Year	3rd
Semester	5th
Session	2025

TherapEase

Physiotherapy made easy

Submitted by

Keshav Pal
Somay Singh
Shivam Yadav
Kunwar Harshit Singh

Submitted to

Ms. Uma Ojha

DEVELOPED BY

KESHAV PAL (23/38046)

SOMAY SINGH (23/38024)

SHIVAM YADAV (23/38040)

KUNWAR HARSHIT SINGH (23/38082)

Contents

Cover Page

Contents

1 PROBLEM STATEMENT

1.1 Overview

The services related to Physiotherapy in large hospitals (such as AIIMS), generally face overcrowding and logistical challenges, which makes it difficult for the patients to receive care on time. Most of the patients live far away from the hospital, and require repeated travels for therapy sessions, which can negatively affect their treatment and recovery.

1.2 Challenges in the Current System

1. **Overcrowded Hospitals:** Physiotherapists handle a large number of patients, which further leads to longer waiting times and also reduces one-on-one attention towards certain patients with critical conditions.
2. **Limited Accessibility:** A patient who lives at a far distance from hospitals, faces difficulty in attending regular sessions, which causes interruptions in their therapy.
3. **No Proper Monitoring System:** Performance of exercises at home is not tracked properly (like poses, incorrect rep count), hence, it makes hard for the doctor to assess progress between every visits.
4. **Delayed Feedback:** Any errors in exercise execution go unnoticed until the patient visits the hospital again along with their progress reports, which may delay recovery.
5. **Inadequate Reporting:** There is no systematic way to generate detailed reports on patient adherence, progress, or exercises' pose correctness.
6. **Lack of Remote Consultation:** Patients cannot easily consult doctors remotely for changes in therapy, resulting in repeated hospital visits.

These challenges often create a hurdle for efficient treatment, patient convenience, and effective monitoring. A new system is required to be introduced which provides real-time tracking and progress reporting both patients and doctors.

1.3 Proposed Solution

To solve these common challenges, a new At-Home Physiotherapy Monitoring App named as TherapEase is proposed. It uses modern mobile and web technologies to make therapy easier, more accessible, and trackable in real time. The main features of this app include:

1. **Exercise Pose Tracking:** The app will monitor patients' exercises through camera or sensor input, providing feedback on their performances and progressions.
2. **Automated Progress Reports:** All exercise data will be automatically collected and converted into a clear, structured PDF report. Doctors can quickly review the patient's progress, consistency, and performance.
3. **Remote Prescription Updates:** Doctors can review the reports and adjust exercise plans, intensity, or frequency without needing an in-person visit. This keeps the treatment updated and personalized.
4. **Reduced Hospital Visits:** By enabling home based therapy, this app will greatly reduce the need for frequent hospital travel, easing overcrowding and avoid unnecessary financial burden on travel expenses.
5. **User-Friendly Interface:** The app will be designed for all the age groups, with simple and easy to learn navigation, exercise instructions, and reminders to improve their posture for speedy recovery and correct rep count.
6. **Data Security and Privacy:** All patient data will be stored securely and encrypted, complying with medical data privacy standards.
7. **Mobile and Web Accessibility:** Patients and doctors can access the platform from phones, tablets, or desktops. Everything—from exercise guidance to reports—is just one click away, making it highly convenient.

1.4 Conclusion

The current physiotherapy system struggles with overcrowding, limited accessibility, and the lack of proper monitoring tools, which directly affects patient recovery and overall treatment efficiency. The proposed TherapEase application addresses these issues by introducing a

modern, technology-driven approach that enables real-time tracking, automated reporting, and remote doctor–patient interaction.

By reducing unnecessary hospital visits, providing accurate feedback on exercises, and ensuring consistent monitoring from home, TherapEase enhances both patient convenience and treatment quality. Its secure data handling, user-friendly design, and accessibility across multiple devices make it a practical and scalable solution for hospitals and physiotherapy departments.

In summary, TherapEase has the potential to transform the traditional physiotherapy workflow into a smarter, more efficient, and patient-centric system—ultimately improving recovery outcomes and reducing the burden on healthcare facilities.

2. PROCESS MODEL : AGILE

Given the dynamic nature of our application and the need for constant improvements, the Agile Development Model is the most suitable approach for developing the TherapEase. Agile allows for iterative development, frequent feedback, and real-time changes based on user needs and feedback.

2.1 Phases of the Agile Process

2.1.1 Requirement Gathering & Analysis

- ✓ Gather detailed requirements from patients, physiotherapists, and hospital administration to understand the major challenges in the current physiotherapy workflow.
- ✓ Collect feedback through surveys to identify the essential features needed in the system, such as exercise pose tracking, automated progress reporting, remote prescription updates, and appointment management.
- ✓ Analyze the need for monitoring, offline accessibility (e.g., downloadable PDF reports), secure data storage, and seamless doctor–patient communication to ensure the system meets practical and clinical expectations.

2.1.2 Planning

- ✓ Divide the project into ‘sprints’, with each sprint focusing on delivering specific features.
- ✓ Create a product backlog to manage priorities and ensure mandatory features.

2.1.3 Design

- ✓ Create mockups for the patient and doctor interface, ensuring user-friendly navigation.
- ✓ Design a machine learning model that supports reliable verification of different exercises.
- ✓ Initially for mobile accessibility.

2.1.4 Development

- ✓ Development to be carried out incrementally, with each spring focusing on completing a functional part of the system.
- ✓ Regular meetings to track progress.
- ✓

2.1.5 Testing

- ✓ Each feature is tested individually as it's developed.
- ✓ Testing includes ensuring that the model can filter out exercises correctly and able to produce feedback on each exercise session.
- ✓ Perform user testing to ensure the interface is clear and easy to use.

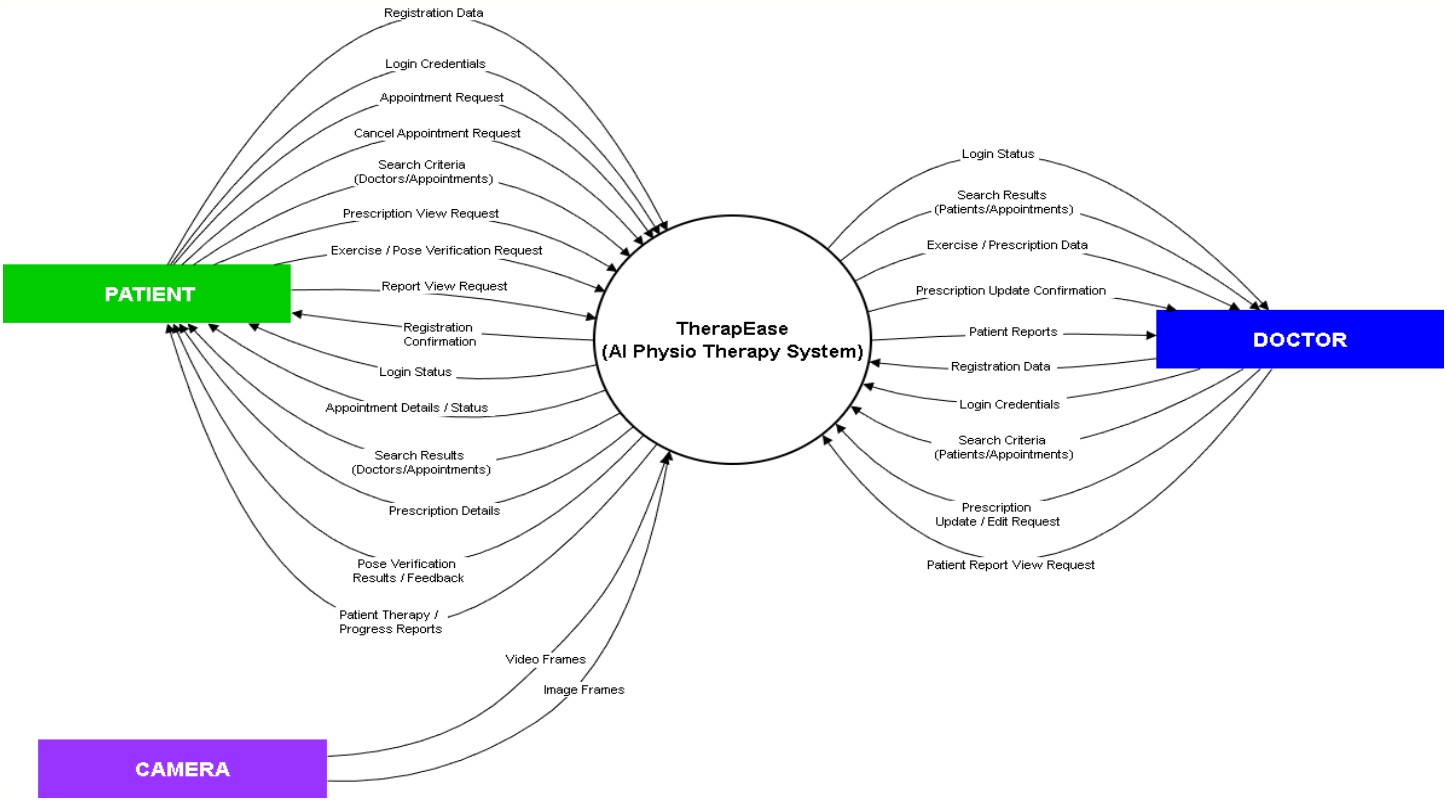
2.1.6 Deployment

- ✓ Once testing is complete, the system is deployed incrementally, with new features being rolled out after each sprint.
- ✓ A feedback loop to address any bugs or errors.

3 SEQUENCE AND DATA FLOW

3.1 Data Flow Diagram Level 0

The DFD Level 0 for the Physiotherapy Monitoring App provides a high-level overview of the system, showing how patients, physiotherapists, and doctors interact with the system to track exercises, receive feedback, and update prescriptions.



3.2 External Entities:

- **Patient:** Performs exercises at home and receives feedback.
- **Doctor/Physiotherapist:** Monitors patient progress, reviews reports, and updates therapy prescriptions.

3.3 Major Processes:

- Exercise Tracking: Captures patient's exercise performance data.
- Feedback Generation: Analyzes exercise performance and provides feedback to the patient.
- Report Generation: Compiles exercise performance information into a structured report for doctor.
- Prescription Update: Allows the allotted doctor to modify therapy instructions based on the patient's progress.

3.4 Data Stores

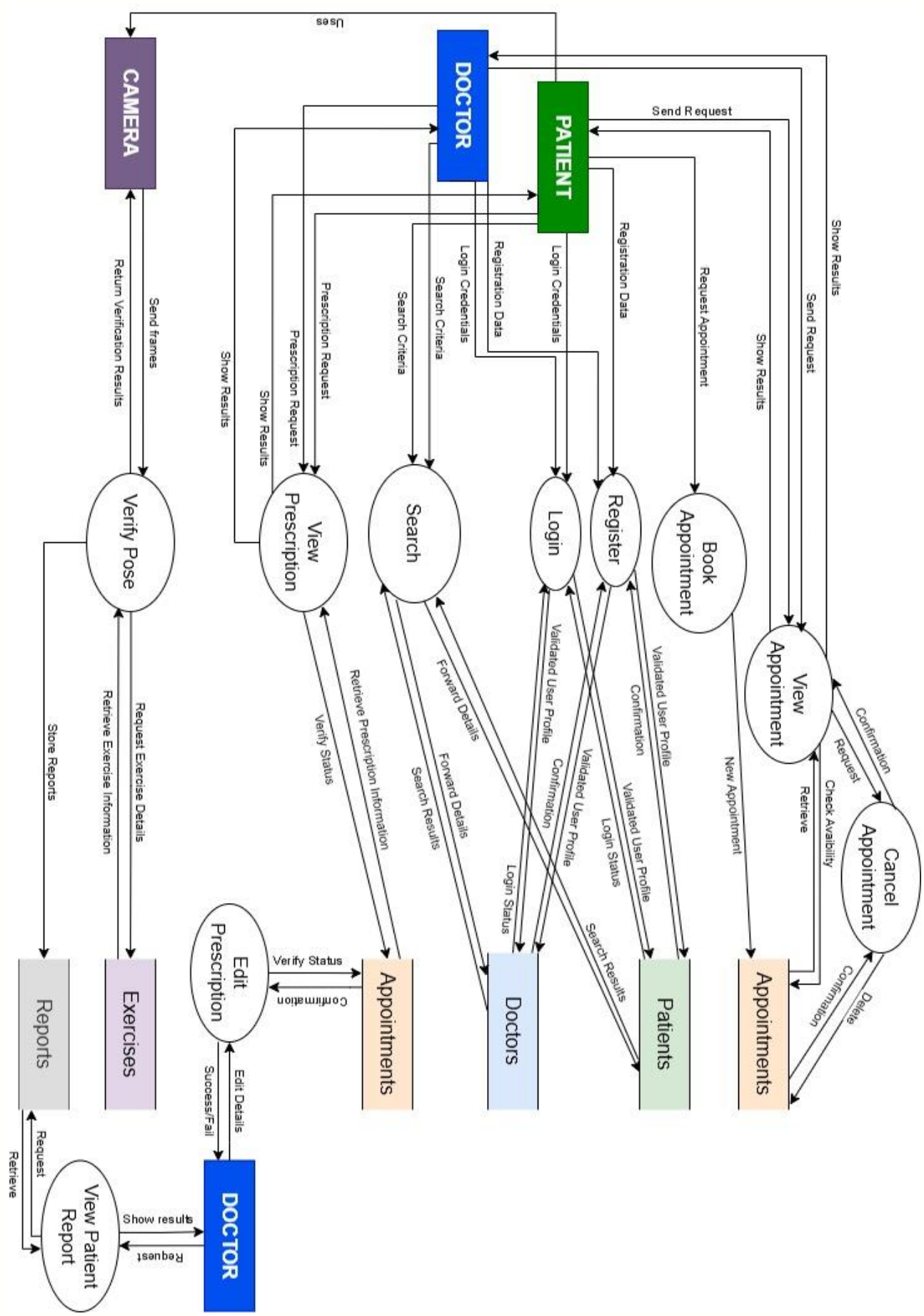
- **Patient:** Contains patient profiles, physical details, and login credentials.
- **Doctor:** Stores doctor profiles, qualifications, availability, and credentials.
- **Exercises:** Holds exercise names, types, and pose-related information.
- **Appointments:** Keeps track of scheduled dates, times, and assigned exercises.
- **Reports:** Saves feedback and performance results from each exercise session.

