FLEX FUSION

WEB MINI-PROJECT

Name: **Prakhar Agrawal**

Reg no**: 22BCE2015**

Course code: **BCSE203E**

Couse Name: Web Programming

Github: https://github.com/prakharagrawal10/FLEX-FUSION.git

Index

Problem Statement

1. **Centralized Fitness Management**:
   - Users struggle with organizing and managing their fitness routines across different platforms or manually.
   - The MERN application aims to provide a centralized platform where users can easily create, customize, and track their workout plans and progress.

2. **Efficient Workout Tracking**:
   - Keeping track of exercises, sets, reps, and progress can be cumbersome and disorganized.
   - The application offers features for logging workouts, recording progress, and visualizing data to help users monitor their performance and stay motivated.

3. **Secure User Authentication**:
   - Privacy and security concerns are paramount, especially when dealing with personal health data.
   - The MERN application prioritizes robust user authentication mechanisms to ensure that users' fitness data is kept secure and confidential, enhancing trust and confidence in the platform.

4. **Enhanced Progress Monitoring:**
   - Enable users to track their workout progress comprehensively, offering features such as exercise logging, performance analytics, and progress visualization
   - To help individuals stay motivated and monitor their fitness journey effectively.

Introduction

F lex Fusion is a dynamic web application designed to revolutionize the way users approach fitness and wellness. With a seamless blend of innovative features and user-friendly interface, Flex Fusion caters to individuals seeking a holistic approach to their health journey. Here's a glimpse of what Flex Fusion offers:

1. **Personalized Fitness Experience**: Flex Fusion empowers users to customize their fitness routines based on their unique goals, preferences, and fitness levels. Whether you're a beginner looking to kickstart your fitness journey or an experienced athlete aiming for peak performance, Flex Fusion provides tailored workout plans and resources to suit your needs.

2. **Comprehensive Wellness Resources**: Beyond traditional workout routines, Flex Fusion encompasses a wide range of wellness resources, including nutrition guidance, mindfulness exercises, and lifestyle tips. By promoting a balanced approach to health, Flex Fusion ensures users can achieve their fitness goals while prioritizing their overall well-being.

3. **Innovative Training Modules**: Flex Fusion offers diverse training modules designed to keep users engaged and motivated. From high-intensity interval training (HIIT) to yoga and meditation sessions, Flex Fusion caters to varied interests and fitness preferences, ensuring there's something for everyone.

4. **Community Engagement and Support**: At Flex Fusion, we believe in the power of community support to fuel motivation and foster accountability. Through interactive forums, group challenges, and social networking features, Flex Fusion cultivates a supportive community where users can connect, share experiences, and inspire each other on their wellness journey.

With Flex Fusion, achieving your fitness and wellness goals has never been more accessible or enjoyable. Join us today and experience the ultimate fusion of fitness, wellness, and community support.

1. ***Frontend***:
    - It's a React application that manages routes using react-router-dom.
    - Components like Navbar, Footer, Home, Login, Signup, WorkoutList, etc., handle different parts of the UI.
    - Contexts like AuthContext and WorkoutsContext manage global state relevant to authentication and workout data.
    - Hooks like useAuthContext, useLogin, useLogout, useSignup, and useWorkoutsContext provide convenient ways to access and manage state within components.
    - Pages like About, ContactUs, WorkoutListPage, etc., provide content for different routes.

2. ***Backend***:
    - It's a Node.js application using Express.js for routing.
    - Controllers handle business logic for user authentication and workout management.
    - Models define the structure of data and include methods for user and workout operations.
    - Middleware like requireAuth ensures routes are protected and require valid authentication tokens.
    - Routes define API endpoints for user authentication (/api/user) and workout management (/api/workouts).

3. ***Database***:
    - MongoDB is used as the database, accessed through Mongoose for schema-based modeling.

TECHNOLOGIES

- ✓ HTML
- ✓ CSS
- ✓ JavaScript
- ✓ ReactJS
- ✓ NodeJS
- ✓ MongoDB
- ✓ ExpressJS

Libraries

- ✓ "date-fns": "^2.28.0",
- ✓ "react": "^18.1.0",
- ✓ "react-dom": "^18.1.0",
- ✓ "react-helmet": "^6.1.0",
- ✓ "react-router-dom": "^6.3.0",
- ✓ "react-scripts": "5.0.1",
- ✓ "react-video-looper": "^1.0.18",
- ✓ "styled-components": "^6.1.8",

- ✓ "bcrypt": "^5.0.1",
- ✓ "dotenv": "^16.0.1",
- ✓ "express": "^4.18.1",
- ✓ "jsonwebtoken": "^8.5.1",
- ✓ "mongoose": "^6.3.6",
- ✓ "validator": "^13.7.0"

Code

Frontend

App.js

```
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom'
import { useAuthContext } from './hooks/useAuthContext'

// pages & components
import Home from './pages/Home'
import WorkoutList from './pages/WorkoutList'
import AddWorkout from './pages/AddWorkout'
import Navbar from './components/Navbar'
import ContactUs from './pages/ContactUs'
import Login from './pages/Login'
import Signup from './pages/Signup'
import Footer from './components/Footer'
import About from './pages/about'
import Faq from './pages/faqpage'
import WorkoutListPage from './pages/WorkoutListPage'



function App() {
  const { user } = useAuthContext()

  return (
    <div className="App">
      <BrowserRouter>
        <Navbar />
        <div className="pages">
          <Routes>
            <Route
              path="/"
              element={user ? <Home /> : <Navigate to="/login" />}
            />
            <Route
              path="/login"
              element={!user ? <Login /> : <Navigate to="/" />}
            />
            <Route
              path="/signup"
              element={!user ? <Signup /> : <Navigate to="/" />}
            />
            <Route path="/contact" element={<ContactUs />} />
            <Route path="/about" element={<About />} />
            <Route path="/workouts" element={<WorkoutList />} />
            <Route path="/add-workout" element={<AddWorkout />} />
            <Route path="/faq" element={<Faq />} />
            <Route path="/list" element={<WorkoutListPage />} />
          </Routes>
        </div>
        <Footer />

      </BrowserRouter>
    </div>
  );
}

export default App;
```

index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { WorkoutsContextProvider } from './context/WorkoutContext'
import { AuthContextProvider } from './context/AuthContext'

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <AuthContextProvider>
      <WorkoutsContextProvider>
        <App />
      </WorkoutsContextProvider>
    </AuthContextProvider>
  </React.StrictMode>
);
```

PAGES

File Name: about.jsx
import React from 'react';

```
const about = () => {
  return (
    <div className="about-me">
      <h2>About Me</h2>
      <p>

        Hi there! My name is [Your Name] and I'm a [Your Profession or Role].
        I enjoy [Your Interests or Hobbies], and I'm passionate about [Your Passion or Interest].
        I have [X] years of experience in [Your Field].
      </p>
      <p>
        In my free time, I like to [Your Activities or Hobbies], and I also enjoy [Other Interests].
        I'm always eager to [Your Goals or Aspirations].
      </p>
      <p>
        Feel free to reach out to me at [Your Email Address] for any inquiries or collaborations.
        You can also connect with me on [Your Social Media Profiles].
      </p>
    </div>
  );
};

export default about;
```

File Name: AddWorkout.jsx

```
// AddWorkoutPage.js
import React from "react";
import WorkoutForm from "../components/WorkoutForm";
import "./addworkout.css";


const AddWorkoutPage = () => {
  return (
    <div className="wrapper">
      <div className="add-workout">
        <h2>Add Workout</h2>
        <WorkoutForm />
      </div>
    </div>
  );
};
```

```
export default AddWorkoutPage;
```

File Name: ContactUs.jsx

```
import React from "react";

const ContactUs = () => {
  return (
    <div className="contact-us">
      <h2>Contact Us</h2>
      <p>If you have any questions or feedback, feel free to reach out to us:</p>
      <ul>
        <li>Email: example@example.com</li>
        <li>Phone: +1234567890</li>
        <li>Address: 123 Example St, City, Country</li>
      </ul>
      <p>You can also fill out the form below:</p>
      <form>
        <label>Name:</label>
        <input type="text" placeholder="Your Name" />
        <label>Email:</label>
        <input type="email" placeholder="Your Email" />
        <label>Message:</label>
        <textarea rows="4" placeholder="Your Message"></textarea><br></br>
        <button type="submit">Send Message</button>
      </form>
    </div>
  );
};

export default ContactUs;
```

File Name: faqpage.jsx

```
import React from 'react';

function faqpage() {
  // Define FAQ data
  const faqs = [
    { question: "What are the benefits of regular exercise?", answer: "Regular exercise can improve cardiovascular health, boost mood, increase energy levels, and help manage weight." },
    { question: "How often should I work out?", answer: "It's recommended to aim for at least 150 minutes of moderate-intensity aerobic activity or 75 minutes of vigorous-intensity activity each week, along with muscle-strengthening exercises on two or more days per week." },
    { question: "What should I eat before and after a workout?", answer: "Before a workout, it's good to consume a balanced meal or snack containing carbohydrates and protein to fuel your exercise. Afterward, focus on replenishing fluids and electrolytes with water and consuming protein and carbohydrates to aid in muscle recovery." },
    { question: "How can I prevent injuries during exercise?", answer: "To prevent injuries, it's essential to warm up before exercising, use proper form and technique, gradually increase the intensity of your workouts, listen to your body, and incorporate rest days into your routine." },
    { question: "Is it okay to exercise if I'm sore?", answer: "It's generally safe to exercise when experiencing mild soreness, but it's essential to listen to your body and avoid overexertion. Consider focusing on different muscle groups or engaging in low-impact activities." },
  ];

  return (
    <div className='contact-us'>
      <h1>Frequently Asked Questions</h1>
      <ul>
        {faqs.map((faq, index) => (
          <li key={index}>
            <h3>{faq.question}</h3>
            <p>{faq.answer}</p>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default faqpage;
```

File Name: Home.jsx

```
import React, { useEffect, useState } from "react";
import { Link } from "react-router-dom";
import "./home.css";

const Home = () => {
  const [showButtons, setShowButtons] = useState(false);

  useEffect(() => {
    const introTimeout = setTimeout(() => {
      setShowButtons(true);
    }, 1000);

    return () => clearTimeout(introTimeout);
  }, []);


