CAPSTONE PROJECT REPORT

(Project Term: January- May 2025)

AI-POWERED PERSONALIZED FITNESS TRAINER

Submitted by

R.Vishwanath	**Registration Number : 12108780**
Arjun Dev R	**Registration Number : 12104884**
Varun Karotiya	**Registration Number : 12105636**

Project Group Number : 18

Course Code : CSE461

Under the Guidance of

Shivangini Gupta

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

May 2025

PAC FORM

DECLARATION

We hereby declare that the project work entitled **AI-Powered Personalized Fitness Trainer** is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science Engineering with Specialization in AI and ML from Lovely Professional University, Phagwara, under the guidance of (Name of Faculty Mentor), during August to November 2022. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: 18

Name of Student 1: R.Vishwanath
Registration Number: 12108780

Name of Student 2: Arjun Dev R
Registration Number: 12104884

Name of Student 3: Varun Karotiya
Registration Number: 12105636

(Signature of Student 1) Date:

(Signature of Student 2) Date:

(Signature of Student 3) Date:

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science Engineering with Specialization in AI and ML from Lovely Professional University, Phagwara.

Signature and Name of the Mentor

Designation

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date :

ACKNOWLEDGEMENT

We humbly take this opportunity to present our votes of thanks to all those guidepost who really acted as lightening pillars to enlighten our way throughout this project that has led to successful and satisfactory completion of this study. We are really grateful to our HOS Mr. Rajeev Sobti and our mentor Shivangini Gupta for providing us with an opportunity to undertake this project in this university and providing us with all the bright and innovative ideas for making our project a really worthwhile of running in an organization. We are highly thankful for his active support, valuable time and advice, whole-hearted guidance, sincere cooperation and pains-taking involvement during the study and in completing the capstone project within the time stipulated.

We are thankful to all those, particularly our friends , who have been instrumental in creating proper, healthy and conductive environment and including new and fresh innovative ideas for us during the project, without their help, it would have been extremely difficult for us to prepare the project in a time bound framework.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 Introduction

Good health and wellness preservation stands as a critical necessity during our fast-changing modern environment. Modern technology creates a lifestyle pattern that involves less physical movement leading to increased pressure and more weight gain and heart-related diseases and mental disorders. Non-communicable diseases which can be prevented by life style changes represent over 70% mortality cases worldwide per the World Health Organization (WHO). Modern society requires personal health management to function rather than view it as desirable option. Most traditional fitness training approaches force users to either schedule physical classes or consult personal trainers or follow standard diet plans without meeting personal requirements. People find these methods too costly or difficult to reach because of time or location barriers. The methods commonly dismiss mental health needs that are equivalent to their focus on physical well-being. Digital health technologies have developed into effective tools during the previous years to help people maintain their health and prevent illness. People monitor their physical movement and measure their food consumption while accessing guided meditation through mobile applications as well as wearable devices and smart assistants. Current developments have shown improvement yet a unified AI system which serves all health aspects including exercise and nutrition and mental wellness at personal levels in an integrated manner remains absent. AI has demonstrated extraordinary abilities to revolutionize healthcare delivery as well as wellness services together. AI presents the possibility to scan extensive user data collections which leads to discovering patterns along with danger forecasting while creating individualized recommendation systems. Different machine learning algorithms which include reinforcement learning and deep learning allow systems to modify themselves through the process of learning from user behavior patterns to enhance results. The technology of NLP brings together interactive empathetic communication which enhances the accessibility and user-friendliness of AI applications. These technological combinations will transform fitness and wellness processes for individual users. This project addresses health because it recognizes health exists across multiple dimensions which interconnect with one another. Everyone who exercises may face difficulties with their eating pattern while coming under mental strain. People maintaining a healthy diet plan may struggle to find enough motivation for doing their regular physical exercise. The path toward excellent health cannot be considered complete when all these critical factors receive separate treatment. The current strategy of fitness applications divides their programs into isolated sections. Each individual fitness application addresses only a single aspect like workouts and calories while mental health stands alone as a third separate application. A deficient system integration results in user experience problems and prevents the creation of a clear health improvement method. Static programs deployed by various mobile health applications do not adapt to the user's evolving requirements and skills as well as personal preferences. The proposed project solves previous limitations through development of an AI-based fitness trainer system that generates adaptable fitness plans

with nutritional guidance together with mental health assistance. The main objective is to build a virtual wellness companion competent in understanding users completely through real-time interaction to evolve from user behavior thus providing prompt and relevant guidance for improved health. The customized method operates through adaptive programming which aims to enhance user participation along with consistent behavior resulting in better health results.

1.2 Dataset Explanation

Data set of Yoga/Strength Training

Sr No	Classes	Images
1	Down Dog	223
2	Goddess	180
3	Plank	266
4	Tree	160
5	Warrior2	252

Table 1: Yoga Dataset Description

Sr No	Classes	Images
1	Push up movement	150
2	Glute bridge up movement	150
3	Squat up movement	145
4	Push down movement	142
5	Glute bridge down movement	141
6	Squat down movement	145

Table 2: Strength Training Dataset Description

Yoga

Warrior 2/ Virabhadrasana II

The standing yoga posture Virabhadrasana II is recognized as Warrior II Pose to forge stamina and stability alongside strength development. This posture derived from the mythic warrior Virabhadra represents both physical and mental readiness for struggle and it maintains its name in dedication to him.

Benefits:

- Strengthens legs, ankles, and shoulders
- Increases stamina and endurance
- Improves balance and stability
- Enhances focus and determination

Figure 1: Virabhadrasana II

Tree pose/Vrikshana

As a Sanskrit name Vrikshasana it represents Tree Pose which offers a fundamental balance pose to stabilize the body while enhancing focus and grounding. Practitioners can improve both their focus and posture and enhance strength in their legs and core by engaging in this pose that draws inspiration from the stable foundation of a tree.

Benefits:

- Improves balance and coordination
- Strengthens thighs, calves, ankles, and spine
- Enhances concentration and mental clarity
- Promotes a sense of grounding and calm

Figure 2: Vrikshana

Plank/Phalakasana

The Sanskrit name of Plank Pose is Phalakasana which stands as a basic yoga stance for developing core power and providing overall body strength and stability. Despite its basic appearance this posture demonstrates exceptional power for targeting several muscle groups in the body.

Benefits:

- Strengthens core, arms, shoulders, and legs
- Improves posture and balance
- Enhances endurance and stability
- Builds awareness of body alignment

Figure 3: Phalakasana

Goddess/Utkata konasana

Utkata Konasana is a Sanskrit term that refers to Goddess Pose while serving as a strong standing yoga position which stretches the hips and builds lower body muscles and generates core strength along with stability. People commonly connect this pose to female spiritual energy and earth connection.

Benefits:

- Strengthens thighs, glutes, calves, and core
- Opens hips and chest
- Improves balance, endurance, and stability
- Energizes and empowers the body and mind

Figure 4: Utkata konasana

Down dog/Adho Mukha svanasana

The Sanskrit name Adho Mukha Svanasana identifies Downward Facing Dog as a fundamental yoga posture which many individuals recognize. This posture extends through the whole body and grows its strength before soothing mental strain.

Benefits:

- Stretches hamstrings, calves, spine, and shoulders
- Strengthens arms, legs, and core
- Improves posture and circulation
- Calms the nervous system and relieves stress

Figure 5: Adho Mukha svanasana

Strength Exercise

Push Up

Push-Up operates as an essential strength exercise despite not being a traditional yoga position because it strengthens both the upper body and core muscles. A properly executed Chaturanga Dandasana in yoga shares considerable similarities with Push-Up due to their shared alignment principles.

Benefits:

- Strengthens chest, shoulders, triceps, and core
- Improves posture and stability
- Enhances muscular endurance
- Requires no equipment, easy to modify

Figure 6: Push Up

Glute Bridge

Glute Bridge stands as a basic yet powerful strength exercise which specifically engages glutes and hamstrings together with lower back. Popular in yoga-based workouts and physical therapy applications as well as strength training protocols because it enables development of posterior chain strength and better hip stability.

Benefits:

- Strengthens glutes, hamstrings, and lower back
- Improves core stability and hip mobility
- Helps correct posture and relieve lower back tension
- Can reduce risk of injury in lower body activities

Figure 7: Glute Bridge

Squat

Squat functions as a primary lower-body exercise which develops strength together with mobility and stability for the hip along with leg areas and core components. People commonly use this body position in their fitness routines as well as their yoga-inspired workouts and athletic conditioning programs due to its natural human posture.

Benefits:

- Strengthens quads, hamstrings, glutes, and calves
- Engages core and improves balance
- Enhances joint flexibility and mobility
- Supports functional movement and athletic performance

Figure 8: Squat

Meal Planner Dataset

The Meal planner module of the AI-Powered Personalized Fitness Trainer uses the structured dataset IndianFoodDataset_Cleaned.csv as its main dietary recommendation resource. The data set includes numerous healthful food selections which cover various cooking styles together with different dietary requirements. Different types of food recipes exist in this dataset as it contains organized attributes for multiple categories. User input directs the backend system to retrieve recommendations from the SQLite database containing the stored dataset. When users apply their search criteria including cuisine type and diet type and course category the system retrieves relevant meal suggestions with complete recipe details displayed to them.

Users receive culturally appropriate healthy easy-to-prepare meal ideas from the meal planner because it uses localized and diverse database information which supports their nutritional and fitness objectives.

1.2 Exploratory Data Analysis

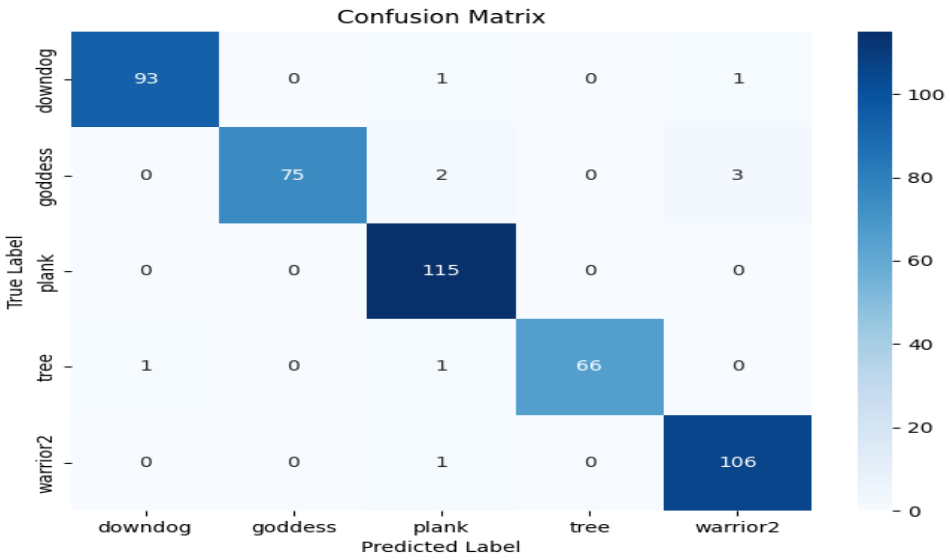

Figure 9: Push Up

Figure 9 shows the performance evaluation of a yoga pose classification model through its detailed assessment of five yoga positions including Downward Dog, Goddess, Plank, Tree and Warrior II. The matrix analyzes test dataset true labels against predicted labels to give both performance evaluation and evaluation of potential errors made by the model.

The correct number of instances for each pose is shown by the elements aligned with the main diagonal of the matrix. The model demonstrated strong predictive ability in identifying three yoga poses including Downward Dog which corresponded to 93 instances and Plank at 115 instances as well as Warrior II at 106 instances. The model successfully identified Tree pose sixty-six times while reaching seventy-five accurate Goddess pose classifications. The model shows consistent performance across most categories of poses based on these measured values where Plank and Warrior II demonstrate the highest reliability.

The misclassification instances appear in the elements that lie outside the main diagonal. The model displays a noteworthy confusion between Goddess and Plank since it falsely identified Goddess twice as Plank and misidentified one Tree pose as Plank as well. One pose from Downward Dog received two incorrect classifications as Tree and Warrior II and Goddess received 3 instances of incorrect classification as Warrior II while Tree received one misidentified instance as Downward Dog and Plank. The misidentified data points could indicate either pose similarity by angular features or training data limitations. The prediction distribution demonstrates that the model makes errors only when posing poses that share few similarities with each other because off-diagonal entries remain sparse. Test data analysis shows that the model succeeds in making predictions because the diagonal elements of the confusion matrix are much higher than the total test instances (465). This indicates an estimated accuracy that exceeds 90% for the main categories. The SVM with an RBF kernel demonstrates effective extraction of angle-based features that distinguish the yoga poses yet additional improvements seem possible to resolve observed confusion points.

