A PROJECT REPORT

On

**WORKOUT TRACKER**

Submitted by

Mr. Sachin Indra kumar Singh

in partial fulfillment for the award of the degree

of

**BACHELOR OF SCIENCE**

in

**COMPUTER SCIENCE**

under the guidance of

Mrs. Sneha Patil



**VPM’s R.Z. SHAH COLLEGE OF ARTS, Science & COMMERCE**

Mithagar Road, Mulund(E), Mumbai 400 081

**(NAAC Accredited "A" Grade)**

Sem v

2023-24

**VPM’s R.Z. SHAH COLLEGE OF ARTS, Science & COMMERCE**

Mithagar Road, Mulund(E), Mumbai 400 081

**(NAAC Accredited "A" Grade)**

—-------------------------------------------------------

**CERTIFICATE**

This is to certify that **Mr. Sachin Indra kumar Singh** of **T.Y.B.Sc(Sem V)** has satisfactorily completed the project **Workout Tracker,** to be submitted in the partial fulfilment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2023-2024**

**Date of Submission:**

**Mrs. Sneha Patil Mr. Vilas Mahajan**

**Project Guide HOD / In-charge**

**DECLARATION**

I **, Sachin Indra kumar Singh** hereby declare that the project entitled **“ Workout Tracker ”** submitted in the partial fulfilllment for the award of **Bachelor of Science** in **Computer Science** during the academic year **2023-2024** is my original work and the project has not formed the basis for the award of any degree ,associateship ,fellowship or any other similar titles .

Signature of Student :

Place :

Date :

**Introduction:**

The project entitled Workout Tracker is a project made for gym freaks person to keep track of their daily exercise routine.

It is a cutting-edge workout tracker application designed to revolutionize your fitness journey. With its user-friendly interface and advanced features, FitTrack empowers users to take control of their fitness goals, track their progress, and stay motivated throughout their workouts.

**Description:**

This project builds builds a workout tracker with express JS and MongoDB. User can start a workout plan and each plan can keep track of multiple exercises.

This app can track many type of exercises :

For resistance exercises it tracks weight , reps , sets.

After all the duration of every exercises tracked as well. User can also see summary of each workout at the main page. If user wants more detailed data , he/she can view dashboard for detailed distribution of exercises performed over last 7 workout plans.

**Objective:**

The objective of the system is to help user in maintaining their routine of exercising and increasing it gradually over a period of time.

By keeping record they can easily analyse their performance and improve it. The system helps user to gain better and improve rapidly.

**Advantages:**

The various advantages of using this workout tracker is mentioned below:

* **Personalized Profiles**: It allows users to create personalized profiles by inputting their physical attributes, fitness goals, and exercise preferences. This information is used to generate tailored workout plans and recommendations.
* **Workout Logging**: Users can easily log their workouts by selecting from a vast database of exercises or creating custom exercises. FitTrack tracks essential metrics such as sets, reps, weights, and duration, providing a detailed record of each session.
* **Progress Tracking**: It automatically collects and analyzes data from logged workouts, offering visual representations of progress over time. Users can track metrics like weight lifted, repetitions completed, calories burned, and more, enabling them to monitor their improvements and identify areas for growth.
* **Goal Setting**:It empowers users to set realistic fitness goals and provides a roadmap to achieve them. Whether it's weight loss, muscle gain, or increased endurance, the application offers tailored plans and milestones to guide users towards success.
* Helps user in keeping record of their overall workout they’ve done this week
* It also helps user in pushing their limits to gain better.
* They can watch their detailed information of total workout in dashboard.
* They also get a summary of daily workout in the form of graph on main page of the application.

The project entitled Library Management System is a pilot project for small

school to manage their library administration process.

Project:

The Library Management System is the software to keep the transaction

records that happens in the library. It avoids tedious and time-consuming

manual system of the library. It provides the features such as to keep the records

of books, issue books and so on.

Methodology:

Library is the place where information and books are stored. It is the place

where people from all fields uses the books and information required for them.

But for that purpose since long time the manual system was followed during

operation using the library card. But with the increasing demands of technology

in various fields are forcing to stop the manual system in library operation. So

to fulfill the increasing demands of the library management software has been

developed.

The project entitled Library Management System is a pilot project for small

school to manage their library administration process.

Project:

The Library Management System is the software to keep the transaction

records that happens in the library. It avoids tedious and time-consuming

manual system of the library. It provides the features such as to keep the records

of books, issue books and so on.

Methodology:

Library is the place where information and books are stored. It is the place

where people from all fields uses the books and information required for them.

But for that purpose since long time the manual system was followed during

operation using the library card. But with the increasing demands of technology

in various fields are forcing to stop the manual system in library operation. So

to fulfill the increasing demands of the library management software has been

developed.

The project entitled Library Management System is a pilot project for small

school to manage their library administration process.

Project:

The Library Management System is the software to keep the transaction

records that happens in the library. It avoids tedious and time-consuming

manual system of the library. It provides the features such as to keep the records

of books, issue books and so on.

Methodology:

Library is the place where information and books are stored. It is the place

where people from all fields uses the books and information required for them.

But for that purpose since long time the manual system was followed during

operation using the library card. But with the increasing demands of technology

in various fields are forcing to stop the manual system in library operation. So

to fulfill the increasing demands of the library management software has been

developed.

