WellnessAI: A Comprehensive Fitness and Wellness Platform

NAZIA BUTT
01-135221-042

ISRA SALEEM
01-135221-064

Supervisor: Mr.Saeed ur Rehman

Bachelor of Science in Information Technology

Department of Computer Science
Bahria University, Islamabad

December 2025

Certificate

We accept the work contained in the report titled "WellnessAI:A Comprehensive Wellness And Fitness Platform", written by Ms. NAZIA BUTT AND Ms. ISRA SALEEM as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Information Technology.

Approved by . . . :

Supervisor: Mr.Saeed ur Rehman

_____

Internal Examiner:

_____

External Examiner:

_____

Project Coordinator: Dr Kabeer Ahmed Bhatti (Assistant Professor)

_____

Head of the Department: Dr Muhammad Khurram Ehsan (Associate Professor)

_____

Abstract

The rapid expansion of online fitness has highlighted a critical gap: the absence of intelligent analysis, real-time form correction, and expert oversight in existing applications. This often results in users failing to achieve intended fitness goals or suffering injuries due to improper technique. To address this necessity for combining professional guidance and artificial intelligence, the **WellnessAI** platform is introduced as a comprehensive, role-based digital wellness ecosystem designed to bridge this divide.

The platform provides a complete and unified user experience on a single, secure interface, encompassing plan browsing and purchasing, trainer appointment booking, in-app payments, professional onboarding, and administrative monitoring. The architecture is founded on a robust **Flutter** application for the mobile experience, with all user and operational data securely managed by **Firebase Authentication** and **Cloud Firestore**.

The core innovation resides in the web-based **AI Workout Module**, developed using **Vue.js** and **Django**. This module leverages **MediaPipe Pose** to extract body landmarks in real time and applies pretrained machine learning models to perform critical functions, including repetition counting, exercise stage classification, and precise error detection for exercises such as squats, lunges, and bicep curls. Supporting the system are components like Firebase Cloud Messaging (FCM) for notifications and Cloudinary for secure media storage.

By integrating this intelligent, real-time posture analysis with a professional service layer, WellnessAI facilitates safe, proper, and effective home-based exercises. This advancement empowers users with smart, expert-inspired feedback, driving improved performance, motivation, and overall well-being, thus marking a significant innovation in the digital fitness landscape.

Acknowledgments

In the name of Allah, the most beneficent and the most merciful. We are highly grateful to the One who created us and blessed us with a privileged life. We would like to thank our parents who have always been a pillar of strength and support. We are thankful to our parents who taught us that how hard work, devotion, and working towards your goal with pure intention can take you to places. Furthermore, we would like to thank and appreciate our supervisor Mr. Saeed ur Rehman who has given us a chance to work on a project that can help in creating value for the Digital Fitness Domain. His professional guidance, time, and effort will always be remembered. Last but not the least, we would like to appreciate our friends who make studying and hard times less challenging. May ALLAH (SWT) guide us to the right path.

NAZIA BUTT AND ISRA SALEEM
Bahria University
E-8, Islamabad, Pakistan

December 2025

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AI	Artificial Intelligence
FCM	Firebase Cloud Messaging
UI	User Interface
UX	User Experience
CV	Computer Vision
RBAC	Role-Based Access Control
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
API	Application Programming Interface
VPS	Virtual Private Server
FPS	Frames Per Second
SSL	Secure Socket Layer
ML	Machine Learning
REST	Representational State Transfer
URL	Uniform Resource Locator
DBMS	Database Management System

Chapter 1

Introduction

1.1 Overview

The online revolution of fitness and wellness has changed the lifestyles of people in the way they seek to live healthier lives [1] . In spite of the broad variety of offered fitness apps and virtual training courses, many of them do not have intelligent analysis, proper form correction, or professional control [2] . Because of that, users tend to fail to gain the intended results or may develop injuries because of unethical exercise. This demonstrates the necessity of a system combining professional guidance and artificial intelligence to provide personalized, high-quality, and safe workout experiences. The WellnessAI is introduced as a fully role-based wellness platform that will fill this gap by combining Flutter-based mobile technology [3] with a web-based AI Workout Module. The platform offers users a single fitness experience- plan browsing, plan purchasing, appointment booking, in-app payments, professional onboarding, and administrative monitoring is available on a secure and easy-to-use platform. WellnessAI technically uses Firebase Authentication and Cloud Firestore to manage users and data [4] , Provider to manage state and gorouter to provide role-based navigation. Firebase Cloud messaging (FCM) is used to handle the notification and Cloudinary is used to store and optimize media stored securely. The AI Workout Module is fundamentally based on pose detection and machine learning to process the exercises of the user in real time. It is developed using Vue.js [5] and Django [6] , and includes MediaPipe Pose to extract body landmarks [7] and pretrained models to count repetitions, classify the stage, and detect errors in sports like squats, planks, bicep curls, and lunges.

WellnessAI is a big innovation in the digital fitness by integrating professional instruction and AI-inspired feedback. It not only makes users train effectively but also safely empowered by smart insights that encourage improved performance, motivation and general well-being.

1.2 Problem Description

Among the many individuals in the modern paced lifestyle, time, guidance and motivation are the major factors that lead to the inability to perform regular fitness routines [1] . Although there are other fitness applications, the majority of them use pre-recorded videos and fixed workouts that cannot guarantee proper exercise posture. This mostly causes futile exercises and increased risk of injuries.

It is also a challenge to find certified trainers or reputable online fitness websites. Current solutions are disconnected–users have to use several dissimilar tools to schedule, make payments and communicate, but there is no real-time form correction. Because of this, users and professionals have challenges in realizing effective training outcomes.

WellnessAI was suggested to counter those challenges by offering a role-based wellness platform, a unified wellness platform that combines human expertise with artificial intelligence. It enables its users to reserve trainers, buy custom workout plans, and get their workouts tracked by AI in the form of pose recognition and machine learning [8] .

Put simply, WellnessAI is the solution that allows closing the gap between professional fitness instructions and home-based exercises so that the user exercises safely, properly, and effectively using intelligent real-time posture analysis.

1.3 Project Objectives

The primary objective of this project is to build a smart wellness platform, which integrates fitness commerce, scheduling, communication and AI-driven workout feedback into one platform.

Key Objectives:

1. User Management: Customers, professionals, and admins are authenticated and authorized with Firebase Auth and gorouter.
2. Professional Features: enable trainers to deal with fitness plans, media, schedules and earnings using wallet and booking features.
3. Customer Features: Customers are able to browse and order plans, make appointments, update and chats with professionals.
4. Payment System: Assistance in in-app purchases and deduction of platform fees and automatic confirmation of emails.
5. Notifications: Add Firebase Cloud Messaging (FCM) to push and notifications reminders.

6. Chat System: Add chat connected to the booking to have real-time communication between the user and trainers.
7. AI Fitness Module: Take a posture test (squats, planks, bicep curls, lunches) with MediaPipe and pre-trained models.
8. Admin Oversight: Establish user and financial management dashboard and workflow and trainer approval.

1.4 Project Scope

The goal of the project is to create a single wellness environment, capable of offering fitness schedules, artificial intelligence, and intelligent workout feedback, and uninterrupted communication between the customers, trainers, and administrators. This system aims at uniting role-based access, artificial intelligence analysis, and interactivity in real-time in one mobile platform and web-based platform.

The key factors in this project are:

1. Mobile App (Flutter): The system will have a customer, trainer, and administrator role-based experience that includes the ability to browse plans, purchase, book, chat, get push notifications, use a wallet, and in-app functionality of the AI workout module.
2. AI Workout (Vue.js + Django): Provides web camera and video exercises analysis with MediaPipe Pose detector and pretrained modules to analyze squats, planks, bicep curls, and lunges with stage detection, repetition counting, and posture correction visual feedback overlay.
3. Data & Services: Uses Firebase Firestore to store logistics, user, plans, bookings, payments, chats, reviews and notifications, as well as automated email confirmation and reminder alerts.

1.5 Out-of-Scope

- Diagnosis and recommendations of medical or health reasons.
- Real payment gateway integration (e.g. Stripe, PayPal).
- Fine tuning or training the machine learning models on devices of end-users.
- Performance optimization on a large scale.
- Connection to wearable devices or sensors.

- Voice recognition or other advanced motion sensing other than pose detection.
- Video call / live meeting in appointments (text and media chat are supported only)

1.6 Project Boundaries

- The user-facing workflows handled by the mobile app include authentication, scheduling and communication.
- The AI component is concerned with input pose and exercise feedback.
- The two elements are compatible to each other - the app will open AI sessions and the web module will scan the poses and provide feedback.
- The information of the data is strictly divided: the app receives the information about the user, and the AI module does the analysis of the exercises.

1.7 Limitations and Constraints

- Performance AI feedback requires stable 60-120 FPS video and low latency (<200 ms).
- Browser Security: HTTPS (SSL) is required to access cameras/mic.
- Deployment: Local testing is restricted to intra-device testing. Production is being needed in VPS.
- Academic Constraints:
    1. Models do not undergo live learning.
    2. The mobile notifications can be different depending on the OS.
    3. The project is concerned with demonstration level functionality.

1.8 Integration Note

However, the mobile app and AI workout module are different systems that integrate with each other well. Through the app, people can access the web-based AI Workout Module where they can upload workout videos or get an exercise in real-time, providing immediate feedback. Such integration will provide cohesive and smart wellness to users and trainers.

Chapter 2

Literature Review

This chapter examines technologies and systems, which WellnessAI can use such as Computer Vision, MediaPipe Framework and MediaPipe Pose then compares applicable AI-fueled fitness applications and their drawbacks. The review features the enhanced aspect of WellnessAI, compared to existing systems, in the realms of advanced AI-based posture detection, professional trainer consultation, and built-in management capabilities.

2.1 Computer Vision

Computer Vision helps machines to understand or interpret images or videos in a manner close to humans [9] . It derives visual data based on pixels with pattern, shape, and movement identification algorithms.

Applications of computer vision are:

- Healthcare: Diagnosing a disease by use of the X-rays or MRI scan.
- Automotive: Helping autonomous vehicles to identify pedestrians and signs.
- Security: Recognizing people using their faces.

Within the framework of the WellnessAI, computer vision enables the system to identify body motions and postures in order to get the correct analysis of exercises.

2.2 MediaPipe Framework

MediaPipe is a Google open-source platform that is used to create perception pipelines to process images, video, and audio in an efficient way [7]. It links multiple processing units

(referred to as calculators) in a graph to achieve tasks like pose detection, face mesh or hand tracking.

Key advantages:

- Android, iOS, Web and Desktop.
- Operates well even in mobile devices.
- Simple interaction with machine learning models.

MediaPipe is also applied in WellnessAI, where camera images are analyzed to restore body landmarks and analyze the movement of the human body with AI.

2.3 MediaPipe Pose

MediaPipe Pose is a model, which is based on BlazePose and can identify 33 body landmarks 3D [8] . It uses a two-step process:

1. Detector - identifies the human body within the picture.
2. Tracker - tracks brings marks about landmarks constantly.

The output includes:

- coordinates (x, y, z) of every landmark.
- Visibility score of the clarity of that point.

Such landmarks are useful in helping WellnessAI understand whether the users are exercising in the correct way e.g. ensure that the elbow angle is correct in pushups or the back position is correct in squats etc.

MediaPipe Pose Output:

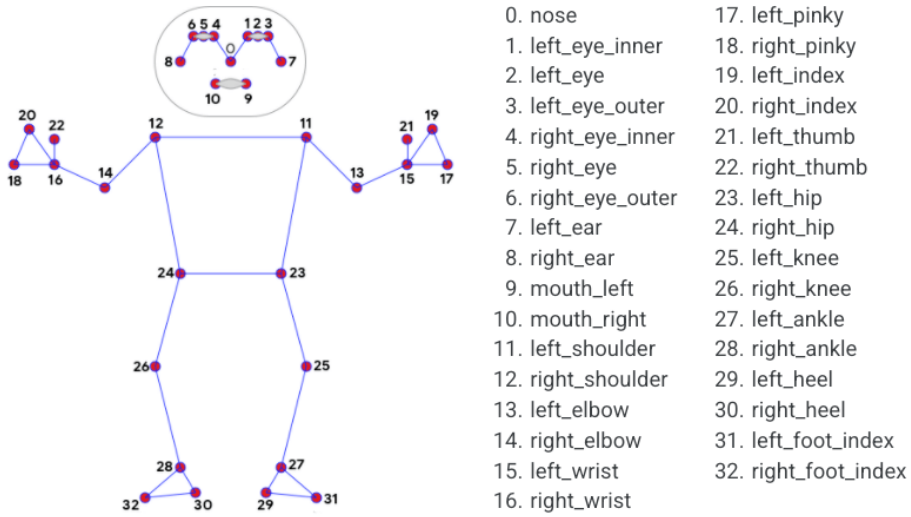

Figure 2.1: The 33 landmarks model in MediaPipe Pose predicts

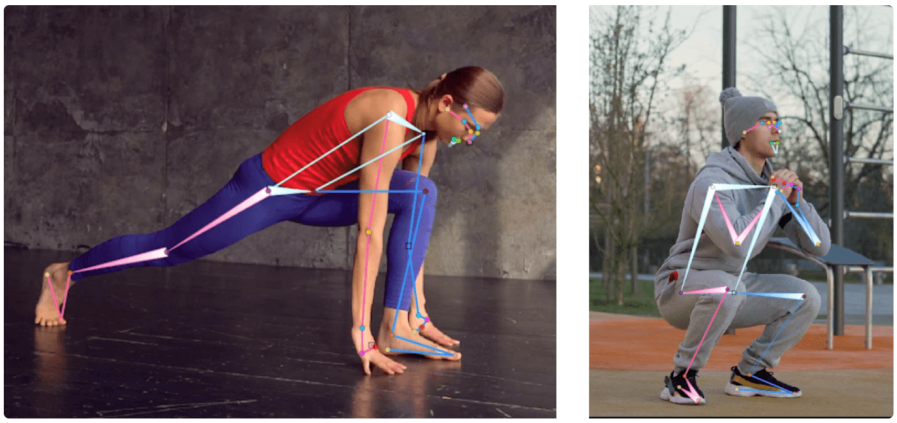

Figure 2.2: Example of pose detection by MediaPipe

2.4 Existing Systems

There are a number of AI-based fitness systems today. Nevertheless, both of them have some shortcomings that WellnessAI hopes to address.

2.4.1 Freeletics (Germany)

Freeletics is an exercise app that provides A.I. personalized training programs depending on the fitness objectives of the user. It works as progress and motivation coaching based on prearranged video workouts.

Table 2.1: Features and Description of the Freeletics App

Feature	**Description**
Developer	Freeletics GmbH (Germany)
Focus Area	AI-generated workout routines
Strengths	Personalized plans, goal tracking, user motivation
Limitations	Does not support camera-based pose correction or real-time AI feedback

2.4.2 Fitbod (USA)

The Fitbod is an AI-driven generator of gym workouts, which customizes exercises depending on muscle recovery and available equipment. It follows up using data analytics.

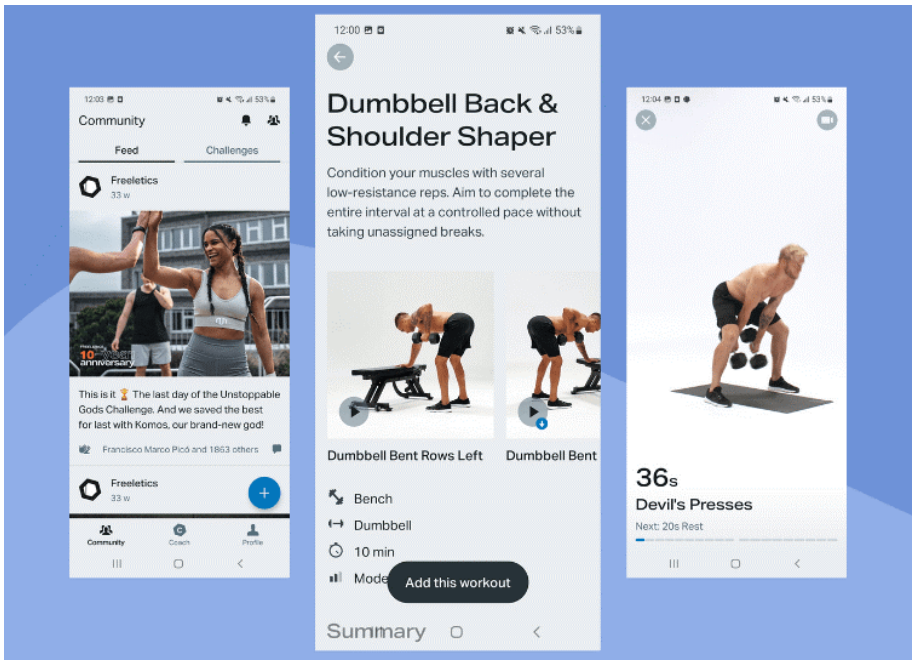

Figure 2.3: Freeletics App UI

Table 2.2: Features and Description of the Fitbod App

Feature	**Description**
Developer	Fitbod Inc. (USA)
Focus Area	Gym-based strength training
Strengths	Smart AI recommendations, progress analytics, muscle recovery tracking
Limitations	No video-based form analysis, limited home workout support

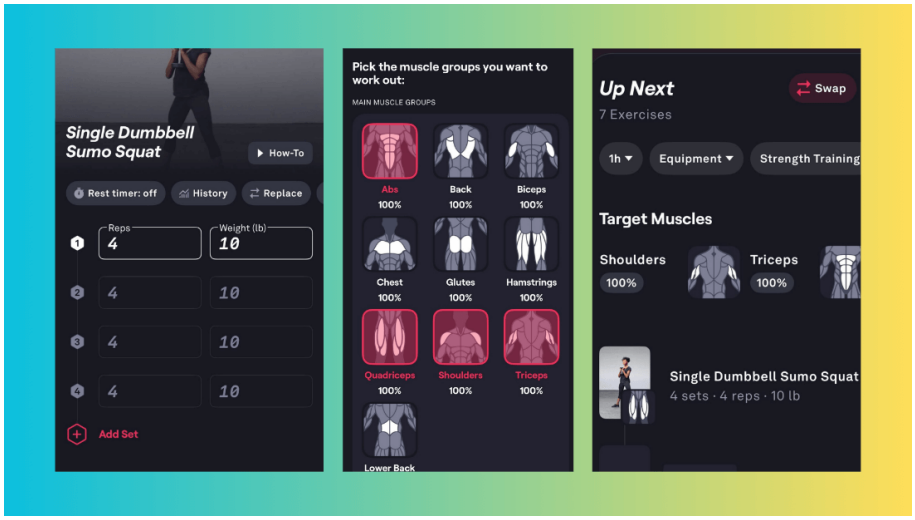

Figure 2.4: Fitbod App UI

2.4.3 Kaia Personal Trainer (Germany)

The concept of Kaia is based on computer vision with a camera on a smartphone that is used to analyze user movement and correct posture in real-time [2]. It offers performance review during exercises.

Table 2.3: Features and Description of the Kaia Personal Trainer App

Feature	**Description**
Developer	Kaia Health (Germany)
Focus Area	AI posture detection and motion tracking
Strengths	Real-time posture correction using camera
Limitations	Limited exercise types, no trainer interaction or payment system

Figure 2.5: Kaia Personal Trainer UI

2.5 Comparative Analysis and Research Gap

The table below shows a comparison of the currently used systems and the ways WellnessAI manages the shortcomings of these systems:

Table 2.4: Comparative Analysis of Existing AI-based Fitness Systems and Proposed WellnessAI

Aspect	**Freeletics**	**Fitbod**	**Kaia Trainer**	**WellnessAI (Propo System)**
Pose Detection	Does not include pose detection.	Does not include pose detection.	Provides limited camera-based pose detection.	Integrates MediaPipe for accurate 3D body mark analysis.
Real-time Feedback	Uses static videos without live feedback.	Offers data-driven routines but no live feedback.	Provides some visual guidance but limited accuracy.	Provides real-time based visual feedb using pose estimation
Trainer Interaction	No direct trainer communication feature.	No trainer interaction feature.	Lacks built-in trainer consultation options.	Includes built-in tra consultation and boo system.
Workout Categories	Basic fitness routines	Strength and gym training	Limited library	Wide variety inclu strength, flexibility, durance workouts
Role-Based Access	Single user type	Single user type	No access differentiation	RBAC with Ad Trainer, User roles
Payment Integration	Not available	Not available	Not available	Integrated payment wallet system
AI Feedback Type	Static videos	Data analytics	Basic pose analysis	Live AI feedback ML-based error detec
System Integration Level	Low	Medium	Medium	High integration (A trainers + payments)

2.6 Research Gap

Having examined the current fitness systems, it is noted that:

- The majority of the apps offer either AI-based planning or video-based tracking, but not the combination of the two.
- No such system exists where AI posture correction, interactivity with a trainer, and payment are integrated into a single platform.
- Very few systems utilize pose detection, and those which do (such as Kaia) have minimal functionality.

As such, WellnessAI fills this gap by integrating:

- MediaPipe Pose to landmark detector,
- Feedback and repetition counting machine learning models,
- In-built trainer consultation and wallet system,

to build a full exercise system among the users.

Chapter 3

Requirement Specifications

3.1 Existing System

The existing fitness and wellness ecosystem is disjointed, offering partial solutions but not providing an end-to-end experience. Current applications and solutions demonstrate major gaps in real-time posture validation, trainer interaction, and holistic workflow integration:

Fitness experience:

- There are apps like Freeletics, Fitbod, and Evolve AI that offer personalized training plans but do not support vision-based form validation.
- Peloton and Apple Fitness+ provide live classes but do not provide automated pose assessments and trainer marketplaces.
- Computer vision (CV) posture apps (e.g., Kaia) deliver camera-based feedback but do not allow booking trainers, in-app purchases, or a full end-to-end workflow [7] .

