A Synopsis of Project on

An Application based Data-Driven AI Fitness Trainer integrating Deep Learning Algorithms and Computer Vision

Submitted in partial fulfillment of the requirements for the award
of the degree of

Bachelor of Engineering

in

Computer Science and Engineering (Data Science)

by

Riya Sawant(21107019)
Rutuja Patil(21107012)
Tanvi Panchal(21107006)
Sneha Sabat(22207008)

Under the Guidance of

Ms. Anagha Aher

Department of Computer Science and Engineering (Data Science)
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W)-400615

UNIVERSITY OF MUMBAI
Academic Year 2024-2025

Approval Sheet

This Project Report entitled ***"An Application based Data-Driven AI Fitness Trainer integrating Deep Learning Algorithms and Computer Vision "*** Submitted by ***"Riya Sawant"(21107019),"Rutuja Patil"(21107012),"Tanvi Panchal" (21107006)*** and ***"Sneha Sabat"(22207008)*** is approved for the partial fulfillment of the requirenment for the award of the degree of ***Bachelor of Engineering*** in ***Computer Science and Engineering (Data Science)*** from ***University of Mumbai***.

Ms. Anagha Aher
Guide

Ms. Anagha Aher
HOD, Computer Science and Engineering (Data Science)

Place: A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"An Application based Data-Driven AI Fitness Trainer integrating Deep Learning Algorithms and Computer Vision "*** submitted by ***"Riya Sawant" (21107019),"Rutuja Patil" (21107012),"Tanvi Panchal" (21107006),"Sneha Sabat" (22207008)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Computer Science and Engineering (Data Science)***,to the University of Mumbai,is a bonafide work carried out during academic year 2024-2025.

Ms. Anagha Aher
Guide

Ms. Anagha Aher
HOD, Computer Science and Engineering (Data Science)

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Internal Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

We have great pleasure in presenting the synopsis report on **An Application based Data-Driven AI Fitness Trainer integrating Deep Learning Algorithms and Computer Vision.** We take this opportunity to express our sincere thanks towards our guide **Ms. Anagha Aher** for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Ms. Anagha Aher** Head of Department for his encouragement during the progress meeting and for providing guidelines to write this report.

We express our gratitude towards BE project co-ordinator **Ms. Poonam M. Pangarkar**, for being encouraging throughout the course and for their guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Riya Sawant
(21107019)

Rutuja Patil
(21107012)

Tanvi Panchal
(21107006)

Sneha Sabat
(22207008)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Riya Sawant(21107019)

Rutuja Patil(21107012)

Tanvi Panchal(21107006)

Sneha Sabat(22207008)

Date:

Abstract

Generic workout routines, a lack of personalized advice, and challenges in time management often result in ineffective workouts and frustration in reaching fitness goals. This AI Fitness Trainer addresses these problems by offering highly personalized recommendations through Deep Neural Networks (MLP), ensuring guidance tailored to each user's unique fitness profile and goals. The AI trainer incorporates Mediapipe and Cvzone technologies along with a Convolutional Neural Network model for advanced pose estimation and real-time analysis. At the same time, a voice assistant provides instant feedback on posture. Additionally, users can view weekly progress through a detailed dashboard and keep track of performance on a dynamic leaderboard. This comprehensive system ensures a tailored fitness experience, combining precise recommendations, real-time feedback, and progress tracking to enhance user engagement and motivation.

Keywords — MediaPipe, Computer Vision Zone (CVZone), Con volutional Neural Network (CNN), Recommendation Engine, Multi-Layer Perceptron (MLP), Role-Based Access Control (RBAC).

Contents

1 Introduction **1**
- 1.1 Motivation . . . 2
- 1.2 Problem Statement . . . 3
- 1.3 Objectives . . . 4
- 1.4 Scope . . . 6

2 Literature Review **8**
- 2.1 Comparative Analysis of Recent Study . . . 8

3 Project Design **14**
- 3.1 Existing System . . . 14
- 3.2 Proposed System . . . 16
  - 3.2.1 Critical Components of System Architecture . . . 17
- 3.3 System Diagrams . . . 21
  - 3.3.1 UML Diagram . . . 21
  - 3.3.2 Activity Diagram . . . 21
  - 3.3.3 Use Case Diagram . . . 23
  - 3.3.4 Sequence Diagram . . . 24

4 Project Implementation **27**
- 4.1 Code Snippets . . . 27
- 4.2 Steps to access the System . . . 34
- 4.3 Timeline Sem VIII . . . 39

5 Testing **43**
- 5.1 Software Testing . . . 43
- 5.2 Functional Testing . . . 46

6 Result and Discussions **50**

7 Conclusion **57**

8 Future Scope **59**
- 8.1 System Setup and Deployment Guide . . . 63

Publication **66**

List of Figures

1.1 System architecture proposed by Didar Divani Sanandaj and Sasan H. Alizadeh et al.(2018) [14] . . . 3

2.1 Categorization of Literature Review . . . 8

3.1 Proposed system architecture . . . 16
3.2 Working of Data-driven AI Fitness Trainer . . . 18
3.3 AI Fitness Trainer Mobile Interface . . . 19
3.4 Working of Workout Recommendation Engine . . . 19
3.5 Activity Diagram . . . 22
3.6 Use Case Diagram . . . 23
3.7 Sequence Diagram . . . 25

4.1 MLP model for predicting customized workout plans from user data . . . 28
4.2 FastAPI backend for real-time personalized workout recommendation . . . 29
4.3 Pose landmark extraction using MediaPipe for exercise classification . . . 30
4.4 Pose classification model trained on exercise landmarks using TensorFlow . . . 31
4.5 Flask backend for exercise classification and real-time tracking integration . . . 32
4.6 MLP model for generating personalized multi-label meal recommendations . . . 33
4.7 Landing Page of AIFT . . . 35
4.8 Login Page of AIFT . . . 35
4.9 User input form . . . 36
4.10 Workout Recommendation . . . 36
4.11 User input form . . . 37
4.12 User input form . . . 37
4.13 Diet Recommendation . . . 37
4.14 User input form . . . 38
4.15 AI Fitness Trainer . . . 38
4.16 Progress tracker of AIFT . . . 38
4.17 Timeline of the Project Milestones . . . 42

6.1 Performance comparison of AI-based fitness trainer systems across key metrics 52
6.2 Performance Evaluation of AI Trainer: Training Vs. Validation . . . 53

List of Tables

2.1 Comparative Analysis of Recent Study 9
2.2 Comparative Analysis of Recent Study 10
2.3 Comparative Analysis of Recent Study 11
2.4 Comparative Analysis of Recent Study 12

5.1 Functional Testing Table - Part 1 46
5.2 Functional Testing Table - Part 2 47

6.1 Comparison of Initial and Enhanced Implementations Across Different Modules 51
6.2 Workout Recommendation Performance Evaluation 54

List of Abbreviations

AI:	Artificial Intelligence
AIFT:	AI Fitness Trainer
DNN:	Deep Neural Networks
CNN:	Convolutional Neural networks
MLP:	Multi-Layer Perceptron
DFDs:	Data Flow Diagram
RBAC:	Role-Based Access Control

Chapter 1

Introduction

In today's fast-paced digital world, the pursuit of health and fitness goals often takes a back seat due to hectic schedules and lifestyle constraints. With increasing awareness about fitness, individuals now approach their wellness journeys with distinct expectations—some aiming for weight loss, others focusing on muscle gain, endurance improvement, or managing specific health conditions. However, most widely available fitness programs and mobile applications continue to follow a generic, one-size-fits-all model. These standardized routines, although accessible, fail to accommodate the diversity of user needs and personal goals. This lack of customization often leads to disengagement, as users feel overlooked and unmotivated when they cannot relate to or benefit from the prescribed workouts. The inability of these traditional programs to address individual challenges results in minimal user retention and suboptimal fitness outcomes.

One of the fundamental limitations of these conventional approaches lies in their rigidity and static nature. They usually offer pre-defined workout templates or instructional manuals with no real-time feedback mechanisms or adaptability based on user performance. As a result, users are frequently left questioning the correctness of their form, the effectiveness of their workouts, or whether they're even making progress. This ambiguity not only affects motivation but also increases the risk of improper form or injury. While some users might utilize fitness trackers or wearables, these tools primarily monitor surface-level metrics like step count or calories burned, without truly guiding the user through the workout or providing actionable feedback. Over time, the lack of engagement, personalization, and support causes users to abandon their fitness journey altogether, leading to inconsistency and frustration.

In this light, there is a pressing need for dynamic, intelligent, and interactive fitness solutions that prioritize personalized fitness guidance. Unlike static models, these systems adapt workout plans to suit the evolving goals, preferences, fitness levels, and health conditions of each individual. By integrating technologies like computer vision, deep learning, and real-time analytics, modern intelligent systems are capable of offering immediate feedback on exercise form, tracking progress with precision, and delivering motivational cues during workouts. These systems mimic the experience of a personal trainer by correcting posture, offering encouragement, and tailoring routines based on the user's data and preferences. This not only enhances user experience but also ensures safer and more efficient workout execution. Moreover, this real-time interaction serves as a powerful motivator—keeping users

engaged and committed to their fitness goals.

Furthermore, these adaptive solutions provide long-term benefits by fostering a deep sense of connection between users and their fitness journey. Rather than forcing users to follow rigid plans, intelligent systems offer flexibility—adapting workouts based on daily progress, user fatigue, motivation levels, or even available equipment. This responsiveness makes the experience more enjoyable and reduces burnout, making users more likely to continue exercising consistently. Over time, such systems can help build sustainable habits, improve health outcomes, and promote a lifestyle centered on well-being and self-improvement. For beginners, this means accessible guidance; for athletes, it means optimization and performance enhancement. Ultimately, dynamic and interactive systems are revolutionizing the fitness landscape by making personalized health support more scalable, effective, and engaging than ever before.

1.1 Motivation

Traditional fitness programs often fall short when it comes to meeting the individualized requirements of users. These programs are largely based on a generic, one-size-fits-all model that assumes every person has the same capabilities, goals, and health conditions. In reality, individuals have diverse needs—some may be recovering from injuries, others may have chronic conditions, and many have specific goals such as building muscle, reducing body fat, or increasing cardiovascular endurance. For example, someone with a knee injury may need low-impact strength exercises that avoid joint strain, while another person may focus on improving core strength for posture correction. Unfortunately, traditional fitness plans rarely take such variations into account. They often offer rigid templates or pre-recorded videos with no adaptability to personal context, leading users to perform exercises that may be ineffective—or worse, harmful to their condition.

Moreover, these outdated approaches rely heavily on manual tracking of progress and lack the dynamic ability to adjust based on user feedback or performance. Users are often required to input their workout logs, assess their form without guidance, and make sense of their progress without professional insight. This makes the experience tedious and disconnected. When goals such as muscle toning, fat loss, or endurance building are not supported with targeted, adaptive programming, users feel discouraged. As their routines fail to yield noticeable results or feel unsuitable to their lifestyle or capabilities, their motivation dwindles. Over time, this frustration accumulates, causing many to abandon their fitness journey altogether. Therefore, there is a growing demand for smart, personalized systems that understand and adapt to individual differences—bridging the gap between user needs and effective fitness solutions.

This illustration in Figure 1.1 shows a hybrid recommendation model including Collaborative Filtering (CF) and Content Based Filtering (CBF). First, user and movie information is integrated. Then, it goes through a neural network, with the output being the final recommendation list. The major drawback with CF and CBF in recommendation systems is handling personalization and dynamic data patterns among users effectively, which is most important for fitness trainer applications [14]. The challenges are overcome in the AI Fitness

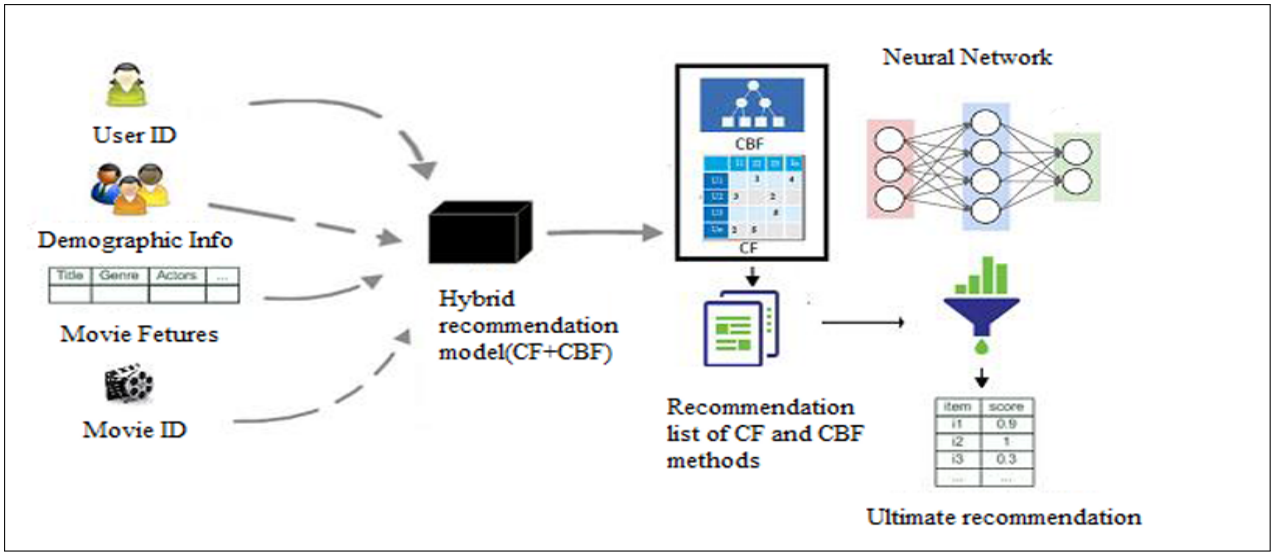

Figure 1.1: System architecture proposed by Didar Divani Sanandaj and Sasan H. Alizadeh et al.(2018) [14]

Trainer by utilizing artificial intelligence to provide a completely personalized fitness experience. It can advise on workouts that are spot on for the user because real-time data like user performance, exercise form, and progress are taken into account. These are no longer static routines as they dynamically evolve into different recommendations all based on fitness level and goals.

Some of the greatest strengths of this AI-enabled technique include instantaneous feedback inspired from. Contrary to old-fashioned programs that need users to guess whether their form is right or whether they are on track with their workout, the AI Fitness Trainer provides instant step-by-step guidance. If there is an issue with the user's form during an exercise, the AI will be able to correct it on the spot, preventing injuries and ensuring the workout is completed effectively.

1.2 Problem Statement

In the era of digital transformation, where artificial intelligence is reshaping various aspects of daily life, the domain of personal health and fitness still faces challenges in providing scalable, personalized, and intelligent support. Despite the growing awareness about staying fit and healthy, many individuals struggle to maintain consistent routines, often due to a lack of real-time guidance, motivation, and tailored feedback. The traditional approach of following static workout plans or generic fitness applications often falls short in addressing the unique needs of each user, creating a significant gap between effort and outcome. This calls for an AI-powered fitness solution that can dynamically guide users, provide corrective feedback, and offer customized fitness and nutrition recommendations in real-time.

Achieving personal fitness goals is often a complex and demanding task, primarily because individuals are expected to manage two major responsibilities simultaneously: performing exercises with correct form and monitoring their overall progress. Balancing both tasks re-

quires sustained cognitive effort and divided attention, which not only increases the chances of performing exercises incorrectly but also heightens the risk of injury. Poor form during exercises can diminish the effectiveness of workouts, delay results, and even lead to long-term physical damage if left uncorrected. This cognitive overload becomes even more challenging for individuals who do not have access to real-time expert guidance or the technical understanding to self-correct their posture or nutrition routines.

In most real-world scenarios, personalized and immediate feedback—which is critical for correcting form and optimizing routines—is either unavailable or costly. Traditional methods such as hiring personal trainers or subscribing to fitness centers may still not offer real-time corrections during workouts. Static, one-size-fits-all workout plans, pre-recorded fitness videos, and generic fitness tracking apps fall short in accommodating the individual's fitness levels, physical limitations, dietary needs, and health conditions. This lack of personalization often leads to demotivation, inconsistencies in workout patterns, and eventually, the abandonment of fitness goals. Users feel disengaged due to the absence of real-time feedback, adaptive changes, and intelligent insights tailored to their progress and well-being.

The AI Fitness Trainer (AIFT) aims to directly address these challenges by offering a smart, responsive, and personalized solution. Leveraging advanced deep learning techniques, pose estimation, and real-time voice feedback, the system ensures that users receive immediate posture correction guidance while exercising. This helps reduce the chances of injury, enhances workout precision, and enables users to derive maximum benefit from their sessions. Additionally, the AI Trainer supports personalized diet planning based on user-specific data such as age, weight, allergies, health conditions, and dietary preferences. This ensures that the user receives balanced, goal-oriented nutrition plans, without manual intervention or the need for expert consultation.

By dynamically adjusting workout and diet recommendations based on real-time inputs and user feedback, the system provides a continuous learning loop that improves with usage. This removes the need for guesswork and empowers users to focus entirely on execution, rather than management. Furthermore, by operating at a significantly lower cost compared to personal trainers or nutritionists, the AI Fitness Trainer democratizes access to customized fitness guidance, making it scalable, inclusive, and accessible to a broader audience. In conclusion, the AI Fitness Trainer overcomes the limitations of traditional fitness methods by integrating technology and personalization. It supports users with real-time corrective feedback, customized workout and diet plans, and intelligent tracking mechanisms, thereby making fitness routines more effective, engaging, and sustainable.

1.3 Objectives

The AI Fitness Trainer plans to change the fitness world by offering personalized guidance and real-time feedback via modern technologies-a blend of deep learning, computer vision and convolutional neural networks (CNNs). Some aims of the system include creating custom workout and diet plans, tracking user performance along the way, and keeping them motivated through an entire process. By utilizing these contemporary techniques, it handles greatest challenges in fitness and keeps users on track, optimizes their performance, and

accomplishes their health goals.

- To provide real-time feedback on exercise form, gesture, posture, and accuracy percentage, as well as to detect and track exercises performed, calculate calories burned, and address user slacking off during workouts using deep learning-powered human pose estimation, computer vision technologies and convolutional neural networks (CNNs): The AI Fitness Trainer utilizes deep learning-powered human pose estimation and computer vision technologies to monitor users as they perform exercises. Through CNNs, the system tracks body movements, evaluates posture, and calculates the accuracy of each exercise, providing real-time feedback to help users maintain proper form and prevent injuries. It also identifies and tracks specific exercise types, estimates calories burned, and assesses whether users are slacking off during workouts. This continuous analysis allows the users to get the most out of their routines while ensuring safety and efficiency in one complete exercise routine.

- To provide personalized workout routines based on the user's health conditions, age, weight, height, workout levels, fitness-based goals using a deep learning-based Multilayer Perceptron network: The system creates customized workout routines by leveraging a deep learning-based Mul- tilayer Perceptron (MLP) network. By analyzing key inputs such as the user's health conditions, age, and specific fitness goals, the AI Fitness Trainer recommends workouts that are safe, effective, and aligned with the user's personal objectives. This personal- ized approach eliminates the one-size-fits-all nature of traditional fitness plans, ensuring that users can progress toward their fitness goals at their own pace and within their physical capabilities.