**Requirement Specification**

**Hardware Requirement:**

Operating System : Windows/Linux

Hard Disk :50 GB

RAM : 4GB

**Software Requirement:**

VS Code

Mongo DB

**Technologies Used:**

HTML

CSS

JS

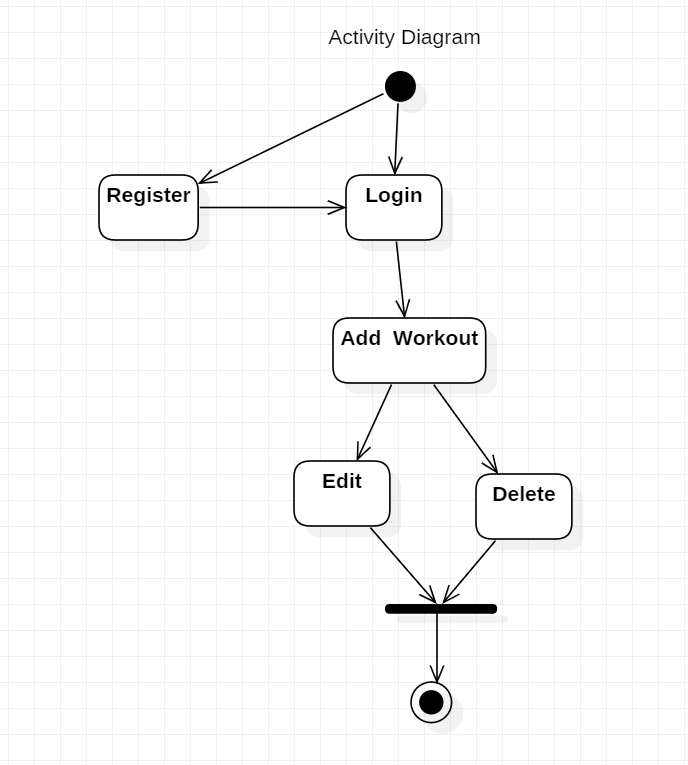