Key limitations:

- Lack of coherent flow encompassing training plans, bookings, payments, chat, reminders, and AI-based analyzing forms.
- Communication associated with bookings is not integrated with guidance and execution quality.
- Absence of explainable, real-time cues associated with measurable posture characteristics.
- Deployment limitations, such as missing HTTPS for camera access and high frame-rate WebSocket streaming for smooth feedback.

These shortcomings demonstrate the necessity of an inclusive system combining AI-driven feedback with expert advice and a safe, convenient interface.

3.2 Proposed System

To overcome the gaps in current fitness solutions, we propose **WellnessAI**, a mobile and web-based platform:

Platform Overview:

- Role-based access: Customers, Professionals (trainers), and Admins.
- Built-in features: Plan marketplace, appointment booking, in-app payments with fee distribution, chat linked to bookings, and notifications.

AI Workout Module (Web):

- Technologies: Vue.js and Django, MediaPipe Pose, pretrained sklearn models, OpenCV overlays [7] .
- Features: Stage detection, real-time webcam analysis, uploaded video processing, rep counting, interpretable feedback (e.g., foot/knee position) [2] .

Architectural Highlights:

- Mobile: Flutter, Firebase Auth/Firestore, GoRouter, Provider, Cloudinary, FCM, and local notifications.
- Web AI: sklearn .pkl models, OpenCV overlays, REST/WebSocket endpoints, MediaPipe Pose.
- Deployment: HTTPS required for camera access; VPS suggested for stable, low-latency streaming; local SSL may be used for demos.

WellnessAI brings together mobile accessibility and AI-driven posture detection to deliver a seamless and professional wellness experience.

3.3 Requirement Specifications

3.3.1 Functional Requirements

Authentication and Roles:

- Users can register/login with role selection (Customer, Professional, Admin).
- Professionals submit applications; admins approve or reject.
- Sessions persist via secure local storage.

Plans and Marketplace:

- Professionals can create, edit, and view plans (title, price, media, PDFs).
- Customers can browse, search, view details, and purchase plans.

Bookings and Availability:

- Professionals define availability; customers request and confirm appointments.
- System tracks booking status (pending, confirmed, completed, cancelled).

Payments and Wallet:

- Payments record platform fees and professional earnings.
- Email receipts sent to both customer and professional.
- Admins can view financial overviews; professionals can access wallet and earnings details.

Notifications:

- FCM tokens saved and refreshed.
- Local notifications scheduled for pre-session reminders, chat activation, session ending warnings, and completion alerts.

Chat:

- Booking-linked chat between customers and professionals.
- Accessible via dashboards and booking details.

Profiles and Admin:

- Users can update profiles (bio, goals, media).
- Admin manages trainer applications, users, and finances.

AI Workout (Web):

- Real-time webcam analysis and uploaded video analysis.
- Supported exercises: squat, plank, bicep curl, lunge.
- Outputs include stage classification, rep counting, visual overlays, and text feedback.

Data Requirements:

- Collections: users, professional_applications, plans, bookings, payments, chats, messages, wallets, withdrawals, reviews, reminders, platform_analytics.

- User fields include role, fitness profile, professional attributes (specializations, experience, approval status).

Integration Requirements:

- App launches AI module via secure URL (HTTPS).
- Shared identifiers in booking/chat context; no PHI or regulated medical data collected.

3.3.2 Non-Functional Requirements

App Non-functional Requirements:

- **Performance:** Screens load $<$2 s; chat latency $<$500 ms.
- **Reliability & Availability:** No data loss on intermittent connectivity; graceful degradation if AI module unreachable.
- **Security & Privacy:** HTTPS, Firebase Auth, role-based Firestore rules, minimal data collection.
- **Compatibility & Portability:** Flutter supports recent Android versions; web AI supports modern browsers with getUserMedia.
- **Maintainability:** Modular providers, services, and routes; configuration separated.
- **Compliance & UX:** Explainable feedback, inclusive language, responsive UI.

AI Workout Module Non-functional Requirements:

- **Performance:**
  - Pose estimation $\leq$150 ms per frame; 60 FPS target.
  - API response: $\leq$300 ms control, $\leq$800 ms analysis endpoints.
  - Cold start $\leq$15 s; warm restart $\leq$3 s.
  - Streaming jitter $\leq$50 ms; dropped frames $\leq$1% over 5 minutes.
- **Scalability:** Linear horizontal scaling; GPU optional with CPU fallback; $\geq$50 concurrent users CPU-only, $\geq$200 with GPU.
- **Reliability & Availability:** Uptime $\geq$99.5%; graceful degradation on model failure; auto-retry transient errors.
- **Security & Privacy:** TLS 1.2+, token-based auth, transient frame handling, no video storage by default.

3.4 Use Cases

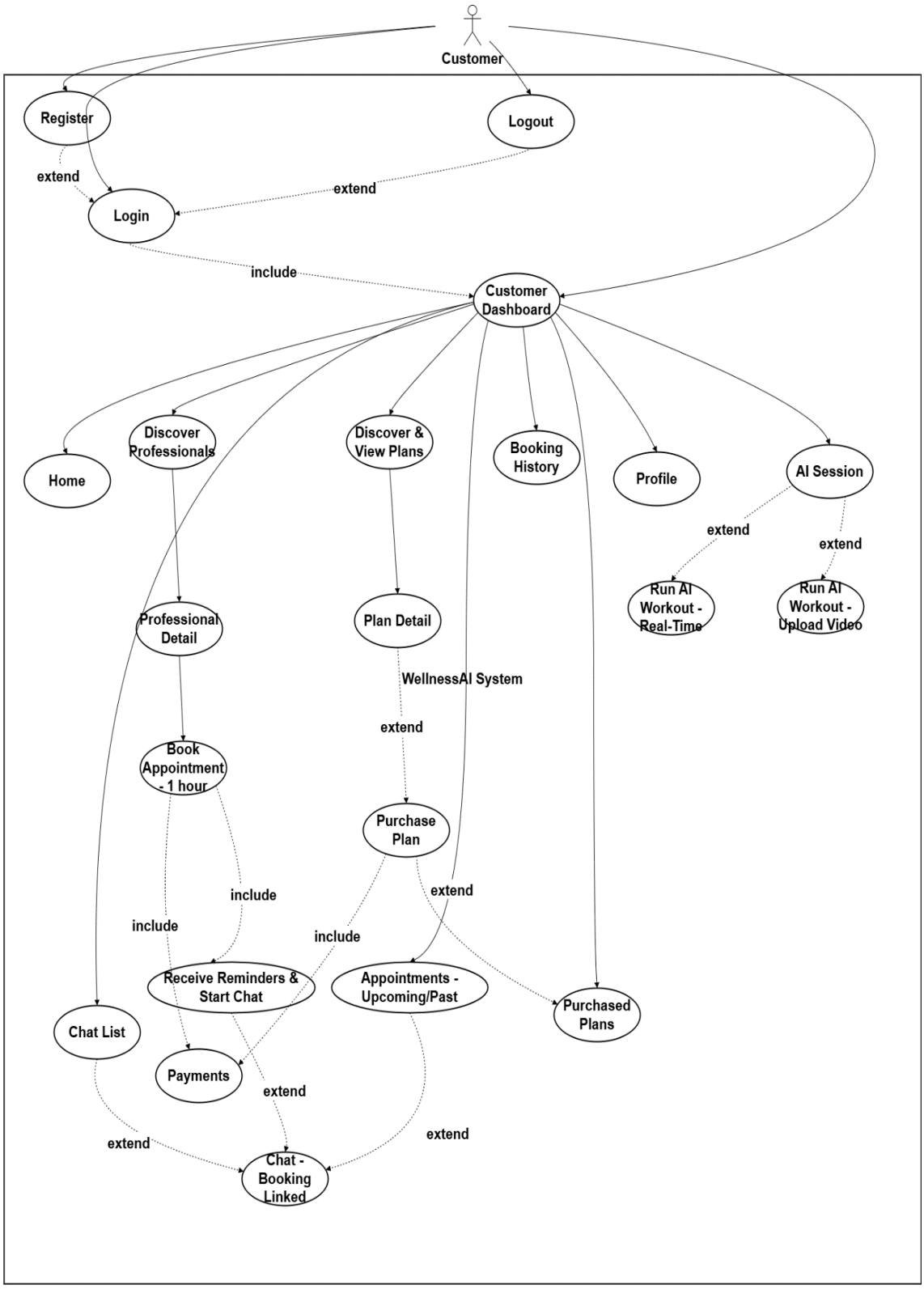

Figure 3.1: Customer Usecase Diagram

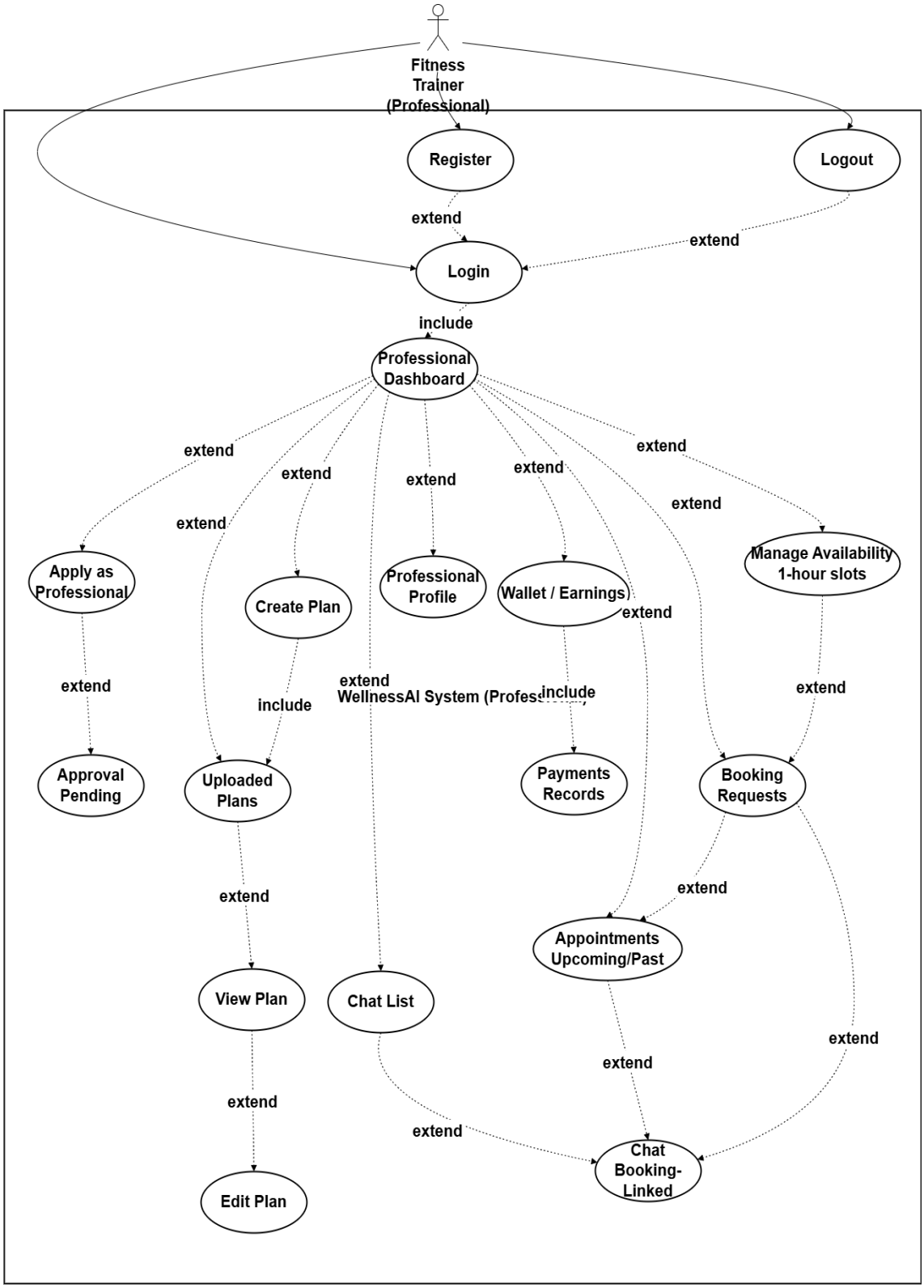

Figure 3.2: Professional Usecase Diagram

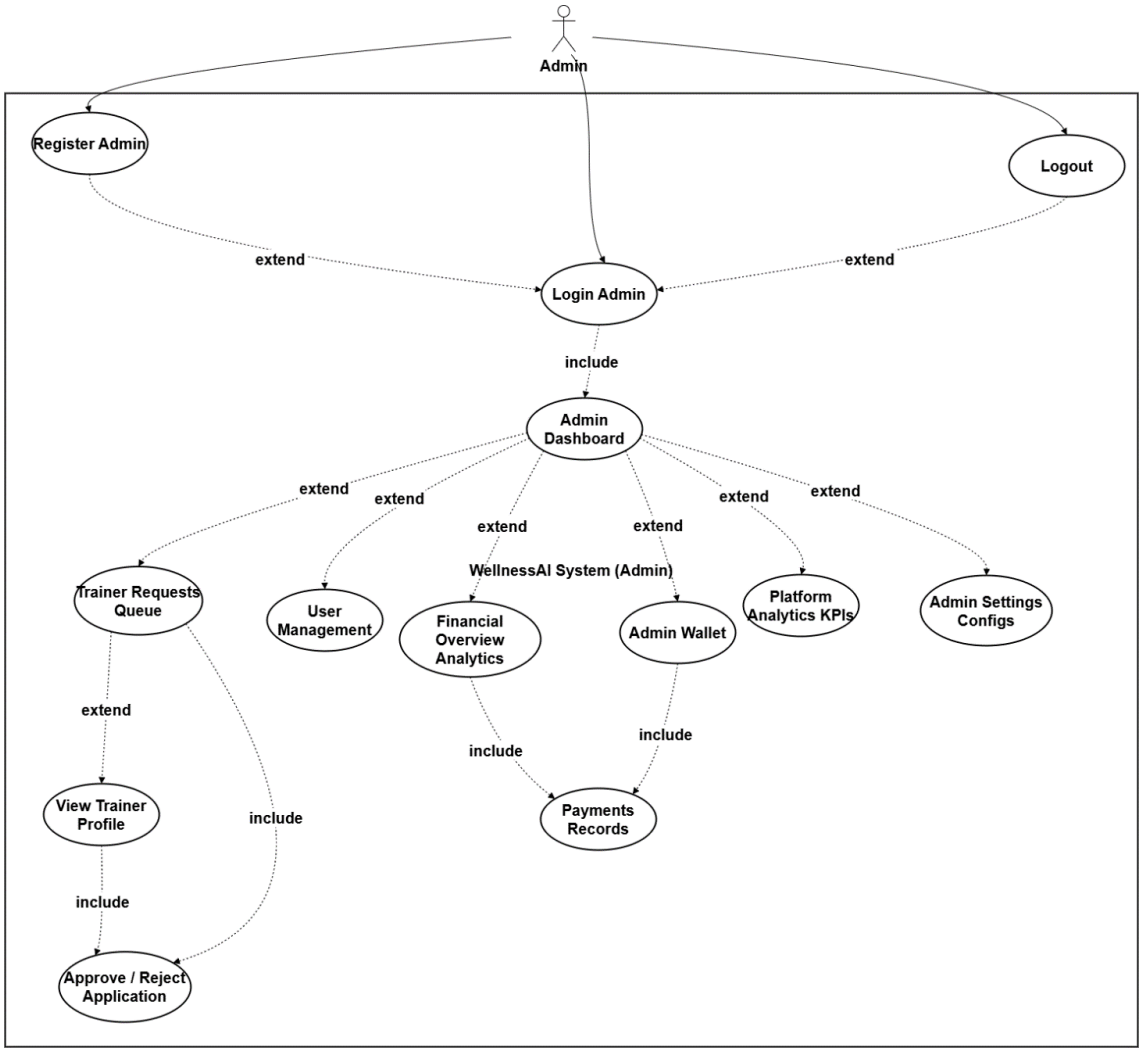

Figure 3.3: Admin Usecase Diagram

Table 3.1: UC-01 Register/Login

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-01	Customer / Professional / Admin	Authenticate and start a session with role-based landing	Auth succeeds and user lands on correct role home	App installed; network available; account exists (or register)	Open app → Enter credentials → Validate → Success → Route to role home

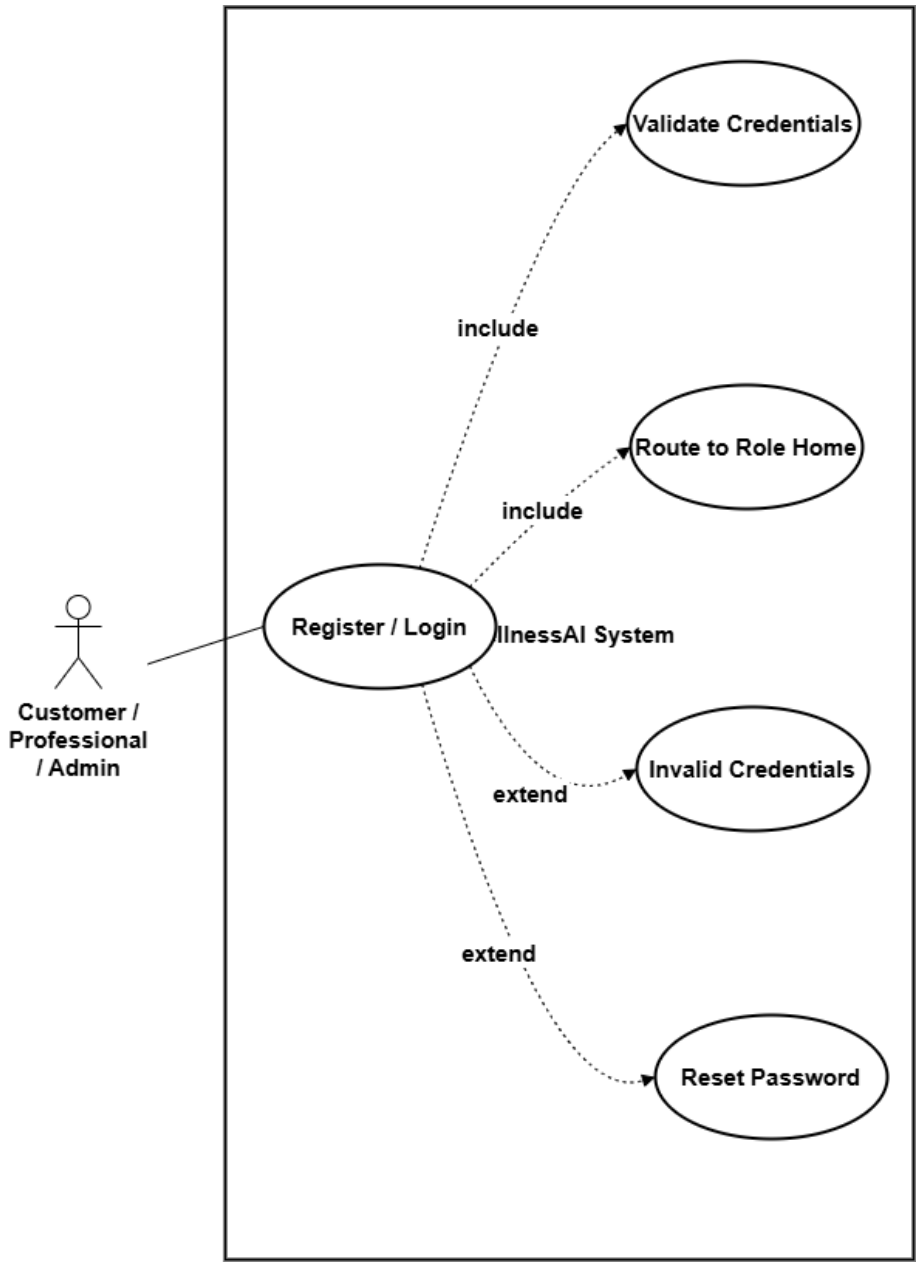

Figure 3.4: UC-01 Register/Login

Table 3.2: UC-02 Apply as Professional

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-02	Professional	Submit trainer application and upload certificate	Application stored as pending for admin review	App installed; Professional registered but awaiting admin approval	Open form → Fill details → Attach certificate → Submit → Status pending

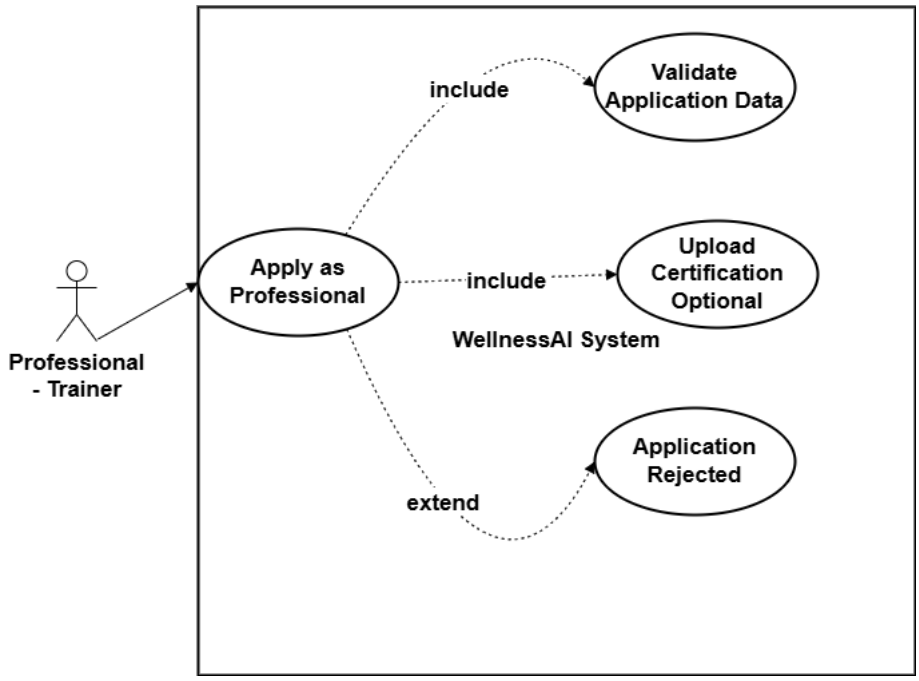

Figure 3.5: UC-02 Apply as Professional

Table 3.3: UC-03 Approve/Reject Professional

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-03	Admin	Moderate trainer applications	Applicant notified; role/status updated	Logged in as admin; pending applications exist	Open queue → Review → Approve/Reject → Update role/status → Notify applicant