- To create a dynamic and customizable diet plan tailored to the user's health conditions, age, preferences, height, weight, and allergies using a deep learning-based Multilayer Perceptron network: The AI Fitness Trainer goes beyond just workouts by offering personalized diet recommendations. Using the same MLP network, it creates dynamic diet plans tailored to the user's health conditions, age, preferences, height, weight, and allergies. This personalized diet approach ensures that users receive balanced nutrition that complements their workout routines, helping them achieve their fitness goals more effectively. The system also adjusts diet plans over time based on progress, making them adaptable to the user's evolving needs.

- To generate weekly performance reports by integrating data from work- outs and diet recommendations, descriptive statistics, and data visualization techniques, and to enhance user engagement. To keep users motivated and informed, the AI Fitness Trainer generates detailed weekly perfor- mance reports. By integrating data from both workouts and diet plans, the system uses data aggregation and descriptive statistics to highlight progress in key areas, such as calories burned, exercises completed, and nutritional intake. Data visualization tech- niques present this information in an easy-to-understand format, helping users stay engaged with their progress.

1.4 Scope

The AI Fitness Trainer (AIFT) represents a cutting-edge, data-driven solution engineered to fit seamlessly into various fitness environments—ranging from traditional fitness centers and home workout spaces to modern corporate wellness initiatives. Its versatility makes it adaptable to the unique demands of different user groups, whether they are beginners striving to get started, seasoned athletes pushing limits, or individuals with specific health concerns requiring tailored fitness interventions. What sets AIFT apart is its core foundation of intelligent personalization, integrating dynamic workout routines and nutritional guidance that evolve in real time to suit the personal goals, medical considerations, and preferences of each user.

AIFT's recommendation engine is designed with exceptional flexibility, allowing it to plug into third-party platforms and existing fitness applications without friction. This integration capability not only expands the tool's usability across various domains but also ensures that users get a consistent and highly personalized experience regardless of the platform they engage with. The AI system does not rely on static routines; rather, it evolves by tracking user progress and continually fine-tuning workout difficulty, rep counts, form corrections, and diet plans, making the overall experience not only adaptive and insightful but also responsive and motivating. This type of adaptability is especially important for maintaining user engagement over the long term and preventing common pitfalls like stagnation or demotivation.

One of AIFT's most powerful features is its ability to provide real-time feedback during workouts using pose estimation and voice guidance, enabling immediate corrections to posture and technique. This functionality significantly reduces the risk of injury, improves workout efficiency, and empowers users with actionable insights during the actual training session—something that even many human trainers may not provide consistently. Additionally, AIFT's dynamic adjustment system constantly monitors user input and feedback to refine dietary recommendations based on changes in fitness goals, body weight, allergies, health conditions, or nutrient preferences. This ensures that users not only exercise more effectively but also nourish their bodies in a way that accelerates progress.

For fitness centers and gyms, incorporating AIFT offers a strategic advantage by enabling them to offer their clients a tech-driven, personalized coaching system at scale. It transforms the conventional group-class model into a one-on-one intelligent training ecosystem, enhancing member satisfaction and retention. For individuals, AIFT becomes a virtual companion—one that motivates, tracks, advises, and evolves with them throughout their fitness journey. Its intuitive design and behavior-tracking capabilities establish a continuous learning loop, adapting workouts and meal plans in response to user behavior and physiological progress, thereby helping users avoid plateaus and push through barriers.

Moreover, AIFT's ability to sync with wearable devices and fitness equipment allows it to collect deeper biometric and performance data, which further enhances its personalization features. Whether someone is working out at a gym, from home, or following corporate wellness protocols, AIFT fits naturally into the lifestyle, delivering precision-driven support and improving adherence to fitness regimens. This fusion of real-time analytics, adaptability, and personal connection transforms the fitness experience into something much more engaging,

effective, and enjoyable.

In essence, the AI Fitness Trainer isn't just a tool—it becomes a personalized fitness ecosystem, guiding users across physical, nutritional, and motivational dimensions. By offering a continuously evolving experience tailored to user progress and preferences, AIFT fosters higher levels of commitment and satisfaction, ensuring users remain active, focused, and inspired to achieve their goals—whether they're at the gym, in their living room, or balancing work-life wellness goals.

Chapter 2

Literature Review

In recent years, the intersection of artificial intelligence and fitness has become a focal point of research, reflecting a growing demand for personalized training solutions. As fitness enthusiasts increasingly seek tailored guidance to achieve their health goals, various studies have explored innovative approaches to enhance user experience and effectiveness in workout routines. This literature review examines the key findings and contributions of notable research in the field, highlighting the challenges and opportunities for developing advanced AI-driven fitness solutions.

2.1 Comparative Analysis of Recent Study

Recent advancements in AI, deep learning, and blockchain have transformed fields like posture estimation, recommendation systems, and secure data sharing. Studies using technologies like MediaPipe for posture analysis and deep neural networks for recommendations have improved real-time accuracy and personalization. Blockchain has enhanced security in subscription payments and medical data sharing. However, limitations such as task-specific constraints, computational demands, and scalability remain. Figure 2.1 reviews these recent studies, highlighting methodologies and shortcomings to identify opportunities for further improvement.

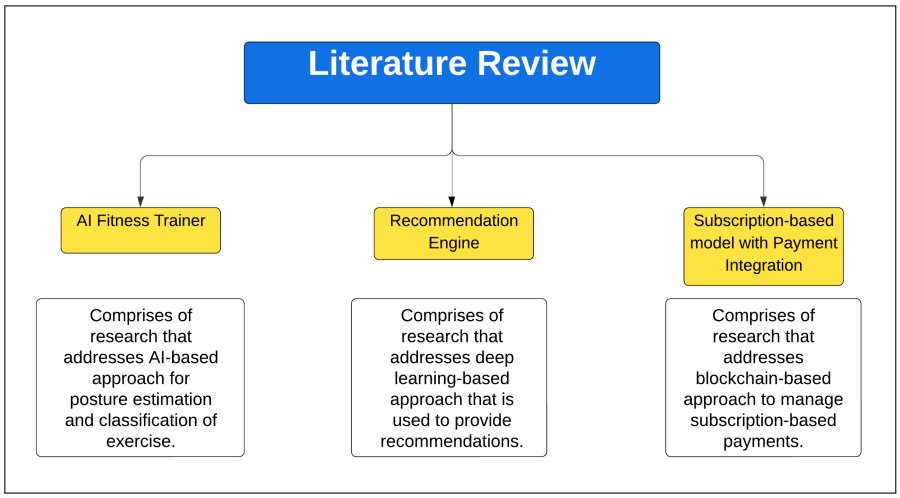

Figure 2.1: Categorization of Literature Review

Comparative analysis of the above research contributions highlights a clear trend toward hybrid, secure, and intelligent systems that balance performance, personalization, and privacy. However, challenges remain in ensuring real-time efficiency, minimizing computational burdens, and improving generalizability across domains. Comparative analysis of the research papers is summarized as follows:

Table 2.1: Comparative Analysis of Recent Study

Sr. No	Title	Author(s)	Year	Methodology	Drawback
1	Robust Intelligent Posture Estimation for an AI Gym Trainer using Mediapipe and OpenCV [2]	Venkata Sai P, Bhamidipati Ishi Saxena, Mrs. D. Saisanthiya, Mrs. D. Saisanthiya, Dr. Mervin Retnadhas	2023	The paper presents an AI approach for gym posture estimation using real-time video processing and pose estimation, tested for accuracy	The system is limited to predefined exercises and may underperform in poor lighting or with low-resolution cameras.
2	AI Trainer: Autoencoder Based Approach for Squat Analysis and Correction [4]	Mukundan Chariar, Shreyas Rao, Aryan Irani, Shilpa Suresh, C S Asha	2023	It classifies squat types and recommends appropriate versions using MediaPipe and deep learning.	The system is limited to squats and may not handle untrained exercises.
3	Real-Time Short-Range Human Posture Estimation Using mmWave Radars and Neural Networks [6]	Han Cui, Naim Dahnoun	2022	It achieves 20 fps real-time estimation with 12.2 cm mean error and 71.3 percent precision.	Hardware requirements restrict broader software exploration.
4	A Framework for Recognition and Prediction of Human Motions in Human-Robot Collaboration Using Probabilistic Motion Models [3]	Thomas Callens, Tuur van der Have, Sam Van Rossom, Joris De Schutter, Erwin Aertbeli	2020	A framework for recognizing and predicting human motions enhances interaction with robotics.	The phase speed estimation module has low performance, limited to certain robotic domains.

Posture estimation systems have increasingly adopted a range of artificial intelligence (AI) techniques to improve both accuracy and real-time response, especially within fitness and human-computer interaction domains. A notable study implements real-time video processing combined with keypoint detection using tools such as MediaPipe and OpenCV for analyzing posture during gym workouts [2]. This method effectively identifies body keypoints for dynamic feedback; however, it suffers from limitations when dealing with unfamiliar or untrained exercise types, and its performance is significantly impacted by external factors like lighting conditions and camera quality, which restrict its applicability across diverse real-world environments.

In a more focused approach, another study introduces a deep learning-based autoencoder integrated with MediaPipe for the specific purpose of squat form classification and correction [4]. The model delivers accurate analysis for squats by detecting improper postures and providing feedback. Nevertheless, its narrow applicability to squats alone limits its adaptability for other exercise forms or general fitness tracking scenarios. Meanwhile, real-time human posture estimation has also seen advances through the use of millimeter-wave (mmWave) radars paired with neural networks [6]. This method achieves high precision and maintains

a frame rate of 20 frames per second, offering excellent performance. Despite its advantages, the dependency on specialized hardware like mmWave sensors makes it unsuitable for broader adoption in software-only or low-resource applications.

In the context of human-robot interaction, one study proposes a framework based on probabilistic motion models for recognizing and predicting human movement patterns [3]. This method significantly enhances real-time motion prediction and facilitates better coordination between humans and robotic systems. However, the framework's primary focus on robotic device control reduces its relevance in other domains such as fitness monitoring or general posture assessment.

Table 2.2: Comparative Analysis of Recent Study

Sr. No	Title	Author(s)	Year	Methodology	Drawback
5	DAN: a deep association neural network approach for personalization recommendation [16]	Xu-na WANG, Qingmei TAN	2020	A deep association neural network (DAN) is proposed for implicit feedback recommendations.	Feature interaction prediction through matrix decomposition may not reflect real-world situations.
6	Deep learning for recommendation systems [7]	Badiâa Dellal-Hedjazi, Zaiai Alimazighi	2020	A deep learning recommendation system using MLP combines demographic and content-based filtering for accuracy.	The algorithm's recommendations may lack personalization due to computational complexity.
7	Realizing an Efficient IoMT-Assisted Patient Diet Recommendation System Through Machine Learning Model [9]	Celestine Iwendi, Suleman Khan, Joseph Henry Anajemba, Ali Kashif Bashir, Fazal Noor	2020	The system leverages IoMT and machine learning for personalized diet recommendations.	Requires robust data processing and model optimization.

Transitioning to the domain of recommendation systems, deep learning has remained a cornerstone technique. A model known as the Deep Association Neural Network (DAN) applies matrix decomposition to predict feature interactions, enabling personalized recommendation outputs [16]. While the model excels in scenarios where features interact meaningfully, it faces performance degradation when feature interactions are weak or non-existent, resulting in less accurate recommendations. Similarly, another approach integrates demographic filtering with content-based filtering using multilayer perceptrons (MLPs) to enhance personalization and speed [7]. Though effective in improving recommendation quality, the high computational complexity of MLP models imposes limitations on their real-time performance in large-scale applications.

One other approach integrates IoMT (Internet of Medical Things) with machine learning models to design an efficient, patient-centric diet recommendation system [9]. By collecting real-time physiological data such as blood pressure, glucose levels, and daily activity patterns through wearable sensors, the system dynamically adjusts dietary suggestions tailored to individual health needs. This intelligent combination not only assists in managing chronic conditions like diabetes or hypertension but also helps in reducing the burden on healthcare

providers by automating the nutritional assessment process.

Table 2.3: Comparative Analysis of Recent Study

Sr. No	Title	Author(s)	Year	Methodology	Drawback
8	An Embedding-based Deep Learning Approach for Movie Recommendation [13]	Ram Murti Rawat, Vikrant Tomar, Vinay Kumar	2020	The recommendation system employs deep learning and embedding techniques, evaluated by RMSE and MAE.	MLP can solve complex tasks but increases computation time.
9	An Efficient Hybrid Recommendation Model With Deep Neural Networks [8]	Zhenhua Huang, Chang Yu, Juan Ni, Hai Liu, Chun Zeng, Yong Tang	2019	The paper introduces a Deep Metric Factorization Learning (DMFL) model combining deep learning, factorization machines, and metric learning for personalized recommendations.	The model struggles with sparse datasets and high computational demands, affecting real-time efficiency.
10	A hybrid recommender system using Multi Layer Perceptron Neural Network [14]	Didar Divani Sanandaj, Sasan H. Alizadeh	2018	A hybrid recommender system integrates collaborative and content-based filtering with an artificial neural network (ANN) to address the cold-start problem.	The ANN may reduce model accuracy.
11	A Survey of Recommender Systems Based on Deep Learning [11]	R. Mu	2018	Provides an extensive review of deep learning-based recommender systems.	Deep learning models require high computational resources.

Further enhancements in the recommendation system landscape include embedding-based techniques. One such model uses embeddings in conjunction with deep learning architectures to build a movie recommendation engine [13]. While these embeddings capture latent features effectively, the computational intensity of the model increases latency, making it less suitable for complex or time-sensitive tasks. Another noteworthy contribution is a hybrid model that combines factorization machines, deep learning, and metric learning to elevate the degree of personalization in recommendation systems [8]. Despite its innovation, the model encounters difficulties when operating on sparse datasets, and the computational overhead limits its responsiveness in real-time environments.

A different study tackles the cold-start problem—a common challenge in recommendation systems—by blending collaborative filtering, content-based filtering, and artificial neural networks (ANN) [14]. This hybrid architecture significantly boosts recommendation accuracy for new users or items. However, further optimization strategies could be employed to improve both computational efficiency and prediction accuracy.

Furthermore, the integration of deep learning into recommender systems has shown to

significantly enhance their accuracy and adaptability. Deep neural networks such as CNNs and RNNs offer powerful capabilities in modeling user behavior and complex item features. These models are capable of learning hierarchical representations and capturing nonlinear relationships between users and their preferences, resulting in more relevant and personalized recommendations [11]. However, challenges related to interpretability, training complexity, and data sparsity still exist, necessitating further research to balance performance with explainability and real-time responsiveness.

Table 2.4: Comparative Analysis of Recent Study

Sr. No	Title	Author(s)	Year	Methodology	Drawback
12	BlockSubPay - A Blockchain Framework for Subscription-Based Payment in Cloud Service [12]	Yustus Eko Oktian, Elizabeth N. Witanto, Sandra Kumi, Sang-Gon Lee	2019	The BlockSubPay methodology utilizes blockchain and smart contracts for secure subscription payments.	Implementing blockchain requires technical expertise.
13	Subscription-Based Data-Sharing Model Using Blockchain and Data as a Service [10]	Fahad Ahmad, Al-Zahrani	2020	A blockchain-based data-sharing model employs a subscription system for secure transactions and Data as a Service (DaaS).	Scalability challenges arise from consensus mechanisms and large data processing.
14	A Blockchain-Based Framework for Data Sharing with Fine-Grained Access Control in Decentralized Storage Systems [1]	S. Wang, Y. Zhang, Y. Zhang	2018	Introduces a blockchain-based framework for secure and controlled data sharing.	Decentralized storage solutions need efficient scalability mechanisms.
15	RBAC-SC: Role-Based Access Control Using Smart Contract [15]	Jason Paul Cruz, Yuichi Kaji, Naoto Yanai	2018	Proposes a role-based access control system utilizing blockchain and smart contracts for security.	Implementation complexity increases with system size.

In the realm of secure data-sharing, innovative models have begun incorporating blockchain technologies. One such model uses smart contracts to manage subscription-based payments in cloud services [12], ensuring transaction transparency and security. In a similar effort, another study develops a blockchain-based framework for electronic medical record (EMR) sharing, employing encryption and zero-knowledge proofs to protect patient privacy [10]. While this approach offers robust security, it faces hurdles in widespread adoption due to the technical infrastructure demands placed on users and healthcare providers. Expanding on this, another work integrates Data-as-a-Service (DaaS) with decentralized data-sharing mechanisms using blockchain [1]. Although this offers transparent and secure transactions, the model struggles with scalability issues when managing large datasets across multiple distributed nodes.

To support secure and efficient data exchange in distributed environments, blockchain-based data sharing frameworks have also emerged as a vital component. These frameworks provide fine-grained access control using smart contracts and attribute-based encryption,

allowing data owners to define precise access policies. Such an approach guarantees not only data integrity and traceability but also minimizes unauthorized access and manipulation. In healthcare and similar domains, this level of control is critical for enabling trustworthy, cross-institutional data sharing, where multiple parties may need to interact without compromising sensitive information[15].

Simultaneously, ensuring secure access to such sensitive health data remains a significant challenge. Blockchain-based access control mechanisms have shown promise in this regard, particularly those that implement Role-Based Access Control (RBAC) models through smart contracts [5]. This decentralized model allows dynamic and tamper-proof permission management, effectively eliminating reliance on centralized authorities. By codifying role definitions and access policies into blockchain smart contracts, the system ensures that only authorized individuals or systems can access particular types of data, which is crucial in health applications where privacy and trust are paramount.

Chapter 3

Project Design

The design phase of the AI Fitness Trainer (AIFT) project plays a crucial role in transforming conceptual ideas into a structured and functional architecture. This chapter outlines the systematic approach taken to develop the core modules, interactions, and data flows within AIFT. The design focuses on ensuring modularity, scalability, and efficiency across both the backend and frontend components, while supporting real-time feedback and intelligent personalization. Each module has been carefully planned to integrate seamlessly—enabling the system to dynamically recommend personalized workout routines, track exercise form, generate nutrition plans, and deliver voice-assisted guidance. Emphasis has been placed on optimizing user experience, reducing latency, and maintaining accuracy throughout the user journey, from data input to actionable fitness insights. The following sections break down the key architectural elements, system workflow, data handling strategies, and integration logic that together form the foundation of this AI-powered fitness ecosystem.

3.1 Existing System

This section discusses existing research and systems in AI-enabled pose estimation, intelligent workout and diet recommendation systems, and blockchain-based subscription management. The reviewed literature provides significant insights into how each of these technologies can enhance user experience, accuracy, security, and personalization in fitness applications. These works served as valuable foundations in shaping our system design. Our approach draws inspiration from proven methodologies while addressing key limitations such as lack of adaptability, runtime inefficiencies, and data privacy challenges. The convergence of these three domains—pose estimation, recommendation systems, and blockchain—offers a comprehensive pathway toward a fully intelligent and user-centric fitness solution.

AI-based pose estimation techniques are increasingly becoming the backbone of interactive fitness platforms, primarily because they allow systems to track user movements in real time and offer corrective feedback. The integration of libraries such as MediaPipe and OpenCV has enabled low-latency pose detection, often achieving inference speeds of under 10 milliseconds per frame on edge devices. This real-time capability ensures privacy as data does not need to be uploaded to cloud servers. Nevertheless, these solutions often falter

when tracking complex or fast-paced dynamic movements, such as those involved in high-intensity interval training (HIIT) or compound weightlifting. To overcome these challenges, probabilistic joint motion tracking has shown promise in capturing intricate human motion more accurately, especially in controlled settings. Additionally, autoencoders have been successfully applied in posture correction scenarios, such as squat assessments, by learning the correct form and flagging deviations. However, scalability of such models to cover a wide range of exercises and body types remains a technical bottleneck. mmWave radar-based posture estimation is also emerging as a novel technique, offering precise skeletal tracking without requiring cameras, but the expensive hardware setup limits its feasibility for large-scale consumer adoption.