MongoDB

Node JS

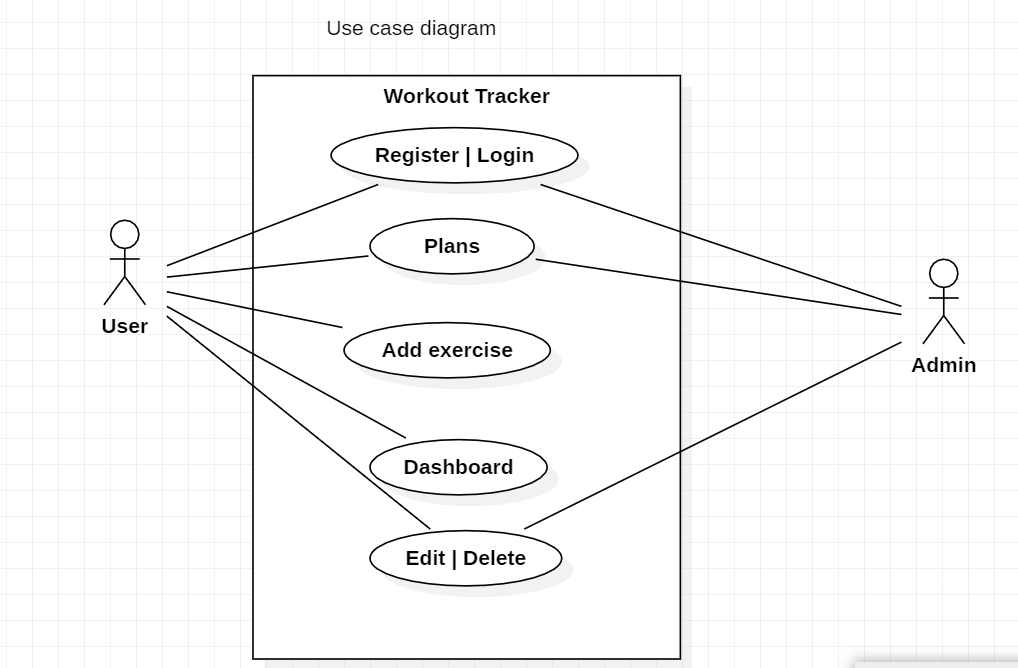
Express JS

UML Diagram

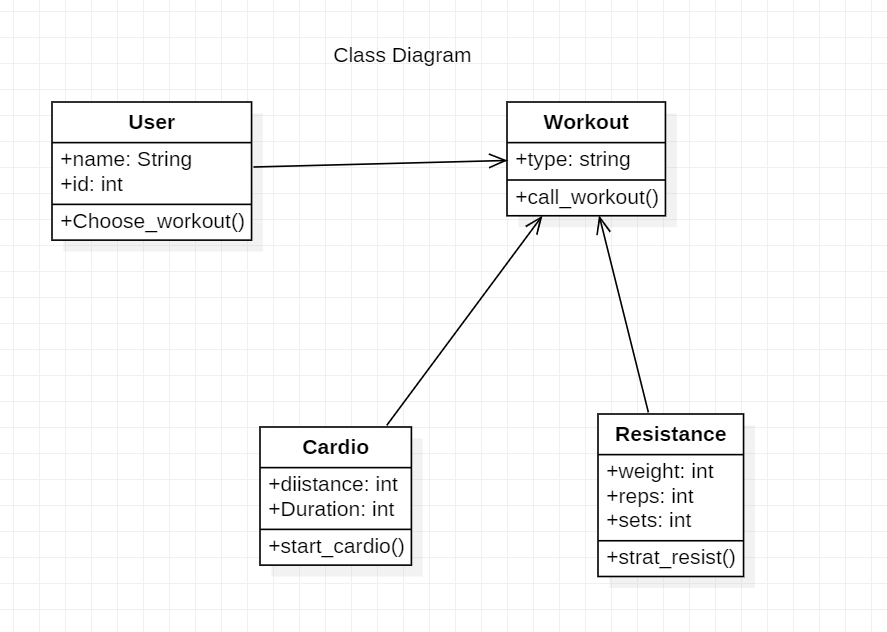
1. Activity Diagram



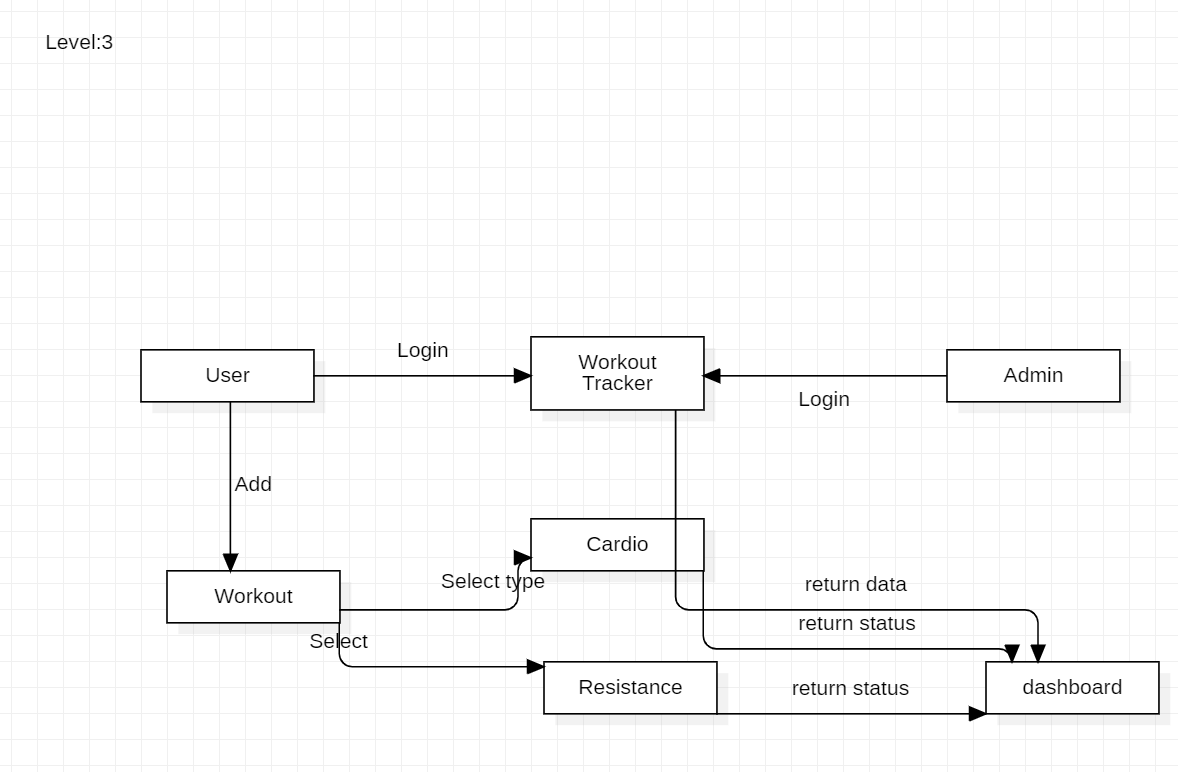
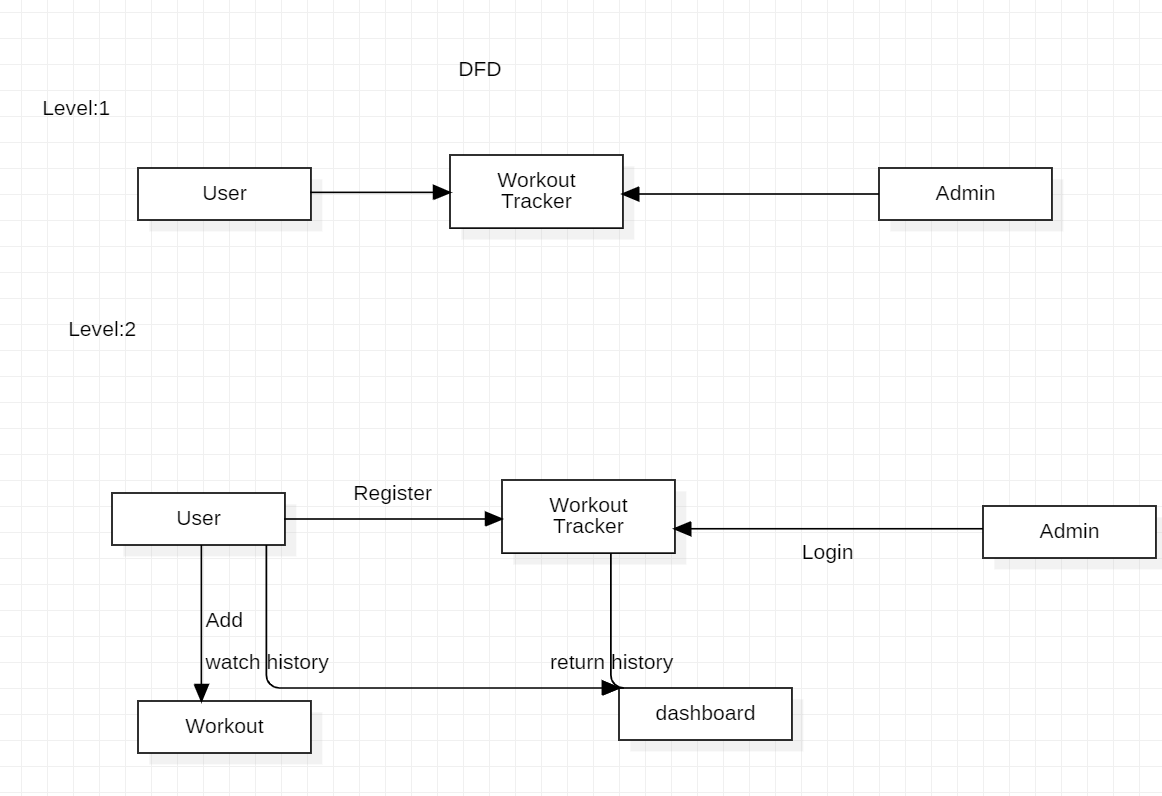
2. Use case Diagram