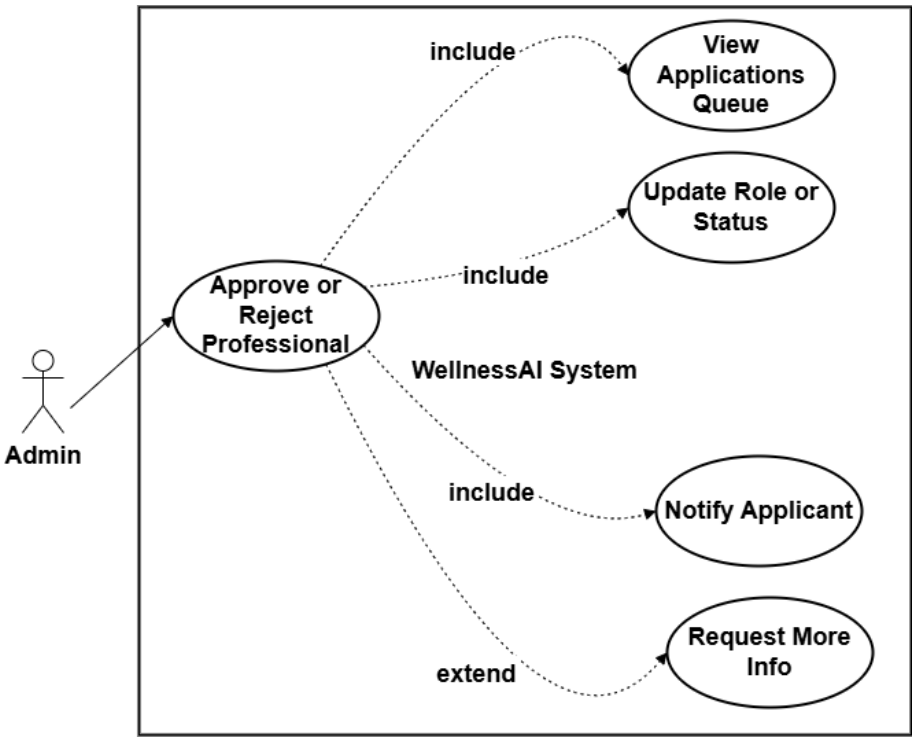

Figure 3.6: UC-03 Approve/Reject Professional

Table 3.4: UC-04 Create/Edit Plan

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-04	Professional	Publish or update a plan with media and price	Plan saved and visible if published	Logged in as approved professional	New/Edit → Enter details → Upload media → Save/Publish

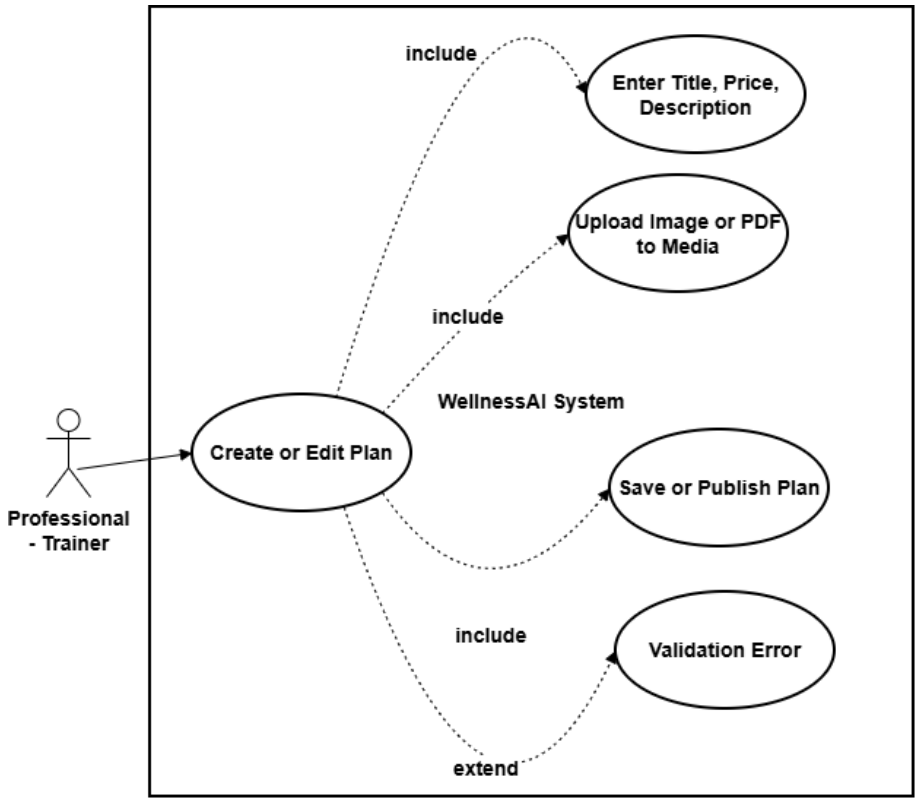

Figure 3.7: UC-04 Create/Edit Plan

Table 3.5: UC-05 Browse and Purchase Plan

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-05	Customer	Discover and purchase a plan	Payment recorded; plan in library; receipt sent	Logged in; payment method ready	Browse/Search → Open details → Pay → Receipt email → Access in library

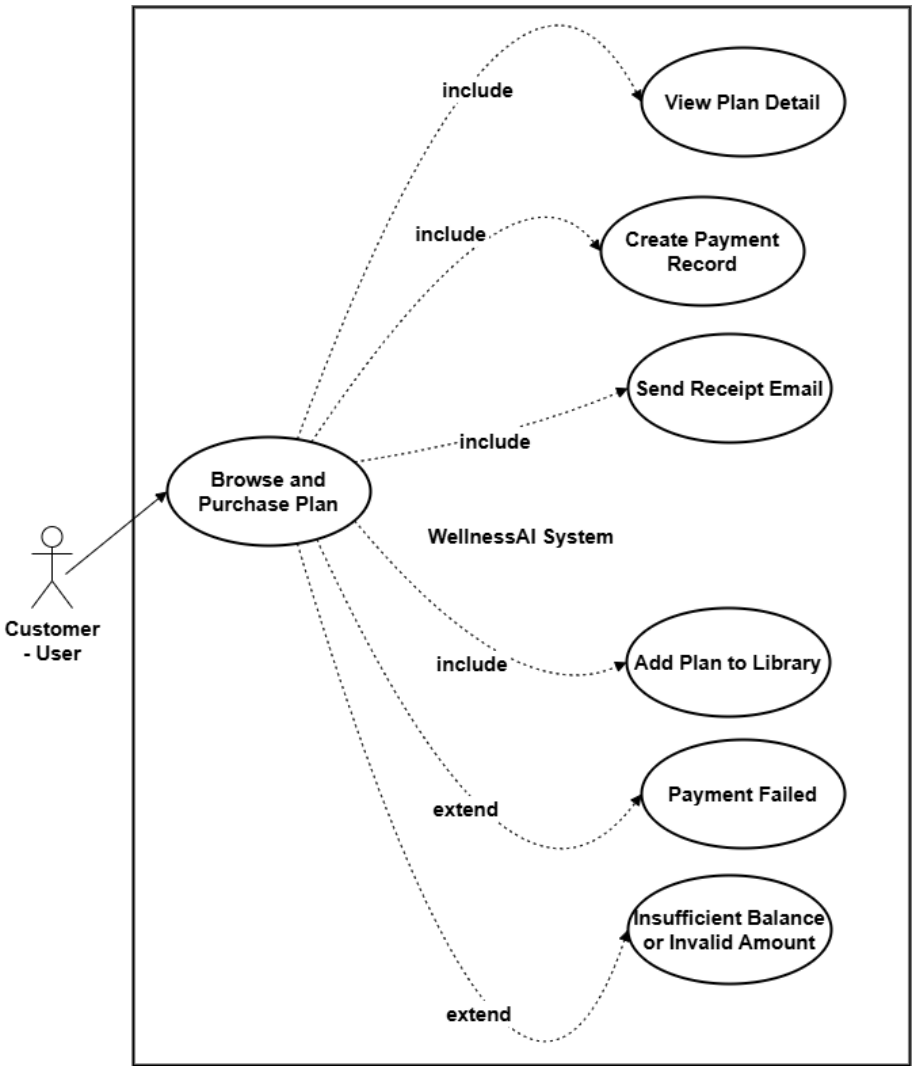

Figure 3.8: UC-05 Browse and Purchase Plan

Table 3.6: UC-06 Set Availability and Manage Booking Requests

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-06	Professional (Trainer)	Define weekly 1-hour availability slots, set 1-hour rate, and manage booking lifecycle	Availability saved to professional_availability; bookings move from Pending → auto-Confirmed → Completed	Logged in as approved professional; network available	1) Open Availability Management → toggle hourly slots for Mon–Sun → set one-hour rate → Save 2) Open Booking Requests → filter by status → view details → tap "Mark Complete" when booking confirmed

Figure 3.9: UC-06 Set Availability and Manage Booking Requests

Table 3.7: UC-07 Book Appointment (Customer)

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-07	Customer	Schedule a 1-hour appointment with a professional using available calendar slots	Booking created (pending), confirmation emails sent, routed to payment	Logged in; professional has published availability; network reachable	1) Load professional availability 2) Select date 3) Load available time slots 4) Validate inputs 5) Check active booking 6) Re-check slot 7) Create booking (pending) 8) Send confirmation emails 9) Navigate to payment screen

Figure 3.10: UC-07 Book Appointment (Customer)

Table 3.8: UC-08 Receive Reminders and Start Chat

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-08	Customer / Professional	Timely nudges + session communication	All reminders delivered; chat available at start	Booking exists; notifications permitted; FCM token saved	15-min reminder → Chat activation → Ending warning → Completion notice

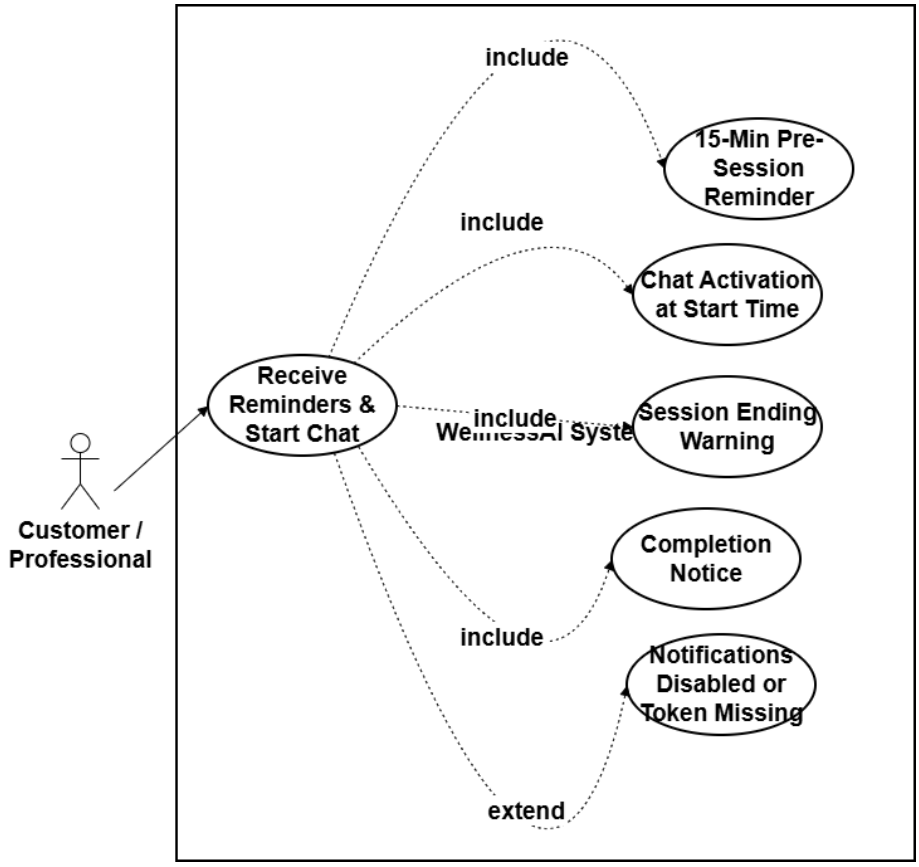

Figure 3.11: UC-08 Receive Reminders and Start Chat

Table 3.9: UC-09 Chat (Booking Linked)

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-09	Customer / Professional	Booking-context chat using chatId; messages persist	Messages saved; last message/metadata updated; both can view/send	Authenticated; valid booking/chat exists; route provides chatId	1) Open chat list → select conversation 2) Load chat by chatId 3) Send/receive messages; persist to Firestore 4) Update last message/time 5) Return to chat list

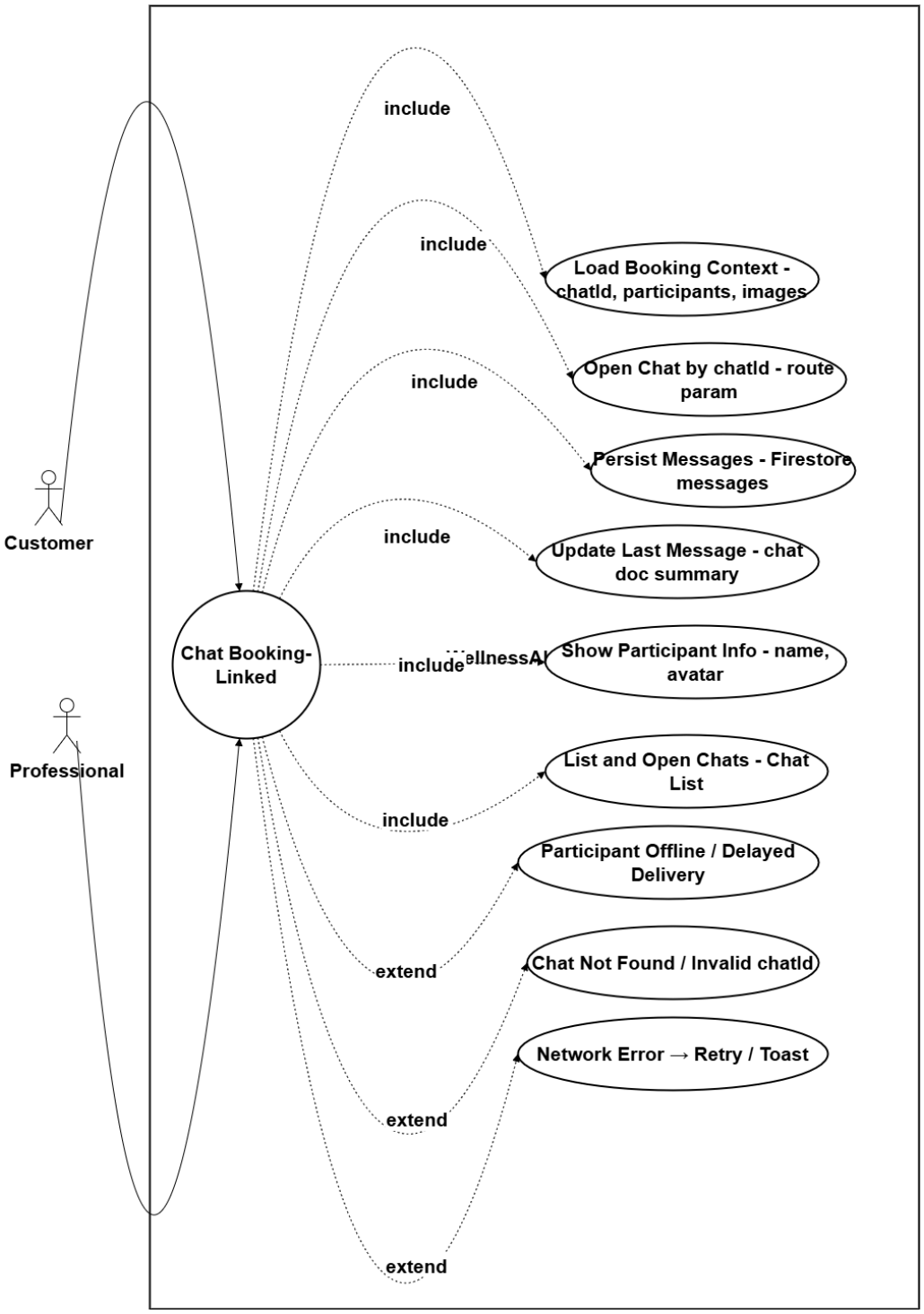

Figure 3.12: UC-09 Chat (Booking Linked)

Table 3.10: UC-10 Run AI Workout — Real-Time

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-10	Customer (User)	Run web AI workout in real-time; webcam frames sent to server for MediaPipe Pose analysis, feedback, rep counting, and overlays	Stable low-latency loop; overlays render smoothly, feedback updates continuously	Access HTTPS web app; camera permission granted; modern browser; stable network	1) Open AI workout route 2) Grant camera permission 3) Select exercise 4) Client encodes frames → server 5) Server: MediaPipe Pose + feedback 6) Return processed frames + feedback 7) UI updates until session ends

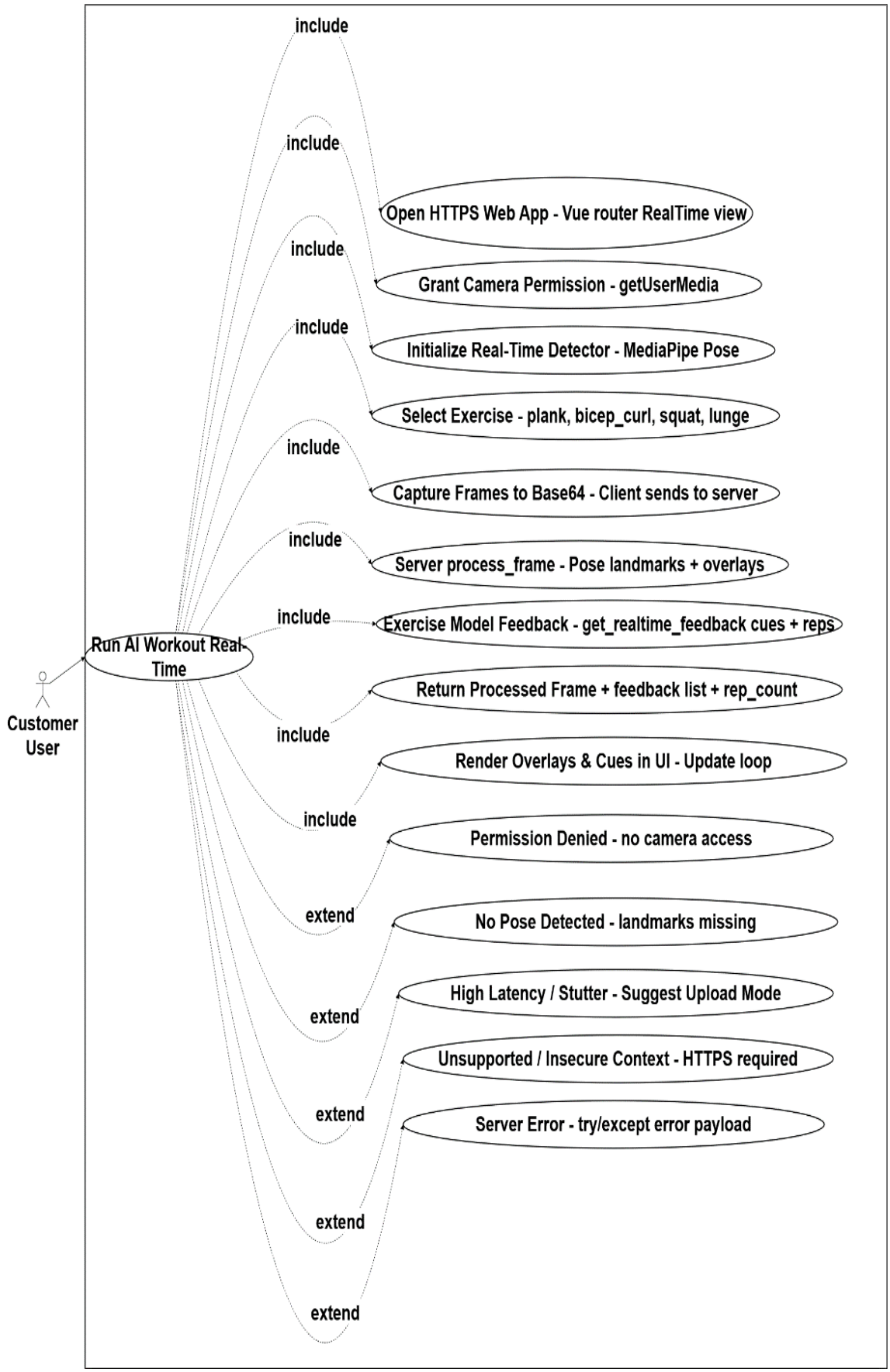

Figure 3.13: UC-10 Run AI Workout — Real-Time

Table 3.11: UC-11 Run AI Workout — Upload Video

Use Case ID	Actor	Description	Main Success Factor	Precondition	Basic Flow
UC-11	Customer (User)	Analyze recorded video in web AI; server runs MediaPipe Pose and models, returns rep counts, overlays, feedback summary	Results page shows evidence frames and clear feedback summary for the chosen exercise.	Access HTTPS web app; compatible video file; stable network	1) Open AI upload view 2) Select exercise and upload clip 3) Server validates and saves file 4) Process frames with MediaPipe Pose Run exercise model to compute reps/errors 6) Generate evidence frames with overlays 7) Return feedback summary + evidence list 8) UI displays results.

