможностью выбора.

Обновления делают компонент более многозадачным и предоставляют родительскому компоненту больше контроля над состоянием упражнений и выбранной частью тела.

import React, { useContext } from "react";

import { Box, Typography, } from "@mui/material";

import BodyPart from "./BodyPart";

import { ScrollMenu, VisibilityContext } from 'react-horizontal-scrolling-menu';

import RightArrowIcon from '../assets/icons/right-arrow.png';

import LeftArrowIcon from '../assets/icons/left-arrow.png';

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, bodyPart, setBodyPart }) => {

    return (

        <ScrollMenu LeftArrow={LeftArrow} RightArrow={RightArrow}>

            {data.map((item) => (

                <Box

                    key={item.id || item}

                    itemId={item.id || item}

                    title={item.id || item}

                    m="0 40px"

                >

                <BodyPart item={item} setBodyPart={setBodyPart} bodyPart={bodyPart} />

                </Box>

            ))

            }

        </ScrollMenu>

    )

}

export default HorizontalScrollbar;

Новый Компонент HorizontalScrollbar использует библиотеку react-horizontal-scrolling-menu для создания горизонтального скроллбара.

Импорты:

Импорт компонентов из библиотеки @mui/material и других компонентов.

Импорт компонентов VisibilityContext, LeftArrow, RightArrow, ScrollMenu из библиотеки react-horizontal-scrolling-menu.

Импорт иконок для стрелок.

Компоненты стрелок:

LeftArrow и RightArrow - это функциональные компоненты, представляющие стрелки для прокрутки влево и вправо. Они используют VisibilityContext для вызова функций scrollPrev и scrollNext при нажатии.

Компонент HorizontalScrollbar:

Этот компонент принимает data, bodyPart и setBodyPart в качестве пропсов.

Использует компонент ScrollMenu для создания горизонтального скроллбара.

Каждый элемент из data отображается внутри ScrollMenu с использованием компонента Box для обеспечения отступов.

Каждый элемент Box содержит компонент BodyPart, который, вероятно, отвечает за отображение части тела.

HorizontalScrollbar использует компоненты LeftArrow и RightArrow для управления прокруткой. Каждый элемент из data представлен в виде Box, в котором содержится компонент BodyPart. Вызываются функции setBodyPart и scrollNext/scrollPrev при взаимодействии с элементами скроллбара и стрелками.

import React from "react";

import { Stack, Typography } from "@mui/material";

import Icon from '../assets/icons/gym.png'

const BodyPart = ({ item, setBodyPart, bodyPart }) => {

    return (

        <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: '20px', width: '270px', height: '282px', cursor: 'pointer', gap: '47px' }}

        onClick={() => {

          setBodyPart(item);

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

        }}>

            <img src={Icon} alt="dumbbell" style={{ width: '40px', height: '40px' }} />

            <Typography fontSize="24px" fontWeight="bold" fontFamily="Alegreya" color="#3A1212" textTransform="capitalize"> {item}</Typography>

        </Stack>

    )

}

export default BodyPart;

Также создали новый компонент BodyPart представляет собой карточку для отображения части тела в горизонтальном скроллбаре.

Принимаемые пропсы:

item: Это часть тела, которую представляет компонент.

setBodyPart: Функция для обновления выбранной части тела.

bodyPart: Текущая выбранная часть тела.

Отображение компонента:

Stack используется для создания контейнера, представляющего карточку части тела.

Установлен атрибут type="button", что делает карточку кликабельной.

При клике на карточку вызывается функция setBodyPart(item), обновляющая выбранную часть тела.

Используется условие для изменения стилей в зависимости от того, является ли текущая часть тела (bodyPart) равной части тела, представленной в item.

Добавлено изображение (<img>) с иконкой, представляющей часть тела.

Добавлен текст (<Typography>) с именем части тела.

Этот компонент представляет красиво стилизованный элемент для выбора части тела с использованием изображения и текста. Стили изменяются в зависимости от того, является ли эта часть тела текущей выбранной частью (bodyPart).

import React, { useState } from "react";

import { Box } from "@mui/material";

import HeroBanner from "../components/HeroBanner";

import SearchExercises from "../components/SearchExercises";

import Exercises from "../components/Exercises";

const Home = () => {

    const [exercises, setExercises] = useState([]);

    const [bodyPart, setBodyPart] = useState('all');

    return (

    <Box>

      <HeroBanner />

      <SearchExercises setExercises={setExercises} bodyPart={bodyPart} setBodyPart={setBodyPart}/>

      <Exercises setExercises={setExercises} bodyPart={bodyPart} exercises={exercises}/>

    </Box>

    );

};

export default Home;

Состояние компонента:

Используется хук useState для управления состоянием компонента.

exercises: Состояние, представляющее список упражнений. Инициализировано пустым массивом.

bodyPart: Состояние, представляющее выбранную часть тела. Инициализировано строкой 'all'.