3.5 Interactions

- Patients enter their exercise data → the Exercise Tracking system analyzes it → feedback is immediately returned to the patient.
- Doctors view the generated reports → update or modify prescriptions → updated details are stored in the Appointments data store.

3.6 DFD Level 1

On the next page, DFD Level 1 offers a clearer and more detailed breakdown of how our Physiotherapy App, TherapEase, functions behind the scenes. It expands the high-level processes into specific subprocesses to show exactly how information moves through the system at each stage. It shows how patients begin by recording their exercises through our app, after which the system captures posture data, analyze their movements, and evaluate performance accuracy and structured progress reports.


CAMERA
DOCTOR
PATIENT
Uses
Send Request
Show Results
Send Request
Show Results
Registration Data
Request Appointment
Login Credentials
Registration Data
Login Credentials
Search Criteria
Prescription Request
Show Results
Show Results
Prescription Request
Search Criteria
Return Verification Results
Send frames
Verify Pose
View Prescription
Search
Login
Register
Book Appointment
View Appointment
Cancel Appointment
Validated User Profile
Confirmation
Validated User Profile
Login Status
Validated User Profile
Confirmation
Forward Details
Search Results
Forward Details
Retrieve Prescription Information
Verify Status
New Appointment
Check Availability
Retrieve
Confirmation
Request
Delete
Confirmation
Store Reports
Retrieve Exercise Information
Request Exercise Details
Edit Prescription
Verify Status
Confirmation
Reports
Exercises
Appointments
Doctors
Patients
Appointments
Success/Fail
Edit Details
Request
Retrieve
View Patient Report
Show results
Request
DOCTOR

3.7 Detailed Processes:

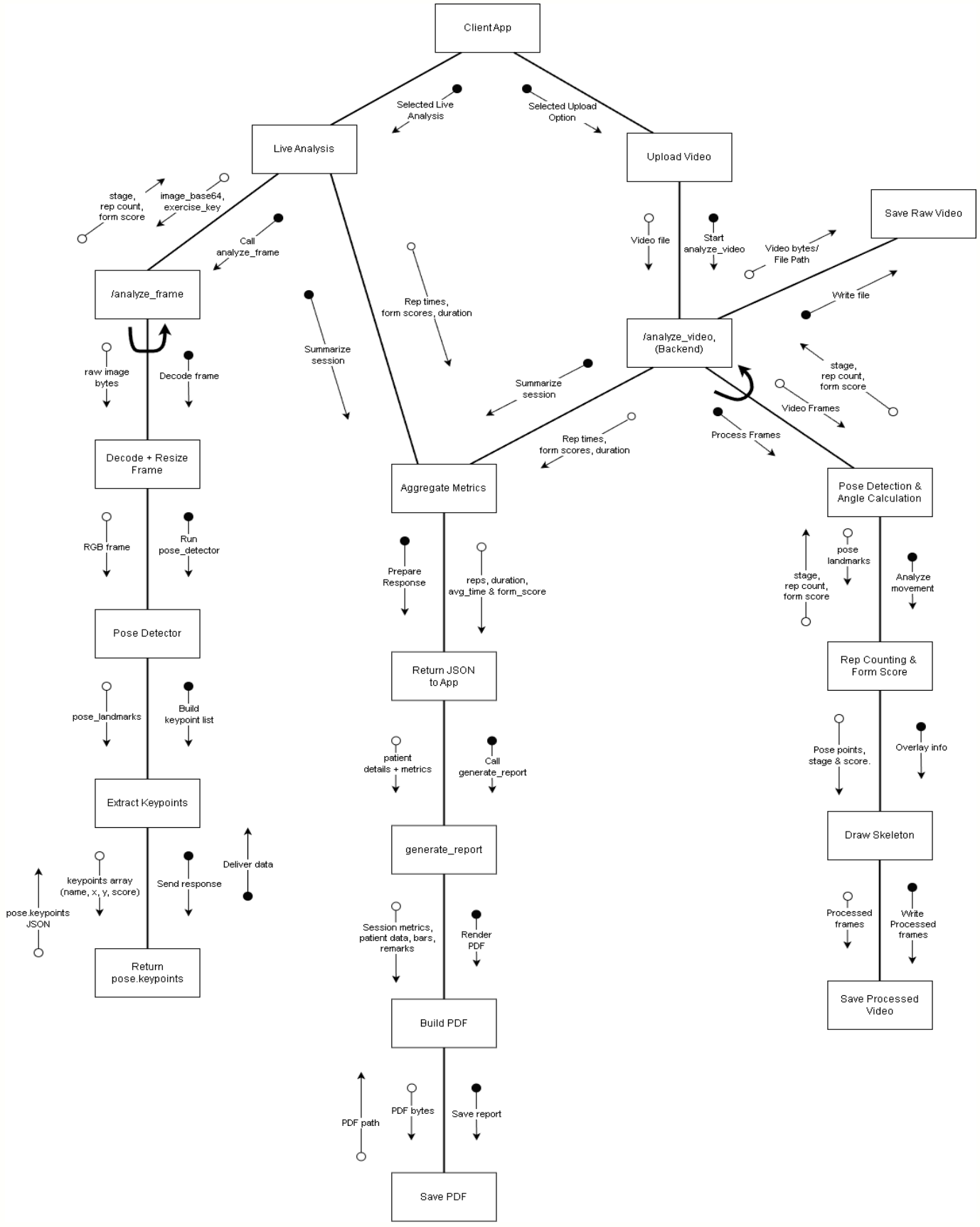
1. **Exercise Tracking Subprocesses:**
    a. Capture Video/Sensor Data
    b. Analyze Movements Against Prescribed Exercise
    c. Detects Error and Pose deviations

2. **Feedback Generation Subprocesses:**
    a. Generate Reports
    b. Provide the exercise form feedback to the patient
    c. Log Feedback into Reports Data Store

3. **Report Generation Subprocesses:**
    a. Compile Exercise Logs
    b. Calculate Pose Correctness and Performance Scores
    c. Send Reports to the allotted doctor

4. **Prescription Update Subprocesses:**
    a. Doctor Reviews Report
    b. He/She modifies Therapy or Exercises
    c. Writes a personal feedback based on the reports provided
    d. Stores Updated Prescription in Appointments Data Store

