

Y e a r
S e m e s t e r
S e s s i o n

3 r d
5 t h
2 0 2 5

TherapEase

Physiotherapy made easy



TherapEase

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	6
• 1.1 Overview	6
• 1.2 Challenges in the Current System	6
• 1.3 Proposed Solution	7
• 1.4 Conclusion	7
2 Process Model : Agile	9
• 2.1 Phases of the Agile Process	9
◦ 2.1.1 Requirement Gathering & Analysis	9
◦ 2.1.2 Planning	9
◦ 2.1.3 Design	9
◦ 2.1.4 Development	10
◦ 2.1.5 Testing	10
◦ 2.1.6 Deployment	10
3 Sequence and Data Flow	11
• 3.1 Data Flow Diagram Level 0	11
• 3.2 External Entities	11
• 3.3 Major Processes	11
• 3.4 Data Stores	12
• 3.5 Interactions	12
• 3.6 Data Flow Diagram Level 1	12
• 3.7 Detailed Processes	14
• 3.8 Structured Chart	15
• 3.9 Sequence Diagram	16

4 Data Dictionary	18
• 4.1 Entity: Patients	18
• 4.2 Entity: Exercises	19
• 4.3 Entity: Doctors	19
• 4.4 Entity: Appointments	20
• 4.5 Entity: Reports	20
5 Use Case Diagram	21
• 5.1 TherapEase System	21
◦ 5.1.1 Description	21
◦ 5.1.2 Main Features and Functionalities	22
◦ 5.1.3 Actors and their Roles	22
• 5.2 Use Case 1: Book Appointment	23
• 5.3 Use Case 2: performExerciseSession	24
• 5.4 Use Case 3: viewProgressReports	25
6 SRS	27
• 6.1 Introduction	27
◦ 6.1.1 Purpose	27
◦ 6.1.2 Scope	27
◦ 6.1.3 Definitions, Acronyms and Abbreviations	28
◦ 6.1.4 References	28
◦ 6.1.5 Overview	29
• 6.2 Overall Description	29
◦ 6.2.1 Product Perspective	29
◦ 6.2.2 Product Function	31
◦ 6.2.3 User Characteristics	31
◦ 6.2.4 Constraints	31
◦ 6.2.5 Assumption and Dependencies	32
• 6.3 Specific Requirements	32
◦ 6.3.1 Functional Requirements	32
◦ 6.3.2 External Interface Requirements	32
◦ 6.3.3 Performance Requirements	33
◦ 6.3.4 Design Constraints	33
◦ 6.3.5 Software System Attributes	34
• 6.4 Glossary	34

7 Function Point Analysis	35
• 7.1 Identify and Classify function	35
• 7.2 Complexity / Weight Assumptions	36
• 7.3 Count and Weight each Function	36
• 7.4 Unadjusted Function Points	37
• 7.5 Value Adjustment Factor	37
• 7.6 Result	38
8 Gantt Chart	41
• 8.1 Introduction	41
• 8.2 Project Timeline	41
9 Testing	42
• 9.1 Unit Testing Overview	42
• 9.2 Black Box Testing in Unit Testing	42
• 9.3 White Box Testing in Unit Testing	46
◦ 9.3.1 Conclusion	49
10 Risks	50
• 10.1 Risk Table	50
◦ 10.1.1 Identified Risks	50
◦ 10.1.2 Mitigation Strategies	50
◦ 10.1.3 Explanation of Probability and Impact	51
11 Feasibility Study	52
• 11.1 Introduction	52
◦ 11.1.1 Technical Feasibility	52
◦ 11.1.2 Operational Feasibility	52
◦ 11.1.3 Financial Feasibility	53
• 11.2 Conclusion	53
11 UI Designs	54
• 12.1 Startup	54
• 12.2 Patient	55
• 12.3 Doctor	56
• 12.4 Extras	57
• 12.5 Link to the Repository	58

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

- 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.
- Limited Accessibility:** A patient who lives at a far distance from hospitals, faces difficulty in attending regular sessions, which causes interruptions in their therapy.
- 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.
- 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.
- Inadequate Reporting:** There is no systematic way to generate detailed reports on patient adherence, progress, or exercises' pose correctness.
- 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 sprint 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

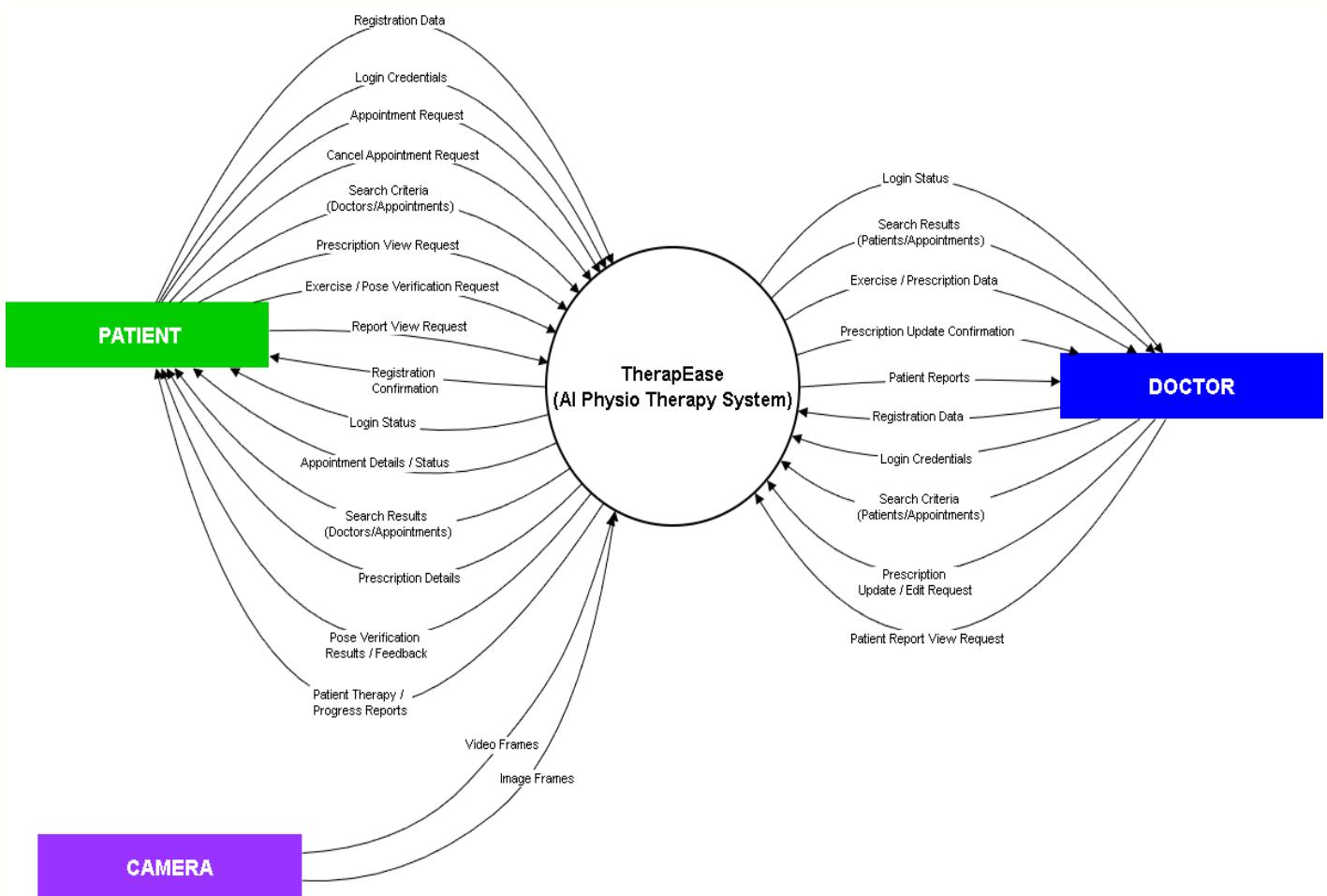
- ✓ Once testing is complete, the system is deployed incrementally, with new features being rolled out after each sprint.
- ✓ A feedback loop to address any bugs or errors.