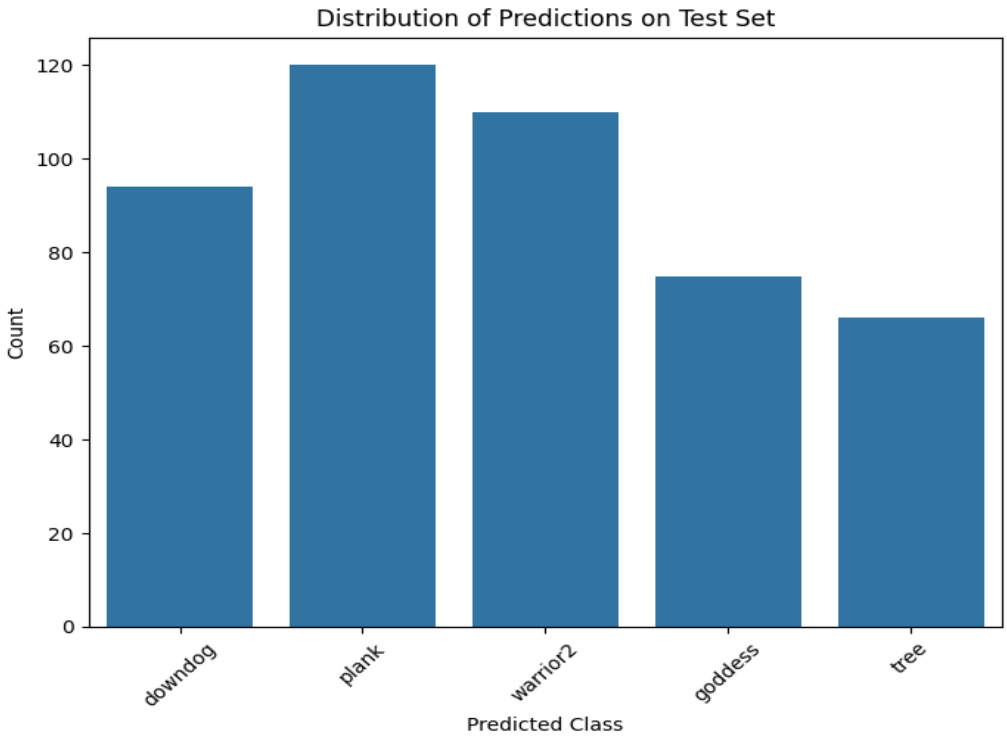

Figure 9: Push Up

The bar chart depicted in Figure 2 illustrates the distribution of predicted yoga poses across the test dataset, comprising 465 samples, for the developed classification model. The chart categorizes predictions into five yoga poses: Downward Dog, Plank, Warrior II, Goddess, and Tree, with the y-axis representing the count of predictions and the x-axis listing the predicted classes. The distribution reveals a varied frequency of predictions among the poses. Plank emerges as the most frequently predicted pose, with approximately 120 instances, indicating a strong tendency of the model to classify samples as Plank. Warrior II follows closely with around 110 predictions, suggesting a similarly high confidence in this pose. Downward Dog is predicted in about 100 instances, reflecting a robust recognition rate for this pose as well. In contrast, Goddess and Tree poses show lower prediction counts, with approximately 80 and 70 instances, respectively. This imbalance may reflect either a higher prevalence of Plank and Warrior II poses in the training data or the model's greater proficiency in distinguishing these postures based on the angle features. The uneven distribution highlights potential biases in the model, which could stem from the composition of the training dataset or the inherent variability in pose execution. For instance, the higher counts for Plank and Warrior II might indicate that these poses have more distinct or consistent angular patterns, making them easier to classify. Conversely, the lower counts for Goddess and Tree suggest possible challenges in accurately capturing their unique characteristics, which could be addressed by augmenting the training data with additional examples or refining the feature extraction process. This visualization underscores the need to evaluate the model's performance across all classes to ensure equitable prediction capabilities, potentially guiding future improvements in data collection or model tuning.

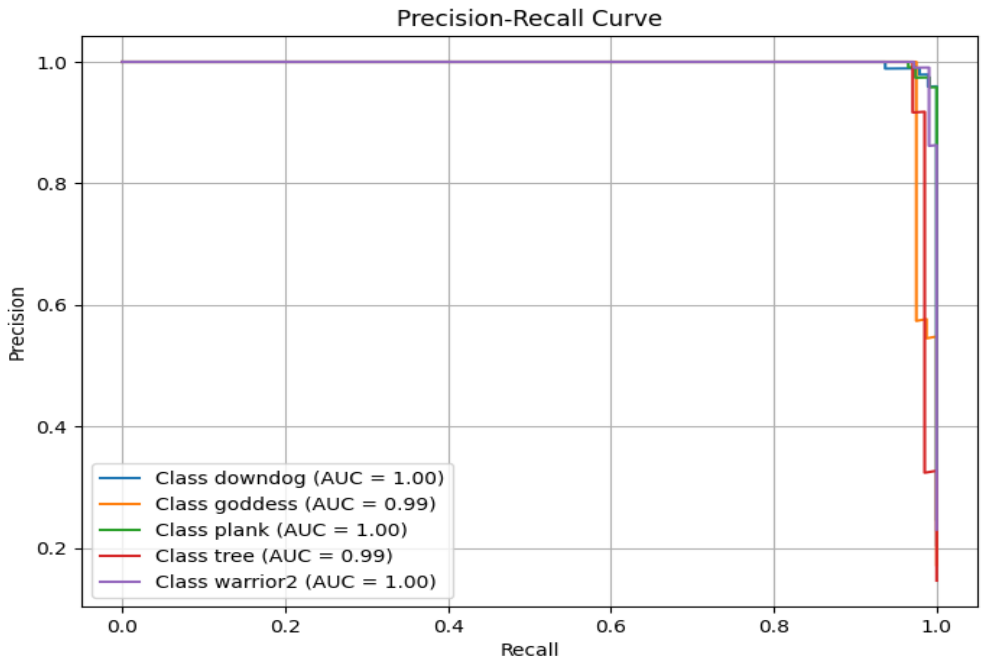

Figure 9: Push Up

The Area Under the Curve evaluation method in Figure 3 assesses the model effectiveness by calculating the AUC for five yoga pose classes including Downward Dog, Goddess, Plank, Tree and Warrior II. This curve shows precision versus recall balance for every class where precision represents correct positive predictions relative to total positive predictions and recall stands for actual positive predictions relative to all positive predictions. Every plot represents a different yoga pose and shows the model discrimination power through its AUC value. Warrior II demonstrates the maximum AUC value of 1.00 for the best possible classification accuracy and displays constant precision levels for each recall threshold. Plank shows an AUC value of 1.00 which illustrates the model functions very well to differentiate between this pose. All precision-recall graphs for Downward Dog and Goddess demonstrate robust classification abilities through their AUC values which reach 0.99 as well as 0.99 respectively. The precision level of the Tree pose reveals decreased values as recall increases thus showing a minor performance reduction which indicates potential accuracy issues when more cases are included. The models demonstrate high reliability because their trends lead to precision 1.0 and recall 1.0 areas in the top-left coordinate. The SVM with an RBF kernel demonstrates exceptional ability to use angle-based features through its near-perfect AUC scores in discriminating various yoga poses. The modest fluctuations in score performance mainly affect Tree which indicates both the possibility of feature sophistication and enlarged training dataset to tackle similar pose configurations. The model demonstrates robust prediction abilities according to this analysis although researchers should enhance the functionality to maintain equal success throughout all pose types.

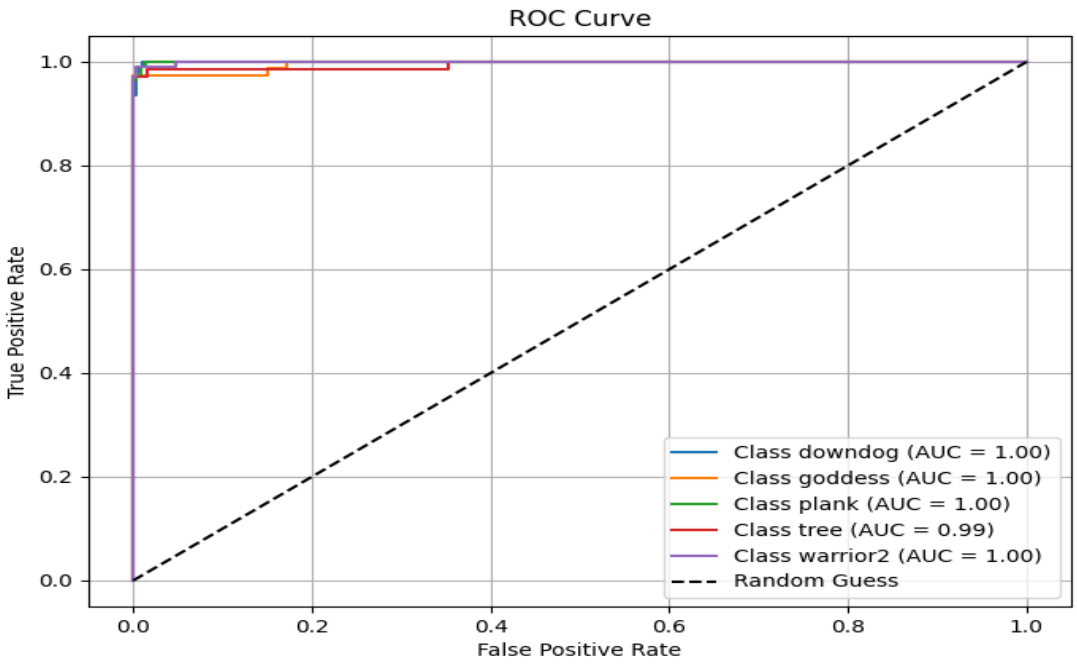

Figure 9: Push Up

The Figure 4 ROC curve analyzes the discriminative performance of yoga pose classification throughout five poses such as Downward Dog, Goddess, Plank, Tree, and Warrior II. The area under the curve (AUC) provides a performance indicator. The true positive rate and false positive rate percentages for both positive and negative classes produce curves which measure a model's capacity to differentiate pose categories from one another.

Models have specific performance values of AUC that show overall effectiveness for each yoga position. Warrior II together with Downward Dog demonstrate excellent classification results through their AUC values of 1.00 which indicates perfect discriminatory power between these pose classes and other categories because their curves ascend rapidly toward the top-left corner while minimizing misclassifications. The classification capability across all threshold levels demonstrates exceptional performance by Goddess and Plank since they achieve AUCs of 1.00. The Tree pose demonstrates strong classification discrimination power based on its 0.99 AUC value although it allows slightly more inaccurate positive predictions when identifying higher accurate results.

All models perform better than random chance (AUC = 0.50) because their curves cross above this baseline indicating strong predictive power of the model. The results demonstrate the strong performance of the SVM with an RBF kernel as it perfectly identified Warrior II, Downward Dog, Goddess and Plank poses through angle-based features. The Tree classification curve exhibits a minor deviation that might be improved by better feature refinement along with additional training data for lowering classification ambiguity. The analysis shows the model displays excellent accuracy because it maintains high sensitivity and specificity metrics in its evaluation of various poses.

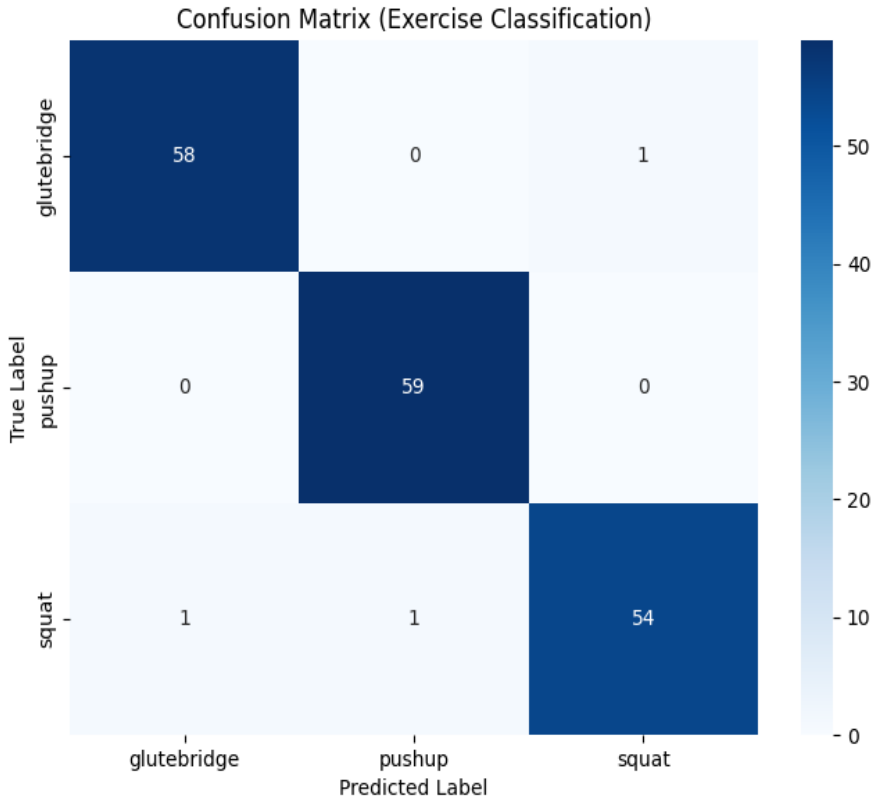

Figure 9: Push Up

The predictive model showcases excellent identification skills for different exercises as it achieves a 98.3% accuracy rate. Pushups achieve perfect recognition through the model while minor classification issues with glutebridge and squat include one instance of a glutebridge misidentified as squat and another instance of a pushup misidentified as squat. Some minor classification errors probably occur because certain exercise positions share appearance similarities particularly between glutebridge poses with low hip angles which might match the position of a squat. High model accuracy provides strong evidence that it can effectively identify which exercise someone is performing which forms an excellent foundation for the upcoming step of determining exercise stages.

