INSPIRED BY LIFE

MANIPAL
ACADEMY of HIGHER EDUCATION
(Deemed to be University under Section 3 of the UGC Act, 1956)

DBS MINI PROJECT
NUTRIFY: CALORIE TACKING AND HEALTH MANAGEMENT SYSTEM

Asmi Pandey-230962073
Prachita Singal-230962046

Abstract

Nutrify is an intelligent and interactive fitness tracking system designed to empower users in achieving their personal health goals through detailed tracking and insights. Built using Python (Streamlit) and Oracle SQL, the system allows users to register, log their meals and workouts, and monitor their daily calorie balance with ease. Users can record comprehensive meal details, including food items and quantities, as well as log workout sessions with specific exercises and durations.

Nutrify calculates calorie intake and expenditure using built-in nutritional and exercise data, enabling users to view real-time statistics and monitor progress aligned with goals such as weight loss, muscle gain, or maintenance. The platform also features goal-based analysis, identifying whether a user's current net calories are in sync with their objectives.

On the backend, complex SQL queries power leaderboards, food frequency analytics, and personalized reports. With its user-friendly interface and robust database integration, Nutrify provides a centralized, scalable solution for health-conscious users to take control of their fitness journey and make informed lifestyle choices.

Problem Statement:

The key focus of the system is its robust database design and SQL-driven features. The project uses Oracle SQL as the relational database, with a schema that includes users, food, meals, workouts, exercises, and goal tracking — each with clear relationships and referential integrity.

Key problems addressed include:

● Creating a **normalized relational schema** with proper foreign keys for structured data access across meals, exercises, and user activity logs.

● Enabling users to **log food intake and workouts** through intuitive interfaces while maintaining data consistency across related entities.

● Dynamically **calculating calorie consumption and expenditure** using multi-table JOINs and arithmetic expressions over quantity and duration values.

● Recording **weight and fitness goal changes** over time, with support for history tracking and progress evaluation based on user objectives.

SQL-centric Functionalities

Calorie Tracking and Net Balance:
SQL queries calculate total calories consumed and burned using expressions like `f.calories * quantity` and `duration * calories_burned / 60`.
Example: `SELECT COALESCE(SUM(...))` for both intake and burn tracking.

Goal Progress Analysis:
Combines SQL-based data (from meals, workouts, goals) with logic to check if users are on track with objectives like weight loss or gain.

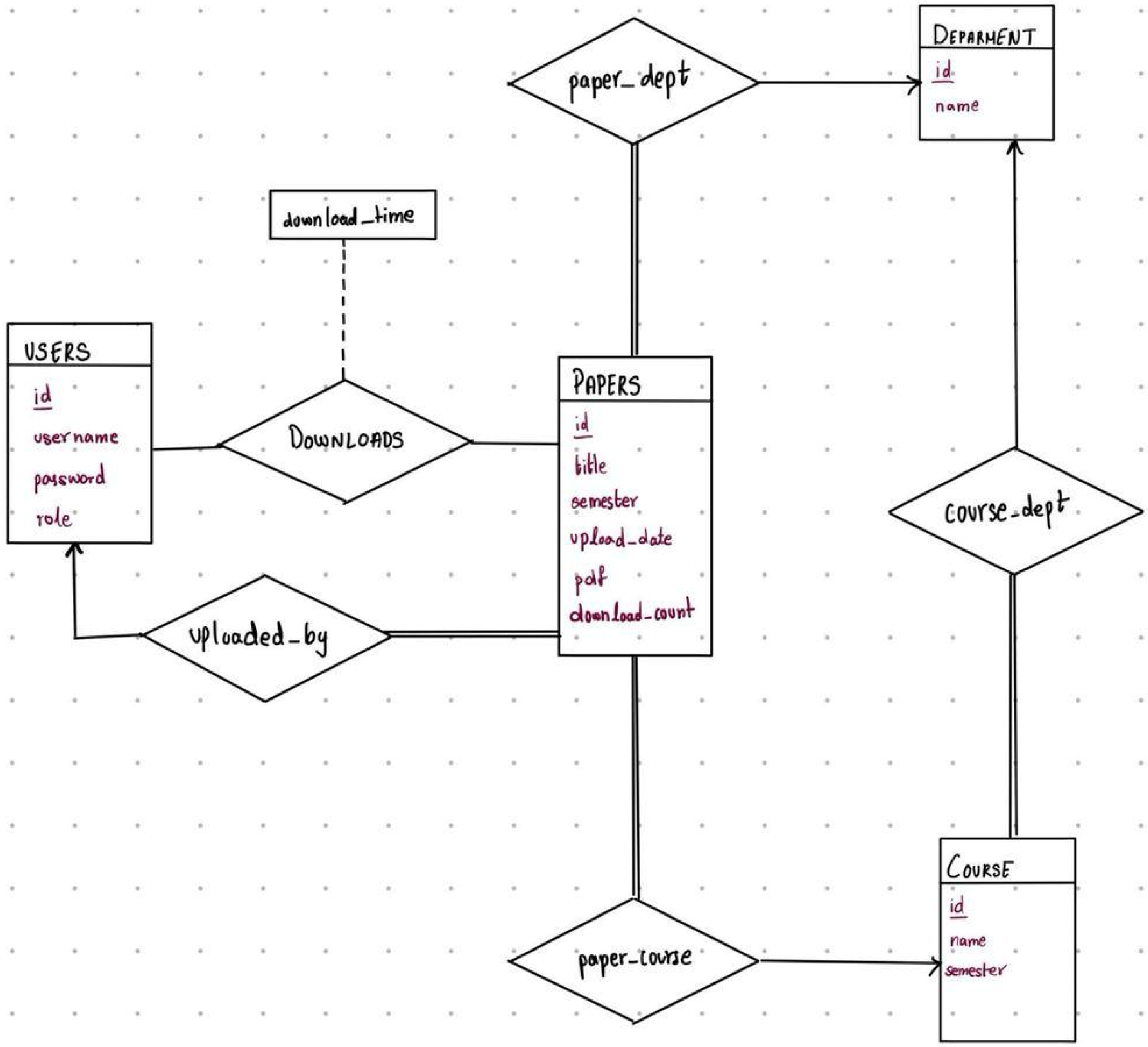
User Activity Leaderboard:
Uses `GROUP BY`, `ORDER BY`, and date filters to rank users based on workout counts or food log frequency over recent periods.

Behavioral Insights:
Reveals patterns such as most logged foods, top exercises, and users with weight fluctuations using `JOIN`, `GROUP BY`, `HAVING`, and `WHERE NOT EXISTS`.

Daily Reports:
Applies SQL date-based aggregation to show daily calories consumed or burned, supporting time-series analysis.

Advanced Features:
Implements `WITH` clauses for reusable views, conditional aggregation via `HAVING`, and potential for trigger-based auto-logging.

ER Diagram:


paper_dept
DEPARMENT
id
name
download_time
USERS
id
username
password
role
DOWNLOADS
PAPERS
id
title
semester
upload_date
pdf
download_count
course-dept
uploaded_by
COURSE
id
name
semester
paper_course

TABLES:

1. Users Table

```
CREATE TABLE Users (
   user_id INT PRIMARY KEY,
   name VARCHAR2(100),
   email VARCHAR2(100) UNIQUE,
   password VARCHAR2(100),
   age INT CHECK (age >= 10),
   gender VARCHAR2(10),
   height FLOAT CHECK (height > 0),
   weight FLOAT CHECK (weight > 0),
   goal VARCHAR2(20)
);
```

2. Food Table

```
CREATE TABLE Food (
   food_id INT PRIMARY KEY,
   name VARCHAR2(100) NOT NULL,
   calories FLOAT CHECK (calories >= 0),
   protein FLOAT CHECK (protein >= 0),
   carbs FLOAT CHECK (carbs >= 0),
   fats FLOAT CHECK (fats >= 0)
);
```

3. Exercise Table

```
CREATE TABLE Exercise (
   exercise_id INT PRIMARY KEY,
   name VARCHAR2(100) NOT NULL,
   calories_burned FLOAT CHECK (calories_burned >= 0)
);
```