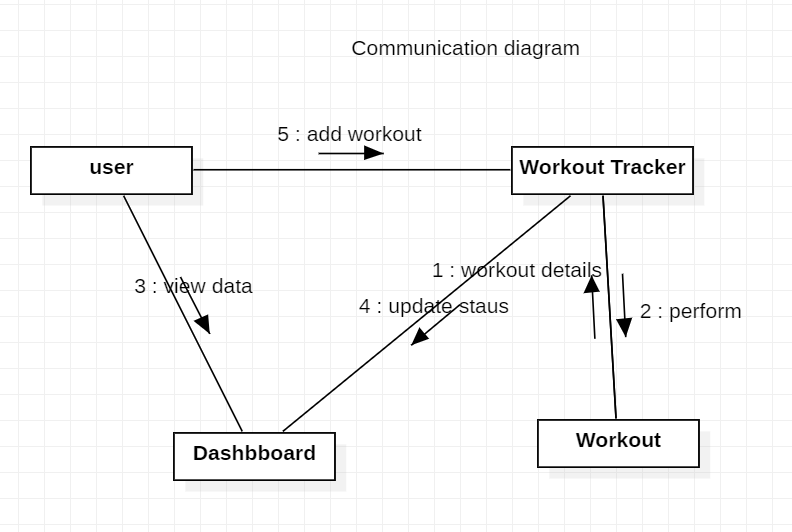
3.Class Diagram



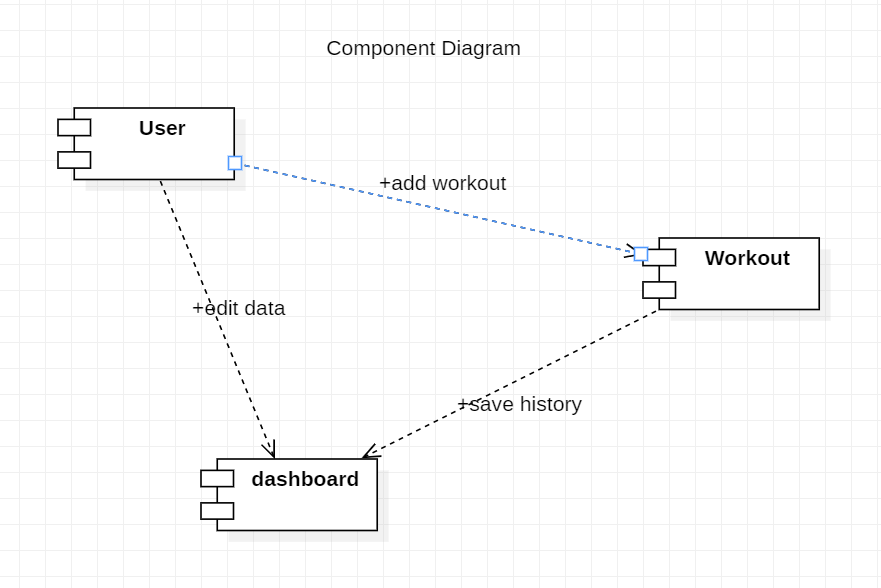
4.DFD



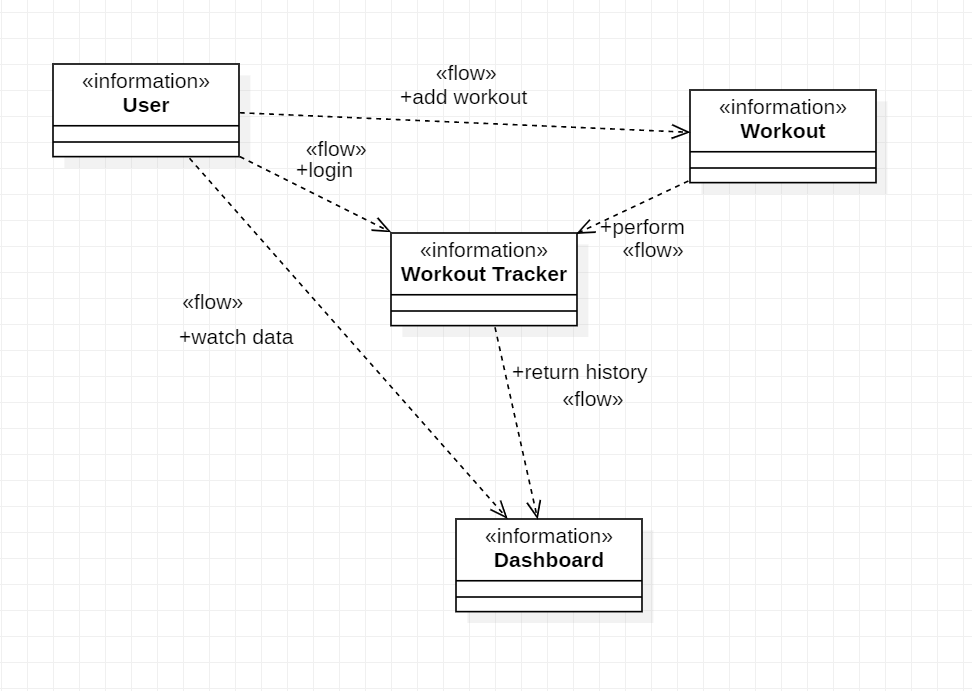
5 . Communication Diagram



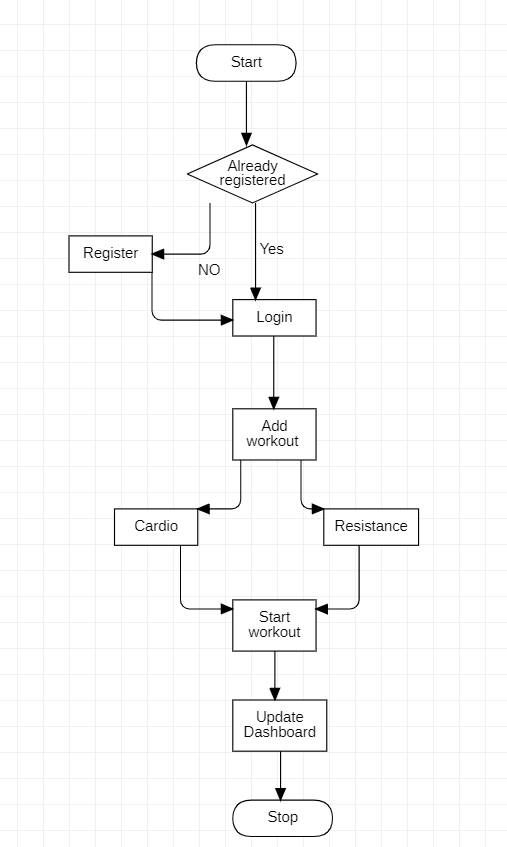
6. Component Diagram



7. Information Flow Diagram



8. Flow Chart



**System Implementations**

**Frontend files**

**Home.jsx**

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

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

import MotivationImage from "./Randomimage";

import "../styles/home.css";

const Home = () => {

  const [motivationText, setMotivationText] = useState("");

  const motivationQuotes = [

    //motivational quotes here

  ];

  useEffect(() => {

    // Generate random motivation text when the component mounts (on website load or refresh)

    const randomIndex = Math.floor(Math.random() \* motivationQuotes.length);

    setMotivationText(motivationQuotes[randomIndex]);

  }, []);

  return (

    <>

      <p style={{ fontSize: "18px", color: "#333", lineHeight: "1.5" ,margin:'20px',textAlign:'center',fontFamily:'monospace',fontSize:'22px',fontStyle:'italic'}}>

        <q>{motivationText}</q>

      </p>

      <div className="home-page">

        <div className="first-part">

          <Box className="box1">

            <Typography

              variant="h3"

              color="primary"

              fontWeight="bold"

              sx={{ margin: "20px" }}

            >

              Get Fit

            </Typography>

            <Typography variant="h3" color="textPrimary" fontWeight="bold">

              Sweat, Smile <br /> And Repeat

            </Typography>

            <Typography

              variant="body1"

              color="textSecondary"

              sx={{ marginTop: "30px" }}

            >

              Check out the most effective exercises personalized to you

            </Typography>

            <Stack>

              <a href="/exercises" className="explore-button">

                Explore Exercises

              </a>

            </Stack>

          </Box>

        </div>

        <div className="image-motivation">

          <MotivationImage />

        </div>

      </div>

      <div className="features">

      <ul className="custom-symbol">

      <li>Plan your training and track your progress.</li>

      <br/>

      <li>Keep track of your best sets, body mass index  and more.</li>

      <br/>

      <li>Log unlimited workouts</li>

      <br/>

      <li>Exercise Guides & Muscles Worked</li>

      </ul>

      </div>

    </>

  );

};

export default Home;

**Navbar.jsx**

import React, { useState } from "react";

import {

  AppBar,

  Typography,

  Container,

  IconButton,

  Button,

  Toolbar,

  Tabs,

  Tab,

} from "@mui/material";

import FitnessCenterIcon from "@mui/icons-material/FitnessCenter";

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

import { useAuth0 } from "@auth0/auth0-react";

const Navbar = () => {

  const [value, setValue] = useState("Home");

  const { loginWithRedirect, logout,isAuthenticated,user } = useAuth0();

  return (

    <AppBar position="static" sx={{ height: "80px" }}>

      <Container maxWidth="xl">

        <Toolbar disableGutters variant="regular">

          <IconButton

            variant="contained"

            size="large"

            sx={{ color: "white" }}

            component={Link}

            to="/"

          >

            <FitnessCenterIcon

              sx={{ display: { xs: "none", md: "flex" }, mr: 1 }}

            />

          </IconButton>

          <Typography

            variant="h6"

            href="/"

            sx={{

              ml: 2,

              mr: 2,

              display: { xs: "none", md: "flex" },

              letterSpacing: "0.15rem",

            }}

          >

            Workout Tracker

          </Typography>

          <Tabs

            textColor="inherit"

            value={value}

            onChange={(e, value) => setValue(value)}

            indicatorColor="secondary"

          >

            <Tab label="Home" component={Link} to="/" />

            <Tab label="Exercises" component={Link} to="/exercises" />

            <Tab label="BMI" component={Link} to="/bmi" />

            <Tab label="Add Workout" component={Link} to="/create-exercise" />

            <Tab label="Summary" component={Link} to="/dashboard" />

          </Tabs>

          {isAuthenticated ? (

            <>

            <p style={{marginLeft:'auto',color:"mistyrose"}}> Welcome {user.name}</p>

            <Button

              onClick={() =>

                logout({ logoutParams: { returnTo: window.location.origin } })

              }

              variant=" outlined"

              sx={{ marginLeft: "auto" }}

            >

              Log Out

            </Button>

            </>

          ) : (

            <Button

              onClick={() => loginWithRedirect()}

              variant=" outlined"

              sx={{ marginLeft: "auto" }}

            >

              Log In

            </Button>

          )}

        </Toolbar>

      </Container>

    </AppBar>

  );

};

export default Navbar;