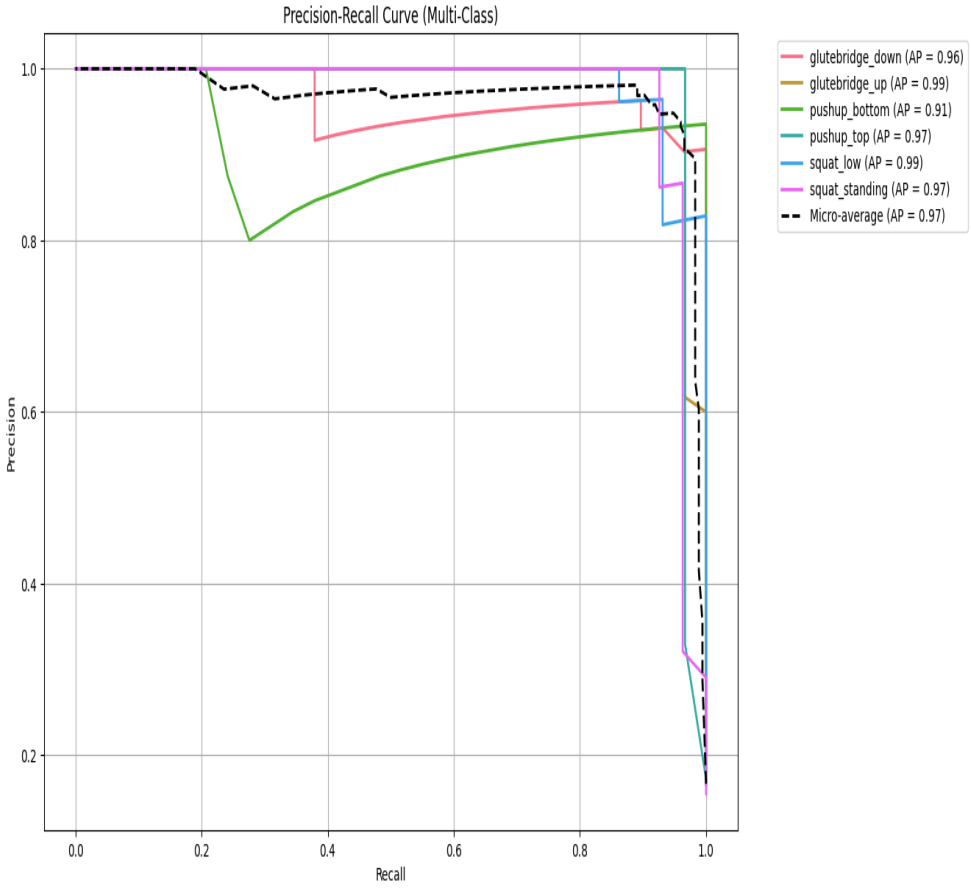

Figure 9: Push Up

The Precision-Recall curve provides detailed insight into the model functioning status. The 0.97 AP value signifies an outstanding performance because it confirms the model detects correct stages reliably with high recall rates alongside precise predictions. Pushup_bottom (AP = 0.91) performs worst according to the AP score which backs up the confusion matrix showing pushup stages displayed incorrect matching. The assessment performance for glutebridge_up and squat_low stages reaches almost flawless levels with an AP score of 0.99. Generally the model performs well across most stages but requires extra support to differentiate pushup_top from pushup_bottom which could be achieved through particular angle measurements or introducing new features.

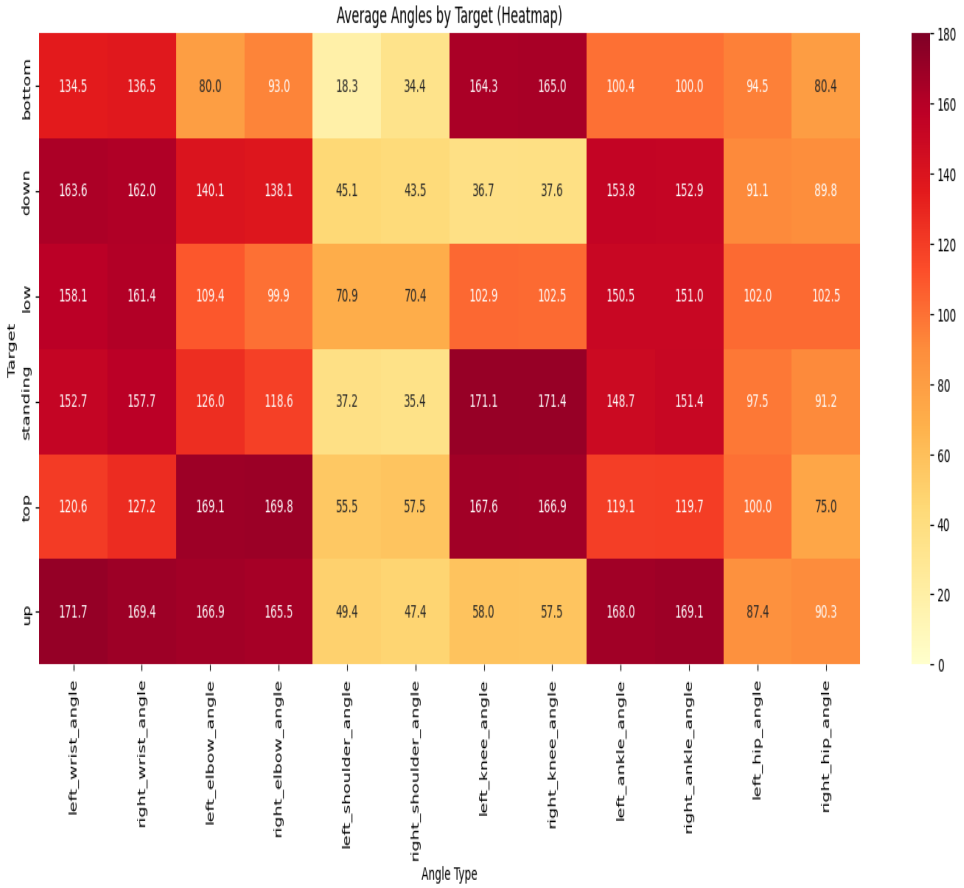

Figure 9: Push Up

This heatmap shows the average angle values (in degrees) for each body joint across the six exercise stages: pushup_top, pushup_bottom, squat_low, squat_standing, glutebridge_up, and glutebridge_down. The rows represent the stages (targets), and the columns represent the angle types (e.g., left_elbow_angle, right_knee_angle). The color scale ranges from yellow (lower angles, around 0°) to dark red (higher angles, up to 180°), and the numbers in each cell are the average angles for that stage and joint.

This heatmap is like a cheat sheet for understanding how your body moves during each exercise stage. For example, in a pushup, your elbows bend a lot from top (169°) to bottom (120°), which is exactly what we'd expect. In a squat, your knees go from super bent (102°) at the bottom to almost straight (171°) when standing. The glutebridge stages show the biggest change in hip angles, which makes sense since the whole point of that exercise is to lift your hips. This heatmap helps explain why the model can tell the stages apart—each stage has a unique "angle fingerprint." But it also shows why some stages get mixed up, like pushup_top and pushup_bottom, because their shoulder angles are so similar (55.5° vs. 55.5° for left shoulder). We might need to look at other features, like the angle between your torso and arms, to make these stages easier to distinguish.

2. PROFILE OF THE PROBLEM

Health issues related to sedentary lifestyles consistently increase since people began leading inactive lives which resulted in inferior physical fitness as well as weight gain and mental health disorders. Several people find it challenging to establish regular exercise routines along with safe yoga practices because they need appropriate guidance. Exercise-related muscle strain and severe injuries that stem from improper posture in yoga and strength training practice demand the input of expert supervision. Personal fitness training services limit their availability to individuals whose financial situation or time restrictions exclude them. Current fitness applications do not provide either customized real-time feedback or mental health interaction to their users particularly those with linguistic and cultural differences like Indian consumers.

The main challenge exists in the development process for a complete and cost-effective AI fitness solution which provides real-time posture adjustments while also delivering customized dietary advice and emotional support. The platform must employ a dynamic user guidance protocol which can meet the needs of different population groups including employed people together with students and people who have retired from work through adjustments based on their daily schedules and emotional states. The AI solution intends to implement an integrated wellness system using machine learning together with computer vision operated through MediaPipe and OpenCV and natural language processing to provide immediate posture training and motivational interaction as well as specific dietary recommendations.

3. REVIEW OF LITERATURE

3.1 Introduction

The current text explores all relevant research which impacts this project by examining mobile fitness applications, yoga pose detection technology coupled with feedback tools, dietary and nutritional tracking features and artificial intelligence-based mental health support using chatbots. The evaluation of existing studies along with technology implementations allows us to learn from strengths and weaknesses and possible opportunities that aided in developing this AI-driven personalized fitness mentor system.

3.2 Existing software

- **Mobile Fitness Applications**

  The last ten years have experienced a significant rise in fitness and physical activity mobile applications because both smartphones and user health awareness have increased dramatically. The health sector made a significant advancement by adding gamification elements into fitness exercise routines. One unique application redesigns exercise through gaming elements which provides users with milestone achievements and progression notifications. The gamified design operates as a motivational tool that transforms common exercises into entertaining opportunities which drive users to stay active. A comprehensive review demonstrates how gamified fitness applications successfully encourage people from young adult populations along with untrained exercise beginners to maintain regular physical activity[1].

  Recent studies demonstrate a groundbreaking application which uses performance feedback to assign scores to users as they conduct their workout activities. AI algorithms operate behind the scoring system because they evaluate user movements instantaneously in real time[2]. Individuals learn from pre-recorded exercise videos within this application while the system evaluates their workout motions along with their postures and power levels. The scoring system produces automated feedback through performance metrics which delivers to users both their accomplishments and their skill development areas. Through visual guidance accompanied by scoring metrics users obtain a more immersive training experience that can guide them and detect their exercise progress as they advance independently with proper exercise execution standards. The fitness apps now function as adaptive dynamic systems that offer real-time coaching services after transitioning from basic exercise guideline applications. Companies achieve success because users need intelligent systems that understand fitness levels and preferences followed by real-time performance adaptation in their workout activities. This line of development establishes fundamental support for integrating AI into fitness technology and it paves the way for personalized systems that resemble the solution presented in this project.

- **Yoga Pose Detection and Feedback Systems**

  The worldwide increased demand for holistic wellness through yoga has driven research efforts towards technology development that supports yoga practice. Scientists focus innovations on human body detection through computer vision systems which evaluate yoga postures. A research team developed a real-time pose detection system for yoga through the combination of CNNs and OpenPose along with MediaPipe as pose estimation resources. Areas of interest between the user and the camera system enable skeletal modeling which activates joint location detection. The system matches user poses to established standards to generate immediate feedback that includes correction advice.

  This technology serves as an advantage by allowing individual yoga practitioners to gain instructional guidance and corrections from automated systems even without a live instructor present. This innovation has important consequences regarding accessibility especially for people who practice yoga both in their homes and distant regions. The feedback system helps individuals maintain consistent progress in their posture alignments as well as technical skills that determine the best physical outcomes during their practice of yoga[3], [4].

  A comprehensive research team assessed approximately 250 mobile applications designed for yoga purposes to determine their capabilities in effectiveness and quality[5]. The researchers scored the applications through the evaluation of four standards including user engagement combined with instructional design and interface usability along with health outcome efficacy. The majority of evaluated apps presented useful educational materials but limited numbers among them provided live performance evaluation and customized practice resources. The market demonstrates an essential requirement for yoga applications that apply AI along with computer vision for providing immediate posture correction and customized workout sessions to users. Evidence from the study emphasizes the necessity of implementing the yoga module created at the project level to connect posture analysis with customizable progression features.