4. Meals Table

```
CREATE TABLE Meals (
   meal_id INT PRIMARY KEY,
   user_id INT REFERENCES Users(user_id) ON DELETE CASCADE,
   meal_type VARCHAR2(20) NOT NULL,
   date_logged DATE NOT NULL
);
```

5. Meal_Items Table

```
CREATE TABLE Meal_Items (
   meal_item_id INT PRIMARY KEY,
   meal_id INT REFERENCES Meals(meal_id) ON DELETE CASCADE,
   food_id INT REFERENCES Food(food_id) ON DELETE CASCADE,
   quantity FLOAT CHECK (quantity > 0)
);
```

6. Workout Table

```
CREATE TABLE Workout (
   workout_id INT PRIMARY KEY,
   user_id INT REFERENCES Users(user_id) ON DELETE CASCADE,
   date_logged DATE NOT NULL
);
```

7. Workout_Exercises Table

```
CREATE TABLE Workout_Exercises (
   workout_exercise_id INT PRIMARY KEY,
   workout_id INT REFERENCES Workout(workout_id) ON DELETE CASCADE,
   exercise_id INT REFERENCES Exercise(exercise_id) ON DELETE CASCADE,
   duration FLOAT CHECK (duration > 0)
);
```

8. Goals_History Table

sql

Copy

Edit

```
CREATE TABLE Goals_History (
   goal_id INT PRIMARY KEY,
   user_id INT REFERENCES Users(user_id) ON DELETE CASCADE,
   goal VARCHAR2(20),
   weight FLOAT CHECK (weight > 0),
   recorded_on DATE DEFAULT SYSDATE
);
```

COMPLEX QUERIES:

Total Calories Consumed (Per User):
Displays the total calories consumed by each user by summing calories per food item multiplied by quantity across all meals.
Query:

```
SELECT u.name, SUM(f.calories * mi.quantity) AS total_calories
FROM Users u
JOIN Meals m ON u.user_id = m.user_id
JOIN Meal_Items mi ON m.meal_id = mi.meal_id
JOIN Food f ON mi.food_id = f.food_id
GROUP BY u.name
ORDER BY total_calories DESC;
```

Total Calories Burned (Per User):
Calculates total calories burned by each user based on workout duration and exercise intensity.
Query:

```
SELECT u.name, SUM(we.duration * e.calories_burned / 60) AS total_burned
FROM Users u
JOIN Workout w ON u.user_id = w.user_id
JOIN Workout_Exercises we ON w.workout_id = we.workout_id
JOIN Exercise e ON we.exercise_id = e.exercise_id
GROUP BY u.name
ORDER BY total_burned DESC;
```

Top 3 Most Frequently Logged Foods:
Lists the top 3 food items most frequently consumed/logged by users across all meal entries.
Query:

```
SELECT f.name, COUNT(*) AS times_logged
FROM Meal_Items mi
JOIN Food f ON mi.food_id = f.food_id
GROUP BY f.name
ORDER BY times_logged DESC
FETCH FIRST 3 ROWS ONLY;
```

TRIGGERS:

1. Trigger for `Users` Table – Auto-generates `user_id` using `user_seq`

Trigger Name: `trg_user_id`
Associated Table: `Users`
Trigger Logic:
Executed before an insert on the `Users` table to assign the next value from the `user_seq` sequence to `user_id`.

```
CREATE OR REPLACE TRIGGER trg_user_id
BEFORE INSERT ON Users
FOR EACH ROW
BEGIN
    :NEW.user_id := user_seq.NEXTVAL;
END;
/
```

---

2. Trigger for `Food` Table – Auto-generates `food_id` using `food_seq`

Trigger Name: `trg_food_id`
Associated Table: `Food`
Trigger Logic:
Assigns the next value from `food_seq` as `food_id` before inserting a new food item.

```
CREATE OR REPLACE TRIGGER trg_food_id
BEFORE INSERT ON Food
FOR EACH ROW
BEGIN
    :NEW.food_id := food_seq.NEXTVAL;
END;
/
```

---

3. Trigger for `Exercise` Table – Auto-generates `exercise_id` using `exercise_seq`

Trigger Name: `trg_exercise_id`
Associated Table: `Exercise`
Trigger Logic:
Sets the primary key `exercise_id` from `exercise_seq` before insertion.

```
CREATE OR REPLACE TRIGGER trg_exercise_id
BEFORE INSERT ON Exercise
FOR EACH ROW
BEGIN
    :NEW.exercise_id := exercise_seq.NEXTVAL;
END;
/
```

---

4. Trigger for `Meals` Table – Auto-generates `meal_id` using `meal_seq`

Trigger Name: `trg_meal_id`
Associated Table: `Meals`
Trigger Logic:
Uses the `meal_seq` sequence to populate `meal_id` automatically.

```
CREATE OR REPLACE TRIGGER trg_meal_id
BEFORE INSERT ON Meals
FOR EACH ROW
BEGIN
    :NEW.meal_id := meal_seq.NEXTVAL;
END;
/
```

---

5. Trigger for `Meal_Items` Table – Auto-generates `meal_item_id` using `meal_item_seq`

Trigger Name: `trg_meal_item_id`
Associated Table: `Meal_Items`
Trigger Logic:
Auto-fills `meal_item_id` using `meal_item_seq` on insert.

```
CREATE OR REPLACE TRIGGER trg_meal_item_id
BEFORE INSERT ON Meal_Items
FOR EACH ROW
BEGIN
    :NEW.meal_item_id := meal_item_seq.NEXTVAL;
END;
/
```

---

6. Trigger for `Workout` Table – Auto-generates `workout_id` using `workout_seq`

Trigger Name: `trg_workout_id`
Associated Table: `Workout`
Trigger Logic:
Automatically sets `workout_id` before a row is inserted using `workout_seq`.

```
CREATE OR REPLACE TRIGGER trg_workout_id
BEFORE INSERT ON Workout
FOR EACH ROW
BEGIN
    :NEW.workout_id := workout_seq.NEXTVAL;
END;
/
```

---

7. Trigger for `Workout_Exercises` Table – Auto-generates `workout_exercise_id` using `workout_exercise_seq`

Trigger Name: `trg_workout_exercise_id`
Associated Table: `Workout_Exercises`
Trigger Logic:
Generates the primary key from `workout_exercise_seq`.

```
CREATE OR REPLACE TRIGGER trg_workout_exercise_id
BEFORE INSERT ON Workout_Exercises
FOR EACH ROW
BEGIN
    :NEW.workout_exercise_id := workout_exercise_seq.NEXTVAL;
END;
/
```

---

8. Trigger for `Goals_History` Table – Auto-generates `goal_id` using `goal_seq`

Trigger Name: `trg_goal_id`
Associated Table: `Goals_History`
Trigger Logic:
Sets the `goal_id` from the `goal_seq` sequence automatically on insertion.

```
CREATE OR REPLACE TRIGGER trg_goal_id
BEFORE INSERT ON Goals_History
FOR EACH ROW
BEGIN
    :NEW.goal_id := goal_seq.NEXTVAL;
END;
/
```

PROCEDURES AND FUNCTIONS:

1. Procedure to Log a Meal with Two Food Items

Procedure Name: `log_meal_simple`
Purpose: Inserts a meal entry for a given user along with two associated food items and their respective quantities.

Definition:

```
CREATE OR REPLACE PROCEDURE log_meal_simple (
    p_user_id      IN Users.user_id%TYPE,
    p_meal_type    IN Meals.meal_type%TYPE,
    p_date         IN DATE,
    p_food_id1     IN Food.food_id%TYPE,
    p_qty1         IN Meal_Items.quantity%TYPE,
    p_food_id2     IN Food.food_id%TYPE,
    p_qty2         IN Meal_Items.quantity%TYPE
) IS
    v_meal_id Meals.meal_id%TYPE;
BEGIN
    INSERT INTO Meals (user_id, meal_type, date_logged)
    VALUES (p_user_id, p_meal_type, p_date)
    RETURNING meal_id INTO v_meal_id;

    INSERT INTO Meal_Items (meal_id, food_id, quantity)
    VALUES (v_meal_id, p_food_id1, p_qty1);

    INSERT INTO Meal_Items (meal_id, food_id, quantity)
    VALUES (v_meal_id, p_food_id2, p_qty2);
END;
/
```

Execution Example:

```
BEGIN
    log_meal_simple(1, 'Dinner', SYSDATE, 1, 2, 2, 1.5);
END;
```

2. Procedure to Log a Workout with Two Exercises

Procedure Name: `log_workout_simple`
Purpose: Records a workout session for a user including two exercises with specific durations.