On the recommendation side, hybrid systems that combine collaborative filtering (based on user similarities) and content-based filtering (based on item features) are now commonly used for crafting personalized workout and diet plans. Deep learning models, particularly multi-layer perceptrons (MLPs), have improved recommendation accuracy by modeling non-linear relationships between user goals, health attributes, and preferences. However, these models are often computationally intensive, which can delay response time and hinder their deployment in real-time fitness apps. To address these constraints, deep associative neural networks (DANs) have been introduced, which offer more flexible and dynamic updates as user behaviors and targets evolve. Moreover, integration of Internet of Medical Things (IoMT) with machine learning enhances recommendation accuracy by feeding real-world physiological data—such as heart rate, calorie expenditure, or sleep cycles—into the system. Despite this, challenges persist in maintaining adaptability to user context and ensuring system responsiveness under constrained device environments.

Blockchain technology is increasingly being explored as a secure solution for managing sensitive data and subscription models in fitness platforms. Blockchain ensures that all user interactions, workouts, payments, and health records are stored with tamper-proof integrity. Subscription models such as BlockSubPay have shown how blockchain can facilitate decentralized, transparent, and fraud-resistant access control. However, these systems also face significant latency during transaction processing, which can be a drawback for real-time systems. Smart contracts—programmable logic layers on the blockchain—add another layer of functionality by enforcing permission-based data access and automating workflows. They ensure that users' personal fitness data is only accessible by authorized services or professionals. In addition, smart contracts can automatically resolve disputes and detect fraudulent activity, increasing the trustworthiness of the platform. Future enhancements in consensus mechanisms and off-chain computation are crucial for making blockchain systems viable for responsive and scalable fitness applications.

In summary, the convergence of AI-enabled pose estimation, intelligent recommendation algorithms, and blockchain-based subscription systems marks a major step forward in developing smart fitness platforms. Each technology contributes uniquely—whether in posture correction, personalized guidance, or secure data handling—while their integration offers a holistic solution to the shortcomings of traditional fitness systems. By leveraging these strengths and addressing common limitations like latency, scalability, and adaptability, our AI Fitness Trainer delivers a robust, real-time, and user-focused experience that sets the stage for the next generation of digital fitness applications.

3.2 Proposed System

The proposed system architecture for the Data-Driven AI Fitness Trainer is designed to provide a comprehensive fitness management solution through a mobile application. The core of the system is the AI Fitness Trainer, which handles user identification, exercise detection, and real-time feedback. It uses MTCNN and FaceNet for face recognition, and MediaPipe with CVzone for pose estimation. A CNN model supports real-time exercise detection, enabling personalized and effective workout guidance.

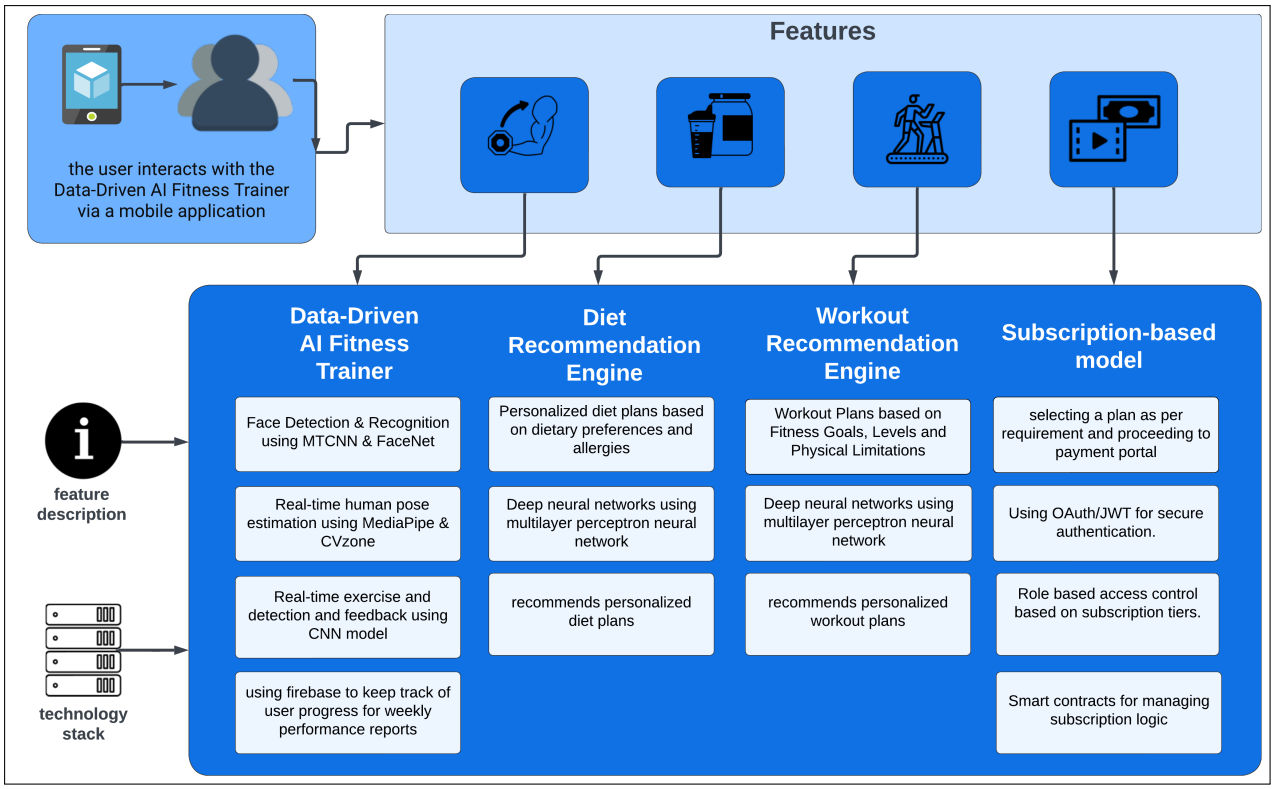

Figure 3.1: Proposed system architecture

The system's architecture as demonstrated in Figure 3.1 is divided into four principal components, each serving distinct yet interconnected functions. The Data-Driven AI Fitness Trainer module leverages MTCNN and FaceNet for precise user identification and authentication, creating unique facial embeddings for secure access control. For movement analysis, the system implements MediaPipe and CVZone technologies that work in tandem to perform real-time human pose estimation. These technologies identify and track key body landmarks and calculate joint angles with remarkable precision, enabling the system to understand complex human movements. The CNN-based exercise detection model analyzes these captured movements, compares them against reference exercise patterns, and provides immediate feedback on form, technique, and repetition counting. All user data flows into a Firebase database, creating a comprehensive repository of workout history, performance metrics, and improvement trends that feeds into weekly performance reports tailored to each user's progress.

Working alongside the core trainer module, the Diet Recommendation Engine applies deep neural networks with multilayer perceptron architecture to generate highly personal-

ized nutrition guidance. This engine processes complex inputs including dietary preferences, allergies, nutritional requirements, and fitness objectives to formulate meal plans that complement the user's workout regimen. Each recommendation is dynamically adjusted based on ongoing progress and changing user preferences, creating an adaptive nutrition strategy that evolves with the user's fitness journey.

The Workout Recommendation Engine employs parallel deep neural network architecture to develop exercise programs uniquely suited to individual users. This sophisticated component considers a multitude of factors including current fitness level, specific fitness goals, physical limitations, available equipment, and even environmental constraints to generate optimized workout routines. The system continuously refines these recommendations by analyzing performance data, adapting to user progress, and recalibrating intensity levels to maintain an optimal challenge-to-capability ratio that maximizes fitness development while minimizing injury risk.

The business layer of the application is managed through a comprehensive Subscription-based Model that implements several technical safeguards and user management systems. This includes a secure authentication protocol using OAuth/JWT standards, role-based access control mechanisms that regulate feature availability according to subscription tier, and smart contract integration for transparent subscription management. The payment processing system enables users to select plans based on their specific requirements, with a streamlined pathway to the payment portal for frictionless transactions.

The entire system represents a remarkable convergence of computer vision, artificial intelligence, cloud computing, and mobile technology, creating an interactive fitness experience that provides the benefits of personal training through digital means. The real-time nature of the feedback system—showing progress bars, countdown timers, repetition targets, and form corrections—transforms static exercise routines into dynamic, guided sessions that adapt to the user's performance. This comprehensive architecture fundamentally reimagines fitness training by harnessing cutting-edge technology to deliver personalized, accessible, and effective physical development programs that evolve with each user's unique fitness journey.

3.2.1 Critical Components of System Architecture

The overall workflow, along with interactions between major components of the system, is displayed here: Data-Driven AI Fitness Trainer, Workout Recommendation Engine, Diet Recommendation Engine, and Subscription-based Model. These modules are designed to offer real-time feedback, deliver personalized workout and dietary recommendations, and provide secure subscription management that defines the AI Fitness Trainer's capabilities.

A. Data-Driven AI Fitness Trainer

The Data-Driven AI Fitness Trainer is a core module for real-time tracking, ratings, and recommendations during workouts, which integrates computer vision and deep learning techniques to give efficient feedback and assistance.

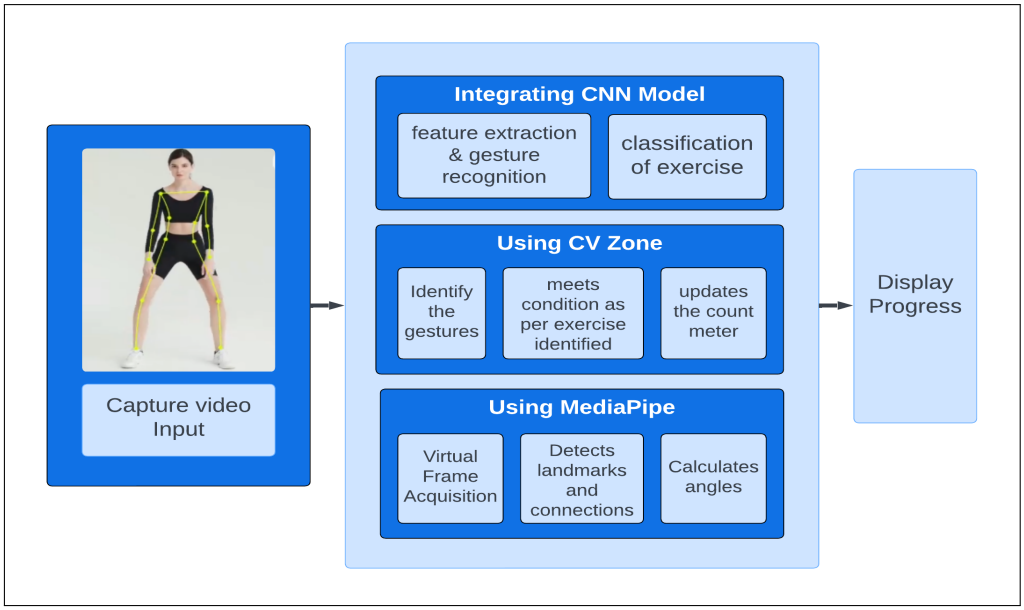

Figure 3.2: Working of Data-driven AI Fitness Trainer

The architecture of the proposed AI Fitness Trainer module demonstrated in Figure 3.2, integrates various components to provide real-time feedback and progress tracking for users. The system begins with video input capture, recording the user's exercise movements. Using MediaPipe, the system acquires a virtual frame, detects key body landmarks, and calculates joint angles essential for pose estimation. These computed features are then processed by CV Zone, which identifies the gestures, evaluates them against predefined conditions for specific exercises, and updates the repetition count accordingly.

A Convolutional Neural Network (CNN) is integrated for feature extraction and gesture recognition, enabling precise classification of exercise types to enhance accuracy. The combined outcome is displayed as real-time progress, offering users an intuitive and interactive interface for monitoring their performance. The facial anchor points are identified through the use of MTCNN, with FaceNet manifesting each authentication-based user with unique embeddings. Further, a firebase database holds user-specific data, including exercise logs, repetition counts, and calorie estimates, which then aggregate into weekly performance reports for tracking progress. Figure 3.3(c) focuses on real-time workout tracking, with a progress bar, a countdown timer for the current set, and target metrics (e.g., "20 reps"), creating an interactive and guided fitness experience.

B. Workout Recommendation Engine

The Workout Recommendation Engine generates personalized workout plans for users based on their body metrics, levels of fitness, goals, and health conditions. Figure 3.3(a) shows the input form for user details while Figure 3.3(b) shows the daily workout plan with selectable difficulty levels and exercise visuals.

Machine learning with MLP (multilayer perceptron) networks runs an input through a number of hidden layers with ReLU activation functions for learning complex interactions between user parameters for precise recommendations. Regularization techniques like dropout

Figure 3.3: AI Fitness Trainer Mobile Interface

and batch normalization provide robust performance and minimal overfitting. The model is trained on a large user profile and workout plan dataset, and these use categorical cross-entropy loss and the Adam optimization algorithm to minimize weights. The output layer presents structured workout plans that cover strength, cardio, and flexibility exercises.

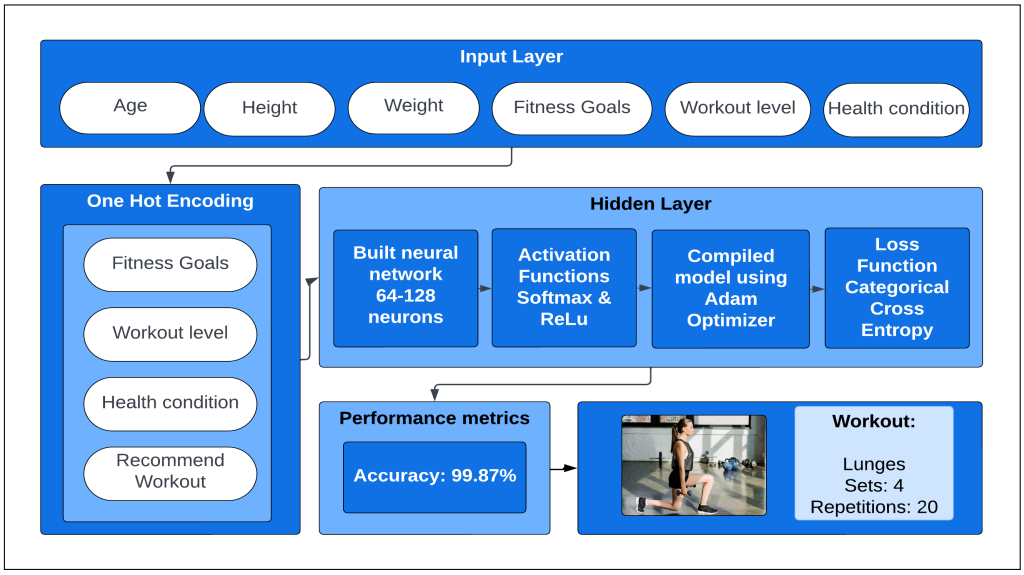

Figure 3.4: Working of Workout Recommendation Engine

Figure 3.4 illustrates the architecture of the recommendation engine used for generating personalized workout plans based on user inputs. The input layer collects key user attributes, including age, height, weight, fitness goals, workout level, and health condition. To handle categorical variables like fitness goals, workout level, and health condition, One Hot Encoding is applied which enables numerical representation for effective processing. These encoded features are fed into a neural network built with 64–128 neurons in the hidden layer.

The network utilizes Softmax and ReLU as activation functions to ensure non-linearity and probabilistic outputs. The model is compiled using the Adam Optimizer, and the loss is computed with Categorical Cross Entropy to optimize multi-class classification. The model's performance is evaluated using metrics such as accuracy, achieving an exceptional score of 99.87%. The output layer recommends tailored workouts, as the figure shows, suggesting lunges with specific sets and repetitions. This framework ensures personalised and accurate workout recommendations, enhancing user engagement and fitness outcomes.

For instance, a beginner with knee problems looking to lose weight would receive a plan with low-impact exercises, while an advanced user focused on muscle gain would receive a tough regimen. The real-time user feedback provides a chance for periodic dynamic updating of plans, thus ensuring truly personalized and evolving fitness solutions.

C. Diet Recommendation Engine

The Diet-Recommendation Engine applies the deep learning principle, especially MLP neural networks, to create personalized dietary plans targeted to user parameters that include age, weight, preferences, allergies, and aims toward fitness. The engine works through a pipeline from input collection and normalizing all the way to feature processing, wherein inputs of a user refer to age and weight for caloric needs, while his preferences on being a vegan or vegetarian and allergies will work to choose the meal offerings.

Input features are represented in an MLP architecture such that the categorical data is one-hot encoded, while continuous data is standardized. Features passes through hidden layers with ReLU activation, giving its power to find complex patterns. And some regularization strategies are there to enforce the dropout and batch normalization, thus providing further robustness for the model. In the output layer are predicted meals, portion sizes, and nutrient distributions according to the user's inputs.

The engine is trained with a variety of datasets using Mean Squared Error, and the Adam optimizer for the new users that undergo cross-validation for ensuring accuracy. The system is also capable of providing dynamically changing meal plans according to the user's progress, providing a customized approach to nutrition planning through an MPL neural network.

D. Progress Tracker & Subscription-based Model

Our project includes a built-in Progress Tracker that plays a crucial role in monitoring development milestones and ensuring timely completion of tasks. It provides a clear visual representation of the project's current status, making it easier to assess how much work has been done and what remains. This feature has been especially useful during team collaborations, as it allows everyone to stay aligned and focused on their respective responsibilities.

The Progress Tracker has helped streamline the workflow by breaking the project into manageable phases and tracking each module's completion in real time. It also facilitates better planning, quicker identification of bottlenecks, and efficient allocation of resources. By keeping a transparent log of progress, it has contributed significantly to the smooth execution and accountability throughout the development cycle.

Our subscription-based model is meant to offer flexible access to premium features with the Basic plan catering to the varying user needs. The basic plan subscription offers a lot of features, including AI fitness trainer, diet recommendations, weekly report generation, workout catalog, and role-defined access to the platform. The Basic Plan is priced at Rs. 199/month and is ideal for individuals looking for a focused, cost-effective fitness solution with access for 1 user. Authentication for the system is handled using OAuth for secure user logins, and JWT (JSON Web Tokens) to manage session roles. Each JWT includes role-based claims (basic or premium), which the backend verifies to grant or deny access to specific features dynamically.

The payment system uses platforms like Stripe to handle subscriptions securely. These gateways are configured for recurring billing, with APIs that process payments and update the user's subscription details. Upon payment confirmation, the backend assigns the premium role to the user and stores their subscription expiry date in the database. To enhance transparency, blockchain-based smart contracts can be integrated. The role-based access controls as per their subscription and smart contracts automate subscription management by recording the user's subscription tier, payment status, and expiry date securely on the blockchain.

3.3 System Diagrams

3.3.1 UML Diagram

The design and functionality of the AI Fitness Trainer (AIFT) system are best understood through the use of Unified Modeling Language (UML) diagrams. These diagrams provide a detailed visual representation of how the system components operate and interact with users and other modules. They serve as a blueprint for understanding the logic, behavior, and structure of the application. UML diagrams not only help in system design but also assist in communication between developers, stakeholders, and project teams.