**Dashboard.jsx**

import React from 'react'

import { useEffect,useState } from 'react';

import Detailsworkout from './Detailsworkout';

const Dashboard = () => {

  const [workouts,setWorkouts]=useState(null)

  useEffect(()=>{

    const fetchWorkouts=async () =>{

      const response=await fetch('/api/workouts')

      const json=await response.json()

      if(response.ok){

        setWorkouts(json)

      }

    }

  fetchWorkouts()

  },[])

  return (

    <div>

    {workouts && workouts.map((workout) =>(

        <Detailsworkout key={workout.\_id} workout={workout} />

      ))}

    </div>

  )

}

export default Dashboard

**Createexercise.jsx**

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

import "../styles/createexercise.css";

const Createexercise = () => {

  const [title, setTitle] = useState("");

  const [load, setLoad] = useState("");

  const [reps, setReps] = useState("");

  const [error, setError] = useState(null);

  const [emptyFields, setEmptyFields] = useState([]);

  const [difficulty, setDifficulty] = useState("");

  const handleSubmit = async (e) => {

    e.preventDefault();

    const workout = { title, load, reps, difficulty };

    const response = await fetch("/api/workouts", {

      method: "POST",

      body: JSON.stringify(workout),

      headers: {

        "Content-Type": "application/json",

      },

    });

    const json = await response.json();

    if (!response.ok) {

      setError(json.error);

      setEmptyFields(json.emptyFields);

    }

    if (response.ok) {

      setError(null);

      setTitle("");

      setLoad("");

      setReps("");

      setEmptyFields([]);

      setDifficulty(""); // Reset the difficulty

      console.log("new workout added:", json);

    }

  };

  return (

    <form className="create" onSubmit={handleSubmit}>

      <h3>Add a New Workout</h3>

      <label>Difficulty Level:</label>

      <select

        onChange={(e) => setDifficulty(e.target.value)}

        value={difficulty}

      >

        <option value="">Select a difficulty level</option>

        <option value="easy">Easy</option>

        <option value="intermediate">Intermediate</option>

        <option value="hard">Hard</option>

      </select>

      <label>Exercise Title:</label>

      <select

        onChange={(e) => setTitle(e.target.value)}

        value={title}

        disabled={!difficulty} // Disable the exercise selection until a difficulty is chosen

      >

        <option value="">Select an exercise</option>

        {difficulty === "easy" && (

          <>

            <option value="High Knees Run">High Knees Run</option>

            <option value="Power Walking">Power Walking</option>

            <option value="Rowing">Rowing</option>

            <option value="Cycling">Cycling</option>

            <option value="Jumping Rope">Jumping Rope</option>

            <option value="Swimming">Swimming</option>

          </>

        )}

        {difficulty === "intermediate" && (

          <>

            <option value="Push ups">Push ups</option>

            <option value="Pull ups">Pull ups</option>

            <option value="Squats">Squats</option>

            <option value="Deadlifts">Deadlifts</option>

          </>

        )}

        {difficulty === "hard" && (

          <>

            <option value="Walking lunges">Walking lunges</option>

             <option value="Bicep Curls">Bicep Curls</option>

             <option value="Overhead Tricep Extensions">Overhead Tricep Extensions</option>

             <option value="Boat Pose">Boat Pose</option>

             <option value="Planks">Planks</option>

          </>

        )}

      </select>

      <label>Number of Sets:</label>

      <input

        type="number"

        onChange={(e) => setLoad(e.target.value)}

        value={load}

      />

      <label>Number of Reps:</label>

      <input

        type="number"

        onChange={(e) => setReps(e.target.value)}

        value={reps}

      />

      <button type="submit">Add Workout</button>

      {error && <div className="error">{error}</div>}

    </form>

     );

};

export default Createexercise;

**Bmi.jsx**

import React ,{useState} from 'react'

import '../styles/bmi.css'

const Bmi = () => {

  const [weight,setWeight]=useState(' ')

  const [height,setHeight]=useState(' ')

  const [bmi,setBmi]=useState(' ')

  const [message,setMessage]=useState(' ')

  let calcBmi = (event) => {

      //prevent submitting to the server

      event.preventDefault()

      if (weight === 0 || height === 0) {

        alert('Please enter a valid weight and height')

      } else {

        let bmi = (weight \*10000 / (height \* height) )

        setBmi(bmi.toFixed(1))

        // Logic for message

        if (bmi < 25) {

          setMessage('You are underweight')

        } else if (bmi >= 25 && bmi < 30) {

          setMessage('You are a healthy weight')

        } else {

          setMessage('You are overweight')

        }

      }

    }

    let reload = () => {

      window.location.reload()

    }

  return (

    <div className="app">

    <div className='container'>

      <h2 className='center'>BMI Calculator</h2>

      <form onSubmit={calcBmi}>

        <div>

          <label>Weight (kgs)</label>

          <input value={weight} onChange={(e) => setWeight(e.target.value)} />

        </div>

        <div>

          <label>Height (cms)</label>

          <input value={height} onChange={(event) => setHeight(event.target.value)} />

        </div>

        <div>

          <button className='btn' type='submit'>Submit</button>

          <button className='btn btn-outline' onClick={reload} type='submit'>Reload</button>

        </div>

      </form>

      <div className='center'>

        <h3>Your BMI is: {bmi}</h3>

        <p>{message}</p>

      </div>

    </div>

    </div>

  )

}

export default Bmi

**Footer.jsx**

import React from 'react'

import FacebookIcon from '@mui/icons-material/Facebook';

import TwitterIcon from '@mui/icons-material/Twitter';

import InstagramIcon from '@mui/icons-material/Instagram';

import FitnessCenterIcon from '@mui/icons-material/FitnessCenter';

import '../styles/footer.css'

import { IconButton } from '@mui/material';

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

const Footer = () => {

  return (

    <footer className="footer">

    <div className="container">

      <div className="footer-content">

        <div className="footer-logo" >

            <IconButton sx={{color:'white ' ,ml:'100px'}} size='medium' variant='contained'

            component={Link} to='/' >

             <FitnessCenterIcon />

             </IconButton>

        </div>

        <div className="footer-contact">

          <p>Contact us:</p>

          <p>Email: <a href="17sachinthakur@gmail.com">17sachinthakur@gmail.com</a></p>

          <p>Phone: <a href="tel:+1234567890">+1234567890</a></p>

        </div>

        <div className="footer-social">

          <p>Follow us:</p>

          <a href="https://www.facebook.com/yourpage" target="\_blank" rel="noopener noreferrer">

            <FacebookIcon/>

          </a>

          <a href="https://www.twitter.com/yourpage" target="\_blank" rel="noopener noreferrer">

            <TwitterIcon/>

          </a>

          <a href="https://instagram.com/mr\_singhsachin?utm\_source=qr&igshid=MzNlNGNkZWQ4Mg==" target="\_blank" rel="noopener noreferrer">

            <InstagramIcon/>

          </a>

        </div>

      </div>

    </div>

  </footer>

  )

}

export default Footer

**CSS files**

**Home.css**

.home-page {

    display: flex;

    align-items: center;

    justify-content: space-between;

    padding: 20px;

  }

  .first-part {

    flex: 1;

  }

  .box1 {

    max-width: 600px;

    margin-top: -150px;

    text-align: center;

    justify-content: center;