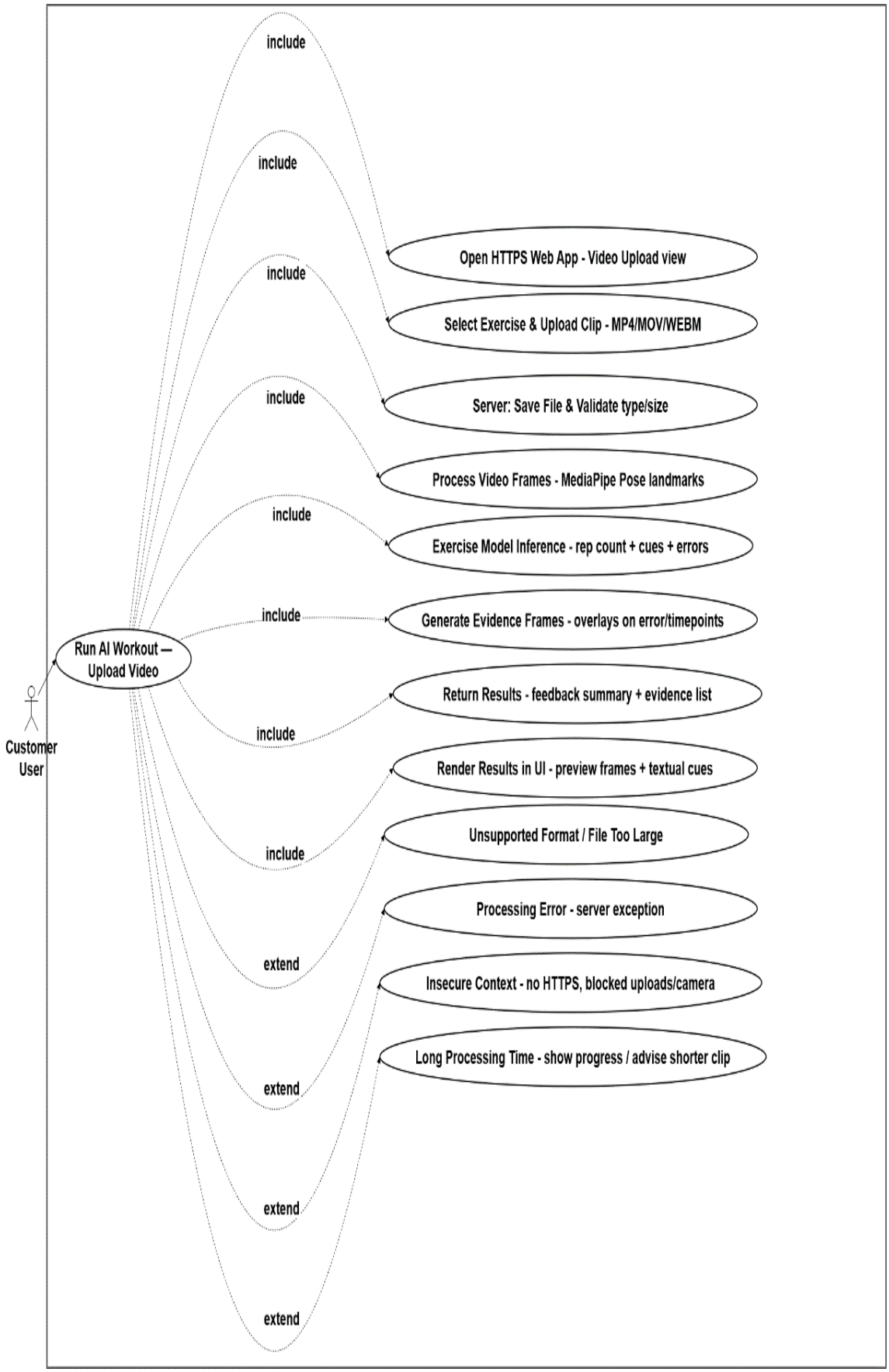

Figure 3.14: UC-11 Run AI Workout — Upload Video

Chapter 4

Design

Systems design is the process of defining a system's architecture, components, modules, interfaces, and data to satisfy specified requirements. In this chapter, we will give a complete design of the WellnessAI system that consists of both the mobile application and the AI Workout web module.

4.1 System Architecture

WellnessAI is a hybrid wellness platform consisting of a mobile application that connects the consumer to an external web-based AI workout analysis platform. The mobile application serves as the primary entry point where consumers discover fitness programs, schedule appointments with professionals, make payments, and communicate through chat [10] . The professionals will interact with the mobile application to create workout plans, manage their available schedules, fulfill requests, and review their earnings. The administrators will utilize the application to oversee the entire platform by leveraging features such as user approval of trainers, user management, overseeing financial back-end reports on transactions, etc.

The system architecture is divided into two subsystems: the Mobile Application (built in Flutter) [10] , and the AI Workout Web Application (built in a Vue.js framework for the front end and Django for the back-end) [11] . The application connects to Firebase services using the mobile application for authentication, data storage, notifications, and media management. The web module for the AI performs independently but will be launched from the mobile app through a secure HTTPS link that will allow the app to provide feedback on the users' form in real-time based on computer vision and machine learning models [12] .

Both systems communicate with external systems such as Firebase, Cloudinary (for media storage), email services, and notifications services. The AI web module must use HTTPS to access the camera on the users' devices and process videos based on MediaPipe

Pose detection and pretrained machine learning models for the users' instant response [12]
.

4.1.1 High-Level System Architecture Diagram

The diagram below shows the entire system architecture with the mobile and web applications placed one on top of the other and connected by the AI Session Launcher.

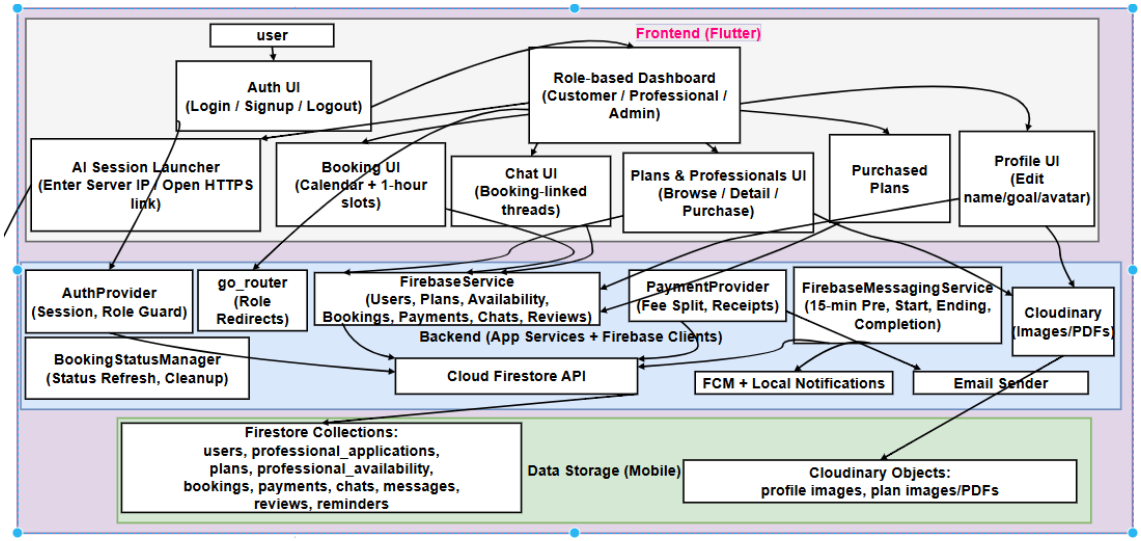

Figure 4.1: Mobile App Architecture Diagram

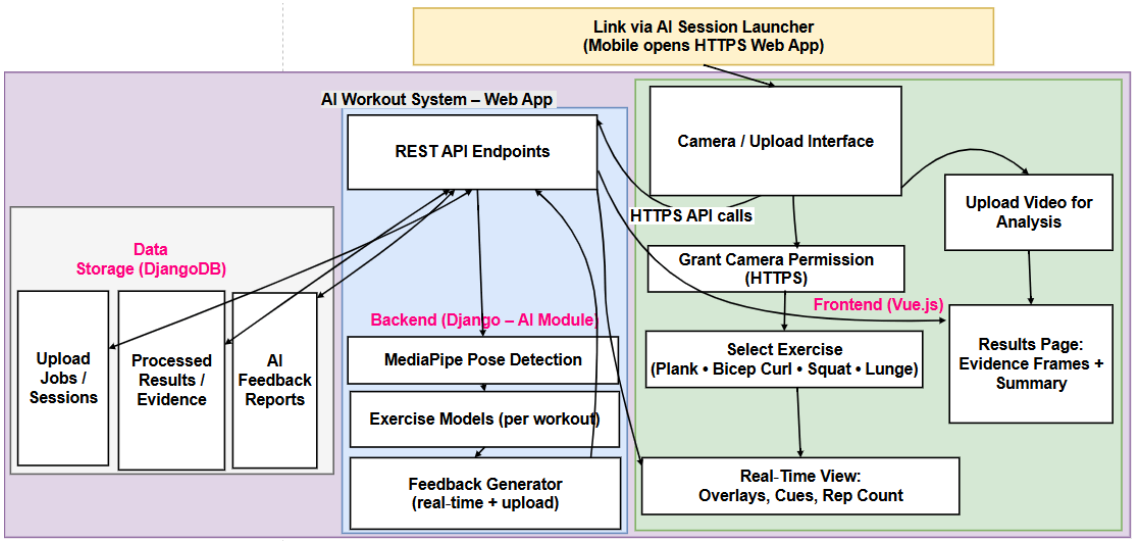

Figure 4.2: Web App (AI Workout System) Architecture Diagram

4.1.2 Context Diagram

The context diagram shows how WellnessAI interacts with external actors and systems.

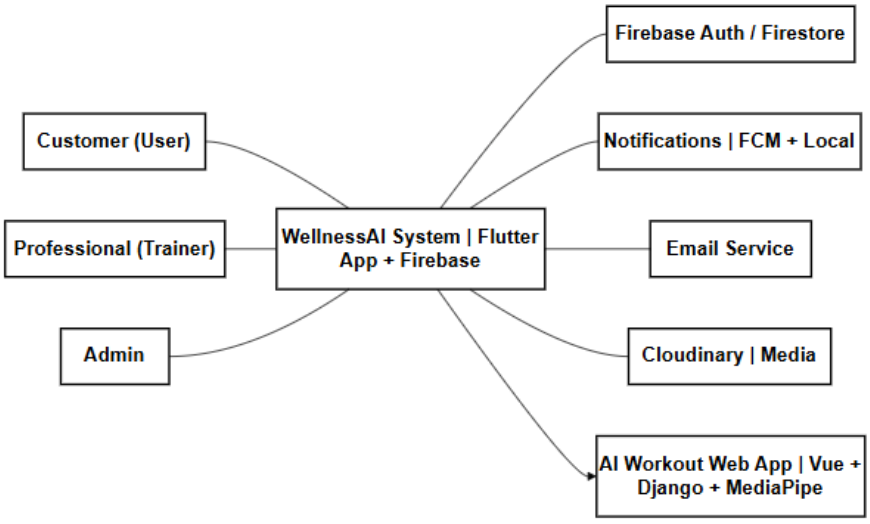

Figure 4.3: Context Diagram (WellnessAI System)

4.2 Design Limitations

4.2.1 Hardware and Platform Limitations

The mobile application must be compatible with Android devices of varying hardware capabilities. If a mid-range device is selected the CPU/GPU and RAM will be limited, affecting performance on complex UI rendering and background processing. The AI web module will also require a modernized browser that will have camera access or is not locked down/standalone which may not have camera/microphone permissions or a proper way to upload and handle larger video downloads since the mobile version will be ran through the web.

4.2.2 Infrastructure and Budget Constraints

To do Real-time AI workout analysis, the analysis will require high-frequency frame streamings (60-120 FPS) to allow seamless operations over a WebSocket connection. Once a production-level deployment is developed, either a VPS (Virtual Private Server), or some other cloud hosting provider must be leveraged to support a valid SSL (Secure Sockets Layer) certificate that would cover substantial to minimal costs on a monthly basis. The application utilizes Firebase's managed services with some constraints in pricing which is based on read/write operations making it important to consider the most efficient access patterns.

4.2.3 Software and Architectural Constraints

The appointment booking system is limited to utilizing the highly-fixed block of time for 1-hour duration time slots to simplify the scheduling logic and reduce complexity from other durations. The data model that is used will contain denormalized Firestore documents to limit read operations and improve performance in a mobile device's typical use case but must also ensure proper synchronized updates for when data does change. Background processing for reminders will be limited by the constraints of utilizing mobile OS devices requiring polling and running cycles, rather than a true background service in this context.

4.2.4 Operational and Deployment Restrictions

To enable camera access, the AI web module must run over HTTPS (exemption described below for local development), as the getUserMedia API requires this. Real-time pose detection performance requires latency to be low (<200ms end-to-end), which can limit acceptable distance between network clients and servers, as well as the time it takes to process in servers. Each in-progress notification means best-effort delivery will result in a notification lost at the OS level (potentially a user permission issue, or a throttling matter).

4.2.5 Trade-offs Made

Real-time Accuracy vs. Performance: In lieu of deep learning models, real-time accuracy is achieved through a hybrid measure, found in the combination of MediaPipe Pose detection with lightweight pretrained models in tandem with checks based on geometric rules (as opposed to only using deep learning-based methods), to have the system keep real-time performance along less intensive hardware.

Simplicity vs. Flexibility: One-hour appointments are considerably more simple to book and can minimize race conditions (in the sense of concurrent state changes) than if variable appointments were instead implemented to improve flexibility of a booking system for a user.

Cost vs. Capability: Firebase's serverless design (or cloud computing), while allowing the majority of the possible hosting options to take care of the deployment overhead, reduces capabilities to do transactional logic or similar behaviors that require the complexity of reliable custom backend (code) logic.

Security vs. Usability: Enforcing HTTPS and role-based access control (RBAC) adds friction for development and/or local test (implying the need to manage a certificate), but nevertheless means secure access to camera devices and protection of user data.

4.2.6 Assumptions

1. Users will grant necessary permissions, such as camera and microphone for the AI module and notifications for reminders.
2. Users possess a stable internet connection for authentication, booking, payments, and chat.
3. Professionals will keep their availability up-to-date.
4. Deployment of the production environment is using VPS and cloud hosting with valid SSL certification.
5. Email delivery and push notifications will be reliable enough for confirmations and reminders.

4.3 Design Methodology

The WellnessAI system employs an object-oriented design methodology based on UML modeling. The design incorporates a layered architecture that contains the following layers: presentation (Flutter UI), business layer (providers and services), and data access layer (Firebase clients). The external AI module adopts a service-oriented architecture, using REST APIs to link the Vue.js frontend to the Django backend.

4.3.1 Design Process

The design of the system was iterative and involved a requirements analysis, use case modeling, and system component identification. We designed modules to be loosely coupled but tightly cohesive, employing SOLID design principles where possible.

4.3.2 Modeling Techniques

During the design process, we modeled using UML diagrams. This included use case diagrams, sequence diagrams, activity diagrams, component diagrams, and class diagrams. Our database design modeled data structures using an Entity-Relationship Diagram (ERD).

4.3.3 Conventions

- All Flutter code follows Dart style guidelines, including clear separation of concerns (screens, providers, services, models, widgets).
- Firebase collections use `snake_case` naming and Flutter models use `camelCase`.
- The AI module follows Django REST framework conventions for API endpoints.

- All external integrations use HTTPS with appropriate authentication and error handling.

4.3.4 Design Patterns

The mobile application uses the Provider pattern for state management, the Repository pattern for abstracting data access, and the Factory pattern for model creation. The AI module makes use of the Strategy pattern for each alternative algorithm for detecting exercise.

4.4 High-Level Design

4.4.1 Conceptual/Logical View (Component Diagram)

The logical view organizes system functionality into major components.

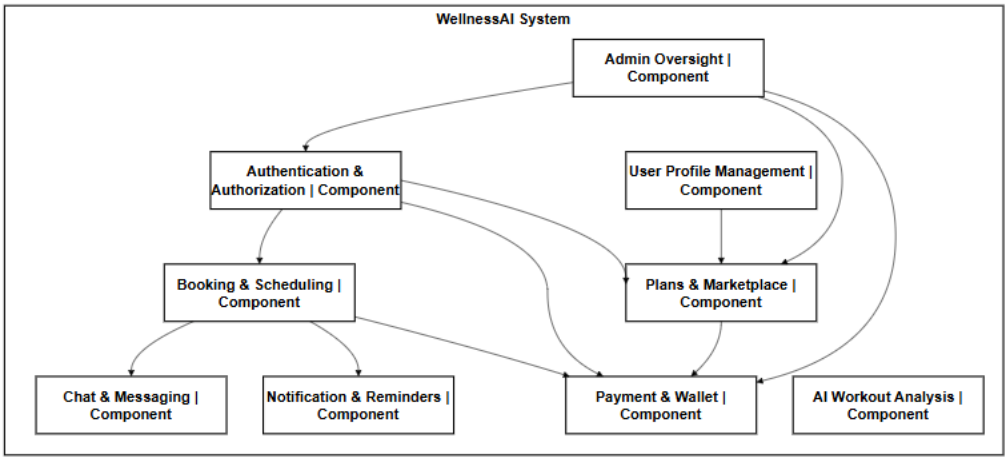

Figure 4.4: Conceptual/Logical View (Component Diagram)

4.4.2 Physical View (Deployment Diagram)

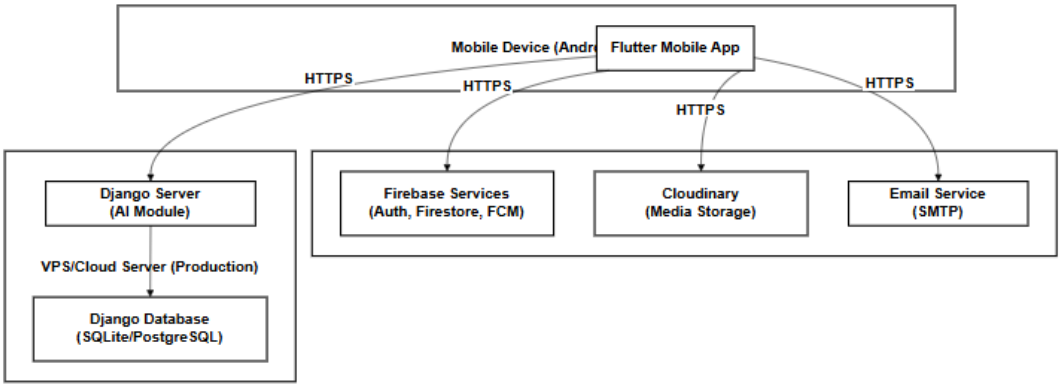

Figure 4.5: Physical View (Deployment Diagram)

4.4.3 Security View

The security architecture addresses authentication, authorization, data protection, and secure communication.

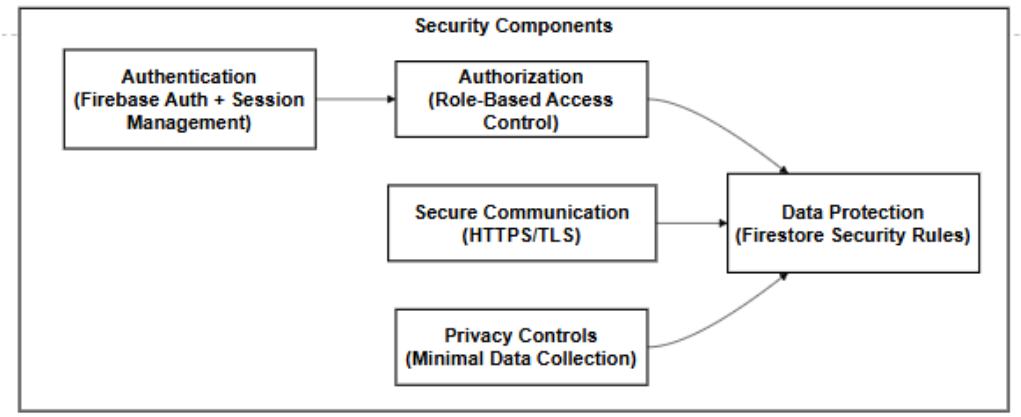

Figure 4.6: Security View Diagram

Authentication: User credentials are handled by Firebase Authentication and secured with hashed passwords and session tokens. Sales authorization is handled by role-based routing (go_router) and enforced by Firestore Security Rules.

Data Protection: All sensitive data is encrypted in transit (using HTTPS), and stored in Firestore (with Firestore encryption).

Secure Communication: All communication with external APIs is done through HTTPS, including camera access in the AI module.

Privacy: The system collects minimal information and allows users to control what information is shared.

4.4.4 Sequence Diagram

A sequence diagram is a graphical representation that illustrates the step-by-step flow of interactions between different components of WellnessAI.

4.4.4.1 User Registration (Customer/Professional/Admin)

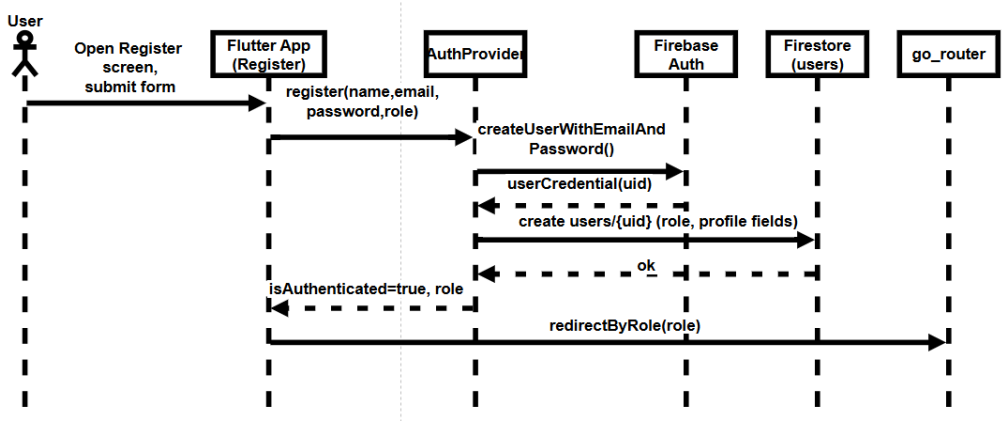

Figure 4.7: Sequence Diagram: User Registration (Customer/Professional/Admin)

This diagram illustrates the step-by-step user registration in WellnessAI, from user input to Firebase account creation and user document write. On success, the app stores role data and redirects the user to the correct dashboard.