  // Inside the Home component
const featuredWorkouts = [
  { name: "Insanity", description: "High-intensity interval training program designed to push your
limits" },
  { name: "CrossFit", description: "Constantly varied functional movements performed at high
intensity" },
  { name: "P90X", description: "Total-body strength and conditioning program utilizing muscle
confusion" }
  // Add more workout objects as needed
];

  return (
    <div className="home1">
      <div className="test">
            <div className="intro">
              <h1>Welcome to Flex Fusion</h1>
            </div>
            <div className="description">
              <p>Your go-to app for tracking and adding workouts!</p>
            </div>
            {showButtons && (
              <div className="buttons">
                <div>
                  <Link to="/workouts">
                    <button className="btn">View Workouts</button>
                  </Link>
                </div>
                <div>
                  <Link to="/add-workout">
                    <button className="btn">Add Workout</button>
                  </Link>
                </div>
                <div>
                  <Link to="/faq">
                    <button className="btn">FAQ's</button>
                  </Link>
                </div>
                <div>
                  <Link to="/list">
                    <button className="btn">Workouts</button>
                  </Link>
```

```
          </div>
        </div>
      )}
    </div>

  {/* Featured Workouts */}
  <div className="featured-workouts">
  <h2>Featured Workouts</h2>
  <div className="scroll-box">
    {featuredWorkouts.map((workout, index) => (
      <div key={index} className="workout-item">
        <h3>{workout.name}</h3>
        <p>{workout.description}</p>
      </div>
    ))}
  </div>
 </div>
 </div>
 );
}

export default Home;
```

File Name: Login.jsx

```
import { useState } from "react";
import { useLogin } from "../hooks/useLogin";
import "./login.css";

const Login = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const { login, error, isLoading } = useLogin();

  const handleSubmit = async (e) => {
    e.preventDefault();

    await login(email, password);
  };

  return (
    <form className="login" onSubmit={handleSubmit}>
      <h3>Log In</h3>
      <label>Email address:</label>
      <input
        type="email"
        onChange={(e) => setEmail(e.target.value)}
        value={email}
      />
      <label>Password:</label>
      <input
        type="password"
        onChange={(e) => setPassword(e.target.value)}
        value={password}
      />

      <button className="login-button" disabled={isLoading}>Log in</button>
      {error && <div className="error">{error}</div>}
    </form>
  );
};

export default Login;
```

File Name: Signup.jsx

```
import { useState } from "react"
import { useSignup } from "../hooks/useSignup"
import './signup.css'

const Signup = () => {
  const [email, setEmail] = useState('')
  const [password, setPassword] = useState('')
  const {signup, error, isLoading} = useSignup()

  const handleSubmit = async (e) => {
    e.preventDefault()

    await signup(email, password)
  }

  return (
    <form className="signup" onSubmit={handleSubmit}>
      <h3>Sign Up</h3>

      <label>Email address:</label>
      <input
```

```
        type="email"
        onChange={(e) => setEmail(e.target.value)}
        value={email}
      />
      <label>Password:</label>
      <input
        type="password"
        onChange={(e) => setPassword(e.target.value)}
        value={password}
      />

      <button disabled={isLoading}>Sign up</button>
      {error && <div className="error">{error}</div>}
    </form>
  )
}

export default Signup
```

File Name: WorkoutList.jsx

```
import { useEffect }from 'react'
import { useWorkoutsContext } from "../hooks/useWorkoutsContext"
import { useAuthContext } from "../hooks/useAuthContext"

// components
import WorkoutDetails from '../components/WorkoutDetails'

const WorkoutList = () => {
  const {workouts, dispatch} = useWorkoutsContext()
  const {user} = useAuthContext()

  useEffect(() => {
    const fetchWorkouts = async () => {
      const response = await fetch('/api/workouts', {
        headers: {'Authorization': `Bearer ${user.token}`},
      })
      const json = await response.json()

      if (response.ok) {
        dispatch({type: 'SET_WORKOUTS', payload: json})
      }
    }

    if (user) {
      fetchWorkouts()
    }
  }, [dispatch, user])

  return (
    <div className="home">
      <div className="workouts">
        {workouts && workouts.map((workout) => (
          <WorkoutDetails key={workout._id} workout={workout} />
        ))}
      </div>
    </div>
  )
}

export default WorkoutList
```

File Name: WorkoutListPage.jsx

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

function WorkoutListPage() {
  // Define workout data
  const workouts = [
    { id: "crunches", name: "Crunches", videoId: "Xyd_fa5zoEU" },
    { id: "curls", name: "Curls", videoId: "ykJmrZ5v0Oo" },
    { id: "squats", name: "Squats", videoId: "MVMNk0HiTMg" },
  ];

  return (
    <div className='contact-us'>
      <h1>Workouts</h1>
      <ul>
        {workouts.map(workout => (
          <li key={workout.id}>
            <Link to={`/workout/${workout.id}`}>
              <h3>{workout.name}</h3>
            </Link>
            <div>
              <iframe
                width="560"
                height="315"
                src={`https://www.youtube.com/embed/${workout.videoId}`}
                title={workout.name}
                frameBorder="0"
                allowFullScreen
              ></iframe>
            </div>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default WorkoutListPage;
```

HOOKS

File Name: useAuthContext.jsx

```
import { AuthContext } from "../context/AuthContext"
import { useContext } from "react"

export const useAuthContext = () => {
  const context = useContext(AuthContext)

  if(!context) {
    throw Error('useAuthContext must be used inside an AuthContextProvider')
  }

  return context
}
```

File Name: useLogin.jsx

```
import { useState } from 'react'
import { useAuthContext } from './useAuthContext'

export const useLogin = () => {
  const [error, setError] = useState(null)
  const [isLoading, setIsLoading] = useState(null)
  const { dispatch } = useAuthContext()

  const login = async (email, password) => {
    setIsLoading(true)
    setError(null)
    const response = await fetch('/api/user/login', {
      method: 'POST',
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify({ email, password })
    })
    const json = await response.json()

    if (!response.ok) {
      setIsLoading(false)
      setError(json.error)
    }
    if (response.ok) {
      // save the user to local storage
      localStorage.setItem('user', JSON.stringify(json))

      // update the auth context
      dispatch({type: 'LOGIN', payload: json})

      // update loading state
      setIsLoading(false)
    }
  }

  return { login, isLoading, error }
}
```

File Name: useLogout.jsx

```
import { useAuthContext } from './useAuthContext'
import { useWorkoutsContext } from './useWorkoutsContext'

export const useLogout = () => {
  const { dispatch } = useAuthContext()
  const { dispatch: dispatchWorkouts } = useWorkoutsContext()

  const logout = () => {
    // remove user from storage
    localStorage.removeItem('user')

    // dispatch logout action
    dispatch({ type: 'LOGOUT' })
    dispatchWorkouts({ type: 'SET_WORKOUTS', payload: null })
  }
  return { logout }
}
```

File Name: useSignup.jsx

```
import { useState } from 'react'
import { useAuthContext } from './useAuthContext'

export const useSignup = () => {
  const [error, setError] = useState(null)
  const [isLoading, setIsLoading] = useState(null)
  const { dispatch } = useAuthContext()

  const signup = async (email, password) => {
    setIsLoading(true)
    setError(null)

    const response = await fetch('/api/user/signup', {
      method: 'POST',
      headers: {'Content-Type': 'application/json'},
      body: JSON.stringify({ email, password })
    })
    const json = await response.json()

    if (!response.ok) {
      setIsLoading(false)
      setError(json.error)
    }
    if (response.ok) {
      // save the user to local storage
      localStorage.setItem('user', JSON.stringify(json))

      // update the auth context
      dispatch({type: 'LOGIN', payload: json})

      // update loading state
      setIsLoading(false)
    }
  }

  return { signup, isLoading, error }
}
```

File Name: useWorkoutsContext.jsx

```
import { WorkoutsContext } from '../context/WorkoutContext'
import { useContext } from 'react'

export const useWorkoutsContext = () => {
  const context = useContext(WorkoutsContext)

  if (!context) {
    throw Error('useWorkoutsContext must be used inside an WorkoutsContextProvider')
  }

  return context
}
```

CONTEXT

File Name: AuthContext.jsx

```
import { createContext, useReducer, useEffect } from 'react'

export const AuthContext = createContext()

export const authReducer = (state, action) => {
  switch (action.type) {
    case 'LOGIN':
      return { user: action.payload }
    case 'LOGOUT':
      return { user: null }
    default:
      return state
  }
}

export const AuthContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(authReducer, {
    user: null
  })

  useEffect(() => {
    const user = JSON.parse(localStorage.getItem('user'))

    if (user) {
      dispatch({ type: 'LOGIN', payload: user })
    }
```

```
 }, [])

 console.log('AuthContext state:', state)

 return (
   <AuthContext.Provider value={{ ...state, dispatch }}>
     { children }
   </AuthContext.Provider>
 )

}
```

File Name: WorkoutContext.jsx

```
import { createContext, useReducer } from 'react'

export const WorkoutsContext = createContext()

export const workoutsReducer = (state, action) => {
  switch (action.type) {
    case 'SET_WORKOUTS':
      return {
        workouts: action.payload
      }
    case 'CREATE_WORKOUT':
      return {
        workouts: [action.payload, ...state.workouts]
      }
    case 'DELETE_WORKOUT':
      return {
        workouts: state.workouts.filter((w) => w._id !== action.payload._id)
      }
    default:
      return state
  }
}

export const WorkoutsContextProvider = ({ children }) => {
  const [state, dispatch] = useReducer(workoutsReducer, {
    workouts: null
  })

  return (
    <WorkoutsContext.Provider value={{...state, dispatch}}>
      { children }
    </WorkoutsContext.Provider>
  )
}
```

