CALM-MIND: AN INTERACTIVE WEB APPLICATION FOR MENTAL WELL-BEING

Mini Project Report

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

Bachelor of Technology (B.Tech)

in

COMPUTER SCIENCE AND ENGINEERING

By

G S MADHU BALA
21AG1A05E8

Under the Esteemed Guidance of
Ms. Shubhangi Mahule
Associate Professor

Department of Computer Science and Engineering
ACE ENGINEERING COLLEGE
An AUTONOMOUS Institution
NBA Accredited B. Tech Courses, Accorded NAAC 'A' Grade
(Affiliated to Jawaharlal Nehru Technological University, Hyderabad, Telangana)
Ankushapur (V), Ghatkesar (M), Medchal – Malkajgiri Dist - 501 301.
FEBRUARY 2025

ACE

Engineering College
An AUTONOMOUS Institution
Website: www.aceec.ac.in E-mail: info@aceec.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project work entitled **"CALM-MIND : AN INTERACTIVE WEB APPLICATION FOR MENTAL WELL-BEING"** is being submitted by **G S MADHU BALA (21AG1A05E8)** in partial fulfillment for the award of Degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** to the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2024-25 is a record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma.

Internal Guide	HoS	Dean-CSE
Ms. Shubhangi Mahule	Mr V Chandra Sekhar Reddy	Dr.M.V.Vijaya Saradhi

Project Coordinator	EXTERNAL EXAMINER
Mrs. Ch. Srivatsa	

ACKNOWLEDGEMENT

I would like to express our gratitude to all the people behind the screen who have helped me transform an idea into a real time application.

I would like to express our heart-felt gratitude to my parents without whom We would not have been privileged to achieve and fulfill my dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy,** for having founded such an esteemed institution. Sincere thanks to our COO **Mr. Y.V. Raghu Vamshi**, for support in doing project work. I am also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project. I am very much thankful to our beloved Vice Principal & Dean-Skill Development **Dr. M. Murali** for his continuous encouragement to fulfill my project.

I profoundly thank **Dr. M. V. Vijaya Saradhi**, Dean of Computer Science and Engineering, who has been an excellent guide and also a great source of inspiration to my work.

I extremely thank **Mr. V Chandra Sekhar Reddy**, HoS & Associate Professor, who helped us in all the way in fulfilling of all aspects in completion of my Mini-Project.

I sincerely thank **Mrs. Ch. Srivatsa**, Assistant Professor, for her continuous support and guidance throughout the project.

I am very thankful to my internal guide **Ms. Shubhangi Mahule**, Associate Professor, who has been an excellent and also given continuous support for the Completion of my project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my task.

By
G S MADHU BALA
(21AG1A05E8)

DECLARATION

I hereby declare that this mini project entitled "**CALM-MIND: An Interactive Web Application for Mental Well-being**" submitted to the ACE Engineering College, is a record of an original work done by me under the guidance of **Ms. Shubhangi Mahule**, Associate Professor, **ACE Engineering College**, and this project work submitted in the partial fulfillment of the requirements for the mini project; the results embodied in this thesis have not been submitted to any other university or institute for award of any degree or diploma.

G S MADHU BALA

(21AG1A05E8)

INDEX

SI. NO	CONTENTS	PAGE NO.
	ABSTRACT	v
1.	INTRODUCTION	1
2.	LITERATURE SURVEY	3
3.	SOFTWARE REQUIREMENT AND ANALYSIS	6
	3. 1 PROBLEM STATEMENT	6
	3. 2 MODULES AND THEIR FUNCTIONALITIES	6
4.	SOFTWARE DESIGN	8
	4. 1 ARCHITECTURE DIAGRAM	8
	4. 2 DATAFLOW DIAGRAMS	11
	4. 2. 1 DFD LEVEL - 0	12
	4. 2. 2 DFD LEVEL - 1	13
	4. 3 UML DIAGRAMS	15
	4. 3. 1 USECASE DIAGRAM	15
	4. 3. 2 CLASS DIAGRAM	16
	4. 3. 3 SEQUENCE DIAGRAM	18
	4. 3. 5 ACTIVITY DIAGRAM	20
	4. 3. 6. STATE CHART DIAGRAM	23
	4. 3. 7 DEPLOYMENT DIAGRAM	24
	4. 3. 8 COLLABORATION DIAGRAM	26
	4. 4 DATABASE DESIGN	28
	4. 4. 1 ER DIAGRAM	28
5.	SOFTWAERE ABD HARDWARE REQUIREMENTS	30
	5. 1 SOFTWARE REQUIREMENTS	30
	5. 2 HARDWARE REQUIREMENTS	30
6.	MODULES	31
	6.1 USER MODULE	31
	6. 2 SYSTEM MODULE	32
7.	IMPLEMENTATION	34

	7.1 USER MODULE IMPLEMENTATION	**34**
	7.2 SYSTEM MODULE IMPLEMENTATION	**37**
8.	TESTING	**51**
	8. 1 TESTCASES	**52**
9.	OUTPUT SCREENS	**54**
10.	DEPLOYMENT	**60**
11.	PERFORMANCE EVALUATION	**62**
12.	COMPARISION WITH EXISTING SYSTEM	**63**
13.	CONCLUSION	**64**
14.	FUTURE ENHANCEMENTS	**65**
15.	REFERENCES	**66**

ABSTRACT

In an era where mental well-being is a critical aspect of overall health, technology-driven solutions are transforming personal wellness practices. This project, "Interactive Web Application for Mental Well-Being," introduces a comprehensive platform that integrates personalized yoga recommendations with an emotion-aware chatbot to support holistic health.

The Yoga Module assesses users' emotional states through 19 predefined moods and suggests three tailored yoga poses from a dataset of 77, ensuring targeted emotional relief. Advanced computer vision techniques, including the YOLO (You Only Look Once) model for precise pose detection and PoseNet for extracting key joint features, enhance accuracy. An angle heuristic algorithm further analyzes posture, providing real-time corrective feedback to optimize user experience and safety.

The Chatbot Module enriches user engagement by recognizing and responding to 59 distinct emotions. Through natural language processing, the chatbot interacts with users, explores the reasons behind their emotions, offers empathetic responses, and recommends relevant YouTube videos for emotional regulation. Continuous feedback collection refines its responses, ensuring a personalized and supportive experience.

This web application delivers a unique and effective solution for individuals seeking to enhance both mental and physical well-being. The system's personalized approach empowers users to cultivate mindfulness, emotional balance, and a healthier lifestyle.

LIST OF FIGURES

FIG. NO.	FIGURE NAME	PAGE NO.
4. 1	ARCHITECTURE DIAGRAM	1
4. 2. 1	DFD LEVEL - 0	12
4. 2. 2	DFD LEVEL - 1	13
4. 3. 1	USECASE DIAGRAM	15
4. 3. 2	CLASS DIAGRAM	16
4. 3. 3	SEQUENCE DIAGRAM	18
4. 3. 4	ACTIVITY DIAGRAM	20
4. 3. 5	STATE CHART DIAGRAM	23
4. 3. 6	DEPLOYMENT DIAGRAM	24
4. 3. 7	COLLABORATION DIAGRAM	27
4. 4. 1	ER DIAGRAM	28
6. 1	MODULES DIAGRAM	33
9. 1	INDEX PAGE	54
9. 2	REGISTRATION PAGE	54
9. 3	LOGIN PAGE	55
9. 4	MOOD SELECTION PAGE	55
9. 5	YOGA RECOMMENDATION PAGE	56
9. 6	CORRECTION RESULT PAGE	56
9. 7	DASHBOARD PAGE	56
9. 8	ABOUT PAGE	57
9. 9	CHATBOT INTERACTION(1)	58
9. 10	CHATBOT INTERACTION(2)	59
9. 11	RECOMMENDED VIDEOS	59
9. 12	CHATBOT INTERACTION(3)	59
11. 1	PERFORMANCE EVALUATION BASED ON TEST CASES	62

LIST OF TABLES

Tab. No.	Table Name	Page No.
2. 1	LITERATURE SURVEY	5
7. 1	USER DATABASE	35
7. 2	YOGA RECOMMENDATION	37
7. 3	DASHBOARD TABLE	38
8. 1	TEST CASES	52
8. 2	TESTCASE MODEL BUILDING	53

LIST OF ABBREVIATIONS

S. No.	Abbrevation	Full Form
1	YOLO	You Only Look Once
2	DL	Deep Learning
3	ER	Entity Relational Diagram
4	SVD	Singular Value Decomposition
5	ML	Machine Learning

CHAPTER 1

INTRODUCTION

In today's fast-paced and digitally-driven world, mental well-being has emerged as a critical component of overall health. The increasing prevalence of stress, anxiety, and other mental health challenges underscores the need for accessible and effective solutions that can seamlessly integrate into individuals' daily lives. Concurrently, the rise of digital health technologies presents unprecedented opportunities to enhance traditional wellness practices through personalization and real-time feedback. It is against this backdrop that the "Interactive Web Application for Mental Well-Being" is envisioned—a pioneering platform that would bring together the best of physical and emotional well-being through the synergistic integration of personalized yoga practices and an emotion-aware chatbot.

Yoga, a practice so revered for its benefits in terms of promoting physical flexibility, strength, and mental tranquility, usually requires consistent guidance to ensure proper posture and technique. Traditional approaches, either in-person or through pre-recorded video instructions, often fail to offer one-on-one feedback and can be time-consuming and, at times, more dangerous. Furthermore, although yoga is inherently conducive to mental well-being, a lack of emotional support system makes it less effective for the complex challenges of mental health.

These gaps are addressed through the proposed web application, which is based on computer vision, machine learning, and natural language processing to offer a comprehensive wellness experience. The application is broadly categorized into two modules: the Yoga Module and the Chatbot Module. The Yoga Module leverages the YOLO (You Only Look Once) model for accurate real-time pose detection and PoseNet for extracting key joint features, enabling precise pose analysis and correction through an angle heuristic algorithm. This ensures that users receive immediate, personalized feedback on their yoga practice, enhancing both safety and effectiveness.

The Chatbot Module complements this physical aspect: it is designed to be the empathetic buddy that engages people by recognizing, responding to and acknowledging a variety of emotions and emotional states; it encourages those conversations that touch on why their emotional state of being is; and it brings supportive resources with curated YouTube videos to help someone regulate their emotion and build psychological resilience. This two-fold approach does not only individualize the experience of yoga to match the users' emotional needs but also provides them with continuous emotional support, hence giving a complete tool for holistic well-being.

Additionally, the application has an easy-to-use interface with functionalities that include mood measurement, pose recommendation according to the individual's needs, upload of pictures for pose measurement, and a dashboard for personal tracking. Strong user authentication and data management safeguard sensitive information in line with rigorous privacy standards.

The integration of these complex technologies within a single platform is one of the giant leaps in digital wellness solutions. This is the "Interactive Web Application for Mental Well-Being," which helps individuals take active steps to preserve and improve their mental and physical well-being through customized physical activity suggestions and strong emotional support. This approach, which is innovative, not only makes yoga practice accessible and effective for users of all levels but also bridges the gap between physical exercise and emotional well-being, creating a balanced and healthy lifestyle in an increasingly complex world.

CHAPTER 2

LITERATURE SURVEY

Several studies have explored various approaches to integrating technology with wellness practices, particularly in the areas of mental health, yoga, and physical well-being.

[1] "Web-Based Cognitive Behavioral Therapy and Chatbot Support" by Andersson & Titov (2014) This study explored the use of web-based cognitive behavioral therapy (CBT) and chatbot support to enhance accessibility to mental health resources. The research highlighted how digital interventions could provide mental health support to a broader audience. However, challenges such as personalization and maintaining user engagement were noted as limitations.