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

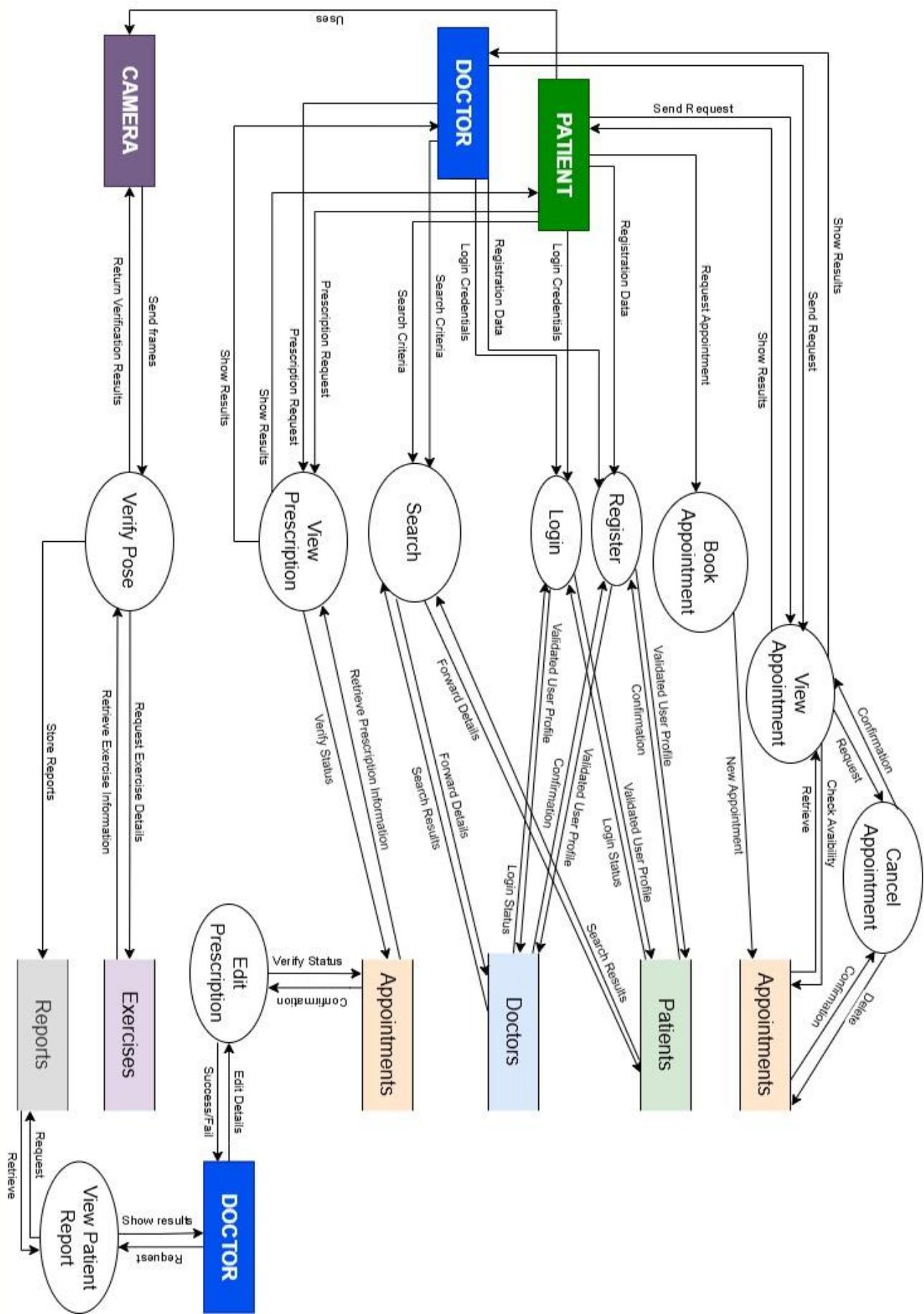
- **Patient:** Contains patient profiles, physical details, and login credentials.
- **Doctor:** Stores doctor profiles, qualifications, availability, and credentials.
- **Exercises:** Holds exercise names, types, and pose-related information.
- **Appointments:** Keeps track of scheduled dates, times, 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.



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

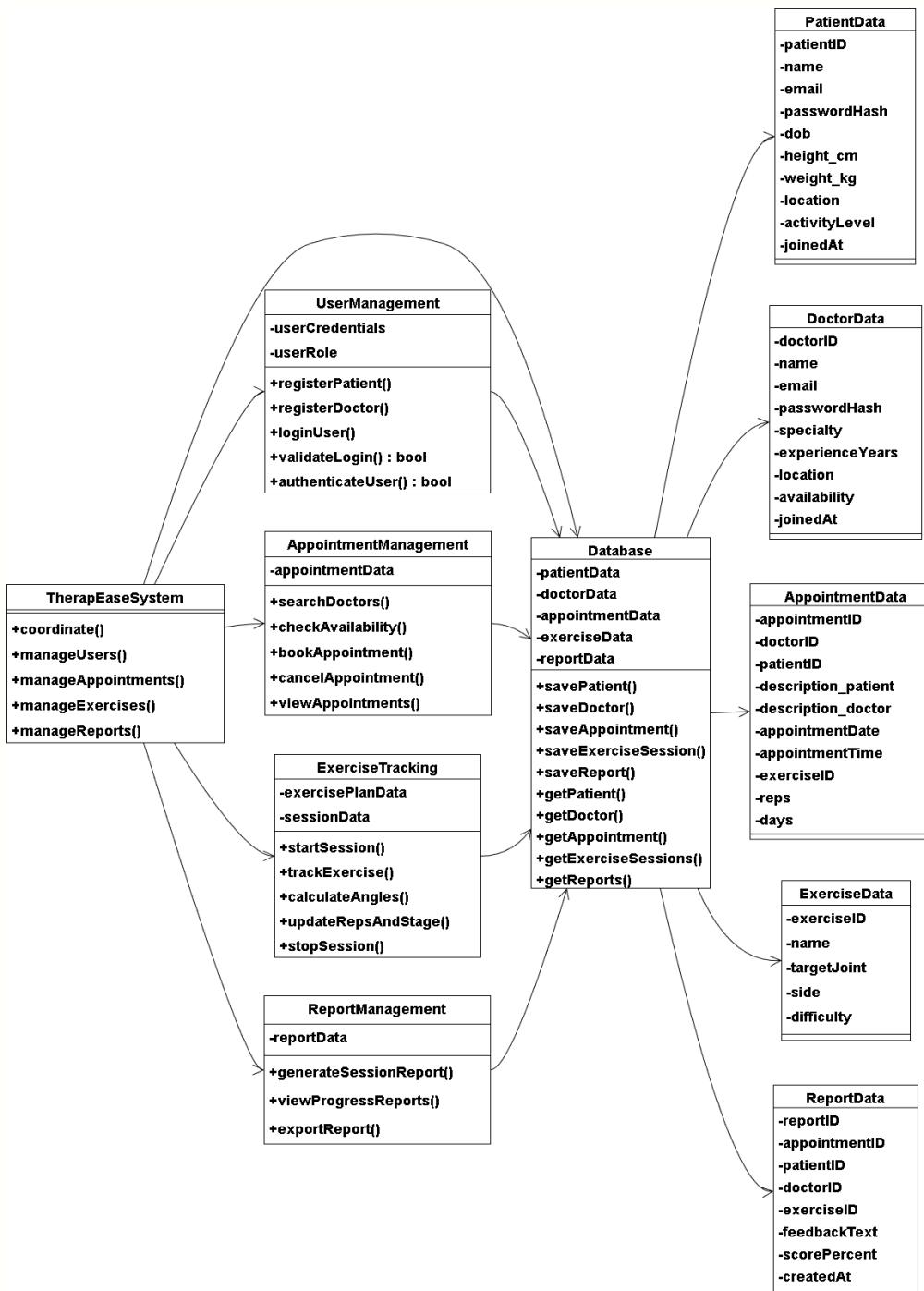
3. Report Generation Subprocesses:

- a. Compile Exercise Logs
- b. Calculate Pose Correctness and Performance Scores
- c. Send Reports to the allotted doctor

