KarmaQuest: Using Real-Time Video Processing and AI-Based Movement Analysis to Optimize Individual Fitness and Health

Sahiti Pattamatta, Samyuktha Gellagigari, Kavyasree Gandla, Saiteja Chedadeepu, Vinay Meda

Department of Computer Science, Florida Atlantic University

ABSTRACT

The mobile fitness application KarmaQuest provides a method to instantly analyse one's posture while exercising using real-time video analytics along with artificial intelligence. The app closes the gap between reaching a personal fitness level and having a personal workout trainer to help reach those goals, using Google's MoveNet pose detection technology to monitor 17 key muscle groups in the body as a user performs a set of squats, push-ups or lunges (among other exercises). The app determines the angles at which a user's joints are bending during those movements, compares those angles to what ideal form looks like, and then produces an instant report for the individual to improve upon their technique and reduce the chances of injury. KarmaQuest also creates a 7-day custom workout and meal plan according to an individual's personal fitness goals. KarmaQuest allows users to select between three different account types: a regular account, a trainer account, and an administrator account. This creates a full range of capabilities in terms of fitness management. Built on the React Native mobile frontend, and Flask based back-end application, KarmaQuest is an example of how to use computer vision within the consumer health technology space.

Keywords: *pose detection, fitness tracking, computer vision, MoveNet, real-time feedback, mobile application*

I. INTRODUCTION

Mobile fitness applications that aim to assist consumers in achieving their health and fitness objectives have seen an explosive amount of growth in the fitness industry. Most applications currently available on the market only provide users with basic workout logging and tracking capabilities and do not have built-in features to provide users with intelligent feedback similar to that of a professional trainer. The leading factor in many exercise-related injuries is improper exercise form; however, the majority of individuals who exercise do not have access to a trainer or instructor due to accessibility and cost concerns.

KarmaQuest solves this issue by offering AI-powered form analysis via real-time feedback on user's smartphones. KarmaQuest utilizes computer vision and machine learning to monitor propulsion form expectations for exercise types while giving users immediate corrective information for their deviation from the expected form of execution. KarmaQuest allows individuals to attain access to high-quality instruction in the fitness industry while minimising the likelihood of injury caused by improper technique.

KarmaQuest provides two potential modes of analysing form during workouts: real-time camera feedback while executing an exercise; and uploading a video of the exercise to KarmaQuest for analysis. This provides flexibility for users in terms of personal preferences and locations where they may train. KarmaQuest provides more than just form analysis; it creates personalized workout and meal plans; tracks progress; and supplements the trainer-client relationship with professional coaching services.

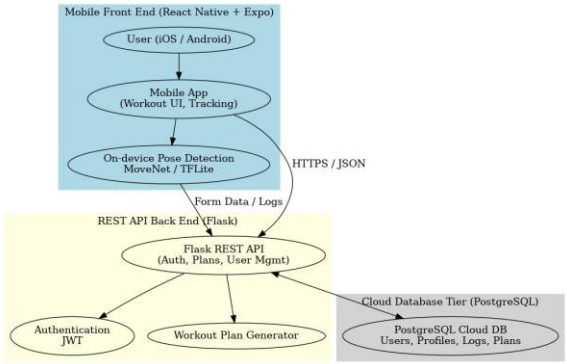
This paper will provide the overall design, implementation, and evaluation details of the KarmaQuest program. Section II will articulate the methodology and architecture for the KarmaQuest program. Section III will articulate the method of implementing the KarmaQuest program. Section IV will provide the overall response to the KarmaQuest program and discuss the results.

II. METHODOLOGY / SYSTEM DESIGN

A. System Architecture

The architecture of KarmaQuest is based on three different tiers, which consist of the mobile front end, REST API back end, and a cloud-based database. The mobile app was developed in React Native with Expo for cross-platform deployment on both iOS & Android. A RESTful API was also created using Python (with Flask) to handle user authentication requests, create workout plans and manage users.