[2] "Ethical Considerations in Emotional AI and Recommender Systems" by Calvo & Peters (2014) This paper addressed ethical concerns related to emotional AI and recommendation systems. It discussed how AI-driven systems could personalize experiences while maintaining ethical standards. However, a significant challenge was balancing personalization with data security, as emotional data is sensitive and prone to misuse.

[3] "Mobile Apps for Mindfulness and Stress Reduction" by Bakker et al. (2016) This research analyzed the effectiveness of mobile applications in promoting mindfulness and stress reduction. The study found that mobile apps provided convenient access to mindfulness techniques, helping users manage stress. However, the success of these applications depended heavily on user commitment and engagement.

[4] "Ethical Issues in AI-Driven Mood Detection and Recommendations" by Mittelstadt et al. (2016) This paper examined the ethical concerns surrounding AI-driven mood detection and recommendation systems. It highlighted the potential risks associated with privacy breaches and biases in AI models. The research stressed the importance of ethical AI development to ensure fairness and security in mental health applications.

[5] "Integrating Lifestyle Changes with Digital Mental Health Tools" by Mohr et al. (2017) The study focused on how digital mental health tools could facilitate lifestyle changes, leading to improved mental health outcomes. It emphasized the necessity of personalization to

ensure the effectiveness of these tools. Without personalized interventions, users were less likely to benefit from digital mental health programs.

[6] "Pose Estimation Techniques for Fitness and Sports Applications" by Papandreou et al. (2018) This research explored pose estimation techniques for fitness and sports applications, achieving high accuracy in detecting keypoints. The study demonstrated the effectiveness of pose estimation models in fitness tracking. However, handling complex yoga postures remained a challenge, requiring further fine-tuning for precise results.

[7] "Yoga Alignment Evaluation Using Pose Estimation Models" by Sharma et al. (2019) This paper evaluated yoga alignment using pose estimation models, providing posture corrections based on AI analysis. The findings indicated that these models could effectively analyze and correct yoga postures. However, accuracy was highly dependent on the availability of well-annotated datasets for training the models.

[8] "Multi-Modal Feedback Systems for Fitness and Well-Being" by Laamarti et al. (2020) The research investigated multi-modal feedback systems combining verbal, textual, and visual corrections for enhanced fitness experiences. The study found that integrating various feedback modalities improved user engagement. However, the development of precise AI models remained a challenge to ensure accurate feedback.

[9] "Hybrid Recommender Systems for Personalized Wellness" by Abbas et al. (2022) This paper explored hybrid recommender systems in personalized wellness applications. The findings indicated that hybrid models significantly improved user engagement and personalization. However, the collection of emotional data raised privacy concerns, highlighting the need for secure data management practices.

[10] "Deep Learning-Based Pose Estimation in Fitness and Yoga" by Chen et al. (2022) The study analyzed the application of deep learning-based pose estimation in fitness and yoga. The research demonstrated improvements in accuracy for fitness applications. However, the model required fine-tuning to accurately interpret complex yoga movements, suggesting room for further enhancement.

Year	Author(s)	Title	Advantages	Disadvantages
2014	Andersson & Titov	Web-Based Cognitive Behavioral Therapy and Chatbot Support	Improved accessibility to mental health resources using web-based CBT.	Faced challenges in personalization and user engagement.
2014	Calvo & Peters	Ethical Considerations in Emotional AI and Recommender Systems	Addressed ethical concerns in emotional AI and recommendation systems.	Balancing personalization with data security remained difficult.
2016	Bakker et al.	Mobile Apps for Mindfulness and Stress Reduction	Mobile apps provided easy access to mindfulness and stress reduction.	Effectiveness depended on user commitment and engagement.
2016	Mittelstadt et al.	Ethical Issues in AI-Driven Mood Detection and Recommendations	Identified ethical concerns in AI-driven mood detection and recommendations.	Highlighted privacy risks and potential biases in AI models.
2017	Mohr et al.	Integrating Lifestyle Changes with Digital Mental Health Tools	Digital tools integrated lifestyle changes, improving mental health.	Emphasized that personalization was critical for effectiveness.
2018	Papandreou et al.	Pose Estimation Techniques for Fitness and Sports Applications	Achieved high accuracy in detecting keypoints for fitness applications.	Struggled with handling complex yoga postures, requiring fine-tuning.
2019	Sharma et al.	Yoga Alignment Evaluation Using Pose Estimation Models	Evaluated yoga alignment and provided posture corrections.	Accuracy depended on well-annotated datasets.
2020	Laamarti et al.	Multi-Modal Feedback Systems for Fitness and Well-Being	Combined verbal, textual, and visual corrections for enhanced experience.	Required advanced AI models for precise feedback.
2022	Abbas et al.	Hybrid Recommender Systems for Personalized Wellness	Hybrid recommender systems enhanced personalization and engagement.	Raised privacy concerns regarding emotional data collection.
2022	Chen et al.	Deep Learning-Based Pose Estimation in Fitness and Yoga	Deep learning-based pose estimation improved fitness and yoga applications.	Needed fine-tuning for specific yoga movements.

Table 2.1. Literature Survey

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

3.1. PROBLEM STATEMENT

Traditional yoga practice often lacks personalized guidance and correction, leading to potential inefficiencies and injury risks, while emotional support systems are limited. By integrating computer vision, machine learning, and natural language processing, our project aims to provide a solution that analyzes users' mood inputs, recommends suitable yoga poses, offers pose correction, and interacts through a supportive chatbot. This addresses the need for personalized guidance, correction, and emotional support in well-being practices, catering to individual needs and enhancing overall well-being.

3.2. MODULES AND THEIR FUNCTIONALITIES

1. System Module:

a. Yoga system module:

- **i.** Mood Assessment: Collects user input on their current mood or emotional state.
- **ii.** Yoga Pose Recommendation: Recommends 3 yoga poses based on the user's selected mood.
- **iii.** Pose Detection: Detects yoga poses from images uploaded by the user.
- **iv.** Pose Estimation: Estimates key features of detected yoga poses, such as joint positions and angles.
- **v.** Pose Correction: Provides real-time feedback and correction guidance on pose execution.

b. Chabot Module:

- **i.** Emotion Detection: Recognizes and responds to 59 distinct emotions based on user input.
- **ii.** Conversational Flow: Guides the user through their emotions, asks for reasons behind their feelings, and provides empathetic support.
- **iii.** Resource Recommendation: Recommends relevant YouTube videos to aid in emotional regulation.

iv. Feedback Gathering: Collects user feedback post-interaction to refine responses and improve user experience.

2. User Module:

- **Register:** Users can register with their credentials such as email and password.
- **Login:** Users can log in with their registered credentials.
- **Mood Selection:** Users can select their current mood.
- **Getting Yoga Recommendations:** Based on the user's selected mood, 3 yoga poses will be recommended.
- **Upload Image:** Users can select one yoga pose from the 3 recommendations and upload their yoga pose image for the selected pose.
- **Getting Pose Correction:** The uploaded yoga pose image will be analyzed by the trained model, which provides corrections for the pose. Users can then view the corrected pose image. This information (selected mood, uploaded image, pose-corrected image) will be stored in the database.
- **Dashboard:** Users can view their history on the dashboard page. Here they can view their selected mood, uploaded image, and pose-corrected image ordered by date. Users can also retrieve history data by a specific date or period.
- **Chat with Bot:** Users can interact with the chatbot for emotional support and recommendations.
- **Logout:** After completing their activities, users can log out from the website.

CHAPTER 4
SOFTWARE DESIGN

4.1. ARCHITECTURE DIAGRAM

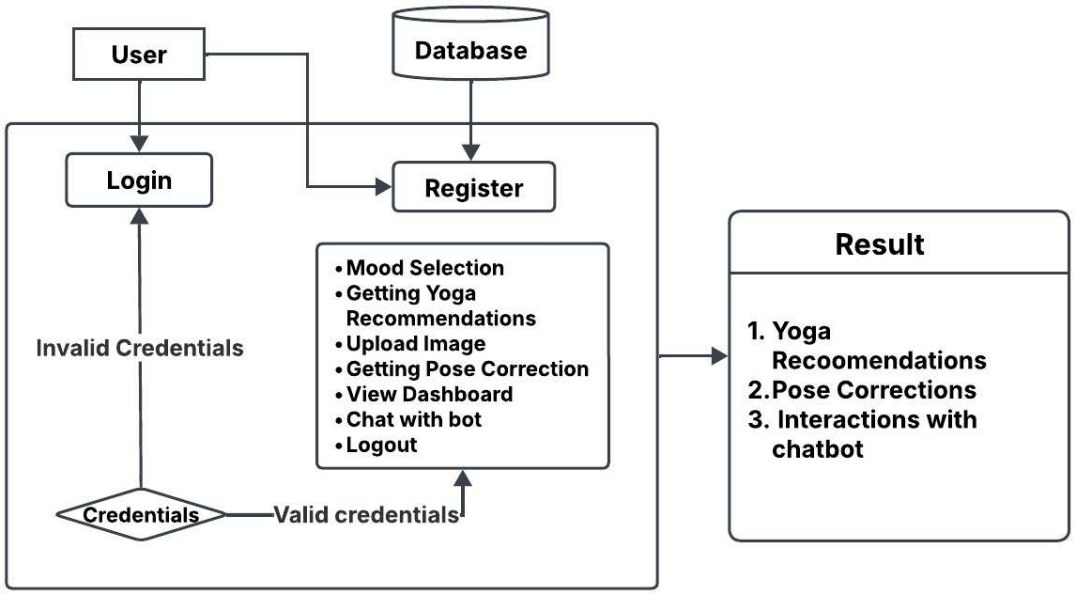

Fig 4. 1. ARCHITECTURE DIAGRAM

This architecture diagram represents a **Yoga Recommendation and Pose Correction System** with user authentication and chatbot interaction.

1. SYSTEM ARCHITECTURE :

The project consists of two main components:

- **Chatbot System** – A static chatbot that interacts with users based on a predefined dataset to provide emotional support.
- **Yoga Pose System** – A system that:
  - Recommends yoga poses based on user moods.
  - Identifies and corrects yoga poses using computer vision techniques.

- **Technology Stack:**
    - **Backend:** Python (Flask)
    - **Frontend:** HTML, CSS, Bootstrap, JavaScript
    - **Data Handling:** JSON for chatbot responses, CSV for yoga pose recommendations
    - **Machine Learning Models:** Singular Value Decomposition (SVD) for recommendations, YOLO & MoveNet PoseNet for pose identification

2. CHATBOT SYSTEM – DATA PROCESSING & ARCHITECTURE:

1. **Dataset Preparation:**
    - Collect predefined responses in a structured JSON format.
    - Standardize the dataset to ensure consistency.
    - Store information in attributes such as:
        - **name:** Emotion type (e.g., happy, sad).
        - **follow_up_question:** Helps refine user emotion.
        - **reasons:** Categories for emotional causes.
        - **comfort:** Empathetic responses.
        - **videos:** Recommended resources.
        - **feedback_question**: Evaluates chatbot effectiveness.
        - **feedback_positive & feedback_negative:** Adapts responses based on user feedback.
2. **Backend Processing with Flask:**
    - **Greeting:** Starts interaction with a welcome message.
    - **Emotion Selection:** Processes selected emotion and asks for reasons.
    - **Resource Sharing:** Provides comforting messages and video recommendations.
    - **Feedback Collection:** Asks for user feedback and adjusts responses.
    - **Closure:** Ends conversation with an option to restart or exit.

3. YOGA POSE RECOMMENDATION SYSTEM:

1. **Dataset Preparation:**
    - Collect and clean data in Recommendation_yoga_data.csv.
    - Ensure consistent formatting for mood and pose names.
2. **Feature Engineering:**
    - Mapping Moods and Poses: Assign unique indices to facilitate matrix creation.
3. **Interaction Matrix Construction:**
    - Construct a sparse matrix where:
        - Rows represent moods.
        - Columns represent yoga poses.
        - Cell values indicate the frequency of interactions.
4. **Collaborative Filtering with SVD:**
    - Apply Singular Value Decomposition (SVD) to reduce dimensions.
    - Extract latent factors representing patterns in yoga pose preferences.
5. **Recommendation Generation:**
    - Compute scores for each yoga pose based on extracted features.
    - Return Top-N Recommended Poses for a given mood.