4. Prescription Update Subprocesses:

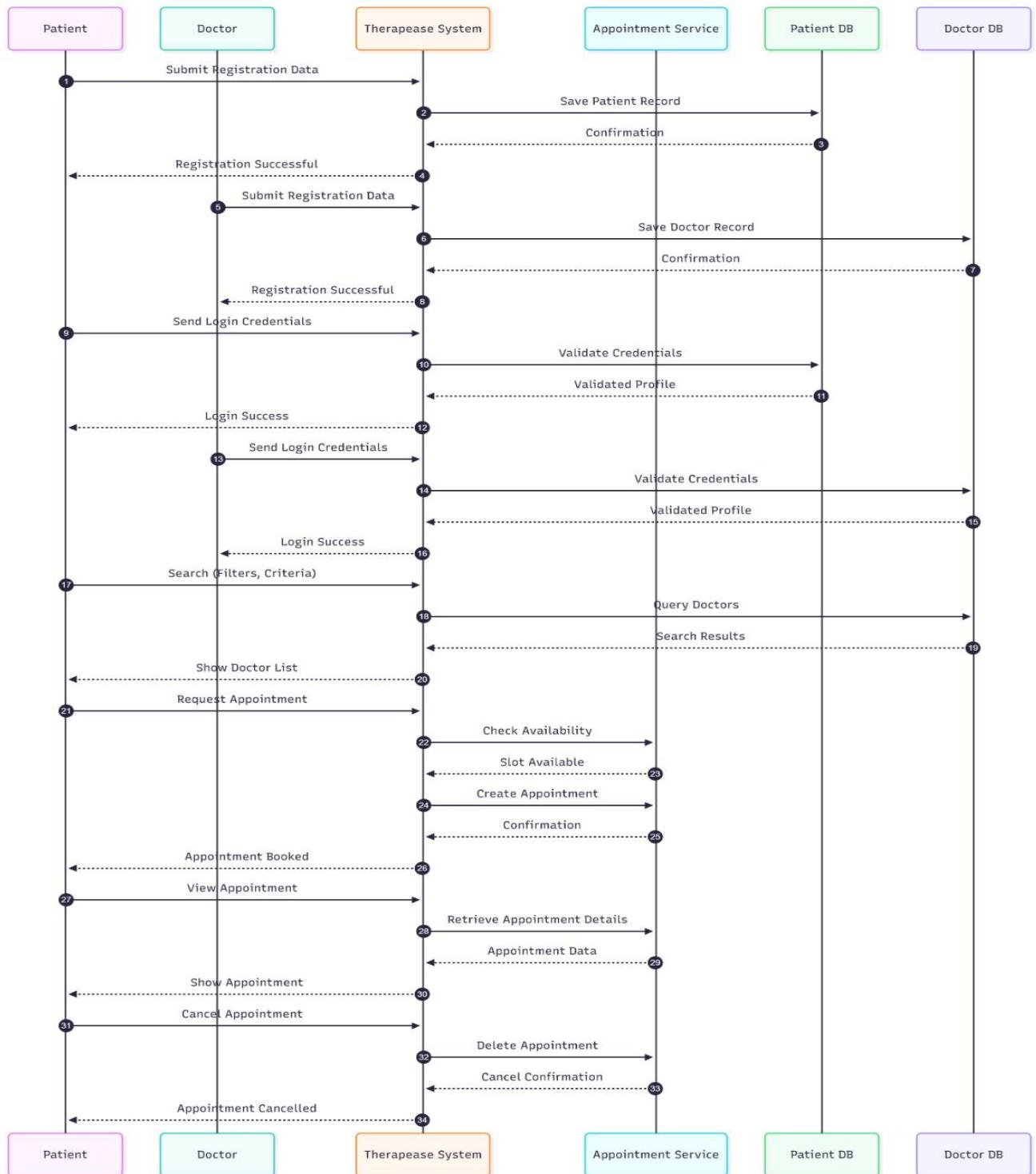
- a. Doctor Reviews Report
- b. He/She modifies Therapy or Exercises
- c. Writes a personal feedback based on the reports provided
- d. Stores Updated Prescription in Appointments Data Store

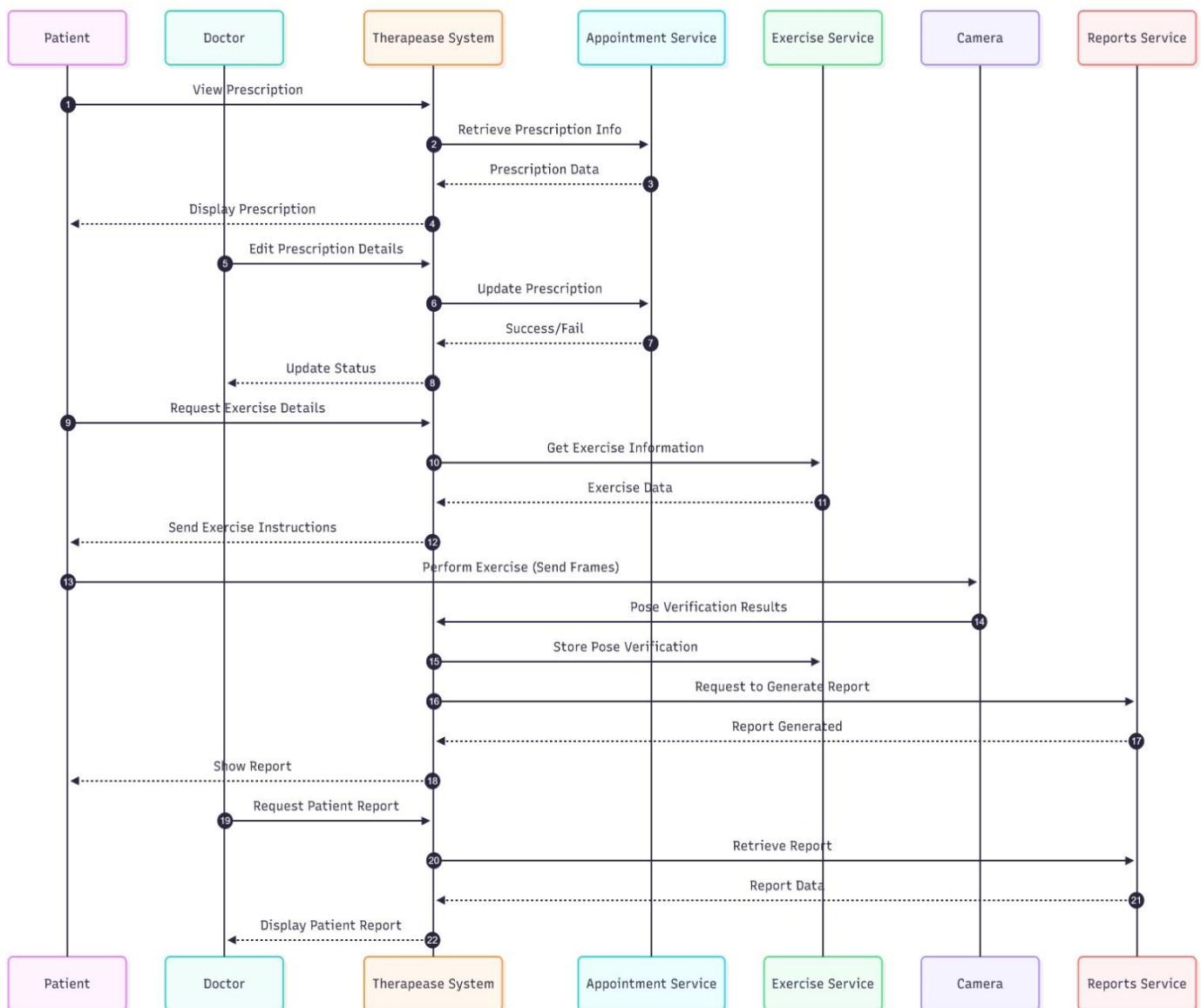
3.8 Structured Chart



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.





