**ZEN FITNESS ZONE TRACKER**

**DHIVYAA.D.M**

**DEEPIKA.P**

**ASHWINI.D**

**AYEESHA FATHIMA.A**

**Abstract :**

A Fitness Tracker Application is a web-based platform developed using HTML, CSS, and JavaScript to help users monitor their fitness journey efficiently. The application allows users to schedule workouts, track calorie burn, log water intake, and review workout history with an interactive and user-friendly dark-themed UI. It features a login/signup system for personalized tracking, ensuring data security. The dashboard presents workout schedules, calorie charts, water intake statistics, and progress highlights using dynamic charts and interactive elements. The application enhances user motivation through features like health points, step tracking, meditation focus, sleep schedule, and achievements. With a neon-themed interface and moving particle effects, the fitness tracker provides an engaging experience while helping users maintain a healthy lifestyle.

**SYNOPSIS :**

**Title: Zen Fitness Zone Tracker Application Using HTML, CSS, and JavaScript**

**Introduction:**

In today’s fast-paced world, maintaining a **healthy lifestyle** is a challenge, and many individuals struggle to keep track of their fitness progress. A **Fitness Tracker Application** can serve as an effective tool to help users **monitor their workouts, track calorie intake, and maintain a healthy routine**. This project aims to develop a **web-based fitness tracker** using **HTML, CSS, and JavaScript**, providing users with a structured and interactive platform to manage their fitness activities.

The application features a **dark-themed UI with neon effects**, making it visually appealing and easy to navigate. Users can **schedule their workouts, log water intake, track calories burned, and view workout history** in an organized manner. The use of **charts and interactive elements** enhances user engagement and provides clear insights into their fitness journey. Additionally, the application includes **step tracking, meditation focus, sleep schedule, and achievements** to encourage users to stay motivated.

By implementing a **secure login/signup system**, users can create personal accounts and store their fitness data securely. The fitness tracker ensures a **personalized and user-friendly experience**, making it a valuable tool for fitness enthusiasts who want to monitor and improve their health effectively.

**Module Description – Zen Fitness Zone Tracker Application :**

The **Fitness Tracker Application** is divided into several modules, each responsible for specific functionalities to ensure a seamless user experience. Below are the key modules of the application:

**Objective:**

* To develop an easy-to-use fitness tracking web application.
* To allow users to **schedule workouts with specific details** such as time, day, and calories burned.
* To provide a **calorie tracking system** with bar charts for better visualization.
* To enable users to **log water intake** and display it using a **bar graph.**
* To maintain a **workout history** with editing options.
* To incorporate **additional fitness features** like step tracking, meditation focus, sleep schedule, and health points.
* To offer a **secure login/signup system** with saved credentials for user authentication.

**Scope:**

This application is designed for **fitness enthusiasts** who want to track their workout routines and health progress in a structured manner. It provides a **personalized dashboard** where users can **manage their fitness data, track progress, and stay motivated**. The use of **charts, animations, and interactive UI elements** makes it more engaging and efficient.

**Features:**

1. **User Authentication:** Signup and login functionality with secure credential storage.
2. **Workout Scheduling:** Users can add workouts with time, day, and calories burned.
3. **Workout History:** Users can view, edit, and manage past workout logs.
4. **Calorie Tracking:** calorie burn trends for the week.
5. **Water Intake Logging:** Users can input daily water intake, represented using a **bar graph.**
6. **Dark-Themed UI:** A visually appealing **neon theme.**

**1. User Authentication Module**

**Purpose:**

* Allows users to **sign up and log in** securely.
* Stores user credentials using **local storage** for authentication.
* Prevents unauthorized access and ensures data privacy.

**Features:**

* **Signup Page:** Users can create an account by entering their **First Name, Last Name, Phone Number, Email, Password, and Confirm Password**.
* **Login Page:** Users can log in with their registered email and password.
* **Credential Validation:** Checks entered details against stored credentials before allowing access.

**2. Dashboard Module**

**Purpose:**

* Acts as the **central hub** where users can view and manage their fitness data.
* Displays **various health metrics** in an organized layout.

**Features:**

* **Workout Scheduling Section:** Users can schedule workouts by selecting **time, workout name, date, and calories burned**.
* **Workout History Section:** Displays past workout logs with an **delete button** for modifications.
* **Calories Burned Chart:** calories burned for the week.
* **Water Intake Pie Chart:** A graphical representation of daily water intake.

**3. Workout Scheduling Module**

**Purpose:**

* Allows users to **add and manage workouts** for better fitness planning.

**Features:**

* Users can input **workout details (time, name, date, calories burned)**.
* Workouts are displayed in the **schedule section** on the dashboard.
* Data is stored using **local storage** for easy access and modification.

**4. Workout History Module**

**Purpose:**

* Keeps a record of **all past workouts**, allowing users to track their progress over time.

**Features:**

* Displays a **scrollable list of previous workouts**.
* Includes an **delete button** to modify workout details.
* Prevents **undefined values** from appearing in the history section.

**5. Calorie Tracking Module**

**Purpose:**

* Helps users visualize their **calories burned** over a period of time.

**Features:**

* A **bar chart** that displays **daily calorie data**.
* Updates dynamically based on **user input** from scheduled workouts.

**6. Water Intake Tracking Module**

**Purpose:**

* Allows users to **log and monitor daily water consumption**.

**Features:**

* Users can **add water intake** while scheduling workouts.
* The **bar graph**  visually represents daily water consumption levels.

**7. UI & Theme Module**

**Purpose:**

* Ensures a **visually appealing and user-friendly interface**.

**Features:**

* **Dark-themed UI** with **neon accents** for a modern look.
* **styled buttons** for signup, login, and navigation.

**System Specifications – Fitness Tracker Application :**

The **Fitness Tracker Application** is a web-based platform developed using **HTML, CSS, and JavaScript**. The following system specifications define the necessary hardware and software requirements for running and developing the application efficiently.