Definition:

```
CREATE OR REPLACE PROCEDURE log_workout_simple (
    p_user_id     IN Users.user_id%TYPE,
    p_date        IN DATE,
    p_ex_id1      IN Exercise.exercise_id%TYPE,
    p_duration1   IN Workout_Exercises.duration%TYPE,
    p_ex_id2      IN Exercise.exercise_id%TYPE,
    p_duration2   IN Workout_Exercises.duration%TYPE
) IS
    v_workout_id Workout.workout_id%TYPE;
BEGIN
    INSERT INTO Workout (user_id, date_logged)
    VALUES (p_user_id, p_date)
    RETURNING workout_id INTO v_workout_id;

    INSERT INTO Workout_Exercises (workout_id,
exercise_id, duration)
    VALUES (v_workout_id, p_ex_id1, p_duration1);

    INSERT INTO Workout_Exercises (workout_id,
exercise_id, duration)
    VALUES (v_workout_id, p_ex_id2, p_duration2);
END;
/
```

Execution Example:

```
BEGIN
    log_workout_simple(1, SYSDATE, 2, 20, 3, 15);
END;
/
```

3. Function to Calculate BMI (Body Mass Index)

Function Name: `calculate_bmi`
Purpose: Computes the BMI for a user based on their height and weight stored in the `Users` table.

Definition:

```
CREATE OR REPLACE FUNCTION calculate_bmi (
    p_user_id IN Users.user_id%TYPE
) RETURN NUMBER IS
    v_height FLOAT;
    v_weight FLOAT;
    v_bmi    NUMBER;
BEGIN
    SELECT height, weight INTO v_height, v_weight
    FROM Users
    WHERE user_id = p_user_id;

    -- Convert height from cm to meters
    v_bmi := v_weight / POWER(v_height / 100, 2);

    RETURN ROUND(v_bmi, 2);
END;
/
```

Execution Example:

```
DECLARE
    bmi NUMBER;
BEGIN
    bmi := calculate_bmi(1);
    DBMS_OUTPUT.PUT_LINE('BMI: ' || bmi);
END;
/
```

DATABASE CONNECTIVITY: (Python and Streamlit)

```
 import streamlit as st
import cx_Oracle
import pandas as pd

# ✅ Connect to Oracle DB
try:
    conn = cx_Oracle.connect("SYSTEM", "prachita4", "localhost:1521/XE")
    cur = conn.cursor()
except cx_Oracle.DatabaseError as e:
    st.error(f"❌ Could not connect to Oracle DB: {e}")
    st.stop()

st.title("🏋️‍♀️ Fitness Tracker Dashboard")

menu = st.sidebar.radio("Navigate", [
    "Add User", "View Users", "Statistics",
    "View Workouts", "Log Workout",
    "View Meals", "Log Meal", "Goal Check"
])

# Add User
if menu == "Add User":
    st.subheader("➕ Register a New User")
    with st.form("add_user_form"):
        name = st.text_input("Full Name")
        email = st.text_input("Email")
        password = st.text_input("Password", type="password")
        age = st.number_input("Age", min_value=10, max_value=100)
        gender = st.selectbox("Gender", ["Male", "Female", "Other"])
        height = st.number_input("Height (cm)", min_value=50.0, max_value=250.0)
        weight = st.number_input("Weight (kg)", min_value=20.0, max_value=200.0)
        goal = st.selectbox("Goal", ["Lose Weight", "Gain Muscle", "Maintain"])
        submit = st.form_submit_button("Add User")
        if submit:
            try:
                cur.execute("""
                    INSERT INTO Users (user_id, name, email, password, age, gender, height, weight, goal)
                    VALUES (user_seq.NEXTVAL, :1, :2, :3, :4, :5, :6, :7, :8)
                """, (name, email, password, age, gender, height, weight, goal))
                conn.commit()
                st.success(f"✅ User '{name}' added successfully!")
            except Exception as e:
                st.error(f"❌ Error inserting user: {e}")
```

```
# View Users
elif menu == "View Users":
    st.subheader("📋 Registered Users")
    try:
        cur.execute("SELECT user_id, name, email, age, goal FROM Users")
        df = pd.DataFrame(cur.fetchall(), columns=[desc[0] for desc in cur.description])
        st.dataframe(df)
    except Exception as e:
        st.error(f"❌ Error fetching user data: {e}")


# Statistics
elif menu == "Statistics":
    st.subheader("📊 Fitness Statistics Overview")
    try:
        st.markdown("### 🥗 Total Calories Consumed Per User")
        cur.execute("""
            SELECT u.name, SUM(f.calories * mi.quantity) AS total_calories
            FROM Users u
            JOIN Meals m ON u.user_id = m.user_id
            JOIN Meal_Items mi ON m.meal_id = mi.meal_id
            JOIN Food f ON mi.food_id = f.food_id
            GROUP BY u.name
            ORDER BY total_calories DESC
```

```
""")
df = pd.DataFrame(cur.fetchall(), columns=[desc[0] for desc in cur.description])
st.dataframe(df)


st.markdown("### 🔥 Total Calories Burned Per User")
cur.execute("""
    SELECT u.name, SUM(we.duration * e.calories_burned / 60) AS total_burned
    FROM Users u
    JOIN Workout w ON u.user_id = w.user_id
    JOIN Workout_Exercises we ON w.workout_id = we.workout_id
    JOIN Exercise e ON we.exercise_id = e.exercise_id
    GROUP BY u.name
    ORDER BY total_burned DESC
""")
df = pd.DataFrame(cur.fetchall(), columns=[desc[0] for desc in cur.description])
st.dataframe(df)


st.markdown("### 🍕 Top 3 Most Frequently Logged Foods")
cur.execute("""
    SELECT f.name, COUNT(*) AS times_logged
    FROM Meal_Items mi
    JOIN Food f ON mi.food_id = f.food_id
    GROUP BY f.name
```