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	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\$\$HELLO32##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_patient	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

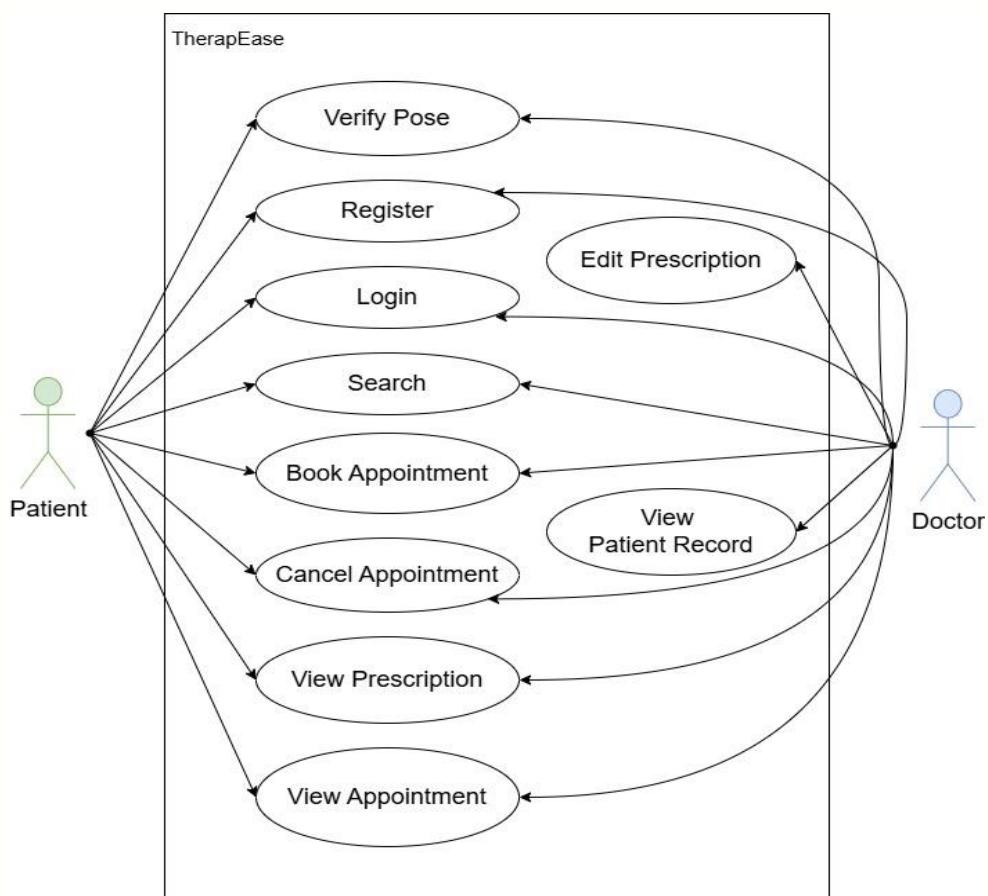
5 USE CASE DIAGRAM

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: bookAppointment

Actors:

- Patient
- (Doctor – as a notified/affected actor)

Description:

This use case defines the process of booking an appointment with a doctor. The patient searches for doctors, views doctor details, checks available time slots, and confirms a suitable appointment. The system ensures that only free slots are booked and stores the appointment details.

Preconditions:

- The patient is registered and logged into the TherapEase system.
- Doctor profiles and availability information are already stored in the system.

Postconditions:

- An appointment is created with a specific doctor, date, and time.
- Appointment details are stored and visible to both the patient and the doctor.

Flow of Events:

1. The patient logs into the TherapEase system.
2. The patient selects the option to search for doctors.
3. The system displays search filters (specialization, location, availability, etc.).
4. The patient enters search criteria and submits the request.
5. The system retrieves and displays a list of matching doctors.
6. The patient selects a doctor to view detailed information.
7. The patient chooses the “Book Appointment” option.
8. The system checks available appointment slots for the selected doctor.
9. The patient selects a preferred date and time slot.
10. The system verifies that the selected slot is still available.
11. The system creates an appointment entry and stores it in the appointment database.
12. A confirmation is shown to the patient, and the doctor’s schedule is updated.

Alternative Flows:

- If no doctors match the search criteria, the system displays an appropriate message and suggests modifying filters.

- If the selected time slot becomes unavailable during booking, the system notifies the patient and prompts them to choose another slot.

Exceptions:

- Failure to fetch doctor list due to backend or network error.
- Failure to create an appointment record due to database error.

5.3 Use Case 2: performExerciseSession

Use Case Name: performExerciseSession

Actors:

- Patient
- (Doctor – indirectly benefits by later viewing reports)

Description:

This use case describes how a patient performs a guided physiotherapy exercise session using TherapEase. The system uses the camera feed, tracks the patient's movements, evaluates the exercise form, counts repetitions, and stores session data for later reporting.

Preconditions:

- The patient is registered and logged into the TherapEase system.
- The patient has an active prescription or exercise plan defined by a doctor.
- The patient's device has a functioning camera and a stable internet connection.

Postconditions:

- Exercise session data (angles, counts, stages, timestamps, quality indicators) is stored.
- The system updates the patient's progress, which can be used for reporting and analysis.

Flow of Events:

1. The patient opens the TherapEase app and navigates to the exercise section.
2. The patient selects an exercise from the prescribed list (e.g., bicep curl, squat).
3. The patient starts the session by choosing “Start Exercise Session”.
4. The system sends a request to start a new session on the backend.
5. The backend activates the tracking process and starts reading frames from the camera.
6. As the patient performs the exercise, pose landmarks are detected on each frame.
7. The system calculates joint angles and applies exercise-specific logic to determine stage and count (e.g., “up”, “down”, repetitions).
8. The system continuously updates session data, including angle, count, stage, and form quality.

9. The patient can see live feedback (such as current count and form status) on the app.
10. When finished, the patient selects “Stop Session”.
11. The backend stops tracking and finalizes the session data.
12. The session results are stored and made available for future viewing in reports.

Alternative Flows:

- If pose landmarks cannot be detected reliably (poor lighting or camera angle), the system may show an instruction message asking the patient to adjust position or camera.
- If the session is interrupted (e.g., network loss), partial data may be stored and an error message is shown to the user.

Exceptions:

- Failure to access the camera or permission denied on the device.
- Backend tracking thread not starting or stopping correctly due to server error.

5.4 Use Case 3: viewProgressReports

Use Case Name: viewProgressReports

Actors:

- Patient
- Doctor

Description:

This use case defines how patients and doctors access detailed progress and performance reports generated from past exercise sessions. Reports may include repetition counts, session dates, average angles, and form quality indicators.

Preconditions:

- Exercise session data has been previously recorded for the patient.
- The patient or doctor is logged into the TherapEase system with appropriate access rights.

Postconditions:

- The requested report is generated (if needed) and displayed to the user.
- The user can review progress over time and, if implemented, export or share the report.

Flow of Events:

1. The user (patient or doctor) logs into the TherapEase system.
2. The user navigates to the “Reports” or “Progress” section.
3. The system prompts the user to select a time range or specific sessions (optional).
4. The user selects the desired filters and submits the request.
5. The system gathers stored exercise session data for the selected patient and time range.

6. The system processes the data and generates a structured report (e.g., tables, charts, summary statistics).
7. The generated report is displayed to the user.
8. Optionally, the user may choose to download, export, or share the report (if such functionality is supported).

Alternative Flows:

- If there is no recorded data for the selected period, the system displays a message indicating that no sessions are available for reporting.

Exceptions:

- Failure to retrieve stored session data due to database or network errors.
- Error in report generation logic, resulting in 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 exercise 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 metrics (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.

$$\text{EIF} = 0$$

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)

$$\text{Subtotal (EI)} = 4 + 3 + 4 + 4 + 3 + 3 + 4 = 25$$

External Inquiries (EQ)

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

$$\text{Subtotal (EQ)} = 3 + 4 + 4 = 11$$

External Outputs (EO)

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

$$\text{Subtotal (EO)} = 5 + 4 + 5 = 14$$

Internal Logical Files (ILF)

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

$$\text{Subtotal (ILF)} = 10 + 10 + 10 + 10 + 10 = 50$$

External Interface Files (EIF)

$$\text{Subtotal (EIF)} = 0$$

7.4 Unadjusted Function Points (UFP)

$$\text{UFP} = \text{EI} + \text{EO} + \text{EQ} + \text{ILF} + \text{EIF}$$

$$\text{UFP} = 25 + 11 + 14 + 50 + 0 = 100$$

7.5 Value Adjustment Factor (VAF)

$$FP = UFP \times VAF$$

Where:

$$VAF = 0.65 + (0.01 \times \Sigma Fi)$$

Two scenarios:

Scenario A — Minimal adjustment (unknown GSC values)

Assume each characteristic = 1:

$$\Sigma Fi = 14$$

$$VAF = 0.65 + 0.14 = 0.79$$

$$FP = 100 \times 0.79 = 79$$

Scenario B — Typical medium complexity

Assume each characteristic = 3:

$$\Sigma Fi = 42$$

$$VAF = 0.65 + 0.42 = 1.07$$

$$FP = 100 \times 1.07 = 107$$

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.

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.