4. YOGA POSE IDENTIFICATION & CORRECTION SYSTEM

1. **Image Dataset Preparation:**
    - Collect images of yoga poses, stored in directories labelled with pose names.
2. **Person Detection & Cropping:**
    - Utilize **YOLOv3** to detect and crop the person in the uploaded image.
3. **Pose Key point Extraction:**
    - Use **PoseNet** to extract **17 key points** from the cropped image.
4. **Reference Key points Storage:**
    - Save extracted key points in a structured JSON file.
5. **Pose Evaluation with Angle Calculation:**
    - Define key points specific to each yoga pose.
    - Use **law of cosines** to compute angles between key points.
6. **Pose Correction Feedback Generation:**
    - Compare extracted angles with reference angles.
    - Generate real-time feedback on incorrect positioning.

7. **Feedback Delivery & User Guidance:**
    - Display pose correction feedback to help users adjust their posture.

5. FINAL INTEGRATION STEPS

1. **Load Pre-trained Models:**
    - **YOLOv3** for person detection.
    - **PoseNet** for pose estimation.
    - **SVD for pose recommendations.**
2. **Frontend & Backend Communication:**
    - Flask API handles user requests.
    - JSON responses provide chatbot and yoga guidance outputs.
3. **User Interaction Flow:**
    - Chatbot dynamically retrieves and displays emotion-based suggestions.
    - Yoga system processes images and provides pose correction feedback

4.2. DATA FLOW DIAGRAMS

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

This **Data Flow Diagram (DFD)** represents the **interaction between users, the system components, and the datastore** in your application. It consists of two primary entities:

1. **User (External Entity)** – Represents actions performed by users while interacting with the system.
2. **System (Processes)** – Represents different system functionalities that process user inputs and interact with the datastore.
3. **Datastore (Storage)** – Stores processed data, user inputs, and system outputs.

4.2.1. DFD LEVEL-0

A Level-0 Data Flow Diagram (DFD): It provides a high-level overview of the system, showing the main processes, user interactions, and data flow without detailing internal operations.

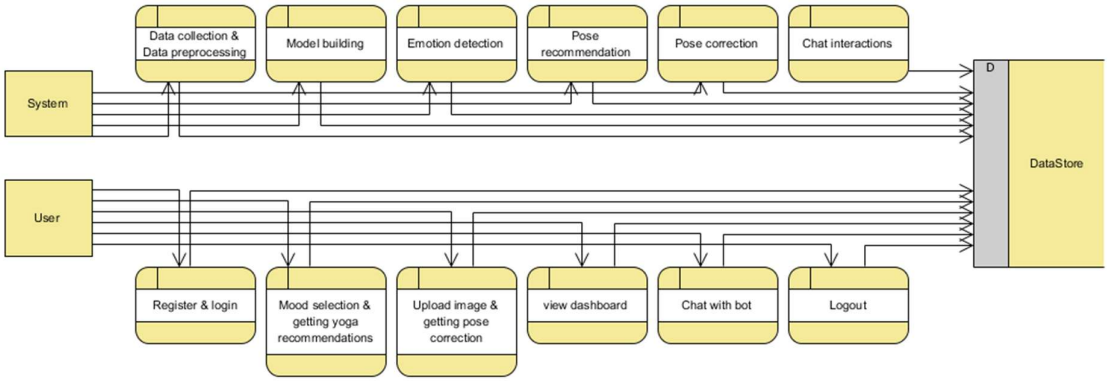

Fig 4.2.1. DFD LEVEL-0

1. User Interactions:

Users interact with the system through the following steps:

- **Register & Login:** User authentication and account access.
- **Mood Selection & Yoga Recommendations:** Users input their mood, and the system recommends suitable yoga poses.
- **Upload Image & Pose Correction:** Users upload an image for yoga pose validation, and the system provides corrective feedback.
- **View Dashboard:** Users access their past interactions and recommendations.
- **Chat with Bot:** Users interact with the chatbot for emotional support.
- **Logout:** Users exit the system.

Each of these user actions sends data to the **system processes**, which interact with the datastore to retrieve or store information.

2. System Processing:

The system performs the following key operations:

- **Data Collection & Preprocessing:** Gathers and processes data for chatbot responses and yoga recommendations.
- **Model Building:** Trains and refines models for pose correction and recommendations.
- **Emotion Detection:** Identifies emotions from user inputs to provide relevant responses.

- **Pose Recommendation:** Uses collaborative filtering (SVD) to suggest yoga poses based on moods.
- **Pose Correction:** Uses computer vision (YOLO & PoseNet) to analyse and correct yoga poses.
- **Chat Interactions:** Manages conversations using a predefined dataset to provide support.

Each system process exchanges data with the datastore to maintain user history, chatbot responses, and model outputs.

3. Datastore:

The **datastore (D)** serves as a central repository that stores:

- User credentials and session data.
- Chatbot dataset and conversation history.
- Yoga pose recommendations and user interactions.
- Pose correction reference key points and processed feedback.

4.2.2. DFD LEVEL-1

Data Flow Diagram (DFD) Level-1: The main functionalities of a system into detailed processes and interactions between the User, System, and Data Store.

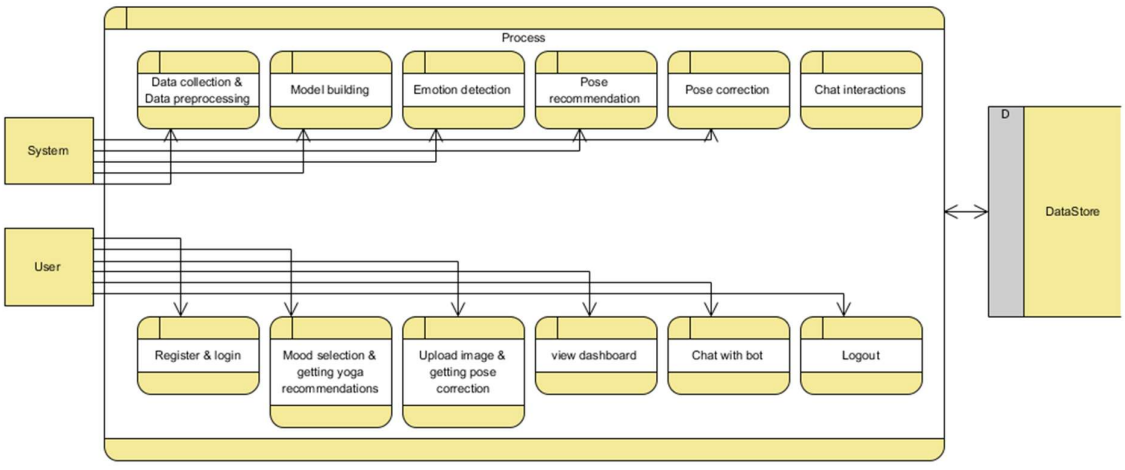

Fig 4.2.2. DFD LEVEL-1

1. **Entities:**
   - **User**: Represents the end-user who interacts with the system.
   - **System**: Contains various processes that handle user inputs and perform operations.
   - **Data Store**: A storage unit that holds data for various operations.

2. **User Interactions:**
   - **Register & Login**: Users register and log in to the system.
   - **Mood Selection & Getting Yoga Recommendations**: Users select their mood, and the system provides personalized yoga recommendations.
   - **Upload Image & Getting Pose Correction**: Users upload an image, and the system corrects their pose.
   - **View Dashboard**: Users can view their analytics or progress on a dashboard.
   - **Chat with Bot**: Users interact with a chatbot for guidance.
   - **Logout**: Users exit the system.

3. **System Processes:**
   - **Data Collection & Preprocessing**: The system gathers and prepares data for further analysis.
   - **Model Building**: AI/ML models are built to process user inputs.
   - **Emotion Detection**: The system detects the user's emotions based on inputs.
   - **Pose Recommendation**: The system suggests yoga poses based on the user's mood.
   - **Pose Correction**: The system analyzes and corrects the user's posture.
   - **Chat Interactions**: A chatbot interacts with users to assist them.

4. **Data Flow & Storage:**
   - Arrows indicate the flow of data between the **User**, **System**, and **Datastore**.
   - All processes store or retrieve relevant data from the **Datastore**, ensuring efficient system operation.

4.3. UML DIAGRAMS

4.3.1. USECASE DIAGRAM

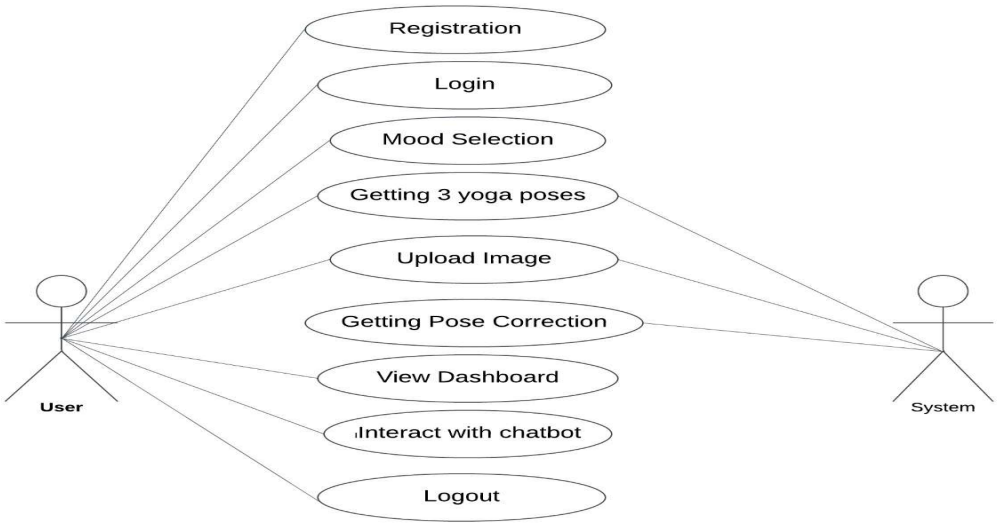

Fig 4.3.1 USECASE DIAGRAM

This image is a Use Case Diagram that represents the interaction between a User and a System

Description:

The diagram consists of two actors:

User (on the left)

System (on the right)

The user interacts with various functionalities of the system, represented by ovals. These functionalities include:

- **Registration** – Allows a new user to sign up.
- **Login** – Authenticates an existing user.
- **Mood Selection** – Enables users to choose their current mood.
- **Getting 3 Yoga Poses** – Provides three recommended yoga poses based on user input.
- **Upload Image** – Lets users upload an image, possibly for analysis.
- **Getting Pose Correction** – Analyses the uploaded image and provides feedback on the yoga pose.
- **View Dashboard** – Displays user progress and other relevant data.
- **Interact with Chatbot** – Allows communication with a chatbot for assistance.
- **Logout** – Ends the user session.