COMPONENTS

File Name: Footer.jsx

```
// Footer.js
import React from "react";
import gmailIcon from "../assets/gmail.png";
import facebookIcon from "../assets/facebook.png";
import instagramIcon from "../assets/instagram.png";
import youtubeIcon from "../assets/youtube.png";
import "./footer.css";

const Footer = () => {
  return (
    <footer>
      <div className="container">
        <p>&copy; 2024 FLEX FUSION. All rights reserved.</p>
        <div className="social-icons">
          <a href="https://mail.google.com/mail/u/0/#inbox">
            <img src={gmailIcon} alt="Gmail" />
          </a>
          <a href="https://www.facebook.com">
            <img src={facebookIcon} alt="Facebook" />
          </a>
          <a href="https://www.instagram.com">
            <img src={instagramIcon} alt="Instagram" />
          </a>
          <a href="https://www.youtube.com">
            <img src={youtubeIcon} alt="YouTube" />
          </a>
        </div>
      </div>
    </footer>
  );
};

export default Footer;
```

File Name: Navbar.jsx

```
import { Link } from 'react-router-dom'
import { useLogout } from '../hooks/useLogout'
import { useAuthContext } from '../hooks/useAuthContext'
import './navbar.css';

const Navbar = () => {
  const { logout } = useLogout()
  const { user } = useAuthContext()

  const handleClick = () => {
    logout()
  }

  return (
    <header>
      <div className="container">
        <Link to="/">
          <div className='logoname'>
            FLEX FUSION
          </div>
        </Link>
        <nav>
          <div className="nav-links">
            {user && (
              <div>
                <span>{user.email}</span>
                <button className="nav-btn" onClick={handleClick}>Log out</button>
              </div>
            )}
            {!user && (
              <div>
                <Link to="/login" className="nav-btn">Login</Link>
                <Link to="/signup" className="nav-btn">Signup</Link>
              </div>
            )}
            <Link to="/contact" className="nav-btn">Contact Us</Link>
            <Link to="/about" className="nav-btn">About Me</Link>
          </div>
        </nav>
      </div>
    </header>
  )
}

export default Navbar;
```

File Name: WorkoutDetails.jsx

```
import { useWorkoutsContext } from '../hooks/useWorkoutsContext'
import './workoutdetails.css';
// date fns
import formatDistanceToNow from 'date-fns/formatDistanceToNow'

const WorkoutDetails = ({ workout }) => {
  const { dispatch } = useWorkoutsContext()

  const handleClick = async () => {
    const response = await fetch('/api/workouts/' + workout._id, {
      method: 'DELETE'
    })
    const json = await response.json()

    if (response.ok) {
      dispatch({type: 'DELETE_WORKOUT', payload: json})
    }
  }

  return (
    <div className="workout-details">
      <h4>{workout.title}</h4>
      <p><strong>Load (kg): </strong>{workout.load}</p>
      <p><strong>Reps: </strong>{workout.reps}</p>
      <p>{formatDistanceToNow(new Date(workout.createdAt), { addSuffix: true })}</p>
      <span className="material-symbols-outlined" onClick={handleClick}>delete</span>
    </div>
  )
}

export default WorkoutDetails
```

File Name: WorkoutForm.jsx

```
import { useState } from "react"
import { useWorkoutsContext } from "../hooks/useWorkoutsContext"
import { useAuthContext } from '../hooks/useAuthContext'
```

```
const WorkoutForm = () => {
  const { dispatch } = useWorkoutsContext()
  const { user } = useAuthContext()

  const [title, setTitle] = useState('')
  const [load, setLoad] = useState('')
  const [reps, setReps] = useState('')
  const [error, setError] = useState(null)
  const [emptyFields, setEmptyFields] = useState([])

  const handleSubmit = async (e) => {
    e.preventDefault()

    if (!user) {
      setError('You must be logged in')
      return
    }

    const workout = {title, load, reps}

    const response = await fetch('/api/workouts', {
      method: 'POST',
      body: JSON.stringify(workout),
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${user.token}`
      }
    })
    const json = await response.json()

    if (!response.ok) {
      setError(json.error)
      setEmptyFields(json.emptyFields)
    }
    if (response.ok) {
      setTitle('')
      setLoad('')
      setReps('')
      setError(null)
      setEmptyFields([])
      dispatch({type: 'CREATE_WORKOUT', payload: json})
    }
  }
  return (
    <form className="create" onSubmit={handleSubmit}>
      <h3>Add a New Workout</h3>

      <label>Excersize Title:</label>
      <input
        type="text"
        onChange={(e) => setTitle(e.target.value)}
        value={title}
        className={emptyFields.includes('title') ? 'error' : ''}
      />

      <label>Load (in kg):</label>
      <input
        type="number"
        onChange={(e) => setLoad(e.target.value)}
        value={load}
        className={emptyFields.includes('load') ? 'error' : ''}
      />

      <label>Reps:</label>
      <input
        type="number"
        onChange={(e) => setReps(e.target.value)}
        value={reps}
        className={emptyFields.includes('reps') ? 'error' : ''}
      />

      <button>Add Workout</button>
      {error && <div className="error">{error}</div>}
    </form>
  )
}

export default WorkoutForm
```

BACKEND

Server.js

```
require('dotenv').config()

const express = require('express')
const mongoose = require('mongoose')
const workoutRoutes = require('./routes/workouts')
const userRoutes = require('./routes/user')

// express app
const app = express()

// middleware
app.use(express.json())

app.use((req, res, next) => {
  console.log(req.path, req.method)
  next()
})

// routes
app.use('/api/workouts', workoutRoutes)
app.use('/api/user', userRoutes)

// connect to db
mongoose.connect(process.env.MONGO_URI)
  .then(() => {
    // listen for requests
    app.listen(process.env.PORT, () => {
      console.log('connected to db & listening on port', process.env.PORT)
    })
  })
  .catch((error) => {
    console.log(error)
  })
```

Controllers

userController.js

```
const User = require('../models/userModel')
const jwt = require('jsonwebtoken')

const createToken = (_id) => {
  return jwt.sign({_id}, process.env.SECRET, { expiresIn: '3d' })
}

// login a user
const loginUser = async (req, res) => {
  const {email, password} = req.body
  try {
    const user = await User.login(email, password)

    // create a token
    const token = createToken(user._id)

    res.status(200).json({email, token})
  } catch (error) {
    res.status(400).json({error: error.message})
  }
}

// signup a user
const signupUser = async (req, res) => {
  const {email, password} = req.body

  try {
    const user = await User.signup(email, password)

    // create a token
    const token = createToken(user._id)

    res.status(200).json({email, token})
  } catch (error) {
    res.status(400).json({error: error.message})
  }
}

module.exports = { signupUser, loginUser }
```

workoutController.js

```
const Workout = require('../models/workoutModel')
const mongoose = require('mongoose')

// get all workouts
const getWorkouts = async (req, res) => {
  const user_id = req.user._id

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

  res.status(200).json(workouts)
}

// get a single workout
const getWorkout = async (req, res) => {
  const { id } = req.params

  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 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 doc to db
  try {
    const user_id = req.user._id
    const workout = await Workout.create({title, load, reps, user_id})
    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(404).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(404).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
}
```

requireAuth

```
const jwt = require('jsonwebtoken');
const User = require('../models/userModel');

const requireAuth = async (req, res, next) => {
  // Extract the URL path from the request
  const url = req.originalUrl;
  // Check if the request is for deleting a workout
  if (req.method === 'DELETE' && url.includes('/api/workouts/')) {
    // Skip authorization check for deleting workouts
    return next();
  }

  // Verify user is authenticated
  const { authorization } = req.headers;

  if (!authorization) {
    return res.status(401).json({ error: 'Authorization token required' });
  }

  const token = authorization.split(' ')[1];

  try {
    const { _id } = jwt.verify(token, process.env.SECRET);

    req.user = await User.findOne({ _id }).select('_id');
    next();

  } catch (error) {
    console.log(error);
    res.status(401).json({ error: 'Request is not authorized' });
  }
};

module.exports = requireAuth;
```

userModel

```
const mongoose = require('mongoose')
const bcrypt = require('bcrypt')
const validator = require('validator')

const Schema = mongoose.Schema

const userSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  }
})

// static signup method
userSchema.statics.signup = async function(email, password) {

  // validation
  if (!email || !password) {
    throw Error('All fields must be filled')
  }
  if (!validator.isEmail(email)) {
    throw Error('Email not valid')
  }
  if (!validator.isStrongPassword(password)) {
    throw Error('Password not strong enough')
  }

  const exists = await this.findOne({ email })

  if (exists) {
    throw Error('Email already in use')
  }

  const salt = await bcrypt.genSalt(10)
  const hash = await bcrypt.hash(password, salt)

  const user = await this.create({ email, password: hash })

  return user
}

// static login method
userSchema.statics.login = async function(email, password) {

  if (!email || !password) {
    throw Error('All fields must be filled')
  }

  const user = await this.findOne({ email })
  if (!user) {
    throw Error('Incorrect email')
  }

  const match = await bcrypt.compare(password, user.password)
  if (!match) {
    throw Error('Incorrect password')
  }

  return user
}

module.exports = mongoose.model('User', userSchema)
```