To offer a comprehensive overview, this section includes three types of UML diagrams: the Activity Diagram, which outlines the flow of actions performed during system operations; the Sequence Diagram, which shows how objects interact in a particular time sequence; and the Use Case Diagram, which identifies the system's various functions and the actors involved. Together, these diagrams illustrate the internal working and external interaction of the AIFT system, supporting a clearer understanding of its overall architecture.

3.3.2 Activity Diagram

The Activity Diagram shown in Figure 3.5 provides a comprehensive overview of the user journey and the logical flow of activities across various modules of the AI Fitness Trainer app. When a user opens the app, they are prompted to either sign in or sign up. Upon successful login, the user enters the home screen where they can choose between different

options: using the AI fitness trainer, requesting a diet plan, receiving a workout schedule, subscribing to premium plans, or generating a weekly report. Each option leads the user through a structured path of input collection, AI processing, and result presentation.

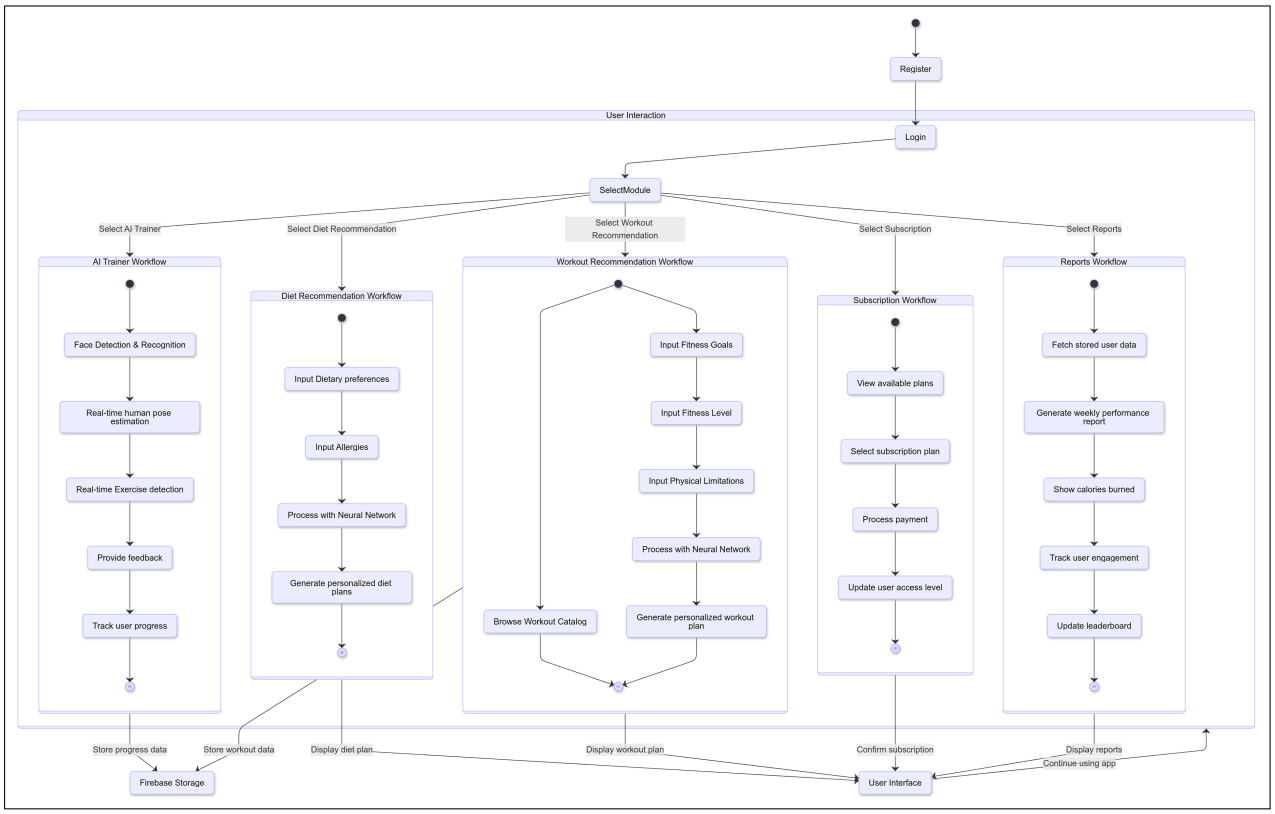

Figure 3.5: Activity Diagram

In the AI Fitness Trainer flow, the app activates the mobile device's camera and begins face recognition using MTCNN, followed by pose estimation using MediaPipe. The detected pose is fed into a CNN-based model which identifies the exercise being performed. Based on real-time feedback, the system may suggest posture corrections, count repetitions, or display motivation cues. This information is continuously logged to Firebase for tracking progress. The Diet Recommendation module requires the user to fill in personal and health-related inputs. These inputs are passed to a backend neural model which generates a personalized meal plan that aligns with the user's fitness goals and dietary constraints. Likewise, the Workout Recommendation module functions in a similar fashion, collecting data on the user's fitness level and suggesting relevant workout routines. The subscription activity allows the user to browse plans, make payments, and gain access to premium features. Lastly, the Weekly Report activity collates all data from Firebase, processes the statistics, and presents the user with a performance summary and their rank on the leaderboard. This diagram effectively visualizes the modularity and logical progression of the app's user interaction model.

3.3.3 Use Case Diagram

The Use Case Analysis chapter provides a comprehensive overview of the user interactions and functional requirements of the Data-Driven AI Fitness Trainer application. The User interacts with the system through a mobile application, which facilitates a range of functionalities designed to offer a personalized fitness experience.

The key use cases, as demonstrated in the figure 3.6 include Registration, where the user begins by registering on the application and providing necessary details to create a personalized profile, forming the basis for tailored recommendations and progress tracking. The Face Detection & Recognition use case utilizes MTCNN and FaceNet technologies to identify and verify the user, ensuring that personalized data and recommendations are securely linked to the correct individual. Real-time Human Pose Estimation employs CVZone and MediaPipe to estimate the user's body pose in real time, a crucial feature for tracking exercises and ensuring proper form, which is essential for effective workouts and injury prevention.

Additionally, Real-time Exercise Detection and Feedback uses a CNN model to detect the specific exercises performed by the user, providing immediate feedback on form and performance to help users correct their technique and maximize workout efficiency. The system also incorporates Tracking User Progress, storing workout data in Firebase to generate weekly reports that give users insights into their progress, areas for improvement, and overall fitness journey. Based on the user's dietary preferences and allergies, the Diet Recommendation feature employs a deep neural network to suggest personalized diet plans that align with the user's specific needs and goals.

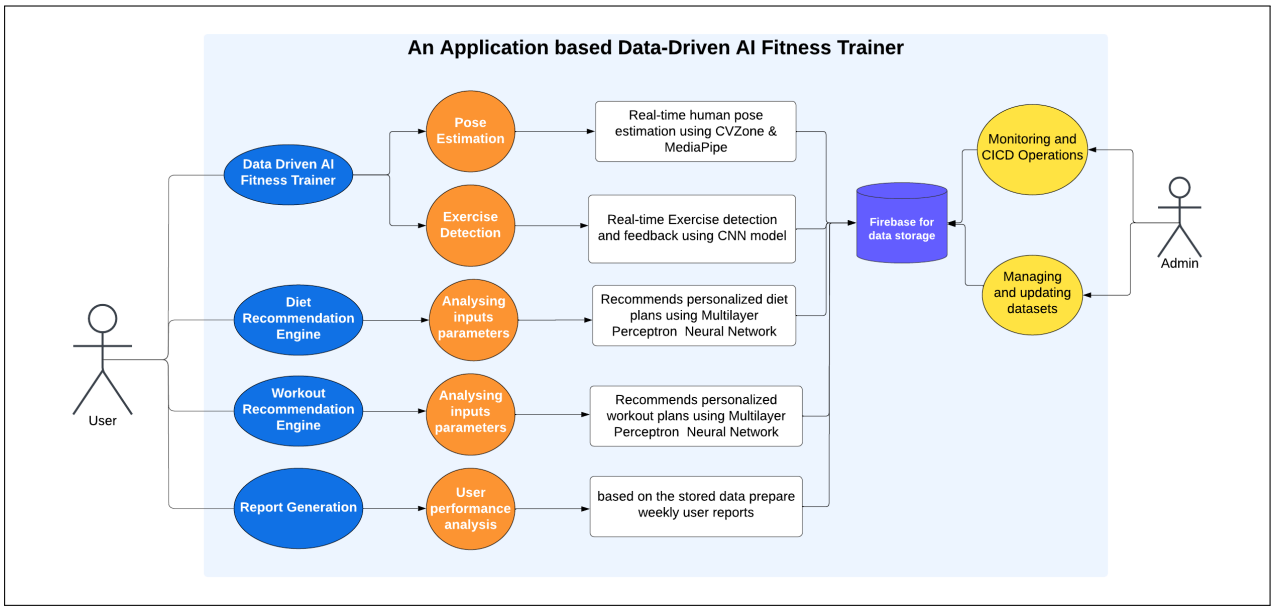

Figure 3.6: Use Case Diagram

Furthermore, the system provides Workout Recommendations tailored to the user's fitness goals, level, and any physical limitations, leveraging deep neural networks to align the user's input with suitable workout modules. The Subscription Management use case allows users to select and subscribe to different plans according to their requirements, with the system processing payments through a secure payment gateway and using smart contracts

to update user access based on their subscription plan. Lastly, Report Generation and Leader board features enable the system to generate detailed weekly reports highlighting exercises performed, calories burned, and user engagement levels, along with a leader board that updates based on daily tasks completed and overall user activity, fostering a sense of competition and motivation among users.

Each of these use cases is interconnected, forming a comprehensive fitness management system that not only guides users through their workouts but also provides dietary advice and progress tracking. The integration of advanced technologies like MTCNN, FaceNet, CV-Zone, MediaPipe, and deep neural networks ensures that the system is robust, accurate, and capable of delivering a highly personalized fitness experience. In conclusion, the Use Case Analysis chapter underscores the versatility and functionality of the Data-Driven AI Fitness Trainer, highlighting how the system leverages cutting-edge technology to offer a holistic approach to fitness, encompassing exercise, diet, and user engagement through a seamlessly integrated mobile application.

3.3.4 Sequence Diagram

The Sequence Diagram illustrated in Figure 3.7 showcases the detailed dynamic behavior of the AI Fitness Trainer system, emphasizing how various system components interact sequentially with the user's actions. The process initiates when a user either registers a new account or logs in using Firebase Authentication. This ensures that each user has a secure, personalized experience within the application. The authentication request is sent from the mobile app to Firebase, which verifies the credentials and returns the authentication status. Once authenticated, the user data is stored in the Firebase Database, establishing a profile that includes personal details, preferences, and historical fitness records. This authentication and data storage step is crucial, as it enables personalized features and controls the user's access to specific modules and services offered by the system.

Upon successful login, the user can navigate through different modules available in the app, one of which is the exercise session module. When a fitness session is started, the system activates the device's camera to capture live video of the user performing exercises. To ensure the session is accurately personalized and monitored, the system uses MTCNN (Multi-task Cascaded Convolutional Neural Networks) and FaceNet for real-time face detection and verification. These models verify the identity of the user by analyzing the facial features captured in the video feed. Once the face is recognized, the system proceeds with pose estimation using MediaPipe or CVZone, which tracks specific body landmarks and skeletal movements. This allows the AI system to understand the user's posture, alignment, and form throughout the workout.

The captured pose data is processed through a Convolutional Neural Network (CNN) model, specially trained to detect the type of exercise being performed based on body movements and joint positions. This AI-based detection enables the system to classify exercises such as squats, push-ups, or lunges with high accuracy. Based on the exercise recognized and

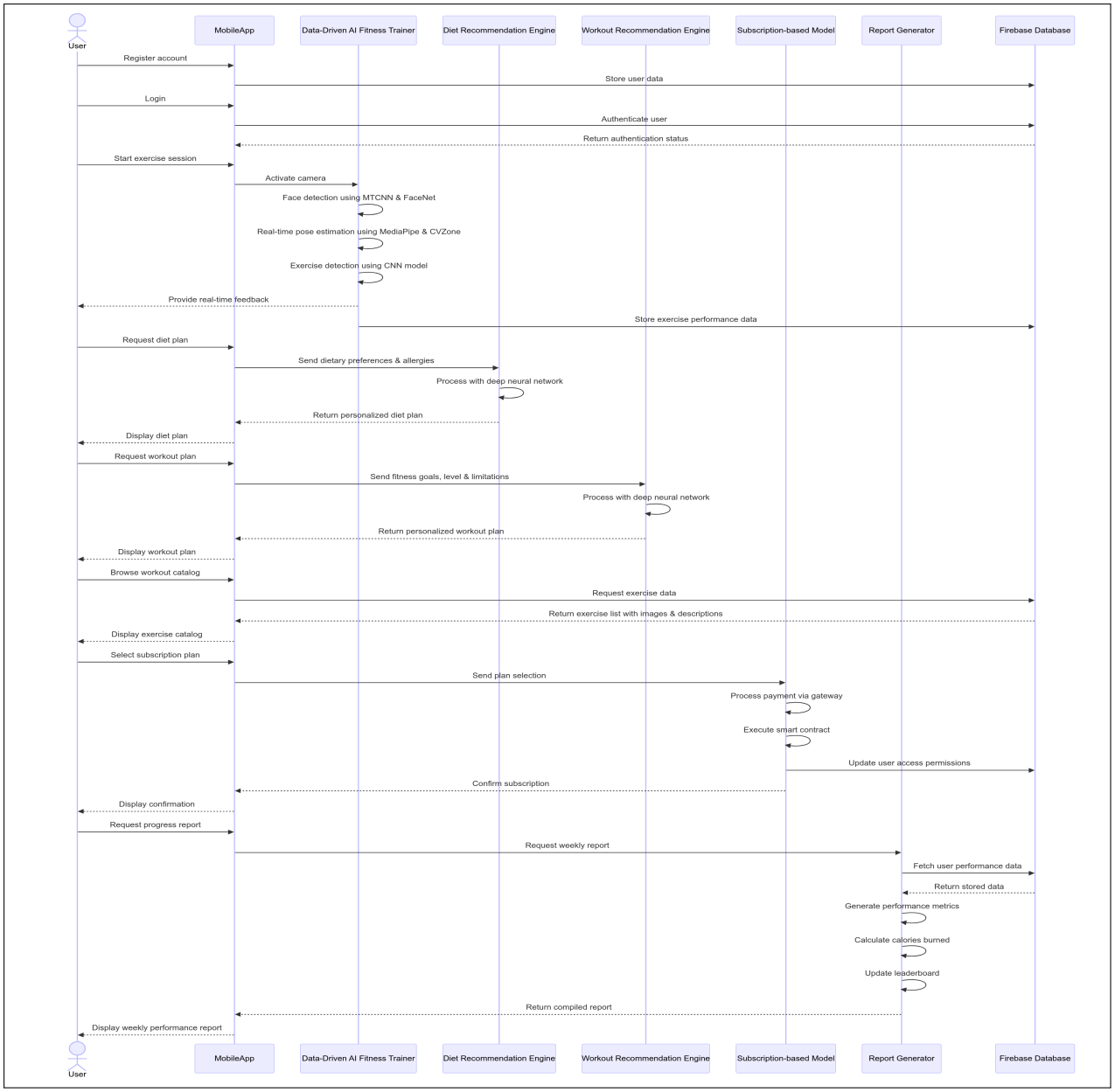

Figure 3.7: Sequence Diagram

the user's posture, the system provides real-time feedback through the mobile app, guiding the user to adjust their form, increase intensity, or maintain consistency. Simultaneously, exercise performance data such as reps, sets, and duration is continuously updated in the Firebase Database, ensuring that a detailed log of the user's fitness activities is maintained for further analysis and reporting.

In addition to workouts, the app features Diet and Workout Recommendation modules that offer users tailored fitness and nutrition guidance. When users input their dietary preferences, allergies, age, weight, and fitness goals, this data is transmitted to the Diet Recommendation Engine. This engine, powered by a deep neural network, processes the input and generates a personalized meal plan designed to align with the user's health goals and nutritional requirements. The same approach is applied within the Workout Recommenda-

tion Engine, where the user's fitness level, physical limitations, and objectives are taken into account. Using another deep neural network, the engine recommends a customized workout plan, detailing suitable exercises, their descriptions, images, and suggested durations. These AI-generated plans are then stored in Firebase and displayed in the app's UI for the user to follow.

To manage access to these premium features, the system integrates a subscription-based model. Users can browse various subscription plans within the app, selecting one based on their preferences, budget, and required features. Once a plan is selected, the system processes the payment through a secure gateway and executes a smart contract to automatically update the user's access permissions in Firebase. This ensures that users gain access to newly unlocked features, such as advanced workout plans, detailed nutrition reports, and AI-driven progress analysis. After successful payment, the system confirms the subscription and updates the app interface, displaying the available premium modules based on the new access level. An essential part of maintaining user motivation and tracking progress is the weekly performance report module. Users can request a weekly progress report through the mobile app, triggering the Report Generator module to retrieve the latest performance data from Firebase. This includes exercise logs, calories burned, workout frequency, and session durations. Additionally, the system updates leaderboards to foster a sense of competition and motivation among users. The compiled report is sent back to the mobile app, giving users a comprehensive overview of their fitness journey over the past week.

In conclusion, the Sequence Diagram effectively captures the real-time, AI-driven interactions between the user, mobile app, backend services, AI models, and Firebase Database. It highlights how various technologies — including MTCNN, FaceNet, MediaPipe, CVZone, CNN models, and deep neural networks — work together to deliver a personalized, data-driven fitness experience. From authentication and live feedback during workouts to diet and workout recommendations, subscription handling, and performance analysis, every interaction is designed to offer a seamless and engaging experience. The diagram illustrates how data flows and processes are sequenced logically, ensuring that user actions trigger intelligent, responsive, and personalized services that evolve with their fitness journey.

Chapter 4

Project Implementation

This chapter provides a comprehensive overview of the implementation phase of the project, detailing the technical execution and development process. It includes key code snippets that illustrate core functionalities, step-by-step instructions for accessing and interacting with the system, and a timeline that outlines the progression of development milestones. This chapter aims to connect the conceptual framework with its practical execution, illustrating how the proposed design was methodically developed into a working system.

4.1 Code Snippets

To better understand the implementation of the AI Fitness Trainer system, this section includes important code snippets that highlight key components of the project. These examples demonstrate how various modules—such as personalized workout and diet plan generation using multi-layer perceptron, real-time pose detection, exercise classification and counter using MediaPipe and Cvzone, and the voice assistant functionality—are developed and integrated to work together. Each snippet is carefully selected to show the core logic and functionality behind the personalized and interactive fitness experience offered by the system.

This code illustrated in Figure 4.1 is designed to train a multi-output machine learning model that recommends customized workout plans based on user attributes. It starts by encoding categorical features using LabelEncoder, which converts non-numeric fields such as fitness level and health conditions into machine-readable form for training.

Target variables—namely recommended workouts, sets, and repetitions—are split and processed into individual binary flags or numerical values. For example, each workout in the recommendation list becomes a separate column indicating whether that workout is part of the recommendation (1) or not (0). This transforms the problem into a multi-label classification task.