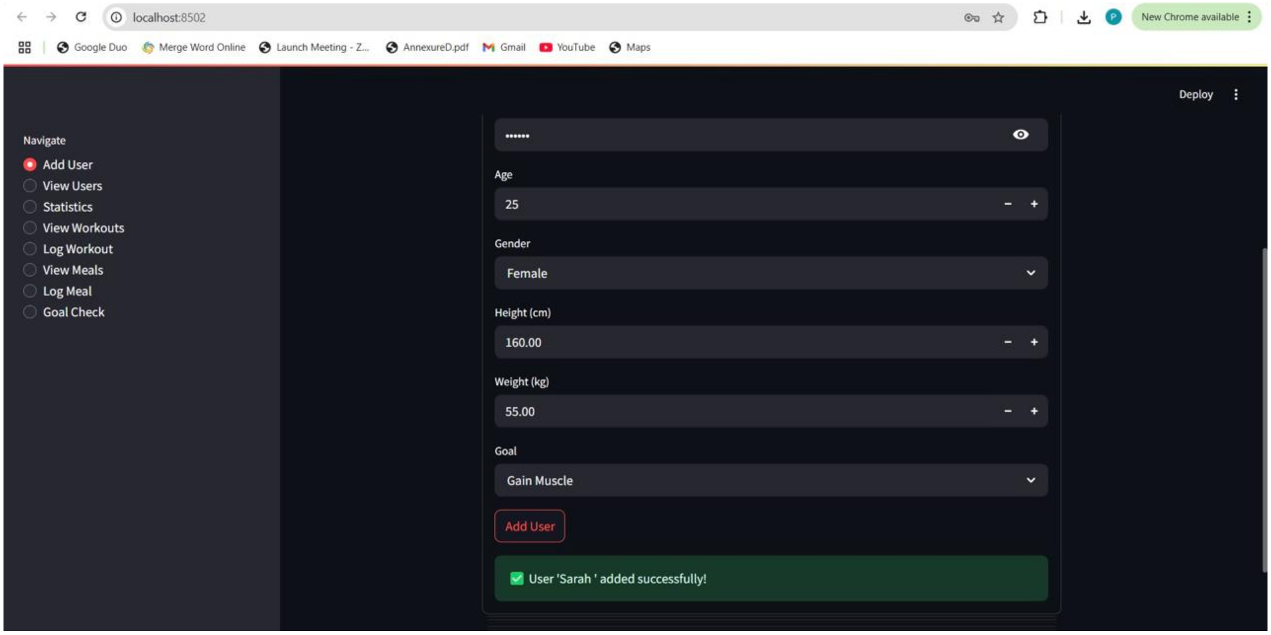
```
            ORDER BY times_logged DESC
            FETCH FIRST 3 ROWS ONLY
        """)
        df = pd.DataFrame(cur.fetchall(), columns=[desc[0] for desc in cur.description])
        st.dataframe(df)
    except Exception as e:
        st.error(f"❌ Failed to load statistics: {e}")


# View Workouts
elif menu == "View Workouts":
    st.subheader("🏃 Workout Logs")
    try:
        cur.execute("""
            SELECT w.workout_id, u.name AS user_name, w.date_logged, e.name AS exercise_name, we.duration
            FROM Workout w
            JOIN Users u ON w.user_id = u.user_id
            JOIN Workout_Exercises we ON w.workout_id = we.workout_id
            JOIN Exercise e ON we.exercise_id = e.exercise_id
            ORDER BY w.date_logged DESC
        """)
        df = pd.DataFrame(cur.fetchall(), columns=[desc[0] for desc in cur.description])
        st.dataframe(df)
    except Exception as e:
        st.error(f"❌ Error fetching workout data: {e}")

# Log Workout
elif menu == "Log Workout":
    st.subheader("📝 Log a New Workout")
    try:
        cur.execute("SELECT user_id, name FROM Users")
        users = {name: uid for uid, name in cur.fetchall()}
        cur.execute("SELECT exercise_id, name FROM Exercise")
        exercises = {name: eid for eid, name in cur.fetchall()}

        with st.form("log_workout_form"):
            user_name = st.selectbox("User", list(users.keys()))
            date = st.date_input("Date")
            exercise_name = st.selectbox("Exercise", list(exercises.keys()))
            duration = st.number_input("Duration (minutes)", min_value=1.0)
            submit = st.form_submit_button("Log Workout")

            if submit:
                uid = users[user_name]
                eid = exercises[exercise_name]
                cur.execute("INSERT INTO Workout (user_id, date_logged) VALUES (:1, :2)", (uid, date))
                cur.execute("SELECT MAX(workout_id) FROM Workout WHERE user_id = :1", (uid,))
                workout_id = cur.fetchone()[0]
                cur.execute("INSERT INTO Workout_Exercises (workout_id, exercise_id, duration) VALUES (:1, :2, :3)", (workout_id, eid, duration))
                conn.commit()
                st.success("✅ Workout logged!")
    except Exception as e:
        st.error(f"❌ Could not log workout: {e}")


# View Meals
elif menu == "View Meals":
    st.subheader("🍽️ Meal Logs")
    try:
        cur.execute("""
            SELECT m.meal_id, u.name AS user_name, m.meal_type, m.date_logged, f.name AS food_name, mi.quantity
            FROM Meals m
            JOIN Users u ON m.user_id = u.user_id
            JOIN Meal_Items mi ON m.meal_id = mi.meal_id
            JOIN Food f ON mi.food_id = f.food_id
            ORDER BY m.date_logged DESC
        """)
        df = pd.DataFrame(cur.fetchall(), columns=[desc[0] for desc in cur.description])

        st.dataframe(df)

    except Exception as e:

        st.error(f"❌ Error fetching meal data: {e}")


# Log Meal

elif menu == "Log Meal":

    st.subheader("📝 Log a New Meal")

    try:

        cur.execute("SELECT user_id, name FROM Users")

        users = {name: uid for uid, name in cur.fetchall()}

        cur.execute("SELECT food_id, name FROM Food")

        foods = {name: fid for fid, name in cur.fetchall()}


        with st.form("log_meal_form"):

            user_name = st.selectbox("User", list(users.keys()))

            date = st.date_input("Date")

            meal_type = st.selectbox("Meal Type", ["Breakfast", "Lunch", "Dinner", "Snack"])

            food_name = st.selectbox("Food", list(foods.keys()))

            quantity = st.number_input("Quantity", min_value=0.1)

            submit = st.form_submit_button("Log Meal")


            if submit:
                uid = users[user_name]
                fid = foods[food_name]
                cur.execute("INSERT INTO Meals (user_id, meal_type, date_logged) VALUES (:1, :2, :3)", (uid, meal_type, date))
                cur.execute("SELECT MAX(meal_id) FROM Meals WHERE user_id = :1", (uid,))
                meal_id = cur.fetchone()[0]
                cur.execute("INSERT INTO Meal_Items (meal_id, food_id, quantity) VALUES (:1, :2, :3)", (meal_id, fid, quantity))
                conn.commit()
                st.success("✅ Meal logged!")
    except Exception as e:
        st.error(f"❌ Could not log meal: {e}")


# Goal Check
elif menu == "Goal Check":
    st.subheader("🎯 Check Progress Towards Goal")
    try:
        cur.execute("SELECT user_id, name, goal FROM Users")
        user_data = cur.fetchall()
        user_dict = {name: (uid, goal) for uid, name, goal in user_data}

        user_name = st.selectbox("Select User", list(user_dict.keys()))
        uid, goal = user_dict[user_name]
```

```
st.markdown(f"**Goal:** {goal}")

# Total calories consumed
cur.execute("""
    SELECT COALESCE(SUM(f.calories * mi.quantity), 0)
    FROM Meals m
    JOIN Meal_Items mi ON m.meal_id = mi.meal_id
    JOIN Food f ON mi.food_id = f.food_id
    WHERE m.user_id = :1
""", (uid,))
calories_in = cur.fetchone()[0]

# Total calories burned
cur.execute("""
    SELECT COALESCE(SUM(we.duration * e.calories_burned / 60), 0)
    FROM Workout w
    JOIN Workout_Exercises we ON w.workout_id = we.workout_id
    JOIN Exercise e ON we.exercise_id = e.exercise_id
    WHERE w.user_id = :1
""", (uid,))
calories_out = cur.fetchone()[0]

net = calories_in - calories_out
```

```
    st.markdown(f"📥 **Calories Consumed:** {calories_in:.2f}")

    st.markdown(f"🔥 **Calories Burned:** {calories_out:.2f}")

    st.markdown(f"🧮 **Net Calories:** {net:.2f}")


    if goal == "Lose Weight":

        if net < 1500:

            st.success("✅ On track for weight loss!")

        else:

            st.warning("⚠️ Too many net calories for weight loss.")

    elif goal == "Gain Muscle":

        if net > 2500:

            st.success("✅ On track for muscle gain!")

        else:

            st.warning("⚠️ Increase calorie intake for gaining muscle.")

    else:

        if 1800 <= net <= 2200:

            st.success("✅ Maintaining well!")

        else:

            st.warning("⚠️ Your intake isn't aligned with maintenance."

except Exception as e:

    st.error(f"❌ Could not fetch goal data: {e}")
```

UI DESIGN:

1) Adding a User:



••••••

Age

25

Gender

Female

Height (cm)

160.00

Weight (kg)