workoutModek

```
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
  },
  user_id: {
    type: String,
    required: true
  }
}, { timestamps: true })

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

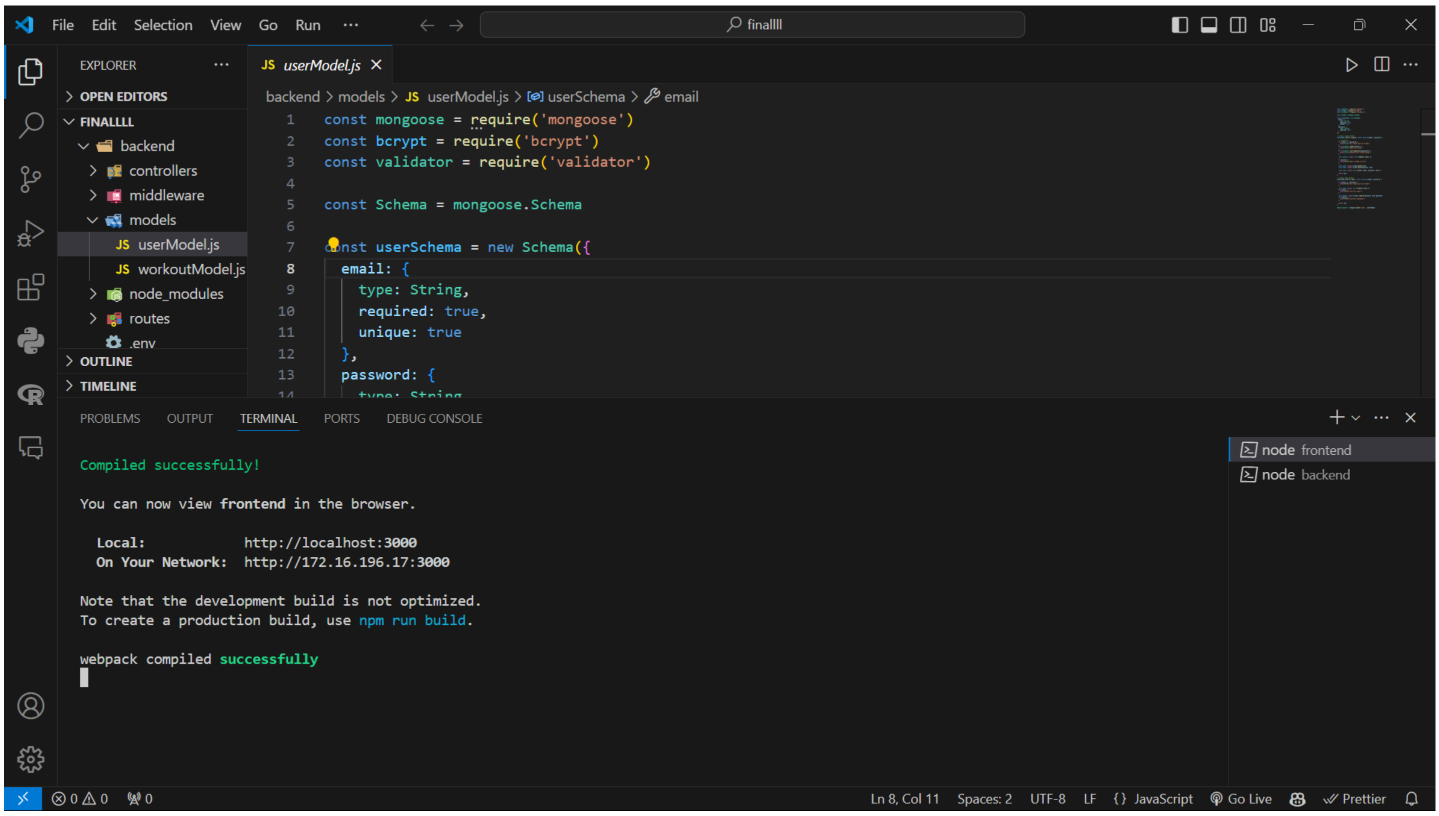
User.js

```
const express = require('express')

// controller functions
const { loginUser, signupUser } = require('../controllers/userController')
const router = express.Router()

// login route
router.post('/login', loginUser)

// signup route
router.post('/signup', signupUser)

module.exports = router
```

workout.js

```
const express = require('express')
const {
  createWorkout,
  getWorkouts,
  getWorkout,
  deleteWorkout,
  updateWorkout
} = require('../controllers/workoutController')
const requireAuth = require('../middleware/requireAuth')

const router = express.Router()

// require auth for all workout routes
router.use(requireAuth)

// 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
```

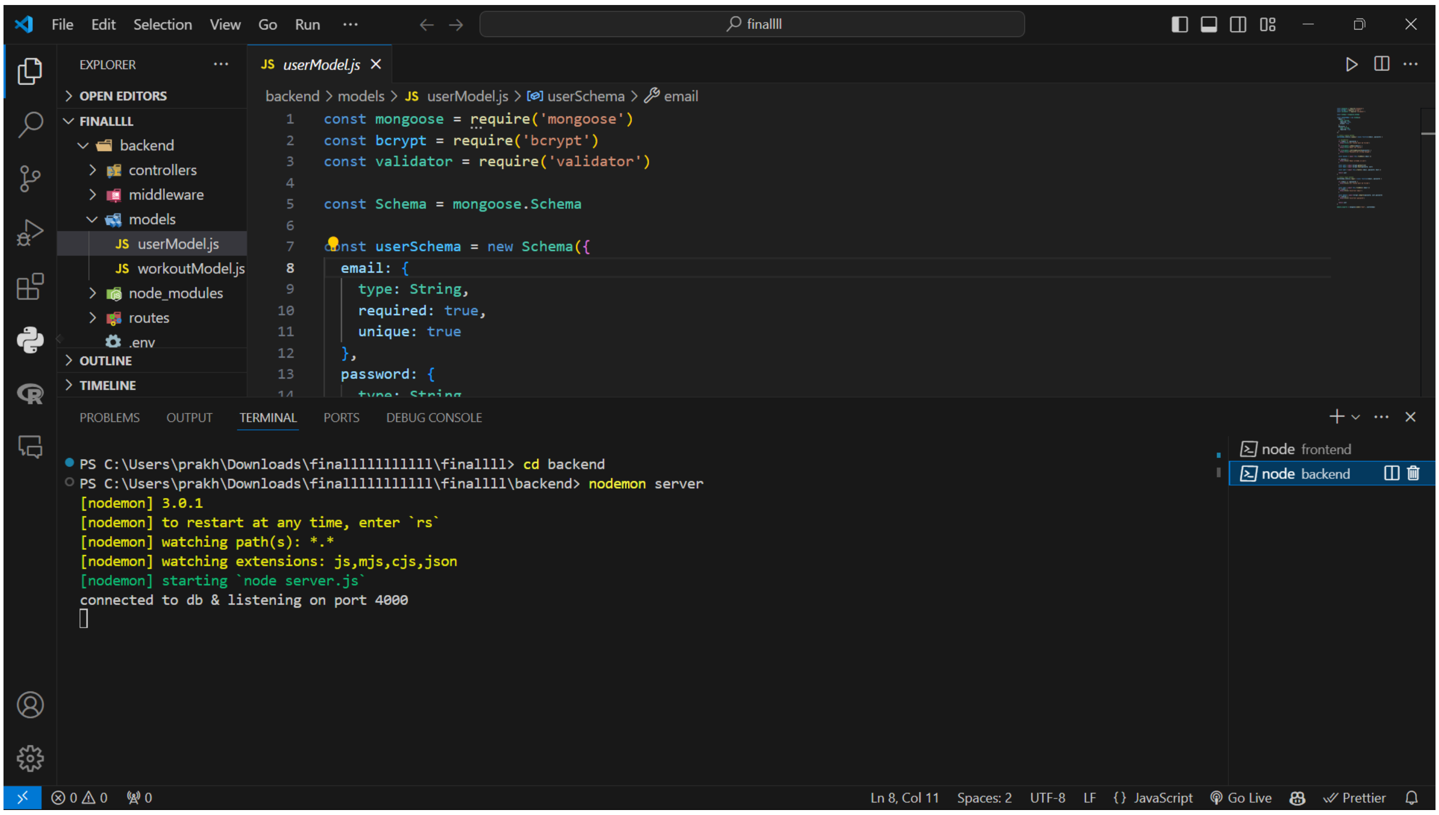
SampleOutput



File Edit Selection View Go Run ···
finallll
EXPLORER
userModel.js
OPEN EDITORS
FINALLLL
backend
controllers
middleware
models
userModel.js
workoutModel.js
node_modules
routes
.env
OUTLINE
TIMELINE
backend > models > userModel.js > userSchema > email
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
node frontend
node backend
Ln 8, Col 11 Spaces: 2 UTF-8 LF {} JavaScript Go Live Prettier

```
const mongoose = require('mongoose')
const bcrypt = require('bcrypt')
const validator = require('validator')

const Schema = mongoose.Schema

const userSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String
```

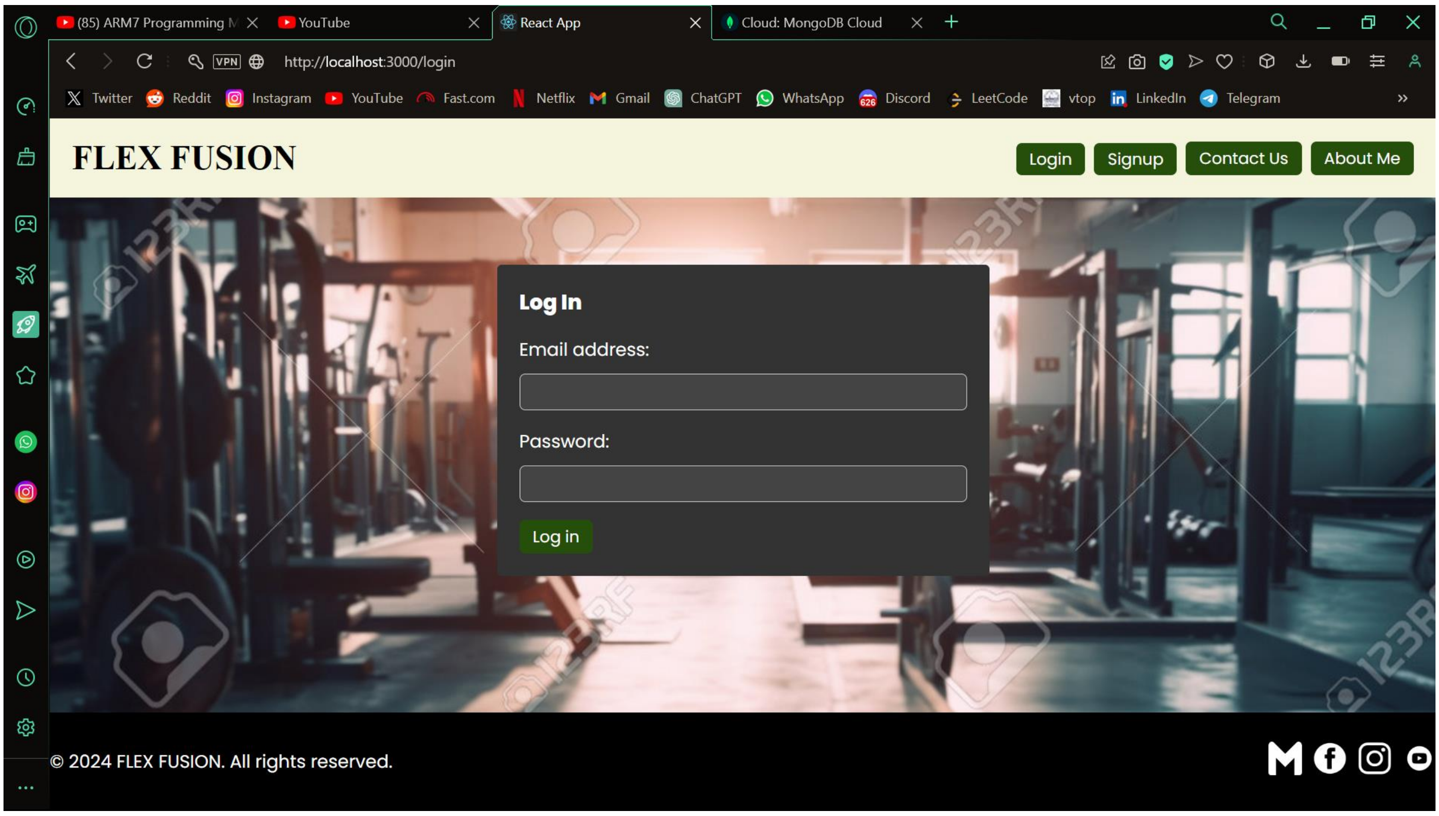
```
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://172.16.196.17:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```