**1. Hardware Requirements**

**For Development:**

* **Processor:** Intel Core i3 or higher (or AMD equivalent)
* **RAM:** Minimum 4GB (Recommended: 8GB or more)
* **Storage:** At least 10GB of free space
* **Display:** 1366×768 resolution or higher
* **Graphics Card:** Integrated or dedicated GPU (optional for UI design tools)
* **Internet Connection:** Required for development tools and libraries

**For End Users (Minimum Requirements):**

* **Device:** Desktop and laptop.
* **Processor:** Any modern processor (Dual-core or higher)
* **RAM:** 2GB or more
* **Storage:** Minimal, as the application is web-based
* **Internet Browser:** Chrome, Firefox, Edge, or any modern browser
* **Internet Connection:** Required for online usage (if hosted on a server)

**2. Software Requirements**

**For Development:**

* **Operating System:** Windows 10/11, macOS, or Linux
* **Code Editor:** VS Code, Sublime Text, or any preferred text editor
* **Browser:** Google Chrome, Mozilla Firefox, or Microsoft Edge (for testing)

**For End Users:**

* **Operating System:** Windows, macOS, Linux, Android, or iOS
* **Browser Compatibility:**
  + Google Chrome (Latest Version)
  + Mozilla Firefox
  + Microsoft Edge
  + Safari (for macOS/iOS users)

**3. Technology Stack**

**Frontend:**

* + HTML (Structure)
  + CSS (Styling)
  + JavaScript (Functionality & Interactivity)

**Backend:**

* + JavaScript

**Data storage:**

* + Local storage

**UI Enhancements:**

* + CSS Animations
  + JavaScript Interactivity
  + Chart.js (For visualizing data)

**4. Deployment Requirements**

* **Local Deployment:** Can run directly on a browser without a server.
* **Web Hosting (Optional for Online Use):**
  + GitHub Pages

**System Study – Advantages & Disadvantages**

The **Fitness Tracker Application** is designed to provide users with an efficient way to track their fitness activities. However, like any system, it has both advantages and disadvantages.

**Advantages:**

**1. User-Friendly Interface**

* The application features a **dark-themed UI with neon effects**, making it visually appealing and easy to navigate.
* Interactive elements such as buttons, charts, and animations enhance the user experience.

**2. Easy Workout & Health Tracking**

* Users can **schedule workouts, track calorie burn, and log water intake** easily.
* The **calorie burned bar chart** and **water intake bar graph** provide visual insights into fitness progress.

**3. Web-Based & Cross-Platform Compatibility**

* Since the application is developed using **HTML, CSS, and JavaScript**, it can be accessed from **any device with a web browser**.
* No need for installation, making it lightweight and easy to use.

**4. Secure User Authentication**

* The login/signup system ensures **secure access to personal fitness data**.
* Credentials are stored safely using **local storage** (or can be extended with Firebase for cloud storage).

**5. Data Visualization**

* The application uses **charts and graphs** to represent **calories burned, water intake, and workout history**, helping users analyze their progress better.

**6. Customization & Scalability**

* The application can be **expanded** to include features like **goal setting, diet tracking, and AI-based recommendations** in the future.

**Disadvantages:**

**1. Limited Data Storage**

* Since the application currently relies on **local storage**, users may lose their data if they **clear browser cache or switch devices**.
* No **cloud backup** (unless Firebase or a database is integrated).

**2. No Real-Time Syncing**

* The data does not sync across multiple devices unless an **online database** is implemented.
* Users must use the same device to access their workout history.

**3. No Automated Workout Suggestions**

* The system does not provide **personalized workout recommendations** based on user performance.
* AI or machine learning integration would be needed for **automated fitness advice**.

**4. Dependency on Manual Input**

* Users have to **manually enter workout details, water intake, and calories burned**.
* No integration with **fitness wearables or step counters** for automated tracking.

**System Testing for Fitness Tracker Application**

System testing ensures that the **Fitness Tracker Application** functions correctly and meets user requirements. It involves different types of testing to identify and fix errors before deployment.

**1. Testing Types**

**A. Functional Testing**

Checks if all features work as expected.  
✅ **Test Cases:**

* **User Authentication** (Login/Signup)
  + Valid credentials → Successful login
  + Invalid credentials → Error message
* **Workout Scheduling**
  + Add workout → Saved successfully
  + Delete workout → Removed from history
* **Water Intake Tracking**
  + Add water intake → Displays correct amount
* **Calorie Tracking & Charts**
  + Enter calories → Updates bar chart correctly
* **Dashboard & History**
  + Display correct past workouts

**B. User Interface (UI) Testing**

Ensures UI elements work properly and follow design guidelines.  
✅ **Test Cases:**

* Buttons, text fields, and navigation work smoothly
* Dark-themed UI with **neon effect** is properly displayed
* Responsive design adapts to different screen sizes

**C. Performance Testing**

Measures the system’s speed, responsiveness, and stability.  
✅ **Test Cases:**

* Application loads within **2 seconds**
* Handles **multiple users’ data** without lag
* Charts and analytics update instantly

**D. Security Testing**

Verifies that user data is protected.  
✅ **Test Cases:**

* Password encryption works correctly

**E. Compatibility Testing**

Checks if the application works on different devices and browsers.  
✅ **Test Cases:**

* Runs smoothly on **Chrome, Firefox, Edge**
* Functions correctly on **laptop and desktop**

**F. Usability Testing**

Ensures that the app is easy to use.  
✅ **Test Cases:**

* Users can navigate the app easily
* Workout tracking process is intuitive
* Clear error messages for invalid inputs

**2. Test Execution & Bug Fixing**

* Execute test cases **manually or using automation tools**
* Identify and fix bugs
* Retest the system to ensure fixes work

**3. Test Summary Report**

A final report detailing test results, bugs found, and fixes applied.