The database tier contains user profiles, workout sessions, logs of exercises, and generated plans, all of which use PostgreSQL and are stored in the cloud. As far as the design is concerned, the architecture is based on a philosophy of separation of concerns; that is, the requirements of real-time performance for AI pose detection can be met by executing this functionality locally on a mobile device, while more intensive processing for generating workout plans can be performed on back-end servers.



B. AI Pose Detection Pipeline

Google's MoveNet Lightning System, which allows for analysis of human motions, incorporates core artificial intelligence capabilities to enable mobile device-based data collection on human movements. The MoveNet Lightning System is implemented in TensorFlow Lite for mobile devices, enabling the detection of 17 human body key or joints: the nose, eyes, ears, shoulders, elbows, wrists, hips, knees and ankles. Each joint provides an xy coordinate indicating where the joint is located (on the x-axis and y-axis) within an image. The xy coordinates are scaled between "0" and "1" with an associated confidence score generated by the AI.

Pose detection is performed by processing the camera frames at approximately 30 frames per second. The individual frame is resized from the original camera dimension to fit the MoveNet Lightning model's required pixel dimensions of "192 x 192" pixels. The model will then output the xy coordinates of the key points detected within that frame. These coordinated are then processed by a Motion Analysis Module.

C. Form Analysis Algorithm

The calculation of joint angles as part of determining an athlete's form is conducted through vector geometry. To calculate the angle of a joint comprised of three keypoints (A, B, C) with B being the vertex point for the angle, the dot product formula is utilized. The relevant joint angle(s) for different exercise types are calculated using this formula.

In the case of the squat exercise, the angle of the knee (hip-knee-ankle), the angle of the hip (shoulder-hip-knee) and the angle of the back (ear-shoulder-hip) are all monitored. Ideal squat form has a knee angle of between 85-95 degrees at its lowest point, hip angle of between 70-90 degrees, and back angle of between 145-165 degrees in order to maintain a neutral spine. Specific feedback messages such as "Knees caving inward" or "Back rounding - keep chest up" are sent to athletes when they deviate from the ideal joint angles.

