CUSTOM WORKOUT PLANNER

INTRODUCTION :

- A custom workout planner is an exercise program designed to meet an individual's specific fitness goals, preferences, and abilities.
- Unlike generic workout plans, a custom workout planner takes into account factors such as fitness level, available equipment, time constraints, and personal objectives, such as weight loss, muscle gain, endurance, or flexibility.
- These plans can be crafted by fitness professionals or generated using online tools and apps that allow users to input their personal details and goals.

PROBLEM STATEMENT :

Custom workout planner

POC:

- CRUD (Create, Read, Update, Delete) for workout plans.
- Generate custom workouts for athletes.
- Generate custom workout plans based on athletes' specific needs.
- Monitor and track workout completion data

Real life Example :

Imagine a **Sports academy** or fitness center that offers personalized workout plans for professional athletes .Each athlete has unique physical goals and needs .such as strength training,endurance , or injury recovery . A **Custom Workout Planner** System can help create tailored workouts plans and track their completion.

MAIN COMPONENTS:

❖ CLASSES:

Workout plan: holds details about a workout plan.

Athlete: information about the athlete.

Workout Planner: Manages workout plans and athletes , generates custom workouts.

❖ Features:

- ✓ CRUD operations for workout plans.
- ✓ Generate custom workouts based on athlete specifications.
- ✓ Monitor and track workout completions.

We will also include unit tests using Python's unittest framework to test functionalities like CRUD operations and custom workout generation.

CORE FEATURES:

- Create Workout Plans
- Read Workout Plans
- Update Workout Plans
- Delete Workout Plans

Technology Stack:

- Programming Language: Python (due to its flexibility and wide library support for data manipulation, handling APIs, and building scalable applications)
- Backend : Flask/Django: For building REST APIs that handle CRUD

Monitoring Tools:

Prometheus/ Grafana: For monitoring the system’s performance and user activity . Workout Plan Customization

- **Algorithms:**

Machine Learning (using scikit -learn or Tensor Flow) to suggest customized workouts based on the athlete's historical data or fitness goals . operations (create, read, update, delete) for workout plans and athletes.

IMPLEMENTATION:

- Step 1: Workout Generation Logic :Create logic that evaluates athlete's current fitness level (perhaps through historical workout data) and provides a specific workout plan using predefined templates or a recommendation system.
- Step 2: Monitoring Completion Log the athlete's workout completion data . Create visual progress trackers that athletes can view through the frontend.
- Step 3: Testing Implement unit tests for CRUD operations and custom workout generation functions . Perform integration testing for the whole system.
- Step 4: Development and Monitoring the system on the selected cloud platform . Set up monitoring tools to keep track of API usage, response times, and other .

FUTURE SCOPE:

- 1. AI-Powered Customization Use Machine Learning to analyze user data and provide more refined and personalized workout plans based on progress over time.
- 2. Wearable Device Integration Integrate wearable devices to collect real-time fitness data for better customization of workout plans.
- 3. Expansion to a Marketplace Model Enable coaches/trainers to register on the platform, sell their workout plans, and manage their athlete groups.
- 4.Mobile app development : Create native mobile apps (iOS/Android) for a seamless and enhanced user experience.

Create a customer workout plan :

CRUD: Workout Plans

- The system stores workout plans for each athlete, allowing trainers to create new plans, view existing ones, update exercises, or delete old plans.
- **Create**: A new workout plan is created for an athlete who specializes in running, focusing on leg strength and endurance exercises.
- **Read**:The trainer views an athlete's existing workout plan to assess progress or suggest new exercises.
- **Update**: The trainer updates the plan to increase the intensity of the exercises after the athlete successfully completes the initial phases.
- **Delete**: Once the athlete's program ends, the old workout plans are removed to make space for new athletes

Generate custom workouts(athlete id):

Each athlete has a specific goal, whether it is to build strength, increase speed, or recover from an injury. The system generates a custom workout plan for an athlete based on their physical requirements, injury history, and goals.

- For example, a sprinter will need a different plan compared to a basketball player recovering from a knee injury.

Example:

- **Emily,** as printer, needs to improve her speed for an upcoming competition. The system generates a **high-intensity interval training (HIIT)** plan combined with strength training to boost her explosive power and sprinting ability. Her custom workout includes sprint drills, plyometrics, and leg-strengthening exercises.

Monitor workout completion:

- The system also allows tracking workout progress by monitoring which sessions have been completed by the athlete. The coach or trainer can track if the athlete is following the schedule and how many sessions have been completed each week.

Example:

- Emily has a 4-week workout plan. The system tracks that in the first week, she successfully completed **5 out of 6 planned sessions.** The trainer can then adjust the next week's schedule based on her performance, ensuring that the training intensity aligns with her progress

CONCLUSION :

This Solution provides a basic structure for managing a custom workout planner. It supports essential operations like creating , updating, and deleting workout plans, automatically generates plans based on fitness levels, and monitors workout completion . The POC can be expanded further to include additional features such as integration with fitness apps , more complex customization logic, and detailed progress tracking.

THANK YOU