**SOURCE CODE:**

**DASHBOARD :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fitness Dashboard</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;600&display=swap');

body {

font-family: 'Poppins', sans-serif;

color: #ffffff;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

flex-direction: column;

padding: 10px;

position: relative;

overflow: hidden;

}

/\* Background Video Styling \*/

#background-video {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

object-fit: cover;

z-index: -1;

}

.container {

text-align: center;

background: rgba(30, 30, 30, 0.8);

padding: 15px;

border-radius: 10px;

box-shadow: 0px 0px 10px #00ffcc;

margin: 10px 0;

border: 1.5px solid #00ffcc;

}

.dashboard-container {

display: grid;

grid-template-columns: repeat(2, 1fr);

grid-gap: 10px;

max-width: 750px;

width: 100%;

padding: 5px;

}

.tracker{

background: rgba(40, 40, 40, 0.85);

backdrop-filter: blur(8px);

padding: 15px;

border-radius: 8px;

box-shadow: 0 0 8px rgba(0, 255, 255, 0.5);

text-align:center;

width: 100%;

height: 270px;

display: flex;

flex-direction: column;

justify-content: space-between;

}

.tracker.calories-tracker {

height: 280px;

}

.graphs-container {

display: flex;

justify-content: space-between;

margin-top: 10px;

width: 100%;

}

.graph-container{

width: 48%;

background: rgba(30, 30, 30, 0.9);

padding: 30px;

border-radius: 8px;

box-shadow: 0 0 8px rgba(0, 255, 255, 0.5);

text-align: center;

}

input {

width: 100%;

padding: 12px;

margin: 4px 0;

border: none;

border-radius: 5px;

outline: none;

text-align: center;

background: #333;

color: #FFF;

font-size: 12px;

}

button {

background: #05d3f2;

border: none;

padding: 6px;

border-radius: 8px;

cursor: pointer;

transition: 0.3s;

color: #0a0a0a;

font-weight: bold;

font-size: 12px;

}

button:hover {

background: #32ed38;

}

canvas {

max-width: 100%;

height: 100px;

}

</style>

</head>

<body>

<!-- Background Video -->

<video id="background-video" autoplay loop muted>

<source src="run.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

<div class="main" style="text-align: center;">

<h2>DASHBOARD TRACKER</h2>

<div class="dashboard-container">

<div class="tracker">

<h4>Workout</h4>

<form>

<input type="text" id="exercise" placeholder="Exercise">

<input type="number" id="duration" placeholder="Duration (mins)">

<button type="button" onclick="addWorkout()">Track</button>

</form>

</div>

<div class="tracker">

<h4>Diet & Water</h4>

<form>

<input type="number" id="caloriesMeal" placeholder="Calories">

<input type="number" id="water" placeholder="Water (ml/l)">

<button type="button" onclick="addDiet()">Track</button>

</form>

</div>

<div class="tracker calories-tracker">

<h4>Calories</h4>

<form>

<input type="number" id="age" placeholder="Age">

<input type="number" id="height" placeholder="Height (cm)">

<input type="number" id="weight" placeholder="Weight (kg)">

<button type="button" onclick="calculateCalories()">Track</button>

</form>

<p>Calories: <span id="caloriesResult">--</span> kcal</p>

</div>

<div class="tracker">

<h4>Heartbeat</h4>

<button onclick="trackHeartbeat()">Track</button>

<p>Heart Rate: <span id="bpm">--</span> BPM</p>

</div>

</div>

<div class="graphs-container">

<div class="graph-container">

<h4>Workout Progress</h4>

<canvas id="workoutChart"></canvas>

</div>

<div class="graph-container">

<h4>Diet & Water Intake</h4>

<canvas id="dietChart"></canvas>

</div>

</div>

<script>

function trackHeartbeat() {

document.getElementById("bpm").innerText = Math.floor(Math.random() \* (120 - 60) + 60);

}

function calculateCalories() {

let age = document.getElementById("age").value;

let height = document.getElementById("height").value;

let weight = document.getElementById("weight").value;

if (age && height && weight) {

let calories = Math.round((10 \* weight) + (6.25 \* height) - (5 \* age) + 5);

document.getElementById("caloriesResult").innerText = calories;

}

}

function addDiet() {

let calMeal = document.getElementById("caloriesMeal").value;

let water = document.getElementById("water").value;

if (calMeal && water) {

updateChart("dietChart", [parseInt(water)], "Water Intake (ml/l)", "rgba(0, 255, 255, 0.5)");

}

}

function addWorkout() {

let duration = document.getElementById("duration").value;

if (duration) {

updateChart("workoutChart", [parseInt(duration)], "Workout Duration (mins)", "rgba(0, 255, 0, 0.5)");

}

}

function updateChart(id, data, label, color) {

new Chart(document.getElementById(id).getContext("2d"), {

type: "bar",

data: {

labels: Array.from({ length: data.length }, (\_, i) => i + 1),

datasets: [{ label: label, data: data, backgroundColor: color }]

},

options: { scales: { y: { beginAtZero: true } } }

});

}

</script>

</body><br>

<body>

<div class="content">

<button class="MOVE TO CALENDAR">

<a href="calendar todo.html" class="btn"> 📅 MOVE TO CALENDAR</a>

</div><br>

<div class="content">

<button class="MOVE TO TIMER & STOPWATCH">

<a href="combined.html" class="btn"> MOVE TO 🕑 TIMER & STOPWATCH</a>

</div>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script src="DOWNJS.js"></script>

</body>

</html>

**DOWNJS.js :**

document.addEventListener("DOMContentLoaded", function() {

let workoutData = JSON.parse(localStorage.getItem("workoutData")) || {};

let dietData = JSON.parse(localStorage.getItem("dietData")) || {};

updateChart("workoutChart", workoutData);

updateDietChart(dietData);

});

let charts = {}; // Store chart instances globally

function updateChart(id, storedData) {

let ctx = document.getElementById(id).getContext("2d");

let labels = Object.keys(storedData); // Get dates

let data = labels.map(date => storedData[date].reduce((a, b) => a + b, 0)); // Sum values per day

// Generate different colors for each day's workout bar

let colors = id === "workoutChart"

? labels.map(() => `hsl(${Math.random() \* 360}, 70%, 60%)`)