File Edit Selection View Go Run ···
finallll
EXPLORER
userModel.js
OPEN EDITORS
FINALLLL
backend
controllers
middleware
models
userModel.js
workoutModel.js
node_modules
routes
.env
OUTLINE
TIMELINE
backend > models > userModel.js > userSchema > email
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
node frontend
node backend
Ln 8, Col 11 Spaces: 2 UTF-8 LF {} JavaScript Go Live Prettier

```
const mongoose = require('mongoose')
const bcrypt = require('bcrypt')
const validator = require('validator')

const Schema = mongoose.Schema

const userSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String
```

```
PS C:\Users\prakh\Downloads\finalllllllllll\finallll> cd backend
PS C:\Users\prakh\Downloads\finalllllllllll\finallll\backend> nodemon server
[nodemon] 3.0.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
connected to db & listening on port 4000
```



FLEX FUSION

Login Signup Contact Us About Me

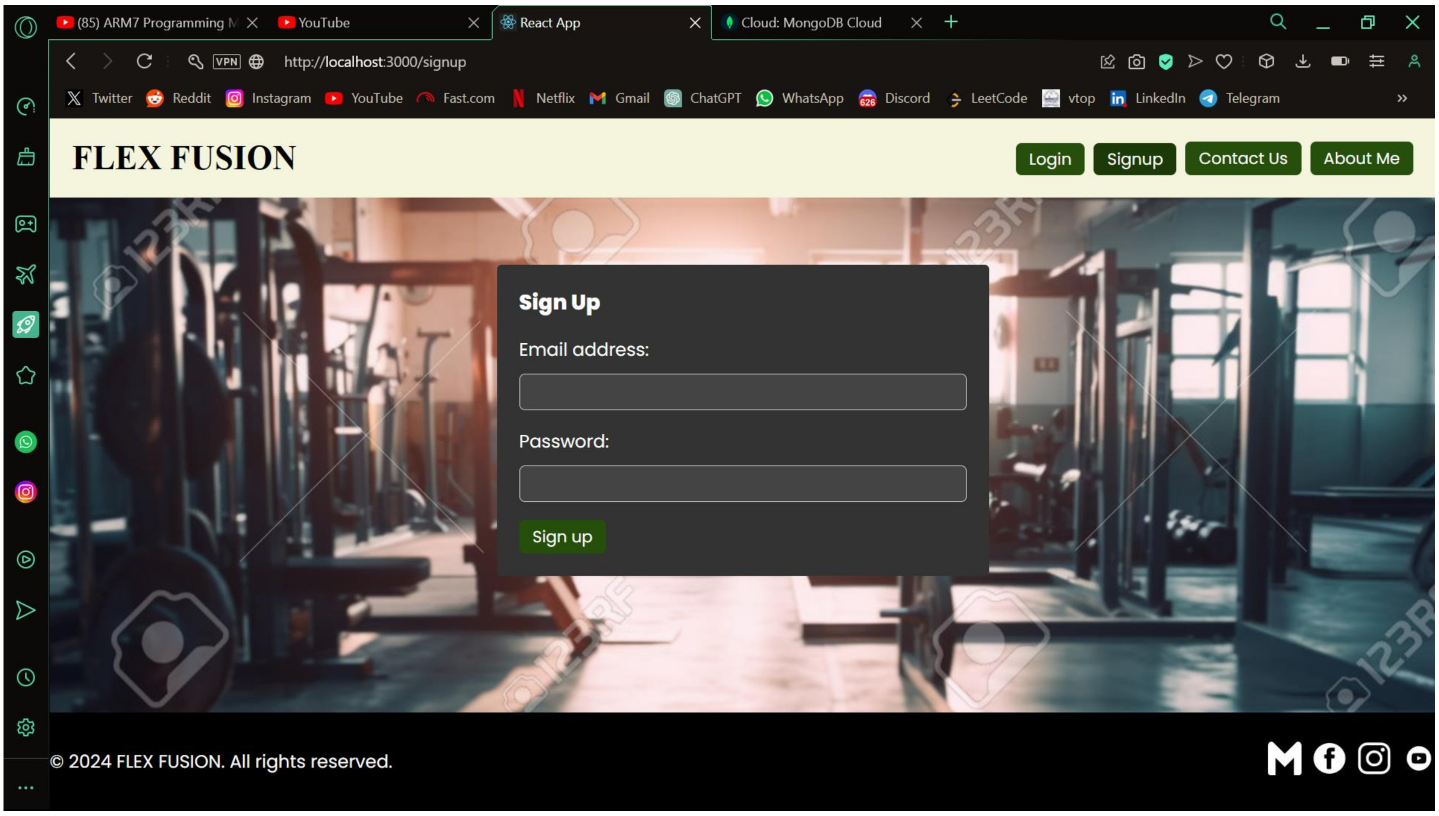
Log In

Email address:

Password:

Log in

© 2024 FLEX FUSION. All rights reserved.



FLEX FUSION

Login Signup Contact Us About Me

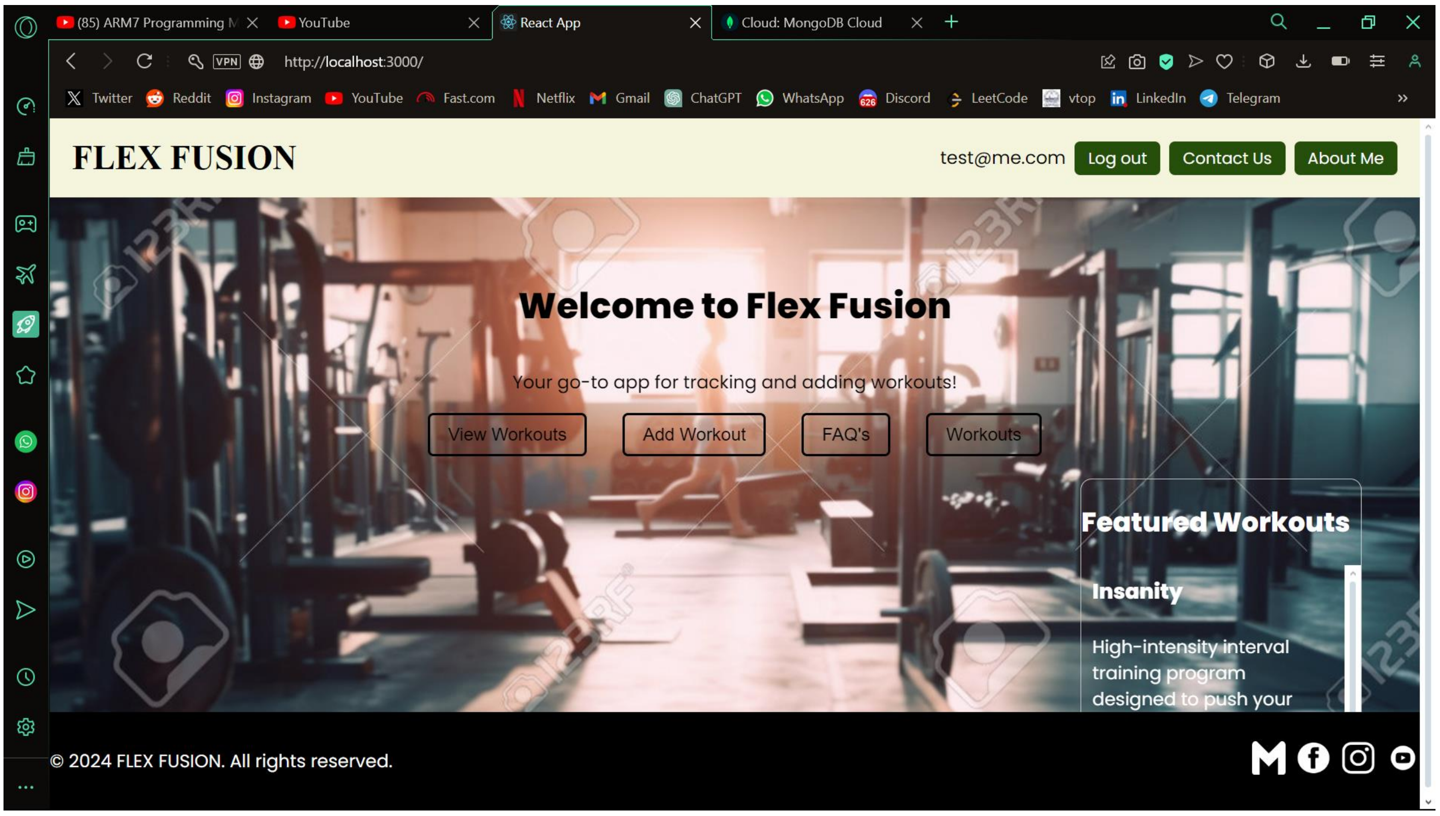
Sign Up

Email address:

Password:

Sign up

© 2024 FLEX FUSION. All rights reserved.