4.4.4.2 Authentication & Role Management

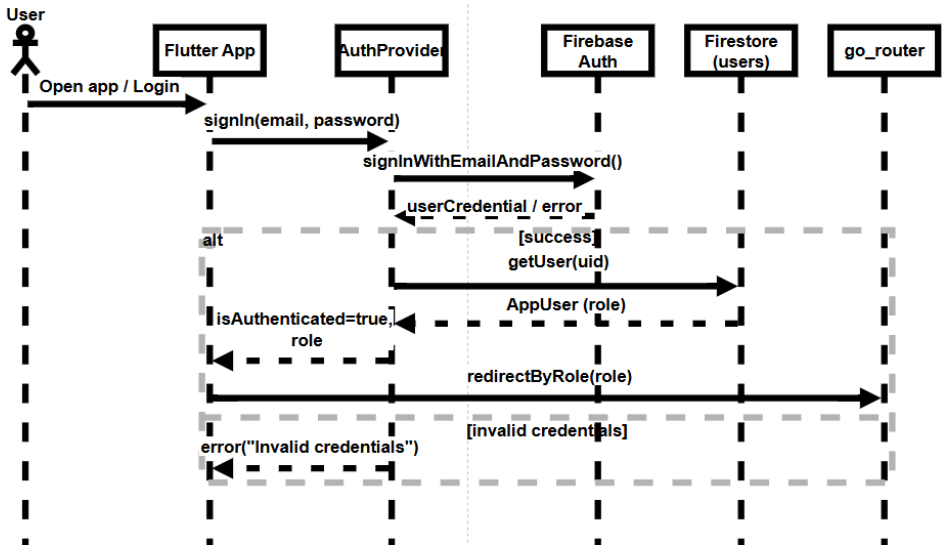

Figure 4.8: Sequence Diagram: Authentication & Role Management

4.4.4.3 Professional – Apply as Trainer

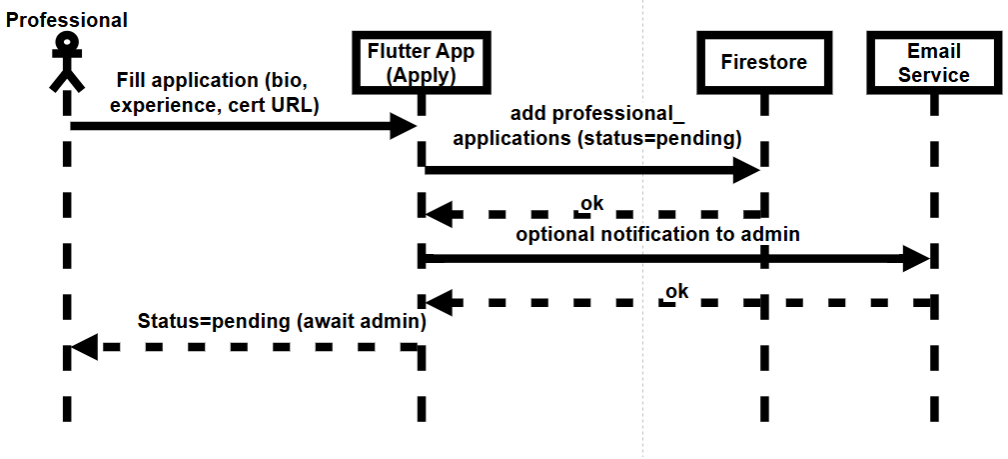

Figure 4.9: Sequence Diagram: Professional – Apply as Trainer

This diagram shows submission of a trainer application that becomes pending for admin review. Optional email notification can inform admins.

4.4.4.4 Admin – Approve/Reject Professional

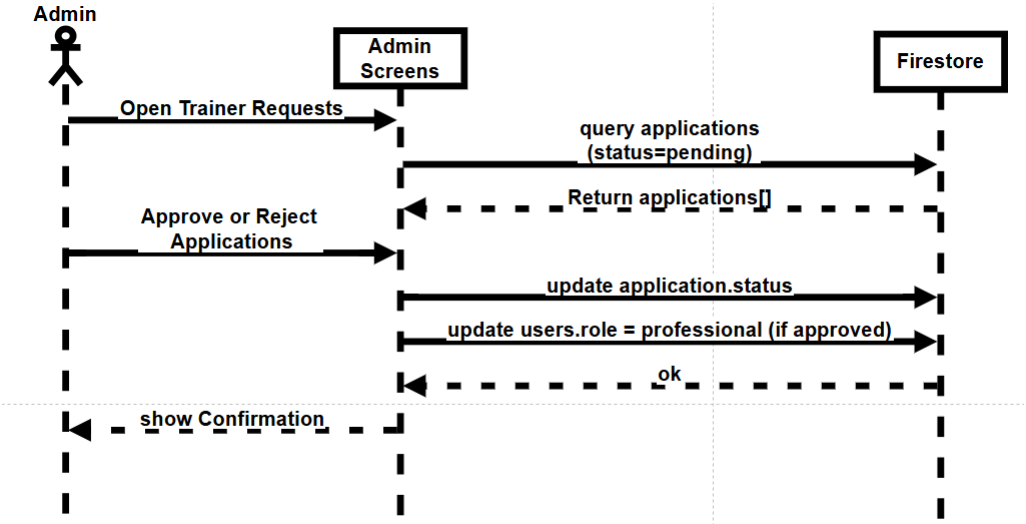

Figure 4.10: Sequence Diagram: Admin – Approve/Reject Professional

This diagram shows the moderation workflow that updates both the application and the user's role. On approval, the trainer gets access to professional features.

4.4.4.5 Customer – View & Purchase Plan

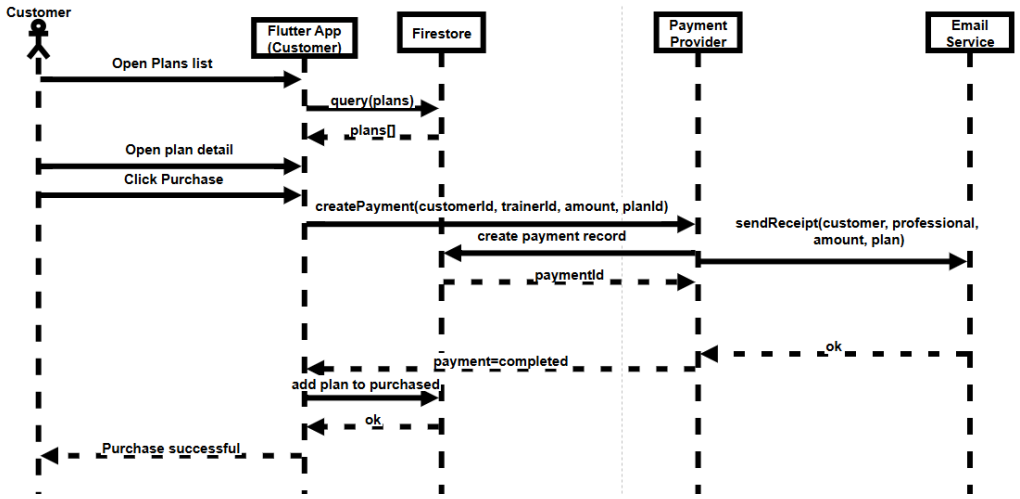

Figure 4.11: Sequence Diagram: Customer – View & Purchase Plan

This diagram illustrates a customer browsing and purchasing a fitness plan in WellnessAI. The customer browses the list of plans, selects a plan, and proceeds to purchase. The system creates a payment record in Firebase, emails receipts, and updates purchased plans. In the end, the customer is notified that the purchase was successful.

4.4.4.6 Booking & Scheduling (1-hour)

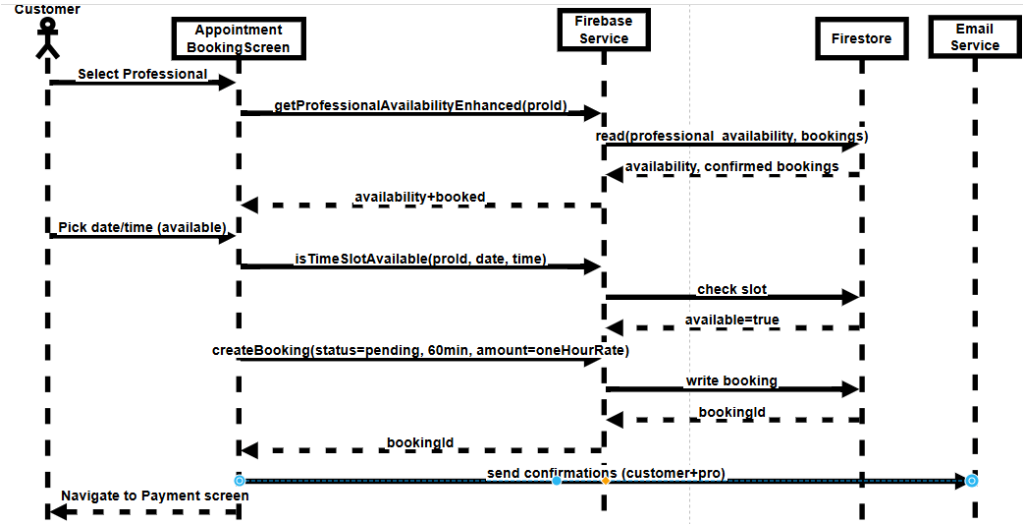

Figure 4.12: Sequence Diagram: Booking & Scheduling (1-hour)

This diagram illustrates how a customer schedules a 1-hour appointment in WellnessAI. The customer selects a professional, chooses an available date and time, the system verifies

the slot, creates a pending booking in Firestore, sends confirmation emails to both customer and professional, and then navigates the customer to the payment screen.

4.4.4.7 Chat (Booking-Linked)

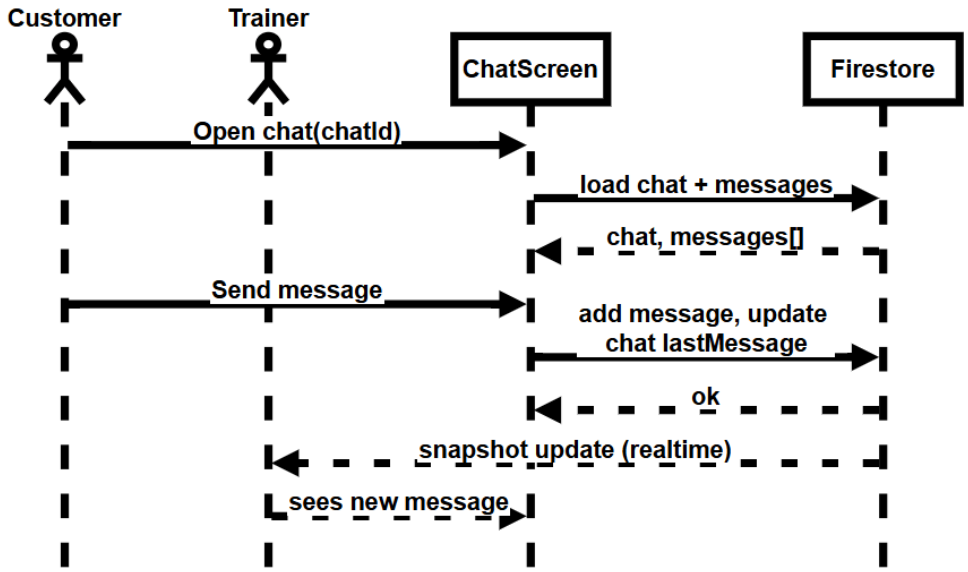

Figure 4.13: Sequence Diagram: Chat (Booking-Linked)

This diagram illustrates how a booking-linked chat works in WellnessAI. The customer opens the chat, loads previous messages from Firestore, sends a new message, which updates Firestore and triggers a real-time update for the trainer. The trainer sees the new message instantly, enabling seamless communication between customer and professional.

4.4.4.8 Payments & Wallets

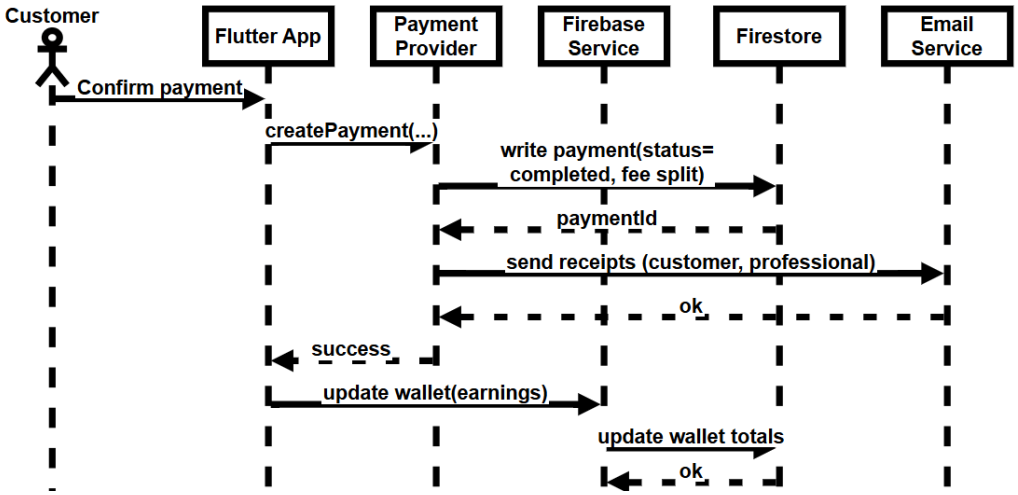

Figure 4.14: Sequence Diagram: Payments & Wallets

This diagram shows the payment and wallet workflow in WellnessAI. When a customer confirms a payment, the app interacts with the payment provider to create the payment, record it in Firestore with fee allocation, and send receipts to both customer and professional. After successful payment, the app updates wallet balances in Firebase, ensuring accurate earnings tracking for professionals.

4.4.4.9 Notifications & Reminders

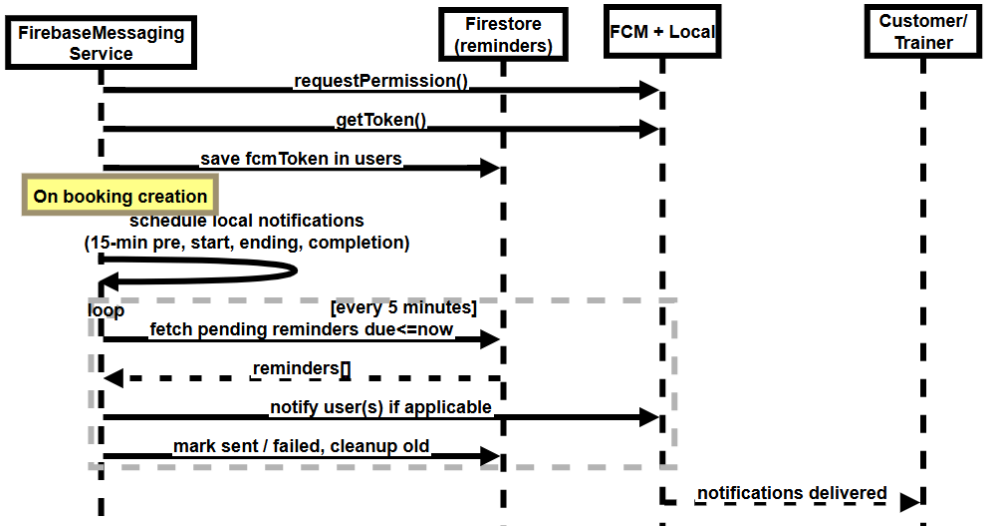

Figure 4.15: Sequence Diagram: Notifications & Reminders

This diagram illustrates how WellnessAI handles notifications and reminders. The app registers for FCM, stores the token in Firestore, and schedules local notifications for key

events (pre-session, start, ending, completion). Periodically, it fetches due reminders, sends notifications via FCM, and updates Firestore to track delivery status, ensuring timely alerts for customers and trainers.

4.4.4.10 Professional – Plans & Availability

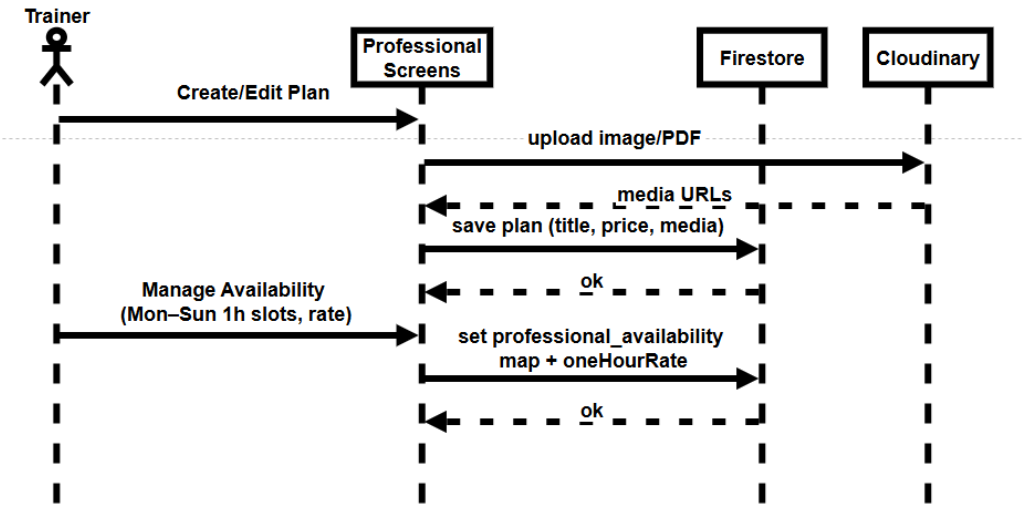

Figure 4.16: Sequence Diagram: Professional – Plans & Availability

This diagram shows how a professional (trainer) manages their plans and availability in WellnessAI. The trainer creates or edits plans by uploading media to Cloudinary and saving plan details in Firestore. They also set their weekly availability in 1-hour slots along with rates, which are stored in Firestore for customers to book.

4.4.4.11 Media Upload (Profiles/Plans)

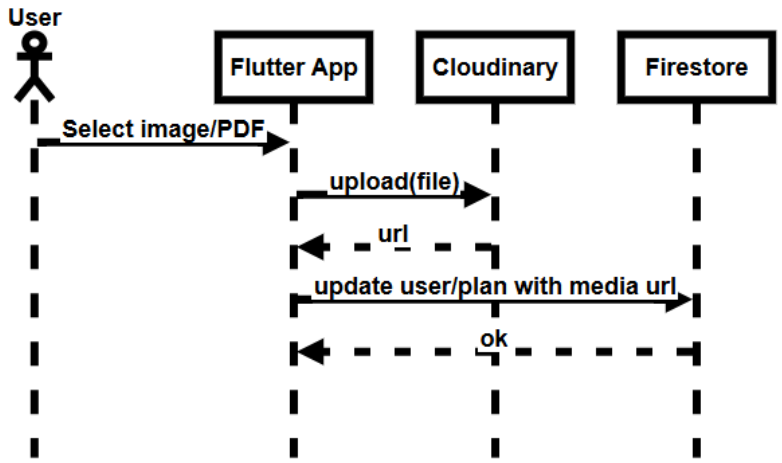

Figure 4.17: Sequence Diagram: Media Upload (Profiles/Plans)

This diagram illustrates how users (customers or professionals) upload media in WellnessAI. The user selects an image or PDF, which the app uploads to Cloudinary. The returned media URL is then saved in Firestore under the user profile or plan, making it available for display within the app.

4.4.4.12 AI Workout – Real Time (Web)

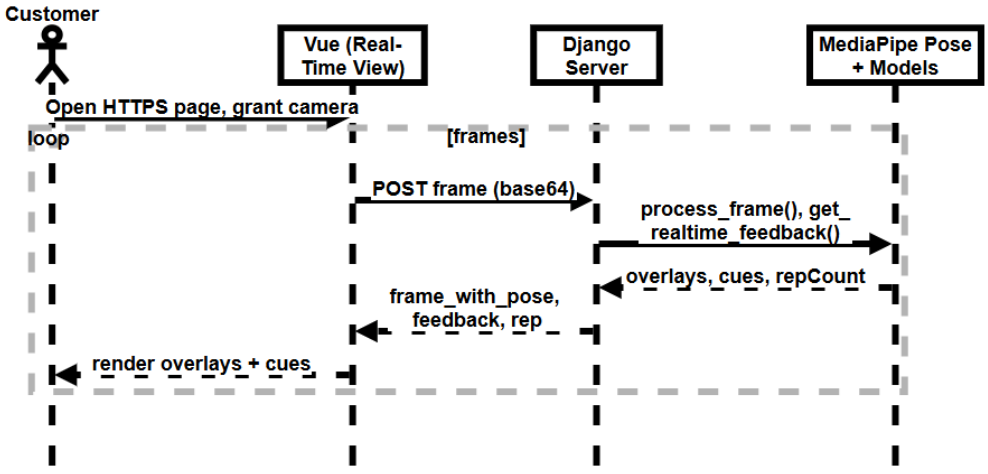

Figure 4.18: Sequence Diagram: AI Workout – Real Time (Web)

4.4.4.13 AI Workout – Upload Video (Web)

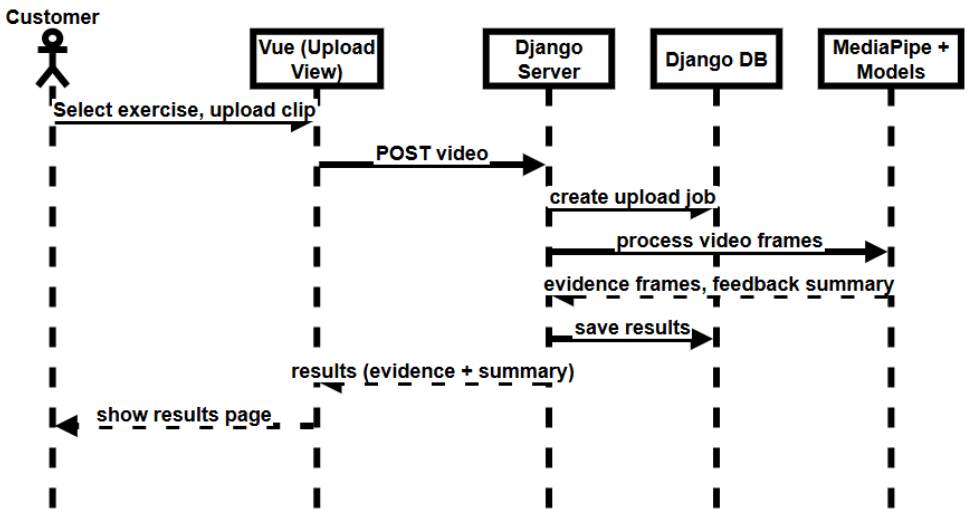

Figure 4.19: Sequence Diagram: AI Workout – Upload Video (Web)

This diagram illustrates the AI Workout video upload flow. The customer selects an exercise and uploads a video clip. The Django server creates a job in the database and processes the video frames using MediaPipe Pose and ML models. It generates evidence frames and a feedback summary, saves the results, and the web app displays the analysis to the customer.