55.00

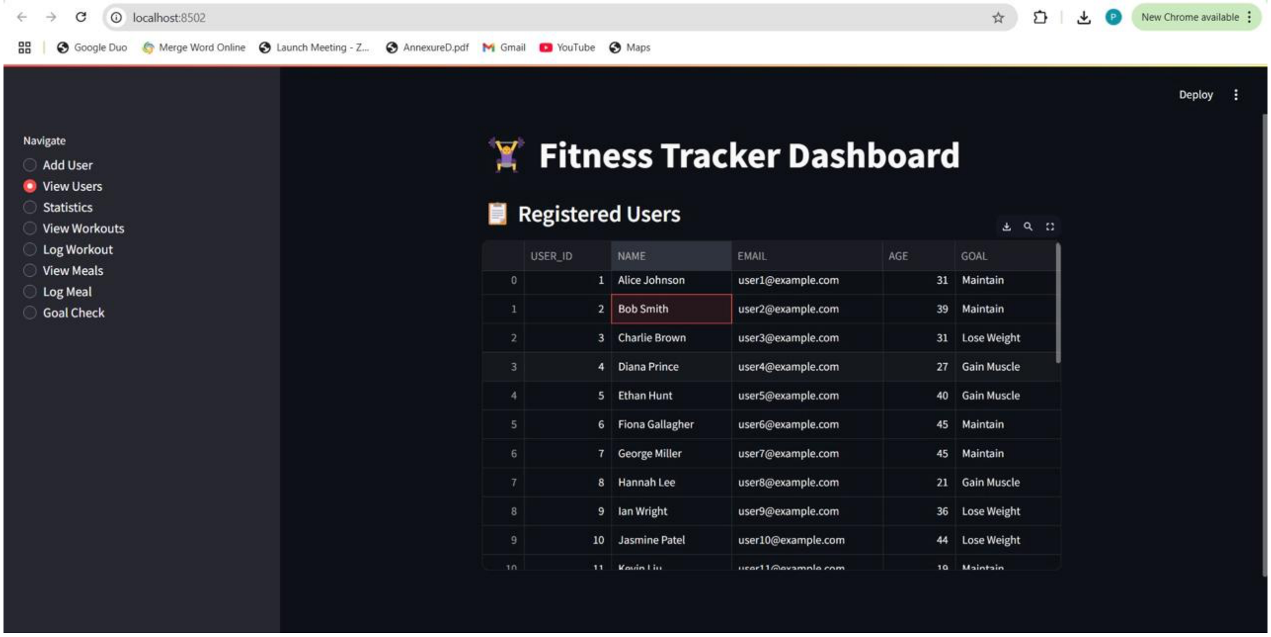
Goal

Gain Muscle

Add User

✅ User 'Sarah ' added successfully!

2) Viewing all Users



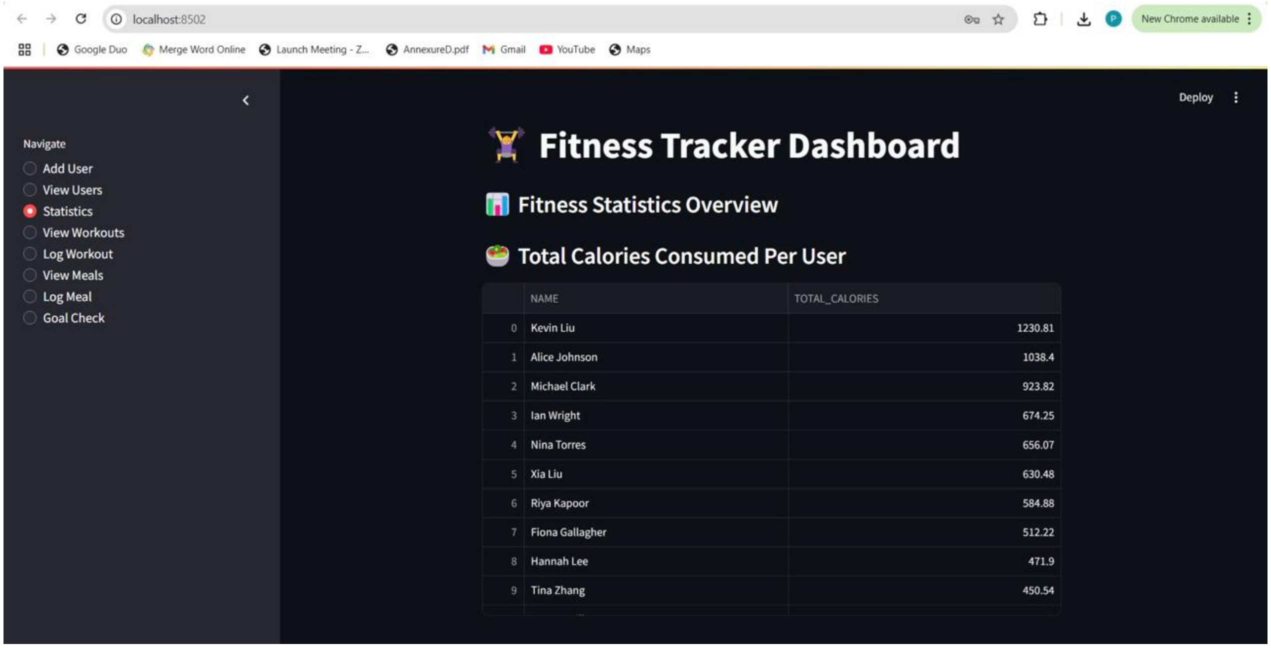
🏋️ Fitness Tracker Dashboard

📋 Registered Users

	USER_ID	NAME	EMAIL	AGE	GOAL
0	1	Alice Johnson	user1@example.com	31	Maintain
1	2	Bob Smith	user2@example.com	39	Maintain
2	3	Charlie Brown	user3@example.com	31	Lose Weight
3	4	Diana Prince	user4@example.com	27	Gain Muscle
4	5	Ethan Hunt	user5@example.com	40	Gain Muscle
5	6	Fiona Gallagher	user6@example.com	45	Maintain
6	7	George Miller	user7@example.com	45	Maintain
7	8	Hannah Lee	user8@example.com	21	Gain Muscle
8	9	Ian Wright	user9@example.com	36	Lose Weight
9	10	Jasmine Patel	user10@example.com	44	Lose Weight
10	11	Kevin Liu	user11@example.com	19	Maintain