: "rgba(0, 255, 255, 0.5)"; // Keep diet chart color unchanged

if (charts[id]) {

charts[id].destroy(); // Destroy previous chart instance before updating

}

charts[id] = new Chart(ctx, {

type: "bar",

data: {

labels: labels,

datasets: [{

label: id === "workoutChart" ? "Workout Duration (mins)" : "Water Intake (ml)",

data: data,

backgroundColor: colors // Apply unique colors only to workout bars

}]

},

options: {

scales: { y: { beginAtZero: true } }

}

});

}

function addWorkout() {

let duration = parseInt(document.getElementById("duration").value);

if (duration) {

let today = new Date().toISOString().split('T')[0];

let workoutData = JSON.parse(localStorage.getItem("workoutData")) || {};

if (!workoutData[today]) {

workoutData[today] = [];

}

workoutData[today].push(duration);

localStorage.setItem("workoutData", JSON.stringify(workoutData));

updateChart("workoutChart", workoutData); // ✅ Update graph instantly

}

}

function addDiet() {

let calories = parseInt(document.getElementById("caloriesMeal").value);

let water = parseInt(document.getElementById("water").value);

if (calories || water) {

let today = new Date().toISOString().split('T')[0];

let dietData = JSON.parse(localStorage.getItem("dietData")) || {};

if (!dietData[today]) {

dietData[today] = { calories: 0, water: 0 };

}

dietData[today].calories += calories || 0;

dietData[today].water += water || 0;

localStorage.setItem("dietData", JSON.stringify(dietData));

updateDietChart(dietData); // ✅ Update graph instantly

}

}

function updateDietChart(storedData) {

let ctx = document.getElementById("dietChart").getContext("2d");

let labels = Object.keys(storedData); // Get dates

let caloriesData = labels.map(date => storedData[date].calories);

let waterData = labels.map(date => storedData[date].water);

if (charts["dietChart"]) {

charts["dietChart"].destroy(); // Destroy previous chart instance

}

charts["dietChart"] = new Chart(ctx, {

type: "bar",

data: {

labels: labels, // ✅ Display Dates as labels

datasets: [

{

label: "Calories Intake (kcal)",

data: caloriesData,

backgroundColor: "rgba(255, 99, 132, 0.5)"

},

{

label: "Water Intake (ml)",

data: waterData,

backgroundColor: "rgba(0, 255, 255, 0.5)"

}

]

},

options: {

scales: {

x: { title: { display: true, text: "Date" } }, // ✅ Add x-axis title

y: { beginAtZero: true, title: { display: true, text: "Amount" } } // ✅ Add y-axis title

}

}

});

}

**CALENDER TODO :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Calendar and Workout Tracker</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<style>

body {

font-family: 'Poppins', sans-serif;

display: flex;

flex-direction: column;

align-items: center;

height: 100vh;

color: white;

padding: 10px;

}

#background-video {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

object-fit: cover;

z-index: -1;

}

.container {

display: flex;

flex-wrap: wrap;

justify-content: center;

align-items: center;

gap: 20px;

width: 90%;

}

.calendar-container, .todo-container {

background: rgba(0, 0, 0, 0.6);

padding: 10px;

border-radius: 15px;

box-shadow: 0 0 20px #00ffcc;

text-align: center;

width: 60%;

transition: 0.3s;

opacity: 0;

transform: translateY(20px);

animation: fadeInUp 0.8s ease-out forwards;

}

.todo-container {

width: 800%;

max-width: 1000px;

}

#workoutInput {

width: 300%;

max-width: 300px;

}

#workoutList {

max-height: 100px;

overflow-y: auto;

}

.calendar {

width: 300%;

max-width: 300px;

}

.calendar-container:hover, .todo-container:hover {

box-shadow: 0 0 15px #00ffcc;

transform: scale(1.05);

transition: 0.3s;

}

.chart-container {

display: block;

margin-top: 10px;

width: 55%;

background: rgba(0, 0, 0, 0.5);

padding: 15px;

border-radius: 15px;

box-shadow: 0 0 20px #ff00ff;

text-align: center;

transition: 0.3s;

opacity: 0;

animation: fadeInUp 1.5s ease-out forwards;

animation-delay: 0.5s;

}

canvas {

max-width: 600px;

max-height: 650px;

margin: auto;

}

h1, h3 {

font-size: 1.8em;

}

input, button {

font-size: 1.2em;

}

::placeholder {

font-size: 1.1em;

}

input[type="date"] {

font-size: 1.2em;

padding: 5px;

}

@keyframes fadeInUp {

from {

opacity: 0;

transform: translateY(20px);

}

to {

opacity: 1;

transform: translateY(0);

}

}

</style>

</head>

<body>

<video autoplay muted loop id="background-video">

<source src="grp.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

<h1>📅 CALENDAR & WORKOUT TRACKER 🏋</h1>

<div class="container">

<div class="calendar-container">

<h3>📅 Calendar</h3>

<input type="date" id="workoutDate" class="calendar">

</div>

<div class="todo-container">

<h3>🏋 Completed Workouts</h3>

<input type="text" id="workoutInput" placeholder="Add a workout">

<button onclick="addWorkout()">Add</button>

<ul id="workoutList"></ul>

</div>

</div>

<div class="chart-container" id="chartContainer">

<h3>📊 Workout Overview</h3>

<canvas id="workoutChart"></canvas>

</div>

<script>

let workoutData = JSON.parse(localStorage.getItem('workouts')) || {};

let colors = JSON.parse(localStorage.getItem('workoutColors')) || {};

let colorPalette = ["#FFD700", "#36A2EB", "#FF6384", "#4BC0C0", "#9966FF", "#FF9F40"];

let colorIndex = Object.keys(colors).length;

function addWorkout() {

let input = document.getElementById("workoutInput");

let dateInput = document.getElementById("workoutDate").value;

let workout = input.value.trim();

let dateTime = dateInput ? ${dateInput} ${new Date().toLocaleTimeString()} : new Date().toLocaleString();

if (workout === "") return;

let ul = document.getElementById("workoutList");

let li = document.createElement("li");

li.innerHTML = ${workout} <span style='font-size: 0.8em; color: gray;'>(${dateTime})</span> <span class='delete' onclick='removeWorkout(this, "${workout}", "${dateTime}")'>✖</span>;

ul.appendChild(li);

if (!Array.isArray(workoutData[workout])) {

workoutData[workout] = [];