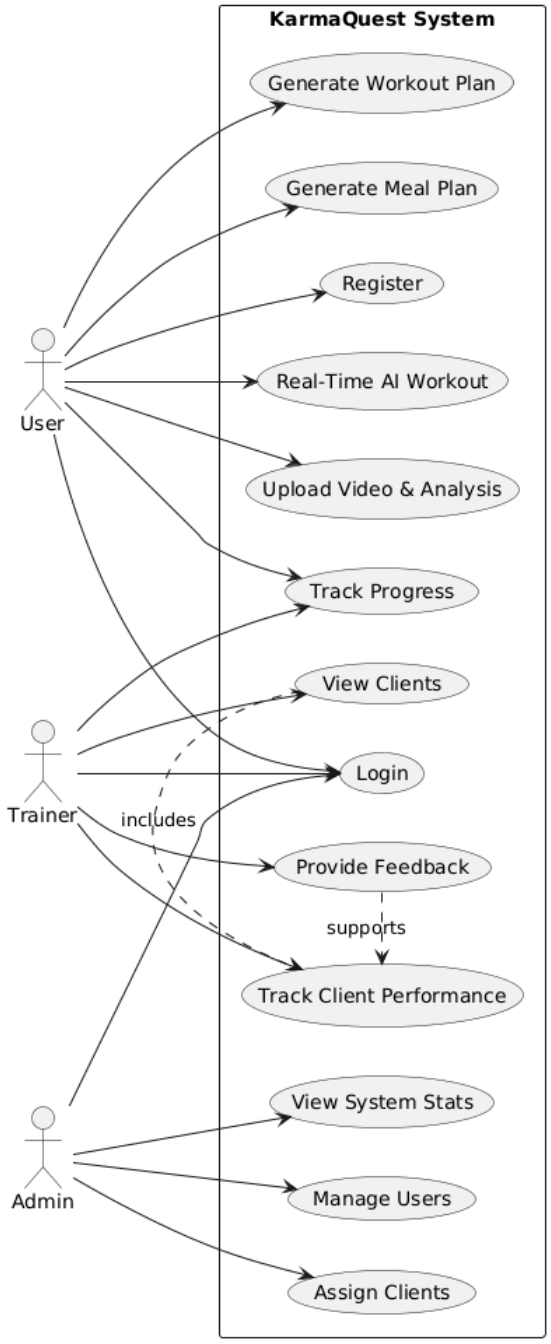
D. Repetition Counting

The state machine used to monitor the phase of exercise during repetition counting, uses 3 states: START, DOWN and UP. For Squats the signal to move from START to DOWN is when the user's knee angle falls below 95 degrees. The transition signal from DOWN to UP and thus to increment the count of repetitions occurs when the user has an angle of greater than 165 degrees in their knee. A hysteresis mechanism is used to eliminate erroneous counting due to sensor noise, having the system require three consecutive frames in each state.

E. User Roles and Use Cases

The application supports three types of users with different limitations of access. Regular users can create accounts, complete workouts via AI analysis, design personalized workout plans and track their overall performance; Trainers can see clients assigned to them, track performance statistics and workout history, and provide feedback; and Admins can create trainers, manage users, assign clients to trainers, and view statistics on the system.

Some of the key use cases for the application include: User Registration and Profile Creation, AI Workout with Real Time Form Analysis, Video Upload and Analysis of Workout Performance, Generation of Workout Plans based on User Goals, Generation of Meal Plans with Caloric and Macronutrient Values, Progress Tracking with Weight and Measurements, Tracking Clients as a Trainer, Managing Users as an Administrator, and Synchronization of Real Time Data between Users.



III. IMPLEMENTATION

A. Technology Stack

With React Native and Expo for cross-platform mobile development, the frontend mobile app includes example camera access functionality through the use of expo-camera and on-device AI inference via the TensorFlow.js and pose-detection libraries, as well as screen navigation with React Navigation, while also providing HTTP communication between the backend API and the frontend application via the Axios library.

The backend API is implemented in Python using the Flask framework, with SQLAlchemy as the ORM to manage entity relationships and define database operations, the use of Flask-JWT-Extended for implementing token-based authentication, and database schema versioning management via Flask-Migrate. OpenCV is used for frame-by-frame video processing in the upload analysis functionality.

The database technology being used in the project is PostgreSQL hosted on a cloud solution, and the schema includes a total of six main tables: Users, UserProfiles, WorkoutSessions, ExerciseLogs, WeeklyWorkoutPlans, and WeeklyMealPlans. There are also foreign key relationships with cascade delete constraints to ensure referential integrity within the database schema.

B. Authentication and Security

Authentication allows users access through the JWT (JSON Web Token) method which utilizes separate refresh and access tokens during authentication. The access token has a limited lifespan of 2 hours while the refresh token can be used for 30 days before expiration. Passwords are saved securely via a hashing process using bcrypt, with the addition of a unique salt for each password. Additionally, the application enforces the following password complexity rules: at least 8 characters long, including a Capital letter, lowercase letter, and a numerical value.

Each user of the application has a Role-based access control (RBAC) that limits access to endpoints based on their assigned role(s). The user roles will be verified by using custom decorators which will check to see the attached claim(s) on each users JWT before they are granted access to any secured resources. All API communications made by the end-user's device will be made over HTTPS, ensuring that data in transit is encrypted via SSL.

C. Database Schema

The Users table stores all the necessary data for each user to maintain an account (user_id (UUID Primary Key), Email, Password Hash, First Name, Last Name, User Role (User, Trainer, Admin), and timestamps). The User Profiles table will have a foreign key reference to the Users table and will contain all fitness-related fields (current weight, height, target weight, fitness goal, fitness level).

WorkoutSessions capture the information describing a user's workout session (i.e., session_id, user_id, session_date, duration, total calories burned, and average posture score). ExerciseLogs hold all information on the individual exercises performed during a session (i.e., exercise type, sets completed, correct and incorrect reps, average form score, and detected posture issues) and are stored in JSON format.