4.4.4.14 Customer Profile Management

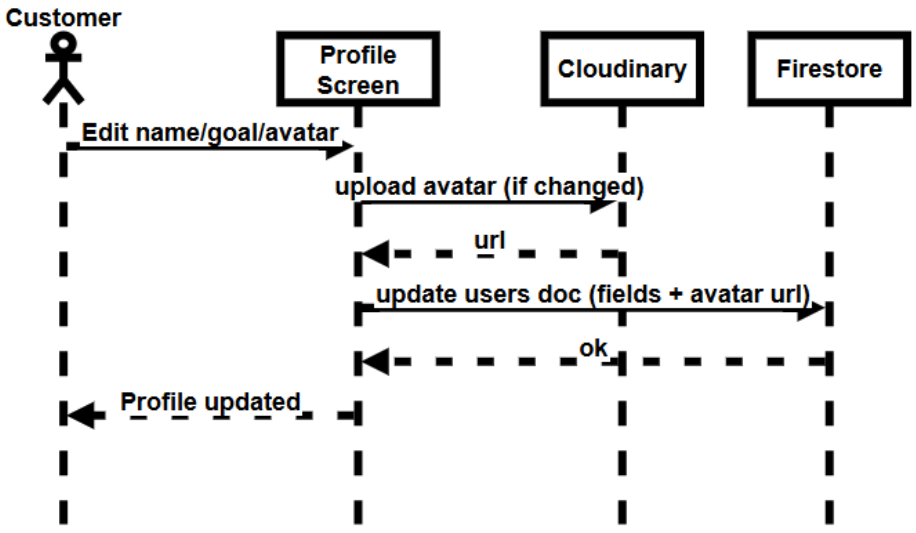

Figure 4.20: Sequence Diagram: Customer Profile Management

This diagram illustrates the Customer Profile Management flow. The customer edits their name, fitness goals, or avatar. If the avatar is changed, it is uploaded to Cloudinary and the returned URL is saved. Firestore updates the user document with the new details, and the app confirms that the profile has been successfully updated.

4.4.4.15 Availability Save (Professional)

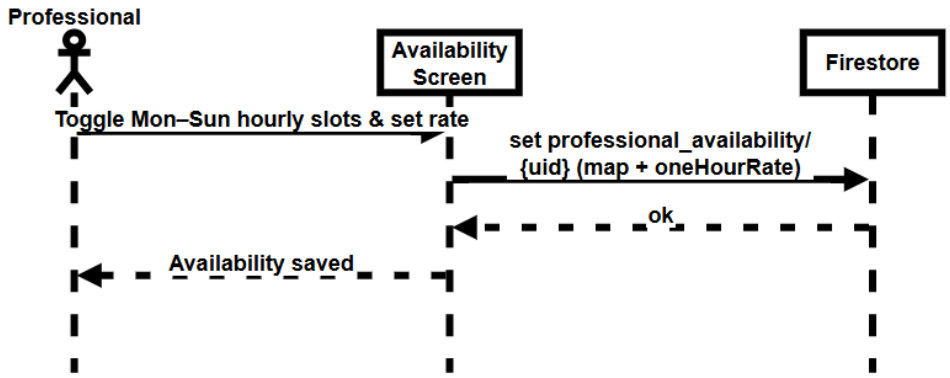

Figure 4.21: Sequence Diagram: Availability Save (Professional)

This diagram shows how a professional manages and saves their availability. The professional selects hourly slots for each day and sets their rate. The app stores this information in Firestore under the professional's availability document. Once saved, the app confirms that the availability has been successfully updated.

4.4.5 Activity Diagram

The below activity diagrams illustrate the workflow and interactions of Customers, Professionals, Admin, and the Web App within the system.

4.4.5.1 Customer Activity Flow

Figure 4.22: Customer and Professional Activity Flow Diagrams

4.4.5.2 Professional Activity Flow

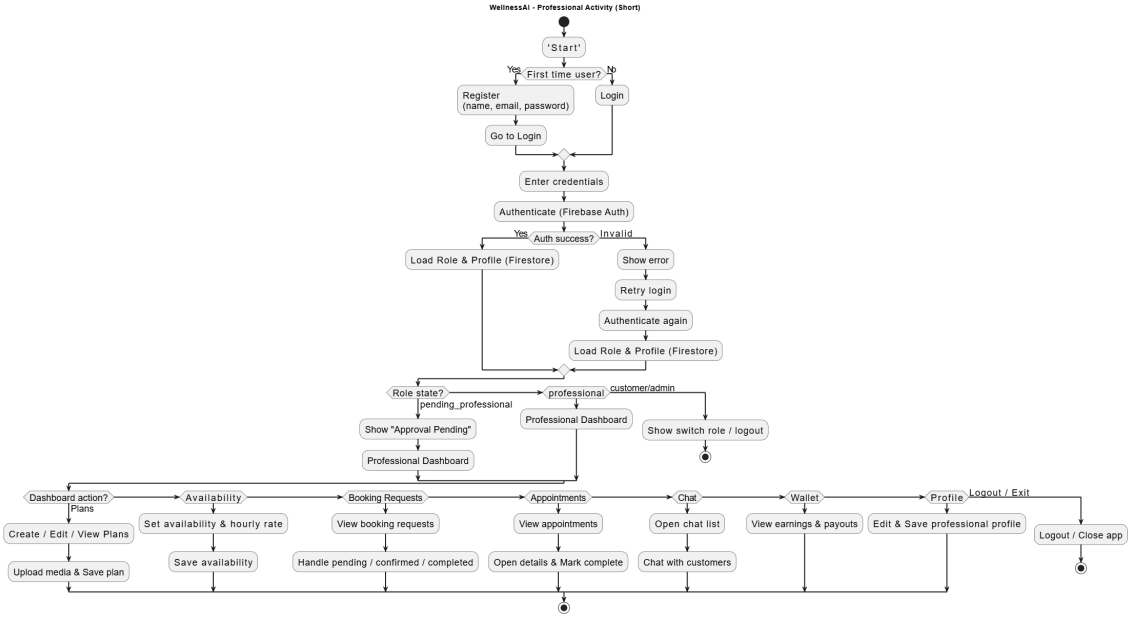

Figure 4.23: Professional and Customer Activity Flow Diagrams

4.4.5.3 Admin Activity Flow

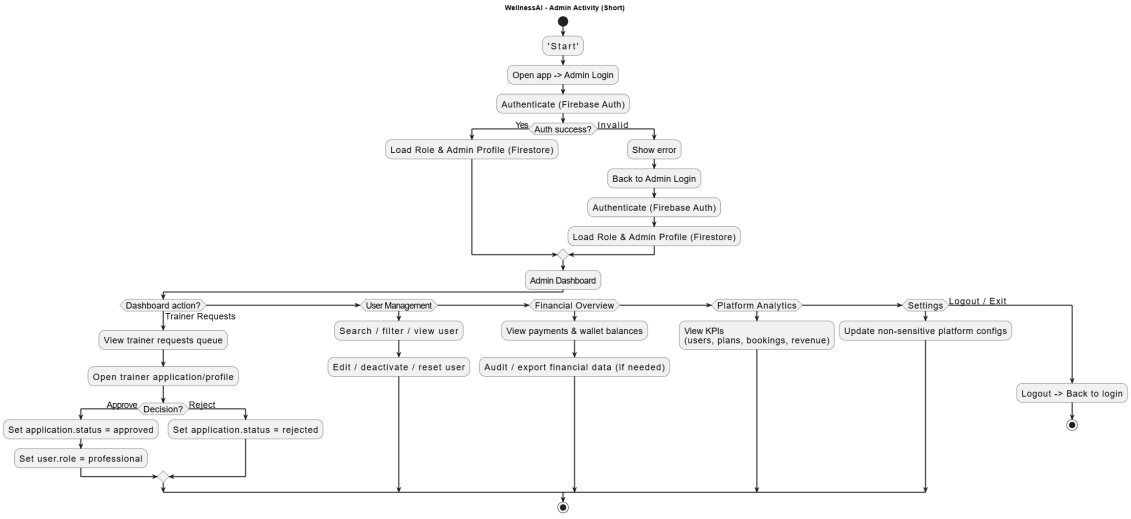

Figure 4.24: Admin Activity Flow Diagram

4.4.5.4 Web App Activity Flow

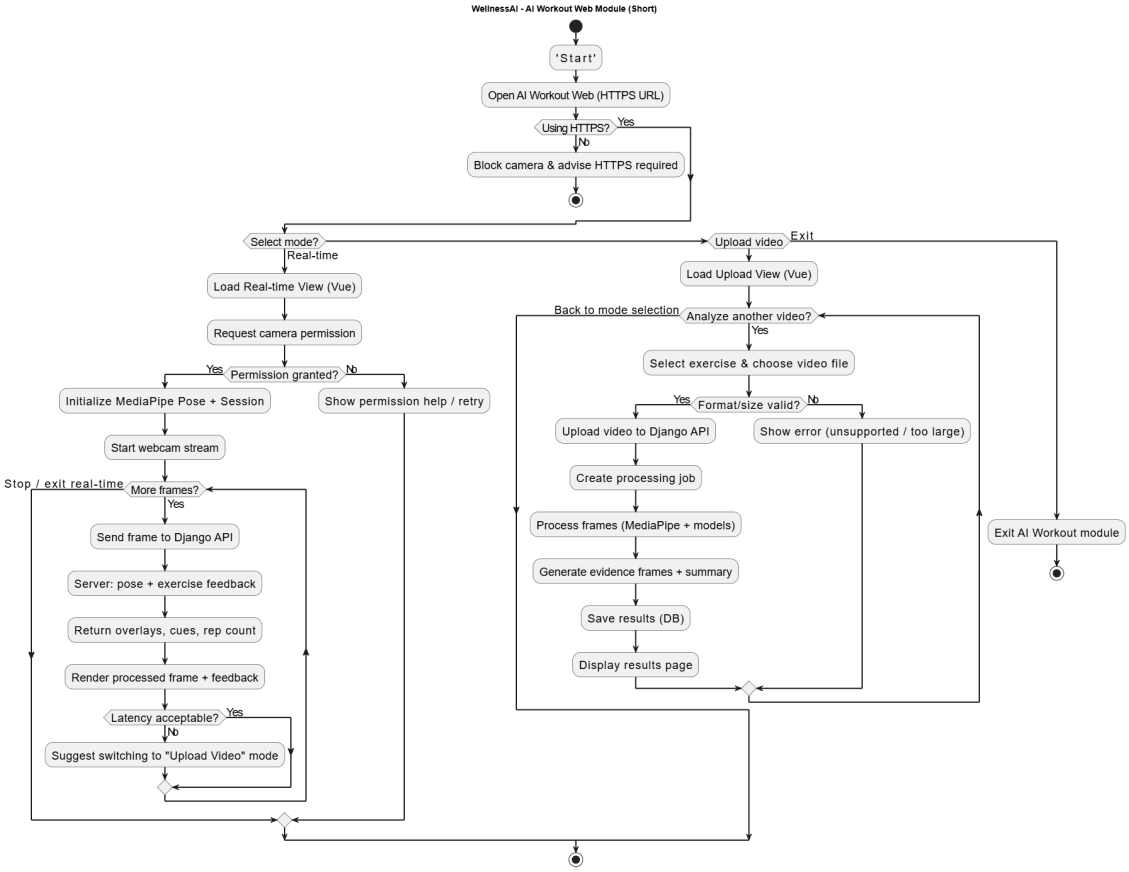

Figure 4.25: Web App Activity Flow Diagram

4.5 Low Level Design

4.5.1 Mobile App Component Breakdown

4.5.1.1 Authentication Module

- **AuthProvider:** Responsible for managing authentication state, restoring the session, and role-based navigation routes.
- **FirebaseService:** Responsible for Firebase Auth interactions (signIn, signUp, signOut, and resetPassword).
- **AppUser model:** Represents user information and role-specific information.

4.5.1.2 Plans Module

- **Plan model:** Represents Plan information (title, price, category, description, media URLs).

- **CreatePlanScreen:** Professional-facing screen for creating/editing plans.
- **PlansScreen:** Customer-facing screen for browsing or discovering plans.
- **PlanDetailScreen:** Displays information about the plan being viewed and allows purchase.

4.5.1.3 Booking Module

- **Booking model:** Represents appointment information (date, time, status, participants, amount).
- **AppointmentBookingScreen:** Calendar interface for selecting date/time slots for the appointment.
- **BookingStatusManager:** Component to manage moving the appointment through its status lifecycle and manage any cleanup afterwards.
- **AvailabilityManagementScreen:** Professional-facing screen for setting weekly availability.

4.5.1.4 Payment Module

- **Payment model:** Logs transaction information with fee details.
- **PaymentProvider:** Handles creation of payments, fee calculations, wallet updates.
- **PaymentConfig:** Configuration of percentage fees on the platform and logic for calculating fees.

4.5.1.5 Chat Module

- **Chat model:** Represents chat threads which are associated with bookings.
- **Message model:** Information on individual messages.
- **ChatScreen:** Real-time messaging interface.
- **ChatListScreen:** List of all chat threads.

4.5.2 AI Web Module Component Breakdown

4.5.2.1 Frontend (Vue.js)

- **RealTime.vue:** View for real-time webcam analysis.
- **VideoStreaming.vue:** View for video upload and analysis.

- **Webcam.vue:** Camera capture component.
- **Video.vue:** Video upload component.
- **Result.vue:** Result display component.

4.5.2.2 Backend (Django)

- **realtime.py:** Real-time processing of frames for MediaPipe Pose.
- **squat.py, plank.py, bicep_curl.py, lunge.py:** Detection models for specific exercises.
- **utils.py:** Utility functions for pose analysis and visualization.
- **views.py:** REST API endpoints for processing frames and uploading videos.

4.5.3 Class Diagram (Main Models)

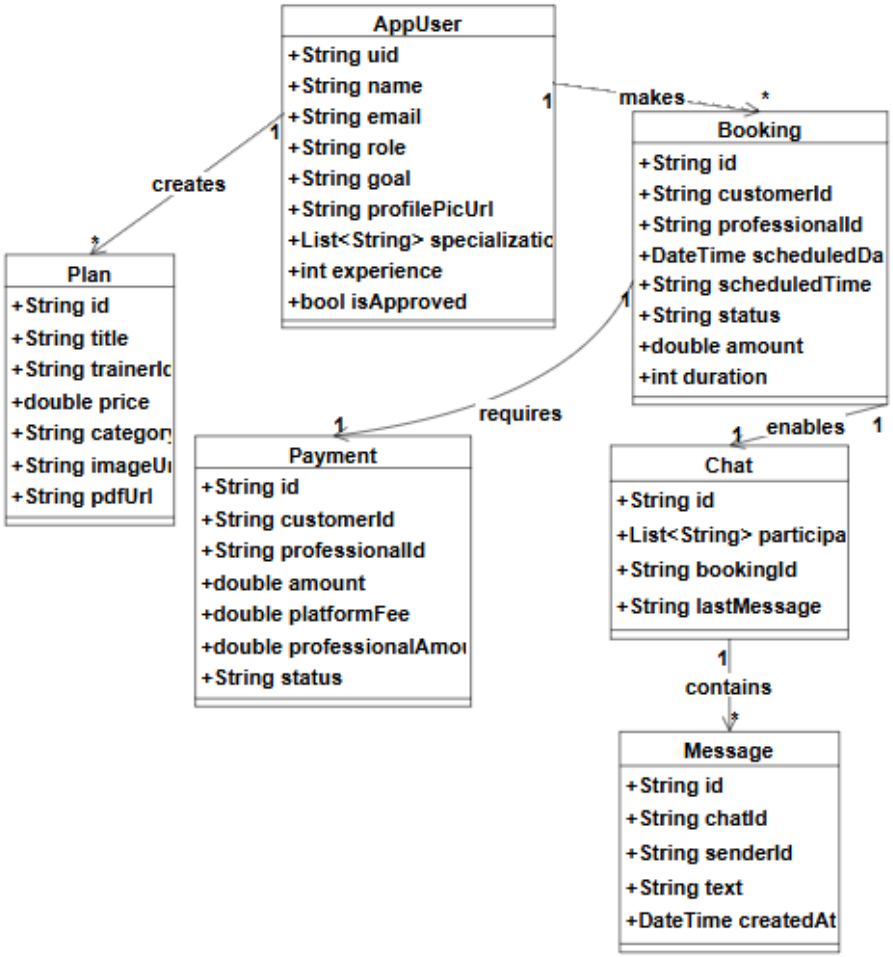

Figure 4.26: Class Diagram (Key Models)

4.6 Database Design

4.6.1 Entity-Relationship Diagram (ERD)

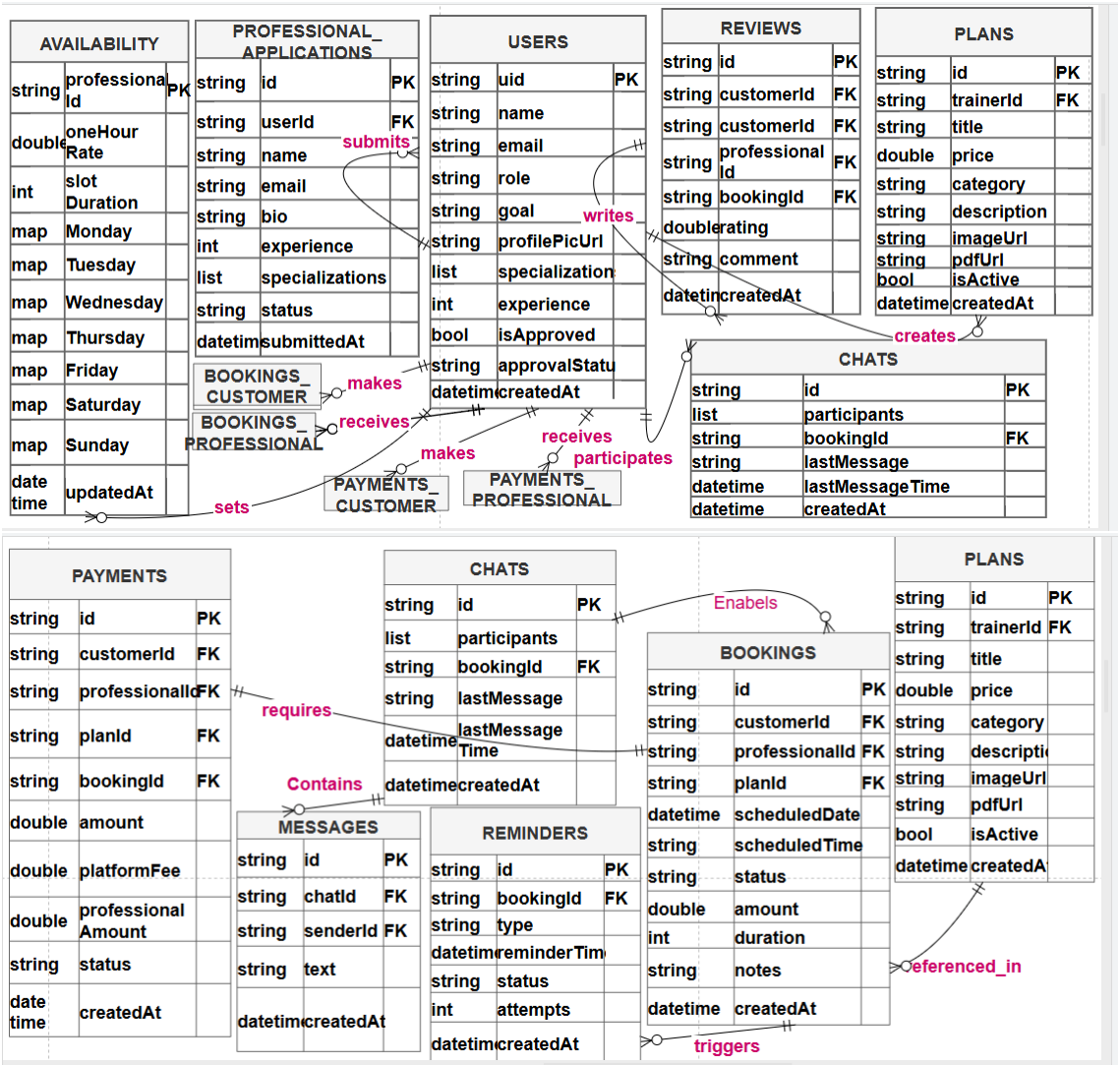

Figure 4.27: Entity-Relationship Diagrams (ERD)

4.6.2 Non-DBMS Files

4.6.2.1 Cloudinary Media Storage

- **User profile (avatar) images:** Stored as images and accessible through public URLs.
- **Plan images:** Cover images for the workout plans.
- **Plan PDFs:** Full workout plan document.
- **File naming pattern:** Either {userId}_{timestamp} or {planId}_{timestamp}.