3.8 Structured Chart

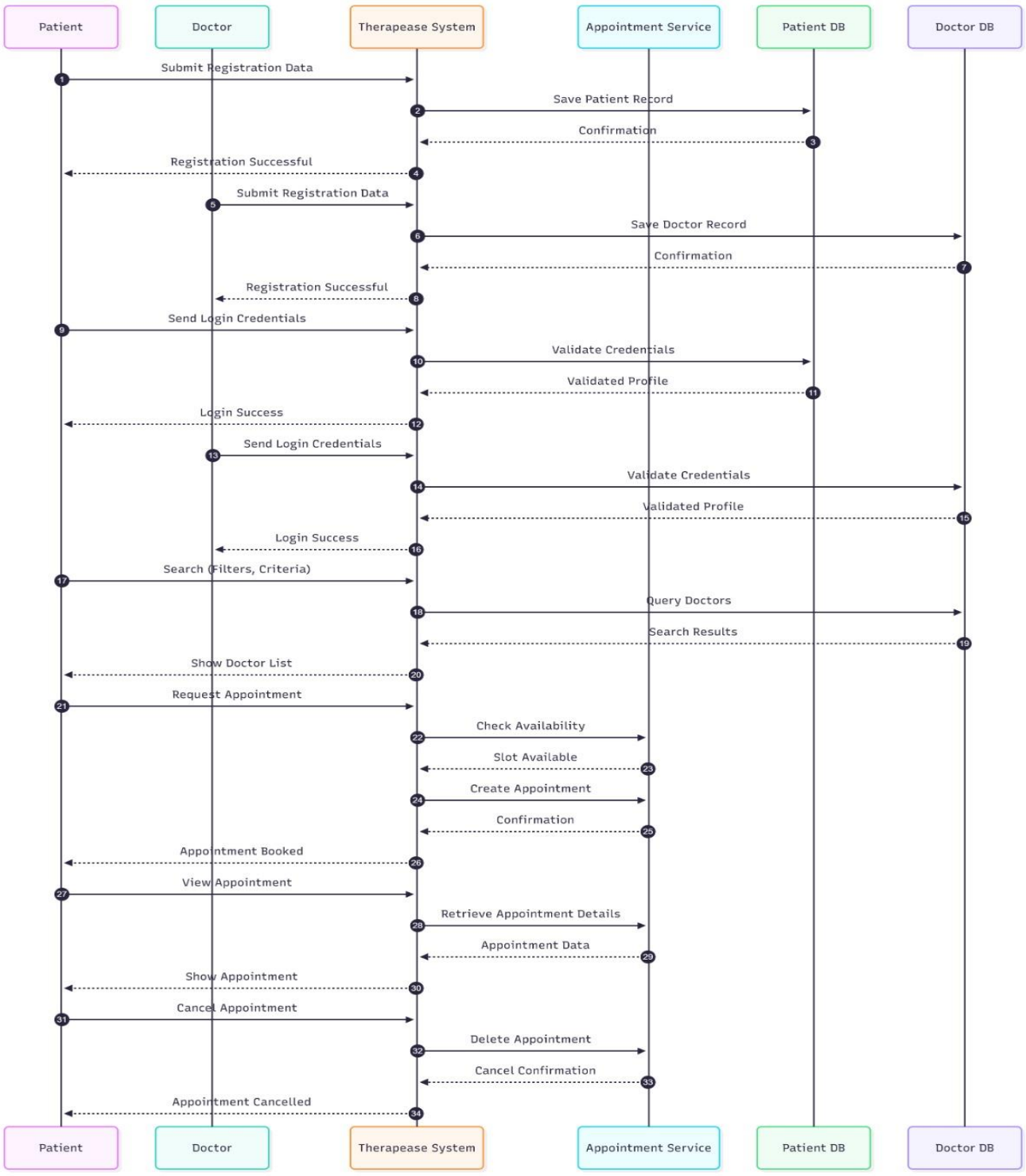


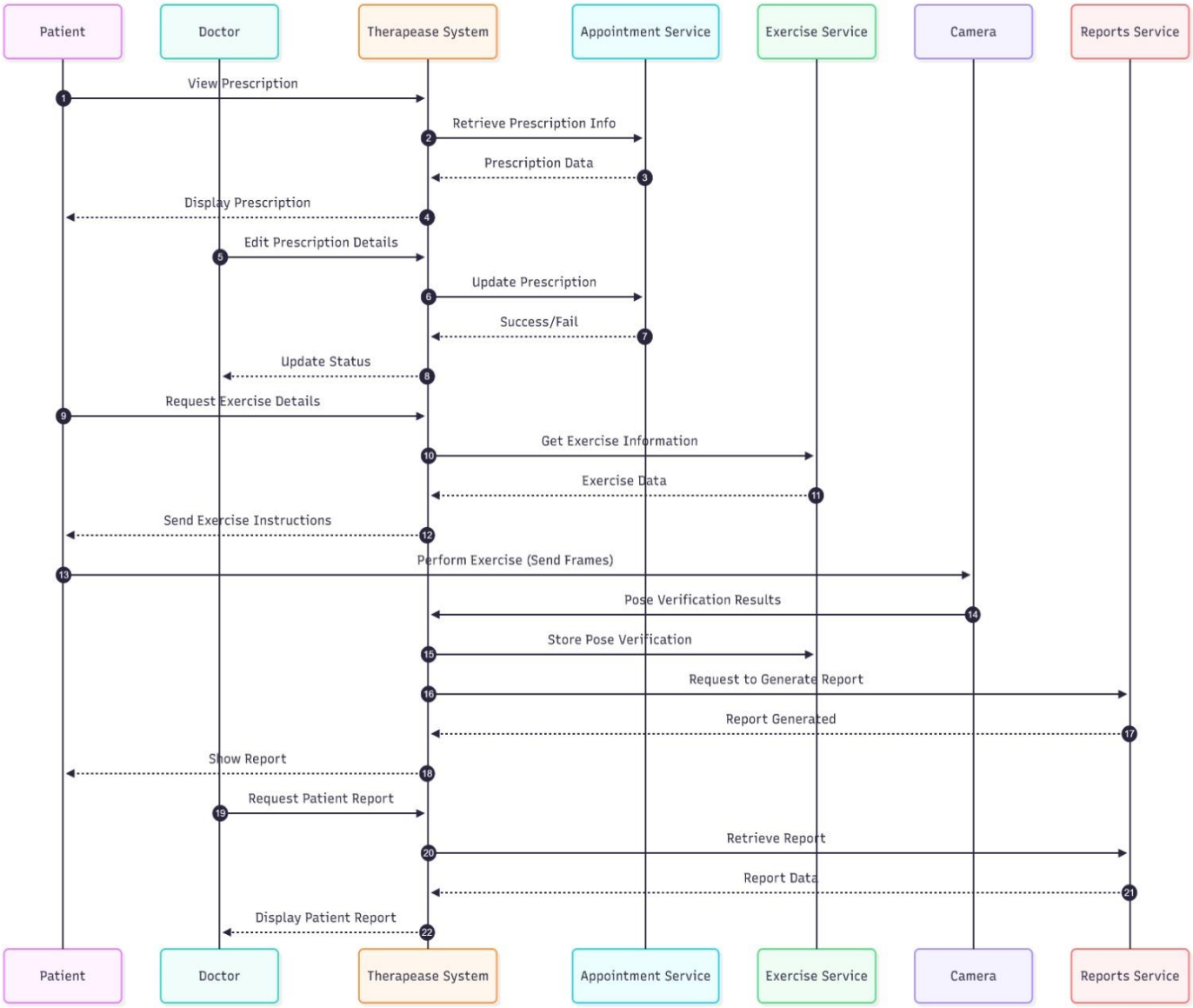
Modularity and Maintainability Insights

The structure chart clearly demonstrates that the backend is organized into well-defined modules, each responsible for a single, specific task such as pose detection, angle calculation, rep counting, video writing, and PDF generation. By separating the system into control modules, submodules, and reusable library modules, the design minimizes coupling and ensures that changes in one part do not affect others. The repeated operations in the video-analysis loop are isolated inside their own submodules, making the logic easier to understand and modify. This modular decomposition improves maintainability, supports easier debugging and testing, and allows individual components to be reused or upgraded independently without impacting the overall system.

3.9 Sequence Diagram

This sequence diagram explains the flow for patient and doctor registration, authentication, searching for doctors/patients, and booking appointments. It highlights how the Therapease System, At-Home Physiotherapy communicates with backend services such as the Appointment Service, Patient Database, Detection Service, and Doctor Database.




Patient
Doctor
Therapease System
Appointment Service
Exercise Service
Camera
Reports Service
1 View Prescription
2 Retrieve Prescription Info
3 Prescription Data
4 Display Prescription
5 Edit Prescription Details
6 Update Prescription
7 Success/Fail
8 Update Status
9 Request Exercise Details
10 Get Exercise Information
11 Exercise Data
12 Send Exercise Instructions
13 Perform Exercise (Send Frames)
14 Pose Verification Results
15 Store Pose Verification
16 Request to Generate Report
17 Report Generated
18 Show Report
19 Request Patient Report
20 Retrieve Report
21 Report Data
22 Display Patient Report
Patient
Doctor
Therapease System
Appointment Service
Exercise Service
Camera
Reports Service

4 DATA DICTIONARY

4.1 ENTITY: PATIENTS

PATIENTS = `id + email + password + name + dob + height_cm + weight_kg + location + activity + bio + badges_id + gender + joined_at`

Attribute	Description	SQL Data Type	Example
id	Unique patient identifier	INT PRIMARY KEY	1024
email	Email ID	VARCHAR(100)	riya@example.com
password	Encrypted password	VARCHAR(255)	34$2328AJDK3##
name	Full name of patient	VARCHAR(50)	Riya Sharma
dob	Patient's birth date	DATE	2005-05-02
height_cm	Height in centimeters	INT	167
weight_kg	Weight in kilograms	INT	55
location	Location/address.	TEXT	Delhi
activity	Activeness scale	ENUM	minimal
bio	A little bit of profile enhancement	TEXT	I love football!
badges_id	Awards to signify achievements	TEXT	100 REPS
gender	Gender of the patient	ENUM	female
joined_at	Date on which patient joined the app	DATE	2025-25-11

4.2 ENTITY: EXERCISES

EXERCISES = `id + name`

Attribute	Description	SQL Data Type	Example
id	Unique report identifier	INT PRIMARY KEY	0006
appointment_id	Appointment identifier	INT	001
patient_id	Patient identifier	INT	123
doctor_id	Doctor generating/reviewing report	INT	112
exercise_id	Exercise identifier	INT	2025-10-07
feedback	Feedback provided by the system	TEXT	Form can be better
score%	Overall score for the session	INT	65
created_at	Report generation date	DATE	2025-12-01

4.3 ENTITY: DOCTORS

DOCTORS = `id + email + password + name + age + specialty + experience + location + availability + bio + gender + joined_at`

Attribute	Description	SQL Data Type	Example
id	Unique doctor identifier	INT PRIMARY KEY	11
email	Email ID	VARCHAR(100)	anil3433@gmail.com
password	Encrypted password	VARCHAR(255)	4$$HELL032##RE
name	Full name of doctor	VARCHAR(50)	Anil Kumbal
specialty	Area of expertise	TEXT	Shoulder
experience	Years of experience.	INT	3
dob	Doctor’s date of birth	DATE	1990-11-11
location	Address	TEXT	Faridabad
availability	When doctor is active	ENUM	weekdays
bio	Doctor profile text	TEXT	An apple a day :(
gender	Gender of the doctor	ENUM	Male
joined_at	Date on which doctor joined the app	DATE	2025-11-11

4.4 ENTITY: APPOINTMENTS

APPOINTMENTS = id + doctor_id + patient_id + description_patient + description_doctor + appointment_date + appointment_time + exercise_id + reps + days

id	Unique identifier for appointment	INT PRIMARY KEY	002
doctor_id	Doctor allotted	INT	211
patient_id	Patient involved	INT	100
description_pation	Patient-provided notes for appointment.	TEXT	I am suffering from asthma.
description_doctor	Doctor's notes or diagnosis for the appointment.	TEXT	Do it before lunch.
appointment_date	Scheduled date	DATE	2025-11-20
appointment_time	Scheduled time	TIME	15:30
exercise_id	Prescribed exercise for this session	INT FOREIGN KEY	2
reps	Number of repetitions prescribed.	INT	30
days	For how many days prescribed for	INT	5

4.5 ENTITY: REPORT

REPORTS = id + appointment_id + patient_id + doctor_id + exercise_id + feedback + score + created_at

Attribute	Description	SQL Data Type	Example
id	Unique report identifier	INT PRIMARY KEY	0006
appointment_id	Appointment identifier	INT	001
patient_id	Patient identifier	INT	123
doctor_id	Doctor generating/reviewing report	INT	112
exercise_id	Exercise identifier	INT	2025-10-07
feedback	Feedback provided by the system	TEXT	Form can be better
score	Overall score for the session	TEXT	65%
created_at	Report generation date	DATE	2025-12-01

5 USE CASE DIAGRAM

5.1 Use Case Diagram: TherapEase System

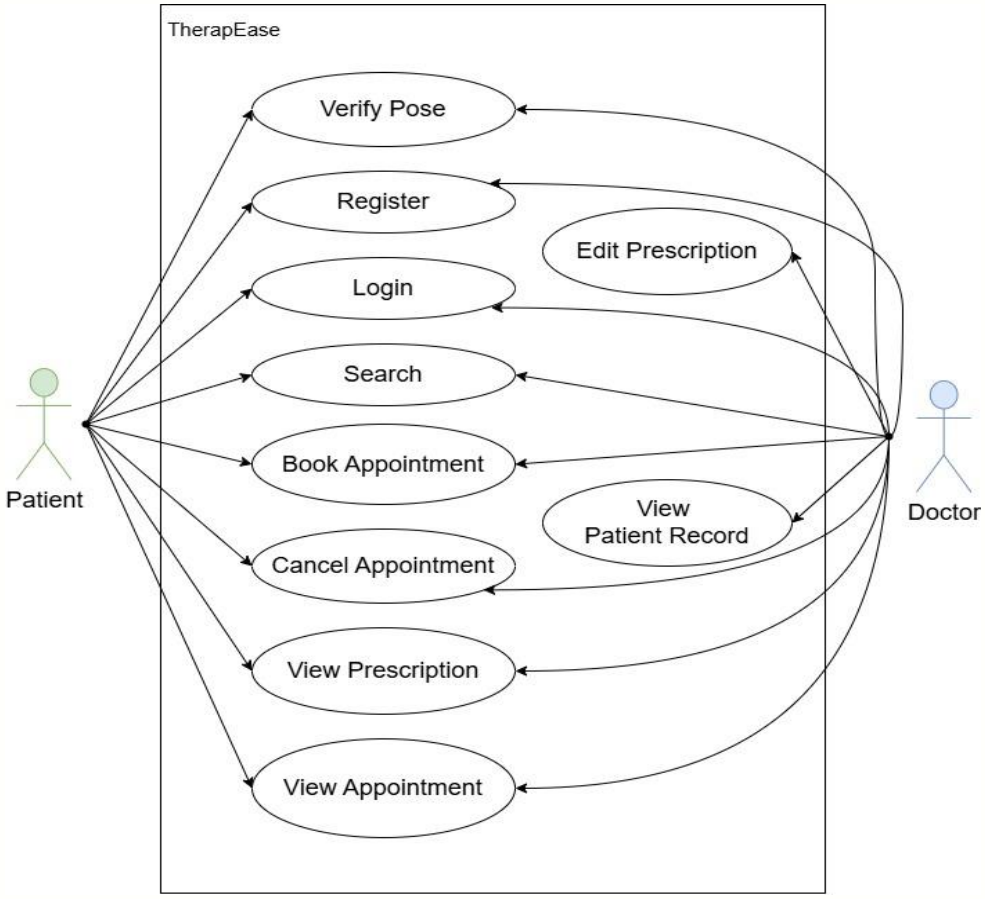
5.1.1 Description

The Use Case Diagram for the TherapEase system represents the interaction between different types of users and the clinical functionalities they can access. The primary actors identified are:

• Patient: Uses the system to register, log in, search doctors, book and manage appointments, view prescriptions, perform guided exercise sessions, and track progress through reports.

• Doctor: Uses the system to manage their profile, view patient information, create and update prescriptions, define exercise plans, and review patient progress and reports.

• TherapEase Admin: Manages users and system configurations, ensuring that the platform remains secure, consistent, and up to date.



The diagram highlights how these actors interact with the TherapEase system to support end-to-end physiotherapy workflows, from appointment booking to exercise tracking and reporting.

5.1.2 Main Features and Functionalities

1. Registration and Authentication:
   • Patients, doctors, and admins can register and log in to the system.
   • Users can manage their profile information after authentication.
2. Appointment Management:
   • Patients can search for doctors and view doctor details.
   • Patients can book appointments after checking doctor availability.
   • Patients can view and cancel their upcoming appointments.
3. Prescription and Exercise Plan Management:
   • Doctors can view and update patient prescriptions.
   • Doctors can define or modify exercise plans based on the patient's condition.
   • Patients can view the latest prescribed exercises and related instructions.
4. Exercise Tracking and Posture Evaluation:
   • Patients can start a guided exercise session using the system.
   • The system evaluates exercise form (e.g., posture and angles) during the session.
   • The system updates internal data such as repetition counts, stages, and form quality.
5. Progress and Reporting:
   • Both patients and doctors can view progress and performance reports.
   • The system generates reports using tracked exercise data and past sessions.
6. Administration and System Management:
   • The admin manages user accounts (patients and doctors).
   • The admin configures system settings and monitors overall system behavior.

Figure 5.5: Use Case Diagram: TherapEase System

5.1.3 Actors and Their Roles

• Patient: Registers, logs in, searches for doctors, books and manages appointments, views prescriptions and exercise instructions, performs exercise sessions, and reviews progress reports.