  }

  .explore-button {

    display: inline-block;

    width: 50%;

    text-decoration: none;

    margin-top: 20px;

    padding: 10px 10px;

    font-size: 22px;

    text-transform: uppercase;

    background-color: #FF2625;

    color: white;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    transition: background-color 0.3s ease;

    position: relative;

    left: 50%;

    transform: translateX(-50%);

  }

  .explore-button:hover {

    background-color: #D91E1E;

  }

  .image-motivation {

    flex: 1;

    text-align: center;

  }

  .features{

    margin: 10px 20px;

    padding: 20px;

    /\* background-color: #f0f0f0; \*/

    border-radius: 5px;

    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

    transition: background-color 0.3s ease;

    width: 95%;

    font-size: 20px;

    line-height: 30px;

  }

  li:hover{

    background-color: #e0e0e0;

    color:#007bff;

  }

  ul.custom-symbol {

    list-style: none; /\* Remove default list-style \*/

    padding-left: 1.5em; /\* Adjust the padding as needed for spacing \*/

}

ul.custom-symbol li::before {

    content: "\00BB"; /\* Unicode character for a checkmark (✔) \*/

    color: #007bff; /\* Change the color as desired \*/

    font-size: 1.2em; /\* Adjust the font size as needed \*/

    margin-right: 0.5em; /\* Adjust the spacing between the symbol and text \*/

}

**Createexercise.css**

.create {

    max-width: 400px;

    margin: 0 auto;

    margin-top: 40px;

    margin-bottom: 40px;

    padding: 20px;

    border: 1px solid #ccc;

    border-radius: 5px;

    background-color: #fff;

    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

  }

  .create h3 {

    font-size: 1.2rem;

    margin-bottom: 20px;

    text-align: center;

  }

  .create label {

    display: block;

    font-weight: bold;

    margin-top: 10px;

  }

  .create input {

    width: 100%;

    padding: 10px;

    margin-top: 5px;

    margin-bottom: 15px;

    border: 1px solid #ccc;

    border-radius: 3px;

    font-size: 1rem;

  }

  .create select{

    width: 100%;

    padding: 10px;

    margin-top: 5px;

    margin-bottom: 15px;

    border: 1px solid #ccc;

    border-radius: 3px;

    font-size: 1rem;

  }

  .create button {

    display: block;

    width: 100%;

    padding: 10px;

    background-color: #007bff;

    color: #fff;

    border: none;

    border-radius: 3px;

    font-size: 1rem;

    cursor: pointer;

    transition: background-color 0.2s;

  }

  .create button:hover {

    background-color: #0056b3;

  }

  .error {

    color: #ff0000;

    margin-top: 10px;

    text-align: center;

  }

**Footer.css**

/\* Footer.css \*/

/\* Style for the main footer container \*/

.footer {

    background-color:gray;

    color:#fff;

    bottom:0;

  }

  /\* Center the content within the container \*/

  .container {

    display: flex;

    justify-content: space-around;

    align-items: center;

  }

 .footer-logo{

    display: flex;

    align-items: center;

    flex-wrap: wrap;

    align-items: center;

 }

  /\* Style for the contact and social media sections \*/

  .footer-contact,

  .footer-social {

    text-align: center;

  }

  .footer-contact p,

  .footer-social p {

    margin: 5px 0;

  }

  .footer-contact a,

  .footer-social a {

    color: #fff;

    text-decoration: none;

    margin: 0 10px;

  }

  .footer-contact a:hover,

  .footer-social a:hover {

    text-decoration: underline;

    background-color:steelblue;

    /\* color: #fff; \*/

  }

  /\* Style for Font Awesome icons \*/

  .footer-social .fa {

    margin: 0 10px;

  }

  /\* Responsive styles \*/

  @media (max-width: 768px) {

    .container {

      flex-direction: column;

      text-align: center;

    }

    .footer-logo img {

      margin-bottom: 10px;

    }

    .footer-contact,

    .footer-social {

      margin-top: 20px;

    }

  }

**Bmi.css**

\* {

    box-sizing: border-box;

    margin: 0;

    padding: 0;

  }

/\* Center the entire .container vertically \*/

.container {

  display: flex;

  flex-direction: column;

  justify-content: center;

  align-items: center;

  box-shadow: 0px 0px 12px #ccc;

  border-radius: 8px;

  padding: 2rem; /\* Reduce padding for a more compact appearance \*/

}

/\* Adjust input styling for better visual alignment \*/

input {

  width: 100%;

  font-size: 1.2rem;

  padding: 12px; /\* Increase padding for better touch/click area \*/

  margin: 8px 0;

  border-radius: 8px;

  border: 1px solid #ccc; /\* Add a border for visual separation \*/

}

/\* Style the buttons consistently \*/

.btn {

  display: inline-block; /\* Change to inline-block to fit content width \*/

  font-size: 1.2rem;

  margin: 8px;

  padding: 12px 24px; /\* Increase padding for better button size \*/

  border: none;

  border-radius: 8px;

  cursor: pointer;

  transition: background-color 0.3s ease; /\* Add a subtle hover effect \*/

}

.btn:hover {

  background-color: #0055cc; /\* Change background color on hover \*/

  color: #fff;

}

/\* Center the result text and adjust margins \*/

.center {

  text-align: center;

  margin: 24px 0;

}

/\* Style the result message with a colored background \*/

.center h3 {

  background-color: #0077ee; /\* Add a background color to the heading \*/

  color: #fff;

  padding: 8px 16px; /\* Increase padding for better visibility \*/

  border-radius: 4px;

  margin: 0;

}

**Backend files**

**workoutController.js**

const Workout = require('../models/workoutModels')

const mongoose = require('mongoose')

// get all workouts

const getWorkouts = async (req, res) => {

  const workouts = await Workout.find({}).sort({createdAt: -1})

  res.status(200).json(workouts)

}

// get a single workout

const getWorkout = async (req, res) => {

  const { id } = req.params

//id validation

  if (!mongoose.Types.ObjectId.isValid(id)) {

    return res.status(404).json({error: 'No such workout'})

  }

  const workout = await Workout.findById(id)

  if (!workout) {

    return res.status(404).json({error: 'No such workout'})

  }

  res.status(200).json(workout)

}

// create a new workout

const createWorkout = async (req, res) => {

  const {title, load, reps} = req.body

  let emptyFields=[]

  if(!title){

    emptyFields.push('title')

  }

  if(!load){

    emptyFields.push('load')

  }

  if(!reps){

    emptyFields.push('reps')

  }

  if (emptyFields.length>0){

    return res.status(400).json({error:"Please fill in all the fields",emptyFields})

  }

  // add to the database

  try {

    const workout = await Workout.create({ title, load, reps })

    res.status(200).json(workout)

  } catch (error) {

    res.status(400).json({ error: error.message })

  }

}

// delete a workout

const deleteWorkout = async (req, res) => {

  const { id } = req.params

  if (!mongoose.Types.ObjectId.isValid(id)) {

    return res.status(400).json({error: 'No such workout'})

  }

  const workout = await Workout.findOneAndDelete({\_id: id})

  if(!workout) {

    return res.status(400).json({error: 'No such workout'})

  }

  res.status(200).json(workout)