4.3.2. CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

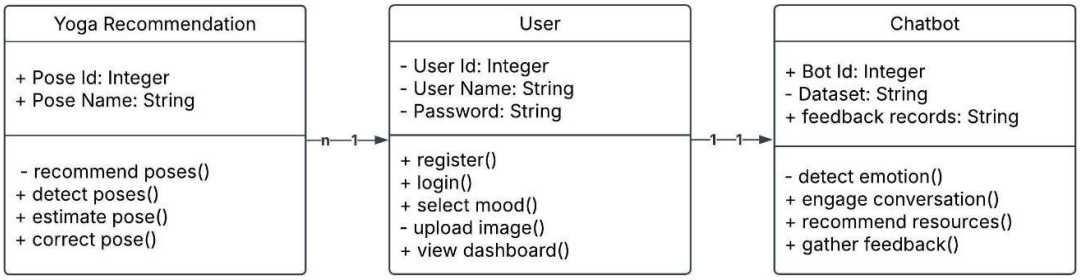

Fig 4.3.2 CLASS DIAGRAM

1. Classes and Their Attributes

- **Yoga Recommendation**
  - **Attributes**:
    - **Pose ID:** An integer uniquely identifying a yoga pose.
    - **Pose Name:** A string representing the name of the yoga pose.
  - **Methods**:
    - **recommend poses():** Suggests yoga poses to the user.
    - **detect pose():** Identifies the user's current pose.
    - **estimate pose():** Evaluates how close the user's pose is to the recommended one.
    - **correct pose():** Provides suggestions for improving the pose.
- **User**
  - **Attributes**:
    - **User ID:** An integer uniquely identifying a user.
    - **User Name:** A string representing the username.
    - **Password:** A string for user authentication.
  - **Methods**:
    - **register():** Allows a new user to create an account.

- **login():** Handles user authentication.
- **select mood():** Enables the user to choose their mood to customize recommendations.
- **upload image():** Allows the user to upload images for pose detection or tracking.
- **view dashboard():** Displays user-specific information, progress, or recommendations.

- **Chat Bot**
  - **Attributes**:
    - **Bot ID:** An integer uniquely identifying the chatbot.
    - **Dataset:** A string representing the dataset the chatbot uses.
    - **Feedback records:** A string containing user feedback data.
  - **Methods**:
    - **detect emotion():** Analyses the user's emotions based on text or input.
    - **engage conversation():** Facilitates interaction with the user.
    - **recommend resources():** Suggests additional resources based on the user's needs.
    - **gather feedback():** Collects user feedback for improvement.

2. Relationships

- **User and Yoga Recommendation:**
  - Cardinality: 1:n
  - Each user can access multiple yoga recommendations, but each recommendation is tied to a single user.
- **User and Chat Bot:**
  - Cardinality: 1:1
  - Each user interacts with a unique instance of the chatbot.

4.3.3. SEQUENCE DIAGRAM

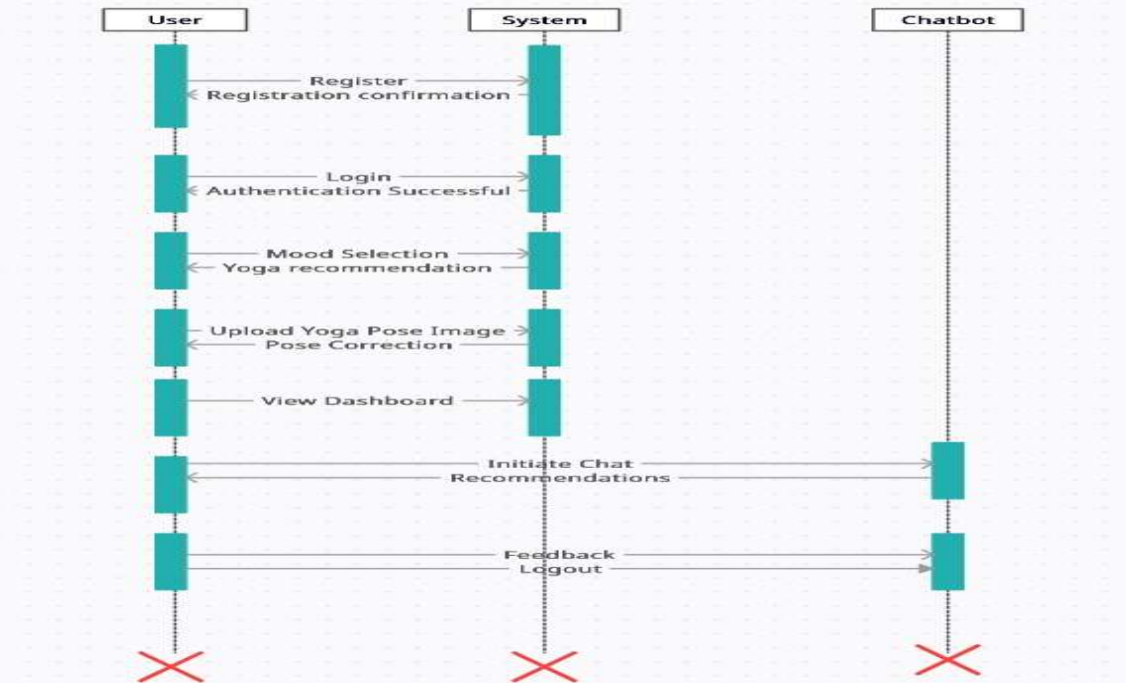

Fig 4.3.3 SEQUENCE DIAGRAM

The sequence diagram illustrates the interaction between the User, System, and Chatbot in a yoga-based mental well-being application. It represents the flow of actions from user initiation to system responses, ensuring smooth functionality.

Actors Involved

1. User – The individual interacting with the system.

2. System – The core application that processes requests and provides responses.

3. Chatbot – A virtual assistant providing recommendations and support.

Process Flow

1. User Registration:

- The User sends a Register request to the System.
- The System processes the request and responds with Registration Confirmation.

2. User Login:

- The User logs in by providing credentials.
- The System authenticates the credentials and sends an Authentication Successful response.

3. Mood Selection & Yoga Recommendations:

- The User selects a mood.
- The System provides Yoga Recommendations based on the selected mood.

4. Yoga Pose Upload & Correction:

- The User uploads an image of a yoga pose.
- The System analyses the image and provides Pose Correction feedback.

5. Dashboard View:

- The User accesses the dashboard to review insights and progress.

6. Chat Interaction:

- The User initiates a chat with the Chatbot.
- The System sends a request to the Chatbot, which provides recommendations.

7. Feedback & Logout:

- The User provides feedback.
- The User logs out, and the System processes the logout request.

Purpose of the Diagram

- This sequence diagram provides a visual representation of system interactions in a structured order.
- It helps developers understand the workflow and ensures seamless communication between the User, System, and Chatbot.
- It enhances the application's usability by clearly defining user actions and system responses.

4.3.4. ACTIVITY DIAGRAM

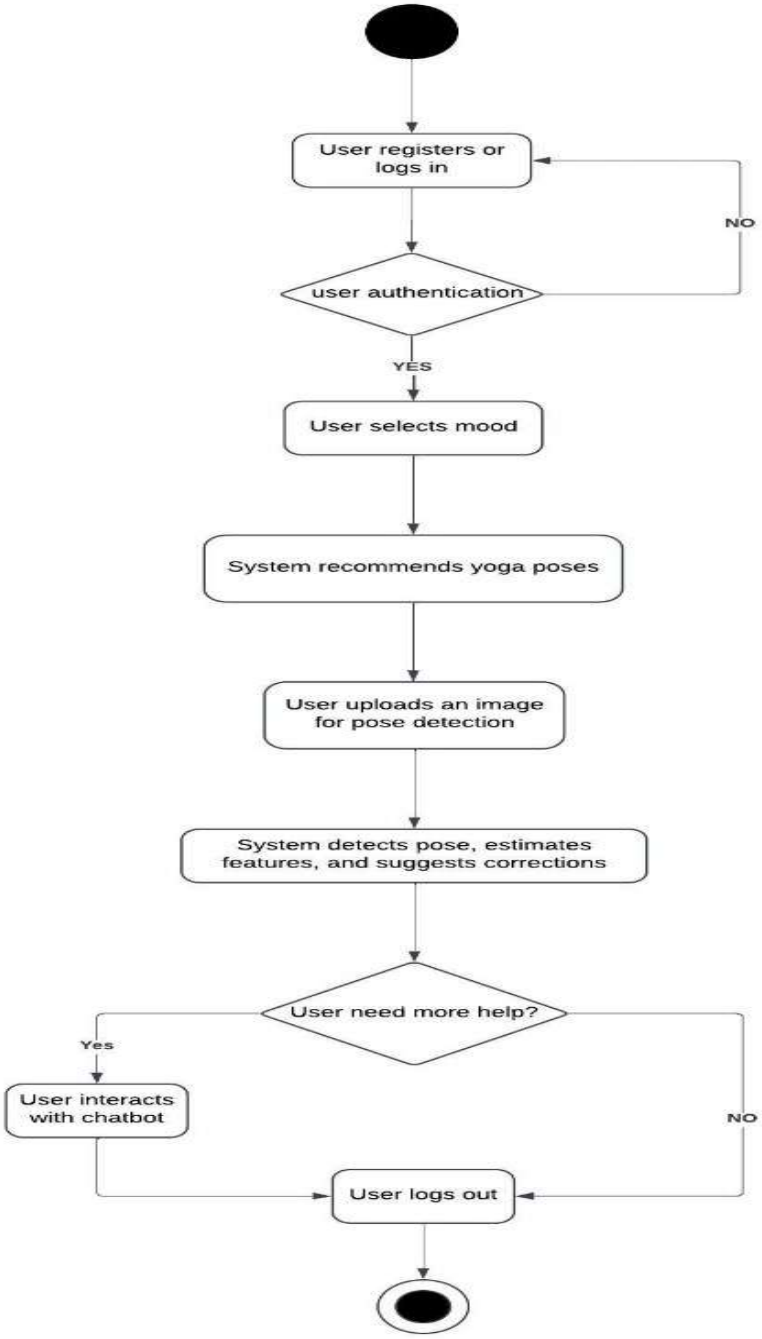

Fig 4.3.5. ACTIVITY DIAGRAM

The given image represents an Activity Diagram for a Yoga Pose Recommendation and Correction System. An activity diagram is a type of UML (Unified Modeling Language) diagram that visually represents the workflow of a system, illustrating the sequence of activities, decision points, and the overall flow of processes.

Explanation of the Activity Diagram

1. Start Node:

- The process begins with the user initiating the system, represented by the black start node.

2. User Registration or Login:

- The user must either register a new account or log in to an existing one.
- If authentication fails (decision point: "User authentication?"), the user is redirected back to the login process.
- If authentication is successful, the user proceeds to the next step.

3. User Selects Mood:

- Once logged in, the user selects their mood, which helps the system personalize yoga pose recommendations.

4. System Recommends Yoga Poses:

- Based on the selected mood, the system suggests appropriate yoga poses.

5. User Uploads an Image for Pose Detection:

- The user uploads an image of themselves performing the suggested yoga pose for analysis.

6. System Detects Pose and Suggests Corrections:

- The system processes the uploaded image, detects the user's pose, estimates relevant features, and provides corrective suggestions to improve posture.

7. Decision Point – "User Needs More Help?":

- If the user requires further assistance, they interact with a chatbot for additional guidance.
- If no additional help is needed, the user proceeds to the next step.

8. User Logs Out:

- The user logs out, marking the completion of the session.

9. End Node:

- The black end node signifies the termination of the process.

Key Features of This Activity Diagram

1. **Decision Points:** Represented by diamonds, they indicate different paths based on user input (e.g., authentication check, need for further assistance).
2. **Activities:** Represented by rounded rectangles, they indicate the actions performed by the user or system.
3. **Flow Arrows:** Show the sequence of actions and the logical flow between activities.
4. **Start and End Nodes:** Represent the beginning and termination of the process.

This activity diagram effectively models the user interaction within the Yoga Pose Recommendation and Correction System, making it easier to understand the sequence of activities and decision-making flow.

4.3.5. STATE CHART DIAGRAM

A State Chart Diagram (State Machine Diagram) represents the different states of an object in a system and the transitions between those states based on events. In this diagram, each rectangular box represents a state that a user goes through, and the arrows indicate transitions between states.