3) Health Statistics

Helping you keep a track of your calorie intake



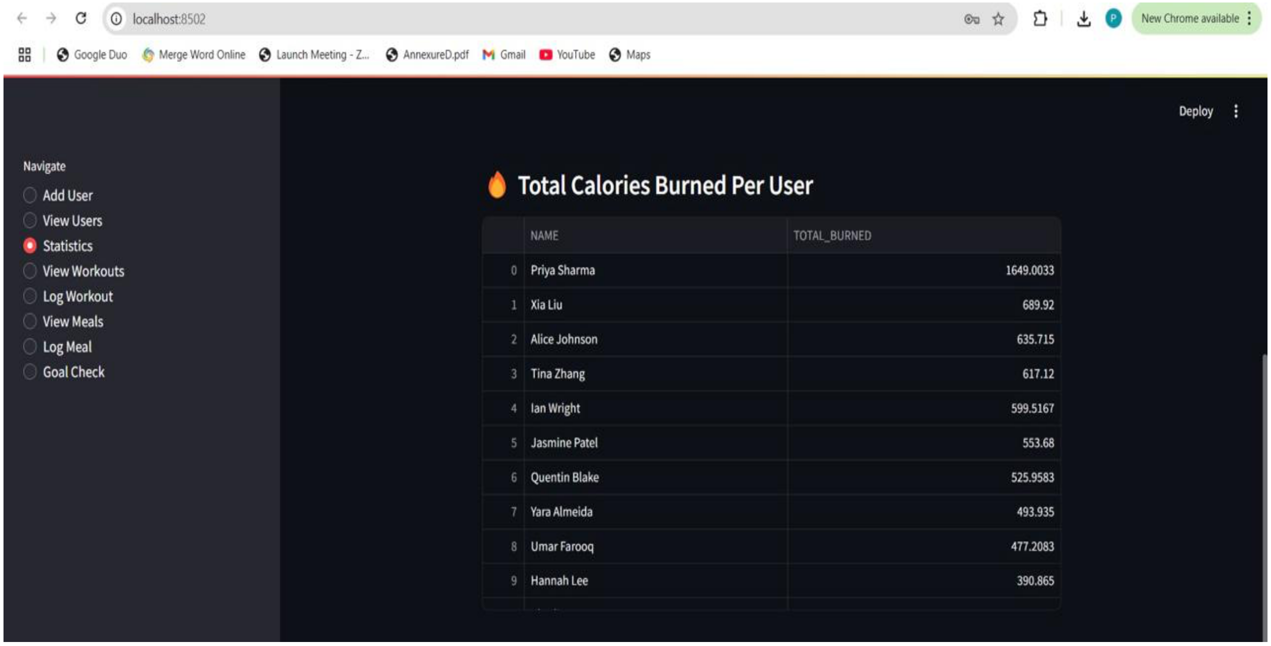
Fitness Tracker Dashboard

Fitness Statistics Overview

Total Calories Consumed Per User

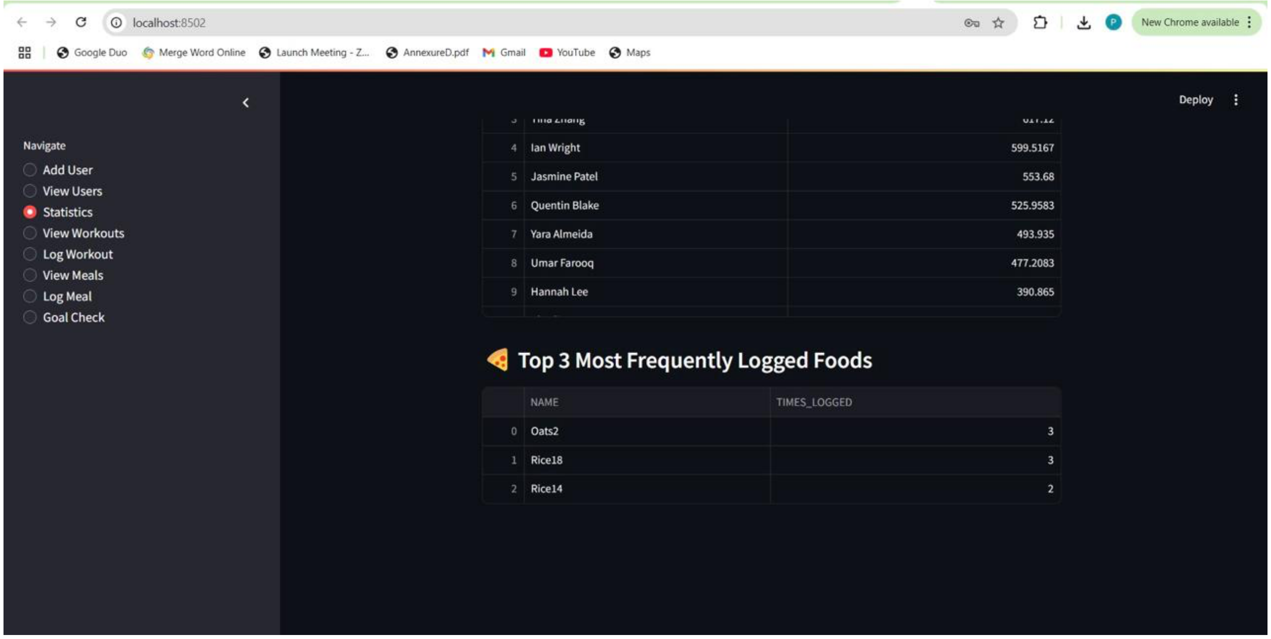
	NAME	TOTAL_CALORIES
0	Kevin Liu	1230.81
1	Alice Johnson	1038.4
2	Michael Clark	923.82
3	Ian Wright	674.25
4	Nina Torres	656.07
5	Xia Liu	630.48
6	Riya Kapoor	584.88
7	Fiona Gallagher	512.22
8	Hannah Lee	471.9
9	Tina Zhang	450.54

How much did you work today?



Total Calories Burned Per User

	NAME	TOTAL_BURNED
0	Priya Sharma	1649.0033
1	Xia Liu	689.92
2	Alice Johnson	635.715
3	Tina Zhang	617.12
4	Ian Wright	599.5167
5	Jasmine Patel	553.68
6	Quentin Blake	525.9583
7	Yara Almeida	493.935
8	Umar Farooq	477.2083
9	Hannah Lee	390.865



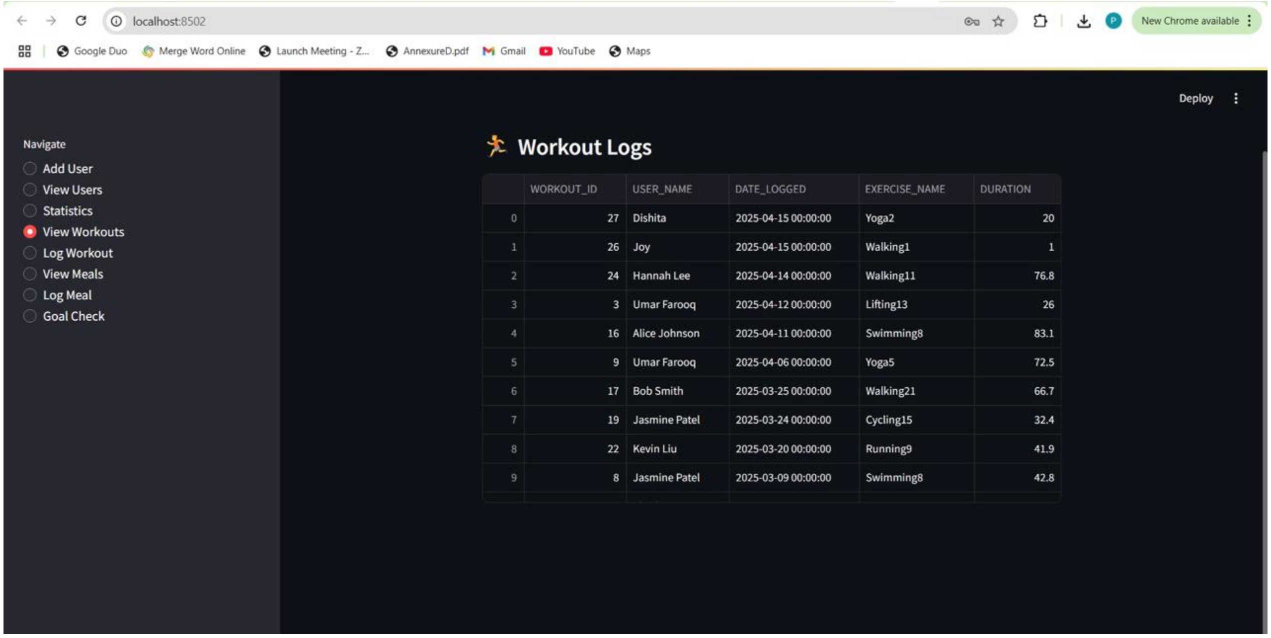
3	Tina Zhang	617.12
4	Ian Wright	599.5167
5	Jasmine Patel	553.68
6	Quentin Blake	525.9583
7	Yara Almeida	493.935
8	Umar Farooq	477.2083
9	Hannah Lee	390.865