}

workoutData[workout].push(dateTime);

if (!colors[workout]) {

colors[workout] = colorPalette[colorIndex % colorPalette.length];

colorIndex++;

}

localStorage.setItem('workouts', JSON.stringify(workoutData));

localStorage.setItem('workoutColors', JSON.stringify(colors));

updateChart();

input.value = "";

}

function removeWorkout(element, workout, dateTime) {

element.parentElement.remove();

if (Array.isArray(workoutData[workout])) {

workoutData[workout] = workoutData[workout].filter(entry => entry !== dateTime);

if (workoutData[workout].length === 0) delete workoutData[workout];

}

localStorage.setItem('workouts', JSON.stringify(workoutData));

updateChart();

}

function updateChart() {

let ctx = document.getElementById("workoutChart").getContext("2d");

if (window.workoutChartInstance) window.workoutChartInstance.destroy();

window.workoutChartInstance = new Chart(ctx, {

type: "pie",

data: {

labels: Object.keys(workoutData),

datasets: [{

data: Object.keys(workoutData).map(workout => workoutData[workout].length),

backgroundColor: Object.keys(workoutData).map(workout => colors[workout] || "#ddd"),

}],

},

options: {

plugins: {

legend: {

position: 'right',

labels: {

color: 'white',

}

}

}

}

});

}

function loadWorkouts() {

let ul = document.getElementById("workoutList");

for (let workout in workoutData) {

if (Array.isArray(workoutData[workout])) {

workoutData[workout].forEach(dateTime => {

let li = document.createElement("li");

li.innerHTML = ${workout} <span style='font-size: 0.8em; color: gray;'>(${dateTime})</span> <span class='delete' onclick='removeWorkout(this, "${workout}", "${dateTime}")'>✖</span>;

ul.appendChild(li);

});

}

}

updateChart();

}

window.onload = loadWorkouts;

</script>

</body>

</html>

**TIMER AND STOP WATCH :**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Clock & Workout</title>

<style>

@import url('https://fonts.googleapis.com/css2?family=Rajdhani:wght@400;700&display=swap');

body {

font-family: 'Rajdhani', sans-serif;

display: flex;

justify-content: space-between;

align-items: center;

height: 100vh;

color: white;

margin: 0;

padding: 20px;

animation: fadeIn 1.5s ease-in-out;

}

/\* Background Video Styling \*/

#background-video {

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

object-fit: cover;

z-index: -1;

}

@keyframes fadeIn {

from { opacity: 0; }

to { opacity: 1; }

}

.left-container, .right-container {

width: 48%;

height: 90vh;

padding: 10px;

background: rgba(0, 0, 0, 0.3);

border-radius: 10px;

box-shadow: 0px 0px 10px cyan;

overflow: auto;

animation: slideIn 1.5s ease-out;

}

.left-container {

animation: glowEffect 3s infinite alternate;

animation: floatEffect 2s infinite alternate;

}

@keyframes slideIn {

from { transform: translateY(50px); opacity: 0; }

to { transform: translateY(0); opacity: 1; }

}

@keyframes floatEffect {

from { transform: translateY(0px); }

to { transform: translateY(-5px); }

}

@keyframes glowEffect {

from { box-shadow: 0px 0px 10px cyan; }

to { box-shadow: 0px 0px 30px cyan; }

}

.container {

text-align: center;

background: rgba(30, 30, 30, 0.7);

padding: 15px;

border-radius: 10px;

box-shadow: 0px 0px 10px #00ffcc;

margin: 40px 0;

border: 1.0px solid #00ffcc;

animation: popIn 1s ease-out;

}

@keyframes popIn {

from { transform: scale(0.8); opacity: 0; }

to { transform: scale(1); opacity: 1; }

}

.time-display {

font-size: 2.5em;

margin: 10px 0;

font-weight: bold;

color: #00ffcc;

}

button {

margin: 15px;

padding: 10px 15px;

border: none;

background: #00ffcc;

color: #121212;

cursor: pointer;

border-radius: 6px;

font-size: 0.9em;

font-weight: bold;

transition: transform 0.3s;

}

button:hover {

transform: scale(1.1);

}

.exercise-container {

display: grid;

grid-template-columns: repeat(4, 1fr);

gap: 20px;

justify-content: center;

}

.modal-content img {

width: 300px; /\* Adjust as needed \*/

height: 300px; /\* Adjust as needed \*/

object-fit: cover; /\* Ensures the image scales properly \*/

border-radius: 8px;

}

.exercise {

width: 90px;

cursor: pointer;

text-align: center;

padding: 10px;

background: rgba(30, 30, 30, 0.7);

border-radius: 8px;

border: 2px solid cyan;

box-shadow: 0 0 6px cyan;

transition: transform 0.3s, box-shadow 0.3s;

animation: floatEffect 2s infinite alternate;

}

@keyframes floatEffect {

from { transform: translateY(0px); }

to { transform: translateY(-5px); }

}

.exercise:hover {

transform: scale(1.1);

box-shadow: 0 0 35px cyan;

}

.exercise img {

width: 100%;

border-radius: 10px;

}

.modal {

display: none;

position: fixed;

top: 0;

left: 0;

width: 100%;

height: 100%;

background: rgba(0, 0, 0, 0.5); /\* Dark overlay \*/

justify-content: center;

align-items: center;

}

.modal-content {

position: relative;

text-align: center;

background: transparent;

border: none;

}

.modal-content img {

width: 100%;

max-width: 600px;

height: auto;

border-radius: 8px;

}

</style>

</head>

<body>

<!-- Background Video -->

<video id="background-video" autoplay loop muted>

<source src="mou.mp4" type="video/mp4">

Your browser does not support the video tag.