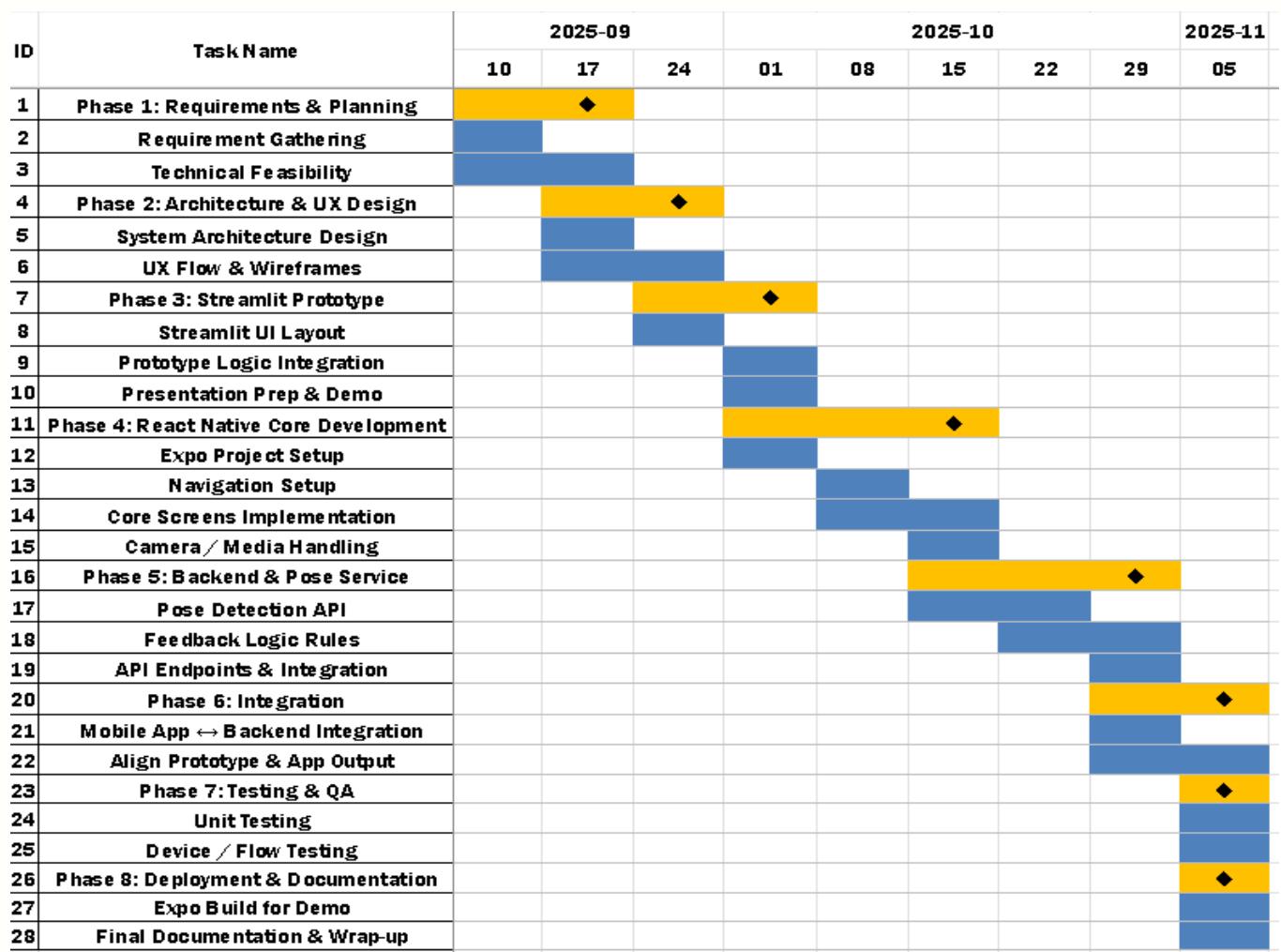
• Phase 8 – Deployment & Documentation

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

Project Timeline & Milestones					
Task ID	Description	Start Date	End Date	Duration	Budget (USD)
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 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.

Test Case BB-01: Angle Calculation with Known Points

Input:

Coordinates of three points forming a perfect right angle at the elbow, such as:

- $a = (1, 0)$
- $b = (0, 0)$
- $c = (0, 1)$

Expected Output:

`calculate_angle()` should return an angle close to 90° .

Actual Result:

The system returned 89.96° , which is within expected floating-point tolerance.

Status: PASS

Test Case BB-02: Bicep Curl Logic – Single Full Curl

Input:

Simulated pose landmark sequence where the elbow angle transitions from:

- Extended → angle $> 160^\circ$
- Flexed → angle $< 40^\circ$

Expected Output:

- Stage transitions "down" → "up"
- Count increases by +1

Actual Result:

- Stage correctly changed from "down" to "up"
- Count incremented from 0 to 1

Status: PASS

Test Case BB-03: Bicep Curl Logic – Multiple Repetitions

Input:

Landmark angle sequence for 5 complete curls repeating:

$160^\circ \rightarrow 35^\circ \rightarrow 165^\circ \rightarrow 40^\circ \rightarrow \dots$ (5 cycles)

Expected Output:

Total count = 5

Actual Result:

The backend returned count = 5, matching the expected repetitive pattern.

Status: PASS

Test Case BB-04: Squat Logic – Deep Squat Detection

Input:

Landmark angle sequence for 5 complete curls repeating:

- Knee angle transitions:
- Standing angle $\approx 170^\circ$
- Squat angle $\approx 85^\circ$

Expected Output:

- Stage transitions correctly: "up" → "down" → "up"
- Count increments by 1 after a full cycle

Actual Result:

- Stage changed as expected
- Count incremented by 1

Status: PASS**Test Case BB-05: No Landmarks Detected (Camera Not Seeing User)****Input:**

A frame with no visible human body

Expected Output:

- No crash
- current_data must not update angles or count
- Stage should remain unchanged

Actual Result:

- System remained stable
- No angle/count changes occurred
- No exceptions were thrown

Status: PASS**Test Case BB-06: Invalid Video Upload Format****Input:**

User uploads a non-video file (e.g., .txt)

Expected Output:

Backend should reject the file and return an error response.

Actual Result:

Backend responded with:

```
{"detail": "Invalid file format"}
```

App displayed an appropriate error message.

Status: PASS

Test Case BB-07: API Error Handling – Backend Offline

Input:

User attempts to start a session while the backend server is turned off.

Expected Output:

Frontend should show an error and remain stable.

Actual Result:

App showed:

“ Unable to connect to server.”

The application continued running without any crash.

Status: PASS

Test Case BB-08: Data Endpoint Response Under Load

Input:

User performs quick, continuous movements for 20 seconds.

Expected Output:

- /data should still return valid readings
- No delay > 1 second
- Angle + count values must update continuously

Actual Result:

- No delays observed
- All values updated smoothly
- No frame skips significant enough to affect output consistency

Status: PASS

9.3 White Box Testing in Unit Testing

Cyclomatic Complexity of Backend Functions

Cyclomatic Complexity (CC) is used to measure the structural complexity of a function.

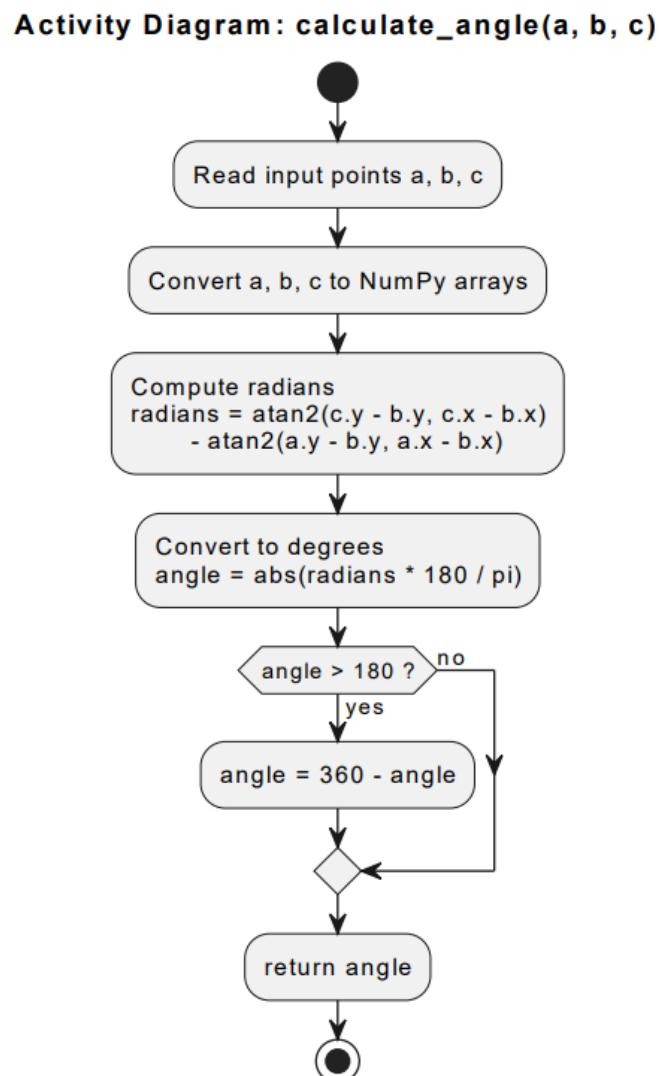
In this project, we calculate CC using the formula:

$$CC = D + 1$$

Where, D = number of predicate (decision) nodes (e.g., if, elif, while, for).