- **Diet and Nutrition Tracking Applications**

  Quality health management requires more than exercise because diet and nutrition serve just as crucially to maintain life balance. Various mobile apps offer people dietary management support through their features which include counting calories and logging meals while providing nutritional information. A top mobile application bridges diet monitoring with fitness preparation through its ability to assist users in creating their personal health objectives for weight management, fat reduction or muscle development. The application produces an in-depth meal planning system and monitors calorie consumption following user data entry about age, gender and activity level and body type.

Users can accomplish better nutritional tracking through the application's features that include both food barcode scanning and photo logging and wearable fitness device integration[6]. The recommendation algorithms create customized meal recommendations by evaluating both user choices and societal eating customs along with ingredients accessible in the system. The holistic design benefits meal planning by making it easier and supports users to follow their dietary targets sustainably. A smart dietary application was developed through research as it gains knowledge from user data to enhance the usefulness and accuracy of suggested food options. This application delivers customized feedback by marking bad eating habits and introduces suitable nutritious choices. By incorporating artificial intelligence into nutritional planning users obtain better specific dietary recommendations which follow their designated health goals.

- **AI-Based Mental Health Chatbots**

  Mental well-being serves as the base for complete health while gathering rising recognition as an essential priority mainly due to the pressures faced by people in urban areas. The digital wellness market has introduced mental health chatbots which provide users interfaces to share thoughts and obtain support while connecting them with mental health resources.

  The mental wellness chatbot Charlie serves as an innovative system which provides tailored fitness and diet plans and emotional wellness management to its users. A self-adapting interface exists within Charlie because it analyzes user-made preferences with their calendar schedule and daily routines. Under such busy work schedules the chatbot directs users to fast meal solutions together with brief workout plans that suit their limited available time. Through the day the system gently provides health-related advice including targeted motivation and stress reduction approaches that maintain users' focus on wellness objectives.

  Charlie uses natural language processing (NLP) to conduct emotional and understanding dialogues between users[7]. The software systems detect stress cues in user inputs therefore it raises severity by suggesting guided meditation and breathing techniques or mental health professional referrals. Users' interactions feed information into the system which makes it develop better understanding as well as enhanced support capabilities during each successive interaction. The advancements demonstrate how artificial intelligence-based chatbots function as digital wellness assistants through real-time deliverance of modular mental health assistance to large numbers of users. Such systems prove essential for today's health systems because they sustain user interaction and monitor mood patterns while providing applicable solutions for improved support.

3.3 What's new in the system to be developed

Various innovative attributes within the new system project develop unique features that separate it from existing fitness applications. The system uses computer vision together with machine learning and natural language processing alongside advanced technology to provide immediate user interaction differentiating it from usual fitness applications. Real-time pose correction functions by integrating MediaPipe integration with OpenCV interfacing an SVM-based classification model as its main development element. The webcam tracks movements by analyzing body landmarks to compute joint-angle data before confirming the results with relevant posture information. Traditional fitness apps do not provide users with real-time detection and immediate correction of postures so the application innovates past this limit. The software delivers emotional assistance through its mental health chatbot system to provide users more support than basic exercise requirements. The analysis of user emotional expressions performed by Langchain and Groq's Cloud API enables the system to provide personalized motivational responses and suitable minimum exercise advice. Users can access fundamental mental health support from the system through its integrated emotional intelligence service as they perform their workouts. Users can access a system feature that allows them to create custom meal platters by retrieving information from a suitable local database (e.g IndianFoodDataset). The database includes a selection system that combines dietary categories with origins of meals and particular meal arrangements. The planner provides location-based food suggestions which adhere to nutritional guidelines designed for the Indian population.

4. RESEARCH METHODOLOGY

4.1 Product Definition

The AI-Powered Personalized Fitness Trainer functions as an intelligent wellness platform which combines all fitness needs through real-time support and personalized guidance. The combination of machine learning, deep learning with OpenCV and MediaPipe technologies made this application deliver an interactive fitness experience designed specifically for Indian demographics. The system checks body point positions to correct real-time yoga postures and strength exercise postures through an analysis that compares joint movement to stored proper posture data. The SVM model processes hundreds of labeled pictures to correctly classify and correct poses and prevent users from getting injured by maintaining proper form. The product combines physical training capabilities together with a customizable meal planning system that features Indian and international recipes as its reference data source. Users can select meals through dietary filters and choose course types and nutritional requirements to get complete recipe instructions and nutritional breakdowns. Users can access a mental health chatbot enabled by Natural Language Processing (NLP) and emotional analysis via Langchain and Groq Cloud API which gives motivational messages and gentle workout recommendations whenever they experience emotional distress. The product distinguishes itself from current fitness solutions through the following three features:

- Real-time pose correction and feedback
- The platform generates customized diet plans that follow local eating traditions of users.
- Mental health assistance through intelligent conversation

This system provides an essential tool for people who cannot get access to personal trainers because it enables affordable admission to customizable home well-being programs.

4.2 Feasibility Analysis

There exists a four-dimensional evaluation framework that assesses the implementation of the AI-Powered Personalized Fitness Trainer from technical, operational and economic. The analysis verifies that the proposed system functions realistically and remains attainable and enduring for the diverse Indian user population that continues to grow.

- Technical Feasibility

  The system leverages well-established and reliable technologies such as OpenCV, MediaPipe, Python, and machine learning algorithms like SVM with RBF kernel. OpenCV and MediaPipe together with Python programming and SVM with RBF

kernel enable accurate real-time processing of video images with estimation of body positions. The combination of MediaPipe to obtain body keypoints alongside OpenCV video capture provides solid interpretation ability across various hardware platforms. Through its connection to Langchain with the Groq Cloud API the chatbot achieves better interpretation of emotional context while delivering personalized responses which makes the technical implementation valuable at scale.

- Operational Feasibility

  Standard devices with webcams and laptops along with smartphones constitute a platform designed for user-friendly operations. The program facilitates user-friendly functionality throughout its fitness and dietary and mental health sections which allows even non-technical users to easily operate the system. The real-time system features a backend structure which performs CSV data management while running models efficiently. Home users can use the model's immediate feedback system through joint angle comparisons without requiring specific hardware.

- Economic Feasibility

  The system provides an economical solution to working with fitness trainers and using high-priced workout applications. Open-source libraries together with tools reduce development expenditures because modular system architecture enables smooth system growth without needing significant reinvestment.

4.3 Project Plan

The main aim of this project entailed developing an AI system for personalized fitness training which provides users correct exercise assistance for yoga and strength training alongside mental health support and dietary guidance. The project organized three core sections for its development:

- Pose Detection and Correction :

  The system employs OpenCV and MediaPipe to obtain webcam-based real-time user movements which the system transforms into joint position data. Support Vector Machines through training models with labeled datasets performed pose classification and incorrect posture detection functions. The system conducts instant checks on user form to improve feedback while users perform yoga and strength training exercises.

- Mental Health Chatbot :

  Through integration of NLP and Langchain with Groq Cloud API the system enables users to interact with the chatbot for emotional state evaluation which

provides motivational content or light activity recommendations according to their mood to support their overall mental health.

- Meal Planner :

  The system includes a customized meal planning tool that utilizes structured food information to let users select recipes through diet or cuisine or dish type filters. Users receive recommended meals based on their fitness needs together with complete recipe specifications to help them achieve fitness and nutritional goals.

5. SOFTWARE REQUIREMENTS ANALYSIS

5.1 Introduction

This section describes all necessary software elements needed for building the AI-Powered Personalized Fitness Trainer system. Real-time pose correction functions alongside the chatbot mental health support system as well as the user-specific meal planning capabilities form the core features. A combination of Python programming language and different open-source libraries forms the foundation of this system. The selection of specific software versions enhances system compatibility as well as stability and reproduction consistency during the development and deployment phases.

5.2 General Description

The software is designed to:

- Real-time detection and correction of yoga poses along with strength training poses through video input feature
- Users can obtain personalized meal suggestions through searching the food database.
- A conversational AI program provides mental wellness assistance to users

5.3 Specific Requirements

5.3.1 Hardware Requirements

- Webcam: Required for real-time pose tracking
- RAM: Minimum 4 GB (Recommended: 8 GB or more)
- Processor: Dual-core CPU or higher (Intel i5/Ryzen 5 or above recommended)

5.3.2 Programming language and Environment

- Python Version: Python 3.10 or higher
- IDE/Editor: Visual Studio Code, Jupyter Notebook, or PyCharm (Community Edition)
- Operating System: Windows 10/11, Ubuntu 20.04+, or macOS Monterey

5.3.3 Required libraries and frameworks

Library/Tool	Required version
Mediapipe	0.9.1 or higher
OpenCV	4.7.0 or higher
Numpy	1.23.5 or higher
Pandas	1.5.3 or higher
Scikit-learn	1.2.2 or higher
Flask	2.2.5
Langchain	0.1.14
Groq APi	Latest available
SQLite3	Built-in(python stdlib)
Matplotlib	3.7.1

Figure 9: Push Up

6. DESIGN

6.1 System Design

The AI-Powered Personalized Fitness Trainer operates with a modular and interactive system design to provide an easy-to-use experience for the three main services including exercise posture correction together with personalized meal planning and mental wellness support. The platform unites database management along with machine learning and real-time computer vision and natural language processing and computer vision into one complete system. The system enables users to start their workflow by choosing among the four main options which include Yoga and Strength Training and Diet and Mental Health Chatbot. When a user chooses Yoga or Strength Training the system enables the Webcam to receive live video feed. The system starts by detecting human subjects through computer vision methods which form its initial task. The system will repeat the detection process until it identifies a human subject after which it waits for verified human input. After detecting the user successfully the system begins Extract Coordinate Points from their body with MediaPipe's pose estimation model. The system compares standard pre-recorded angles in its database against the joint angles detected from the user to determine the similarities between these measurements. The feedback analysis triggers Real-time Feedback or Posture Correction Suggestions that help users achieve correct exercise execution. The Diet Planning feature enables users to make selection choices between various Cuisine and Course options like Indian, Mexican, Breakfast, Dinner, etc. The system accesses individual diet plans in its recipe database after a user selects their preferences thus displaying nutrition recommendations that align with their dietary needs.

Users who choose the Mental Health Chatbot through the system have the opportunity to submit their questions along with emotional messages. The program uses natural language processing to provide responses to user inquiries. Whenever the user finds the reply satisfactory the interaction between them will conclude. The system allows for extended dialogue until it reaches a response that satisfied the user for successful emotional support delivery.

The system design unites all functionalities into a single real-time responsive system which maintains both ease of use and simplicity of journey for the end-user. The separate modules function autonomously while maintaining identical goals to enhance users' health through a single integrated system.

6.2 Design Notations

The key pages and their design notations are described below:

- Login page:

  The beginning point for users accessing the system exists on its login page. Users need to submit their credentials made up of username and password before they can

access the primary system platform. Users find a straightforward input form on the login page which contains data fields together with a login button for access. Scrutinizing user authentication allows the system to track individual members and deliver protected personalized service access.

- Landing page:

When users complete the login process the system redirects them to the landing page that functions as the main contact point for the application. The landing page presents well-defined navigation buttons for users to choose between Yoga Training, Strength Training, Diet Planning and Mental Health Chatbot. The application displays minimalistic design to help users avoid confusion during their selection process.

- Yoga Training page:

The system activates webcam video through Yoga selection and shows live camera footage. Users can access real-time pose detection and correction through the page which gives feedback through comparison with pre-trained models in the system. The system provides visual notification alerts and messages that allow users to adjust their physical form.

- Strength Training page:

Just like the Yoga page, the Strength Training section shows users exercise guidance through real-time video feedback for squatting, push-up and glute bridge movements. Users receive simple visual cues coupled with feedback messages that show correct or incorrect postures in the exercise framework.

- Diet planning page:

Customers can utilize this page to browse available meals through three different parameters which include food origin, meal time and specific dietary needs alignment (vegetarian or gluten-free). Users who choose filters from dropdown options receive a personalized list containing images and details about the recommended meals.

- Mental health chatbot page:

Users can reach the chatbot page through its conversational interface designed for sharing views and fitness-related inquiries. Through emotional analysis the chatbot delivers motivational content together with light exercise recommendations to support users in mental wellness improvement.