D. Testing

Out of thirteen (13) non-functional requirements (NFR's), eleven (11) of the NFR's were completely satisfied. Average Response Times per API request were below two hundred (200) milliseconds (NFR1), Pose Estimation was maintained at a frame rate of 12+ frames per second (NFR2) on all devices tested, Modular Architecture allows for

independent updates to individual components (NFR10), All communication with any user device will be done over HTTPS and Passwords will be encrypted while stored (NFR5), The Mobile Element of the Application is supported by Android 10.0 and up and iOS 14.0 and up (NFR9).

IV. RESULTS

A. Functional Requirements Completion

The software successfully developed and implemented all functional requirements included in the specification. User Account Management, Real Time Monitoring of Exercise Performance, Workout Summary/Results, (providing an overview of what was done; how well the user did) Personalized (weekly workout and meal plans based on user's own goals), Cloud Data Management (by automatically keeping the user's data synchronized between devices via a PostgreSQL database), and Administration Tools (providing system dashboards to the administrators, monitoring the operations of the system, and managing the Users).

B. Non-Functional Requirements Status

The software has also achieved completion of 11 of the 13 non-functional requirements. For example, API Response Time (of less than 200 ms) was achieved (NFR1); Pose Estimation (Pose Estimation) - 12 fps or greater on the devices tested (NFR2); Modular Design allows for independent updating of the software components (NFR10); All Communication between devices are performed using HTTPS with encrypted passwords stored in the Database (NFR5). The software has been tested and is compatible with Android 10+ & iOS 14+ (NFR9).

The remaining two non-functional requirements were incomplete, due to scheduling constraints. The integration of Google Fit and/or Apple Health will not be completed until release 2.0, because it will take at least 4-6 weeks for OAuth Authorization to be completed (NFR12). Full compliance with GDPR has been achieved by providing the capability to delete data from within the application. However, full customer compliance with GDPR requirements will require a legal review, and is beyond the scope of this project (NFR13).

C. Additional Features Implemented

In addition to fulfilling the initial requirements, many improvements were made. The video upload and real-time camera analysis provide flexibility in the types of exercises being performed. When processed, videos can include skeleton overlays which assist with the feedback review process visually. The form feedback system breaks down into actionable items instead of generic rank-based scoring. The meal planning module incorporates a daily caloric intake based on weight goals and suggested macro-balanced meals based on those caloric needs.

V. CONCLUSION

KarmaQuest has shown that AI-based fitness coaching is a viable solution for consumers on their mobile devices. Utilizing the MoveNet pose detection model along with a proprietary methodology for analyzing proper form, KarmaQuest gives a method to receive immediate feedback on how to perform exercises correctly that was only available in the past through personal trainers. The architecture of the three-tier architecture effectively segregated the areas of mobile AI calculation from backend business logic and the persistence of data in the cloud.

KarmaQuest met all functional requirements and 11 out of 13 non-functional requirements. Notable items include Real-time pose detection at 30 frames per second with detailed form analysis and feedback, and recommendations for a personalized plan. The system supports all 16 use cases across three user roles, including real-time data synchronization.

In the future, we intend to complete Google Fit and Apple Health integration with our app, continue building our growing exercise library, implement social sharing features to send workouts to friends, and evaluate using advanced pose recognition in order to improve the accuracy of our apps and improve the data collected from them. Our app's modular architecture means all of these changes and additions can be made without major reworking of other components.

REFERENCES

[1] Google, "MoveNet: Ultra fast and accurate pose detection model," TensorFlow Hub, 2021.

[2] React Native Documentation, "Getting Started," Meta Platforms, Inc., 2023.

[3] Flask Documentation, "Flask Web Development," Pallets Projects, 2023.

[4] SQLAlchemy Documentation, "SQLAlchemy ORM Tutorial," SQLAlchemy, 2023.

[5] TensorFlow Lite Documentation, "TensorFlow Lite for Mobile," Google, 2023.