Software Design Specification (SDS)

Project Name: [RepTrack]
Prepared By: [Abdullah Yasser, Abdalla Tamer, Ahmed Baioumy, Mohamed Sami, Ziad Moutaz]
Date: [11/8/2024]

---

1. Introduction

1.1 Purpose

The purpose of this document is to describe the design, architecture, and technical specifications of RepTrack. It outlines the functionality, system components, and design decisions to be followed during the development process. It outlines the functionality and design decisions to be followed during the development process to ensure that the system is able to develop a secure, user-friendly platform.

1.2 Scope

This SDS covers the design and implementation details of the RepTrack. The website will perform the following major tasks:

- Track and log user workout data, nutrition and progress.
- provide coaches with tools to manage client workout plans, communicate with them and monitor their progress.
- Facilitate community interactions through posts, comments and news feeds

---

2. System Overview

The system consists of the following components:

- **Frontend**: Django
- **Backend**: Django
- **Database**: Supabase

---

3. System Architecture

3.1 Architectural Design

This project follows the client-server architecture, where:

- **Frontend** communicates with the backend using RESTful API calls.
- **Backend** interacts with the database to manage and retrieve data.

3.2 Data Flow

1. **User Interaction**: The user interacts with the UI to perform an action such as logging a workout or updating their profile.
2. **Request Processing**: The frontend sends an API request to the backend server.
3. **Data Handling**: The backend processes the request, interacts with the database, and fetches or updates the necessary data.
4. **Response**: The backend sends the response back to the frontend, updating the UI.

---

4. Database Design

The system will store data in relational (MySQL) and NoSQL (SSMS) with the following entities and relationships:

Table 1: [Gym]

Gym_ID: Primary key, unique identifier for each gym.
Gym_Name: Name of the gym.
Address: Address of the gym.
Relationship: One-to-many with Profiles (each user can be associated with one gym).

Table 2: [Current Gyms Subscribed]

Gym_ID: Foreign key referencing **Gyms(Gym_ID)**.
User_ID: Foreign key referencing **Users(User_ID)**.
Admin_ID: ID of the admin who manages the gym (manually assigned, not linked to Users table).
Business_Owner_Name: Name of the business owner of the gym.
Business_Owner_Contact: Contact information of the business owner.
Start_Date: Start date of the gym subscription.
End_Date: End date of the gym subscription.

Table 3: [Users]

- **User_ID:** {***unique identifier***}for each user
- **Name:** Name of the user
- **Email**: User's email address where he will receive all marketing emails/ sign in
- **Phone Number**: User's phone number
- **Account_Type:** Type of account that separates the user roles from each other EG(coach,l trainee, nutritionist)
- **Password:** user's password for login

Table 4: [Profiles]

- **Profile_id:**Primary key, unique identifier for each profile.
- **User_ID :**(PR.Foreign key) foreign key linked to {user table} to identify each profile
- **Profile_Pic**: profile picture of the user
- **Age:** age of the user

- **Bio:** Short Biography
- **Height:** User Height {Nullable}
- **Weight:** User Weight {Nullable}

  Relationship: One-to-one with Users, one-to-many with Gyms

Table 5: [Memberships]

- **Membership_ID:** Primary key, unique identifier for each membership.
- **User_ID :**(PR.Foreign key) foreign key linked to {user table} to identify each profile
- **Gym_ID:**(Foreign key(**Gym**(**Gym_ID**)) to identify where the user is current enrolled in
- **Plan_ID**: Foreign key referencing **Gym Plans(Plan_ID)**.
- **Start_Date:** the date that user starter hie membership
- **End_Date:** the date announced for membership end date
- **Relationship**: Many-to-one with Users and Plans.

Table 6: [Gym Plans]

- **Plan_ID**: Primary key, unique identifier for each gym plan.
- **Gym_ID**: Foreign key referencing **Gyms(Gym_ID)**.
- **Plan_Description**: Detailed information about the gym plan (e.g., monthly, yearly).
- **Price**: Price of the plan.
- **Duration**: Duration of the plan (e.g., 1 month, 1 year).

Table 7: [Nutritionst]

- **Nutritionist_ID** (Primary Key): Unique identifier for each nutritionist entry.
- **User_ID** (Foreign Key to **Users(User_ID)**): Links to a user with a nutritionist role.
- **Certificate**: Certification or qualification details.
- **National_ID**: National identification or registration number.
  Relationship: One-to-one with Users.

Table 8: [user Workouts Details]

- **Workout_ID**: Primary key, unique identifier for each workout session.
- **User_ID**: Foreign key referencing **Users(User_ID)** to identify the user performing the workout.
- **Workout_Date**: Date when the workout session took place.
- **Start_Time**: Start time of the workout.
- **End_Time**: End time of the workout.

- **Completed**: Boolean to indicate if the workout was completed (true or false).
- **System_Workout_ID**: Foreign key referencing **System_Workouts(System_Workout_ID)** to link the session to a predefined workout plan or routine from the system.

Table 9:[system Workout]

- **System_Workout_ID**: Primary key, unique identifier for each predefined workout routine.
- **Workout_Name**: Name of the workout (e.g., "Full Body Strength" or "Cardio").
- **Description**: Description of the workout routine, covering its objectives or benefits.
- **Exercises**: A list of exercises included in the workout, possibly in a JSON or structured format to accommodate multiple exercises.
- **Duration**: Estimated duration of the workout in minutes.
- **Difficulty_Level**: Indicates the difficulty level (e.g., beginner, intermediate, advanced).
- **Target_Muscle_Group**: Main muscle group targeted by the workout (e.g., "Upper Body," "Lower Body").
-

---

5. Technology Stack

- **Frontend**: Django
- **Backend**: Django
- **Database**: Supabase
- **Hosting**: Docker.

---

6. Testing Plan

6.1 Unit Testing

Each module and function, including data logging, user authentication and API endpoints, will undergo unit testing to ensure they are functioning as expected

6.2 Integration Testing

Integration tests will validate that different modules such as communication between frontend and backend, or backend and database work together as expected.

6.3 User Acceptance Testing (UAT)

End users, including trainees, coaches and nutritionists will be involved in testing the system to verify that it meets their requirements and expectations.

6.4 Performance Testing

Stress and load testing will be conducted to ensure the system can handle up to 10 concurrent users and all necessary operations without any degradation in performance

---

7. Conclusion

RepTrack is designed to fulfill the specified functional and non-functional requirements as described in this SDS. The design outlined here will ensure that the system is robust, scalable, and user-friendly, providing the intended value to its users.