4.6.2.2 Local Storage (Mobile App)

- **SharedPreferences:** Stores session data (isAuthenticated, userRole, userEmail, userUid) so the user can re-enter their session offline.
- **FCM tokens:** Stored in Firestore users collection for push notification purposes.

4.6.2.3 AI Module Files

- **Pretrained models:** static/model/
- **Evidence frames:** Processed images that are saved in the Django media directory for uploads to analyze evidence messages.

4.7 GUI Design

4.7.1 Mobile App GUI Design

4.7.1.1 Customer Interface

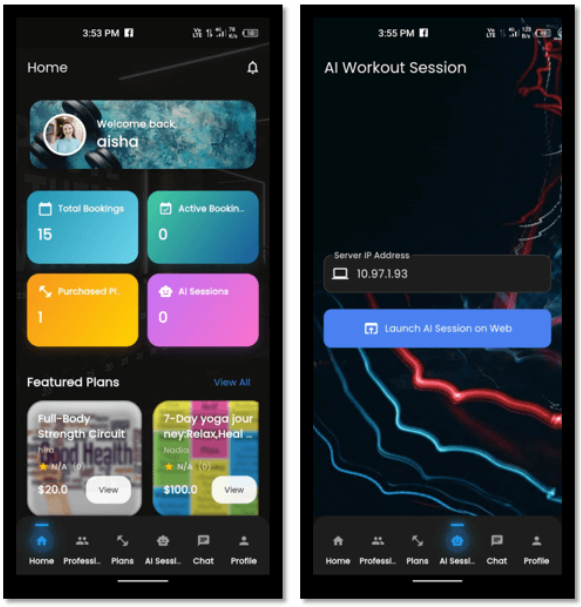

Figure 4.28: Dashboard Screen, AI Session Launcher

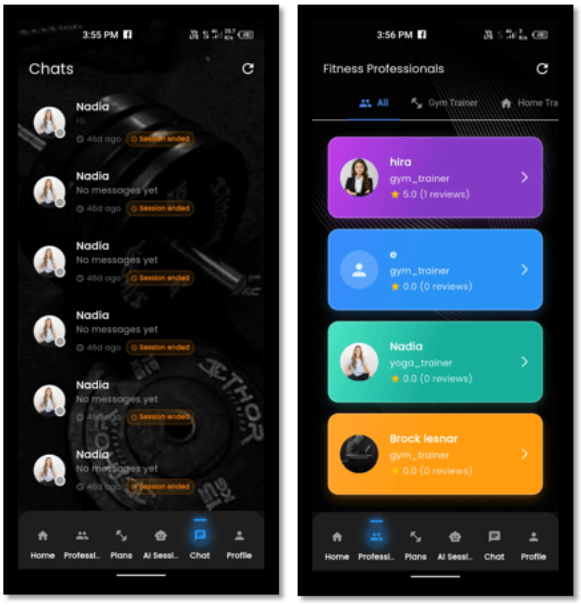

Figure 4.29: Chat Screen, Professional Discovery Screen

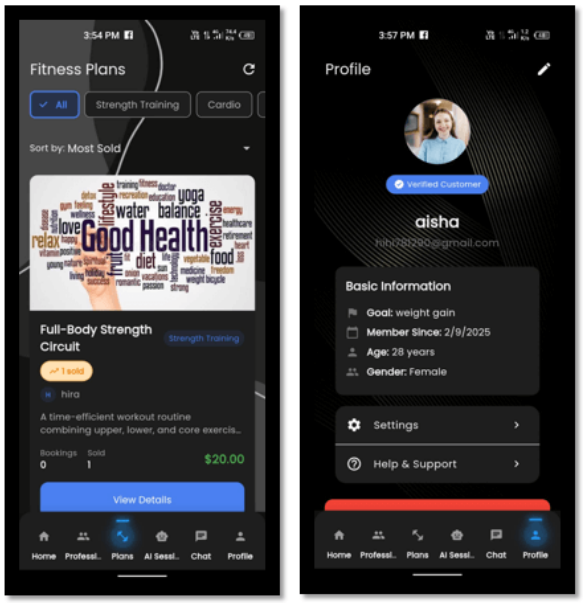

Figure 4.30: Plan Discovery Screen, Profile Screen

4.7.1.2 Professional Interface

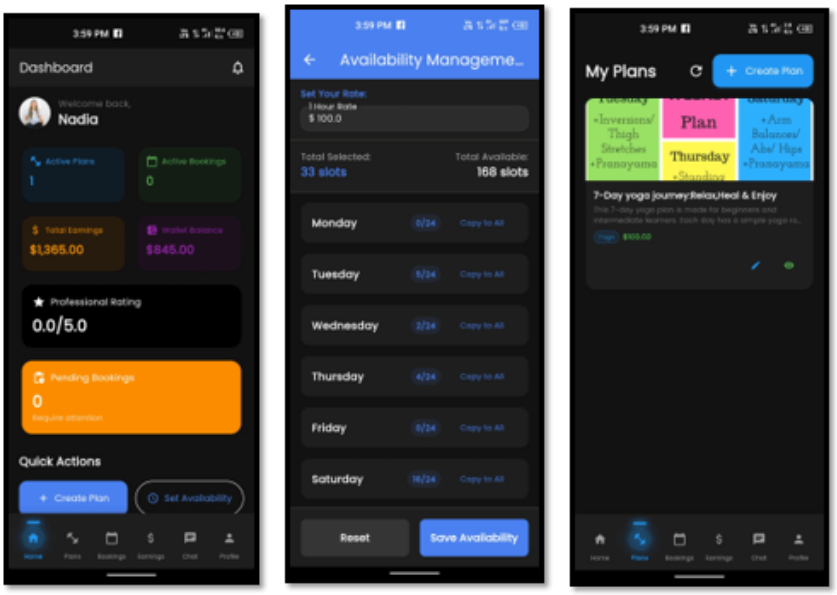

Figure 4.31: Professional Dashboard, Availability Management Screen, Create/Edit Plan Screen

4.7.1.3 Admin Interface

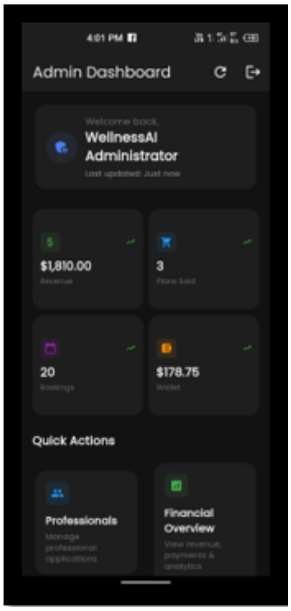

Figure 4.32: Admin Dashboard

4.7.2 Web App GUI Design (AI Module)

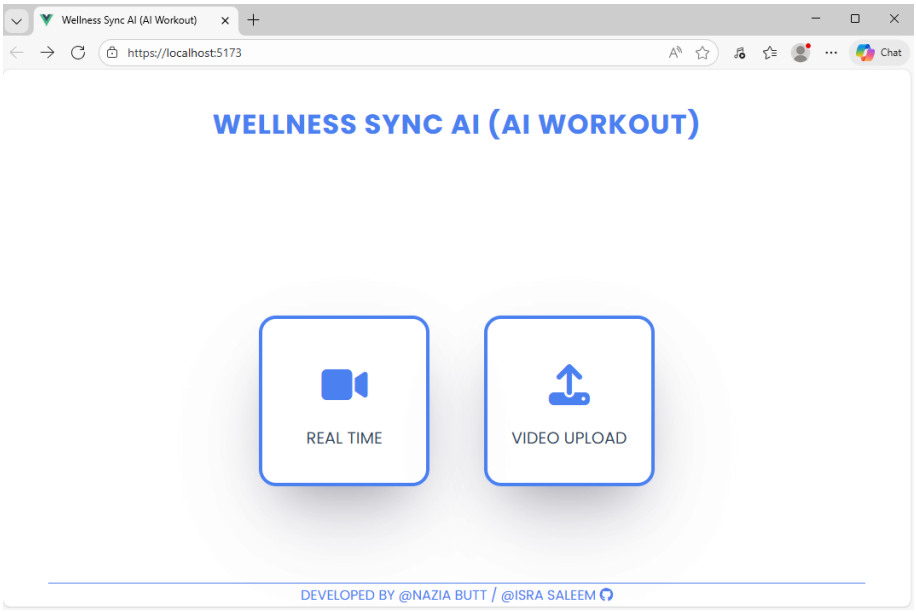

Figure 4.33: Home Screen

4.8 External Interfaces

4.8.1 Firebase Authentication Interface

Purpose: This interface is in charge of managing how a user logs in, signs up, and maintains their login status for a session.

Type of Interface: Firestore's REST API is utilized via the Firebase SDK.

Method of Authentication: Users log in via email and password. Firebase generates a unique token for every user when they authenticate.

Main Operations:

- Sign in with email and password to let users log in.
- Create a new account for user registration.
- Send a password reset link to the user's email.
- Log out the user from the application.
- Track when a user signs in or signs out.

Data Exchange:

- **Input:** Email and password of the user (sent securely).
- **Output:** A user object with UID, email, and authentication token.

Error Handling: If anything goes wrong, such as invalid credentials, connection issues, or account-related problems, Firebase returns the appropriate error message.

4.8.2 Cloud Firestore Interface

Purpose: This interface acts as the primary database for the application, where all user and system information is stored.

Interface Type: It is a NoSQL document-oriented database, which operates over the Firebase SDK.

Authentication: Access to the database is secured with Firebase authentication tokens so that only verified users can read or write data.

Main Operations:

- Add or update records in the database.
- Read stored data when needed.
- Delete unneeded records.
- Search or filter data based on criteria.
- Get live updates for any changes to a record.

Data Exchange:

- **Input:** Data is sent as key–value pairs (like a small data map).
- **Output:** Data returned is either the requested data or the result of a query.
- Data returned includes real-time updates for any change to the data stored in Firestore.

Security: Access to data is governed by Firestore Security Rules so that users may see or edit information pertaining only to themselves, based on their role.

4.8.3 Cloudinary Interface

Objective: To store and retrieve media safely, such as profile pictures and plan documents.

Interface Method: REST API, through Cloudinary SDK.

Authentication: Entry of API key and API secret in CloudinaryConfig.

4.8.4 AI Workout Web Module Interface

Objective: To access an external web app that analyzes movement during exercise in real-time or through video upload.

Interface Method: HTTPS link that can be launched from the mobile app.

Connection Details:

- **Access Method:** The mobile app launches using the HTTPS URL in the default WebView or browser.
- **Protocol/Transmission:** HTTPS, necessary to access the camera.
- **Base URL:** Configurable server IP address/domain name/thread, depending on the user's entry in the mobile app.

Operations:

- **Real-Time Analysis:** Streaming frames to web app via a WebSocket or REST API method on the client.
- Client sends frames in base64 format.
- Server sends frames back with overlays, feedback, and number of repetitions demonstrated, in video format.
- **Video Upload Analysis:** REST API (POST /upload-video) where the client uploads videos.
- The server processes the video and returns evidence frames with a summary of feedback.

Security Settings:

- HTTPS is a requirement (browser will block the camera on HTTP).
- Valid SSL certificate needed.

4.8.5 Notification Service Interface (FCM and Local)

Objective: This is the interface for issuing push notifications and scheduling upcoming user appointment reminders.

Interface Type: Both Firebase Cloud Messaging (FCM) and local notifications are integrated to improve communications and reminders.

Connection Details:

- **FCM:** Uses the Firebase Cloud Messaging service to handle push notifications from the server to the user's device.
- **Local:** Uses the Flutter Local Notifications plugin to schedule and notify reminders directly on the device.

Token Management: Each device that a user has is given a unique FCM token that is stored securely in Firestore under their user record. The token ensures the correct device receives notifications.

Main Operations:

- **Token Registration:** Generate and save the FCM token in Firestore when the app starts.
- **Token Refresh:** Update the token in Firestore when it changes.
- **Push Notifications:** Messages are pushed from the server via FCM notifications to the user.
- **Local Scheduling:** Schedule notifications for the user to remind them of upcoming appointments.

Chapter 5

System Implementation

This chapter presents a comprehensive look at the strategies used to implement the Wellness AI system. The implementation phase involves the conversion of theoretical designs, plans, and algorithms into a fully operational software solution. In this project, it encompasses creating a mobile application based on Flutter, an AI Workout Module, and backend services utilizing Firebase and Django, all while ensuring integration for real-time analysis of exercises through AI.

5.1 Tools and Technologies

The system is developed using an array of technologies including Flutter, Firebase, Django, Vue.js, Cloudinary, and AI/ML frameworks. These were chosen for their efficiency, scalability, and ease of maintenance. The tools are categorized into four main sections: frontend, backend, AI module, and supporting technologies.

5.1.1 Flutter Framework (Mobile Application)

The cross-platform mobile app designed for users, professionals, and administrators is created with the Flutter framework using the Dart programming language. Flutter is selected due to its impressive performance, ability to maintain a single codebase for both iOS and Android platforms, as well as its extensive library of pre-designed widgets that streamline the creation of a cohesive and responsive user interface [13].

Key components and configurations utilized in this Flutter application are as follows:

- **Firebase Auth & Cloud Firestore:** This serves as the backend-as-a-service (BaaS) for user authentication processes like login and signup [14]. It also stores user profiles, bookings, chat messages, and payment information.

- **Provider:** A state management solution that helps manage application state predictably, including aspects such as user sessions, theme preferences, and booking history.
- **GoRouter:** A sophisticated routing package that supports declarative navigation along with role-based access control (Customer, Professional, Admin), ensuring users can only navigate to permitted screens.
- **Cloudinary:** This service is integrated for secure handling of file uploads and image management throughout the app—this includes user profile pictures and professional certification documents [15].
- **Http & Dio:** These packages enable API interactions such as sending emails through a set up SMTP service and facilitating payment processing.

5.1.2 Python & Machine Learning (AI Module)

The main exercise correction module powered by AI is developed using Python, taking advantage of its rich ecosystem for data science and machine learning.

- **MediaPipe Pose:** This vital library offers real-time human pose estimation [7]. It identifies 33 body landmarks from video frames, which serve as essential data for all further analysis.
- **OpenCV (cv2):** This library handles various video processing tasks such as reading video files, capturing webcam footage, resizing frames, and marking output videos with landmarks and feedback [16].
- **Scikit-learn (sklearn):** This tool is used to create and train traditional machine learning models like Logistic Regression, K-Nearest Neighbors, Support Vector Machines, and Random Forest for classifying postures and detecting exercise stages [17].
- **Keras with TensorFlow Backend:** It's utilized for constructing, training, and assessing deep learning models known as Multi-Layer Perceptrons [18, 19]. Keras Tuner aids in automating hyperparameter optimization to discover the most effective model architecture.
- **Pandas & NumPy:** These libraries are crucial for manipulating data, loading CSV files, and conducting numerical operations on landmark information.
- **Pickle:** This tool specializes in serializing trained machine learning or deep learning models along with scalers. It enables these components to be loaded later for inference during detection without requiring retraining.

5.1.3 Vue.js Framework (Web Dashboard)

The web application, which permits video uploads and real-time webcam analysis, is developed using Vue.js 3 [20], a progressive JavaScript framework that was selected because of its component-based architecture and reactivity system, allowing for a responsive and maintainable frontend.

The application integrates some key libraries with Vue.js as follows:

- **Vue Router:** Provides client-side routing so that the user can navigate to the Home, Video Upload, and Real-Time pages without completely reloading the page.
- **Pinia:** The official state management library for Vue to help manage a global application state.
- **Axios:** A promise-based HTTP client to communicate with the backend server, for uploading files and fetching analysis results.
- **WebSocket API (Native):** Allows for a full-duplex communication channel that stays open between the backend server and the client for real-time webcam frame transmission and instant feedback [21].

5.1.4 Framework (Backend Server)

The backend server that runs the web dashboard was developed in Django, a high-level Python web framework. It was chosen for its built-in features, security, and ecosystem. The project uses Django Channels for WebSocket support, enabling real-time communication and native async capabilities. [22].

- **Uvicorn:** ASGI servers that run the Django application with Channels, providing high concurrency for handling many requests simultaneously.
- **WebSocket Endpoints:** Added specifically to manage the real-time communication flow, taking frames from the frontend, sending them through the AI models, and sending back annotated results with reduced latency [21].
- **HTTP Endpoints:** Developed to manage video file uploads as well as streaming processed videos back to the client.

5.1.5 Database & Cloud Services

- **Firebase Firestore:** A flexible, scalable NoSQL cloud database that will act as the primary data store for the Flutter mobile application.
- **Firebase Storage:** Used alongside Firestore in order to store larger files that fall outside of Firestore limits, like video recordings and certification documents.

5.1.6 AI Models and Datasets

A structured pipeline was created, which consisted of multiple phases, via a systematic implementation for each of the four exercises supported: Bicep Curl, Squat, Lunge, and Plank. To systematically conduct this, labeled datasets were generated from pose landmarks, and models were developed utilizing a standardized approach to classify posture and determine if errors had occurred.

5.1.6.1 Bicep Curl Model

The Bicep Curl model performed binary classification, with classification being between the correct posture in the Bicep Curl and the common error of leaning backward, along with rep counting based off elbow angle.

Dataset Format The dataset was constructed using 9 upper-body landmarks extracted from video frames. Each frame was manually identified by an observer during the data collection process.

Table 5.1: Features of the dataset used for training the Bicep Curl model.

Landmarks Tracked	Features per Landmark	Label	Description
Shoulders, Elbows, Wrists, Hips	X, Y, Z coordinates and Visibility Score	C	Correct Form
Shoulders, Elbows, Wrists, Hips	X, Y, Z coordinates and Visibility Score	L	Low Back / Incorrect Form (Leaning)

Performance Metrics Multiple models were trained and evaluated. The K-Nearest Neighbors (KNN) model demonstrated the best overall performance for this classification task and was selected as the final model.

Table 5.2: Bicep Curl Model Performance Metrics

Model Type	Model	Precision	Recall	F1-Score	Accuracy
Classical ML	K-Nearest Neighbors (KNN)	0.94	0.95	0.94	0.94
Classical ML	Logistic Regression	0.92	0.93	0.92	0.92
Deep Learning	7-Layer MLP	0.90	0.91	0.90	0.90

5.1.6.2 Squat Model

The Squat model is a binary classifier that identifies the 'Up' and 'Down' stages of a squat to count repetitions. A separate rule-based system analyzes body alignment ratios (e.g., knee-to-foot) for form feedback.

Dataset Format The dataset focused on 9 key lower-body and core landmarks critical for identifying the depth of the squat.

Table 5.3: Features of the dataset used for training the Squat model.

Landmarks Tracked	Features per Landmark	Label	Description
Shoulders, Hips, Knees, Ankles	X, Y, Z coordinates and Visibility Score	Up	Standing (top) position
Shoulders, Hips, Knees, Ankles	X, Y, Z coordinates and Visibility Score	Down	Bottom of the squat position

Performance Metrics Logistic Regression proved to be the most effective and computationally efficient model for classifying the squat stage, forming the basis for the repetition counter.

Table 5.4: Squat Stage Model Performance Metrics

Model Type	Model	Precision	Recall	F1-Score	Accuracy
Classical ML	Logistic Regression	0.98	0.98	0.98	0.98
Classical ML	Support Vector Classifier	0.97	0.97	0.97	0.97
Deep Learning	5-Layer MLP	0.96	0.96	0.96	0.96

5.1.6.3 Lunge Model

The Lunge system employs two specialized models: a multi-class model for stage detection and a binary model for error detection, specifically "Knee Over Toe."

Dataset Format Two distinct datasets were created. The first, for stage detection, used 13 landmarks. The second, for error detection, used the same landmarks but was labeled for knee alignment.

Table 5.5: Features of the dataset used for the Lunge Stage model.

Landmarks Tracked	Features per Landmark	Label	Description
Shoulders, Hips, Knees, Ankles, Heels, Foot Index	X, Y, Z coordinates and Visibility Score	I	Initial (Standing)
Shoulders, Hips, Knees, Ankles, Heels, Foot Index	X, Y, Z coordinates and Visibility Score	M	Mid (Transition)
Shoulders, Hips, Knees, Ankles, Heels, Foot Index	X, Y, Z coordinates and Visibility Score	D	Down (Lunge position)

Table 5.6: Features of the dataset for Lunge Error detection.

Landmarks Tracked	Features per Landmark	Label	Description
Shoulders, Hips, Knees, Ankles, Heels, Foot Index	X, Y, Z coordinates and Visibility Score	C	Correct Knee Alignment
Shoulders, Hips, Knees, Ankles, Heels, Foot Index	X, Y, Z coordinates and Visibility Score	L	Incorrect Knee Alignment (Over Toe)

Performance Metrics The Ridge Classifier excelled at the multi-class stage detection task, while Logistic Regression was most accurate at identifying the critical knee alignment error.

Table 5.7: Lunge Stage Model Performance Metrics

Model Type	Model	Precision	Recall	F1-Score	Accuracy
Classical ML	Ridge Classifier	0.99	0.99	0.99	0.99
Classical ML	Support Vector Classifier	0.98	0.98	0.98	0.98

Table 5.8: Lunge Error Model Performance Metrics

Model Type	Model	Precision	Recall	F1-Score	Accuracy
Classical ML	Logistic Regression	0.96	0.96	0.96	0.96
Deep Learning	7-Layer MLP	0.94	0.94	0.94	0.94

5.1.6.4 Plank Model

The Plank model is a multi-class classifier designed to identify a correct plank hold and two common errors: hips too high and hips too low (sagging).