}

// update a workout

const updateWorkout = async (req, res) => {

  const { id } = req.params

  if (!mongoose.Types.ObjectId.isValid(id)) {

    return res.status(400).json({error: 'No such workout'})

  }

  const workout = await Workout.findOneAndUpdate({\_id: id}, {

    ...req.body

  })

  if (!workout) {

    return res.status(400).json({error: 'No such workout'})

  }

  res.status(200).json(workout)

}

module.exports = {

  getWorkouts,

  getWorkout,

  createWorkout,

  deleteWorkout,

  updateWorkout

}

**workoutModel.js**

const mongoose=require('mongoose');

const Schema=mongoose.Schema

const workoutSchema=new Schema({

    title:{

        type:String,

        required:true

    },

    reps:{

        type:Number,

        required:true

    },

    load:{

        type:Number,

        required:true

    }

},{timestamps:true})

module.exports=mongoose.model('Workout',workoutSchema )

**workout.js(routes)**

const express = require('express')

const {

  getWorkouts,

  getWorkout,

  createWorkout,

  deleteWorkout,

  updateWorkout

} = require('../controllers/workoutControllers.js')

const router = express.Router()

// GET all workouts

router.get('/', getWorkouts)

// GET a single workout

router.get('/:id', getWorkout)

// POST a new workout

router.post('/', createWorkout)

// DELETE a workout

router.delete('/:id', deleteWorkout)

// UPDATE a workout

router.patch('/:id', updateWorkout)

module.exports = router

**Server.js**

require("dotenv").config();

const workoutRoutes = require("./routes/workouts");

const express = require("express");

const mongoose = require("mongoose");

const app = express();

app.use(express.json());

app.use((req, res, next) => {

  console.log(req.path, req.method);

  next();

});

app.use("/api/workouts", workoutRoutes);

mongoose

  .connect(process.env.MONGO\_URI)

  .then(() => {

    app.listen(process.env.PORT, () => {

      console.log("Listening on port", process.env.PORT);

    });

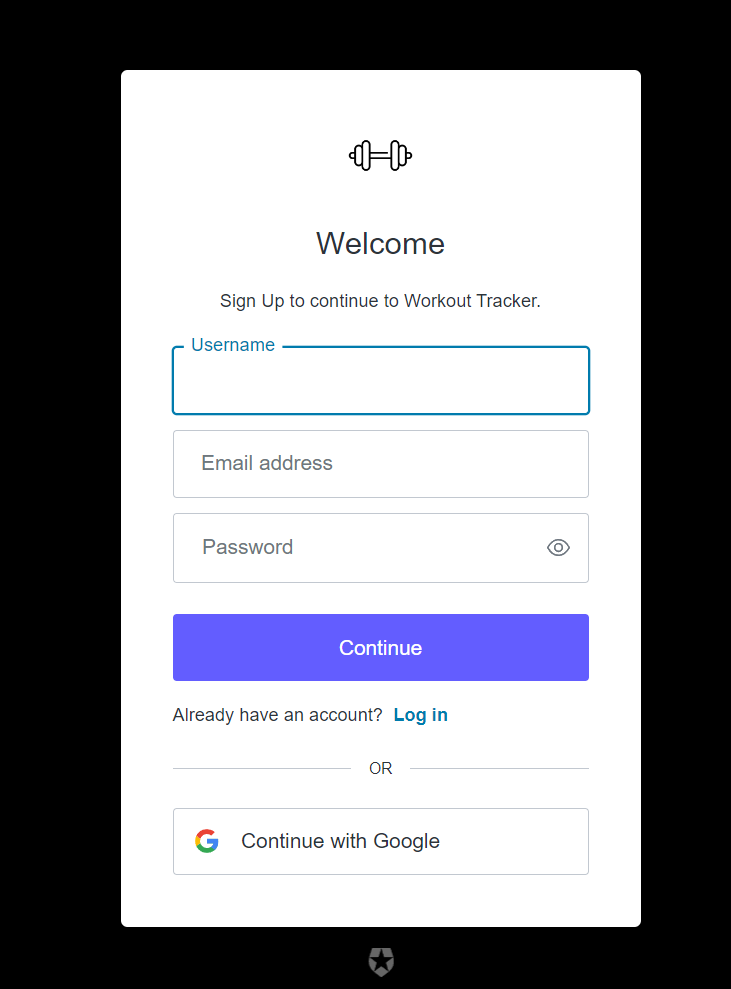
  })