• Doctor: Registers, logs in, manages their profile, views patient prescriptions, updates prescriptions and exercise plans, and reviews patient performance reports.

• TherapEase Admin: Logs in to manage users (patients and doctors), maintain system settings, and ensure smooth operation of the platform.

5.2 Use Case 1: Book Appointment

Use Case Name	Book Appointment
Primary Actor	**Patient**
Preconditions	• Patient is registered and logged in. • Doctor profiles and availability exist in the system.
Main Success Scenario	1. Patient logs into the TherapEase system. 2. Patient selects the option to search for doctors. 3. System displays search filters (specialization, location, availability). 4. Patient enters search criteria and submits request. 5. System retrieves and displays matching doctors. 6. Patient selects a doctor to view details. 7. Patient chooses “Book Appointment”. 8. System displays available time slots. 9. Patient selects a date and time slot. 10. System verifies that the slot is available. 11. System creates and stores the appointment. 12. Confirmation is shown and doctor schedule is updated.
Exception Scenarios	• No doctors found → system displays message and suggests modifying filters. • Selected slot becomes unavailable → system asks user to pick another slot. • Backend/network error while fetching doctors. • Database error during appointment creation.

5.3 Use Case 2: Perform Exercise Session

Use Case Name	**Perform Exercise Session**
Primary Actor	**Patient**
Preconditions	• Patient is registered and logged in. • Patient has an active exercise plan prescribed by a doctor. • Device camera and internet connection are available.
Main Success Scenario	1. Patient opens the TherapEase app and goes to the exercise section. 2. Patient selects an exercise from the prescribed list. 3. Patient taps “Start Exercise Session”. 4. System sends request to backend to begin the session. 5. Backend activates tracking and reads live camera frames. 6. Pose landmarks are detected on each frame. 7. System calculates angles and determines stage + rep counts. 8. System updates session data continuously (angles, reps, form quality). 9. Patient sees live feedback on the app. 10. Patient taps “Stop Session”. 11. Backend finalizes and stores session data. 12. Results become available for future reports.
Exception Scenarios	• Poor lighting or incorrect camera angle → system prompts patient to adjust position. • Network loss → session interrupted, partial data saved. • Camera permission denied → system shows error. • Backend tracking fails to start or stop due to server error.

5.4 Use Case 3: viewProgressReports

Use Case Name	**View Progress Reports**
Primary Actor	**Patient / Doctor**
Preconditions	• Exercise session data already recorded. • User is logged in with correct permissions.
Main Success Scenario	1. User logs into TherapEase. 2. User navigates to “Reports / Progress”. 3. System prompts for time range or session selection. 4. User selects filters and submits request. 5. System retrieves session data for the selected period. 6. System processes data and generates the report (tables/charts/stats). 7. Report is displayed to the user. 8. User may choose to download or export the report.
Exception Scenarios	• No data available for selected range → system shows “No sessions available”. • Database or network failure while retrieving data. • Report generation error leading to incomplete or invalid output.

6 SRS

6.1 Introduction

The Introduction section provides an overview of the project, defining its goals, scope, and terminology. This section serves as a foundation to understand the document's purpose and its relevance to all stakeholders involved in the system.

6.1.1 Purpose

The purpose of this document is to define the Software Requirements Specification (SRS) for the **Physiotherapy Exercise Tracking System (PETS)**.

This document outlines:

- The functional requirements of the system, including exercise tracking, posture analysis, and repetition counting.
- The non-functional requirements such as performance, reliability, usability, and security.
- The interface requirements for users, software components, and hardware such as cameras and mobile devices.

The PETS aims to assist physiotherapists and patients by providing a system that uses pose estimation to monitor physiotherapy exercises. It tracks joint angles, counts repetitions, and provides visual feedback using a skeleton overlay. The system enables real-time monitoring using a webcam (for the backend) and a mobile application for interaction and display.

6.1.2 Scope

The **Physiotherapy Exercise Tracking System (PETS)** is designed to address the challenges of unsupervised physiotherapy at home and basic exercise monitoring in clinics. The system's primary objectives are:

- **Exercise Tracking:** Monitor specific physiotherapy exercises like bicep curls, squats, shoulder abductions, knee extensions, leg raises, and side bends.
- **Pose Analysis:** Use pose estimation to calculate joint angles and determine the stage of movement (e.g., up/down, bent/extended).
- **Repetition Counting:** Automatically count repetitions based on movement patterns.

• **Real-Time Feedback:** Display ongoing exercise metrics such as angle, count, stage, and basic form quality.
• **Media-Based Analysis:** Allow users to upload exercise videos or capture frames for analysis.

Key Users:
• **Patients:** Perform exercises and view real-time feedback.
• **Physiotherapists:** Use the system to demonstrate exercises and review basic tracking.
• **Administrators/Developers:** Manage deployment, configuration, and system maintenance.

6.1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
PETS	Physiotherapy Exercise Tracking System
SRS	Software Requirements Specification
API	Application Programming Interface
UI	User Interface
Pose Estimation	Technique for detecting human joint locations from images or video
Repetition (Rep)	One complete cycle of an exercise movement
Backend	Server-side component (FastAPI, MediaPipe, OpenCV)
Frontend	Client-side mobile application (React Native with Expo)

Table 6.1: Definitions, Acronyms, and Abbreviations

6.1.4 References

• IEEE 830-1998 Standard for Software Requirements Specifications
• MediaPipe Pose official documentation
• FastAPI official documentation
• OpenCV documentation
• React Native and Expo documentation
• Institutional/project-level guidelines for health-related software

6.1.5 Overview

The document is organized as follows:

• **Section 6 (SRS):** Provides the purpose, scope, overall description, and specific requirements of the system.
• **Section 9 (Testing):** Describes unit, system, and user acceptance testing with black box and white box approaches and executed test cases.
• **Section 10 (Risk):** Presents a risk table and mitigation strategies.
• **Section 11 (Feasibility Study):** Evaluates technical, operational, and financial feasibility.

This SRS document serves as a guideline for developers, testers, physiotherapists, and other stakeholders, ensuring clarity and consistency throughout the development lifecycle of PETS.

6.2 Overall Description

This section provides an overarching view of the **Physiotherapy Exercise Tracking System (PETS)**, describing its environment, interactions, and major characteristics. It outlines the product perspective, its users, constraints, assumptions, and requirement distribution.

6.2.1 Product Perspective

The PETS is a **client–server based application** that combines AI-based pose estimation and a mobile-friendly interface.

System Architecture:
• **Backend:** A FastAPI-based server processes frames captured from a webcam or uploaded media. It uses MediaPipe Pose and OpenCV to extract body landmarks, calculate joint angles, determine exercise stage, and count repetitions.
• **Frontend:** A React Native (Expo) mobile application acts as the client, allowing users to select exercises, start/stop sessions, upload or capture media, and visualize data including skeleton overlays.

Position in the Environment:
PETS operates as an independent monitoring tool but can be integrated in the future with:
• Hospital Information Systems
• Electronic Health Record (EHR) systems
• Cloud storage or analytics dashboards

System Interfaces:

PETS interacts with the following systems and components:

- **Camera/Webcam:** For video input to the backend.
- **Mobile Device Hardware:** For video/image capture and display.
- **Network Services:** For HTTP communication between mobile app and backend.

User Interfaces:

The system provides a simple, interactive UI for:

- **Patients:**
  - Choose an exercise from a list
  - Start and stop sessions
  - Upload videos or capture frames
  - View real-time values (angle, count, stage, form) and visual skeleton overlay
- **Physiotherapists:**
  - Demonstrate exercises
  - Observe counts and angles for basic evaluation
- **Administrators/Developers:**
  - Configure backend IP, monitor logs, and maintain the system

UI Design includes:

- A **dashboard-like screen** showing selected exercise and live metrics.
- Buttons for **start session**, **upload video**, **capture frame**, and **stop session**.
- A preview area with video/image and skeleton overlay.

Hardware Interfaces:

- **Client Devices:** Android smartphones (and optionally iOS) running the React Native app.
- **Server Machine:** Laptop or desktop with webcam support to run FastAPI backend.
- **Network Equipment:** Routers/switches for communication between phone and server.

Software Interfaces:

- **Operating Systems:** Android for mobile; Windows/Linux/macOS for server.
- **Libraries:** FastAPI, MediaPipe, OpenCV, NumPy, Axios, React Native, Expo.
- **APIs:** RESTful endpoints for `/start_session`, `/stop_session`, `/data`, `/analyze_frame`.

Communication Interfaces:

- HTTP/HTTPS for API calls between mobile and backend.
- Local network (e.g., Wi-Fi) using IP address like `http://192.168.x.x:8000`.

6.2.2 Product Functions

The primary functions of PETS include:
• **Exercise Selection:** Allow users to choose between supported physiotherapy exercises.
• **Session Management:** Start and stop tracking sessions using the backend webcam.
• **Pose Detection and Angle Calculation:** Use pose estimation to compute angles at key joints (shoulder, elbow, hip, knee, ankle).
• **Repetition Counting:** Apply exercise-specific logic to count repetitions based on angle thresholds and stage transitions.
• **Data Streaming:** Provide current values (angle, count, stage, form, keypoints) through `/data` endpoint.
• **Video/Image Analysis:** Accept uploaded videos or frames and return analysis results.
• **Skeleton Visualization:** Provide keypoint locations so the frontend can draw a skeleton overlay.

6.2.3 User Characteristics

The system is intended for users with varied technical backgrounds:

• **Patients:**
– Basic smartphone proficiency.
– Able to follow on-screen instructions and stand in front of a camera.

• **Physiotherapists:**
– Moderate to advanced technology usage.
– Interested in understanding basic numeric feedback (angle, count, stage).

• **Administrators/Developers:**
– Familiar with networking, deployment, and debugging.

6.2.4 Constraints

• **Environment Constraints:**
– Requires sufficient lighting and clear visibility of the user’s body.
– Camera must capture the full body or relevant limb for accurate detection.

• **Technical Constraints:**
– Backend and mobile must share the same network (at least in development).
– Performance limited by CPU/GPU of the backend machine.

• **Regulatory and Ethical Constraints:**
– Although this is an academic prototype, care must be taken if used on real patients.
– Data privacy and consent practices must be respected.

6.2.5 Assumptions and Dependencies

• Users have access to a smartphone and a stable Wi-Fi connection.
• Only one active user/session at a time (global `current_data` structure).
• Pose estimation model (MediaPipe) works reliably under typical home/clinic lighting.
• The backend IP is correctly configured in the mobile app.

6.3 Specific Requirements

This section provides a breakdown of the functional, external interface, performance, design, and software attribute requirements for PETS.

6.3.1 Functional Requirements

Exercise Management
• The system shall display a list of exercises (bicep_curl, squat, shoulder_abduction, knee_extension, leg_raise, side_bend).
• The system shall allow the user to select exactly one exercise at a time.

Session Control
• The system shall start a new session when the user selects an exercise and initiates start.
• The system shall stop the current session upon user request and release the webcam.

Pose Detection and Analysis
• The system shall capture frames from the webcam during an active session.
• The system shall detect pose landmarks using MediaPipe Pose for each processed frame.
• The system shall calculate joint angles required for the selected exercise.

Repetition and Stage Logic
• The system shall maintain a `stage` value (e.g., “up”, “down”, “bent”, “extended”).
• The system shall increment `count` whenever a complete cycle of motion is detected based on angle thresholds.

Data Provisioning
• The system shall expose the current `angle`, `count`, `stage`, `form`, `exercise`, and `keypoints` to the frontend through the `/data` endpoint.

Media Upload and Analysis

• The system shall accept uploaded video/image files through `/analyze_frame`.

• The system shall return a JSON response including at least `form`, `angle`, `count`, `stage`, and `exercise` (current implementation uses placeholder logic).

Visualization Support

• The system shall provide normalized (x, y) coordinates of detected landmarks so the frontend can render a skeleton.

6.3.2 External Interface Requirements

User Interfaces

- A mobile interface shall show:
  – Selected exercise name.
  – Reporting netrics (angle, count, stage, form).
  – Media uploading confirmation
  – Buttons for starting/stopping sessions, uploading video, capturing frames etc.

Software Interfaces

- The frontend shall interact with the backend via RESTful APIs (`/start_session`, `/stop_session`, `/data`, `/analyze_frame`).
- The backend shall use MediaPipe and OpenCV libraries for pose estimation and frame processing.

Communication Interfaces

• All API calls shall use HTTP over the local network.

• The system should be configurable to use HTTPS in production setups.

6.3.3 Performance Requirements

• The `/data` endpoint should respond within **500 ms** under normal load.

• The system should function acceptably with a single active user on a standard laptop.

6.3.4 Design Constraints

• The backend must be implemented using Python with FastAPI.

• Pose detection must be done using MediaPipe Pose.

• The frontend application must be developed using React Native with Expo.

• The solution must run on mid-range consumer hardware (no GPU dependency assumed).