Below are the white box activity diagrams and CC calculations for the three primary functions used in exercise tracking.

1. Activity Diagram: calculate_angle(a, b, c)



This activity diagram visualizes the internal logic of the calculate_angle() function. It accepts three points (a, b, c), converts them into arrays, computes the angle using inverse tangent functions, converts to degrees, and ensures the angle remains within the 0–180° usable range.

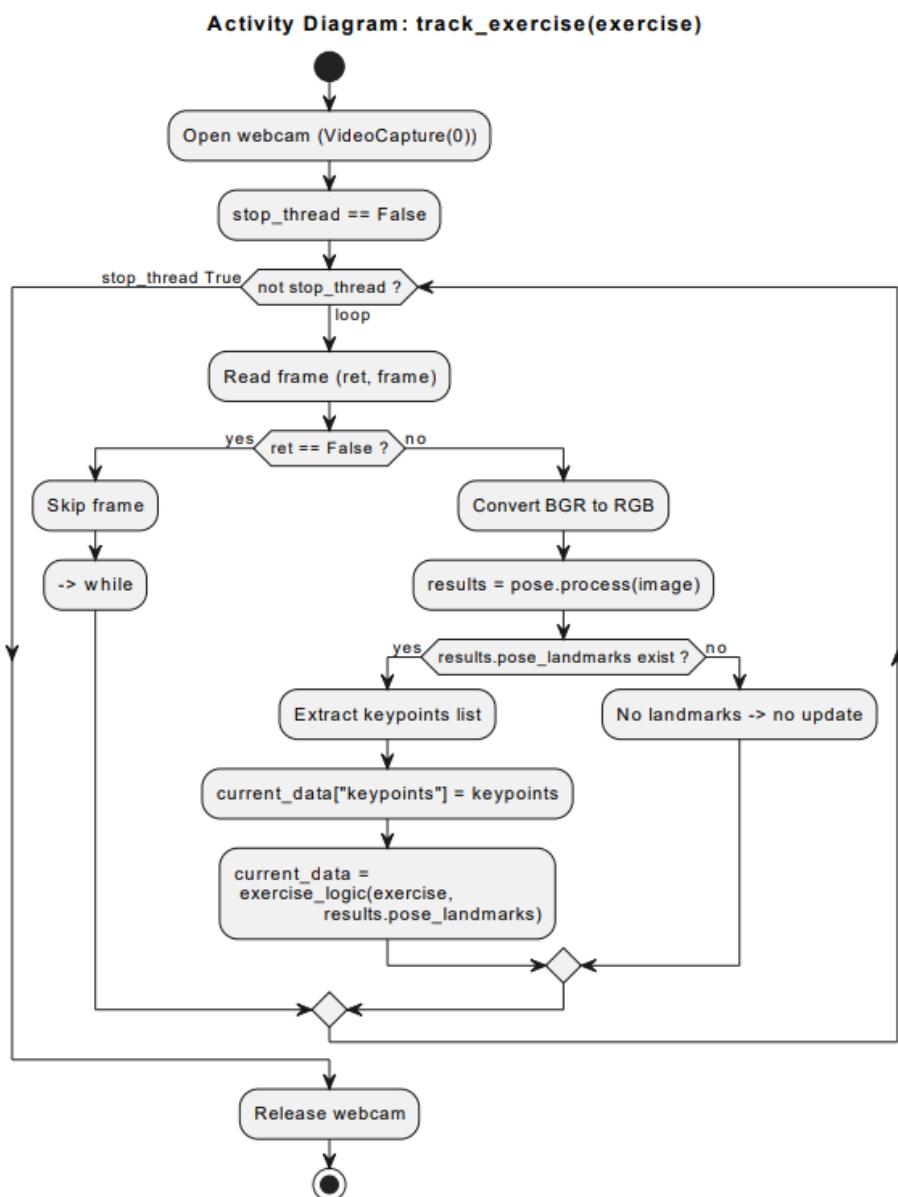
Decision Nodes

- if angle > 180 → 1 predicate

Cyclomatic Complexity Calculation

- D = 1
- CC = D + 1 = 1 + 1 = 2

2. Activity Diagram: track_exercise(exercise)



This diagram illustrates the real-time tracking loop that captures webcam frames, processes pose landmarks, updates keypoints, applies exercise logic, and continues until the stop flag is triggered.

Decision Nodes

- while not stop_thread → 1
- if ret == False → 1
- if results.pose_landmarks exist → 1

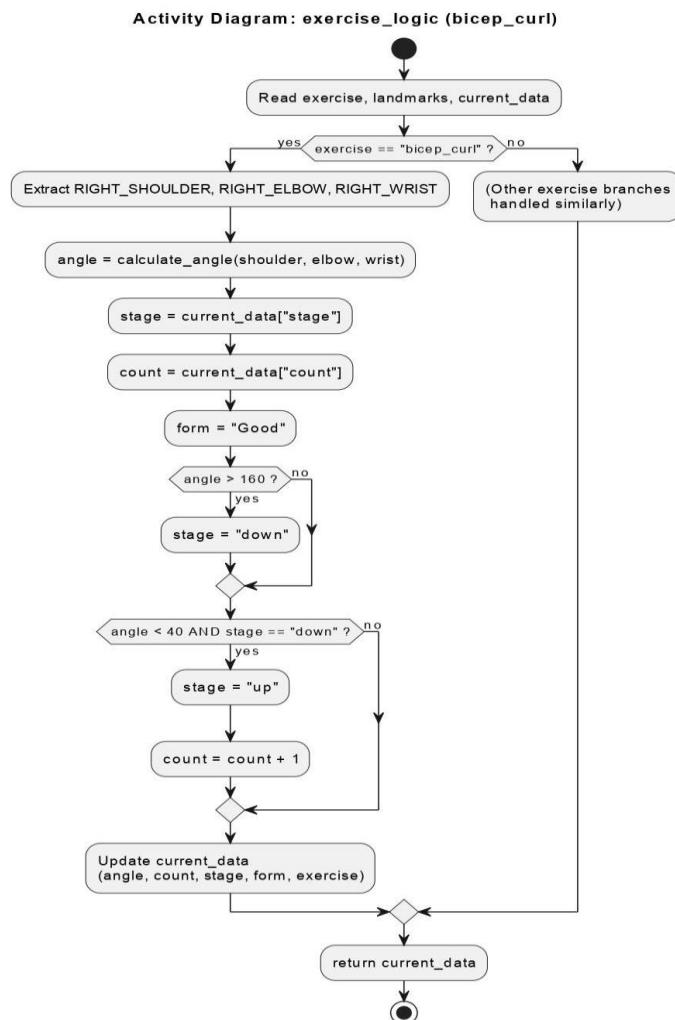
Total predicates: 3

Cyclomatic Complexity Calculation

- D = 3
- CC = 3 + 1 = 4

Final CC for analyze_frame(): 1

3. Activity Diagram: track_exercise(exercise)



This activity diagram represents the internal logic for the bicep_curl branch inside exercise_logic(). The full function contains multiple branches for other exercises, but this diagram shows one complete flow: extracting landmarks, computing angles, updating stage, validating movement, and incrementing repetition count.