</video>

<div class="left-container">

<h1 style="text-align:center;">Time Breaker 🕑</h1>

<div class="container">

<h2>Clock</h2>

<div id="clock" class="time-display"></div>

</div>

<div class="container">

<h2>Timer</h2>

<div id="timer" class="time-display">00:00</div>

<button onclick="startTimer()">Start</button>

<button onclick="pauseTimer()">Pause</button>

<button onclick="resetTimer()">Reset</button>

</div>

<div class="container">

<h2>Stopwatch</h2>

<div id="stopwatch" class="time-display">00:00:00</div>

<button onclick="startStopwatch()">Start</button>

<button onclick="pauseStopwatch()">Pause</button>

<button onclick="resetStopwatch()">Reset</button>

</div>

</div>

<div class="right-container">

<h1 style="text-align:center;">Basic Warm-up Exercises</h1>

<div class="exercise-container">

<body>

<div class="exercise" onclick="openModal('Dumbbell Lateral Lunge', 'Works legs and glutes, improves balance.', '30 seconds each side', 'pexels-photo-999257.jpg')">

<img src="pexels-photo-999257.jpg" alt="Dumbbell Lateral Lunge">

<p>Dumbbell Lateral Lunge</p>

</div>

<div class="exercise" onclick="openModal('Seated Hamstring Stretch', 'Stretches the hamstrings and lower back.', '20-30 seconds each leg', 'pexels-photo-3771071.jpg')">

<img src="pexels-photo-3771071.jpg" alt="Seated Hamstring Stretch">

<p>Seated Hamstring Stretch</p>

</div>

<div class="exercise" onclick="openModal('Plank', 'Strengthens core and improves stability.', 'Hold for 30-60 seconds', 'images.jpg')">

<img src="images.jpg" alt="Plank">

<p>Plank</p>

</div>

<div class="exercise" onclick="openModal('Skipping', 'Improves coordination and cardiovascular endurance.', '1-2 minutes', 'Skipping-3-1200x900-1-1024x768.jpg')">

<img src="Skipping-3-1200x900-1-1024x768.jpg" alt="Skipping">

<p>Skipping</p>

</div>

<div class="exercise" onclick="openModal('Squat', 'Strengthens lower body muscles and improves flexibility.', '12-15 reps', 'pexels-photo-7187856.jpg')">

<img src="pexels-photo-7187856.jpg" alt="Squat">

<p>Squat</p>

</div>

<div class="exercise" onclick="openModal('Standing Quadriceps Stretch', 'Stretches the quadriceps and improves flexibility.', '20-30 seconds each leg', 'oi.jpg')">

<img src="oi.jpg" alt="Standing Quadriceps Stretch">

<p>Standing Quadriceps Stretch</p>

</div>

<div class="exercise" onclick="openModal('Meditation', 'Meditation is a practice that involves focusing your attention and awareness. It can help you to relax, reduce stress, and improve your concentration.', '5 to 45 minutes per day.', 'med.jpg')">

<img src="med.jpg" alt="Meditation">

<p>Meditation</p>

</div>

<div class="exercise" onclick="openModal('Tree Pose', 'Improves balance and stability', '30 seconds to 1 minutes per side depending on your skill level.', 'tree.jpg')">

<img src="tree.jpg" alt="Meditation">

<p>Tree pose</p>

</div>

<div class="exercise" onclick="openModal('Downward-facing Dog pose', 'Improves the blood flow through your body', '30 seconds to 1 minutes.', 'moun.jpg')">

<img src="moun.jpg" alt="Downward-facing Dog pose">

<p>Downward-facing Dog pose</p>

</div>

<div class="exercise" onclick="openModal('Standing Forward Bend', 'Relieves tension in the spine, hips and lower limbs', '30 seconds to 1 minutes depending on your skill level.', 'knee.jpg')">

<img src="knee.jpg" alt="Standing Forward Bend">

<p>Standing Forward Bend</p>

</div>

<div class="exercise" onclick="openModal('Cobra Pose', 'Tones the abdomen. May be beneficial for people with respiratory issues like asthma.', 'Hold the pose for 5-10 breaths with daily practice can be gradually increased.', 'sna.jpg')">

<img src="sna.jpg" alt="Cobra Pose">

<p>Cobra Pose</p>

</div>

<div class="exercise" onclick="openModal('Camel Pose', 'May help with menstrual discomfort. Strengthens back muscles.', 'Beginners, holding the pose for around 20-30 seconds is recommended, gradually increasing the duration as flexibility improves.', 'camel.jpg')">

<img src="camel.jpg" alt="Camel Pose">

<p>Camel Pose</p>

</div>

</div>

</div>

<div id="modal" class="modal" onclick="closeModal()">

<div class="modal-content">

<span class="close" onclick="closeModal()">&times;</span>

<h2 id="exercise-title"></h2>

<img id="exercise-img" src="" alt="Exercise">

<p id="exercise-definition"></p>

<p id="exercise-time"></p>

</div>

</div>

<script>

function openModal(title, imgSrc) {

document.getElementById("exercise-title").innerText = title;

document.getElementById("exercise-img").src = imgSrc;

document.getElementById("modal").style.display = "flex";

}

function closeModal() {

document.getElementById("modal").style.display = "none";

}

</script>

</body>

</div>

</div>

<script>

function updateClock() {

document.getElementById("clock").innerText = new Date().toLocaleTimeString();

}

setInterval(updateClock, 1000);

updateClock();

let timerInterval, timerTime = 0;

function startTimer() {

if (!timerInterval) {

timerInterval = setInterval(() => {

timerTime++;

document.getElementById("timer").innerText = new Date(timerTime \* 1000).toISOString().substr(14, 5);

}, 1000);

}

}

function pauseTimer() { clearInterval(timerInterval); timerInterval = null; }

function resetTimer() { pauseTimer(); timerTime = 0; document.getElementById("timer").innerText = "00:00"; }

// Stopwatch (Fast speed with step-by-step increments)

let stopwatchInterval;

let stopwatchTime = 0;

function startStopwatch() {

if (!stopwatchInterval) {

stopwatchInterval = setInterval(() => {

stopwatchTime++;

let date = new Date(stopwatchTime \* 1000);

document.getElementById("stopwatch").innerText = date.toISOString().substr(11, 8);