Next, the feature matrix (X) and target matrix (y-workouts) are prepared, and the data is split into training and testing sets. The model used is a MultiOutputClassifier that wraps

```python
# Replace nan with 'None' string in health_conditions
df['health_conditions'] = df['health_conditions'].fillna('None')

# 2. Encoding categorical variables
le_dict = {}
categorical_cols = ['fitness_goals', 'fitness_level', 'health_conditions',
                    'Joint Issues', 'Daily Activity Type', 'Favorite Workout Types',
                    'Fatigue Level']

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    le_dict[col] = le

# 3. Process target variables
df['recommended_workout'] = df['recommended_workout'].str.split(', ')
df['recommended_sets'] = df['recommended_sets'].str.split(', ')
df['recommended_repetitions'] = df['recommended_repetitions'].str.split(', ')

all_workouts = ['Squats', 'Lunges', 'Step-ups', 'Pushups', 'Glute Bridge',
                'Arm Circles', 'Calf Raises', 'Hammer Curl', 'Tricep Dips',
                'Bicycle Crunches', 'Russian Twist', 'Leg Raises', 'Plank', 'Crunches']

for workout in all_workouts:
    df[f'workout_{workout}'] = df['recommended_workout'].apply(lambda x: 1 if workout in x else 0)

df['recommended_sets'] = df['recommended_sets'].apply(lambda x: [int(i) for i in x])
df['recommended_repetitions'] = df['recommended_repetitions'].apply(lambda x: [int(i) for i in x])

# 4. Prepare feature and target matrices
X = df[features]
y_workouts = df[[f'workout_{w}' for w in all_workouts]]

# 5. Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y_workouts, test_size=0.2, random_state=42)

# 6. Train the model with MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(100, 50), max_iter=500, random_state=42,
                    activation='relu', solver='adam', learning_rate_init=0.001)
multi_target_model = MultiOutputClassifier(mlp, n_jobs=-1)
multi_target_model.fit(X_train, y_train)

```

Figure 4.1: MLP model for predicting customized workout plans from user data

around a MLPClassifier (Multi-Layer Perceptron). The MLP is configured with hidden layers, ReLU activation, and an Adam optimizer, enabling it to learn complex patterns in the data.

A custom function recommend-workout() is also defined, which accepts user data such as age, height, weight, fitness goals, and health conditions. It computes derived attributes like BMI and preferred workout duration based on fitness level. These processed inputs are then used for making predictions via the trained model.

This code mentioned in figure 4.2 sets up a backend service using FastAPI for generating personalized workout recommendations. It begins by loading a pre-trained machine learning model (workout-recommendation-mlp-model.pkl) and associated label encoders (label-encoders-mlp.pkl), which are essential for processing categorical features like fitness goals, levels, and health conditions. These encoders help translate human-readable inputs into machine-readable formats that the model was trained on.

The /predict endpoint accepts a POST request containing user information such as age, weight, height, fitness goal, fitness level, and any health conditions. The categorical inputs are first encoded using the label encoders. The input is then formatted as a 13-feature vector (with padding zeros for missing fields) to match the model's input shape. The model predicts a suitable workout plan based on this input, which is returned as a JSON response.

workout_recommendation_api.py

functions > workout_recommendation_api.py > ...

```python
# filepath: c:\Users\sawan\Documents\FlutterApps\ai_fitness_trainer\functions\app.py
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import pickle
import os

# Initialize FastAPI app
app = FastAPI()

import os

# Get the directory of the current script
current_dir = os.path.dirname(os.path.abspath(__file__))

# Load the model and label encoders
with open(os.path.join(current_dir, 'workout_recommendation_mlp_model.pkl'), 'rb') as model_file:
    model = pickle.load(model_file)

with open(os.path.join(current_dir, 'label_encoders_mlp.pkl'), 'rb') as encoders_file:
        label_encoders = pickle.load(encoders_file)
        print(label_encoders.keys())  # Add this line to debug


'''# Load the model and label encoders
with open('workout_recommendation_mlp_model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

with open('label_encoders_mlp.pkl', 'rb') as encoders_file:
    label_encoders = pickle.load(encoders_file)'''

# Define the request body schema
class WorkoutRequest(BaseModel):
    age: int
    weight: float
    height: float
    fitness_goal: str
    fitness_level: str
    health_condition: str

# Define the predict endpoint
@app.post("/predict")
async def predict(request: WorkoutRequest):
    try:
        # Extract input features
        age = request.age
        weight = request.weight
        height = request.height
        fitness_goal = request.fitness_goal
        fitness_level = request.fitness_level
        health_condition = request.health_condition

        # Map the keys to match the existing label_encoders keys
        fitness_goal_encoded = label_encoders['fitness_goals'].transform([fitness_goal])[0]
        fitness_level_encoded = label_encoders['fitness_level'].transform([fitness_level])[0]
        health_condition_encoded = label_encoders['health_conditions'].transform([health_condition])[0]

        # Prepare input for the model (pad with zeros to match 13 features)
        input_features = [[
            age, weight, height, fitness_goal_encoded, fitness_level_encoded, health_condition_encoded,
            0, 0, 0, 0, 0, 0, 0  # Add zeros for missing features
        ]]

        # Predict workout plan
        prediction = model.predict(input_features)

        # Since 'workout_plan' key is missing, return the raw prediction as a temporary fix
        workout_plan = prediction[0]  # Assuming the model returns a string or label directly

        return {"workout_plan": workout_plan}
    except Exception as e:
        raise HTTPException(status_code=400, detail=str(e))
```

Figure 4.2: FastAPI backend for real-time personalized workout recommendation

This FastAPI microservice makes real-time workout recommendation integration seamless and efficient for the AI Fitness Trainer application.

extract_landmarks.py

```python
import cv2
import mediapipe as mp
import numpy as np
import os
import pandas as pd

mp_pose = mp.solutions.pose
pose = mp_pose.Pose()

DATA_PATH = "D:\\RUTUJA\\PROJECTS\\Major Project\\virtual trainer\\outputs"
train_data_path = os.path.join(DATA_PATH, "train")

landmarks_list = []
labels_list = []

def extract_landmarks(image_path, label):
    image = cv2.imread(image_path)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image_rgb)

    if results.pose_landmarks:
        landmarks = [lm.x for lm in results.pose_landmarks.landmark] + \
                    [lm.y for lm in results.pose_landmarks.landmark] + \
                    [lm.z for lm in results.pose_landmarks.landmark]

        landmarks_list.append(landmarks)
        labels_list.append(label)

# Loop through dataset
for exercise in os.listdir(train_data_path):
    exercise_path = os.path.join(train_data_path, exercise)

    for img_file in os.listdir(exercise_path):
        img_path = os.path.join(exercise_path, img_file)
        extract_landmarks(img_path, exercise)

# Convert to DataFrame and save
df = pd.DataFrame(landmarks_list)
df["label"] = labels_list
df.to_csv("exercise_landmarks.csv", index=False)

print("Landmark extraction completed!")
```

Figure 4.3: Pose landmark extraction using MediaPipe for exercise classification

This code in figure 4.3 is used to extract pose landmarks from exercise images for training a pose classification model in the AI Fitness Trainer application. It utilizes MediaPipe Pose, an advanced pose estimation framework, to identify key body landmarks such as joints and limbs from each image.

The process begins by reading images from a labeled dataset structured by exercise type (e.g., squats, lunges, push-ups). Each image is converted to RGB and passed through MediaPipe's pose detection pipeline. If landmarks are detected, their x, y, and z coordinates are extracted and stored along with the corresponding exercise label.

The collected landmark data and labels are then organized into a Pandas DataFrame and saved as a CSV file named exercise-landmarks.csv. This CSV acts as a training dataset for downstream machine learning models used in real-time exercise recognition and correction features. Overall, this script forms the backbone of the data preparation process for the pose-based functionality of the AI Fitness Trainer system.

This code in figure 4.4 is responsible for training a deep learning model that classifies exercises based on body pose landmarks, a key component of the AI Fitness Trainer system. It starts by loading the dataset of pose landmarks and corresponding exercise labels generated

training_model.py

```python
import tensorflow as tf
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
import pickle

# Load dataset
df = pd.read_csv("exercise_landmarks.csv")

# Split features and labels
X = df.drop(columns=["label"]).values
y = df["label"].values

# Encode labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Normalize data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Save encoders
with open("label_encoder.pkl", "wb") as f:
    pickle.dump(label_encoder, f)

with open("scaler.pkl", "wb") as f:
    pickle.dump(scaler, f)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.2, random_state=42)

# Build model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(len(np.unique(y_encoded)), activation="softmax")
])

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# Train model
model.fit(X_train, y_train, epochs=50, batch_size=32, validation_data=(X_test, y_test))

# Save model
model.save("exercise_pose_model.h5")

print("Model training completed!")
```

Figure 4.4: Pose classification model trained on exercise landmarks using TensorFlow

in the previous step.

The labels are encoded using LabelEncoder to convert them into numerical form, and the landmark features are normalized using StandardScaler to ensure consistent input for the neural network. Both the encoder and scaler are saved using pickle so they can be reused during inference.

The dataset is split into training and testing sets to evaluate the model's performance. A simple feedforward neural network is built using TensorFlow, consisting of two hidden layers with ReLU activation and a softmax output layer for multi-class classification.

The model is compiled with the Adam optimizer and trained using sparse categorical cross-entropy loss. After 50 epochs of training, the model is saved as exercise-pose-model.h5 for future use in real-time pose classification. This enables the AI Fitness Trainer to accurately detect and differentiate between various exercises based on user posture.

This code shown in figure 4.5 represents the backend logic for the AI Fitness Trainer

app.py

functions > app.py > ...

```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import subprocess
import os
import pickle
import tensorflow as tf

app = Flask(__name__)
CORS(app)

# Load the necessary files
current_dir = os.path.dirname(os.path.abspath(__file__))

# Load the Label Encoder
with open(os.path.join(current_dir, 'label_encoder.pkl'), 'rb') as encoder_file:
    label_encoder = pickle.load(encoder_file)

# Load the Standard Scaler
with open(os.path.join(current_dir, 'scaler.pkl'), 'rb') as scaler_file:
    scaler = pickle.load(scaler_file)

# Load the Exercise Pose Model
model_path = os.path.join(current_dir, 'exercise_pose_model.h5')
exercise_model = tf.keras.models.load_model(model_path)

# Endpoint to start exercise tracking
@app.route('/start_exercise', methods=['POST'])
def start_exercise():
    try:
        # Parse the JSON request
        data = request.get_json()
        exercise = data.get('exercise')
        reps = data.get('reps')

        # Validate input
        if not exercise or not reps or not isinstance(reps, int) or reps <= 0:
            return jsonify({"error": "Invalid exercise or repetitions"}), 400

        # Log the request
        print(f"Starting exercise tracking for {exercise} with {reps} repetitions")

        # Start the exercise detection script
        subprocess.Popen([
            "python",
            os.path.join(current_dir, "exercise_detection.py"),
            exercise,
            str(reps)
        ])

        return jsonify({"message": f"Exercise tracking started for {exercise} with {reps} repetitions"}), 200

    except Exception as e:
        print(f"Error: {e}")
        return jsonify({"error": str(e)}), 500

# Endpoint to test the model (optional, for debugging)
@app.route('/test_model', methods=['POST'])
def test_model():
    try:
        # Parse the JSON request
        data = request.get_json()
        landmarks = data.get('landmarks')  # Expecting a list of body landmarks

        if not landmarks or not isinstance(landmarks, list):
            return jsonify({"error": "Invalid landmarks data"}), 400

        # Normalize the landmarks using the scaler
        normalized_landmarks = scaler.transform([landmarks])

        # Predict the exercise using the model
        prediction = exercise_model.predict(normalized_landmarks)
        predicted_class = label_encoder.inverse_transform([prediction.argmax()])[0]

        return jsonify({"predicted_exercise": predicted_class}), 200

    except Exception as e:
        print(f"Error: {e}")
        return jsonify({"error": str(e)}), 500

# Run the Flask app
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```

Figure 4.5: Flask backend for exercise classification and real-time tracking integration

application, developed using the Flask framework in Python. It acts as the communication bridge between the frontend interface and the underlying machine learning models responsible for exercise detection and tracking. The backend is also configured with CORS support to ensure smooth communication between the client and server, even when hosted on different origins.

At the core of the backend, several important components are loaded during initialization. These include a pre-trained TensorFlow model (exercise-pose-model.h5) for exercise classification, a standard scaler (scaler.pkl) for normalizing the landmark input features, and a label encoder (label-encoder.pkl) used to map numerical model outputs back to human-readable exercise names. These files are loaded using Python's pickle and TensorFlow libraries from the local project directory.

The /start-exercise endpoint handles POST requests containing the exercise name and number of repetitions. After validating the input, it launches the exercise-detection.py script as a sub process, which performs real-time pose detection and repetition counting. A confirmation response is then returned to indicate that tracking has started. The /test-model endpoint is mainly used for testing. It takes body landmarks in JSON format, normalizes them using a scaler, and sends them to a trained TensorFlow model for prediction. The result is decoded into a readable exercise label and returned. The Flask app runs on 0.0.0.0:8000, enabling real-time AI-based exercise tracking within the application.

```
# Perform Train-Test Split BEFORE Encoding Outputs
df_train, df_test, X_train, X_test = train_test_split(df, X, test_size=0.2, random_state=42, shuffle=True)

# MultiLabelBinarizer to encode meal recommendations
mlb = MultiLabelBinarizer()

y_train_breakfast = mlb.fit_transform(df_train['breakfast'])
y_train_lunch = mlb.transform(df_train['lunch'])
y_train_dinner = mlb.transform(df_train['dinner'])

y_test_breakfast = mlb.transform(df_test['breakfast'])
y_test_lunch = mlb.transform(df_test['lunch'])
y_test_dinner = mlb.transform(df_test['dinner'])

# Combine all meal outputs
y_train = np.hstack((y_train_breakfast, y_train_lunch, y_train_dinner))
y_test = np.hstack((y_test_breakfast, y_test_lunch, y_test_dinner))

# Define MLP Model
model = Sequential([
    Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.4),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dense(y_train.shape[1], activation='sigmoid')  # Multi-label classification
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['binary_accuracy', Precision(), Recall()])

# Train the model
history = model.fit(X_train, y_train, epochs=50, batch_size=32,
                    validation_data=(X_test, y_test))