🍕 Top 3 Most Frequently Logged Foods

	NAME	TIMES_LOGGED
0	Oats2	3
1	Rice18	3
2	Rice14	2

4) View Workouts Details

Keep track of your workout goals

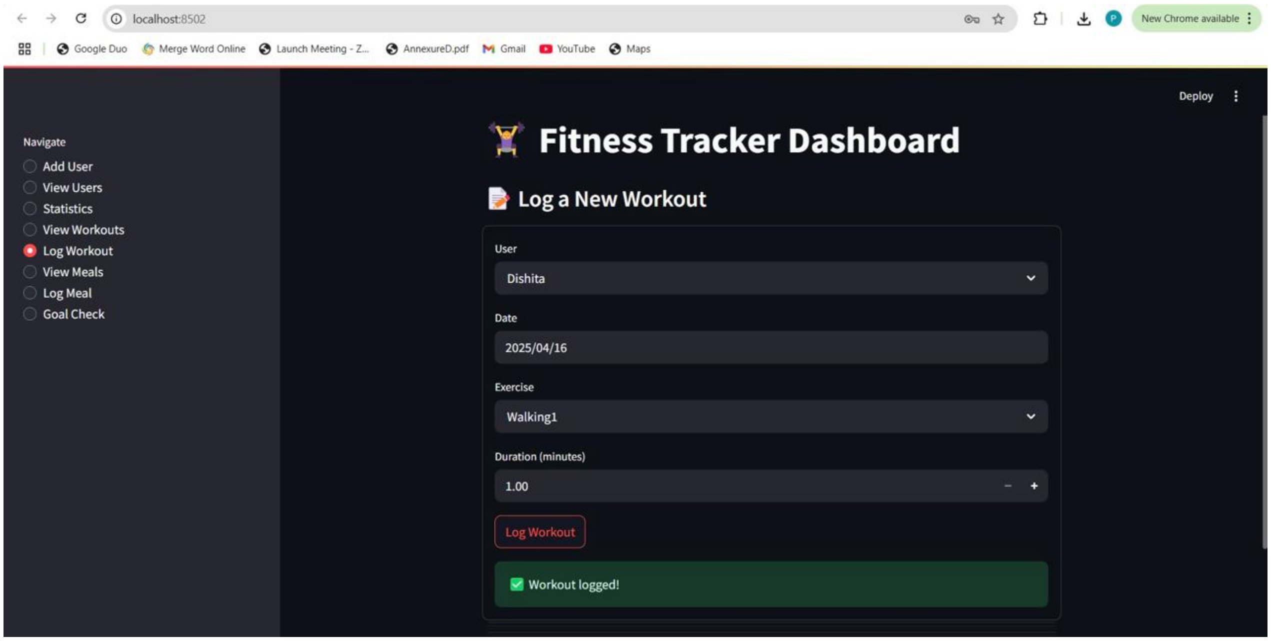


🏃 Workout Logs

	WORKOUT_ID	USER_NAME	DATE_LOGGED	EXERCISE_NAME	DURATION
0	27	Dishita	2025-04-15 00:00:00	Yoga2	20
1	26	Joy	2025-04-15 00:00:00	Walking1	1
2	24	Hannah Lee	2025-04-14 00:00:00	Walking11	76.8
3	3	Umar Farooq	2025-04-12 00:00:00	Lifting13	26
4	16	Alice Johnson	2025-04-11 00:00:00	Swimming8	83.1
5	9	Umar Farooq	2025-04-06 00:00:00	Yoga5	72.5
6	17	Bob Smith	2025-03-25 00:00:00	Walking21	66.7
7	19	Jasmine Patel	2025-03-24 00:00:00	Cycling15	32.4
8	22	Kevin Liu	2025-03-20 00:00:00	Running9	41.9
9	8	Jasmine Patel	2025-03-09 00:00:00	Swimming8	42.8

5) Log a Workout

Tell us every time you workout so we can help you achieve your goal



Fitness Tracker Dashboard

Log a New Workout

User
Dishita

Date
2025/04/16

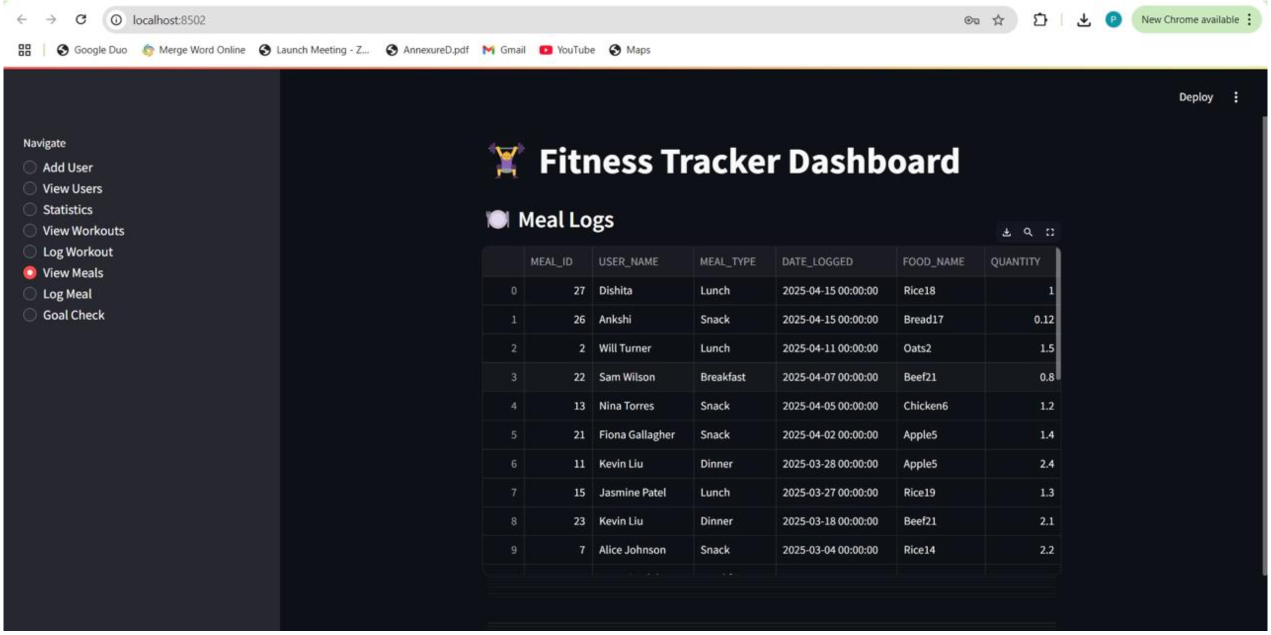
Exercise
Walking1

Duration (minutes)
1.00

Log Workout

✓ Workout logged!