Dataset Format The dataset was built using landmarks that indicate the alignment of the back and hips during a plank hold.

Table 5.9: Features of the dataset used for training the Plank model.

Landmarks Tracked	Features per Landmark	Label	Description
Shoulders, Hips, Knees, Ankles, Elbows	X, Y, Z coordinates and Visibility Score	C	Correct (Straight Back)
Shoulders, Hips, Knees, Ankles, Elbows	X, Y, Z coordinates and Visibility Score	H	Hips Too High
Shoulders, Hips, Knees, Ankles, Elbows	X, Y, Z coordinates and Visibility Score	L	Hips Too Low (Sagging)

Performance Metrics For the plank exercise, Logistic Regression outperformed other complex models, including deep learning networks, in accurately classifying the three posture states with high precision and recall.

Table 5.10: Plank Model Performance Metrics

Model Type	**Model**	**Precision**	**Recall**	**F1-Score**	**Accuracy**
Classical ML	Logistic Regression	0.95	0.95	0.95	0.95
Classical ML	Support Vector Classifier	0.93	0.93	0.93	0.93
Deep Learning	7-Layer MLP with Dropout	0.91	0.91	0.91	0.91

Chapter 6

System Testing and Evaluation

This chapter discusses the testing methodologies and evaluation procedures performed on the Wellness AI application. Software testing is an important step that is done in a controlled environment to evaluate the software applications ability to function, perform, and be reliable in both the normal case (expected) and anomaly case (unexpected). A few minor bugs were reported during the testing phase prior to the application being developed for early implementation, but all of the major feature areas were validated against the requirements.

6.1 Testing Techniques

We utilized a multi-faceted testing strategy to validate the robustness of the solution prior to deployment. This included acceptance testing, performance testing, functional testing, unit testing, integration testing, usability testing, and compatibility testing. Each phase specifically targeted the identification of defects and correction of them, validating that all functional and non-functional requirements were met and that exceptions were gracefully handled.

6.2 Acceptance Testing

Acceptance testing was performed to ensure the Wellness AI application met core purposes for all user roles: Customers, Professionals, and Administrators. The tests validated that the system is ready for launch to create a seamless user experience for booking sessions, managing plans, processing payments, and performing AI-driven workout analysis.

6.3 Performance Testing

Performance testing was essential to verify the application can continue to perform efficiently and responsively under various load scenarios. Key scenarios to be covered include:

- Engineering various simulations of multiple concurrent users booking appointments, and uploading videos.
- Stress testing the AI model inference pipeline during real time webcam video analysis.
- Measuring database query performance related to user management and payment history.

The performance of the system was optimal; API call response times were consistently low during peak, and the application remained operational without interruption during peak load events, ensuring each booking was a seamless experience for users.

6.4 Functional Testing

Testing functionality was conducted to confirm that all features of the Wellness AI application functioned as expected. Test cases were written to include all features pertaining to user registration and authentication, appointment booking, payment processing, plan management, chat, and the fundamental AI functionalities of video analysis to both uploaded videos and from the real-time feed from the webcam.

6.5 Unit Testing

The system's individual components were tested at the unit level. This included:

- Testing the logic responsible for user login, signup, and session restoration.
- Testing the rep-counting and error-detection logics for validity.
- Testing the functions that calculate payment fees.

As part of the early testing, I ensured that code for each discrete code unit was reliable and free of bugs.

6.6 Integration Testing

Integration testing concentrated on confirming that all components of the system worked together as a system. The main integration points tested were the following:

- The flow from the user registering in Flutter to the user registering in Firebase Firestore.
- The communication between the Vue.js front end and the FastAPI back end during video upload and processing.
- The data flow over a WebSocket for real-time pose detection, including sending frames from the browser and receiving annotated results and feedback.
- The integration of the Stripe payment gateway, from the user beginning a payment in the app to confirming the transaction in the database.

We utilized tools such as Postman for testing APIs, and tested all data flows for correctness.

6.7 Usability Testing

Usability testing was carried out to validate that the application would be intuitive and straightforward for the intended audience. Users from varied and diverse backgrounds were asked to complete key tasks—booking a professional, uploading a workout video for analysis, and using the real-time AI coach for feedback. Responses were gathered on the clarity of the AI comments and recommendations (“Low back,” “Hips Too High”), the navigation flow, and the overall design. The interface was revised for greater accessibility for both tech-savvy and non-tech-savvy users.

6.8 Compatibility Testing

In compatibility testing, we confirmed the application works correctly in all supported environments.

- **Flutter Mobile App:** The app was tested thoroughly on an Android emulator and physical Android mobile devices after installing the APK directly to assess the performance and compatibility in real-world usage.
- **Vue.js Web Dashboard:** The app was tested on modern browsers, including Chrome, Firefox, and Safari.

Ultimately, the application tested fully compatible against all expected platforms and browsers.

6.9 Context and Test Plan

1. **Context:** The project involved a multi-platform wellness app that uses AI to correct exercises. Pose detection accuracy, payment processing reliability, and overall system efficiency are critical success factors.

2. **Plan:** A test plan was developed with carefully defined success criteria and measures for functional reliability, AI accuracy and system performance. Key metrics such as AI model inference time, payment process success rate, and UI response time were calculated and evaluated.

3. **Ideal:** Test scenarios included elements meant to mimic real-world use cases: user account creation, booking flows, uploading videos, and AI interaction in real-time.

4. **Prepare & Perform:** The project team executed all planned testing scenarios while closely monitoring system performance. Documenting the results provided important data to help finalize the final optimization phase.

6.9.1 Traceability Matrix

This matrix ensures all critical user requirements are covered by test cases.

Table 6.1: Traceability Matrix

Test Objective	**Test Basis**	**Completion Criteria**
User Registration & Login	User can initiate a process to register and login as Customer/Professional/Admin.	Account gets created and the user can access their respective dashboard.
Professional Discovery	Customer can browse and see professional profiles.	Professionals are listed with relevant profiles, skills, and ratings.
Workout Plan Purchase	Customer browses workout plans and purchases one.	Plan appears in purchased plans and payment processes successfully.
Appointment Booking	Customer can book appointments.	Booking created with correct date, time, and status.
Real-Time Chat	Users communicate during appointment windows.	Messages sent/received in real time.
AI Video Analysis	User uploads video and receives analysis.	JSON result + annotated video stream returned.
Real-Time AI Feedback	User receives live posture correction.	WebSocket returns overlay, rep count, and errors.
Professional Plan Creation	Professional creates plans.	Plan saved to DB and shown in dashboard.
View Earnings	Professional views earnings.	Wallet shows correct balance.
Admin User Management	Admin manages users.	View/delete functions work correctly.
Professional Approval	Admin approves/rejects applications.	Status updates and access granted.

6.10 Test Cases

6.10.1 Test of User Registration and Login

This test verifies that users can successfully register and login to the application.

Table 6.2: Test case of User Registration and Login

Test Case ID	TC-01
Description	Adding a new user account and testing the login process.
Requirement	System must create user accounts and validate credentials.
Steps to be Taken	1. Go to registration screen. 2. Enter the required fields (name, email, password, role). 3. Finish registration. 4. Log out and log in with the new account credentials.
Expected Result	The account is created successfully. The user can log in, and is presented with appropriate dashboard based on his/her role.
Actual Result	Registration successful. Logged in as Customer, redirect to customer dashboard. Logged in as Professional, redirected to wait approval screen.
Status	Success
Remarks	N/A

6.10.2 Test of Professional Discovery and Profile Viewing

This test ensures customers can browse and view professional profiles.

Table 6.3: Test case of Professional Discovery

Test Case ID	TC-02
Description	Browsing through a selection of usually available professionals and viewing their detailed biographies.
Requirement	The customer will see a list of professionals with brief information, including access to the detail of their profiles.
Steps to be Taken	1. Log In as Customer. 2. Go to the Professionals section. 3. Navigate the list. 4. Select a professional to see the complete profile.
Expected Result	The professional list contains names, photographs, and ratings. The detailed profile contains the biography, experience, certifications, and skills.
Actual Result	The list populated correctly. Detailed profile displayed all professional information including uploaded certifications.
Status	Success
Remarks	N/A

6.10.3 Test of Workout Plan Browsing and Purchase

This test verifies the complete plan purchase workflow.

Table 6.4: Test case of Plan Purchase

Test Case ID	TC-03
Description	The customer can navigate through plans and complete a purchase.
Requirement	The system will show plans, and payment will be processed successfully.
Steps to be Taken	1. Customer can log in. 2. Customer can browse plans of workouts. 3. Customer can select a plan and go to purchase. 4. The customer will complete payment in Stripe.
Expected Result	After payment is processed, the plan details are shown. After the payment is completed, the purchased plan shows up in the customer's purchased plans in the app, and a share of the earnings goes into the professional's earnings and is disbursed to the professional.
Actual Result	Purchasing a plan is successful, the transaction has been recorded in the payment system. The plan is added to the customer's account and the professional is updated with a wallet showing a 90% shared earnings.
Status	Success
Remarks	Used Stripe test mode for payment.

6.10.4 Test of Appointment Scheduling and Management

This test validates the appointment booking and management system.

Table 6.5: Test case of Appointment Scheduling

Test Case ID	TC-04
Description	A customer books an appointment and manages their bookings.
Requirement	The system allows the user to create a booking, view a booking, or cancel a booking.
Steps to be Taken	1. Login as a customer 2. Select a professional and available time slot 3. Book the appointment 4. View in booking history 5. Cancel to booking (if within time allowed).
Expected Result	The booking is created with a status of "confirmed" and will display in booking history. Cancelling the booking updates the status to "cancelled".
Actual Result	The booking creation was successful and booking history displayed all bookings. The cancel booking function worked as expected and the status was successfully updated.
Status	Success
Remarks	N/A

6.10.5 Test of Real-time Chat Functionality

This test verifies the chat system between customers and professionals.

Table 6.6: Test case of Real-time Chat

Test Case ID	TC-05
Description	Verifying message exchange between customer and professional.
Requirement	Messages should be real-time during appointment windows.
Steps to be Taken	1. Log-in as Customer then as Professional on a different device. 2. Open a chat from a current/upcoming booking. 3. Exchange several messages back and forth. 4. Confirm chat closes after appointment end time.
Expected Result	Messages are delivered instantly on both devices. The chat screen indicates who is communicating and has time stamps for the message exchange. The chat was no longer available after appointment end time.
Actual Result	Real-time messaging worked as intented and chat automatically disabled after appointment elapsed.
Status	Success
Remarks	N/A

6.10.6 Test of AI Video Upload and Analysis

This test verifies that a user can successfully upload an exercise video and receive a detailed AI analysis.

Table 6.7: Test case of AI Video Upload

Test Case ID	TC-06
Description	A video file can be uploaded to the web dashboard for AI posture analysis.
Requirement	The system will process the video and display the results of its analysis.
Steps to be Taken	1. Go to the Video Upload. 2. Select an Exercise Type (e.g., Plank). 3. Select and upload a video file. 4. Wait for the system to complete its processing.
Expected Result	The system will display a results page with a Summary, Detailed timestamps of errors, and a video that is playable and annotated.
Actual Result	The results page loaded correctly with the "Hips Too High" errors displaying with timestamps and the video.
Status	Success
Remarks	N/A

6.10.7 Test of Real-Time AI Pose Detection

This test validates the real-time feedback feature using a webcam.

Table 6.8: Test case of Real-Time AI Feedback

Test Case ID	TC-07
Description	Implement real-time AI coaching from a webcam on a web dashboard.
Requirement	The system shall provide live posture feedback.
Steps to be Taken	1. View the Real-Time page. 2. Grant access to your camera. 3. Select the type of exercise (e.g., Bicep Curl). 4. Begin the session and conduct the exercise.
Expected Result	Live video is shown with the skeleton overlaid on the video, counting the reps, and labeling in real-time (e.g., "Correct," and "Lean Back").
Actual Result	The camera turned on, the reps were counted, and the error in form, "Lean Back", was identified in real-time.
Status	Success
Remarks	N/A

6.10.8 Test of Professional Plan Creation

This test verifies that professionals can create and manage workout plans.

Table 6.9: Test case of Professional Plan Creation

Test Case ID	TC-08
Description	A professional will create a workout plan.
Requirement	Our system will save the details of the plan for purchase later.
Steps to be Taken	1. Go ahead and log in as an approved Professional. 2. Select "Create Plan." 3. Populate the plan details (title, description, price, category, upload a PDF file if desired). 4. Save the plan.
Expected Result	Your plan will be saved to the database and displayed on the "Uploaded Plans" area for you and the customer plan browsing area.
Actual Result	Congratulations, you have successfully created your plan, and it will be accessible in your Professional dashboard and the customer's plan browsing area.
Status	Success
Remarks	N/A

6.10.9 Test of Professional Earnings Tracking

This test ensures professionals can track their earnings and wallet balance.

Table 6.10: Test case of Earnings Tracking

Test Case ID	TC-09
Description	Professional checks Earnings and Wallet balance.
Requirement	Wallet should reflect accurate balance from completed payments.
Steps to be Taken	1. Successfully logged in as Professional. 2. Proceeded to Earnings and Wallet section. 3. Reviewed current balance and transaction history.
Expected Result	Wallet shows correct balance (90% of All payments received). Transaction History shows all payments made, and details on dates and amounts of payment.
Actual Result	Balance calculated correctly. Transaction history displayed all completed payments with appropriate transaction details.
Status	Success
Remarks	N/A

6.10.10 Test of Admin Professional Approval

This test completes the professional registration workflow from the admin side.

Table 6.11: Test case of Admin Professional Approval

Test Case ID	TC-10
Description	The admins review and approve one professional application each time.
Requirement	The admin can view the application and change the professional status.
Steps to be Taken	1. Login as administrator into the application. 2. Go to trainer requests. 3. Review a pending application and certifications. 4. Approve the application.
Expected Result	Transformation process: The status changes from "pending" to "approved." The professional can now access the full professional every moment.
Actual Result	The application is successful in the owner dashboard immediately updated as having accessed the professional fully.
Status	Success
Remarks	N/A

6.10.11 Test of Admin User Management

This test confirms that an administrator can view and manage all users on the platform.

Table 6.12: Test case of Admin User Management

Test Case ID	TC-11
Description	An admin accesses the user list and removes a user.
Requirement	It is essential that administrators have the ability to view and delete users in the system.
Steps to be Taken	1. Sign into the Flutter app as an Admin. 2. Go to the User Management screen. 3. Apply filters to find and select a test user. 4. Remove the selected user.
Expected Result	The user list appears as intended, and upon deletion, both the user's account and related data are erased from Firestore.
Actual Result	The loading of the user list was successful, confirming that the chosen test user has been permanently removed from the database.
Status	Success
Remarks	N/A

Chapter 7

Conclusions

7.1 Conclusion

The Wellness AI app represents a significant advancement in digital fitness technology, seamlessly linking expert knowledge with personalized workout modifications. This comprehensive platform integrates AI-based exercise evaluations with a flexible system that delivers customized fitness guidance to users, equips professionals with efficient client management tools, and offers administrators a detailed view of overall platform activities.

One of the system's key accomplishments is its capability to offer real-time and precise analysis of posture for various exercises such as bicep curls, squats, lunges, and planks. It boasts an impressive average accuracy rate of 94% in identifying form errors and assessing different stages of exercises by utilizing a mix of traditional machine learning techniques and advanced deep learning models.

Furthermore, the smooth integration between the Flutter mobile app, Vue.js web dashboard, and FastAPI backend guarantees a unified experience across multiple platforms. This project highlights how computer vision and machine learning technologies can effectively address practical challenges within the fitness sector.

7.2 Future Work

7.2.1 Push Notifications System Implementation

The use of Firebase (FCM) for messaging would greatly improve user engagement and retention through effective notifications. It would give users automated reminders about upcoming training sessions, suggested workouts based on user email activity, and motivational notifications to help users stick to their schedule. For professionals, push notifications can serve as direct alerts for new appointments, cancellations, and messages from clients. This allows them to react quickly and efficiently cater to their clients. Meanwhile, administrators can utilize push notifications to communicate updates throughout the platform, inform users of policy or terms changes, and highlight marketing promotions. To effectively roll out this feature, it's essential to segment activities according to user preferences. Moreover, timing plays a key role in preventing users from being overwhelmed by excessive

notifications. By prioritizing messages from different sources, it ensures that notifications stay relevant, unobtrusive, and genuinely improve the overall user experience.

7.2.2 Integrated Video Calling Platform

Introducing a strong video calling feature would revolutionize virtual training sessions by enabling smooth, in-platform communication between trainers and clients. This upgrade would remove the necessity for external video apps, fostering a cohesive environment where workout plans, progress tracking, and live interactions exist within one system. To make this happen, it would be essential to choose a dependable WebRTC provider that guarantees low-latency video streaming across different network conditions while also adding session recording options for later analysis. Other features could include screen sharing for exercise demonstrations, virtual whiteboards to clarify techniques, and integrated payment processing to monitor session duration effectively. Such capabilities would greatly enhance initial assessments, technique improvement meetings, and ongoing virtual training engagements.

7.2.3 Comprehensive Analytics and User Behavior Tracking

Creating an advanced analytics framework would yield valuable insights into user engagement trends, feature usage, and the overall effectiveness of the platform. This system would monitor essential metrics like user retention rates, most popular exercises, average session length, and prevalent form errors across various user groups. For professionals in the field, analytics would uncover client engagement behaviors, preferred training times, and revenue trends.

This implementation process would require integrating analytics SDKs (Software Development Kits), establishing custom event tracking for particular workout interactions, and designing visualization dashboards tailored to different stakeholders' needs. The resulting insights could guide product development choices, pinpoint areas needing improvement, and reveal opportunities for enhancing features based on real usage data rather than mere assumptions.

7.2.4 Robust Offline Support and Data Synchronization

Incorporating extensive offline features would greatly improve accessibility for users who have inconsistent internet connections or prefer to work out in outdoor settings. This enhancement would enable individuals to access previously downloaded workout routines, view exercise demonstrations, and monitor key workout metrics without needing a continuous internet connection.

From a technical standpoint, this entails developing advanced strategies for conflict resolution during data synchronization, fine-tuning local storage management for mobile devices, and implementing smart caching systems for content that is frequently accessed. The synchronization system must be capable of resolving merge conflicts when the same information is altered both online and offline. It should also prioritize the transmission of

essential data while providing clear visual indicators showing the status of sync processes to ensure users feel confident in the reliability of the system.

7.2.5 Advanced Chat System with Media Support

Improving the existing chat functionality with enhanced features would greatly elevate the quality of interaction between users and professionals. The updated system could include file sharing options, enabling professionals to transmit tailored workout plans, exercise demonstrations, or educational materials directly within the chat. Users would also benefit from image and video sharing capabilities, allowing them to submit form check videos and receive feedback that is both time-stamped and visually annotated. Additional improvements might feature read receipts, persistent messages, an organized chat history with search capabilities, and quick-response templates for frequently asked questions. For discussions focused on specific exercises, the chat could connect with an AI analysis system so that professionals could pinpoint particular form errors identified in uploaded videos and offer relevant guidance within the ongoing conversation.

References

[1] Jonathan Smith and Emily Roberts. Digital fitness: The growing role of technology in exercise and wellness. *Journal of Sports Technology*, 8:45–58, 2020. Cited on pp. 1 and 2.

[2] Yuwei Chen, Lihua Wang, and Heng Zhang. Ai-based human pose estimation for exercise monitoring and feedback. *IEEE Access*, 9:110123–110134, 2021. Cited on pp. 1, 9, and 12.

[3] Google LLC. Flutter documentation. https://flutter.dev, 2023. Cited on p. 1.

[4] Google LLC. Firebase documentation. https://firebase.google.com, 2021. Cited on p. 1.

[5] Evan You. Vue.js: The progressive javascript framework. https://vuejs.org, 2020. Cited on p. 1.

[6] Django Software Foundation. Django web framework documentation. https://www.djangoproject.com, 2022. Cited on p. 1.

[7] Camillo Lugaresi, Jiuqiang Tang, and et al. Mediapipe: A framework for building perception pipelines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019. Cited on pp. 1, 5, 11, 12, and 64.

[8] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Fan Zhang, Eduard Vakunov, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking. In *arXiv preprint arXiv:2006.10204*, 2020. Cited on pp. 2 and 6.

[9] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2022. Cited on p. 5.

[10] Amit Kumar and Priya Sharma. Hybrid mobile app architecture using flutter and firebase. *International Journal of Mobile Computing*, 15(3):22–35, 2020. Cited on p. 34.

[11] Minho Lee and Hanul Park. Designing scalable web systems using vue.js frontend and django backend. *Journal of Web Engineering*, 21(1):55–72, 2022. Cited on p. 34.

[12] Camillo Lugaresi and et al. Mediapipe: A framework for building perception pipelines. *arXiv Preprint*, arXiv:1906.08172, 2019. Cited on pp. 34 and 35.

[13] Google Inc. Flutter: Ui toolkit for building natively compiled applications, 2020. Cited on p. 63.

[14] Google Inc. Firebase: Scalable backend services, 2020. Cited on p. 63.

[15] Cloudinary. Cloudinary media management api, 2020. Cited on p. 64.

[16] Gary Bradski. Opencv. In *The Learning OpenCV Book*, 2000. Cited on p. 64.

[17] Fabian Pedregosa and et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Cited on p. 64.

[18] Martín Abadi and et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Cited on p. 64.

[19] François Chollet. *Deep Learning with Python*. Manning Publications, 2018. Cited on p. 64.

[20] Evan You. Vue.js: The progressive javascript framework, 2021. Cited on p. 65.

[21] W3C. Webrtc 1.0: Real-time communication between browsers, 2018. Cited on p. 65.

[22] Sebastián Ramírez. Fastapi: High performance rest framework, 2020. Cited on p. 65.

11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

140 Not Cited or Quoted 11%
Matches with neither in-text citation nor quotation marks

10 Missing Quotations 1%
Matches that are still very similar to source material

1 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
2% Publications
11% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.