6.3.5 Software System Attributes

Security

• Only devices that know the backend IP should be able to access the APIs in the current setup.

• For future production use, authentication and HTTPS should be added.

Reliability

• The system shall handle missing detections gracefully (no crash when no person is in frame).

• The system shall reset `count` and `stage` correctly when a new session starts.

Usability

• The UI shall be simple enough for non-technical users (patients).

• All important actions should be available via clearly labeled buttons.

6.4 Glossary

• **Physiotherapy Exercise:** A prescribed physical movement intended for rehabilitation.

• **Joint Angle:** Angle formed between bones at a joint; used to quantify movement.

• **Stage:** A label indicating current phase of motion in an exercise (e.g., “up”, “down”).

• **Skeleton Overlay:** Visual representation of joints and segments drawn over a video or image.

7 FUNCTION POINT ANALYSIS

7.1 Identify and Classify Functions

External Inputs (EI)

User-originated transactions that update or modify system data:

Function	Description	Type
Register	Patient/doctor enters details to create an account	EI
Login	Credentials input for authentication	EI
Book Appointment	Enter appointment request details	EI
Cancel Appointment	Request to cancel or delete an appointment	EI
Edit Prescription	Doctor updates patient prescription	EI
Generate Report	User requests system to generate therapy report	EI
Video Upload / Frames Input	Video/frame data for pose verification	EI

External Inquiries (EQ)

Interactive input + output with **no change to persistent data**:

Function	Description	Type
Search	Search doctors, patients, exercises, appointments	EQ
View Appointment	Retrieve and display appointment details	EQ
View Prescription	Retrieve and display prescription	EQ

External Outputs (EO)

Formatted or derived information sent out from the system:

Function	Description	Type
Appointment Confirmation / Status	Result of booking/cancel operations	EO
Prescription Display	System-generated formatted prescription	EO
Report (PDF / Summary)	Generated patient therapy report	EO

Internal Logical Files (ILF)

Logical data stores **maintained** within the system:

ILF Name
Patients
Doctors
Appointments
Reports
Exercises

External Interface Files (EIF)

No external shared files used i.e.

EIF = 0

7.2 Complexity / Weight Assumptions

Using IFPUG standard complexity weights:

Type	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

Assumption:

Use **Average complexity** when the function interacts with multiple data stores or performs processing.

Use **Low complexity** for simpler or single-step transactions.

7.3 Count and Weight Each Function

External Inputs (EI)

EI Function	Weight
Register	4 (Avg)
Login	3 (Low)
Book Appointment	4 (Avg)
Cancel Appointment	4 (Avg)
Edit Prescription	3 (Low)
Generate Report	3 (Low)
Video Upload	4 (Avg)

Subtotal (EI) = 4 + 3 + 4 + 4 + 3 + 3 + 4 = 25

External Inquiries (EQ)

EQ Function	Weight
Search	3 (Low)
View Appointment	4 (Avg)
View Prescription	4 (Avg)

Subtotal (EQ) = 3 + 4 + 4 = 11

External Outputs (EO)

EO Function	Weight
Appointment Confirmation / Status	5 (Avg)
Prescription Display	4 (Low)
Report Generation	5 (Avg)

Subtotal (EO) = 5 + 4 + 5 = 14

Internal Logical Files (ILF)

ILF	Weight
Patients	10 (Avg)
Doctors	10 (Avg)
Appointments	10 (Avg)
Reports	10 (Avg)
Exercises	10 (Avg)

Subtotal (ILF) = 10 + 10 + 10 + 10 + 10 = 50

External Interface Files (EIF)

Subtotal (EIF) = 0

7.4 Unadjusted Function Points (UFP)

UFP=EI+EO+EQ+ILF+EIF

UFP=25+11+14+50+0=100

7.5 Value Adjustment Factor (VAF)

FP=UFP×VAF

Where:

VAF=0.65+(0.01×ΣFi)

Two scenarios:

Scenario A — Minimal adjustment (unknown GSC values)

Assume each characteristic = 1:

ΣFi =14

VAF=0.65+0.14=0.79

FP=100×0.79=79

Scenario B — Typical medium complexity

Assume each characteristic = 3:

ΣFi =42

VAF=0.65+0.42=1.07

FP=100×1.07=107

7.6 Result

Unadjusted Function Points (UFP) = 100

Adjusted FP (moderate GSC) ≈ 107

This Function Point Analysis reflects the size and complexity of the AI-based Physiotherapy Management System.

Industry-standard productivity rate:

10 Function Points per Person-Month

Total Effort (Person-Months)

Effort = Productivity Rate / Adjusted FP

Effort = 107 / 10 = **10.7 person-months**

Project Cost

Cost per person per month = ₹50,000

Effort = 10.7 person-months

Cost = 10.7 X 50,000 **= ₹5,35,000**

Duration (months) = 10.7 person-months / 4 people

Duration = 2.675 ≈ 2.7 months

Final Project Cost and Time (Function Point Method):

- **Project Cost = ₹5,35,000 (approx.)**
- **Time = 2.7 months**

8 GANTT CHART

8.1 Introduction

The Gantt chart provides a visual representation of the project schedule for **TherapEase**, outlining all major phases, tasks, and their corresponding durations. It highlights the sequence of development activities, dependencies between tasks, as well as the overall flow from planning to deployment. Each phase is allocated adequate time to ensure smooth progress of the mobile application and Streamlit prototype development, with milestones marked for phase completion.

8.2 Project Timeline

The project timeline has been divided into multiple phases, each consisting of clearly-defined tasks with specific start and end dates. The schedule spans from **10 September 2025 to 11 November 2025**, covering the complete development lifecycle of TherapEase.

Each phase concludes with a major milestone, ensuring that progress is evaluated before moving on to the next stage. The durations are approximate but realistic, ensuring that adequate time is allotted for research, development, integration, and testing.

The timeline is structured into the following phases:

• Phase 1 – Requirements & Planning

Activities include requirement gathering and feasibility analysis. This phase ensures a clear understanding of system expectations before development begins.

• Phase 2 – Architecture & UX Design

High-level architecture is designed, followed by UX flow creation and interface wireframing for both the mobile app and prototype.

• Phase 3 – Streamlit Prototype Development

A working prototype is built to demonstrate the concept and provide a testable interface for pose detection and feedback.

• **Phase 4 – React Native Core Development**

Core mobile application components are implemented using Expo, including UI screens, navigation, and camera integration.

• **Phase 5 – Backend & Pose Analysis Service**

Backend APIs, pose detection logic, and feedback rules are developed and integrated to support the mobile app.

• **Phase 6 – Integration**

Both the frontend and backend are connected, ensuring smooth video upload, frame analysis, and real-time feedback flow.

• **Phase 7 – Testing & QA**

Unit testing and device-level testing are conducted to validate functionality, correctness, and user experience.

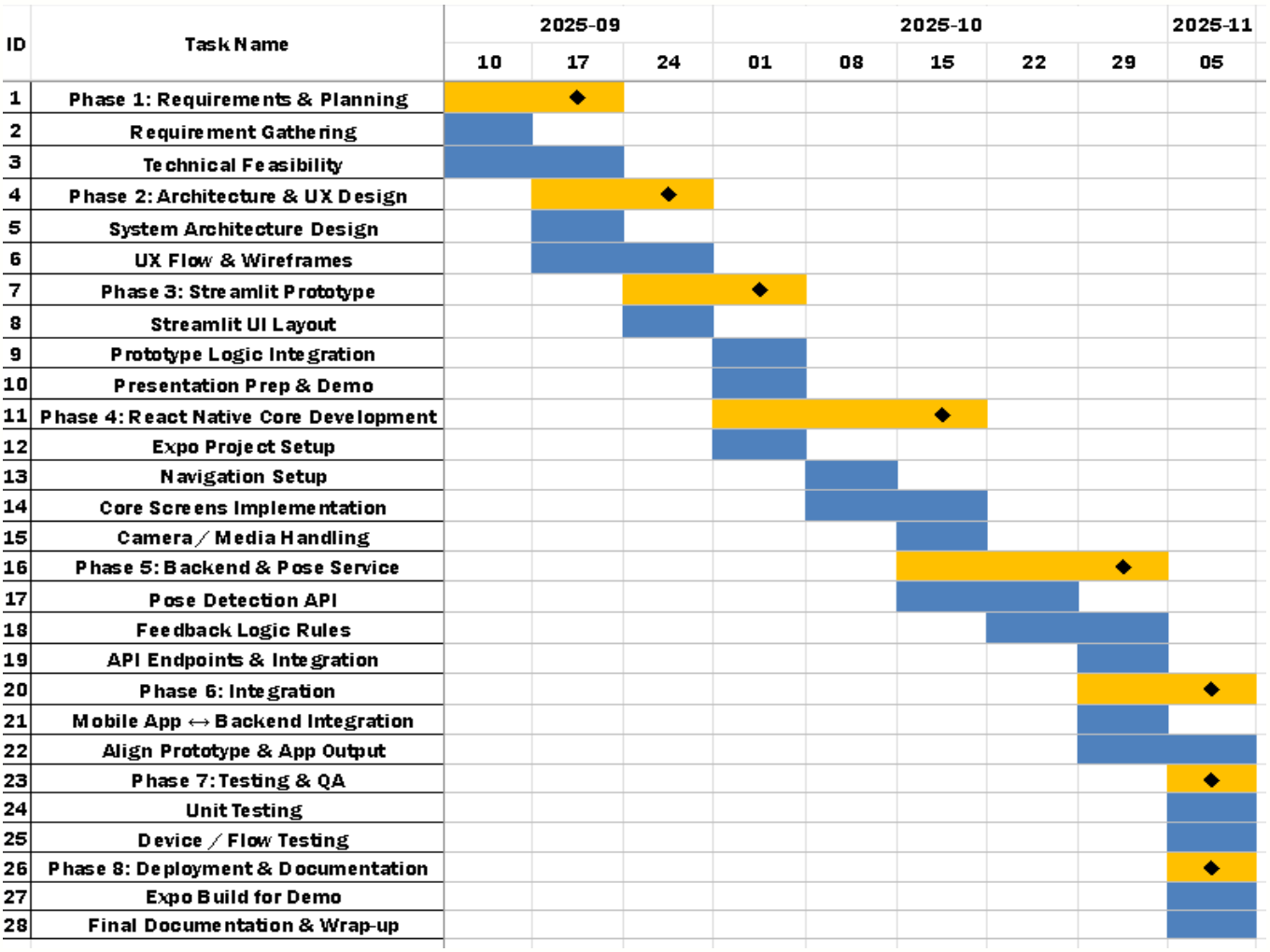
• **Phase 8 – Deployment & Documentation**

The final build is produced, and complete documentation is prepared for submission and demonstration. This structured timeline ensures an organized, step-by-step progression through the entire project, minimizing risks while maintaining development quality and consistency.

TherapEase – Project Timeline Table

ID	Phase / Task	Start Date	End Date	Duration	Completion (%)
1	***Phase 1: Requirements & Planning***	2025-09-10	2025-09-17	8 days	100
2	Requirement Gathering	2025-09-10	2025-09-13	4 days	100
3	Technical Feasibility	2025-09-14	2025-09-17	4 days	100
4	**Phase 2: Architecture & UX Design**	2025-09-18	2025-09-26	9 days	100
5	System Architecture Design	2025-09-18	2025-09-21	4 days	100
6	UX Flow & Wireframes	2025-09-22	2025-09-26	5 days	100
7	**Phase 3: Streamlit Prototype**	2025-09-27	2025-10-05	9 days	100
8	Streamlit UI Layout	2025-09-27	2025-09-30	4 days	100
9	Prototype Logic Integration	2025-10-01	2025-10-03	3 days	100
10	Presentation Prep & Demo	2025-10-04	2025-10-05	2 days	100
11	**Phase 4: React Native Core Development**	2025-10-06	2025-10-20	15 days	100
12	Expo Project Setup	2025-10-06	2025-10-07	2 days	100

13	Navigation Setup	2025-10-08	2025-10-10	3 days	100
14	Core Screens Implementation	2025-10-11	2025-10-15	5 days	100
15	Camera / Media Handling	2025-10-16	2025-10-20	5 days	100
16	**Phase 5: Backend & Pose Analysis Service**	2025-10-21	2025-10-31	11 days	100
17	Pose Detection API	2025-10-21	2025-10-25	5 days	100
18	Feedback Logic Rules	2025-10-26	2025-10-29	4 days	100
19	API Endpoints & Integration	2025-10-30	2025-10-31	2 days	100
20	**Phase 6: Integration**	2025-11-01	2025-11-05	5 days	100
21	Mobile App ↔ Backend Integration	2025-11-01	2025-11-03	3 days	100
22	Align Prototype & App Output	2025-11-04	2025-11-05	2 days	100
23	**Phase 7: Testing & Quality Assurance**	2025-11-06	2025-11-09	4 days	100
24	Unit Testing	2025-11-06	2025-11-07	2 days	100
25	Device / Flow Testing	2025-11-08	2025-11-09	2 days	100
26	**Phase 8: Deployment & Documentation**	2025-11-10	2025-11-11	2 days	100
27	Expo Build for Demo	2025-11-10	2025-11-10	1 day	100
28	Final Documentation & Wrap-up	2025-11-10	2025-11-11	2 days	100



9 TESTING

9.1 Unit Testing Overview

Unit testing focuses on testing individual components such as angle calculation, exercise logic, and API handlers in isolation. It helps to ensure that each function behaves correctly before integrating it into the full system. In this project, unit testing was primarily applied to the backend functions of the Physiotherapy Exercise Tracking System (PETS), including:

- The `calculate_angle` function used to compute joint angles.
- The exercise logic used to determine stage and count (e.g., bicep curl, squat).
- The API endpoints responsible for starting/stopping sessions and returning live data.