}, 200); // Updates every 200ms for a faster effect but step-by-step

}

}

function pauseStopwatch() { clearInterval(stopwatchInterval); stopwatchInterval = null; }

function resetStopwatch() { pauseStopwatch(); stopwatchTime = 0; document.getElementById("stopwatch").innerText = "00:00:00"; }

</script>

<!-- Link the external JavaScript file at the end of the body -->

<script src="comtral.js"></script>

<script>

function openModal(title, definition, time, imgSrc) {

document.getElementById("exercise-title").innerText = title;

document.getElementById("exercise-definition").innerText = definition;

document.getElementById("exercise-time").innerText = "Duration: " + time;

document.getElementById("exercise-img").src = imgSrc;

document.getElementById("modal").style.display = "flex";

}

function closeModal() {

document.getElementById("modal").style.display = "none";

}

// Prevent accidental closing when clicking inside the modal

document.getElementById("modal").addEventListener("click", function (event) {

if (event.target.classList.contains("modal")) {

closeModal();

}

});

</script>

</body>

</html>

**Comtral.js :**

<script>

// Exercise descriptions

const definitions = {

"Dumbbell Lateral Lunge": "A lateral lunge with dumbbells strengthens your glutes, quads, and inner thighs.",

"Seated Hamstring Stretch": "A stretch that helps improve hamstring flexibility and reduces lower back strain.",

"Plank": "An isometric exercise that strengthens the core, shoulders, and back.",

"Skipping": "A cardiovascular workout that improves agility, endurance, and coordination.",

"Squat": "A lower-body exercise that targets the quads, hamstrings, and glutes.",

"Standing Quadriceps Stretch": "A stretch that improves flexibility in the front of the thigh and prevents knee injuries."

};

function openModal(title, imgSrc) {

document.getElementById("exercise-title").innerText = title;

document.getElementById("exercise-img").src = imgSrc;

document.getElementById("exercise-description").innerText = definitions[title] || "No description available.";

document.getElementById("modal").style.display = "flex";

}

function closeModal() {

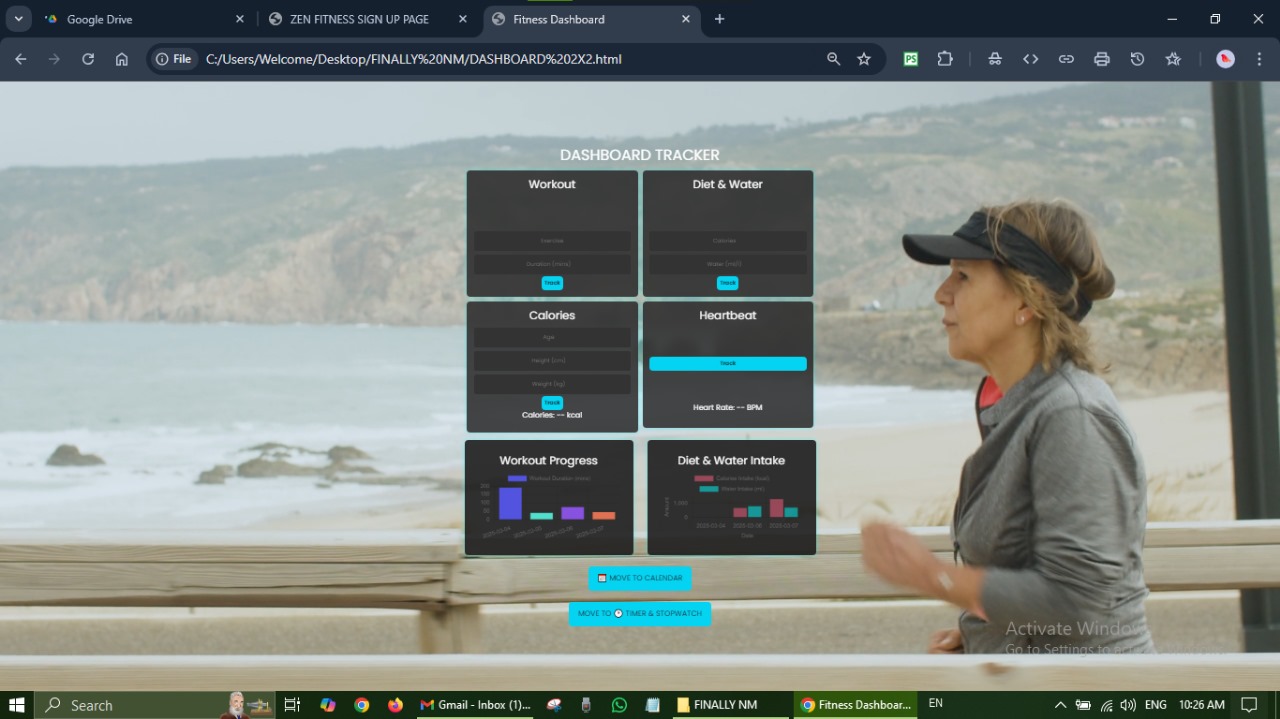
document.getElementById("modal").style.display = "none";