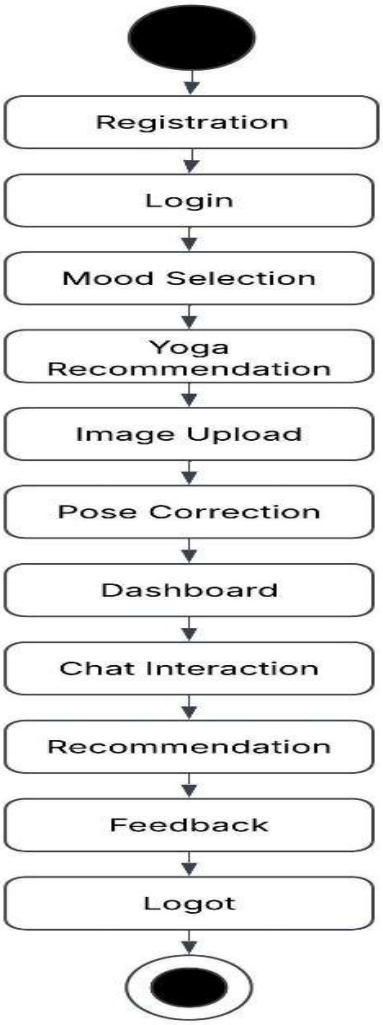

Fig 4.3.5. STATE CHART DIAGRAM

State Transitions in the Diagram

1. **Initial State**
    - The process starts with the initial state (black circle).
2. **User States & Transitions**
    - **Registration:** The user begins by registering.
    - **Login:** After registration, the user logs into the system.
    - **Mood Selection:** The system prompts the user to select their mood.
    - **Yoga Recommendation:** Based on the mood, the system recommends yoga poses.
    - **Image Upload:** The user uploads an image for pose analysis.
    - **Pose Correction:** The system analyses and corrects the user's pose.
    - **Dashboard:** The user views their progress and reports.
    - **Chat Interaction:** The user engages in chat-based assistance.
    - **Recommendation:** The system provides additional recommendations.
    - **Feedback:** The user provides feedback on the experience.
    - **Logout:** The user exits the system.
3. **Final State**
    - The black circle with an outer ring indicates the end state, meaning the session is complete.

4.3.6. DEPLOYMENT DIAGRAM

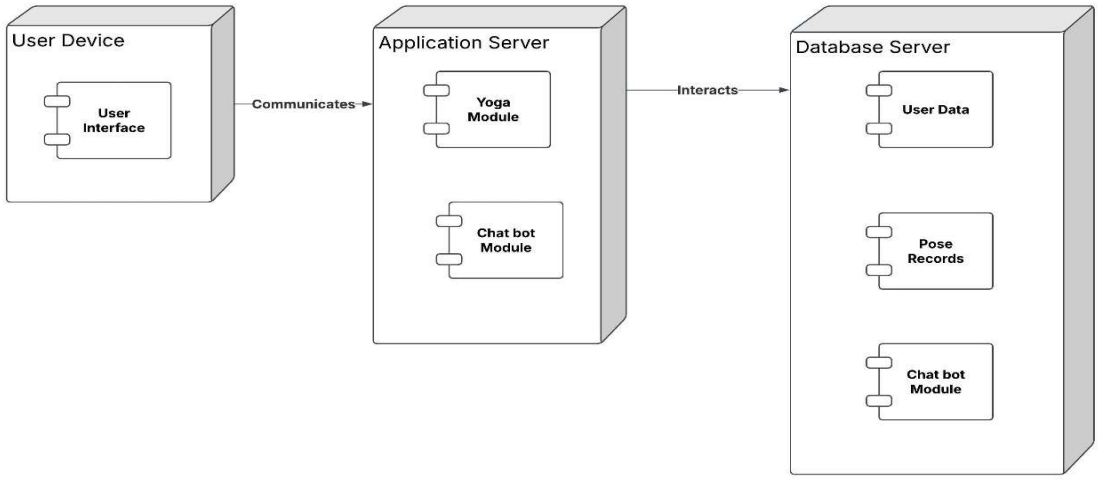

Fig 4.3.6. DEPLOYMENT DIAGRAM

This deployment diagram illustrates the physical deployment of the components in a system designed for a Yoga Recommendation application integrated with a chatbot. It shows the relationships between different hardware and software components and how they interact.

1. Nodes

a) User Device

- **Component**:
  - **User Interface**:
    - This is the front-end interface through which the user interacts with the system. It could be a mobile app, web app, or desktop application.
  - **Role**: Allows the user to input data (e.g., login, pose images, or mood selection) and receive outputs (yoga recommendations, chatbot responses).
  - **Interaction**: Communicates directly with the **Application Server**.

b) Application Server

- **Components**:
  - **Yoga Module**:
    - Processes user inputs to recommend yoga poses, analyse uploaded images, and provide pose correction.
  - **Chatbot Module**:
    - Facilitates conversations with users, detects emotions, and provides additional resources or feedback.
- **Role**: Acts as the intermediary between the **User Device** and the **Database Server**. It processes user requests and delivers results.
- **Interaction**:
  - Receives data from the **User Device**.
  - Interacts with the **Database Server** to retrieve and store data.

c) Database Server

- **Components**:
  - **User Data**: Stores user credentials, preferences, and historical data.
  - **Pose Records**: Stores details of yoga poses, including images, descriptions, and correction parameters.
  - **Chatbot Module**: Stores chatbot datasets, feedback records, and conversation logs.
- **Role**: Provides persistent storage for the system. The application server queries and

updates data in this server.

- **Interaction**: Responds to requests from the **Application Server**.

2. Relationships

- **User Device ↔ Application Server**:
  - The user interface on the device sends requests (e.g., login credentials, uploaded images) to the application server.
  - The application server processes the request and sends the appropriate response back (e.g., recommended poses, chatbot replies).
- **Application Server ↔ Database Server**:
  - The application server interacts with the database server to:
    - Retrieve user-specific data (e.g., preferences, past interactions).
    - Access pose records for recommendations.
    - Log chatbot interactions and gather feedback

3. Purpose of the System

This deployment architecture ensures:

- **Modularity**: Separation of components like the yoga module, chatbot module, and database for easier maintenance and scalability.
- **Interactivity**: Real-time communication between the user device and the backend for a smooth user experience.
- **Data Persistence**: Reliable storage of user, pose, and chatbot data on the database server for future use

4.3.7. COLLABORATION DIAGRAM

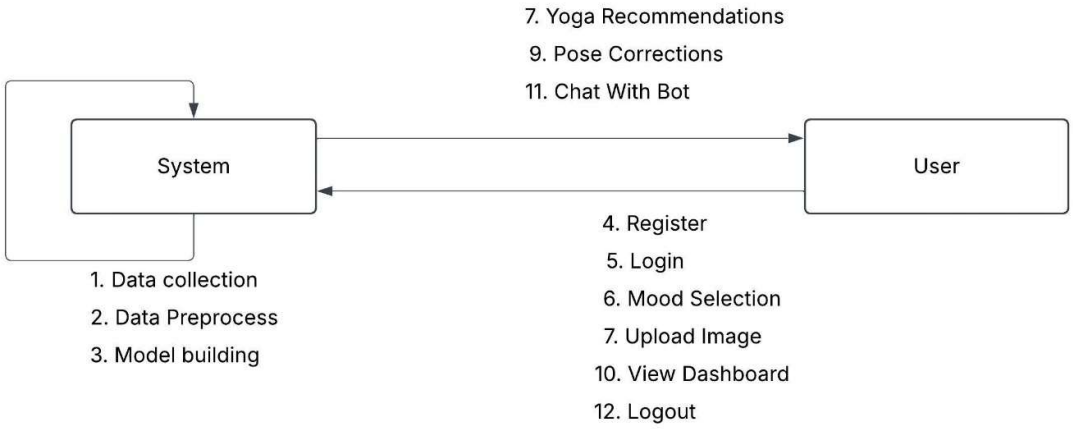

Fig 4.3.7. COLLABORATION DIAGRAM

1. System Actions:

- **Data collection:** The system collects data for training and analysis.
- **Data Preprocess:** The system cleans and prepares collected data.
- **Model building:** The system constructs a model for yoga recommendations and pose corrections.

2. User Actions:

- **Register:** The user registers with the application.
- **Login:** The user logs into the application.
- **Mood Selection:** The user chooses their current mood.
- **Upload Image:** The user uploads an image of their yoga posture.
- **View Dashboard:** The user can access a dashboard with information about their progress.
- **Logout:** The user exits the application.

3. Interactions:

- The user triggers actions like register, login, mood selection, upload image, view dashboard, and logout by interacting with the system.
- The system processes user requests and sends back responses in the form of yoga recommendations, pose corrections, or chat with bot functionality.

4.4 DATABASE DESIGN

4.4.1. ER DIAGRAMS:

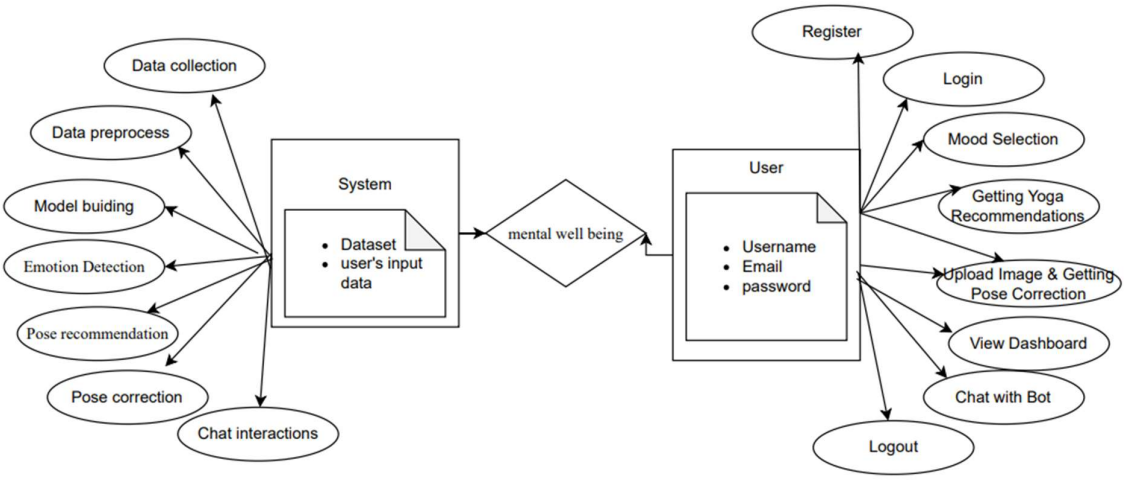

Fig 4.4.1 . ER DIAGRAM

Entity-Relationship (ER) Diagram Description

The ER diagram illustrates the interaction between the User and the System in a yoga-based mental well-being application. The diagram consists of two primary entities:

1. User

The User entity contains attributes such as:

- Username
- Email
- Password

The User interacts with the system through various actions, including:

- Registering and logging into the system.
- Selecting a mood to receive personalized yoga recommendations.
- Uploading images to receive pose correction feedback.
- Viewing the dashboard for insights and updates.
- Engaging in chat interactions with a bot.
- Logging out when finished.

2. System

The System processes the data and provides necessary feedback to the user. It manages:

- Dataset (used for model training and recommendations).
- User's input data (such as uploaded images and mood selections).

The System performs several key functions, including:

- Data collection and preprocessing for training and improving recommendations.
- Model building for yoga pose and emotion detection.
- Emotion detection to understand the user's mental state.
- Pose recommendation based on the user's mood and preferences.
- Pose correction by analyzing the uploaded image.
- Chat interactions to enhance user engagement.

➢ **USER**

The User is the primary actor who interacts with the system. The user performs various actions, such as:

- Registering and logging into the system.
- Selecting their mood to receive personalized yoga recommendations.
- Uploading images to get pose correction feedback.
- Viewing their progress on the dashboard.
- Interacting with a chatbot for additional assistance.
- Logging out of the system after completing their session.