Возвращаемый JSX:

Возвращает JSX, представляющий разметку компонента.

Включает компоненты HeroBanner, SearchExercises, и Exercises.

Каждый компонент получает необходимые пропсы, такие как setExercises, bodyPart, и setBodyPart.

Этот компонент является основным компонентом для отображения домашней страницы и управляет состоянием упражнений и выбранной части тела через хуки useState.

**Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание**

**Step-7**

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);

  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]);

  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 />;

  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 > 9 && (

          <Pagination

            color="standard"

            shape="rounded"

            defaultPage={1}

            count={Math.ceil(exercises.length / exercisesPerPage)}

            page={currentPage}

            onChange={paginate}

            size="large"

          />

        )}

      </Stack>

    </Box>

  );

};

export default Exercises;

Компонент Exercises представляет собой раздел, отображающий упражнения.

Импорты:

Импорты React и компонентов из библиотеки @mui/material.

Импорт компонента ExerciseCard.

Импорт компонента Loader для отображения загрузки данных.

Импорт функций fetchData и объекта exerciseOptions из утилит fetchData.

Состояние компонента:

Используется хук useState для управления текущей страницы и количества упражнений на странице.

Загружается список упражнений для выбранной части тела (или всех упражнений, если bodyPart === 'all').

Разделение списка упражнений на страницы:

Рассчитываются индексы первого и последнего упражнения для текущей страницы.

Создается новый массив currentExercises, содержащий упражнения для текущей страницы.

Обработка событий пагинации:

Функция paginate вызывается при изменении страницы пагинации.

Обновляется состояние currentPage.

Происходит прокрутка к верхней части экрана с использованием window.scrollTo с плавным эффектом.

Визуализация данных:

Если упражнения еще не загружены, отображается компонент Loader.

Иначе отображается блок с заголовком, карточками упражнений, и пагинацией.

Этот компонент отображает упражнения в соответствии с выбранной частью тела, а также обеспечивает пагинацию для просмотра большого списка упражнений.

import React from "react";

import { Link } from 'react-router-dom';

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

const ExerciseCard = ({ exercise }) => {

    <Link className="exercise-card" to={`/exercise/${exercise.id}`}>

        <img src={exercise.gifUrl} alt={exercise.name} loading="lazy" />

        <Stack direction="row">

            <Button sx={{ ml: '21px', color: '#fff', background: '#FFA9A9', fontSize: '14px', borderRadius: '20px', textTransform: 'capitalize' }}>

                {exercise.bodyPart}

            </Button>

            <Button sx={{ ml: '21px', color: '#fff', background: '#FCC757', fontSize: '14px', borderRadius: '20px', textTransform: 'capitalize' }}>

                {exercise.target}

            </Button>

        </Stack>

        <Typography ml="21px" color="#000" fontWeight="bold" sx={{ fontSize: { lg: '24px', xs: '20px' } }} mt="11px" pb="10px" textTransform="capitalize">

            {exercise.name}

        </Typography>

    </Link>

}

export default ExerciseCard;

Компонент ExerciseCard представляет собой карточку упражнения для отображения на вашем сайте.

Компонент:

Компонент ExerciseCard принимает пропс exercise, который содержит информацию об упражнении.

Возвращается JSX, представляющий карточку упражнения.

Link используется для создания ссылки на страницу деталей упражнения с использованием exercise.id.

Карточка включает изображение упражнения, две кнопки с информацией о частях тела (bodyPart и target), и заголовок с именем упражнения.

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();

  }, []);

  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);

    }

  };

  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: '#FF2625', 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;

Компонент SearchExercises:

Используется функциональный компонент SearchExercises.

Используются хуки useState и useEffect для управления состоянием компонента и выполнения побочных эффектов.

Внутри useEffect выполняется асинхронный запрос к API для получения списка частей тела и установки его в состояние.

Функция handleSearch вызывается при нажатии кнопки поиска, фильтрует упражнения и устанавливает их в состояние setExercises.

Возвращается JSX, представляющий форму поиска и горизонтальную полосу прокрутки.

**Изображение выглядит как зарисовка, обувь, рисунок, скелет

Автоматически созданное описание**

**Изображение выглядит как зарисовка, текст, дизайн, иллюстрация

Автоматически созданное описание**

**Step-8**

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 (() => {

        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();

    }, [id]);

    return (

        <Box>

            <Detail exerciseDetail={exerciseDetail}/>

            <ExerciseVideos />

            <SimilarExercises />

        </Box>

    )

}

export default ExerciseDetail;

Компонент ExerciseDetail визуализирует детали упражнения, видеоролики по упражнению и похожие упражнения.