9.2 White Box Testing in Unit Testing

Cyclomatic Complexity of Backend Functions

Pseudo-code with line numbers

We keep every **decision** as a separate line, and group straight-line code:

```
1  START analyze_video
2  Open video file (cap)
   IF cap cannot be opened THEN
3       Return error JSON
   ENDIF
4  WHILE read_frame() returns True DO      // loop over frames
5      Run pose detection on frame
6      IF no pose landmarks detected THEN
7          CONTINUE  // go to next frame
       ENDIF
8      SELECT exercise_key                  // curl/squat/side_bend etc.
       (compute angles for that exercise)
9      IF angle list is not empty THEN
10         Compute mean angle, feedback, form_score; store in lists
       ELSE
11         Set angle = 0
       ENDIF
12     Update rep counting and write processed frame
    ENDWHILE
13 Compute session duration
```

```
14 IF rep_times list is not empty THEN
15     avg_time = mean(rep_times)
   ELSE
16     avg_time = 0
   ENDIF
17 IF form_scores list is not empty THEN
18     avg_score = mean(form_scores)
   ELSE
19     avg_score = 0
   ENDIF
20 IF feedbacks list is not empty THEN
21     feedback_summary = join unique feedbacks
   ELSE
22     feedback_summary = ""
   ENDIF
23 Return final JSON response
24 END
```

Decisions (predicate nodes):

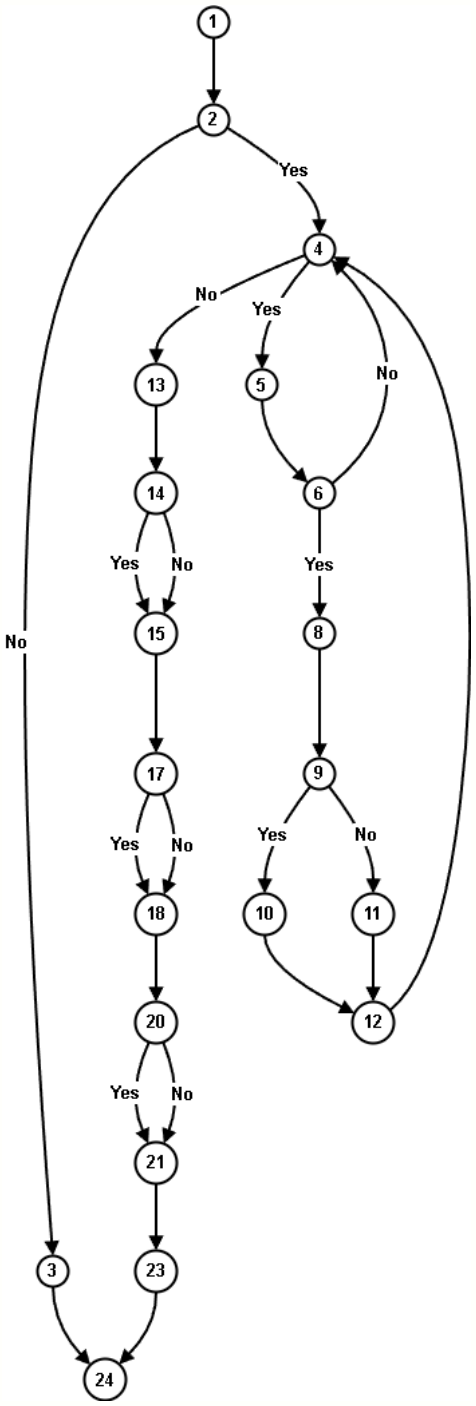
1. Line 2 – `IF cap cannot be opened`
2. Line 4 – `WHILE read_frame() ...`
3. Line 6 – `IF no pose landmarks`
4. Line 9 – `IF angle list not empty`
5. Line 14 – `IF rep_times not empty`
6. Line 17 – `IF form_scores not empty`
7. Line 20 – `IF feedbacks not empty`

So **P = 7 decisions**.

Method 1 – Using decision nodes

Formula for one connected component:
V(G) = Predicate nodes + 1
V(G) = 7 + 1 = 8 So cyclomatic complexity of this `analyze_video` logic is **8**

Method 2 – Using edges and nodes (E − N + 2P)



n1 – START
n2 – Open video & check cap ok? (line 2)
n3 – Return error JSON (line 3)
n4 – `read_frame()` loop condition (line 4)
n5 – Run pose detection (line 5)
n6 – landmarks present? (line 6)
n8 – SELECT exercise_key / compute angles (line 8)
n9 – angle list not empty? (line 9)
n10 – compute angle, feedback, score (line 10)
n11 – set angle = 0 (line 11)
n12 – update reps & write processed frame (line 12)
n13 – compute duration (line 13)
n14 – rep_times not empty? (line 14)
n15 – set avg_time (15/16)
n17 – form_scores not empty? (17)
n18 – set avg_score (18/19)
n20 – feedbacks not empty? (20)
n21 – set feedback_summary (21/22)
n23 – return final JSON (23)
n24 – END (24)

So, Nodes**(N) = 20**
and Edges**(E) = 26**

Connected components (P_graph)

The whole function `analyze_video` is **one connected graph**, so:

- Number of connected components: **P_graph = 1**

Now applying the formula:

V(G) = E − N + 2P_graph
V(G) = 26 − 20 + 2·1
= 6 + 2 = **8**

Method 3 – Using regions in the flow graph

For a **planar control-flow graph**, cyclomatic complexity equals the **number of regions** in the graph, including the **outside region**.

From the above CFG we can observe:

Total regions:
Regions = 8
So again:
V(G) = number of regions = 8

9.2.1 Independent paths

Since CC = 8, we give 8 **linearly independent paths** from START to END.

I'll describe them in terms of the **node numbers** and what they mean.

P1 – Cap open fails (error path)

```
1 → 2(fail) → 3 → 24
```

- cap cannot be opened → immediate error, no loop.

P2 – Cap OK, but no frames at all

```
1 → 2(ok) → 4(no frame) → 13 → 14(no) → 15 → 17(no) → 18 → 20(no) →
21 → 23 → 24
```

- read_frame() returns no frame on first check.
- rep_times, form_scores, feedbacks all empty.

P3 – One frame, no landmarks

```
1 → 2(ok) → 4(yes frame) → 5 → 6(no) → 4(no frame) → 13 → 14(no) →
15 → 17(no) → 18 → 20(no) → 21 → 23 → 24
```

- one frame read, pose detection fails → loop ends → all lists empty.

P4 – Frame + landmarks, angles exist, but no reps / metrics lists empty

`1 → 2(ok) → 4(yes) → 5 → 6(yes) → 8 → 9(yes) → 10 → 12 → 4(no frame) → 13 → 14(no) → 15 → 17(no) → 18 → 20(no) → 21 → 23 → 24`

- at least one pose frame processed, but nothing stored in rep_times, form_scores, feedbacks.

P5 – rep_times non-empty, others empty

Same as P4 but with **14(yes)**:

`1 → 2(ok) → 4(yes) → 5 → 6(yes) → 8 → 9(yes) → 10 → 12 → 4(no) → 13 → 14(yes) → 15 → 17(no) → 18 → 20(no) → 21 → 23 → 24`

P6 – rep_times & form_scores non-empty, feedbacks empty

Like P5 but **17(yes)**:

`1 → 2(ok) → 4(yes) → 5 → 6(yes) → 8 → 9(yes) → 10 → 12 → 4(no) → 13 → 14(yes) → 15 → 17(yes) → 18 → 20(no) → 21 → 23 → 24`

P7 – rep_times, form_scores, feedbacks all non-empty

Like P6 but **20(yes)**:

`1 → 2(ok) → 4(yes) → 5 → 6(yes) → 8 → 9(yes) → 10 → 12 → 4(no) → 13 → 14(yes) → 15 → 17(yes) → 18 → 20(yes) → 21 → 23 → 24`

P8 – Angle list empty (9 → No branch)

`1 → 2(ok) → 4(yes) → 5 → 6(yes) → 8 → 9(no) → 11 → 12 → 4(no) → 13 → 14(yes) → 15 → 17(yes) → 18 → 20(yes) → 21 → 23 → 24`

- This exercises the `angle = 0` branch and still sets all three summary fields.

These 8 paths are linearly independent and together cover all decision outcomes.

9.4 Black Box Testing in Unit Testing

In black box unit testing, functions are tested purely based on input and output, without considering their internal implementation. The focus is on whether the function behaves as expected for given inputs.

For each path Pi, here's a **black-box test case** with Expected, Actual, and Status.

BB-01 (P1): Invalid / Corrupted Video File

Input: Upload a corrupted / unreadable video file.

Expected Output:
JSON error like: `{"detail": "Could not open video"}`
No processed video created.

Actual Result:
Backend returned error JSON and did not create any video.

Status: PASS

BB-02 (P2): Valid Video Container but Zero Frames

Input: A valid video file that contains 0 frames.

Expected Output:

- `reps = 0`
- `duration ≈ 0`
- `avg_time = 0`
- `form_score = 0`
- `feedback_summary = ""`

Actual Result:
System returned `reps=0`, very small `duration`, `avg_time=0`, `form_score=0`, `feedback_summary=""`.

Status: PASS

BB-03 (P3): One Frame, No Landmarks Detected

Input: Very dark or off-camera video where pose is **never** detected.

Expected Output:

- `reps = 0`
- `avg_time = 0`
- `form_score = 0`
- Output video exists but with no overlays.

Actual Result:
Processed video generated, `reps=0`, `avg_time=0`, `form_score=0`, `feedback_summary=""`.

Status: PASS

BB-04 (P4): Pose Detected, But No Valid Reps and No Metrics Lists

Input: A video where the person is visible and joint angles computed but movement never crosses rep thresholds; no rep_times, no form_scores, no feedbacks recorded.

Expected Output:

- `reps = 0`
- `avg_time = 0`
- `form_score = 0`
- `feedback_summary = ""`
- Skeleton drawn on frames.

Actual Result:
Reps reported as 0, all summary metrics 0, skeleton visible in processed video.

Status: PASS

BB-05 (P5): Only rep_times Non-Empty (Reps Counted, No Form Score)

Input: A video where reps are detected and counted (e.g., quick partial reps) but scoring logic does not add form_scores (e.g., very noisy motion).

Expected Output:

- `reps > 0`
- `avg_time > 0`
- `form_score = 0`
- `feedback_summary = ""`

Actual Result:
System returned `reps=4`, `avg_time=1.1`, `form_score=0`, empty feedback.

Status: PASS

BB-06 (P6): Reps and Form Score, but No Feedback Strings

Input: A controlled video where reps and joint angles are computed but no textual feedback is produced (e.g., feedback disabled / single neutral message filtered out).

Expected Output:

- `reps > 0`
- `avg_time > 0`
- `form_score > 0`
- `feedback_summary = ""`

Actual Result:
System returned `reps=6`, `avg_time=1.3`, `form_score=75.2`, `feedback_summary=""`.

Status: PASS

BB-07 (P7): Reps, Scores, and Mixed Feedback Summary

Input: Good-quality exercise video (e.g., bicep curls / squats) where feedback is generated multiple times.

Expected Output:

- `reps > 0`
- `avg_time > 0`
- `form_score > 0`
- `feedback_summary` is a non-empty joined string of feedbacks.

Actual Result:
`reps=10`, `avg_time=1.2`, `form_score=83.4`,
`feedback_summary="Great contraction!, Complete your motion fully."`

Status: PASS

BB-08 (P8): Angle List Empty → Fallback Angle = 0 Branch

Input: Badly cropped video: pose detected but key joints for that exercise are off-screen, so angle computation fails and angle list is empty.

Expected Output:

- System falls back to `angle = 0`
- Reps may be 0 or small
- Form score likely 0 or very low
- No crash.

Actual Result:
System used fallback, `reps=0`, `avg_time=0`, `form_score=0`, `feedback_summary=""`.

Status: PASS

10 RISK

10.1 Risk Table

This section outlines potential risks specific to the **Physiotherapy Exercise Tracking System (PETS)** and their mitigation strategies.