FLEX FUSION

test@me.com Log out Contact Us About Me

Welcome to Flex Fusion

Your go-to app for tracking and adding workouts!

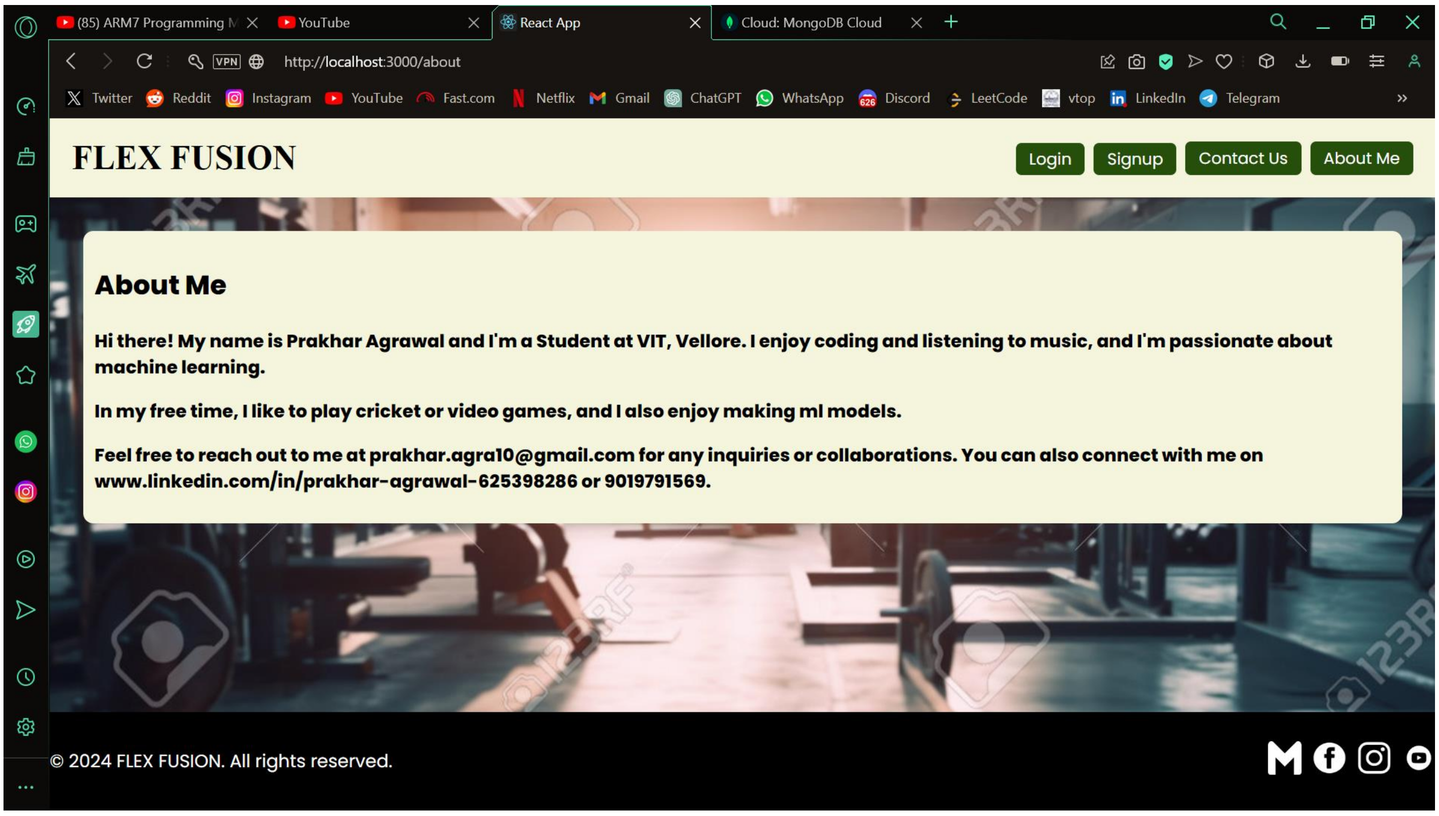
View Workouts Add Workout FAQ's Workouts

Featured Workouts

Insanity

High-intensity interval training program designed to push your

© 2024 FLEX FUSION. All rights reserved.



FLEX FUSION

Login Signup Contact Us About Me

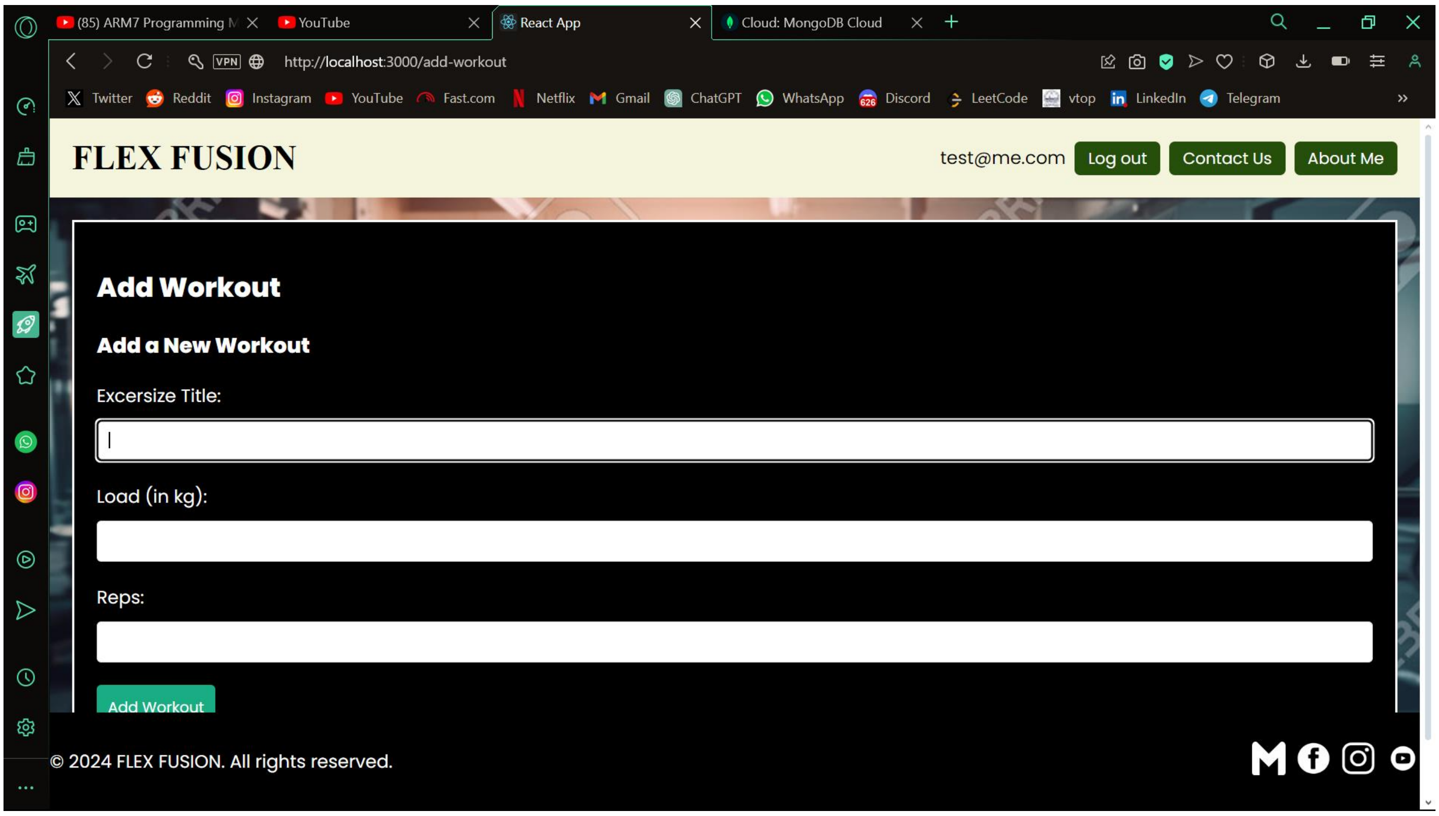
About Me

Hi there! My name is Prakhar Agrawal and I'm a Student at VIT, Vellore. I enjoy coding and listening to music, and I'm passionate about machine learning.

In my free time, I like to play cricket or video games, and I also enjoy making ml models.

Feel free to reach out to me at prakhar.agra10@gmail.com for any inquiries or collaborations. You can also connect with me on www.linkedin.com/in/prakhar-agrawal-625398286 or 9019791569.

© 2024 FLEX FUSION. All rights reserved.