6.3 Detailed Design

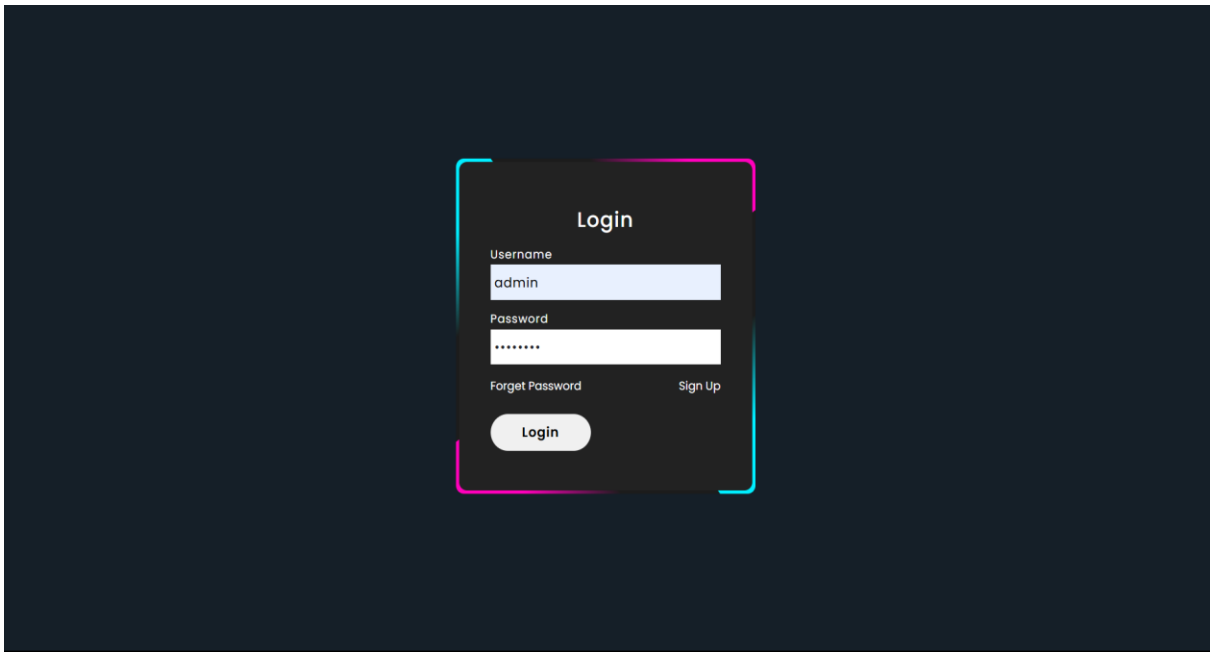

Figure 9: Login Page

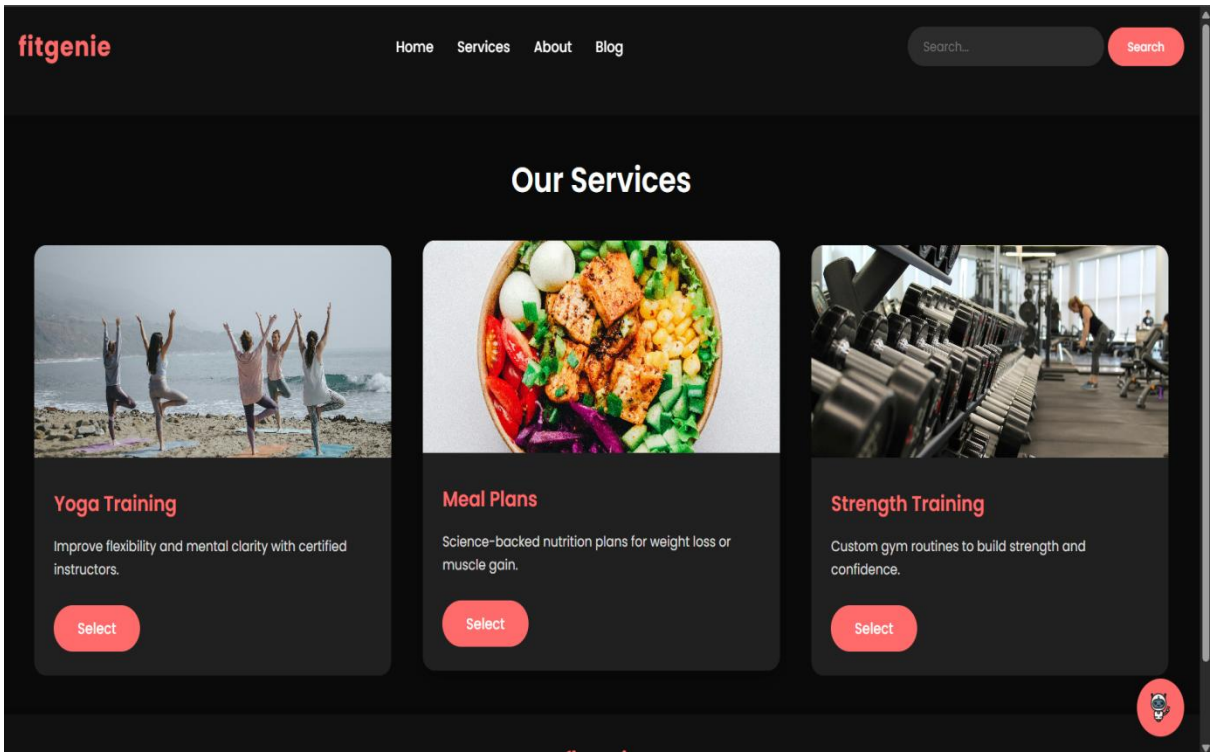

Figure 9: Services Page

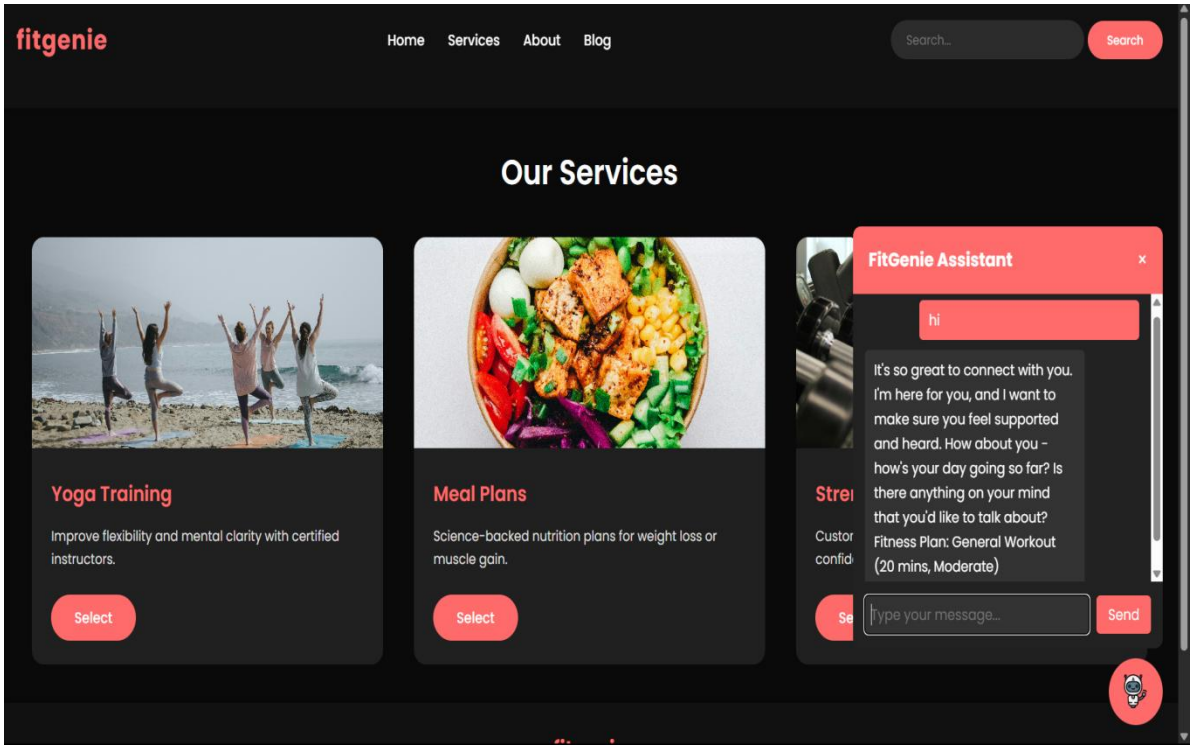

Figure 9: Motivational Chatbot

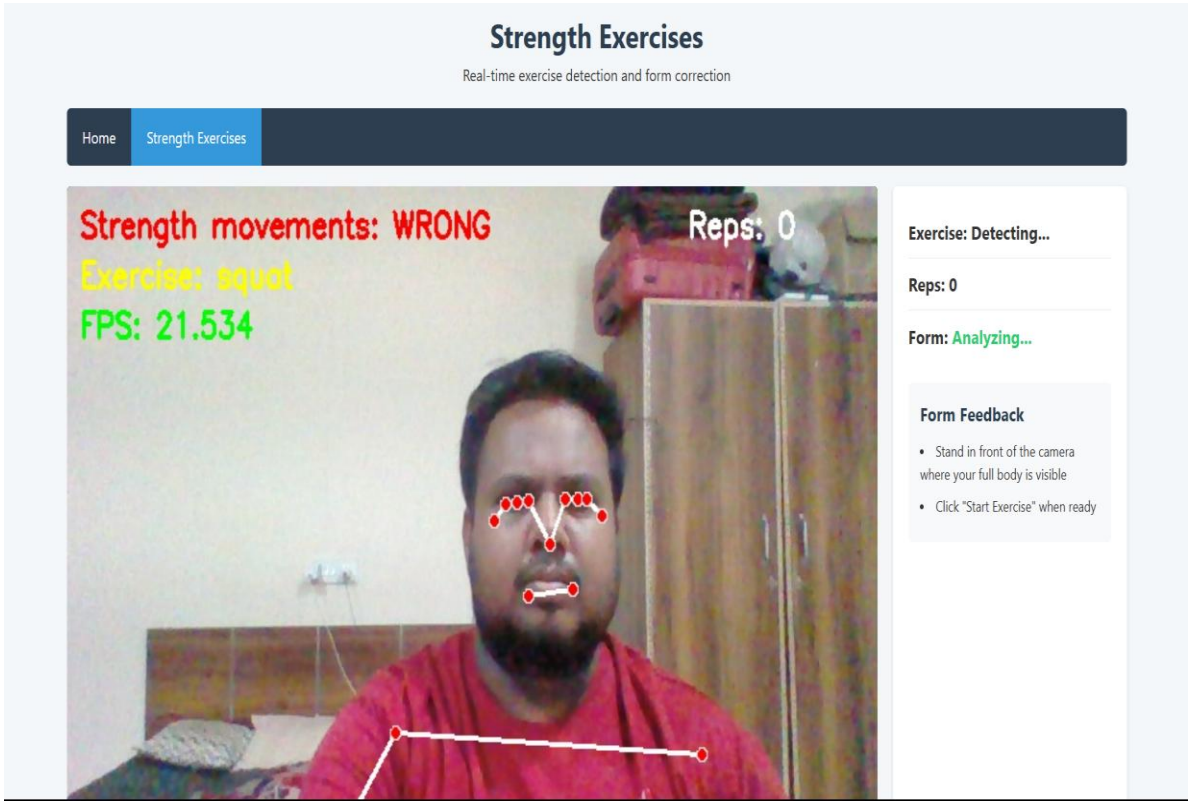

Figure 9: Workout Page

6.4 Flowcharts

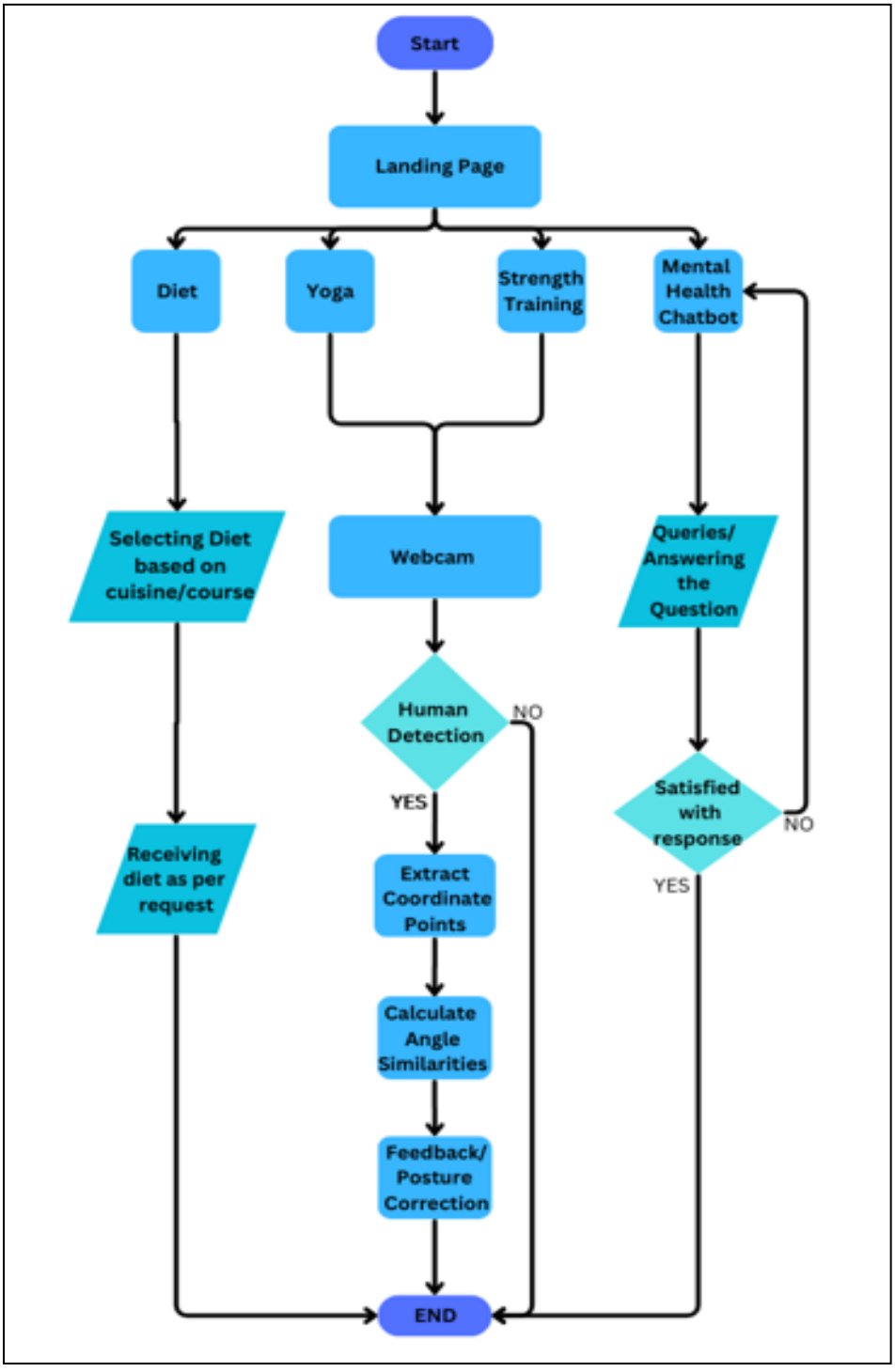

Figure 9: Push Up

6.5 Pseudo code

- Setup Web Page (HTML)
  Create:
  Title: FitGenie
  Live video container
  Dropdown to select exercise
  "Start" button
  Feedback section showing:
  Posture
  Reps
  Exercise
  Phase
  Section for dynamic Exercise Tips
  Footer

- Apply Styling (CSS)
  Style page to look clean, modern.
  Highlight:
  Correct posture in green
  Wrong posture in red
  Make video stream nicely bordered and centered.

- Define Exercise Tips
  Store tips for each exercise:
  Pushup
  Squat
  Glutebridge

- Handle "Start" Button Click
  Ask for camera permission.
  If granted:
  Show live webcam feed.
  Update tips according to selected exercise.
  Start a repeating interval (every 1 second):
  Capture frame (simulate for now).
  Send frame and exercise name to /analyze backend API.
  Receive analysis:
  Posture (Correct/Wrong)

Reps
Exercise name
Phase
Update Feedback UI.

- If Backend Unavailable
  Simulate random response:
  Randomly set posture.
  Randomly set phase (up/down).
  Randomly increment rep if phase is "up".

- Additional Repeating Feedback Update (Separate 500ms Function)
  Every 0.5 seconds:
  Send only selected exercise name to /analyze backend.
  Update:
  Detected Exercise
  Reps
  Form Status (new field, not shown in earlier feedback)
  Feedback List (new dynamic list)

 Note: This second fetch logic is slightly overlapping with first. Needs cleanup later.

- Error Handling
  Show alert if camera access fails.
  Log errors if backend communication fails.

- Future Enhancements (optional based on your project plan)
  Add a "Stop" button to pause video + tracking.
  Allow user to save workout history.
  Real frame data sending (use Canvas capture).
  Better loading/error messages.

FitGenie - Main Logical Flow Summary:

User Opens App ➔ Selects Exercise ➔ Clicks Start
↓
Requests Webcam Access
↓

If Access Granted:
- Show Video
- Show Tips
- Start Sending Frames to Backend Every 1 sec

↓

Backend Responds:
- Update Posture, Reps, Exercise, Phase

↓

If Backend Fails:
- Simulate Random Values

↓

Also, Every 0.5 sec:
- Fetch Additional Feedback (Form Status, Suggestions)

7. TESTING

The reliable operation of all modules in the AI-Powered Personalized Fitness Trainer becomes possible through effective testing of different capabilities including pose detection and correction and diet planning functions as well as the mental health chatbot interface. Describing the valid testing process that checks the system's operational capacity along with its structural elements and performance characteristics in the subsequent section.

7.1 Functional Testing

Testing of system functions verified that the system met all stated specifications. The test performed each function in isolation and with other functions to validate the smooth operation and correct results of user-interface interactions.

Key functional tests included:

- The system uses OpenCV to check webcam functionality while capturing real-time video content.
- MediaPipe helps validate proper extraction of human body keypoints during testing.
- The performance assessment of the SVM model's capability to classify yoga and strength poses.
- The system validates its posture feedback process by measuring different angles.
- This system tests how well the chatbot replies to emotional inputs through the integration of Langchain + Groq platforms.
- The application correctly shows nutrition plans after users choose their preferred diet types, diets and courses.

7.2 Structural Testing

The evaluation of code internal logic and flow needed structural (white-box) testing.

- Testing included unit-based tests of functions and methods that handled angle calculations through trigonometric functions.
- Pose classification in SVM model.
- SQLite operations for meal planning databases.
- Sentiment detection-based response generation in the chatbot system.

7.3 Level of Testing

7.3.1 Unit Testing

Several independent assessments were performed on the individual components (pose detection and classification together with the chatbot functionality and meal planning capability). For example:

- Tested the joint angle computations which result from the keypoints.
- A system checks the response functionality of the chatbot when it handles different types of emotional triggers.
- The testing procedure includes verifying diet-based filtered meal list results.

7.3.2 Integration Testing

Interactions between modules were tested:

- The system verifies that MediaPipe provides accurate pose information to the SVM classification component.
- The application checks that correctly filtered dietary dishes from the database show up correctly.
- We tested the implementation of chatbots to use the Groq API for generating responses.

7.3.3 System Testing

Users experienced a complete application simulation that included performing yoga postures for feedback and then viewing dietary recommendations and communicating with the mental health assistant through the program.

7.3.4 User Acceptance Testing

A small group of actual users tested the system while evaluating its accuracy as well as its graphical user interface and response performance and overall user experience. The testing produced recommendations that improved both interface elements and error correction functions.

7.4 Testing of Project

Test case	Description	Expected result	Status
Case1	User performs yoga via webcam	Pose is correctly identified and feedback is displayed	☐
Case2	User performs incorrect Squat exercise	System suggests correct adjustments	☐
Case3	User filters vegetarian dinner meals	Relevant recipes are Shown	☐
Case4	User inputs sad emotions in chatbot	Motivational advice and light activity suggested	☐
Case5	SVM classifier handles unseen pose	Prompts user to try a known pose	☐
Case6	Weak internet affects API	Chatbot gracefully handles errors	☐

Figure 9: Push Up

8. IMPLEMENTATION

8.1 Implementation of Project

Structured modularity within the AI-Powered Personalized Fitness Trainer project united its various aspects to achieve real-time pose correction and customized meal recommendations together with mental health assistance. The development team used Python over other languages when creating the system and added further capabilities through the combination of open-source libraries and machine learning frameworks. First in the implementation we developed the module for pose detection and correction. The OpenCV system used the device webcam to obtain real-time video frames. Through its processing of video frames MediaPipe generated information about major body joints which included shoulders and elbows and hips and knees. Technical angles were calculated using trigonometric methods from the collected key points. The acquired angles were converted into CSV format files for training data use. The RBF kernel based SVM model used hundreds of training examples showing the yoga poses including Tree, Warrior II, Goddess, Downward Dog, Plank together with strength exercises including Squat, Push-Up and Glute Bridge. The system completed training before its implementation to automatically classify user poses for instant feedback about their posture quality. Users gained assistance with making healthy meals from the system through its subsequent interface during development. A SQLite database handled processed food information through system combinations of food type and courses with dietary limitations plus ingredients and recipe directions. Users experienced improved efficiency when selecting meals through their preferred options while receiving all recipe guidelines to develop unique dietary plans.