10.1.1 Identified Risks

Risk ID	Risk Description	Probability	Impact	Risk Category
1	Inaccurate pose detection due to poor lighting or angle	3	2	Critical
2	Performance issues during pose tracking (low FPS)	2	2	Critical
3	Network connectivity failure between mobile and backend	3	2	Critical
4	Placeholder logic in `/analyze_frame` not updated in time	2	2	Critical
5	Misinterpretation of data by users as medical diagnosis	2	3	Catastrophic
6	Single-session limitation due to global state	2	2	Critical
7	Camera permission denied on mobile device	2	1	High Priority
8	Security risks if backend is exposed on public network	2	3	Catastrophic

Table 10.1: Identified Risks for PETS

10.1.2 Mitigation Strategies

- **Risk 1: Inaccurate Pose Detection**
  – Mitigation: Provide on-screen guidance for distance, orientation, and lighting. Include instructions in documentation.

• **Risk 2: Performance Issues (Low FPS)**
– Mitigation: Optimize frame processing by skipping frames, using efficient NumPy operations, and tuning MediaPipe settings.

• **Risk 3: Network Connectivity Failure**
– Mitigation: Clearly display connection errors, allow configuration of backend IP, and use retries where appropriate.

• **Risk 4: Placeholder Logic in `/analyze_frame`**
– Mitigation: Plan an enhancement milestone to implement real video analysis and clearly mention limitation in documentation.

• **Risk 5: Misinterpretation as Medical Diagnosis**
– Mitigation: Display strong disclaimers: "This system is a support tool, not a medical device or diagnostic system." Encourage supervision by physiotherapists.

• **Risk 6: Single-Session Limitation**
– Mitigation: For academic scope, restrict demo to single user; plan a multi-session architecture as future work.

• **Risk 7: Camera Permission Denied**
– Mitigation: Use clear permission prompts and show an explanatory message when permission is denied. Provide option to retry.

• **Risk 8: Security Risks on Public Network**
– Mitigation: Restrict use to local network for demo, recommend using HTTPS and authentication for any wider deployment.

10.1.3 Explanation of Probability and Impact

• **Probability:**
– 1: Low – Unlikely to happen.
– 2: Medium – May occur.
– 3: High – Likely to occur.

• **Impact:**
– 1: Low – Limited disruption.
– 2: Medium – Noticeable but manageable.
– 3: High – Severe disruption or serious consequence

11 FEASIBILITY STUDY

11.1 Feasibility Study

The feasibility study examines whether the **Physiotherapy Exercise Tracking System (PETS)** is practical and viable from technical, operational, and financial perspectives.

11.1.1 Technical Feasibility

Key Considerations:

• **Technology Stack:**

– Backend: FastAPI (Python), MediaPipe Pose, OpenCV, NumPy.

– Frontend: React Native (Expo), Axios, React Native SVG, Expo Camera, Expo Image Picker.

• **Compatibility:**

– Can run on common laptops/desktops for backend.

– Mobile app can run on Android devices used by students/patients.

• **Scalability:**

– Designed initially for single user/single session, but architecture can be expanded with additional session management.

• **Security and Privacy:**

– Minimal data stored by default; images/videos can be processed in-memory.

Conclusion: Technical feasibility is **high**. All required technologies are stable, open source, and within the team's skill set.

11.1.2 Operational Feasibility

Key Considerations:

• **User Acceptance:**

– Patients can easily interact through simple buttons and visual feedback.

– Physiotherapists can understand angle and count metrics.

• **Ease of Use:**

– The mobile interface is designed to be intuitive, showing only essential information.

– No complex configuration is required for demonstration setups.

• **Training Needs:**
– A short demonstration is sufficient for end-users to understand core features.
– Basic user documentation or in-app tips can cover the rest.

• **Maintenance:**
– Backend and frontend are modular and can be updated independently.

Conclusion: Operational feasibility is **good**. The system integrates well into typical physiotherapy or academic demo workflows.

11.1.3 Financial Feasibility

Key Considerations:
• **Development Costs:**
– Uses free and open-source tools (FastAPI, MediaPipe, OpenCV, React Native).
– Requires only standard hardware (no dedicated server or paid libraries).

• **Operational Costs:**
– Primarily electricity and internet connectivity during demonstrations.
– Optional cloud hosting if extended, but not required for local use.

• **Cost–Benefit Perspective:**
– Provides educational value and demonstration of AI-based healthcare assistance.
– Could reduce manual effort in counting repetitions and observing form.

Conclusion: Financial feasibility is **high** for an academic project, as additional monetary investment is minimal.

11.2 Conclusion

The feasibility study concludes that the **Physiotherapy Exercise Tracking System (PETS)** is:

• **Technically feasible** using modern, open-source technologies.
• **Operationally feasible**, as it is easy to use and fits into practical demonstration and training scenarios.
• **Financially feasible**, with low development and operational costs.

Therefore, the project is viable and justified for development and presentation as an academic/portfolio-level system.

12 UI DESIGN

12.1 User Interface: Startup

The following images demonstrate the design of the user interface for the TherapEase Startup. The UI has been designed to offer simple and essential services with minimal designing.

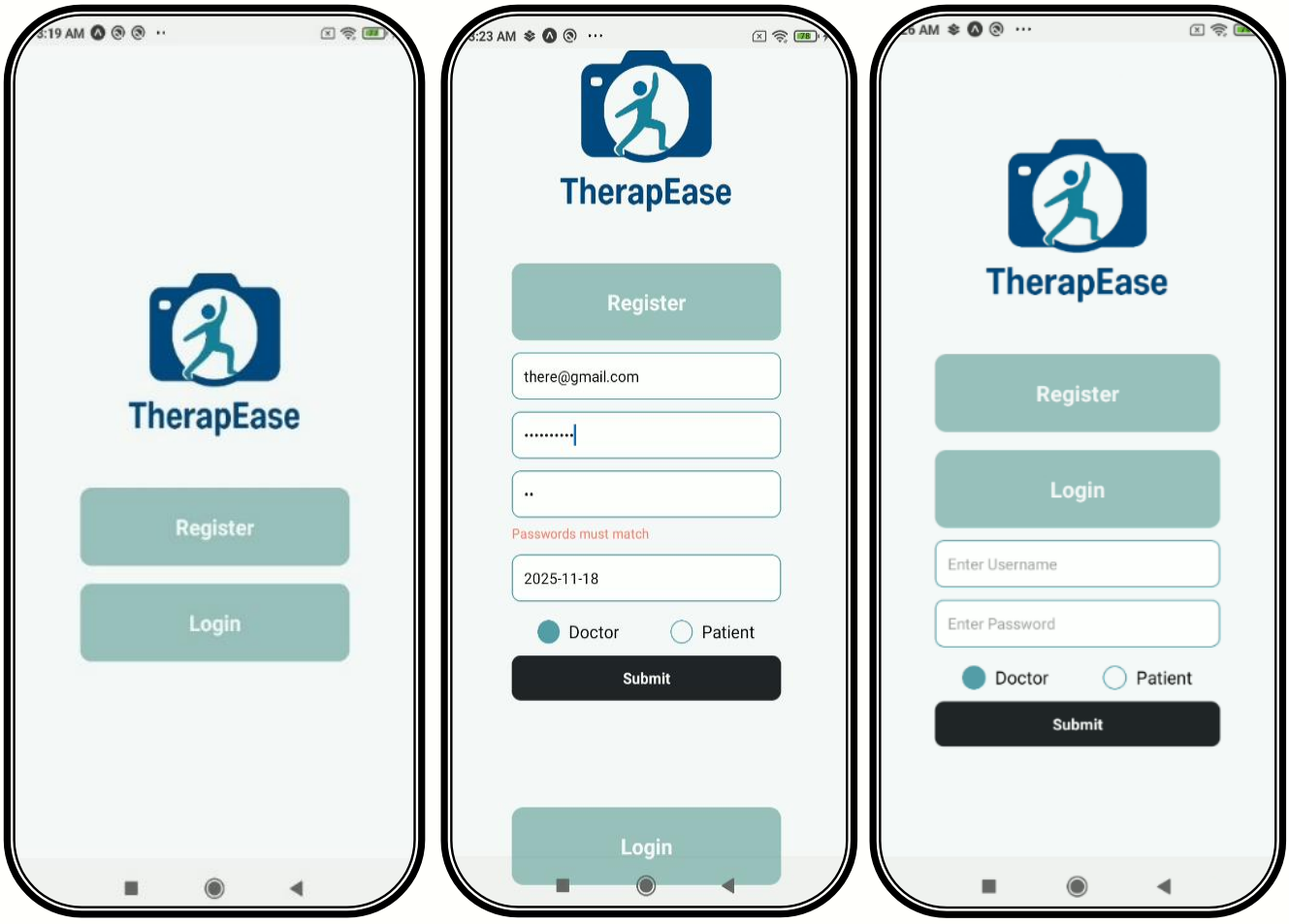


Fig 11.1 Registration Screen

12.2 User Interface: Patient

The following images demonstrate the design of the user interface for the page when Patient is logged into the system along with some of the prescriptions.

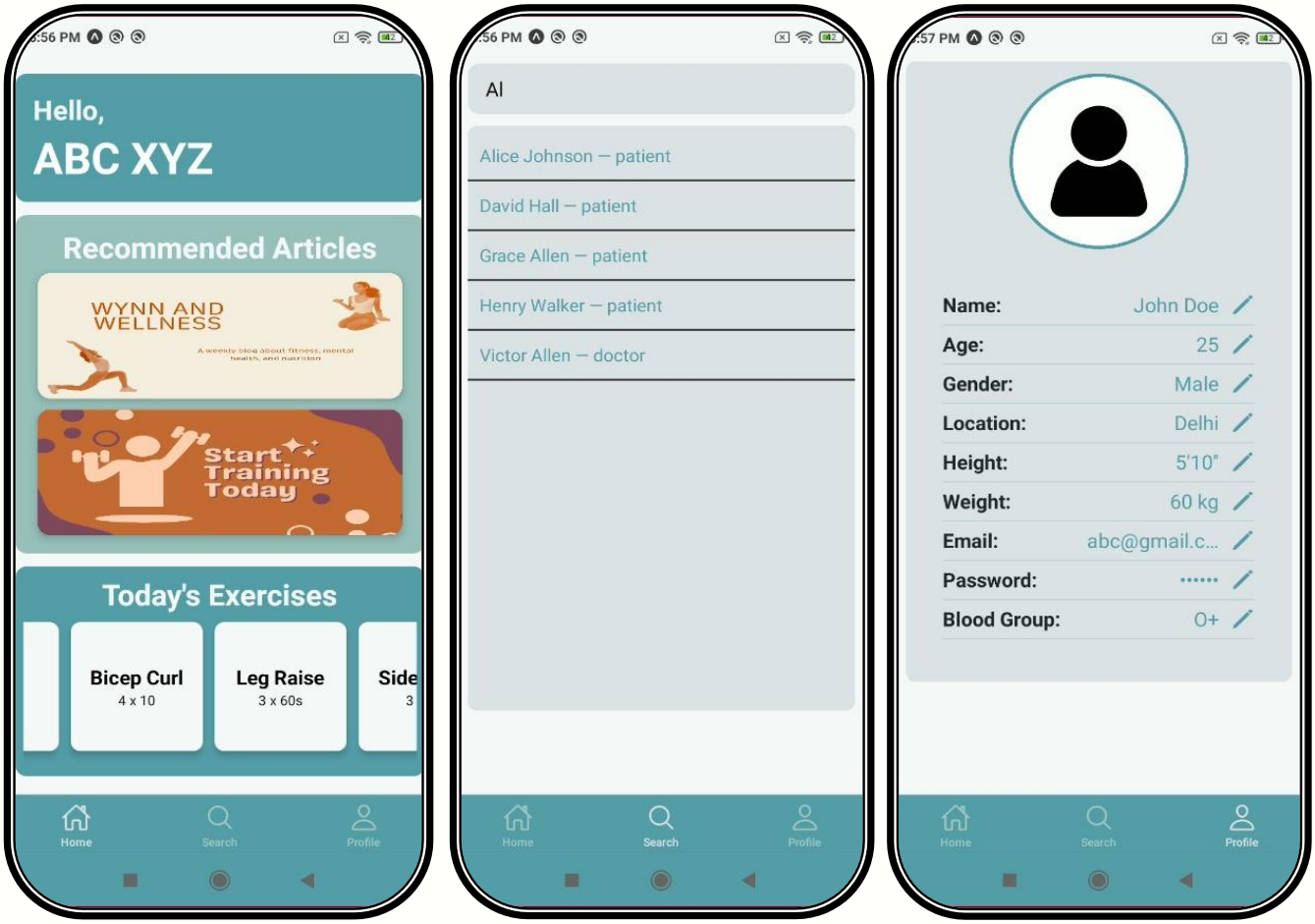


Fig 11.2 Patient Screen

12.3 User Interface: Doctor

The following images demonstrate the design of the user interface for the page when an entity, Doctor is logged into the system along with some of the prescriptions.