FLEX FUSION

test@me.com Log out Contact Us About Me

Add Workout

Add a New Workout

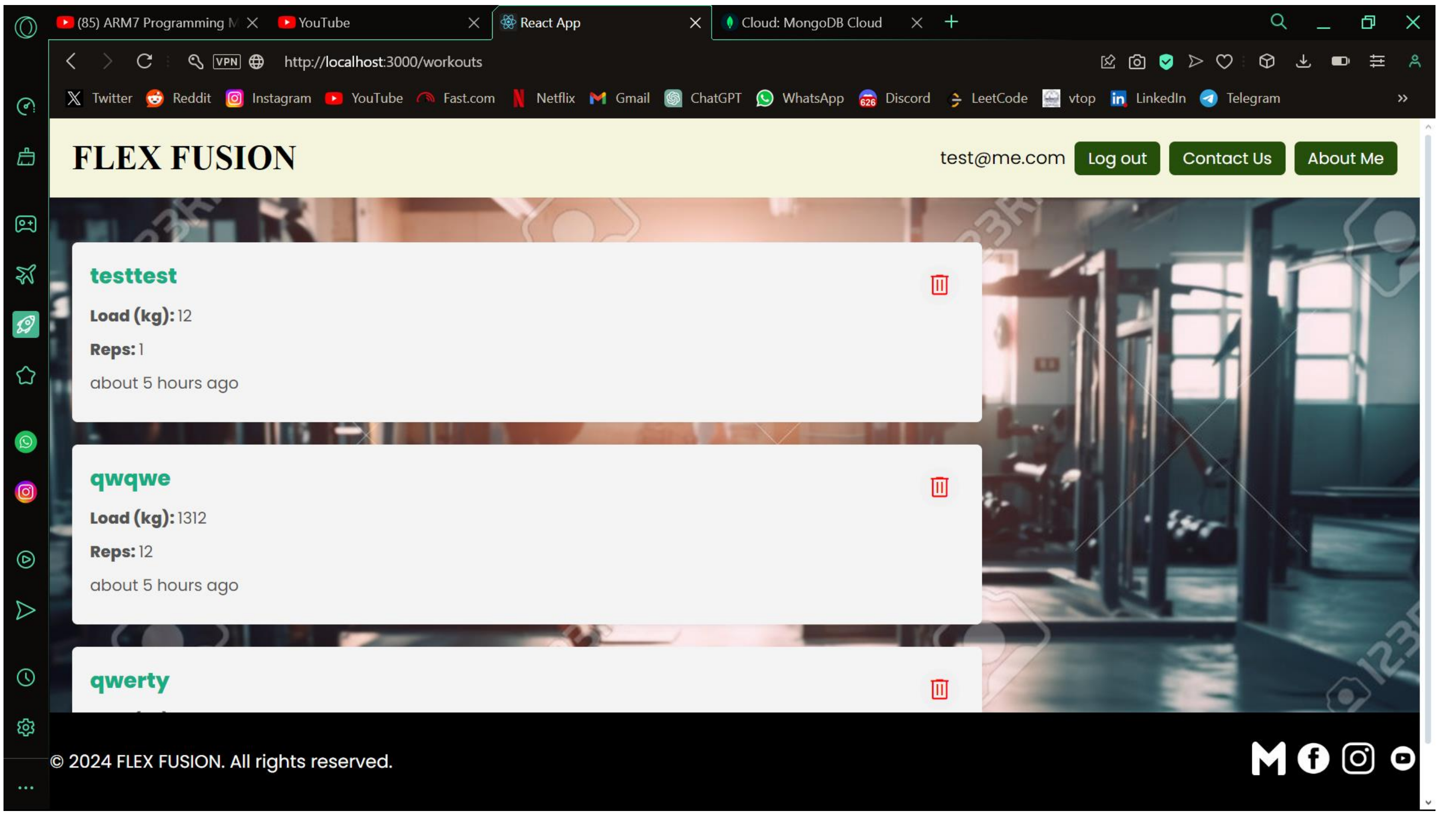
Excersize Title:

Load (in kg):

Reps:

Add Workout

© 2024 FLEX FUSION. All rights reserved.



FLEX FUSION

test@me.com Log out Contact Us About Me

testtest

Load (kg): 12

Reps: 1

about 5 hours ago

qwqwe

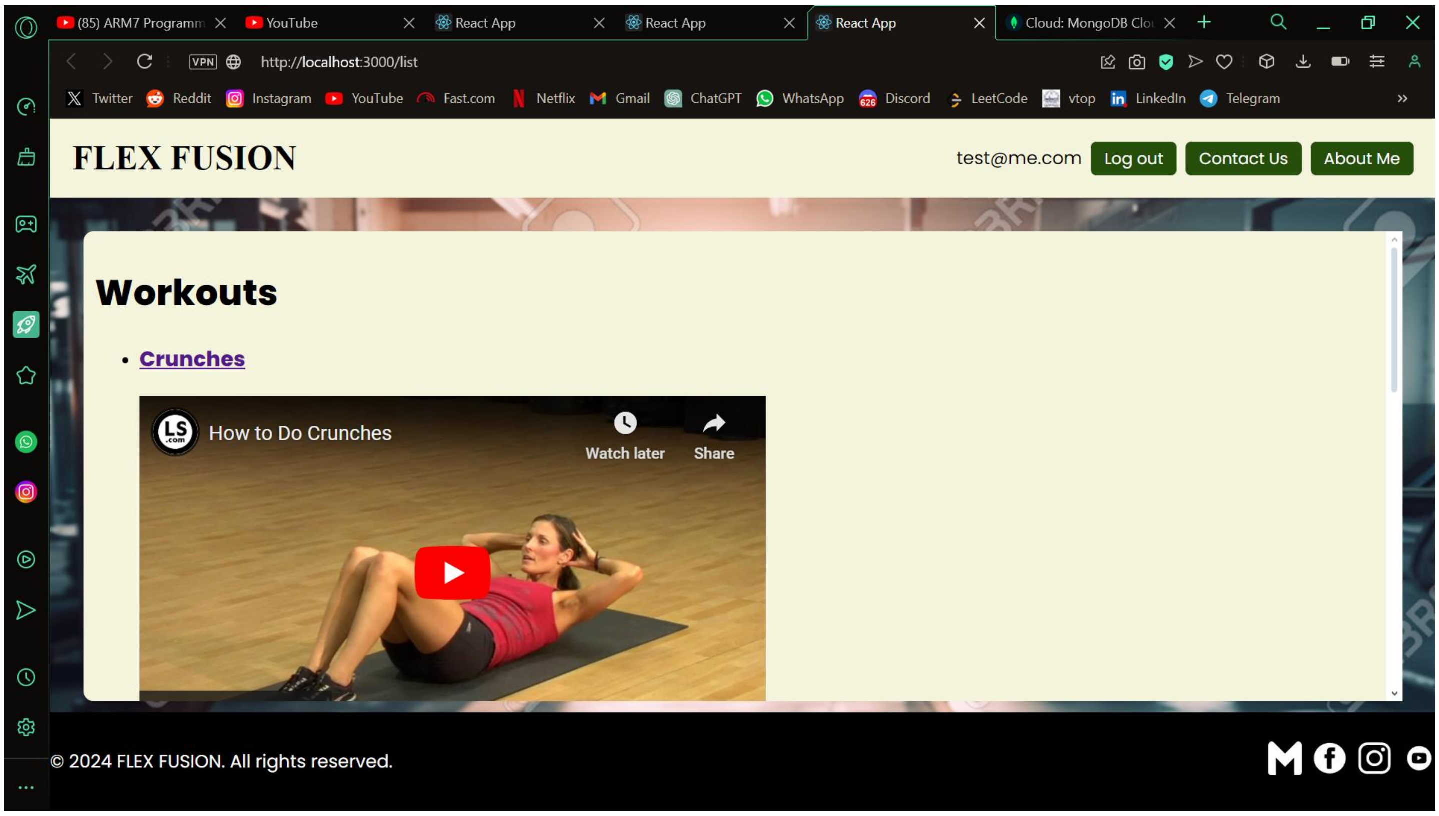
Load (kg): 1312

Reps: 12

about 5 hours ago

qwerty

© 2024 FLEX FUSION. All rights reserved.



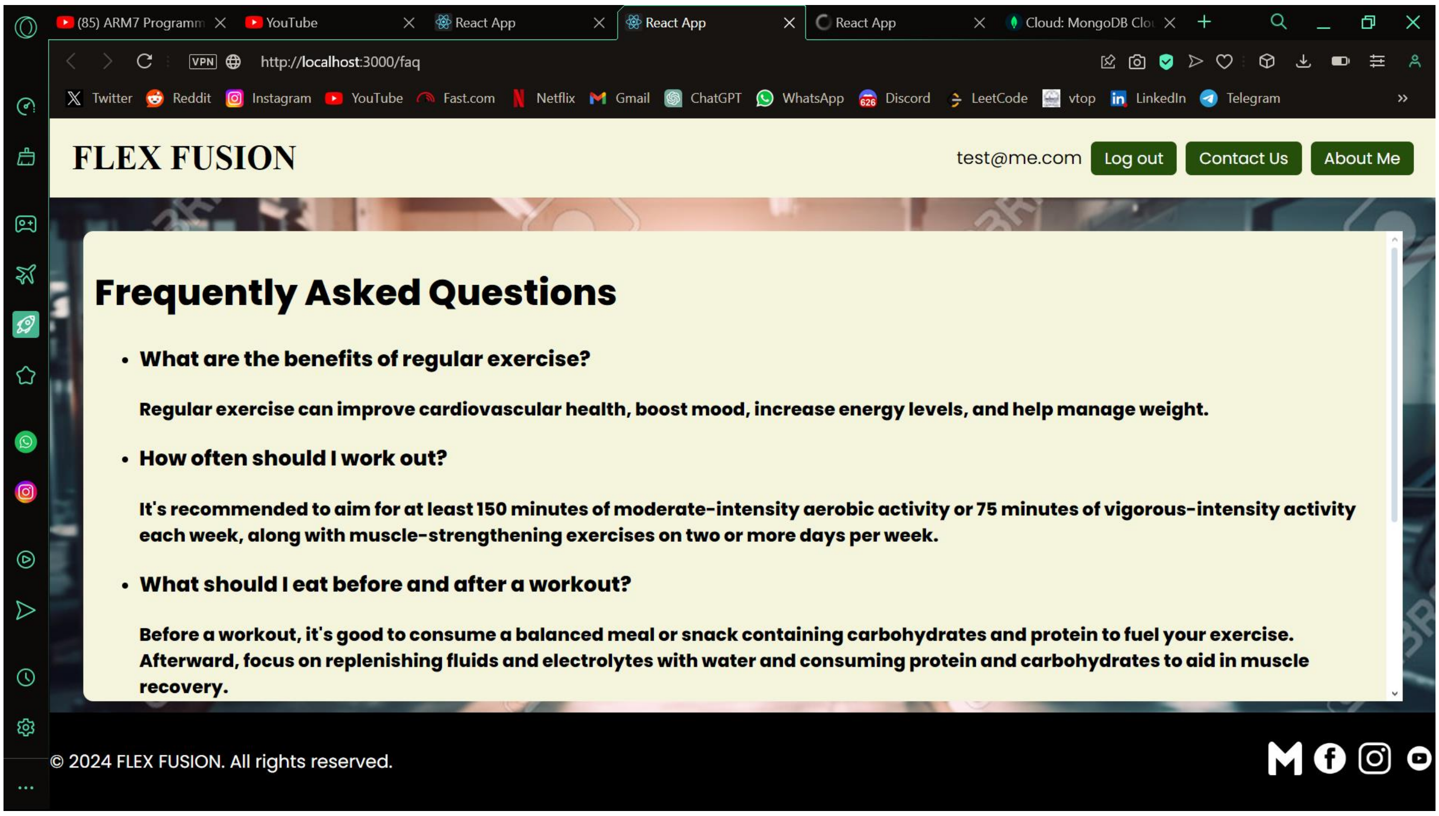
FLEX FUSION

test@me.com Log out Contact Us About Me

Workouts

- Crunches

© 2024 FLEX FUSION. All rights reserved.