Импорты:

Импорты компонентов и хуков из библиотек react и @mui/material.

Импорты функций fetchData и опций запросов из файла fetchData.

Компонент ExerciseDetail:

Использует хук useEffect для выполнения асинхронных запросов при монтировании компонента.

Получает параметр id из URL с использованием useParams из react-router-dom.

Выполняет запросы к базе данных упражнений и YouTube для получения деталей упражнения, видеороликов и похожих упражнений.

Состояния хранят полученные данные.

Рендерит компоненты Detail, ExerciseVideos и SimilarExercises, передавая им полученные данные.

Этот компонент представляет собой страницу с деталями упражнения, видеороликами по упражнению и похожими упражнениями.

**Step-9**

import React from "react";

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

import BodyPartImage from '../assets/icons/body-part.png';

import TargetImage from '../assets/icons/target.png';

import EquipmentImage from '../assets/icons/equipment.png';

const Detail = ({ exerciseDetail }) => {

    const { bodyPart, gifUrl, name, target, equipment } = exerciseDetail;

    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="#4F4C4C">

                Exercises keep you strong.{' '}

                <span style={{ textTransform: 'capitalize' }}>{name}</span> bup is one

                of the best <br /> exercises to target your {target}. It will help you improve your{' '}

                <br /> mood and gain energy.

            </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;

Компонент Detail предназначен для отображения деталей упражнения, таких как название, изображение, описание и дополнительные сведения (часть тела, цель, оборудование).

Компонент Detail:

Получает exerciseDetail в качестве пропсов, который содержит информацию об упражнении.

Создает массив extraDetail с дополнительными сведениями о части тела, цели и оборудовании.

Использует компоненты Typography и Stack для стилизации текста и компоновки элементов.

Рендерит изображение упражнения, название, описание и дополнительные сведения.

Этот компонент структурирует и стилизует информацию о деталях упражнения, включая изображение, название, описание и дополнительные сведения.

**Изображение выглядит как текст, снимок экрана

Автоматически созданное описание**

**Step-10**

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();

  }, [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} />

    </Box>

  );

};

export default ExerciseDetail;

Компонент ExerciseDetail предназначен для отображения деталей конкретного упражнения.

Компонент ExerciseDetail:

Использует хук useEffect для выполнения запроса данных при загрузке компонента.

Использует хук useState для хранения состояния данных об упражнении, видеороликах и похожих упражнениях.

Использует хук useParams для получения параметра id из URL.

Выполняет запрос данных об упражнении, видеороликах и похожих упражнениях с помощью функции fetchData.

Использует компоненты Detail, ExerciseVideos, и SimilarExercises для отображения соответствующих данных.

Этот компонент загружает данные об упражнении, видеороликах и похожих упражнениях при монтировании, затем отображает эти данные с использованием соответствующих компонентов.

export const youtubeOptions = {

    method: 'GET',

    headers: {

        'X-RapidAPI-Key': 'ff33ae0779msh2be7dc5cae73e3fp193e9ajsnfe199ebf34b7',

        'X-RapidAPI-Host': 'youtube-search-and-download.p.rapidapi.com'

    }

};

Добавили новый параметры для HTTP-запроса. headers: {: Задаются заголовки (headers) для запроса. Заголовки - это часть HTTP-запроса, содержащая метаданные о запросе. method: 'GET',: Устанавливается метод запроса как GET. Это означает, что вы хотите получить (запросить) данные с сервера.

**Step-11**

import React from 'react';

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

import Loader from './Loader';

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>

  );

};