➢ **SYSTEM**

The System is the secondary actor that responds to the user's actions. It handles:

- User authentication and account management.
- Providing yoga recommendations based on the user's mood.
- Analyzing uploaded yoga pose images and giving feedback.
- Managing dashboard data for user insights.
- Facilitating chatbot interactions for guidance and recommendations

The relationship between the User and the System is centered on mental well-being, where the system aids users in improving their yoga practice and emotional health through AI-driven insights and recommendations.

CHAPTER 5

SOFTWARE AND HARDWARE REQUIREMENTS

5. 1. SOFTWARE REQUIREMENTS

- Operating System : Windows 11
- Server-side Script : HTML, CSS, Bootstrap & JS
- Programming Language : Python
- Libraries : Flask, Pandas, MySql connector, os, Scikit-learn, numpy
- IDE/Workbench : PyCharm
- Technology : Python 3.6+
- Server Deployment : Xampp Server

5. 2. HARWARE REQUIREMENTS

- Processor - I3/Intel Processor
- Hard Disk - 160GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA
- RAM - 8GB

CHAPTER 6
MODULES

PROJECT MODULES

The project consists of two primary modules: the **System Module** and the **User Module**. Each module plays a crucial role in enhancing user experience and well-being through personalized interactions and recommendations.

6.1. SYSTEM MODULE

The **System Module** is designed to provide wellness support and enhance user engagement through two key submodules: **Yoga System Module** and **Chatbot Module**. These submodules work together to offer both physical and emotional well-being solutions to users by leveraging advanced technologies such as pose detection and natural language processing.

I. Yoga System Module

The **Yoga System Module** is a core component aimed at improving physical and mental well-being through personalized yoga recommendations. The module begins with **Mood Assessment**, where users input their current emotional state, allowing the system to analyze and recommend suitable yoga poses. Based on the selected mood, the **Yoga Pose Recommendation** feature suggests three yoga postures tailored to help the user achieve relaxation and balance.

To ensure users perform yoga poses correctly, the **Pose Detection & Estimation** feature enables users to upload images of their yoga postures, which the system analyzes to detect key features such as joint positions and angles. This is crucial for improving accuracy and ensuring users get the full benefits of their yoga practice. Additionally, the **Pose Correction** feature provides real-time feedback to users, guiding them on how to adjust their posture for better alignment, reducing the risk of injury, and enhancing the effectiveness of each pose.

II. Chatbot Module

The **Chatbot Module** serves as an emotional support system that engages users in meaningful conversations to help them navigate their emotions. It begins with **Emotion Detection**, where the chatbot analyzes user input to identify their emotional state. The chatbot is trained to recognize 59 distinct emotions, ensuring it can provide highly relevant and empathetic responses.

Once the chatbot identifies the user's emotional state, it utilizes **Conversational Flow** to interact in a natural and supportive manner. The chatbot's objective is to engage the user in a conversation that helps them process their emotions and find comfort. Additionally, the chatbot offers **Resource Recommendation**, suggesting relevant external content, such as YouTube videos, articles, or guided meditation exercises, to further support the user in managing their emotions. To continuously enhance its performance, the **Feedback Gathering** feature collects user input about their chatbot interactions, allowing the system to refine its responses and improve future conversations.

6.2. USER MODULE

The **User Module** is the gateway through which users interact with the system, providing a seamless and intuitive experience. It begins with **Register/Login**, allowing users to create an account and securely access system features. Once logged in, users can engage with various functionalities designed to improve their well-being.

A key feature of this module is **Mood Selection**, where users indicate their current emotional state. This serves as the foundation for **Yoga Recommendations**, enabling the system to suggest three poses specifically tailored to address their emotional and physical needs. Users who choose to perform these poses can utilize the **Upload Image** feature to submit photos of their postures for analysis. The system then processes these images using advanced pose estimation algorithms and provides **Pose Correction**, helping users refine their technique and maximize the benefits of each yoga pose.

To enhance user engagement and retention, the **Dashboard** feature compiles a history of past interactions, including selected moods, uploaded yoga pose images, and corrected postures. This allows users to track their progress over time and gain insights into their wellness journey. Additionally, users can interact with the **Chat with Bot** feature to receive emotional support and personalized recommendations. The chatbot serves as a companion that offers guidance, encouragement, and relevant resources based on user inputs.

Finally, the **Logout** feature ensures that users can securely exit the system after completing their activities. This function protects user data and maintains privacy by preventing unauthorized access to their personal wellness records. Overall, the User Module streamlines interaction with the system while ensuring a personalized and secure experience for every user.

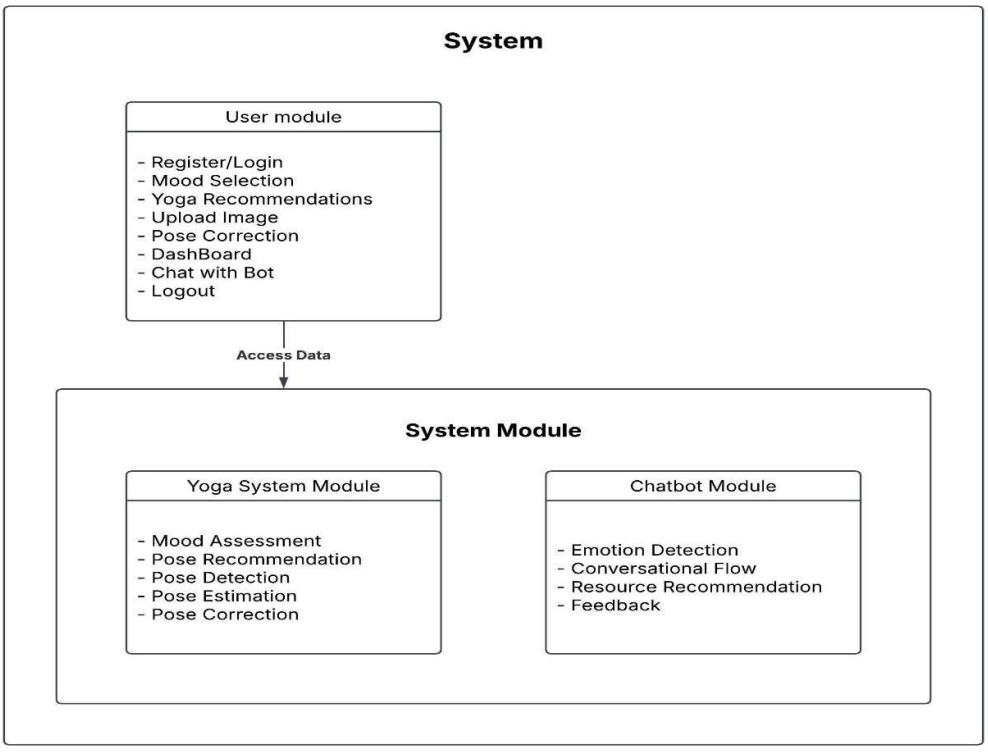

Fig 6.1. Modules Diagram

CHAPTER 7

IMPLEMENTATION

7. 1. USER MODULE

1. User Authentication Module

- **Purpose:** Handles user registration and login.
- **Technologies Used:** Flask, MySQL

```
@app.route('/register', methods=["GET", "POST"])
def register():
    if request.method == "POST":
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        c_password = request.form['c_password']

        if password == c_password:
            query = "INSERT INTO users (name, email, password) VALUES (%s, %s,  %s)"
            values = (name, email, password)
            executionquery(query, values)
            return render_template('login.html', message="Successfully Registered!")
        return render_template('register.html', message="Passwords do not match!")
    return render_template('register.html')
```

- **Key Functions and Methods**

- **User Registration**: Stores user details in MySQL.
- **User Login**: Validates credentials and creates a session.

- **Database Table: Users (Stores user credentials)**

Field Name	Data Type	Description
`id`	INT (Primary Key)	Auto-incremented user ID
`name`	VARCHAR(255)	Stores user's full name
`email`	VARCHAR(255)	Stores user's email (unique)
`password`	VARCHAR(255)	Hashed password for security

Table 7.1 User database

2. User Authentication and Session Management (Flask)

CODE:

```
@app.route('/register', methods=["GET", "POST"])
def register():
    if request.method == "POST":
        name = request.form['name']
        email = request.form['email']
        password = request.form['password']
        c_password = request.form['c_password']

        if password == c_password:
            query = "SELECT email FROM users WHERE email = %s"
            existing_data = retrivequery1(query, (email,))
            if not existing_data:
                query = "INSERT INTO users (name, email, password) VALUES (%s, %s,
%s)"
                executionquery(query, (name, email, password))
                return render_template('login.html', message="Successfully Registered!")
            return render_template('register.html', message="This email ID already exists!")
        return render_template('register.html', message="Passwords do not match!")
    return render_template('register.html')
```

- **Functionality**: Manages user registration, login, and session management. Provides a secure system for user authentication.

- **Key Concepts**:
  - **Flask Sessions**: Used to manage user data during their visit.
  - **Hashing Passwords**: In a real-world scenario, user passwords should be hashed using a hashing algorithm (e.g., bcrypt, hashlib).

3. Dashboard Module

- **Purpose:** Displays user history of yoga sessions, images, and feedback.
- **Technologies Used:** Flask, MySQL
- **Key Functions and Methods**
  - Filters Data by Date Range
  - Displays Uploaded and Corrected Images

CODE:

```
@app.route('/dashboard', methods=["POST","GET"])
def dashboard():
    user_email = session["user_email"]
    query = "SELECT * FROM dashboard WHERE email = %s"
    values = (user_email,)
    dashboard_data = retrivequery1(query, values)
    dashboard_list = [
        {
            'user_name': item[1],
            'mood_name': item[3],
            'yoga_name': item[4],
            'feedback': item[7],
            'date': item[8]
        }
        for item in dashboard_data]
    return render_template('dashboard.html', dashboard_data = dashboard_list)
```

7.2. SYSTEM MODULE

I. YOGA MODULE:

1. Yoga Pose Recommendation Module

```
def recommend_poses(mood, u, sigma, vt, top_n=3):
    mood_idx = mood_to_idx[mood]
    mood_latent_features = np.dot(u[mood_idx, :], sigma)
    pose_scores = np.dot(mood_latent_features, vt)

    top_pose_indices = np.argsort(pose_scores)[::-1][:top_n]
    recommended_poses = [poses[idx] for idx in top_pose_indices]
    return recommended_poses
```

- **Purpose:** Suggests yoga poses based on user mood using Singular Value Decomposition (SVD).
- **Technologies Used:** Pandas, NumPy, SciPy
- **Database Table: yoga_recommendation (stores mood and recommended poses)**

Field Name	Data Type	Description
id	INT (Primary Key)	Auto-incremented ID
mood	VARCHAR(255)	User's mood before practice
pose	VARCHAR(255)	Recommended yoga pose

Table 7.2 Yoga Recommendation

- **Key Functions and Methods**
  - **Matrix Factorization:** Uses SVD for pose recommendation**.**
  - **Mood-based Recommendation:** Maps mood to suitable poses.

2. Yoga Posture Correction Module

- **Purpose:** Uses Mediapipe and OpenCV to analyze user posture and provide corrections.
- **Technologies Used:** OpenCV, Mediapipe, NumPy
- **Database Table: dashboard**

Field Name	Data Type	Description
Id	INT (Primary Key)	Auto-incremented ID
Name	VARCHAR(255)	User's name
Email	VARCHAR(255)	User's email
yoga_pose	VARCHAR(255)	Name of yoga pose
uploaded_img	BLOB	User-uploaded image
corrected_img	BLOB	Annotated image
Feedback	TEXT	Pose correction feedback
Date	DATE	Timestamp of submission

Table 7.3 Dashboard table

- **Key Functions and Methods**
  - **Angle Calculation:** Determines body joint angles.
  - **Correction Feedback:** Compares detected angles to ideal pose angles.