Decision Nodes

For bicep_curl, decisions are:

- if exercise == "bicep_curl"
- If angle > 160
- if angle < 40 AND stage == "down"

Total for this single branch: 3

However, the full function contains **6 exercise branches and 12 inner conditions**.

Full Function Cyclomatic Complexity Calculation

- Outer exercise selection: 6
- Inner conditions: 12
- $D = 18$
- $CC = D + 1 = 18 + 1 = 19$

9.3.1 Conclusion

- calculate_angle() has a Cyclomatic Complexity of 2, meaning it is simple, linear, and easy to test fully.
- track_exercise() has a Cyclomatic Complexity of 4, typical for loop-based frame processing with multiple checks. It remains testable but requires path-based verification for all branches.
- exercise_logic() has a Cyclomatic Complexity of 19, indicating:
 - High decision density
 - Multiple branches
 - Requires extensive test coverage
 - Ideal candidate for path coverage and decision coverage testing techniques

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.

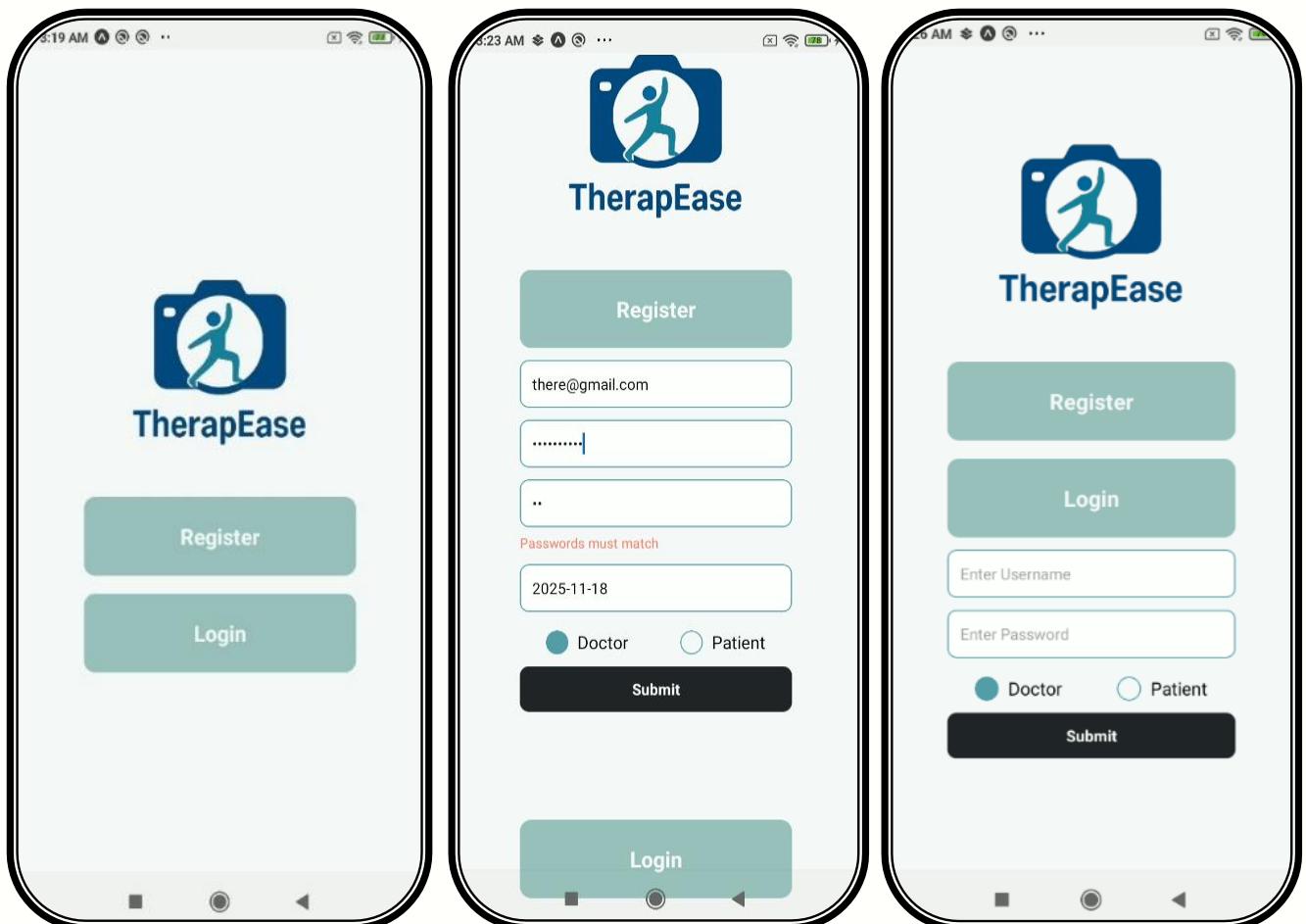


Fig 11.1 Registration Screen

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.

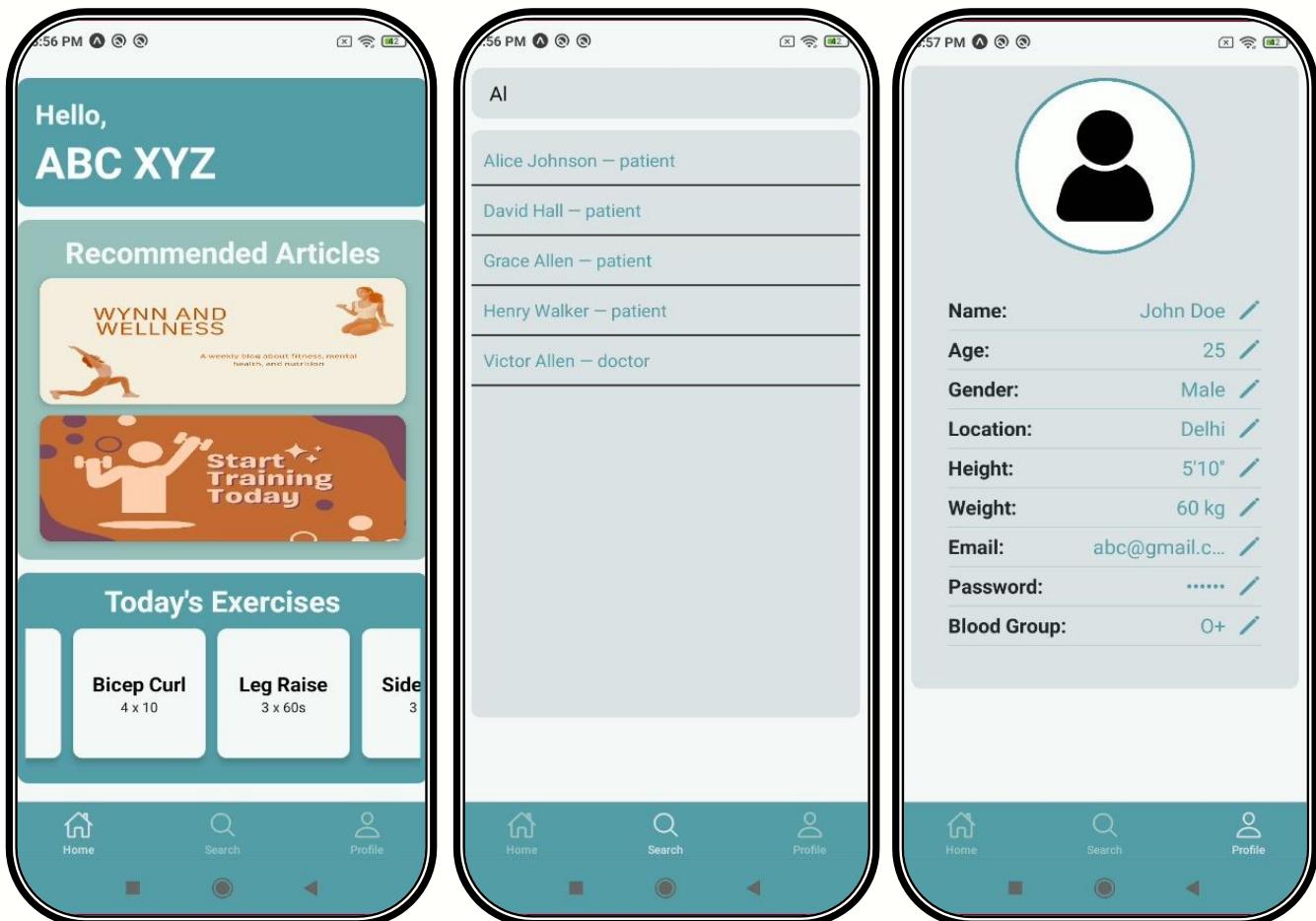


Fig 11.2 Patient Screen

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.