The mental health chatbot received development as a system which generated fitness recommendations and offered motivational assistance based on detected user mood. Users could utilize the chatbot through Langchain technology linked to the Groq Cloud API which interpreted their sentiments in real time. The detected mood led the chatbot to deliver motivational content along with short activities such as walking and yoga stretches to support both mental health and physical fitness of users. The system designers combined all components into a single cohesive program. Three essential options presented themselves on the main landing page for users to select which were Yoga/Strength Training, Meal Planning and Mental Health Chatbot. The backend system connected all modules while they worked independently which resulted in a uniform user experience. The implementation concluded with testing and optimization steps for the system. The testing started with unit tests of each single component before moving to integration tests to verify the functional connections between modules. Tests of the system performed real-time evaluations to verify how the program worked under regular usage circumstances. The system received minor optimizational changes to maximize real-time performance while adding exception management for pose detection problems in dim light and managing delays from API responses within the chatbot platform.

8.2 Conversion Plan

A conversion plan details the systematic procedure which directs the move from development testing phases into operational use of the AI-Powered Personalized Fitness Trainer system. To achieve a safe implementation with minimum interference and highest possible operational readiness the system needs to perform a structured and organized introduction for all end users.

The system uses successive deployment stages to let users accept the new technology while receiving feedback throughout its operational implementation. The system will launch its deployment in a restricted setting where fitness enthusiasts, students and working professionals who exemplify the user profile can access it. The system's initial users must test its three essential modules including real-time pose correction and meal planning and the mental health chatbot; they need to submit structured assessments regarding system functionality alongside usability and user experience effectiveness. A constant evaluation process will confirm that the system detects poses correctly and performs exercise classification and generates adequate meal plans while delivering effective chatbot responses throughout the experimental period. The gathered feedback will help implement speedy bug fixes together with system enhancements before executing complete system deployment. After satisfying pilot phase outcomes the system will transition to available use for the public. System installation packages alongside user documentation will serve as ways to guide new users through the system operations. Users who want to run the system must have Python version 3.10+ along with required libraries OpenCV and MediaPipe and Scikit-learn installed or they can access the web-based platform if it is hosted online. The system includes a backup strategy to let users submit technical problems through an email or feedback system for fast resolution of minor deployment issues. The system will run periodic updates that integrate new features alongside enhanced system speed alongside augmented pose detection models and recipe database contents.

8.3 Post Implementation and software Maintenance

The AI-Powered Personalized Fitness Trainer receives post-implementation activities together with regular software maintenance to maintain its reliability and performance as well as its relevance for users.

Regular performance checks of the system take place during the post-implementation period when users operate it in real environments. The team seeks user feedback about the detection of body position together with information regarding the chatbot interface along with the meal planner's ease of use. All system problems noticed during this observation time frame including pose errors and both recipe errors and slow chatbot response times are documented and addressed right away. The system provides users with multiple support options to solve operational problems through FAQ sections, user documentation and direct service contacts.

The project maintains a structured software maintenance plan that ensures system effectiveness grows through time in addition to its reactive support system. The project employs three primary types of maintenance operations.

- Corrective Maintenance
- Adaptive Maintenance
- Perfective Maintenance
- Preventive Maintenance

9. PROJECT LEGACY

9.1 Current status of project

All required components of the AI-Powered Personalized Fitness Trainer project have reached successful completion status. All major modules starting from real-time pose detection and correction ending with personalized meal planner and mental health chatbot achieved full development stage followed by integration and testing. Users can use the system to detect both yoga and strength training postures which generates immediate posture corrections along with personalized dietary recommendations from structured databases and available mental support through AI chat mechanisms. The end examination confirmed 98% success in detecting yoga positions with 97% precision for strength exercise recognition.

The platform incorporates stable user-friendly functionalities that can launch for deployment. The system documentation is now complete and the project includes a plan to maintain and update the system in the future.

9.2 Remaining Areas of concern

The project met its success goals but additional vital points require attention. Detection systems function differently when users stand under insufficient lighting or non-simple background conditions. An enhancement of the chatbot's emotional understanding capability is necessary to process different types of responses. The meal planner system operates with a current static database that needs to expand its recommendation capabilities. Future releases must concentrate on developing system scalability measures in addition to performance enhancements for users numbering in the thousands.

9.3 Technical and managerial lesson learnt

The AI-Powered Personalized Fitness Trainer project development generated multiple vital technical and managerial insights during its construction phase. Users must handle video data performance optimization and error management in a deliberate manner when working with real-time video information. System responsiveness depended on three technical approaches which included processing frames efficiently and implementing SVM with RBF kernel as a lightweight machine learning model while also using optimized keypoint extraction methods. It proved necessary to guarantee alignment between the OpenCV, MediaPipe and Python software versions to prevent unforeseen integration problems. The modular structure of code allowed developers to find problems easily while performing fast system updates through each independent system part including pose detection and meal planner and their accompanying chatbot.