CODE:

```
def calculate_angle(a, b, c):
    a = np.array(a)
    b = np.array(b)
    c = np.array(c)
    ba = a - b
    bc = c - b
    cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
    angle = np.arccos(np.clip(cosine_angle, -1.0, 1.0))
    return np.degrees(angle)
```

3. Yoga Pose Recommendation using SVD

CODE:

```
import pandas as pd
import numpy as np
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import svds

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

# Extract unique moods and yoga poses
moods = df['Mood Before'].unique()
poses = df['Yoga Practice'].unique()

# Create dictionaries to map moods and poses to indices
mood_to_idx = {mood: idx for idx, mood in enumerate(moods)}
pose_to_idx = {pose: idx for idx, pose in enumerate(poses)}

# Initialize an interaction matrix
interaction_matrix = np.zeros((len(mood_to_idx), len(pose_to_idx)), dtype=int)

# Populate the interaction matrix
for _, row in df.iterrows():
    mood_idx = mood_to_idx[row['Mood Before']]
    pose_idx = pose_to_idx[row['Yoga Practice']]
    interaction_matrix[mood_idx, pose_idx] += 1

# Convert to sparse matrix for SVD
interaction_matrix_sparse = csr_matrix(interaction_matrix, dtype=np.float32)

# Perform Singular Value Decomposition (SVD)
num_features = min(interaction_matrix_sparse.shape) - 1  # Set a suitable number of features
u, sigma, vt = svds(interaction_matrix_sparse, k=num_features)

# Convert sigma into a diagonal matrix
sigma_diag_matrix = np.diag(sigma)

# Function to recommend yoga poses based on mood
def recommend_poses(mood, u, sigma, vt, top_n=3):
    if mood not in mood_to_idx:
        return f"Error: Mood '{mood}' not found in dataset."

    mood_idx = mood_to_idx[mood]
    mood_latent_features = np.dot(u[mood_idx, :], sigma)
    pose_scores = np.dot(mood_latent_features, vt)

    # Get top N recommended poses
    top_pose_indices = np.argsort(pose_scores)[::-1][:top_n]
```

```
    recommended_poses = [poses[idx] for idx in top_pose_indices]

    return recommended_poses

# Example: Get top 3 yoga poses for an "Angry" mood
mood_input = "Angry"
recommended_poses = recommend_poses(mood_input, u, sigma_diag_matrix, vt, top_n=3)

# Print recommendations
print(f"Recommended yoga poses for mood '{mood_input}': {recommended_poses}")
```

Functionality: Recommends yoga poses based on a user's mood using Singular Value Decomposition (SVD) for collaborative filtering.

Key Concepts:

- **SVD** (Singular Value Decomposition) is a matrix factorization technique that is used to find latent features in a user-item interaction matrix.
- **Interaction Matrix**: A matrix representing the interactions between moods and yoga poses.

Classes/Functions:

- **Data Preprocessing**:
    - Reads the dataset (Recommendation_yoga_data.csv)
    - Converts the data into an interaction matrix, mapping moods to yoga poses
    - Applies SVD for dimensionality reduction to extract latent features
- **Recommendation**:
    - recommend_poses(mood, u, sigma, vt, top_n=3):
        - **Input**: mood (string), u, sigma, vt (SVD components), top_n (int)
        - **Output**: List of recommended yoga poses for the specified mood.

4. Pose Estimation

This module involves using the PoseNet model from TensorFlow for detecting human keypoints from images.

- **Class:** PoseNetModel
- **Purpose:** Loads the PoseNet model and extracts keypoints for body poses.
- **Key Methods:**
  - **load_posenet_model():**
  - **Input:** None
  - **Output:** Loaded PoseNet model.
- **Description:** Loads the pre-trained PoseNet model from TensorFlow Hub.
- **extract_keypoints_using_posenet(image_tensor, model):**

☞ **Input:**

- **image_tensor:** A tensor representing the preprocessed image.
- **model:** The loaded PoseNet model.

☞ **Output:** A NumPy array representing the keypoints of the human body detected in the image.

CODE:

```
import tensorflow as tf
import tensorflow_hub as hub

def load_posenet_model():
    model_url = 'https://tfhub.dev/google/movenet/singlepose/lightning/4'
    model = hub.load(model_url)
    return model

def extract_keypoints_using_posenet(image_tensor, model):
    image_tensor = tf.image.resize(image_tensor, (192, 192))  # Resize image
    image_tensor = image_tensor / 255.0  # Normalize to [0, 1]
    image_tensor = tf.expand_dims(image_tensor, axis=0)  # Add batch dimension
    results = model.signatures['serving_default'](image_tensor)
    keypoints = results['output_0'].numpy()[0]  # Extract keypoints
    return keypoints
```

- **Description:** This method resizes the input image, normalizes it, and extracts keypoints using the PoseNet model.
- **Technologies Used:**
- **Language:** Python
- **Libraries:** TensorFlow, TensorFlow Hub, NumPy, PIL (Pillow)

5. Object Detection with YOLO

This module uses the YOLO model for detecting and cropping a person from the input image.

CODE:

```
import cv2
import numpy as np
def load_yolo_model():
    net = cv2.dnn.readNetFromDarknet('yolov3.cfg', 'yolov3.weights')
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers().flatten()]
    return net, output_layers
def detect_and_crop_person(image_path, net, output_layers):
    image = cv2.imread(image_path)
    H, W = image.shape[:2]
    blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    layer_outputs = net.forward(output_layers)
    for output in layer_outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.5 and class_id == 0:  # Person class
                box = detection[0:4] * np.array([W, H, W, H])
```

```
                (centerX, centerY, width, height) = box.astype("int")
                x = int(centerX - (width / 2))
                y = int(centerY - (height / 2))
                cropped_image = image[y:y+height, x:x+width]
                return cropped_image
        return None
```

- **Class: YOLOModel**
- **Purpose:** Detects and crops the person from the input image using the YOLO object detection model.
- **Key Methods:**
  1. **load_yolo_model():**
     - **Input:** None
     - **Output:** Loaded YOLO model and output layers.
     - **Description:** Loads the YOLOv3 model and prepares it for use in detection.
  2. **detect_and_crop_person(image_path, net, output_layers):**
     - **Input:**
       - **image_path:** Path to the input image.
       - **net:** The loaded YOLO network.
       - **output_layers:** The layers of the network where object detection outputs are available.
     - **Output:** Cropped image of the person if detected, None if no person is detected.
- **Description:** Detects objects in the image and crops the area where the person is detected.
- **Technologies Used:**
  - **Language:** Python
  - **Libraries:** OpenCV, NumPy

6. Yoga Pose Feedback System

CODE:

```
import math
def calculate_angle(landmark1, landmark2, landmark3):
    x1, y1, _ = landmark1
    x2, y2, _ = landmark2
    x3, y3, _ = landmark3
    angle = math.degrees(math.atan2(y3 - y2, x3 - x2) - math.atan2(y1 - y2, x1 - x2))
    if angle < 0:
        angle += 360
    return angle

POSE_ANGLES_KEYPOINTS = {
    'Parighasana': [(5, 1, 6), (11, 12, 24)],
}

def provide_pose_correction_feedback_using_angles(user_keypoints, reference_keypoints, intended_pose_name):
    feedback = []
    angle_keypoints_indices = POSE_ANGLES_KEYPOINTS.get(intended_pose_name, [])
    for indices in angle_keypoints_indices:
        user_kp1, user_kp2, user_kp3 = user_keypoints[indices[0]][0][:2], user_keypoints[indices[1]][0][:2], user_keypoints[indices[2]][0][:2]
        ref_kp1, ref_kp2, ref_kp3 = reference_keypoints[indices[0]][0][:2], reference_keypoints[indices[1]][0][:2], reference_keypoints[indices[2]][0][:2]

        user_angle = calculate_angle(user_kp1, user_kp2, user_kp3)
        ref_angle = calculate_angle(ref_kp1, ref_kp2, ref_kp3)

        feedback.append(f"Pose: {intended_pose_name}, Keypoints: {indices}, User angle: {user_angle:.2f}, Reference angle: {ref_angle:.2f}, Difference:
```

```
{abs(user_angle - ref_angle):.2f} degrees.")

    return feedback
```

This module calculates the angles between keypoints and provides feedback based on the comparison between user and reference poses.

- **Class: PoseFeedback**
- **Purpose:** Compares angles of detected keypoints with reference keypoints and gives feedback for pose correction.

- **Key Methods:**
    1. calculate_angle(landmark1, landmark2, landmark3):
        - **Input:**
            - landmark1, landmark2, landmark3: Three keypoints (x, y) coordinates.
        - **Output:** The calculated angle between the three keypoints.
        - **Description:** Uses the law of cosines to calculate the angle formed by three keypoints.
    2. provide_pose_correction_feedback_using_angles(user_keypoints, reference_keypoints, intended_pose_name):
        - **Input:**
            - user_keypoints: Keypoints detected from the user image.
            - reference_keypoints: Reference keypoints for the intended pose.
            - intended_pose_name: Name of the pose being evaluated.
        - **Output:** Feedback for the user with angle differences and corrections.
        - **Description:** Compares angles between user and reference poses and generates feedback based on differences.

Technologies Used:

- **Language:** Python
- **Libraries:** NumPy, math

II.CHATBOT MODULE

This section outlines the implementation of the Emotion Chatbot, covering the necessary files, classes, methods, technologies used, and their functionality. We'll also describe important concepts, syntax, and provide sample code snippets.

Technologies Used:

- Frontend:
    - HTML
    - CSS
    - JavaScript (for chatbot logic, DOM manipulation, and interaction)
    - YouTube API (for video suggestions)
- Backend (Optional for future enhancement):
    - Node.js or Python (Flask/Django) for server-side handling
    - Database (Optional for storing user data or conversation logs):
    - MySQL / MongoDB (if a database is required for persistent storage)

- **Files Structure:**
    - **index.html** - Basic structure for the chatbot interface.
    - **styles.css** - Styling for the chatbot interface.
    - **chatbot.js** - JavaScript code for chatbot logic and interaction.
    - **videoHandler.js** - Code (optional) for handling and suggesting videos from YouTube (in case backend interaction is implemented).

Chatbot Class Breakdown:

1. EmotionData Class (or Object)

- Purpose: Stores data about various emotions, including names, follow-up questions, comfort messages, videos, and feedback.

- Methods/Functionality:
  - This object is initialized with predefined emotions (like happy, embarrassed, ashamed) and their associated content such as reasons, comfort messages, and feedback.
  - Each emotion has a list of possible reasons (health, work, etc.) and feedback messages based on the user input.

CODE:

```
const emotionsData = {
  "emotions": [
    {
      "name": "happy",
      "follow_up_question": "What made you feel happy?",
      "reasons": ["Health", "Work", "Relationships", "Other"],
      "comfort": "That's amazing! Here are videos to keep your happiness flowing:",
      "videos": [
        { "title": "The Science of Happiness - Ted Talk", "link":
"https://www.youtube.com/watch?v=fLJsdqxnZb0" },
        { "title": "10 Simple Habits for a Happier Life", "link":
"https://www.youtube.com/watch?v=78nsxRxbf4w" }
      ],
      "feedback_question": "How do you feel after exploring your happy?",
      "feedback_positive": "That's great! Embracing your happy is a step forward.
🌟",
      "feedback_negative": "I'm here for you. Let’s find a way to navigate through this
```

happy together."}]};

- **Input**: Emotion names and reasons (e.g., "happy", "work").
- **Output**: A structured response containing follow-up questions, comfort messages, videos, and feedback prompts.

2. Chatbot Class

- **Purpose**: Controls the flow of the conversation with the user.
- **Methods/Functionality**:
    - **startChat()**: Initiates the conversation by sending a welcome message and asking for the user's name.
    - **processInput()**: Handles the user's input based on the current step in the conversation (e.g., name input, emotion selection, etc.).
    - **sendBotMessage()**: Sends a message to the chat window.
    - **generateButtons()**: Generates interactive buttons for the user to select an emotion, reason, or feedback.