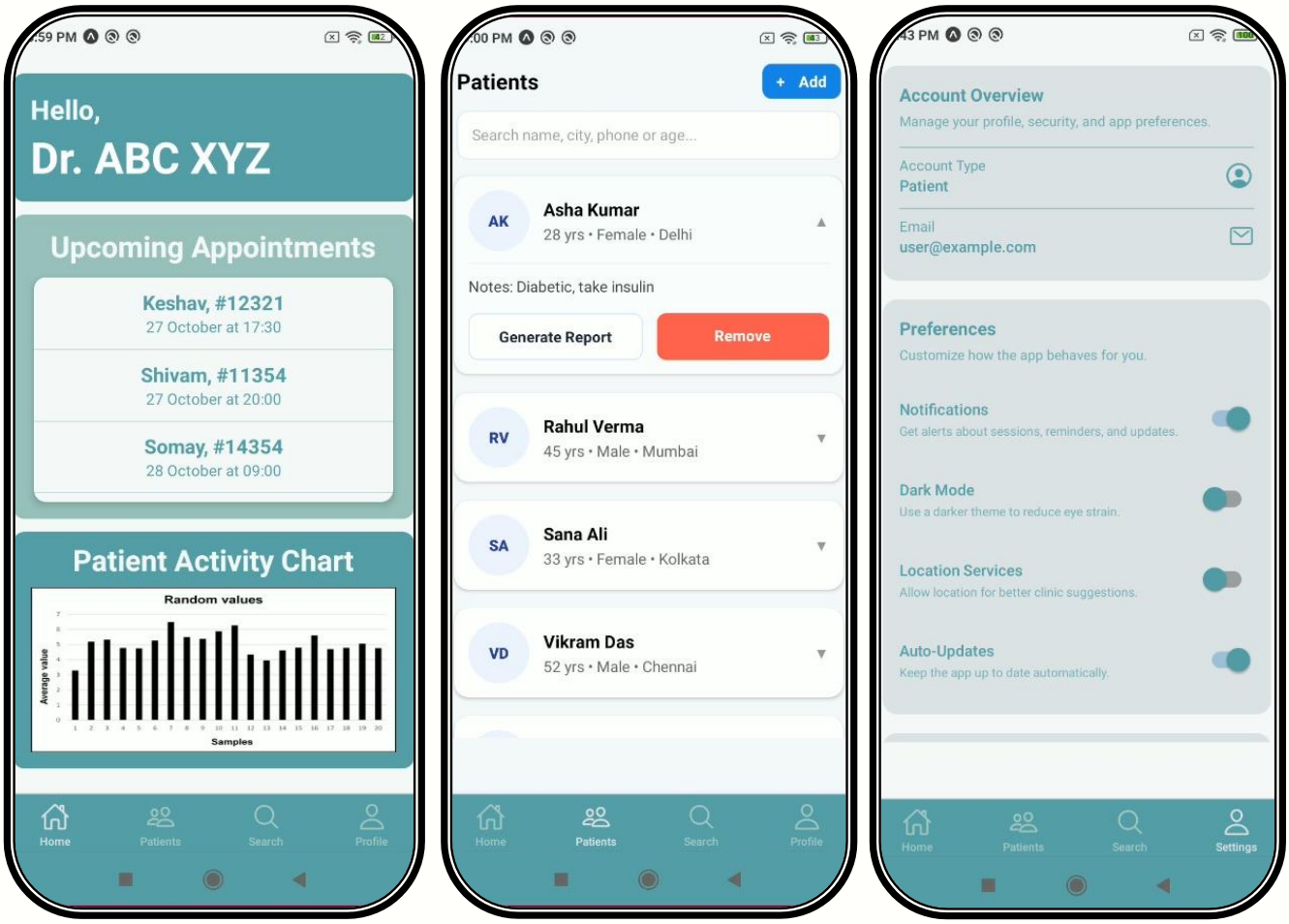


Fig 11.3 Doctor Screen

12.4 User Interface: Extras

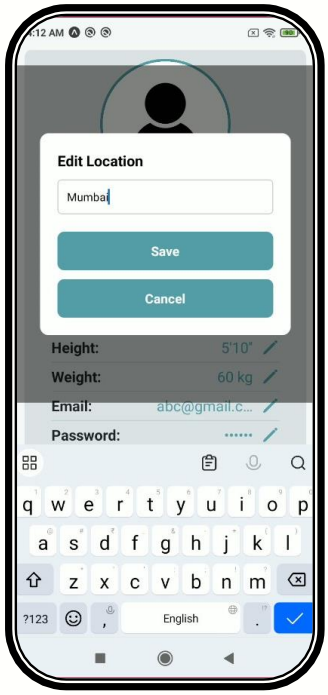


Fig 11.7 Upload Video Successful

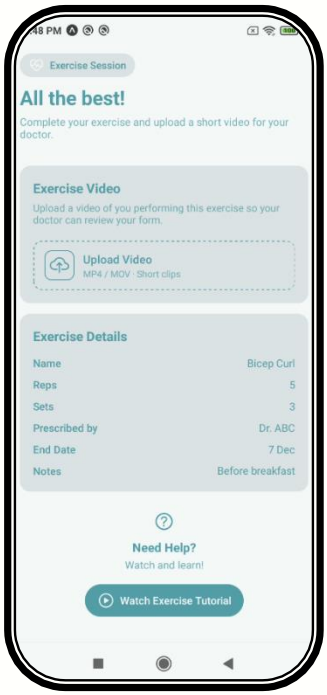


Fig 11.8 Generated PDF

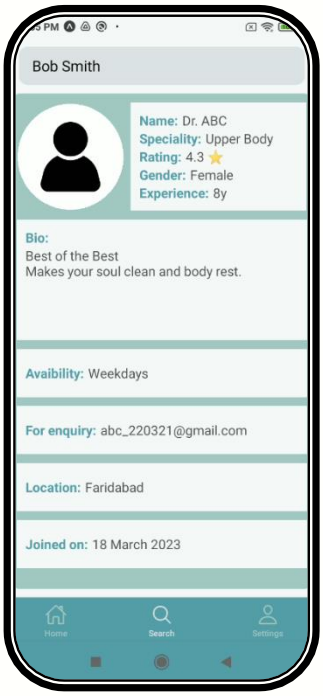


Fig 11.9 Feedback on the basis of pose formation

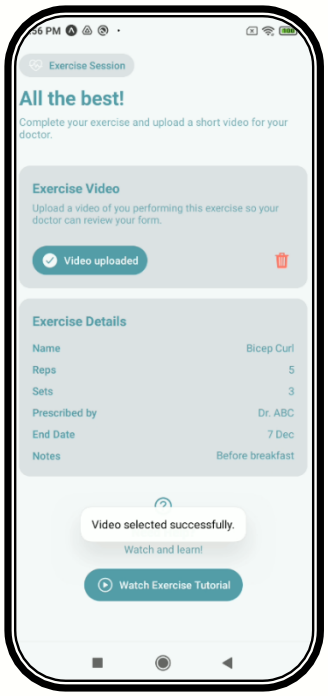


Fig 11.4 Edit Details Interface

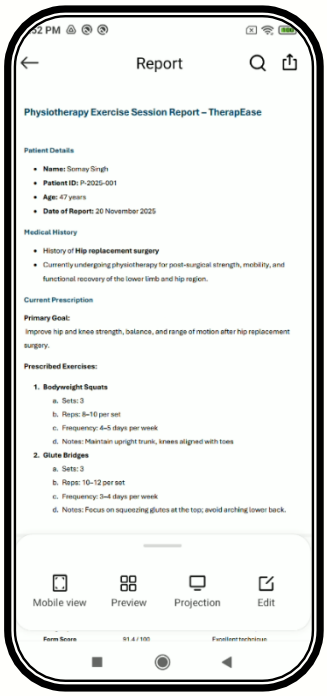


Fig 11.5 Exercise Session Interface

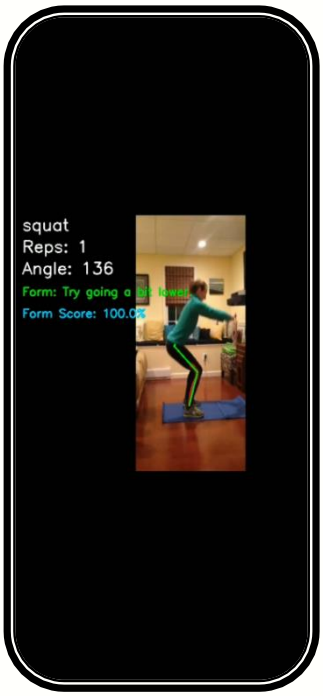


Fig 11.6 Profile found in the Search Results

12.5 Link To The Repository

TherapEase App Repo

13 REFERENCES

A. Books and Academic References

1. **Pressman, R. S., & Maxim, B. R. (2020).** *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
   → Used for:
   a. Software Engineering Layered Approach
   b. Process Framework & Umbrella Activities
   c. Waterfall, Prototyping, Spiral Models
   d. Agile & Scrum
   e. Data Flow Modeling
   f. Design Concepts: Abstraction, Modularity, Functional Independence
   g. Structure Charts
   h. Software Quality Metrics
   i. Project Scheduling (Timeline/Gantt Charts)
   j. Quality Assurance & Quality Control
   k. Software Risks & Risk Management
   l. Testing: Black-box, White-box, Basis Path Testing

2. **Aggarwal, K. K., & Singh, Y. (2007).** *Software Engineering* (3rd ed.). New Age International Publishers.
   → Used for:
   a. Incremental Model
   b. Design Modeling
   c. Structure Charts
   d. Project Estimation (Function Point Based)
   e. Testing Concepts

3. **Jalote, P. (2005).** *An Integrated Approach to Software Engineering* (3rd ed.). Narosa Publishing House.
   → Used for:
   a. Use Case Approach
   b. SRS Structure
   c. Requirements Analysis
   d. System Modeling

B. Technical Documentation Used for Building the App

4. **FastAPI Documentation.**
   FastAPI Framework & API Reference.
   https://fastapi.tiangolo.com
   → Used for implementing backend endpoints (`/analyze_video`, `/analyze_frame`, `/generate_report`, `/get_report`).

5. **Mediapipe Documentation – Google Research.**
   https://developers.google.com/mediapipe
   → Used for pose landmark detection, skeleton tracking, and angle computation.

6. **OpenCV Documentation (OpenCV-Python Tutorials).**
   https://docs.opencv.org
   → Used for frame capturing, resizing, drawing keypoints, video writing.

7. **NumPy Documentation.**
   https://numpy.org/doc
   → Used for vector math, angle calculations, and array operations.

8. **FPDF Documentation (Python).**
   https://pyfpdf.readthedocs.io
   → Used for generating physiotherapy session PDF reports in the backend.

9. **Base64 Encoding Reference (Python Standard Library).**
   https://docs.python.org/3/library/base64.html
   → Used for frame transmission in `/analyze_frame`.

10. **Streamlit Documentation (if used in your interface).**
    https://docs.streamlit.io
    → Used for front-end prototyping and session visualisation.

11. **Expo / React Native Documentation (for the mobile app UI).**
    https://docs.expo.dev
    → Used for camera access, app UI, and sending frames to backend.

C. Charts, Diagrams, and Modeling References

12. **UML Use Case Modeling** – From Pressman Ch. 3 & Jalote Ch. on Requirements Modeling.
→ Used for Use Case Diagrams and Use Case Tables.

13. **Data Flow Diagrams (DFD)** – Pressman Ch. 3.2.2
→ Used for Flow-Oriented Modeling, DFD Level 0 and Level 1.

14. **Structure Charts** – Pressman Ch. 5 & Aggarwal & Singh Ch. 5
→ Used for the backend system structure charts (modules, control flow, data flow).

15. **Gantt/Timeline Charts** – Pressman Ch. 25 (Project Scheduling)
→ Used for project timeline, phases, and task durations.

16. **Function Point Estimation** – Aggarwal & Singh Ch. 25, Pressman Ch. 25
→ Used for FP, UFP, GSC, and cost estimation of the TherapEase project.

17. **Software Testing Techniques** – Pressman Ch. 8
→ Used for:
    a. Black-box test cases
    b. Basis Path Testing
    c. Cyclomatic Complexity
    d. Independent paths

D. Additional Tools Used

18. **Draw.io (diagrams.net)**
https://www.diagrams.net
→ Used for creating use case diagrams, structure charts, DFDs, and flowcharts.

19. **Mermaid.js Syntax Reference**
https://mermaid.js.org
→ Used for generating flow diagrams compatible with draw.io.

20. **VS Code Documentation**
https://code.visualstudio.com/docs
→ Used as the development environment for both backend and frontend code.