# Save the model
model.save("diet_recommendation_mlp.h5")
```

Figure 4.6: MLP model for generating personalized multi-label meal recommendations

This code in Figure 4.6 builds a backend system for generating personalized meal recommendations using a multi-label classification approach. It starts by splitting the dataset into training and testing sets before encoding the output labels, which helps maintain data integrity and prevents information leakage during model training.

Using the MultiLabelBinarizer, the labels for each meal (breakfast, lunch, and dinner) are transformed into binary arrays, enabling the model to learn multiple outputs per input instance. These encoded labels are then combined horizontally to form a single multi-label output vector for each data sample.

The model is a Multi-Layer Perceptron (MLP) built with the Keras Sequential API. It consists of three dense layers with ReLU activation and dropout layers to reduce overfitting. The final layer uses a sigmoid activation function, which is suitable for multi-label classification tasks. The model is compiled with a binary cross-entropy loss function and is evaluated using metrics like binary accuracy, precision, and recall.

Finally, the model is trained on the user input features for 50 epochs with a batch size of 32, using the training data and validating on the test set. This setup enables the AI Fitness Trainer to make simultaneous meal recommendations across multiple categories in a scalable and efficient way.

4.2 Steps to access the System

This section explains the step-by-step process to access and use the AI Fitness Trainer system. It covers how users can interact with the mobile application, from launching the app to selecting features like workout tracking, diet recommendations, and subscription plans. The aim is to ensure that users can easily navigate and experience the full functionality of the system.

Open the App

- Locate the **AI Fitness Trainer** app on your device and tap to open it.

Log In or Sign Up

- If you are a new user:
  - Tap on **Sign Up** and create an account by entering your details (e.g., name, email, password).
- If you already have an account:
  - Enter your credentials and tap **Log In**.

Figure 4.7: Landing Page of AIFT

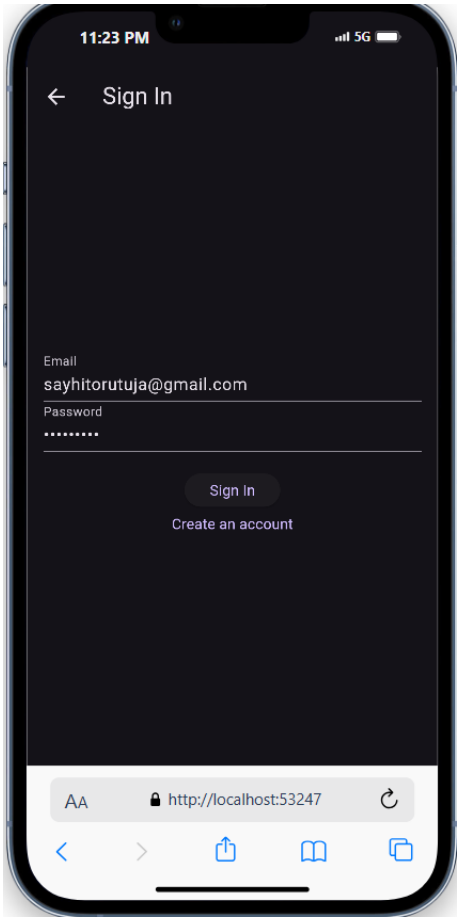

Figure 4.8: Login Page of AIFT

Access the Dashboard

After logging in, you will be taken to the **Dashboard**, where you can see all the features:

- Workout Recommendations
- Diet Recommendations
- AI Fitness Trainer
- Progress Tracker

Using the Features

Workout Recommendations

1. Tap on the **Workout Recommendations** option.
2. Enter your fitness preferences:

- Fitness goal (e.g., weight loss, muscle gain).
- Fitness level (e.g., beginner, intermediate, advanced).
- Available time for workouts.

3. View the recommended workout plan, which includes:
   - Exercises with sets, reps, and durations.
   - Video or animation guides for each exercise.
4. Start the workout by following the instructions.

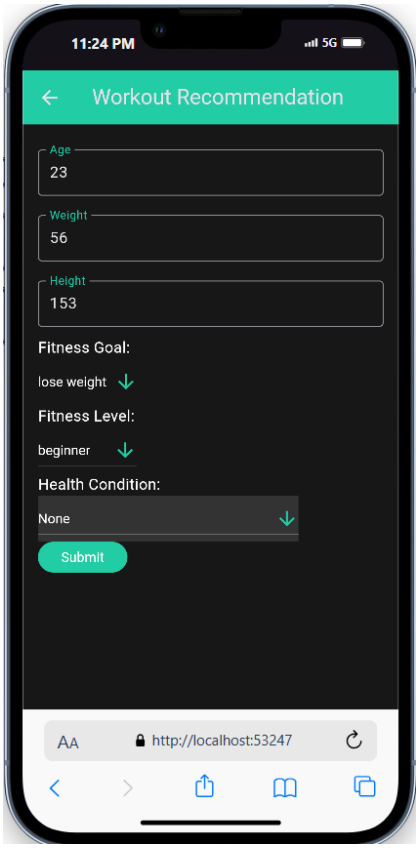

Figure 4.9: User input form

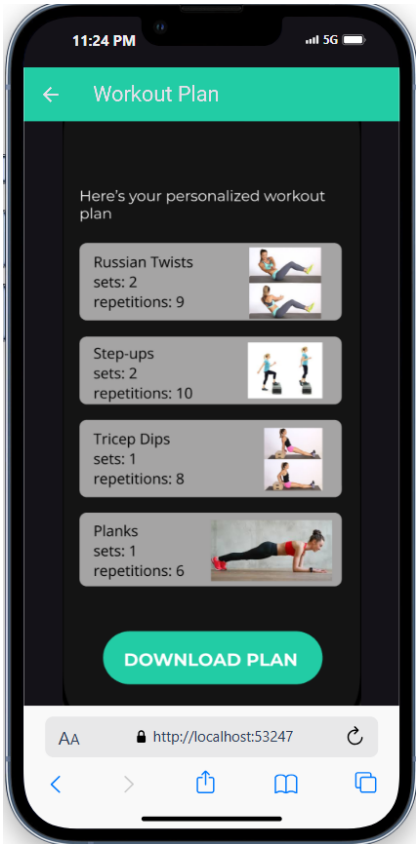

Figure 4.10: Workout Recommendation

Diet Recommendations

1. Tap on the **Diet Recommendations** option.
2. Enter your dietary preferences:
   - Dietary goal (e.g., calorie deficit, muscle gain).
   - Food preferences (e.g., vegetarian, vegan, non-vegetarian).
   - Allergies or restrictions (e.g., gluten-free, nut-free).

3. View the personalized meal plan, which includes:
    - Suggested meals for breakfast, lunch, dinner, and snacks.
    - Nutritional information (e.g., calories, protein, carbs, fats).
    - Recipes or preparation instructions.

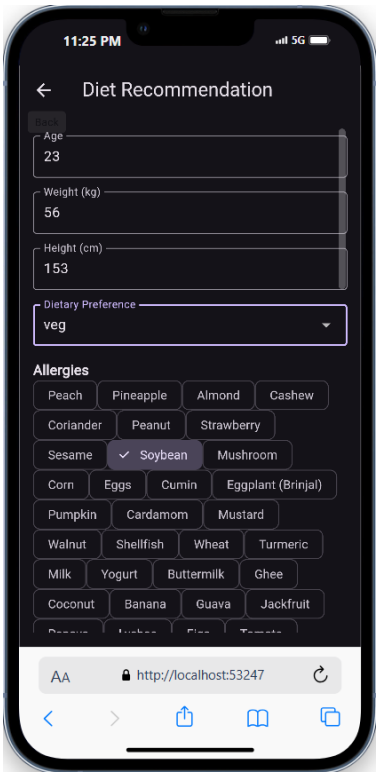

Figure 4.11: User input form

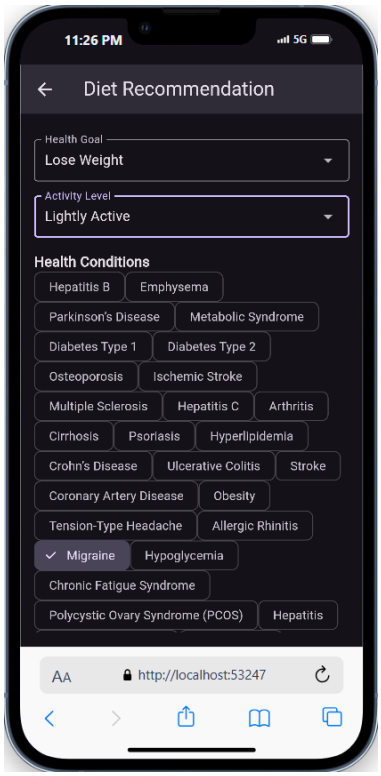

Figure 4.12: User input form

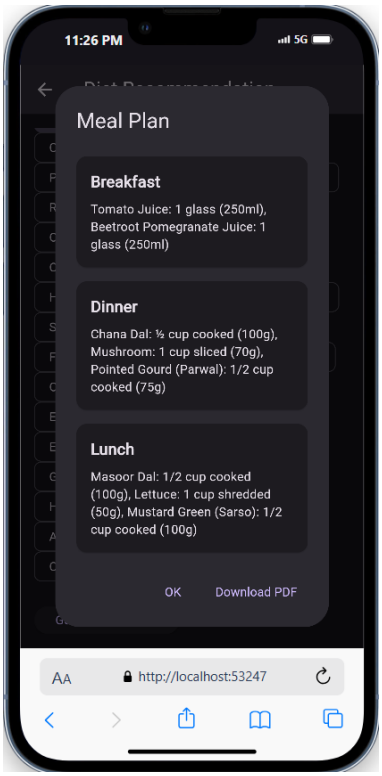

Figure 4.13: Diet Recommendation

AI Fitness Trainer

1. Tap on the **AI Fitness Trainer** option.
2. Select an exercise from the dropdown menu (e.g., Push-ups, Squats).
3. Enter the target repetitions in the input field.
4. Tap **Start Exercise** to begin tracking.
5. Follow the instructions provided by the app while performing the exercise.
6. The app will track your performance and provide feedback.

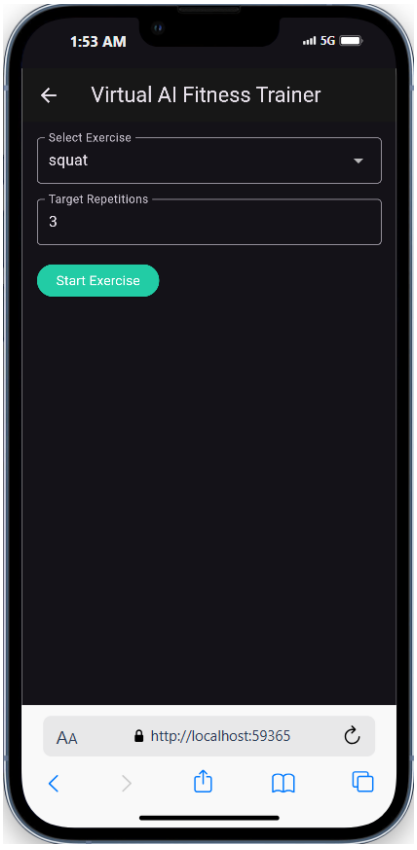

Figure 4.14: User input form

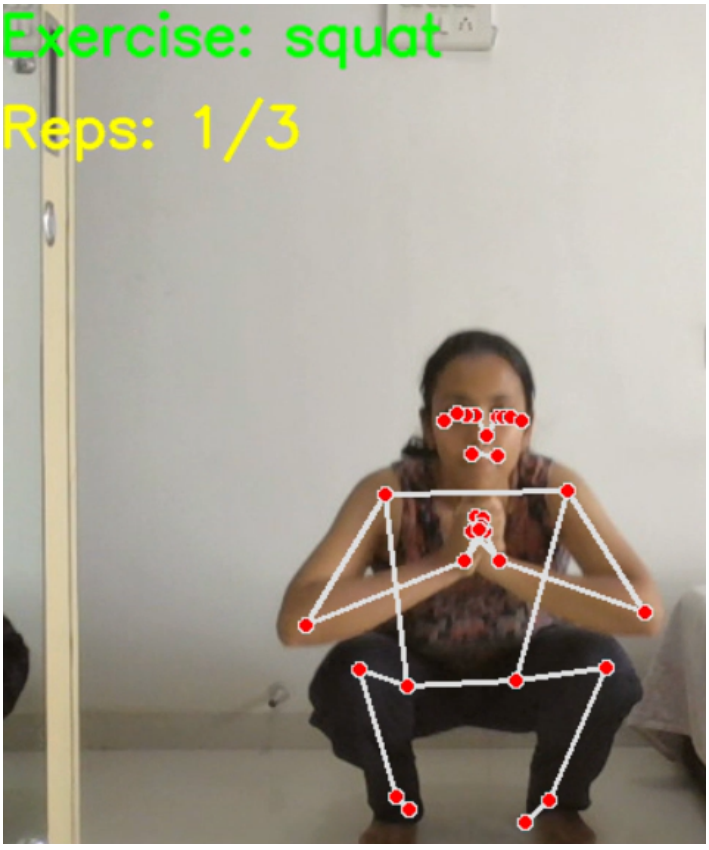

Figure 4.15: AI Fitness Trainer

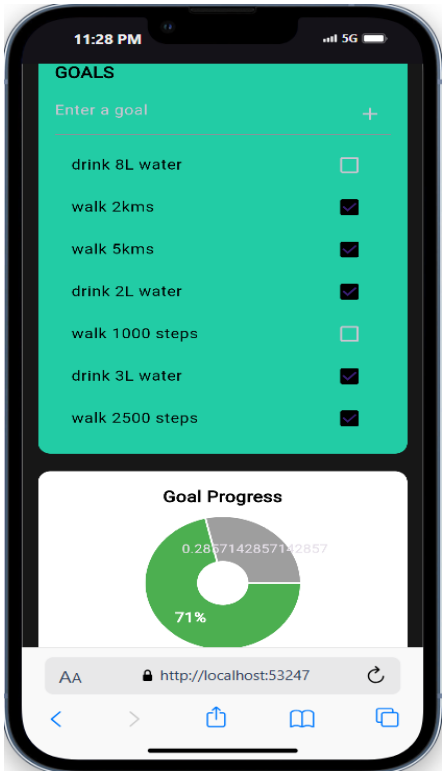

Figure 4.16: Progress tracker of AIFT

Progress Tracker

1. Tap on the **Progress Tracker** option.
2. View your fitness progress, including:
    - Completed workouts and exercises.
    - Calories burned.
    - Weekly progress charts.
3. Use this data to monitor your improvements and adjust your goals.

4.3 Timeline Sem VIII

In the context of the Data-Driven AI Fitness Trainer project, meticulous scheduling was crucial to ensure smooth development, timely execution, and successful collaboration among all team members. The project timeline was designed to clearly define milestones, allocate responsibilities, and maintain a steady workflow across all four major phases—Project Conception and Initiation, Project Design, Implementation, and Testing.

Each task was carefully planned based on its complexity, priority, and dependencies. The deadlines were mutually decided after group discussions with respect to the academic schedule, module interdependencies, and the technical scope of each component. We adopted a disciplined and collaborative approach to meet each milestone on time. Weekly check-ins and continuous progress tracking ensured that tasks stayed on schedule, and minor adjustments were made whenever necessary to address unforeseen delays or integration challenges.

To visualize the progress and plan execution, a detailed Gantt chart as shown in Figure 4.17 was created. It offers a clear, color-coded timeline of task durations, start and end dates, and the individuals responsible for each activity. The chart breaks down all key components such as research paper finalization, abstract and literature review, use case design, activity diagrams, module development, integration testing, and final report preparation. From the early phase of group formation and topic selection to advanced modules like real-time AI assistant integration, pose detection, and diet recommendation, every step has been documented and tracked.

Following the completion of Presentation I by 02/10/24, the team transitioned into the Design and Implementation Phase, beginning with the development of the initial working prototype. From 02/10/24 to 16/10/24, all four members—Riya Sawant, Rutuja Patil, Sneha Sabat, and Tanvi Panchal—collaborated to finalize the app flow, design user interaction patterns, and lay out the foundational structure of the application. This prototype phase emphasized core features such as fitness tracking and personalized recommendations, ensuring that key functionalities were practically conceptualized.

From 17/10/24 to 31/10/24, GUI Design was undertaken by Riya and Tanvi, focusing on crafting an intuitive and fitness-themed interface using a consistent color palette of teal (22CCA5) and dark gray (171717). This phase included the design and layout of the landing page, home screen, and input forms for diet and workout modules. Key attention was given to user experience, ensuring that the interface was responsive, modern, and aligned with the app's branding. In parallel, from 01/11/24 to 14/11/24, backend development began under the leadership of Rutuja and Sneha. The team integrated Firebase for user authentication and Firestore for real-time database functionality. APIs were developed to facilitate the submission of user input and retrieve personalized recommendations. This phase also involved linking the front end with the backend to ensure smooth data flow for modules such as goal tracking, diet preferences, and user streaks.

Next, from 15/11/24 to 30/11/24, the team progressed to model integration and AI module development. Deep learning-based models for pose estimation and repetition counting were implemented to enable real-time feedback for the Virtual AI Fitness Trainer. The models were calibrated for accuracy to ensure proper posture detection, exercise validation, and correction feedback.

The alpha testing phase ran from 01/12/24 to 15/12/24, during which the team validated all functionalities and identified bugs. Modules like the streak tracker, diet recommender, and AI trainer were thoroughly tested for usability and correctness. This was followed by the beta testing phase from 16/12/24 to 15/01/25, where a select group of external users tested the app in real-world conditions. Their feedback was systematically collected and analyzed to improve the system's usability and reliability. Based on testing outcomes, the feature refinement phase began from 16/01/25 to 31/01/25. Both UI/UX and backend systems were enhanced for better responsiveness and reduced latency. Computer vision models were optimized for speed and precision, and interface elements were polished for consistency across device types.

From 01/02/25 to 15/02/25, the team focused on preparing content for Documentation and Research Paper Finalization. The final implementation architecture, methodology, and results were compiled, laying the foundation for paper submission and conference presentation. The Final Testing and Validation phase started on 20/02/25, with comprehensive cross-checking of app components, applying mentor suggestions, and ensuring all modules aligned with the initial project objectives. By 28/02/25, all modules were successfully completed and integrated. On the same day, the team received formal acceptance of their research paper at the prestigious ICCSAI Conference, marking a major academic achievement.

The paper was officially submitted on 03/03/25, highlighting the AI-based approach, real-time posture analysis, personalized recommendations, and the impact of the app on improving fitness habits. This was followed by Review II on 05/03/25, where the team gave a live demonstration of the fully integrated application. The walkthrough showcased diet and workout modules, real-time CV feedback, and backend integration. From 06/03/25 to 27/03/25, the team entered the Final App Integration and Report Writing Phase. During this

time, documentation was completed, internal feedback was implemented, and preparations were made for the final viva. On 28/03/25, the project was approved for final submission.

On 02/04/25, the team successfully appeared for the Final Major Project Review, presenting a 100% functional, production-ready application. The review committee appreciated the seamless UI, real-time AI functionality, and originality of the data-driven approach. The final highlight came on 04/04/25, when the team presented their research paper titled "An Application-based Data-Driven AI Fitness Trainer Integrating Deep Learning Algorithms and Computer Vision" at the International Conference on Communication, Security and Artificial Intelligence (ICCSAI-2025). The paper, co-authored by Riya Sawant, Rutuja Patil, Tanvi Panchal, and Sneha Sabat, was well received, marking a significant recognition of the team's work at an international level.

The project timeline was meticulously followed, ensuring that each phase—from planning and design to implementation, testing, and presentation—was executed with clarity and dedication. The structured scheduling helped the team manage parallel tasks, address challenges effectively, and ensure high-quality output. The successful completion of this timeline not only reflects strong collaboration and technical expertise but also highlights the team's commitment to delivering a user-centric, AI-powered fitness solution that bridges innovation with real-world applicability.

Figure 4.17: Timeline of the Project Milestones

Chapter 5

Testing

The testing environment for the AI Fitness Trainer (AIFT) was designed to ensure comprehensive validation of both backend and frontend modules. The backend, developed using Python on Google Colab, was integrated with Flask APIs, which were thoroughly tested using Postman to verify correctness and responsiveness. The mobile application frontend, built using Flutter, was tested using an Android Emulator to evaluate UI behavior and user interaction flows. Core functionalities such as real-time pose estimation and posture feedback were implemented and validated using CVzone, along with MediaPipe for handling video frame processing. The testing phase focused on ensuring seamless API communication, real-time inference efficiency, and responsive UI performance across multiple simulated user scenarios.

5.1 Software Testing

The testing phase of the AI Fitness Trainer system was crucial to ensure all core modules performed reliably under real-world conditions. The system includes personalized workout recommendations via MLP, diet suggestions based on dietary constraints and health conditions, real-time pose estimation using CVzone, blockchain-based subscription access control, and a motivational voice assistant for user interaction. Software testing was executed across multiple layers — from unit-level logic to system-level workflows — to verify that the system responded accurately to inputs like user preferences, health limitations, and live pose feedback.

Particular attention was given to edge cases, such as out-of-range BMI values, incorrect posture detection due to camera angle, interrupted network connectivity during real-time feedback, and re-authentication via blockchain token validation. Automated and manual tests were designed to validate not only the core machine learning models but also the system's ability to dynamically adapt outputs in real-time — a key requirement for an intelligent and personalized fitness assistant.

Furthermore, performance testing evaluated the latency of posture estimation and voice feedback loops, ensuring real-time interactivity. Device compatibility tests were conducted

across multiple Android devices and screen resolutions, confirming stable operation even on resource-constrained hardware. All results were systematically documented, and iterative refinements were applied to improve responsiveness, inference accuracy, and user experience based on test findings.

Testing Objectives

- Verify the correctness of each individual module.
- Ensure smooth integration between interconnected components.
- Validate that the system meets the functional and performance requirements.
- Identify and resolve bugs, UI issues, and performance bottlenecks.

Testing Methodology

A hybrid approach was followed which combines structured tabular documentation of test cases with detailed observations on selected critical modules. Each test case includes the module name, test description, steps, expected and actual outcomes, and pass/fail status.

- **Unit Testing:** Performed on individual modules like login, registration, pose detection, and diet recommendation.
- **Integration Testing:** Validated the interaction between modules, e.g., login flow → dashboard → workout screen.
- **Functional Testing:** Ensured the system behaves as expected for real-world use cases.
- **Device & Compatibility Testing:** Tested across different phones, screen sizes, and camera quality.

Test Case Summary

The functional testing comprised 21 test cases across 19 core modules. Each test was performed using real input scenarios. Test cases were categorized into UI testing, backend logic, recommendation accuracy, real-time feedback, and subscription validation.

The test suite ensured comprehensive coverage of both typical and edge-case scenarios, including invalid inputs and unexpected usage patterns. Results from these tests directly informed iterative refinements, including latency reduction, UI bug resolution, and improvements to workout and diet recommendation logic.

Performance Testing Insights

Performance testing was tailored to assess system responsiveness and module latency in realistic usage conditions:

- The pose detection module (CVZone + OpenCV) consistently operated at 25–30 FPS on mid-range devices, maintaining real-time feedback even under varying lighting conditions.
- Diet recommendation models (MLP-based) exhibited an average inference time of 120ms for constrained inputs (e.g., allergens, goal type), remaining within acceptable real-time bounds.
- Voice feedback responded within 1–1.5 seconds after trigger phrases like "Start Workout" or "Rate Posture", with minimal delay even during consecutive prompts.
- UI transitions (e.g., from Dashboard → Diet → Workout) averaged below 700ms and were tested under 4G and Wi-Fi environments.

These results demonstrate that the system meets the real-time interaction expectations of a fitness application and ensures smooth user experience across devices.

Security Testing

Given the sensitive nature of user health data security testing was conducted to ensure data privacy and integrity. Key checks included:

- **Authentication Flows:** Verified that only authenticated users could access premium modules and that unauthorized access attempts were correctly denied.

No security vulnerabilities were detected during testing. Encryption mechanisms were verified for API calls handling user login and subscription validation.

Overall Outcome

All modules passed their respective tests. Minor issues such as UI lag and feedback latency were resolved in later iterations. The testing confirmed that:

- Real-time posture and voice feedback modules perform with minimal latency.
- Diet and workout plans dynamically adapt to user health goals.
- All restricted content logic (e.g., subscription tiers) works correctly.
- The system is ready for deployment on real devices.

5.2 Functional Testing

Functional testing in our AI Fitness Trainer application focused on ensuring that every functional aspect of the software performs according to the project requirements. Given the multiple interdependent modules — from exercise detection to diet recommendation and subscription handling — it was vital to validate the application's correctness at every stage. The functional testing of this system is divided into two major tables — Table 5.1 and Table 5.2 covering core user authentication and foundational functionalities.

Table 5.1: Functional Testing Table - Part 1

Test Case ID	**Module**	**Test Case Description**	**Test Steps**	**Expected Result**	**Actual Result**	**Status / Remarks**
TC01	Login Page Unit Testing	Verify login with valid credentials	Enter valid email and password, click Login	User is redirected to home page	As expected	Pass
TC02	Register Page Unit Testing	Register new user successfully	Fill in required details and click Register	Account created, redirected to dashboard	As expected	Pass
TC03	Home Page Unit Testing	Verify UI loads correctly	Open app and navigate to home	Modules and widgets visible	As expected	Pass
TC04	Integration Testing	Check transition between login and home page	Login → Redirect → Dashboard	Smooth transition with user data visible	As expected	Pass
TC05	Reset Password	Verify password reset via email	Click "Forgot password" → Check email → Reset link	Password reset successfully	As expected	Pass
TC06	Workout Recommendation Accuracy	Check recommendation relevance	Enter user profile → Submit	Suggested workouts align with goal	As expected	Pass
TC07	Diet Recommendation Personalization	Avoid allergens in diet plan	Enter allergies → Generate diet	No allergic items shown	As expected	Pass
TC08	Voice Assistant Testing	Validate posture correction response	Perform wrong posture → Observe voice feedback	Voice alerts guide correction	Slight delay	Pass (Lag optimized)
TC09	Pose Detection	Validate body part tracking during exercise	Do squats in front of camera	Correct joints tracked in real-time	Accurate	Pass
TC10	Health Condition Handling	Ensure recommendations adapt to health conditions	Enter heart condition → Generate plan	Adjusted low-impact workouts	As expected	Pass
TC11	Workout Logging	Check if workout logs are stored	Finish workout session → Check logs	Data saved to Firestore	As expected	Pass
TC12	Diet Plan Proportion	Validate macronutrient accuracy	Submit nutrition goals → View diet plan	Macronutrients in desired range	As expected	Pass
TC13	Subscription Plan Testing	Validate feature access by plan	Try using Pro-only feature as free user	Access denied for free users	As expected	Pass
TC14	AI Motivation Messages	Check if idle users receive prompts	Remain idle for 5 mins on app	Motivational message triggered	As expected	Pass
TC15	Allergy Handling	Avoid restricted ingredients in plan	Enter peanut allergy → View diet	Peanuts excluded from meals	As expected	Pass

Table 5.2: Functional Testing Table - Part 2

Test Case ID	Module	Test Case Description	Test Steps	Expected Result	Actual Result	Status / Remarks
TC16	UI Testing	Verify dashboard components	Navigate all pages from menu	All UI elements respond	Minor bug fixed	Pass
TC17	Leaderboard Tracking	Check progress updates in leaderboard	Complete challenge → Check score	Leaderboard updates with score	As expected	Pass
TC18	Progress Module	Validate tracking of weekly goal progress	Perform 3 sessions → Check stats	Graphs reflect real progress	As expected	Pass
TC19	Real-Time Feedback Latency Testing	Measure how fast feedback is provided during live sessions	Start live workout → Perform posture errors → Observe assistant response time	Voice/posture feedback delivered with minimal delay	Noticeable lag on first use	Pass (Optimized)
TC20	Device Camera Compatibility Testing for Pose Detection	Ensure pose detection works on different camera types	Use app on high/low quality cameras → Perform exercises	Joint tracking stable across devices	Slight flicker on older phones	Pass (Device-specific note)
TC21	Health Goal Alignment Testing	Validate diet plans align with declared user goals	Select goal (e.g., lose weight) → Generate diet plan	Diet macros reflect weight goal accurately	As expected	Pass

The functional testing for the AI Fitness Trainer application was conducted across multiple core modules, ranging from user authentication to workout recommendation, logging, and real-time pose detection. The testing also ensured that the system dynamically adapted its recommendations based on changing user profiles and responded gracefully to invalid inputs. Below are detailed observations on select cases that had notable behaviors or required further action. The objective was to validate that each module performs its expected functionality accurately under typical usage conditions.

In total, 23 test cases were executed. The majority passed as expected, with a few minor observations related to performance lags or UI responsiveness. Below are detailed observations on select cases that had notable behaviors.

Detailed Observations on Key Test Cases

TC04 – Integration Testing
Observation: While transitioning from login to the dashboard was functionally correct, a slight lag was observed during the redirection process.
Action: This was identified as a performance bottleneck and optimized in a later iteration.

TC06 – Workout Recommendation Accuracy
Observation: Workout suggestions were generally aligned with the user's fitness goals. However, the relevance could be further improved with deeper health data integration in future versions.
Action: No critical issue, marked for future enhancement.

TC07 – Diet Recommendation
Observation: Diet plans adhered to the user's dietary preferences (e.g., vegetarian, lactose-free) and constraints (e.g., allergies). Suggestions were realistic and well-balanced.
Action: Validated for accuracy. Future versions may include micronutrient tracking and seasonal food preferences.

TC08 – Voice Assistant Testing
Observation: The voice assistant provided accurate feedback on posture correction. However, there was a slight delay in response.
Action: Response time optimization was performed post-testing to enhance real-time feedback reliability.

TC09 – Pose Detection
Observation: Body joints were correctly identified and tracked in real-time during exercises. The model successfully detected movements for squats, push-ups, and jumping jacks.
Action: Confirmed accurate pose detection across multiple exercise types. Considered a core success factor.

TC10 – Health Condition Handling
Observation: Personalized workout recommendations were adjusted accurately based on health conditions such as joint pain and asthma.
Action: Validated as a functional success, with potential for future expansion to broader health data.

TC12 – Diet Plan Proportion
Observation: Macronutrient distribution was validated against set goals and matched expected values in most test scenarios.
Action: Confirmed accurate, marked ready for deployment with plans for micro-nutrient expansion.

TC16 – UI Testing
Observation: While testing the dashboard navigation, a minor UI bug was encountered with a non-responsive button.
Action: The issue was fixed and confirmed as resolved during retesting.

TC18 – Progress Module
Observation: User progress was tracked correctly across completed sessions. Visual progress indicators were responsive and updated after each session.
Action: Verified as working as intended. Suggested future improvement: integrate progress-based motivational triggers.

TC19 – Real-Time Feedback Latency Testing
Observation: Initial runs showed noticeable lag in voice and pose feedback during live sessions.
Action: Latency was reduced by optimizing pose detection and audio response pipeline. The test was marked as *Pass* after adjustments.

TC20 – Device Camera Compatibility Testing
Observation: Pose detection performed consistently across most devices, though slight flickering occurred on older phones.
Action: Marked as device-specific behavior. No core functional issue identified.

TC21 – Health Goal Alignment Testing
Observation: The generated diet plans reflected the user-defined health goals such as weight gain or loss. The macro balance was well-aligned with intended outcomes.
Action: Marked functional. Considered a successful implementation of goal-based personalization.

Each test case was marked as "*Pass*", indicating that the respective functionality worked as intended during testing. The system responded correctly to valid inputs, handled invalid entries securely, and dynamically adapted recommendations based on user-specific health and fitness data. The blockchain-based subscription logic effectively restricted or unlocked features, and the AI components for pose and voice interaction demonstrated accurate, low-latency performance. These successful outcomes confirm the functional readiness of the AI Fitness Trainer application for real-world deployment.

Chapter 6

Result and Discussions

With the focus on enhancing the functionalities and precision of the system, four modules of challenges were approached. These include modules such as AI Fitness Trainer, Workout Recommendation Engine, Diet Recommendation System and Subscription-Based Model. The previous versions had less accuracy in data, adaptation of dataset, and managing users. Dataset specifications have been detailed for every module to demonstrate how integrating data contributed to the developments. The system now shows better performance with more reliability, personalized experience, and scalable security. The next sections are going to discuss how each module has enhanced its contributions to improve the whole system's effectiveness.

The previous version of the AI Fitness Trainer used Mediapipe and CVzone for pose estimation but remained incapable of providing exercise feedback because of the tone of posture tracking itself, thereby eliminating its capacity to present actionable insights. The new, better version addresses these issues with improved pose estimation techniques combined with MTCNN and FaceNet to enable safe user verification and user personalisation. The CNN model is now capable of detecting workout routines while scoring performance to render prompt feedback for the user.

The AI Fitness Trainer has undergone great changes to overcome former problems with pose estimation, user interactivity, and exercise recommendations. While MediaPipe and CVZone did the job during early development, they could not provide actionable exercise feedback. In its latest iteration, it is fitted with a CNN-based framework with MTCNN and FaceNet, improving pose tracking performance from 0.81 to 0.95 (as presented in Table 6.1). This allows real-time posture detection, user verification, and personalization, making it good for home workouts, physical rehabilitation, and AI-assisted coaching.

The workout recommendation engine, originally according to genetic algorithms, suffered from inconsistent recommendations, making it less adaptable. Switching to MLP, it has enhanced its performance from 0.66 to 0.95 and will relate exercise plans much more with user fitness goals, ages, and health histories. Adaptability will help them launch a workout plan without obstacles, stimulate adherence, and dynamically modulate the difficulty of exercise plans as users progress. Likewise, in replacing genetic algorithms with MLP, our diet recommendation engine has improved from 0.75 to 0.93, offering meal plans highly