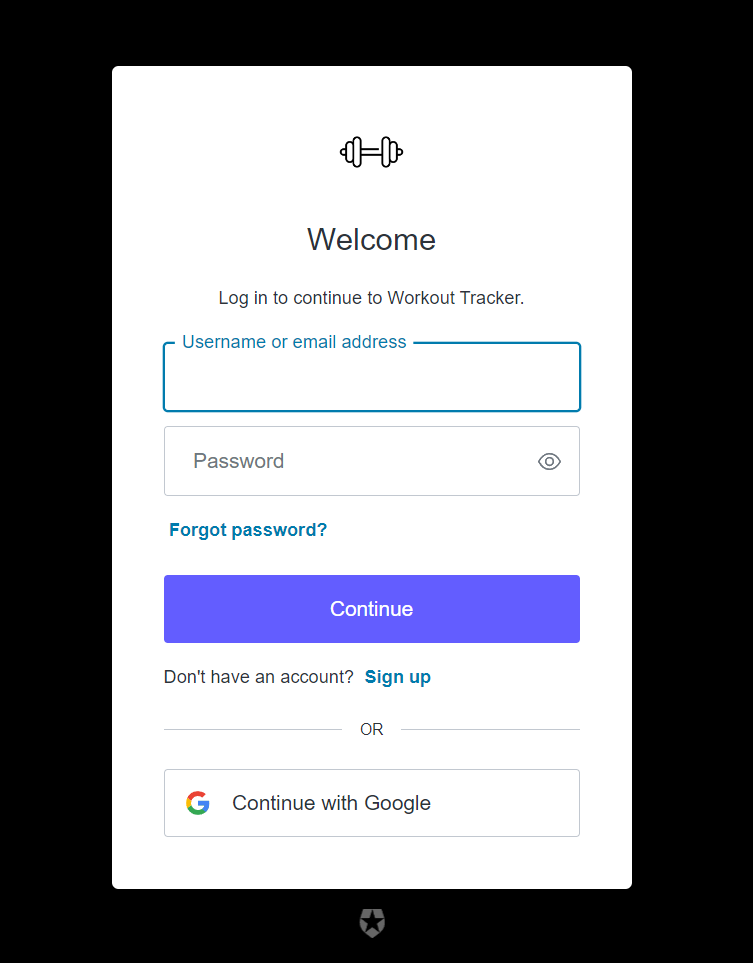
  .catch((err) => {

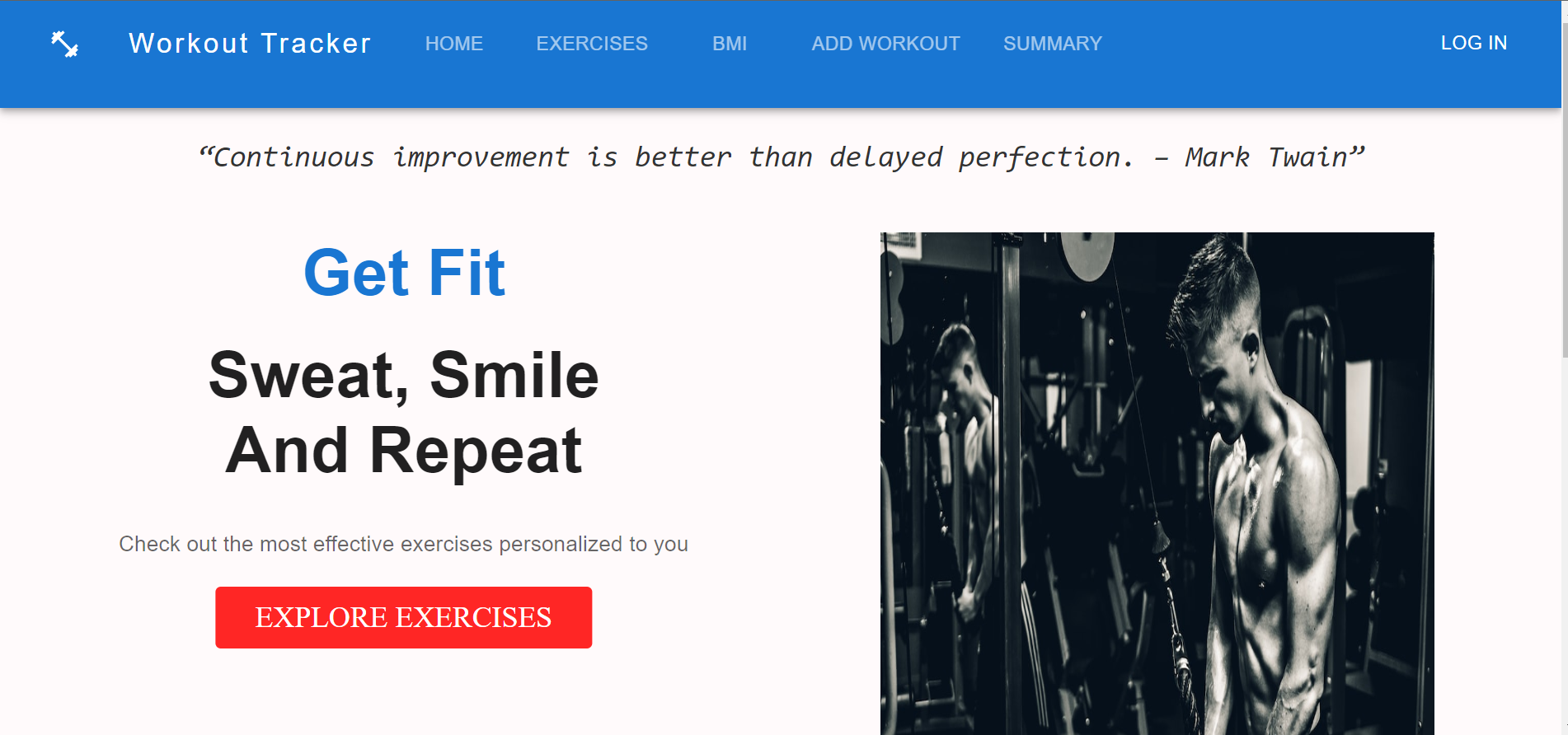
    console.error("Error is", err);

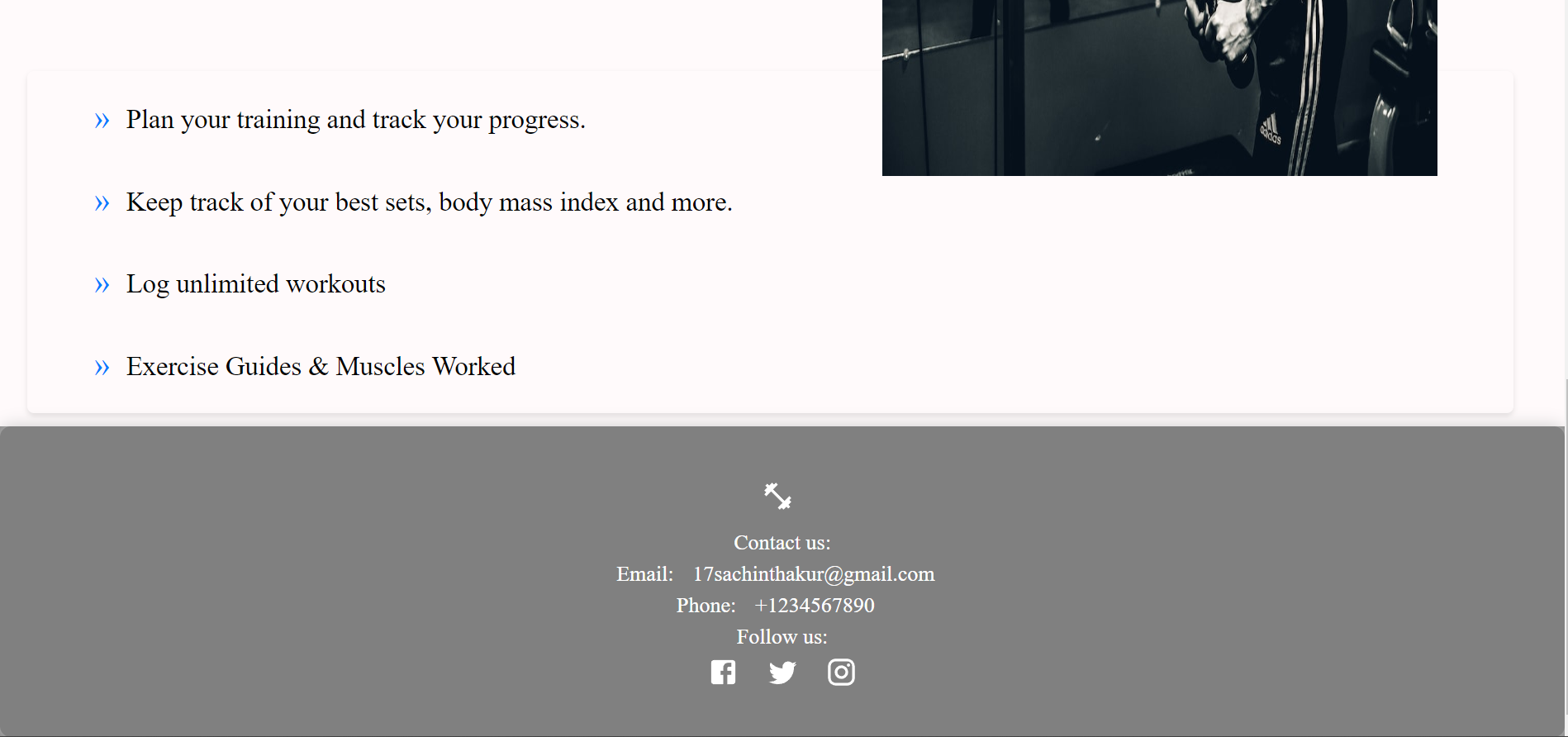
  });

**Result**

****

****

****

****