}

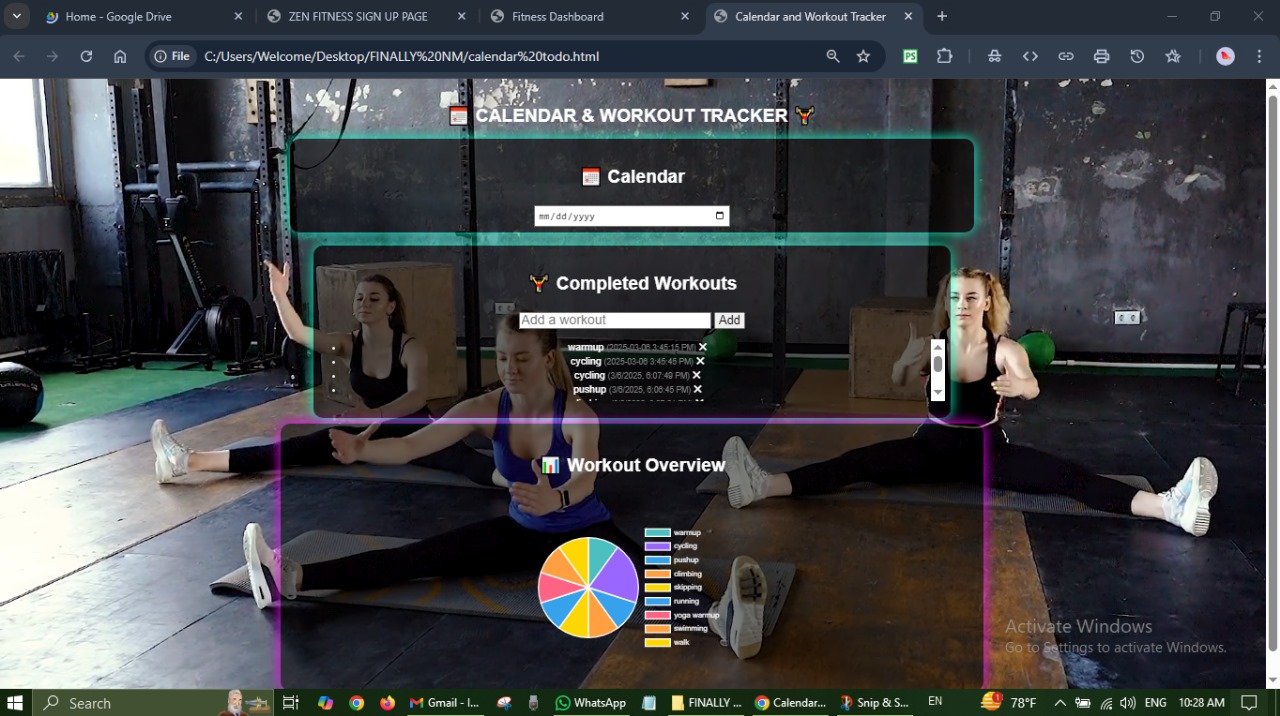
</script>

**SCREENSHOTS:**

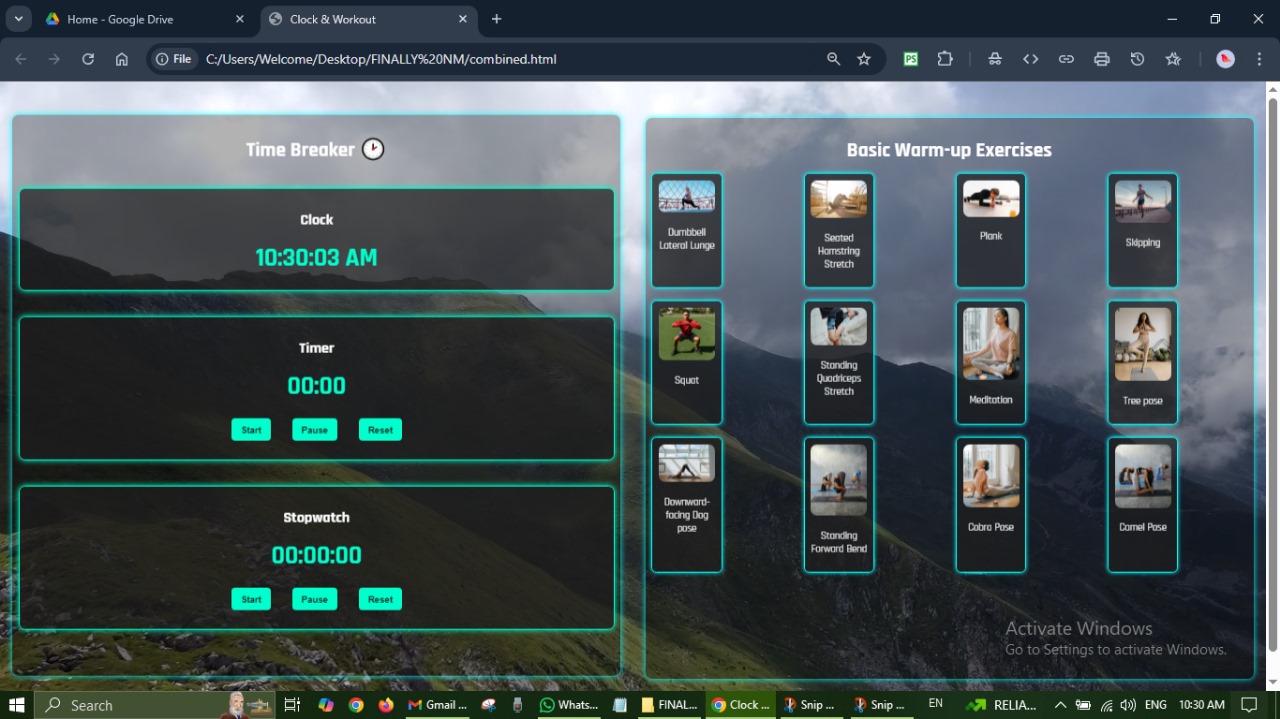
**DASHBOARD :**



**CALENDER TO-DO :**



**TIMER AND STOP WATCH :**



**Conclusion :**

The **Fitness Tracker Application** is a web-based system designed to help users monitor their workouts, water intake, and calorie burn efficiently. Developed using **HTML, CSS, and JavaScript**, it features an interactive **dark-themed UI with a neon effect**, a structured workout scheduler, real-time calorie tracking, and intuitive data visualization through charts.

The development process involved **system analysis, design, implementation, and rigorous testing** to ensure a **smooth user experience, high performance, and security**. The application is **responsive and compatible across different devices and browsers**, making fitness tracking accessible to users anytime, anywhere.

**Future enhancements :**

With **further enhancements**, such as **real-time cloud storage, AI-based fitness recommendations, and wearable device integration**, this system can be expanded into a **comprehensive personal health assistant**. Overall, the project successfully achieves its goal of **helping users stay fit and maintain a healthy lifestyle**.