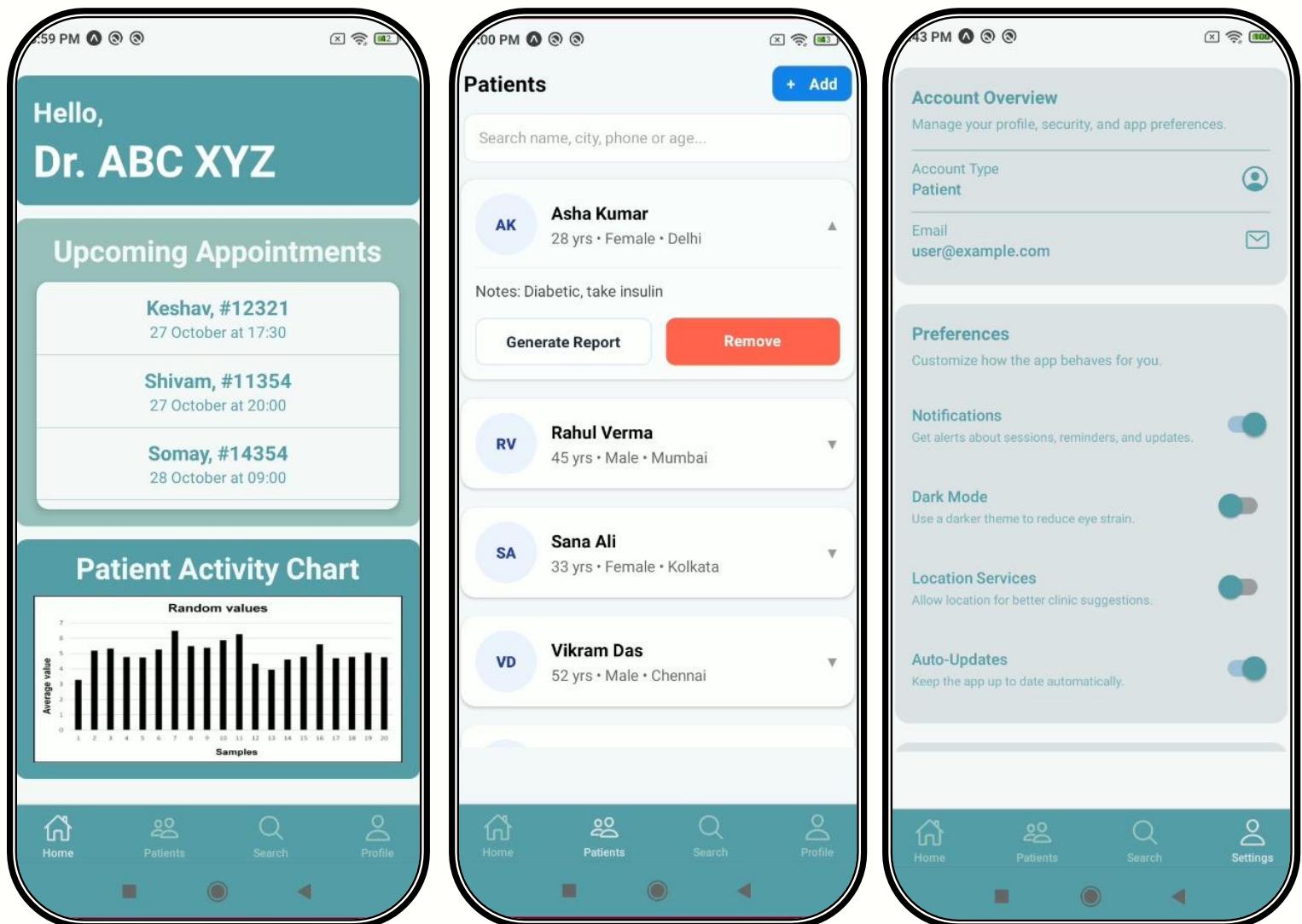


Fig 11.3 Doctor Screen

12.4 User Interface: Extras

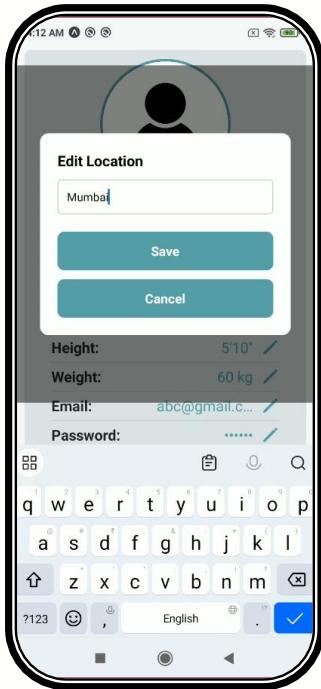


Fig 11.4 Edit Details Interface

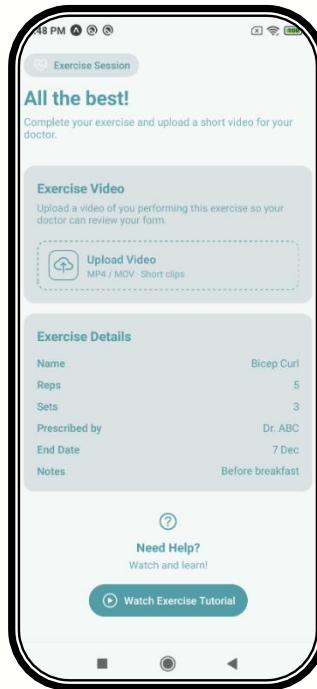


Fig 11.5 Exercise Session Interface

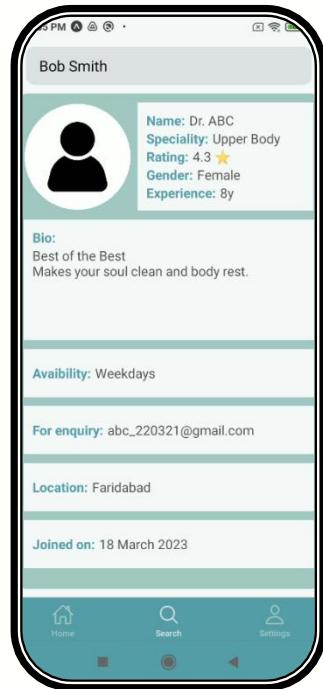


Fig 11.6 Profile found in the Search Results

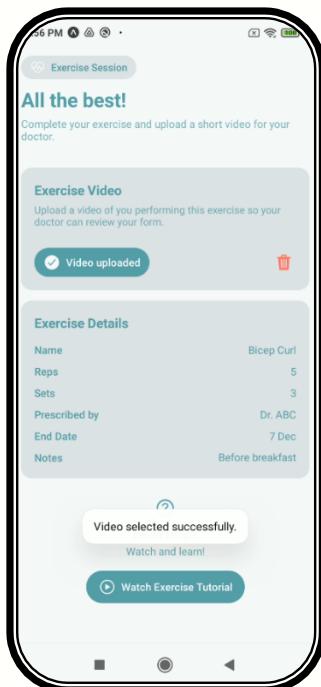


Fig 11.7 Upload Video Successful

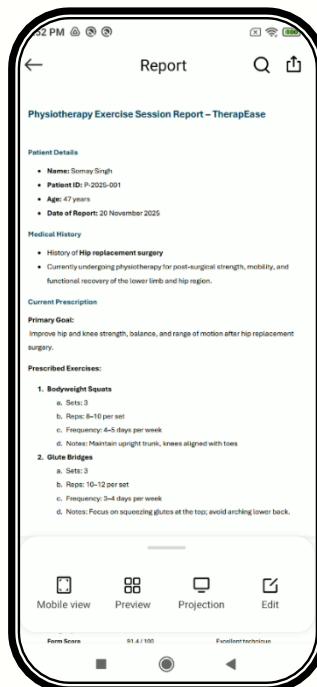


Fig 11.8 Generated PDF



Fig 11.9 Feedback on the basis of pose formation

12.5 Link To The Repository

[www.github.com/\\$\\$\\$\\$\\$\\$\\$\\$\\$\\$](http://www.github.com/$$$$$$$$$$)