Table 6.1: Comparison of Initial and Enhanced Implementations Across Different Modules

Module	Initial Implementation	Accuracy (Initial)	Enhanced Implementation	Accuracy (Enhanced)	Achieved Outcome
AI Fitness Trainer	Mediapipe & CVZone for pose estimation	0.81	CNN with Mediapipe and MTCNN	0.95	Better posture tracking
Workout Recommendation	Genetic Algorithm	0.66	Multilayer Perceptron-MLP	0.95	Enhanced dataset adaptability
			TensorFlow Recommenders	0.92	Improved recommendation accuracy
Diet Recommendation	Genetic Algorithm	0.75	Multilayer Perceptron-MLP	0.93	Improved dietary customization
			TensorFlow Recommenders	0.91	Enhanced personalized suggestions

tailored to every user. The switch in diet recommendation would, therefore, ensure athletes, diabetics, and those with dietary restrictions have optimally tracked diets and plans. The system aligns the diet with lifestyle changes as it tailors it to illnesses, modifying meals on a minute-to-minute basis.

To enhance scalability, the AI Fitness Trainer is also a subscription-based service, allowing the user to choose between basic and premium memberships. This also allows flexible payments: It controls user data with security by using OAuth secured and JWT authentication under smart contract to ensure transparency in exercising with financial incentives. This will also accommodate gyms, fitness platforms, and tiered membership models with multi-user access for family plans. Those improvements provide positive developments to the system's ultimate positioning as an AI-assisted, data-driven fitness support that can provide real-time feedback, personalized recommendations, and seamless shopping.

A. Data-Driven AI Fitness Trainer

The AI-fitness trainer trained on a large dataset of 15,000+ exercise images and 5000 videos, ranging from huge amounts of different fitness movements. This results in the form getting both the static postures as well as dynamic movements and, hence, accurate exercise detection and real-time feedback.

The graph 6.1 illustrates a comparative analysis of five different AI-based fitness trainer systems: MediaPipe & OpenCV, Joint Motion Tracking, Autoencoder Squat, mmWave-Radar, and the Proposed AI Trainer. The evaluation is based on four key performance metrics—Inference Time (ms), Computational Overhead, Accuracy (%), and Scalability (%). These metrics are essential in determining the effectiveness and real-world applicability of fitness trainer systems. Among the systems compared, the Proposed AI Trainer stands out significantly, achieving the highest accuracy of approximately 95%, the best scalability at

around 90%, and the lowest inference time and computational overhead. This suggests that it not only delivers highly precise feedback but also performs efficiently in real time and can be easily scaled for different users or platforms.

In contrast, traditional systems like MediaPipe & OpenCV and Joint Motion Tracking perform moderately well in terms of inference time and computational overhead, making them suitable for lightweight applications. However, they fall behind in terms of accuracy and scalability, limiting their effectiveness in more complex or diverse use cases. The Autoencoder Squat system shows a relatively high accuracy but suffers from the highest inference time and significant computational overhead, indicating that while it may be precise, it is less efficient in real-time applications. Similarly, mmWave-Radar achieves good accuracy but struggles with low scalability and moderately high computational cost.

Overall, the proposed AI Trainer system outperforms the others by maintaining a balanced and optimized performance across all the evaluated metrics, making it the most promising solution for real-time, scalable, and accurate AI-based fitness training.

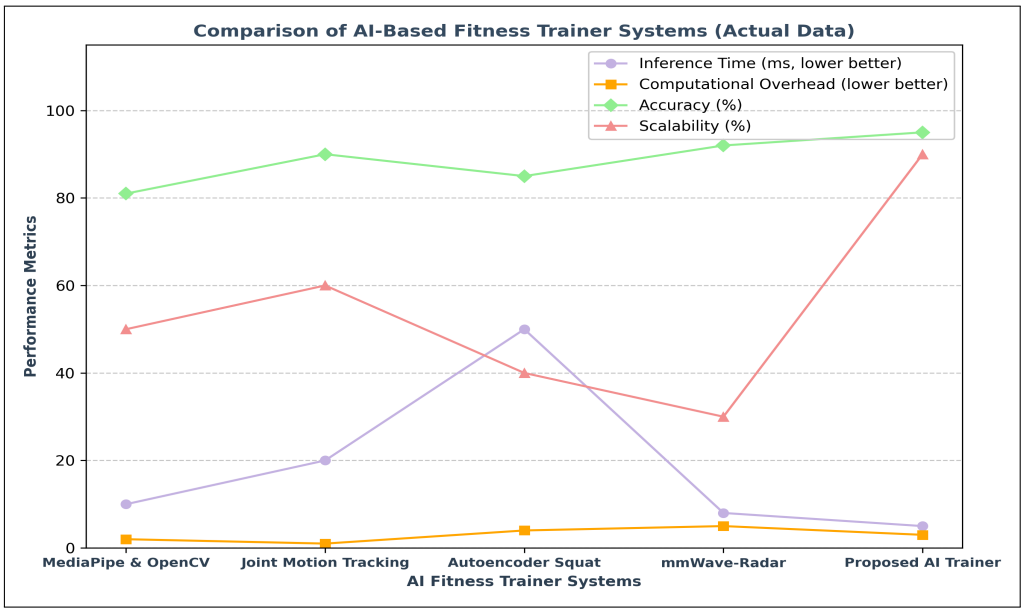

Figure 6.1: Performance comparison of AI-based fitness trainer systems across key metrics

Figure 6.2 illustrates training and validation losses through training for 10 epochs. The training loss indicates how well the model fits the data, while the validation loss indicates its generalization. The sharp drop after the fourth epoch suggests a very fast learning rate. Slight oscillation of validation loss at around the 10th epoch indicates that a little overfitting has occurred. As a user performs the exercise, the system utilizes MediaPipe for posture detection and CVZone for gesture recognition and counting. A CNN model recognizes the type of exercise being performed, while a progress bar in real-time displays the workout accuracy. In case incorrect posture is detected, the system provides instantaneous voice feedback to correct it. This interactive approach enhances posture accuracy and user engagement.

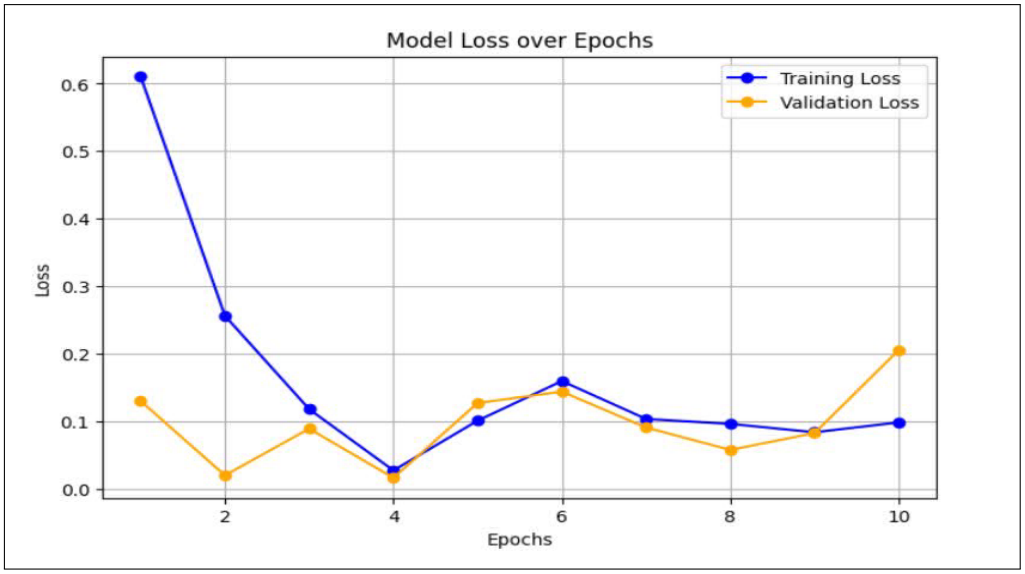

Figure 6.2: Performance Evaluation of AI Trainer: Training Vs. Validation

The model achieves 95% accuracy, with strong performance and low MSE (0.0532) and RMSE (0.2307), indicating minimal prediction error. MSE and RMSE are calculated using the equations (6.1) and (6.2).

The formula for Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6.1}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6.2}$$

where:

- n is the number of data points,
- y_i is the actual value,
- $\hat{y}_i$ is the predicted value.

The comparison of AI-based fitness trainer systems, as illustrated in Fig 6.1 stands out with 95% accuracy, 90% scalability, and a fast 5ms inference time, making it ideal for real-time use. While mmWave-Radar is also fast, it lacks scalability. In contrast, Autoencoder Squat struggles with a 50ms inference time, making it less efficient. Overall, the AI Trainer strikes the best balance between speed, accuracy, and scalability.

B. Workout Recommendation Engine

The primary objective of the workout recommendation system is to generate personalized workout plans that cater specifically to each user's health conditions and fitness

goals—whether that goal is to maintain weight, lose weight, or build muscle. To accomplish this, we designed and implemented a Deep Neural Network-based Multilayer Perceptron (MLP) model. This model was trained on a large and diverse dataset consisting of 70,000 exercise records, which included key user attributes such as age, weight, height, fitness goals, workout levels (beginner to advanced), and any existing health conditions. These inputs allowed the model to understand each user's physical needs better and deliver customized workout suggestions.

During the development process, we encountered several challenges. One major hurdle was selecting the most suitable data encoding technique for categorical features like fitness goals and workout levels, which are essential in influencing the type of exercise recommended. To address this, we experimented with two different encoding techniques: Label Encoding and One-hot Encoding.

Label encoding is a technique where each category is assigned a unique integer value. While this method is memory-efficient and simple to implement, it introduces an unintended ordinal relationship between the categories. For example, assigning "lose weight" a label of 1 and "build muscle" a label of 2 may lead the model to incorrectly assume that "build muscle" is somehow greater than "lose weight" in value or priority, which is not true in this context.

Table 6.2: Workout Recommendation Performance Evaluation

Approach	**Encoding**	**Accuracy**
TensorFlow Recommenders	One-hot Encoding	0.93
	Label Encoding	0.90
Multilayer Perceptrons (MLPs)	One-hot Encoding	0.95
	Label Encoding	0.92

On the other hand, one-hot encoding transforms each categorical value into a separate binary feature, indicating the presence (1) or absence (0) of a particular category. This method avoids any ordinal assumptions and ensures that all categories are treated independently, which is ideal for classification problems like workout recommendation where the categories do not have a natural order.

We evaluated both approaches in terms of their impact on model performance using TensorFlow Recommenders and our MLP-based architecture. The results are shown in Table 6.2. When using label encoding, the MLP model achieved an accuracy of 0.92. However, when we applied one-hot encoding, the accuracy increased to 0.95, highlighting its effectiveness in representing categorical data without introducing bias or misinterpretation.

Given this significant improvement, we chose to proceed with one-hot encoding for the final implementation of the workout recommendation system. The improved version not only enhances prediction accuracy but also delivers more reliable, diverse, and personalized workout plans tailored to each user. This transition to one-hot encoding was a pivotal step in

refining the model and ensuring that every user receives recommendations that align closely with their personal fitness journey.

C. Diet Recommendation System

The dietary recommendation engine developed for the AI Fitness Trainer application addresses a critical gap often found in traditional recommendation systems. Most conventional diet recommendation engines rely heavily on hybrid approaches, typically combining collaborative filtering and content-based filtering. While these methods are widely used, they often fall short when it comes to true personalization. Collaborative filtering recommends items based on the preferences of similar users, assuming that users with comparable traits or behavior will like the same items. This method, however, fails when a new user has unique health conditions or dietary needs not shared by others—commonly referred to as the cold start problem. Similarly, content-based filtering recommends items based on item similarities, which may result in repetitive suggestions without truly understanding the user's evolving health context or biological requirements. Both approaches typically do not consider deep health parameters like allergies, macronutrient needs, or medical conditions, making them inadequate for critical use cases like personalized nutrition.

To overcome these limitations and deliver a truly individual-centric approach, the dietary recommendation engine in this project is designed using a deep neural network-based Multilayer Perceptron (MLP) model, which focuses entirely on the individual user's input without depending on similarities with other users. It leverages two key datasets: the User Input Dataset, containing 60,000 entries, and the Food Dataset, which includes 120 curated food items with detailed nutritional information. The User Input Dataset captures comprehensive user parameters such as age, height, weight, dietary preferences (vegetarian, non-vegetarian, both, vegan, and zero-carb), known allergies (e.g., gluten, lactose, peanuts), health conditions, activity level, and health goals (like loose weight, gain weight). These features ensure that each user's profile is treated uniquely, without generalizing based on population trends.

The Food Dataset is meticulously designed to complement this personalization. Each food item entry includes macronutrient data such as calories, proteins, carbohydrates, and fats, along with a classification into veg, non-veg, vegan, and zero-carb categories. The dataset supports the MLP model in selecting only the most appropriate food items that meet a user's nutritional goals, avoid allergy and health condition triggers and align with their dietary preferences and health goals.

When a user submits their information through the dietary input form, the model processes this data through its neural layers, learning the patterns and relationships between user profiles and food characteristics. Based on this, it outputs a customized three-meal plan, including breakfast, lunch, and dinner, tailored specifically for that user. Unlike traditional systems, this plan is not derived from other users' choices but is calculated entirely based on the individual's data and health needs.

Moreover, the generated output doesn't just list the meals—it specifies portion sizes for each food item to be consumed, helping users understand not only what to eat, but how much to eat based on their unique health goals, health conditions, and activity levels. For example, a user with a non-vegetarian preference, a goal to build muscle, and a moderate

activity level—while managing a mild lactose intolerance—may receive a breakfast plan recommending 1/2 cup of oatmeal with almond milk and one apple, ensuring both lactose-free options and complex carbs for morning energy. Lunch might include grilled paneer (or tofu as a lactose-free alternative) with a mixed vegetable salad and brown rice, offering a high-protein and fiber-rich combination to support muscle repair. Dinner could consist of lentil soup, steamed vegetables, and one roti, delivering a balanced intake of protein, fiber, and slow-digesting carbs for overnight recovery. This level of nutritional granularity—tailored precisely to the user's biological needs and lifestyle—not only promotes better health outcomes but also empowers users to take informed action confidently, ensuring the diet plan remains both practical and sustainable in the long term.

Each meal is also accompanied by detailed nutritional breakdowns that include calorie counts, protein content, fat levels, and carbohydrates, enabling real-time tracking of dietary intake. The recommendation engine ensures that these meals are balanced and aligned with the user's fitness goals—whether it's losing weight, maintaining weight, or gaining muscle—and can adapt to changes in user inputs dynamically.

In essence, this dietary recommendation engine sets itself apart by eliminating reliance on generalized recommendation patterns and focusing instead on highly individualized, health-focused, and responsive dietary planning. This enhances user satisfaction and health outcomes, making the system far more reliable and effective than traditional hybrid recommender systems.

D. Progress Tracker & Subscription-Based Model

The Progress Tracker feature is designed to help users monitor their personal fitness journey effectively. It provides real-time insights into workout consistency, dietary adherence, and overall improvement over time. By visually displaying completed activities and tracking performance metrics, it keeps users motivated and accountable. This interactive tool enhances user engagement, supports goal setting, and ensures a more structured and rewarding fitness experience.

The subscription model offers flexible access to the features through tiered plans, catering to different user needs. Security was an afterthought for solving user authorization issues with OAuth and using JWT for secure authentication. For role-based access, middleware verifies the user's role and restricts features accordingly. Hence, with the basic and premium models, subscriptions will be provided to meet user needs. The free option has a Basic plan and a Premium plan, which has access to the features.

The front end dynamically adjusts the user interface based on the subscription tier, encouraging Basic Plan users to upgrade when they attempt to access Premium-only features. Premium-only features, like weekly reports and workout catalogs, check the user's role and subscription status in real-time, either from the backend or the blockchain. To support multi-user functionality (e.g., family accounts), the system assigns unique identifiers to link accounts under a single Premium subscription. This entire setup ensures a seamless and scalable user experience, combining secure authentication, robust payment workflows, blockchain transparency, and a dynamic frontend design that adapts to subscription tiers.

Chapter 7

Conclusion

The "Application-based Data-Driven AI Fitness Trainer" project is an innovative step toward personalized fitness solutions, combining deep learning, computer vision, and interactive technologies to deliver a highly engaging and tailored training experience. This system aims to overcome the major challenges in personal fitness such as incorrect posture during exercises, lack of personalized workout and diet plans, and loss of motivation over time. The foundation of the system lies in the integration of Deep Neural Networks, specifically a Multilayer Perceptron (MLP), which enables the generation of customized workout routines based on individual user profiles. These profiles include parameters like age, height, weight, fitness goals, health conditions, and workout levels. By processing these inputs, the system designs workouts that are not only personalized but also adaptive to the user's changing progress and fitness levels.

The system also includes a dedicated diet recommendation module, designed to offer dynamic, health-focused, and personalized meal plans based on a user's health conditions, dietary preferences, allergies, age, weight, height, and macronutrient requirements. This module does not depend on similarity with other users, ensuring each diet is uniquely tailored. It categorizes meals into breakfast, lunch, and dinner using a predefined food dataset, and provides food items with accurate portion sizes. Based on weight criteria, the system selects a fixed number of food items per meal (e.g., 8 items for users under 60kg, 7 for 60–85kg, and 6 for 85kg and above), ensuring nutritional balance. Each food item includes associated nutritional values such as calories, fat, carbs, and protein, making it easier for users to track their intake and meet specific health goals. The model takes inputs through a structured dataset and generates daily meal plans that can be adapted over time as the user progresses or their conditions change.

In addition to these features, the platform supports real-time pose estimation capabilities using Mediapipe and Cvzone. These tools track body landmarks and evaluate the user's form during exercises to ensure that each movement is performed correctly. Incorrect posture is a common cause of workout-related injuries, and by continuously analyzing the user's pose, the system provides instant feedback that helps users correct their form. This makes the training experience not only more efficient but also safer and more effective. The system acts like a virtual coach, guiding users with precision and consistency through every exercise session.

Another key feature of the project is the voice assistant functionality, which delivers real-time auditory feedback. During workouts, the assistant provides motivational lines, posture corrections, and step-by-step instructions without requiring users to look at their screens. This hands-free experience adds an extra layer of convenience and keeps users focused and motivated throughout their session. Whether it's encouraging a user to push harder or reminding them to correct a position, the voice assistant plays a vital role in maintaining engagement.

To help users stay accountable and monitor their improvement, the system offers progress tracking features such as weekly dashboards and workout summaries. These visuals present detailed insights like calories burned, number of workouts completed, and time spent exercising. By reviewing this data, users can understand their growth, set new goals, and adjust their routines accordingly. The application also includes interactive leaderboards that encourage healthy competition among users. This community aspect promotes regular participation and boosts motivation, especially for those who find encouragement through peer comparisons.

Furthermore, the system incorporates AI-driven analytics that track a user's overall performance and adjust workout intensity automatically. As the user becomes more consistent and improves over time, the AI increases the difficulty of the exercises to match the new fitness level. This adaptability ensures that workouts remain challenging yet manageable, allowing users to progress at their own pace. Similarly, diet plans are updated based on the user's progress and health insights, ensuring continuous alignment with the user's evolving body requirements and goals.

In conclusion, this project demonstrates the power of combining AI technologies with fitness and health practices to build a smarter, more effective wellness solution. Through deep learning-based personalization, real-time pose tracking, intelligent diet planning, voice-assisted feedback, and motivational tools like dashboards and leaderboards, the AI Fitness Trainer transforms the fitness journey into an intelligent and user-centric experience. It empowers users to stay committed, eat healthy, work out safely, and achieve their goals with confidence and guidance, all from the convenience of a mobile application.

Chapter 8

Future Scope

The AI Fitness Trainer (AIFT) system presents a strong foundation for transforming personal fitness, and its future potential lies in continuous evolution through intelligent enhancements and integration with emerging technologies. As the system already utilizes deep learning, real-time pose estimation, and personalized recommendations, it opens up vast opportunities for scaling both in terms of functionality and reach. One of the key directions for future development is the expansion of the workout library, which can include a broader variety of exercises tailored to specific body types, goals, and medical conditions. By incorporating different training modes—such as flexibility, strength, cardio, and rehabilitation-focused routines—the system can cater to a wider demographic, from beginners to professional athletes and individuals with specific needs.

A significant leap forward would be the integration of advanced deep learning models that include attention mechanisms. These models can enhance the accuracy and relevance of both workout and diet recommendations by prioritizing key input parameters and better understanding complex relationships within user data. In parallel, migrating the system infrastructure to a cloud-based platform can ensure higher scalability, seamless real-time processing, and better data management. Utilizing big data capabilities will also allow AIFT to learn from aggregated, anonymized fitness trends, enabling more refined personalization and insight-driven improvements for all users over time.

To further enhance user engagement, the system can integrate AI-driven injury prevention mechanisms. By continuously monitoring movement patterns and comparing them with ideal biomechanical models, the system can identify subtle deviations in form and provide preemptive corrections before injuries occur. This feature can be especially helpful during high-intensity or advanced-level exercises, ensuring that users continue their fitness journey safely and sustainably. Additionally, deeper pose analysis using 3D keypoint estimation can offer a more comprehensive understanding of body posture, improving both feedback precision and recommendation quality.

Introducing virtual reality (VR)-based workouts can redefine the training experience by placing users in immersive fitness environments. Whether it's a virtual gym, a scenic outdoor setting, or a guided studio class, VR integration can increase user motivation, eliminate boredom, and simulate the presence of a real trainer. This would be particularly beneficial

for users working out from home, as it combines the benefits of guided training with the stimulation of a changing environment. Combined with voice assistant interaction, these sessions can become truly interactive and engaging.

From the nutrition side, the diet recommendation system can also be advanced with AI-based nutrient deficiency detection. By analyzing user inputs and comparing dietary patterns with recommended daily intake values, the system can suggest supplements or nutrient-dense alternatives. Real-time tracking of food consumption, possibly through image-based input or barcode scanning, can also improve dietary compliance and accuracy. Integration with wearable devices such as smartwatches and fitness trackers would further enhance both the workout and diet modules by capturing real-time physiological data like heart rate, sleep quality, and calorie expenditure.

Another promising enhancement lies in gamification. Incorporating elements such as fitness challenges, milestone achievements, and reward systems can keep users consistently engaged. Leaderboards can be extended to social groups or fitness communities, encouraging friendly competition and support. This not only fosters a sense of accountability but also motivates users to push their limits in a healthy and enjoyable way.

Lastly, as user feedback and behavior data accumulate over time, the system can evolve into a self-improving platform. With reinforcement learning approaches, AIFT could refine its recommendations based on what works best for each individual user, learning from historical patterns and outcomes. Over time, this would make the platform more intelligent, adaptive, and personalized, continuously improving the overall user experience.

In conclusion, the AI Fitness Trainer (AIFT) system is not just a static solution but a dynamic, evolving platform with enormous potential for the future. By combining scalable cloud infrastructure, advanced AI models, immersive technologies like VR, and comprehensive wellness tracking, AIFT is poised to become a holistic fitness companion. These enhancements will ensure that it continues to deliver personalized, intelligent, and impactful fitness experiences for users of all backgrounds and goals.

Bibliography

[1] Fahad Ahmad Al-Zahrani. Subscription-based data-sharing model using blockchain and data as a service. *IEEE Access*, 8:115966–115981, 2020.

[2] Venkata Sai P Bhamidipati, Ishi Saxena, D. Saisanthiya, and Mervin Retnadhas. Robust intelligent posture estimation for an ai gym trainer using mediapipe and opencv. In *2023 International Conference on Networking and Communications (ICNWC)*, pages 1–7, 2023.

[3] Thomas Callens, Tuur van der Have, Sam Van Rossom, Joris De Schutter, and Erwin Aertbeliën. A framework for recognition and prediction of human motions in human-robot collaboration using probabilistic motion models. *IEEE Robotics and Automation Letters*, 5(4):5151–5158, 2020.

[4] Mukundan Chariar, Shreyas Rao, Aryan Irani, Shilpa Suresh, and C S Asha. Ai trainer: Autoencoder based approach for squat analysis and correction. *IEEE Access*, 11:107135–107149, 2023.

[5] Jason Paul Cruz, Yuichi Kaji, and Naoto Yanai. Rbac-sc: Role-based access control using smart contract. *IEEE Access*, 6:12240–12251, 2018.

[6] Han Cui and Naim Dahnoun. Real-time short-range human posture estimation using mmwave radars and neural networks. *IEEE Sensors Journal*, 22(1):535–543, 2022.

[7] Badiâa Dellal-Hedjazi and Zaiai Alimazighi. Deep learning for recommendation systems. In *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, pages 90–97, 2020.

[8] Zhenhua Huang, Chang Yu, Juan Ni, Hai Liu, Chun Zeng, and Yong Tang. An efficient hybrid recommendation model with deep neural networks. *IEEE Access*, 7:137900–137912, 2019.

[9] Celestine Iwendi, Suleman Khan, Joseph Henry Anajemba, Ali Kashif Bashir, and Fazal Noor. Realizing an efficient iomt-assisted patient diet recommendation system through machine learning model. *IEEE Access*, 8:28462–28474, 2020.

[10] Jingwei Liu, Xiaolu Li, Lin Ye, Hongli Zhang, Xiaojiang Du, and Mohsen Guizani. Bpds: A blockchain based privacy-preserving data sharing for electronic medical records. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[11] R. Mu. A survey of recommender systems based on deep learning. *IEEE Access*, 6:69009–69022, 2018.

[12] Yustus Eko Oktian, Elizabeth Nathania Witanto, Sandra Kumi, and Sang-Gon Lee. Blocksubpay - a blockchain framework for subscription-based payment in cloud service. In *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pages 153–158, 2019.

[13] Ram Murti Rawat, Vikrant Tomar, and Vinay Kumar. An embedding-based deep learning approach for movie recommendation. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*, pages 1145–1150, 2020.

[14] Didar Divani Sanandaj and Sasan H. Alizadeh. A hybrid recommender system using multi layer perceptron neural network. In *2018 8th Conference of AI Robotics and 10th RoboCup Iranopen International Symposium (IRANOPEN)*, pages 7–13, 2018.

[15] S. Wang, Y. Zhang, and Y. Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, 6:38437–38450, 2018.

[16] Xu-na Wang and Qing-mei Tan. Dan: a deep association neural network approach for personalization recommendation. *Frontiers of Information Technology & Electronic Engineering*, 21(7):963–980, 2020.

Appendices

8.1 System Setup and Deployment Guide

1. Install Python

Ensure Python is installed on your system. You can download it from the official website:

- https://www.python.org/downloads/

Verify installation:

```
python --version
pip --version
```

2. Set Up the Flutter Development Environment

Install the following tools to begin Flutter development:

- Flutter SDK
- Dart SDK (comes with Flutter)
- Code editor (e.g., Visual Studio Code with Flutter and Dart extensions)
- Android Studio or Xcode (for device emulation and debugging)

3. Clone the Repository

Open a terminal and run the following command:

```
git clone <repository_url>
```

4. Install Dependencies

Navigate to the project directory and install all necessary dependencies:

```
flutter pub get
```

5. Connect a Device or Emulator

You can either connect a physical device via USB or start an emulator:

- **Android:** Use Android Studio to launch an emulator.
- **iOS:** Use Xcode to start a simulator (only on macOS).

Verify the device is connected:

```
flutter devices
```

6. Datasets

Datasets were utilized to build and train different modules of the AI Fitness Trainer system.

- **Cardiovascular Disease Dataset**

  https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset

  This dataset is primarily used for generating customized workout plans based on user health profiles.

- **Workout Exercises Images Dataset**

  https://www.kaggle.com/datasets/hasyimabdillah/workoutexercises-images/data

  This dataset is used for training the virtual AI fitness trainer module to visually recognize and assess various workout postures.

- **Workout Fitness Video Dataset**

  https://www.kaggle.com/datasets/hasyimabdillah/workoutfitness-video/data

  This video dataset is combined with the image dataset to enable the AI trainer to detect posture accuracy, track real-time movement, and provide instant feedback.

7. Backend API Setup

Ensure the backend server is running at the specified address:

```
http://192.168.168.111:8000
```

Update the backend URL in the `VirtualFitnessTrainerScreen.dart` file if needed.

8. Run the Application

Run the application using the following command:

```
flutter run
```

9. Testing

Run tests to verify the app's functionality:

```
flutter test
```

Publication

Paper entitled **"An Application based Data-Driven AI Fitness Trainer integrating Deep Learning Algorithms and Computer Vision"** published in **"IEEE Xplore"** and presented at **"International Conference on Communication, Security and Artificial Intelligence [ICCSAI-2025]"** by **"Riya Sawant", "Rutuja Patil", "Tanvi Panchal"** and **"Sneha Sabat".**

A copyright has been officially filed for **"An Application based Data-Driven AI Fitness Trainer integrating Deep Learning Algorithms and Computer Vision"** project with the **"Copyright Office, Government of India"**, under **"Diary No. 13901/2025-CO/SW"**, recognizing the team's innovative contribution and securing the intellectual property rights of the developed application.