FLEX FUSION

test@me.com Log out Contact Us About Me

Frequently Asked Questions

- What are the benefits of regular exercise?

  Regular exercise can improve cardiovascular health, boost mood, increase energy levels, and help manage weight.

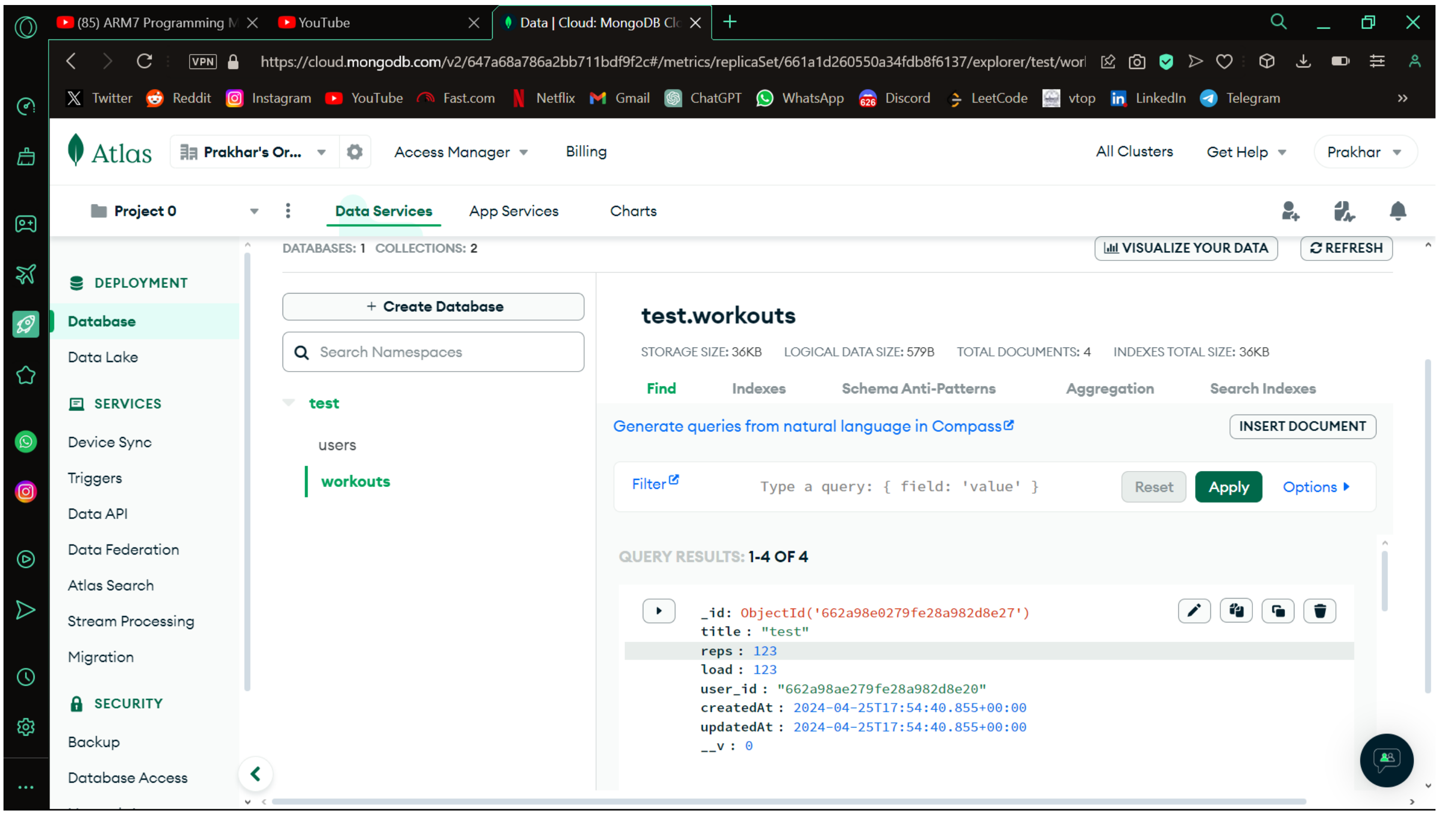
- How often should I work out?

  It's recommended to aim for at least 150 minutes of moderate-intensity aerobic activity or 75 minutes of vigorous-intensity activity each week, along with muscle-strengthening exercises on two or more days per week.

- What should I eat before and after a workout?

  Before a workout, it's good to consume a balanced meal or snack containing carbohydrates and protein to fuel your exercise. Afterward, focus on replenishing fluids and electrolytes with water and consuming protein and carbohydrates to aid in muscle recovery.

© 2024 FLEX FUSION. All rights reserved.



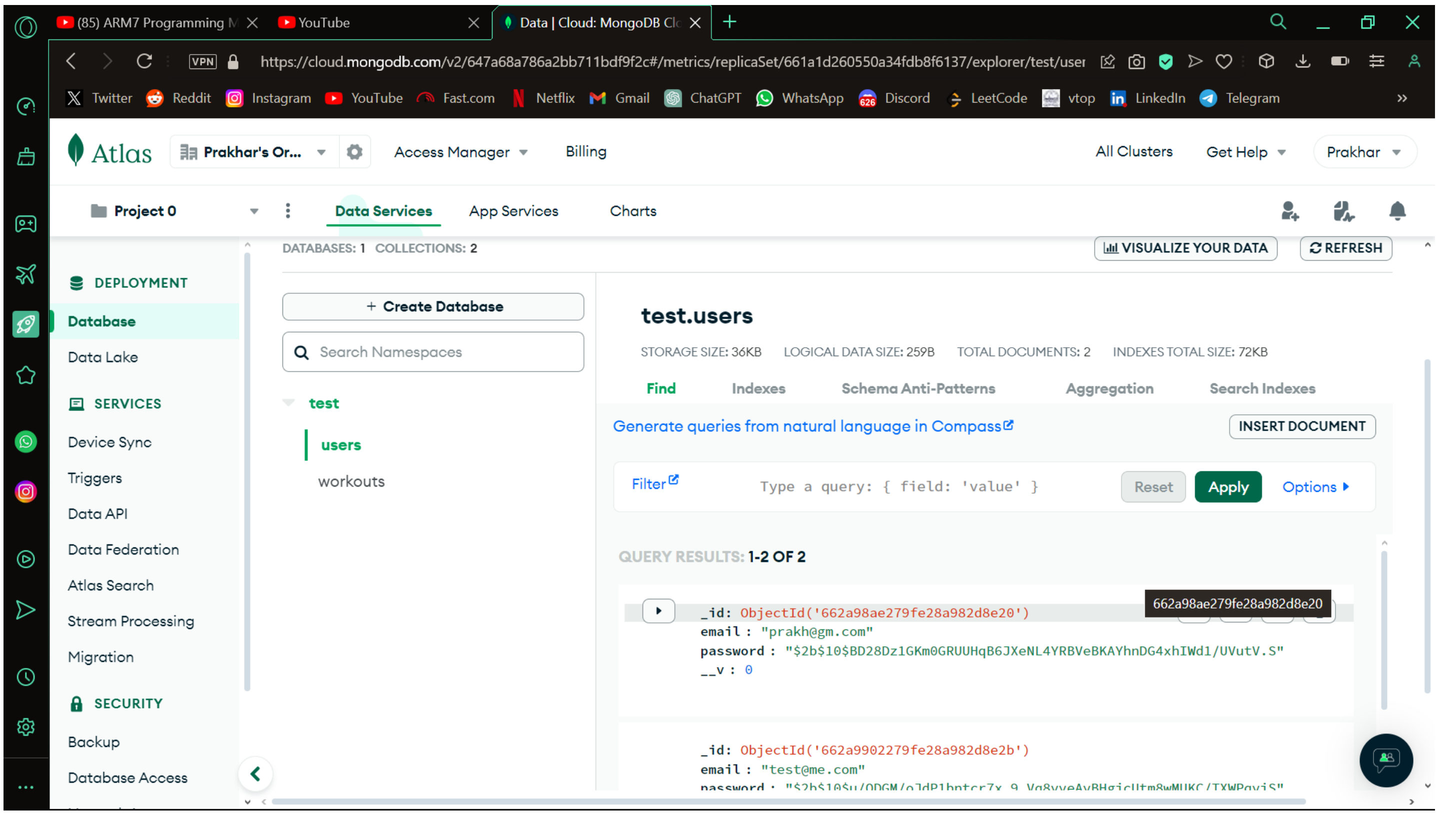
DATABASES: 1 COLLECTIONS: 2

test.workouts

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 579B TOTAL DOCUMENTS: 4 INDEXES TOTAL SIZE: 36KB

QUERY RESULTS: 1-4 OF 4

```
_id: ObjectId('662a98e0279fe28a982d8e27')
title : "test"
reps : 123
load : 123
user_id : "662a98ae279fe28a982d8e20"
createdAt : 2024-04-25T17:54:40.855+00:00
updatedAt : 2024-04-25T17:54:40.855+00:00
__v : 0
```



DATABASES: 1 COLLECTIONS: 2

test.users

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 259B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 72KB

QUERY RESULTS: 1-2 OF 2

```
_id: ObjectId('662a98ae279fe28a982d8e20')
email : "prakh@gm.com"
password : "$2b$10$BD28Dz1GKm0GRUUHqB6JXeNL4YRBVeBKAYhnDG4xhIWd1/UVutV.S"
__v : 0

_id: ObjectId('662a9902279fe28a982d8e2b')
email : "test@me.com"
```