6) View Meals



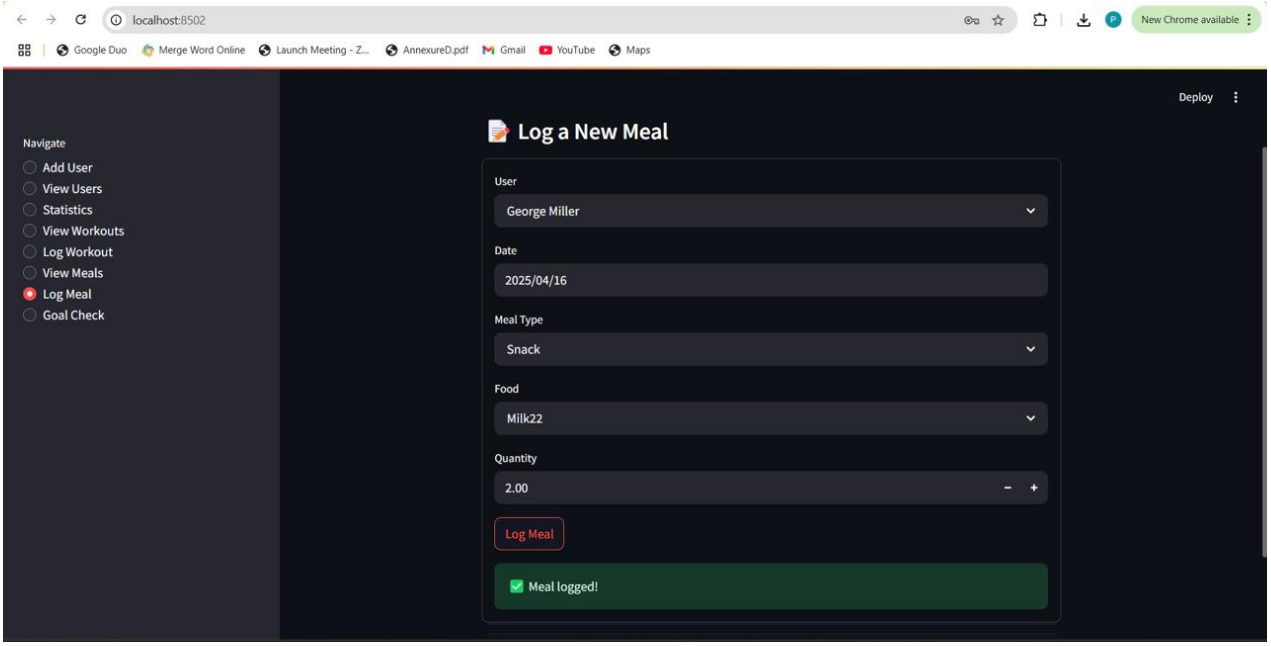
Fitness Tracker Dashboard

Meal Logs

	MEAL_ID	USER_NAME	MEAL_TYPE	DATE_LOGGED	FOOD_NAME	QUANTITY
0	27	Dishita	Lunch	2025-04-15 00:00:00	Rice18	1
1	26	Ankshi	Snack	2025-04-15 00:00:00	Bread17	0.12
2	2	Will Turner	Lunch	2025-04-11 00:00:00	Oats2	1.5
3	22	Sam Wilson	Breakfast	2025-04-07 00:00:00	Beef21	0.8
4	13	Nina Torres	Snack	2025-04-05 00:00:00	Chicken6	1.2
5	21	Fiona Gallagher	Snack	2025-04-02 00:00:00	Apple5	1.4
6	11	Kevin Liu	Dinner	2025-03-28 00:00:00	Apple5	2.4
7	15	Jasmine Patel	Lunch	2025-03-27 00:00:00	Rice19	1.3
8	23	Kevin Liu	Dinner	2025-03-18 00:00:00	Beef21	2.1
9	7	Alice Johnson	Snack	2025-03-04 00:00:00	Rice14	2.2

7) Log a Meal You Had

Don’t cheat, tell us all the junk you had tonight



📝 Log a New Meal

User
George Miller

Date
2025/04/16

Meal Type
Snack

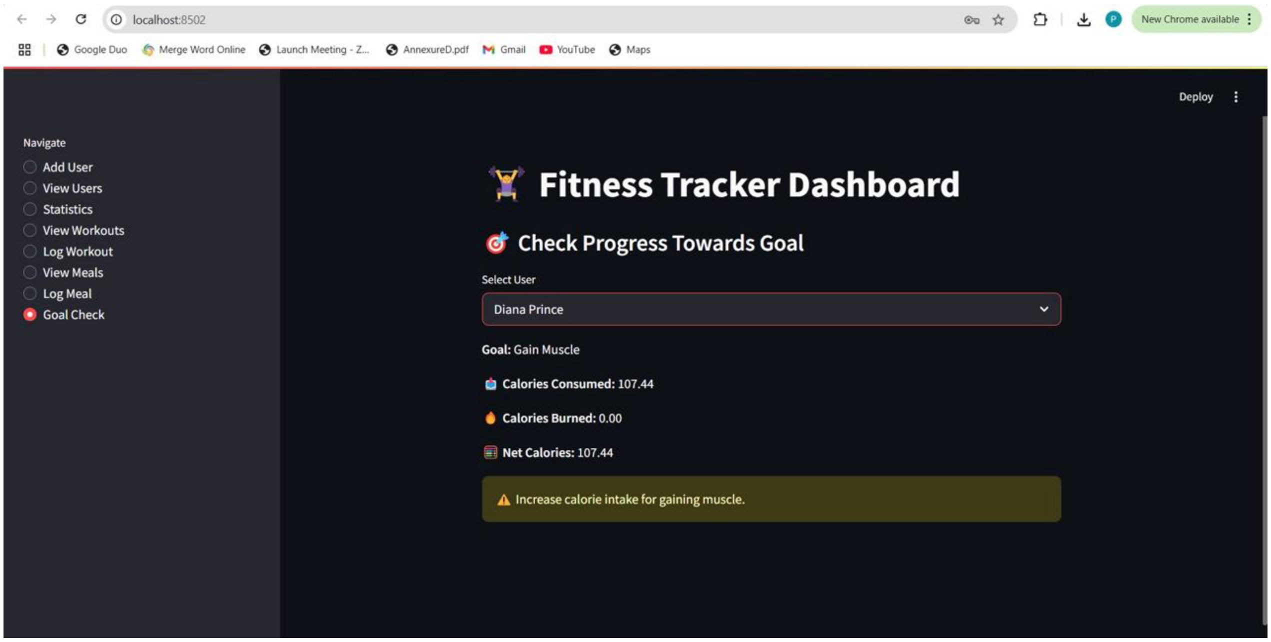
Food
Milk22

Quantity
2.00

Log Meal

✅ Meal logged!

8) Are You In the Right Path towards Your Health Goal? We’ve got you!



🏋️ Fitness Tracker Dashboard

🎯 Check Progress Towards Goal

Select User
Diana Prince

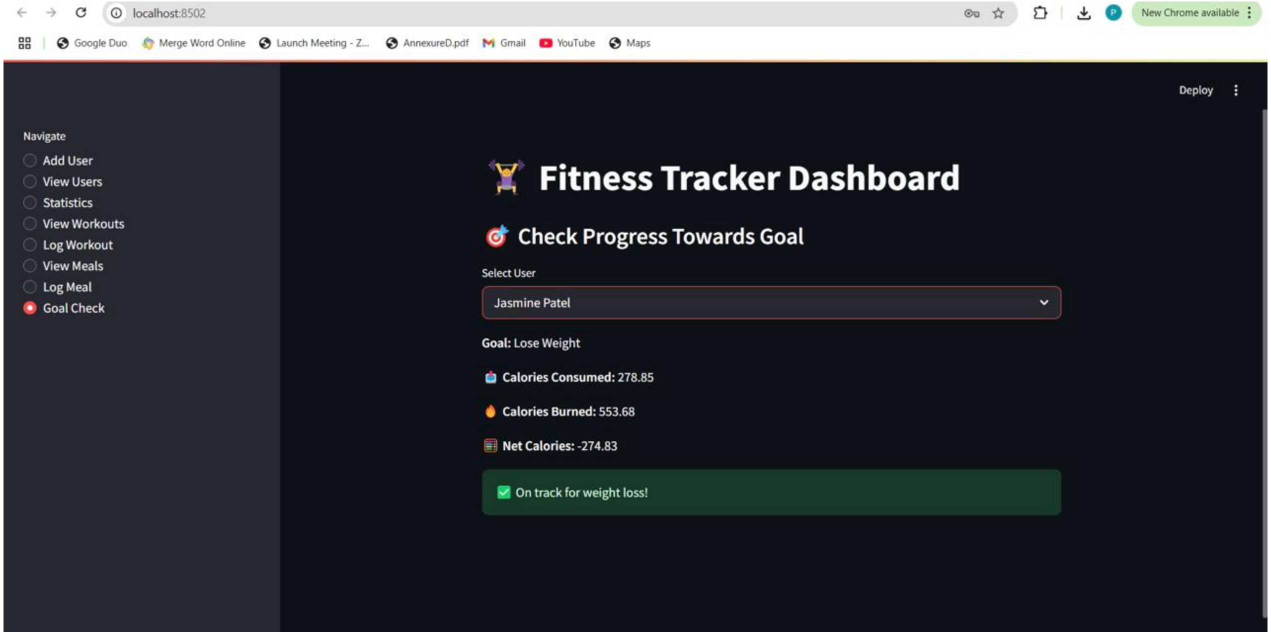
Goal: Gain Muscle

Calories Consumed: 107.44

Calories Burned: 0.00

Net Calories: 107.44

⚠ Increase calorie intake for gaining muscle.



Navigate

○ Add User
○ View Users
○ Statistics
○ View Workouts
○ Log Workout
○ View Meals
○ Log Meal
● Goal Check

🏋️‍♀️ Fitness Tracker Dashboard

🎯 Check Progress Towards Goal

Select User

Jasmine Patel

Goal: Lose Weight

🍱 **Calories Consumed:** 278.85

🔥 **Calories Burned:** 553.68

🧮 **Net Calories:** -274.83

✅ On track for weight loss!

REFERENCES

- **Oracle SQL Documentation**
https://docs.oracle.com/en/database/oracle/oracle-database/

- **PL/SQL Language Reference**
https://docs.oracle.com/en/database/oracle/oracle-database/21/lnpls/

- **Streamlit Documentation** – For UI design and Python web app integration
https://docs.streamlit.io/

- **Python Standard Library**
https://docs.python.org/3/library/