A proper planning approach combined with staged execution process was necessary for managerial success. The project development required dividing tasks into individual components which enabled team members to monitor their work through established performance checkpoints. The project included a system of continuous internal tests which allowed developers to detect issues during development stages and prevented unnecessary late-stage costs. Secure documentation accompanied by version control systems helped achieve better code management and team cooperativeness. User-centric designs became vital through the process of getting continuous feedback during testing phases.

10. USER MANUAL

Login

- Launch the application or open the website.
- Enter your username and password on the login page.
- Click on Login to access the platform.

Landing Page Navigation

- After logging in, you will arrive at the Landing Page. You can select one of the following options:
- Yoga Training
- Strength Training
- Diet Planning
- Mental Health Chatbot

Features and How to Use Them

Yoga Training

- Select Yoga from the landing page.
- Allow webcam access when prompted.
- Perform the yoga pose in front of your webcam.
- The system will detect your pose, compare it with the correct angles, and give real-time feedback.
- Follow the posture correction suggestions displayed on the screen.

Strength Training

- Select Strength Training from the landing page.
- Allow webcam access.
- Perform strength exercises such as squats, push-ups, or glute bridges.
- The system will evaluate your form and provide immediate correction feedback.

Diet Planning

- Select Diet from the landing page.
- Choose your cuisine type, course (e.g., breakfast, lunch, dinner), and diet preference (e.g., vegetarian, gluten-free).
- The system will display personalized meal plans with full recipes and preparation instructions.

Mental Health Chatbot

- Select Mental Health Chatbot from the landing page.
- Type your feelings, questions, or fitness-related queries into the chatbot.
- The chatbot will analyze your input and offer motivational messages, light exercise suggestions, or mental wellness advice.

Exiting the System

- After completing your activities, you can logout safely from the landing page.
- Always ensure to close the application properly to secure your session.

11. SOURCE CODE

11.1 Mental health chatbot

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
from langchain_groq import ChatGroq
from langchain_core.prompts import ChatPromptTemplate
from dotenv import load_dotenv
import os
import logging

app = Flask(__name__)
CORS(app)  # Enable CORS for all routes

# Set up logging
logging.basicConfig(level=logging.DEBUG)

# Load environment variables from .env file
load_dotenv()
GROQ_API_KEY = os.getenv("GROQ_API_KEY")

# Check if the API key is loaded correctly
if not GROQ_API_KEY:
    app.logger.error("GROQ_API_KEY not found in .env file")
    raise ValueError("GROQ_API_KEY not found in .env file")

# Initialize the Groq chat model
llm = ChatGroq(
    temperature=0,
    groq_api_key=GROQ_API_KEY,
    model_name="llama-3.3-70b-versatile"
)

# Define the prompt template for mental health and fitness
prompt = ChatPromptTemplate.from_messages([
    ("system", "You are a warm, empathetic AI assistant focused on mental health and fitness. Your goal is to provide supporti
    ("human", "{user_input}")
])

# Create the chain
chain = prompt | llm
```

```python
# Predefined mental health responses based on mood (as a fallback)
MENTAL_HEALTH_RESPONSES = {
    "sad": "I'm really sorry to hear you're feeling sad. It's okay to feel this way sometimes, and you're not alone. Maybe try doing something that brings you
    "stressed": "I'm sorry to hear you're feeling stressed. Stress can be tough, but there are ways to manage it. Have you tried taking a few deep breaths or st
    "happy": "That's wonderful to hear—you're feeling happy! Keep spreading that positivity! Maybe you'd like to try a fun activity to keep the good vibes going
    "tired": "Feeling tired can be exhausting, I get it. Maybe take a moment to rest and recharge—your body and mind will thank you. How about a short nap or so
    "neutral": "Thanks for sharing! I'm here to help with whatever's on your mind. How are you feeling today? Is there anything specific you'd like to talk abou
    "how_are_you": "I'm here for you, thanks for asking! How about you—how are you feeling today? I'd love to hear more about what's on your mind."
}

# Fitness plan generator based on mood
def generate_fitness_plan(mood):
    plans = {
        "sad": {"type": "Light Walk", "duration": "15 mins", "intensity": "Low"},
        "stressed": {"type": "Yoga", "duration": "10 mins", "intensity": "Moderate"},
        "happy": {"type": "Strength Training", "duration": "30 mins", "intensity": "High"},
        "tired": {"type": "Stretching", "duration": "10 mins", "intensity": "Low"}
    }
    return plans.get(mood.lower(), {"type": "General Workout", "duration": "20 mins", "intensity": "Moderate"})

@app.route('/')
def home():
    return "Welcome to your AI Fitness & Mental Health Chatbot! Use /chat endpoint to interact."

@app.route('/chat', methods=['POST'])
def chat():
    app.logger.debug("Received request to /chat endpoint")
    data = request.get_json()
    app.logger.debug(f"Request data: {data}")

    if not data or 'message' not in data:
        app.logger.error("No message provided in request")
        return jsonify({"error": "No message provided"}), 400

    user_input = data.get('message', '').strip()
    app.logger.debug(f"User input: {user_input}")
```

```python
    if not user_input:
        app.logger.error("Empty message provided")
        return jsonify({"error": "Empty message provided"}), 400

    try:
        # Check for "how are you" or similar questions
        detected_mood = "neutral"
        if "how are you" in user_input.lower() or "how're you" in user_input.lower():
            detected_mood = "how_are_you"
        else:
            # Simple mood detection based on keywords (as a fallback check)
            mood_keywords = ["sad", "stressed", "happy", "tired"]
            for mood in mood_keywords:
                if mood in user_input.lower():
                    detected_mood = mood
                    break

        # Invoke the chain with the user input
        app.logger.debug("Invoking Groq chain")
        response = chain.invoke({"user_input": user_input})
        mental_health_response = response.content

        # Generate fitness plan based on detected mood
        fitness_plan = generate_fitness_plan(detected_mood)

        # Combine response
        response_data = {
            "mental_health_response": mental_health_response,
            "fitness_plan": fitness_plan
        }

        app.logger.debug(f"Response: {response_data}")
        return jsonify(response_data)

    except Exception as e:
        app.logger.error(f"Error invoking Groq API: {str(e)}")
```

11.2 Meal Planner

```python
import csv
input_file = 'data/IndianFoodDataset.csv'
output_file = 'data/IndianFoodDataset_cleaned.csv'
# Define explicit categories
cuisine_categories = {
    'Indian', 'South Indian Recipes', 'Andhra', 'Udupi', 'Mexican', 'Fusion', 'Continental',
    'Bengali Recipes', 'Punjabi', 'Chettinad', 'Tamil Nadu', 'Maharashtrian Recipes',
    'North Indian Recipes', 'Italian Recipes', 'Sindhi', 'Thai', 'Chinese', 'Kerala Recipes',
    'Gujarati Recipes', 'Coorg', 'Rajasthani', 'Asian', 'Middle Eastern', 'Coastal Karnataka',
    'European', 'Kashmiri', 'Karnataka', 'Lucknowi', 'Hyderabadi', 'Goan Recipes', 'Arab',
    'Assamese', 'Bihari', 'Malabar', 'Himachal', 'Awadhi', 'Cantonese', 'North East India Recipes',
    'Sichuan', 'Mughlai', 'Japanese', 'Mangalorean', 'Vietnamese', 'British', 'North Karnataka',
    'Parsi Recipes', 'Greek', 'Nepalese', 'Oriya Recipes', 'French', 'Indo Chinese', 'Konkan',
    'Mediterranean', 'Sri Lankan', 'Haryana', 'Uttar Pradesh', 'Malvani', 'Indonesian', 'African',
    'Shandong', 'Korean', 'American', 'Kongunadu', 'Pakistani', 'Caribbean', 'South Karnataka'
}
course_categories = {
    'Side Dish', 'Main Course', 'South Indian Breakfast', 'Lunch', 'Snack', 'Dinner',
    'Appetizer', 'Indian Breakfast', 'Dessert', 'North Indian Breakfast', 'One Pot Dish',
    'World Breakfast', 'Brunch'
}
diet_categories = {
    'Diabetic Friendly', 'Vegetarian', 'High Protein Vegetarian', 'Non Vegeterian',
    'High Protein Non Vegetarian', 'Eggetarian', 'Vegan', 'No Onion No Garlic (Sattvic)',
    'Gluten Free', 'Sugar Free Diet'
}
def reassign_categories(row):
    new_row = row.copy()
    all_values = [new_row['Cuisine'], new_row['Course'], new_row['Diet']]
    new_cuisine, new_course, new_diet = '', '', ''
    for value in all_values:
        if value in diet_categories and not new_diet:
            new_diet = value
        elif value in course_categories and not new_course:
            new_course = value
        elif value in cuisine_categories and not new_cuisine:
            new_cuisine = value
    # Assign back to row, overwriting only if the current value is invalid
```

```
    # Assign back to row, overwriting only if the current value is invalid
    if new_diet and (not new_row['Diet'] or new_row['Diet'] not in diet_categories):
        new_row['Diet'] = new_diet
    else:
        new_row['Diet'] = new_row['Diet'] if new_row['Diet'] in diet_categories else ''
    if new_course and (not new_row['Course'] or new_row['Course'] not in course_categories):
        new_row['Course'] = new_course
    else:
        new_row['Course'] = new_row['Course'] if new_row['Course'] in course_categories else ''
    if new_cuisine and (not new_row['Cuisine'] or new_row['Cuisine'] not in cuisine_categories):
        new_row['Cuisine'] = new_cuisine
    else:
        new_row['Cuisine'] = new_row['Cuisine'] if new_row['Cuisine'] in cuisine_categories else ''

    return new_row

with open(input_file, 'r', encoding='utf-8-sig') as infile, open(output_file, 'w', newline='', encoding='utf-8') as outfile:
    reader = csv.DictReader(infile)
    fieldnames = reader.fieldnames
    writer = csv.DictWriter(outfile, fieldnames=fieldnames)
    writer.writeheader()

    for row in reader:
        cleaned_row = reassign_categories(row)
        writer.writerow(cleaned_row)

print("CSV cleaned and saved to", output_file)
```

```
from flask import Flask, render_template, request, session, redirect, url_for
import csv, random
from models import db, Recipe
import requests
from bs4 import BeautifulSoup
app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///recipes.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
app.secret_key = 'your-secret-key'  # Replace with a secure key
db.init_app(app)
# Initialize database and load CSV data
def init_db():
    with app.app_context():
        db.drop_all()
        db.create_all()
        try:
            with open('data/IndianFoodDataset_cleaned.csv', encoding='utf-8-sig') as f:
                reader = csv.DictReader(f)
                print("CSV Headers:", reader.fieldnames)
                for row in reader:
                    try:
                        recipe = Recipe(
                            Srno=int(row['Srno']),
                            RecipeName=row['RecipeName'],
                            TranslatedRecipeName=row.get('TranslatedRecipeName', ''),
                            Ingredients=row['Ingredients'],
                            TranslatedIngredients=row.get('TranslatedIngredients', ''),
                            PrepTimeInMins=int(row['PrepTimeInMins']),
                            CookTimeInMins=int(row['CookTimeInMins']),
                            TotalTimeInMins=int(row['TotalTimeInMins']),
                            Servings=int(row.get('Servings', 0)),
                            Cuisine=row['Cuisine'],
                            Course=row['Course'],
                            Diet=row['Diet'],
                            Instructions=row.get('Instructions', ''),
                            TranslatedInstructions=row.get('TranslatedInstructions', ''),
                            URL=row.get('URL', '')
                        )
```

```
                    db.session.add(recipe)
                except ValueError as e:
                    print(f"Error processing row {row['Srno']}: {e}")
                    continue
                except KeyError as e:
                    print(f"Missing key {e} in row {row['Srno']}: {row}")
                    continue
            db.session.commit()
            # Debug distinct values
            cuisines = [c[0] for c in db.session.query(Recipe.Cuisine).distinct().all() if c[0]]
            courses = [c[0] for c in db.session.query(Recipe.Course).distinct().all() if c[0]]
            diets = [d[0] for d in db.session.query(Recipe.Diet).distinct().all() if d[0]]
            print("Distinct Cuisines:", cuisines)
            print("Distinct Courses:", courses)
            print("Distinct Diets:", diets)
            print("Database initialized with", Recipe.query.count(), "recipes.")
        except FileNotFoundError:
            print("Error: 'data/IndianFoodDataset_cleaned.csv' not found.")
        except Exception as e:
            print(f"Error initializing database: {e}")
# Function to extract image from URL
def get_recipe_image(url):
    try:
        response = requests.get(url, timeout=10, headers={'User-Agent': 'Mozilla/5.0'})
        response.raise_for_status()
        soup = BeautifulSoup(response.content, 'html.parser')
        # Target the main recipe image with specific class
        img = soup.find('img', class_=['img-fluid', 'img-thumbnail'])
        if img and 'src' in img.attrs:
            img_url = img['src']
            # Convert relative URL to absolute if necessary
            if img_url.startswith('/'):
                img_url = 'https://www.archanaskitchen.com' + img_url
            print(f"Found image URL: {img_url}")
            return img_url
        img = soup.find('img', src=lambda x: x and ('recipe' in x.lower() or 'food' in x.lower()))
        if img and 'src' in img.attrs:
            img_url = img['src']
            if img_url.startswith('/'):
                img_url = 'https://www.archanaskitchen.com' + img_url
            print(f"Fallback image URL: {img_url}")
            return img_url
        print(f"No suitable image found for {url}")
        return 'https://via.placeholder.com/300x200?text=No+Image'
    except Exception as e:
        print(f"Error fetching image from {url}: {e}")
        return 'https://via.placeholder.com/300x200?text=No+Image'