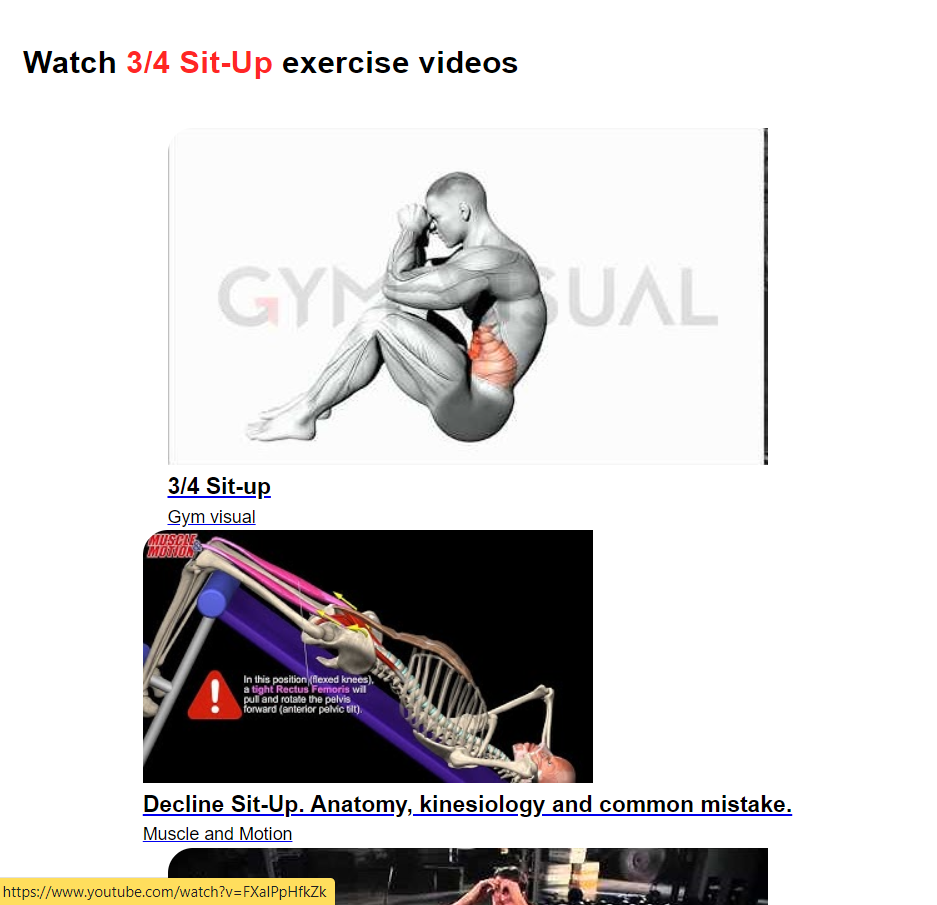
export default ExerciseVideos;

Этот компонент ExerciseVideos представляет собой часть вашего интерфейса пользователя, отображающую видео упражнения для конкретного упражнения.

Если массив exerciseVideos пуст (или ложен), то возвращается компонент Loader. Это может быть индикатор загрузки, который будет показан, пока видео не загрузятся.

Если exerciseVideos не пуст, то компонент возвращает контейнер Box с определенными стилями. Заголовок, который говорит "Смотрите видео упражнений для {name}". Typography - это компонент из Material-UI для стилизации текста. Компонент Stack используется для создания стека элементов. Задаются стили для расположения элементов в ряд на больших экранах (lg) и в столбец на маленьких (xs). Производится маппинг по первым трём элементам массива exerciseVideos. Создаются ссылки на YouTube для каждого видео. Вставляется изображение видео с закругленным верхним левым углом. Добавляются заголовок и имя канала под изображением видео.

Таким образом, ExerciseVideos создает раздел с видео упражнений для определенного упражнения, отображая изображение, заголовок и имя канала для каждого из первых трех видео в списке.



**Step-12**

import React from 'react';

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

import HorizontalScrollbar from './HorizontalScrollbar';

import Loader from './Loader';

const SimilarExercises = ({ targetMuscleExercises, equipmentExercises }) => (

  <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>

);

export default SimilarExercises;

Этот компонент SimilarExercises отображает похожие упражнения по целевой мышце (targetMuscleExercises) и оборудованию (equipmentExercises).

Компонент принимает два параметра: targetMuscleExercises - массив упражнений для похожих целевых мышц и equipmentExercises - массив упражнений для похожего оборудования. Создается контейнер Box с определенными стилями, включая отступ сверху в зависимости от размера экрана. Заголовок, который говорит "Похожие упражнения для целевых мышц". Используется Typography из Material-UI для стилизации текста. Стек элементов Stack в ряд с определенными стилями. В этом стеке размещается горизонтальный скроллбар HorizontalScrollbar, который отображает похожие упражнения по целевой мышце. Если массив targetMuscleExercises не пуст, то отображаются упражнения, в противном случае отображается индикатор загрузки Loader. Еще один заголовок для похожих упражнений по оборудованию. Еще один стек элементов с горизонтальным скроллбаром HorizontalScrollbar для отображения похожих упражнений по оборудованию. Если массив equipmentExercises не пуст, то отображаются упражнения, в противном случае отображается индикатор загрузки Loader. Таким образом, SimilarExercises отображает похожие упражнения для целевой мышцы и оборудования с использованием горизонтальных скроллбаров.

Изображение выглядит как текст, зарисовка, мультфильм, снимок экрана

Автоматически созданное описание