@app.route('/', methods=['GET', 'POST'])
def index():
    # Clear session on initial load or refresh (GET request)
    if request.method == 'GET':
        session.clear()

    # Fetch unique filter options with explicit column selection, excluding empty strings
    cuisines = db.session.query(Recipe.Cuisine).distinct().all()
    courses = db.session.query(Recipe.Course).distinct().all()
    diets = db.session.query(Recipe.Diet).distinct().all()

    # Handle filters (single selection)
    filters = {
        'cuisines': request.form.get('cuisines') if request.method == 'POST' else session.get('cuisines', ''),
        'courses': request.form.get('courses') if request.method == 'POST' else session.get('courses', ''),
        'diets': request.form.get('diets') if request.method == 'POST' else session.get('diets', '')
    }
    session['cuisines'] = filters['cuisines']
    session['courses'] = filters['courses']
    session['diets'] = filters['diets']

    # Query recipes based on filters
    query = Recipe.query
```

```
    query = Recipe.query
    if filters['cuisines']:
        query = query.filter(Recipe.Cuisine == filters['cuisines'])
    if filters['courses']:
        query = query.filter(Recipe.Course == filters['courses'])
    if filters['diets']:
        query = query.filter(Recipe.Diet == filters['diets'])
    filtered_recipes = [recipe.to_dict() for recipe in query.all()]
    # Session state for current recipe and filtered recipes
    current_recipe = session.get('current_recipe')
    show_no_recipes_modal = False
    if request.method == 'POST':
        action = request.form.get('action')
        print(f"Action: {action}, Filters: {filters}, Filtered Recipes: {len(filtered_recipes)}")
        if action == 'clear':
            return redirect(url_for('index'))
        elif action == 'filtered' and not filtered_recipes:
            show_no_recipes_modal = True
        elif action == 'filtered' and filtered_recipes:
            session['filtered_recipes'] = filtered_recipes
            # Add images to filtered recipes
            for recipe in filtered_recipes:
                if recipe['URL']:
                    recipe['image_url'] = get_recipe_image(recipe['URL'])
        elif action == 'random':
            all_recipes = [recipe.to_dict() for recipe in Recipe.query.all()]
            if all_recipes:
                current_recipe = random.choice(all_recipes)
                if current_recipe['URL']:
                    current_recipe['image_url'] = get_recipe_image(current_recipe['URL'])
        elif action == 'select_recipe':
            recipe_name = request.form.get('recipe_name')
            if recipe_name and filtered_recipes:
                for recipe in filtered_recipes:
                    if recipe['RecipeName'] == recipe_name:
                        current_recipe = recipe.copy()
                        if current_recipe['URL']:
                            current_recipe['image_url'] = get_recipe_image(current_recipe['URL'])
                        break
        elif action == 'back':
            current_recipe = None  # Return to filtered recipes list
            print(f"Back action triggered, filtered_recipes in session: {session.get('filtered_recipes', 'None')}")

    # Store the dictionary in the session
    session['current_recipe'] = current_recipe
    print(f"Rendering with current_recipe: {current_recipe is not None}, filtered_recipes: {len(session.get('filtered_recipes', []))}")

    return render_template(
        'index.html',
        cuisines=[c[0] for c in cuisines if c[0]],
        courses=[c[0] for c in courses if c[0]],
        diets=[d[0] for d in diets if d[0]],
        filters=filters,
        current_recipe=current_recipe,
        filtered_recipes=session.get('filtered_recipes', []),
        has_recipes=bool(filtered_recipes),
        show_no_recipes_modal=show_no_recipes_modal
    )

if __name__ == '__main__':
    init_db()  # Run once to populate the database
    app.run(debug=True, port=5000)
```

```python
from flask_sqlalchemy import SQLAlchemy
db = SQLAlchemy()
class Recipe(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    Srno = db.Column(db.Integer)
    RecipeName = db.Column(db.String(200), nullable=False)
    TranslatedRecipeName = db.Column(db.String(200))
    Ingredients = db.Column(db.Text, nullable=False)
    TranslatedIngredients = db.Column(db.Text)
    PrepTimeInMins = db.Column(db.Integer)
    CookTimeInMins = db.Column(db.Integer)
    TotalTimeInMins = db.Column(db.Integer)
    Servings = db.Column(db.Integer)
    Cuisine = db.Column(db.String(100))
    Course = db.Column(db.String(100))
    Diet = db.Column(db.String(100))
    Instructions = db.Column(db.Text)
    TranslatedInstructions = db.Column(db.Text)
    URL = db.Column(db.String(200))
    def to_dict(self):
        return {
            'Srno': self.Srno,
            'RecipeName': self.RecipeName,
            'TranslatedRecipeName': self.TranslatedRecipeName,
            'Ingredients': self.Ingredients,
            'TranslatedIngredients': self.TranslatedIngredients,
            'PrepTimeInMins': self.PrepTimeInMins,
            'CookTimeInMins': self.CookTimeInMins,
            'TotalTimeInMins': self.TotalTimeInMins,
            'Servings': self.Servings,
            'Cuisine': self.Cuisine,
            'Course': self.Course,
            'Diet': self.Diet,
            'Instructions': self.Instructions,
            'TranslatedInstructions': self.TranslatedInstructions,
            'URL': self.URL
        }
```

11.3 Strength Training

```python
# STRENGTH_MODEL/main.py
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curve, auc, average_precision_score
from sklearn.preprocessing import label_binarize
import seaborn as sns
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt
import cv2
import pandas as pd
import numpy as np
from utils import *
from demo import *

print("Starting main.py...")

try:
    print("Loading training and testing data...")
    data_train = pd.read_csv("train_angle.csv")
    data_test = pd.read_csv("test_angle.csv")
    print("Training data shape:", data_train.shape)
    print("Test data shape:", data_test.shape)
    print("Unique exercises in training data:", data_train['exercise'].unique())
except Exception as e:
    print(f"Error loading CSV files: {e}")
    exit(1)

# Print stage distribution
print("Target Counts in Test Data:")
print(data_test['target'].value_counts())

print("Evaluating Exercise Classification Model...")
try:
    exercise_predictions = evaluate_exercise(data_test, show=True)
except Exception as e:
```

```
print("Evaluating Exercise Classification Model...")
try:
    exercise_predictions = evaluate_exercise(data_test, show=True)
except Exception as e:
    print(f"Error evaluating exercise classification model: {e}")
    exit(1)

# 1. Confusion Matrix for Exercise Classification
# print("Creating confusion matrix for exercise classification...")
# cm_exercise = confusion_matrix(data_test['exercise'], exercise_predictions)
# plt.figure(figsize=(8, 6))
# sns.heatmap(cm_exercise, annot=True, fmt='d', cmap='Blues', xticklabels=exercise_model.classes_, yticklabels=exercise_model.classes_)
# plt.xlabel('Predicted Label')
# plt.ylabel('True Label')
# plt.title('Confusion Matrix (Exercise Classification)')
# plt.show()

# Train a model for target (stage) classification using full train and test sets
print("Training model for target (stage) classification...")
angle_columns = ['left_wrist_angle', 'right_wrist_angle', 'left_elbow_angle', 'right_elbow_angle',
                 'left_shoulder_angle', 'right_shoulder_angle', 'left_knee_angle', 'right_knee_angle',
                 'left_ankle_angle', 'right_ankle_angle', 'left_hip_angle', 'right_hip_angle']
X_train = data_train[angle_columns]
y_train = data_train['target']
X_test = data_test[angle_columns]
y_test = data_test['target']

# Encode the target labels
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_test_encoded = label_encoder.transform(y_test)

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Train a RandomForest model
model = RandomForestClassifier(random_state=42)
model.fit(X_train_scaled, y_train_encoded)

# Get predictions and probabilities
y_pred = model.predict(X_test_scaled)
y_pred_proba = model.predict_proba(X_test_scaled)

# Cross-validation on the combined dataset
print("Performing 5-fold cross-validation...")
data_all = pd.concat([data_train, data_test])
X_all = data_all[angle_columns]
y_all = label_encoder.transform(data_all['target'])
X_all_scaled = scaler.transform(X_all)
cv_scores = cross_val_score(model, X_all_scaled, y_all, cv=5, scoring='accuracy')
print("Cross-validation accuracy scores:", cv_scores)
print("Mean CV accuracy:", cv_scores.mean())
```

```
print("Starting real-time feedback...")
try:
    correct_feedback(video=r"squat2.mp4")
except Exception as e:
    print(f"Error in feedback loop: {e}")

print("Cleaning up...")
cv2.destroyAllWindows()
print("Script completed.")
```

11.4 Yoga Training

```
from sklearn.svm import SVC
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import pandas as pd
import os
import cv2

from utils import *
from demo import correct_feedback  # Import only `correct_feedback`

# Check if dataset files exist
if not os.path.exists("train_angle.csv") or not os.path.exists("test_angle.csv"):
    raise FileNotFoundError("Error: Missing dataset files (train_angle.csv or test_angle.csv)")

# Load dataset
data_train = pd.read_csv("train_angle.csv")
data_test = pd.read_csv("test_angle.csv")

# Prepare training data
X, Y = data_train.iloc[:, :-1], data_train['target']

# Train the model
model = SVC(kernel='rbf', decision_function_shape='ovo', probability=True)
model.fit(X, Y)

# Evaluate the model
predictions = evaluate(data_test, model, show=True)

# Create a confusion matrix
cm = confusion_matrix(data_test['target'], predictions)

# Display the confusion matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel('Predicted Label')
```

```
# Evaluate the model
predictions = evaluate(data_test, model, show=True)

# Create a confusion matrix
cm = confusion_matrix(data_test['target'], predictions)

# Display the confusion matrix using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=model.classes_, yticklabels=model.classes_)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()

# Run real-time pose detection and correction using webcam
correct_feedback(model, 0, "teacher_yoga/angle_teacher_yoga.csv")  # Use 0 for webcam input

cv2.destroyAllWindows()
```

12. REFERENCES

[1] Y. Wang and W. B. Collins, “Systematic evaluation of mobile fitness apps: Apps as the Tutor, Recorder, Game Companion, and Cheerleader,” *Telemat. Inform.*, vol. 59, p. 101552, Jun. 2021, doi: 10.1016/j.tele.2020.101552.

[2] M. Kranz *et al.*, “The mobile fitness coach: Towards individualized skill assessment using personalized mobile devices,” *Pervasive Mob. Comput.*, vol. 9, no. 2, pp. 203–215, Apr. 2013, doi: 10.1016/j.pmcj.2012.06.002.

[3] P. Choudhary, A. Kumar, A. Raja, A. Sharma, and K. Jain, “Yoga Pose Detection and Feedback Generation: A Review,” Jan. 27, 2023, *Social Science Research Network, Rochester, NY*: 4990385. doi: 10.2139/ssrn.4990385.

[4] H. Dhakate, S. Anasane, S. Shah, R. Thakare, and S. G. Rawat, “Enhancing Yoga Practice: Real-time Pose Analysis and Personalized Feedback,” in *2024 International Conference on Emerging Systems and Intelligent Computing (ESIC)*, Feb. 2024, pp. 35–40. doi: 10.1109/ESIC60604.2024.10481659.

[5] S. Sarkar, Y. K. Choi, and K. K. Kim, “A Structured Review and Evaluation of Android Mobile Applications for Yoga Support,” in *Healthcare of the Future 2022*, IOS Press, 2022, pp. 75–78. doi: 10.3233/SHTI220325.

[6] I. A. Alao, “Development of diet and fitness tracking app,” ERA. Accessed: Mar. 29, 2025. [Online]. Available: https://era.library.ualberta.ca/items/00be76b1-ad34-4618-8076-0cd2e9650d69

[7] D. Chowdhury, A. Roy, S. R. Ramamurthy, and N. Roy, “CHARLIE: A Chatbot That Recommends Daily Fitness and Diet Plans,” in *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Mar. 2023, pp. 116–121. doi: 10.1109/PerComWorkshops56833.2023.10150359.

13. RESEARCH PAPER(STATUS)

14. PLAGARISM REPORT