CODE:

```
class Chatbot {
  constructor() {
    this.currentStep = 0;
    this.selectedEmotion = null;
    this.userName = "";
  }

  startChat() {
    this.sendBotMessage("Welcome to the Emotion Chatbot! Please type anything to
begin.");
    this.enableUserInput();
  }

  enableUserInput() {
    // Enable the text input and send button
  }

  disableUserInput() {
```

```
        // Disable the text input and send button
    }

    handleUserInput(input) {
        if (this.currentStep === 0) {
            this.userName = input;
            this.currentStep++;
            this.sendBotMessage(`Hi, ${this.userName}! How are you feeling today?`);
            this.displayEmotionOptions();
        } else if (this.currentStep === 1) {
            const emotion = emotionsData.emotions.find(em => em.name.toLowerCase() === input.toLowerCase());
            if (emotion) {
                this.selectedEmotion = emotion;
                this.currentStep++;
                this.sendBotMessage(emotion.follow_up_question, this.generateButtons(emotion.reasons, "reason"));
            } else {
                this.sendBotMessage("I'm sorry, I didn't recognize that emotion. Please select from the options below.", this.generateButtons(emotionsData.emotions.map(e => e.name), "emotion"));
            }
        }
        // More steps follow as described in the original code
    }

    sendBotMessage(message, options = null) {
        // Sends a bot message to the chat window
    }

    generateButtons(options, type) {
        return options.map(option => `<button class="option-button" data-type="${type}" data-value="${option}">${option}</button>`).join(" ");
```

```
    }
}
```

1. **Methods**:
    - Functions designed to perform specific actions, such as sending a message or displaying buttons.
2. **Classes**:

- Blueprints for objects that group methods and properties. Each emotion could be an object inside an emotion class.
- Example: Chatbot, EmotionData, VideoHandler.

3. **DOM Manipulation**:

- Using JavaScript to interact with and update the HTML structure dynamically, e.g., sending messages, enabling buttons, etc.

4. **Event Listeners**:

- Capture user actions such as button clicks or keyboard presses and respond accordingly.

Description:

This implementation covers the core chatbot functionality, including interaction with the user, handling emotions and reasons, providing comfort, and suggesting videos. The conversation flow is controlled by methods in the Chatbot class, and the emotions data is encapsulated in a structured format. If a database is used, the tables can store emotions and videos, and stored procedures can help retrieve data dynamically.

CHAPTER 8
TESTING

1. Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

2. Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.
The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

3. Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

4. White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5. Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

TEST CASES:

Testcase ID	Input	Output	Result
01	Input	Tested for different model given by user on the different model.	Success
02	Model	Tested for different input given by the user on different models are created using the different algorithms and data.	Success
03	Prediction	Prediction will be performed using the different models build from the algorithms.	Success

Table 8. 1. TESTCASES

TEST CASES MODEL BUILDING:

S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Read the datasets.	Dataset's path.	Datasets need to read successfully.	Datasets fetched successfully.	It produced P. If this not F will come
2	Registration	Valid username, email, password.	Verify that the registration form accepts valid user inputs and successfully creates a new account.	User is successfully registered, and an account is created	It produced P. If this is not, it will undergo F.
3	Login	Valid username and password	Verify that users can log in with valid credentials	User is successfully logged in and redirected to the dashboard	It produced P. If this is not, it will undergo F.
4	Yoga recommendation	Input current mood	Output as 3 recommended yoga poses	Output as 3 recommended yoga poses	It produced P. If this is not, it will undergo F
5	Pose prediction	Uploaded yoga image	Yoga pose identification	Yoga pose identification	It produced P. If this is not, it will undergo F
6	Pose correction	Uploaded yoga image	Yoga pose corrected image	Yoga pose corrected image	It produced P. If this is not, it will undergo F

Fig 8.2. Testcase Model building

CHAPTER 9

OUTPUT SCREENS

Index Page: This is the landing page of our website.

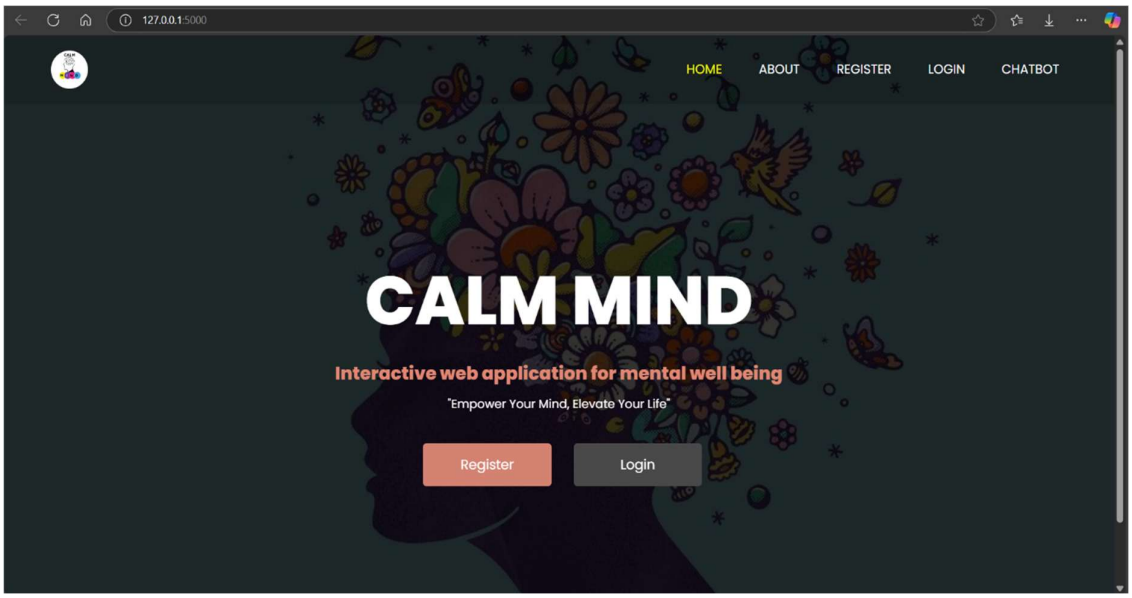

Fig 9.1 Index page

Registration Page: This is Registration page. In here, user can register with their credentials such as email, username, password.

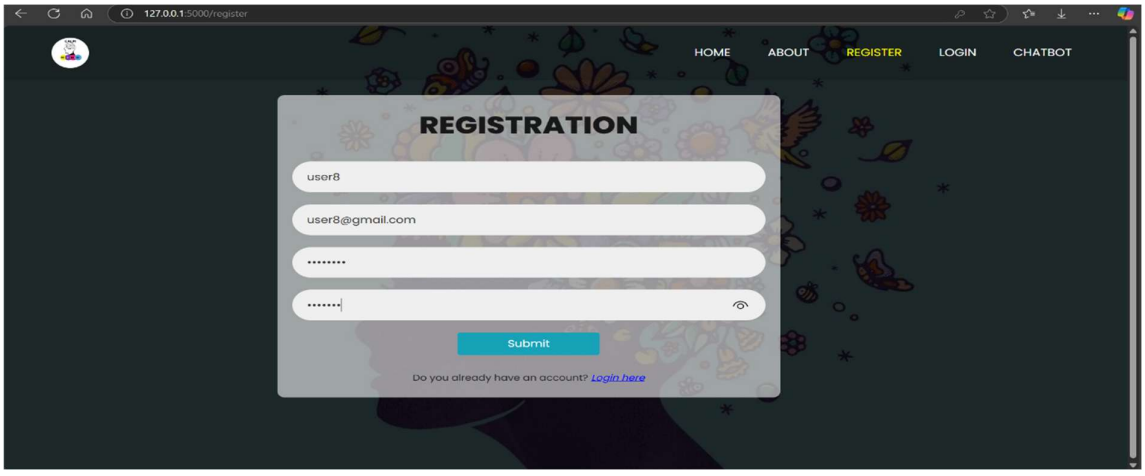

Fig 9.2 Registration Page

Login Page: This is login page. In here user can login with their registered credentials such as email, password.

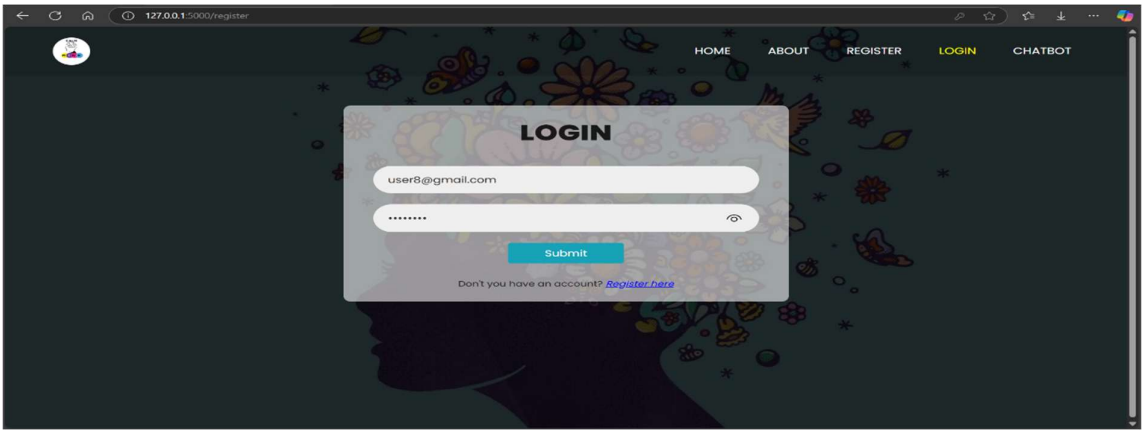

Fig 9.3 Login Page

Mood selection Page: In here user can select their current mood to get yoga recommendation

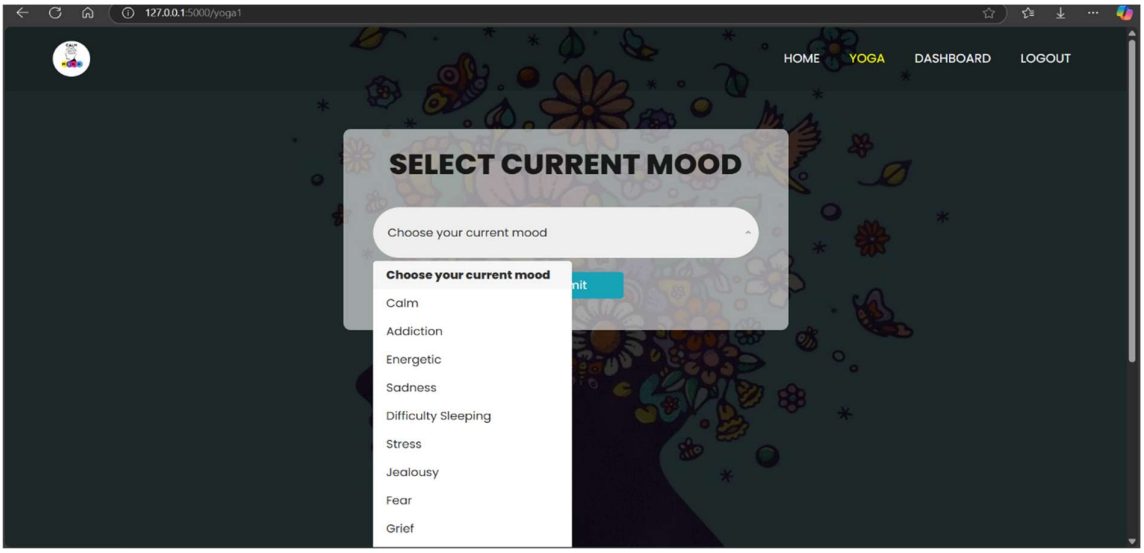

Fig 9.4 Mood selection page

Yoga Recommendation Page: In here user can get yoga recommendation for their selected mood.

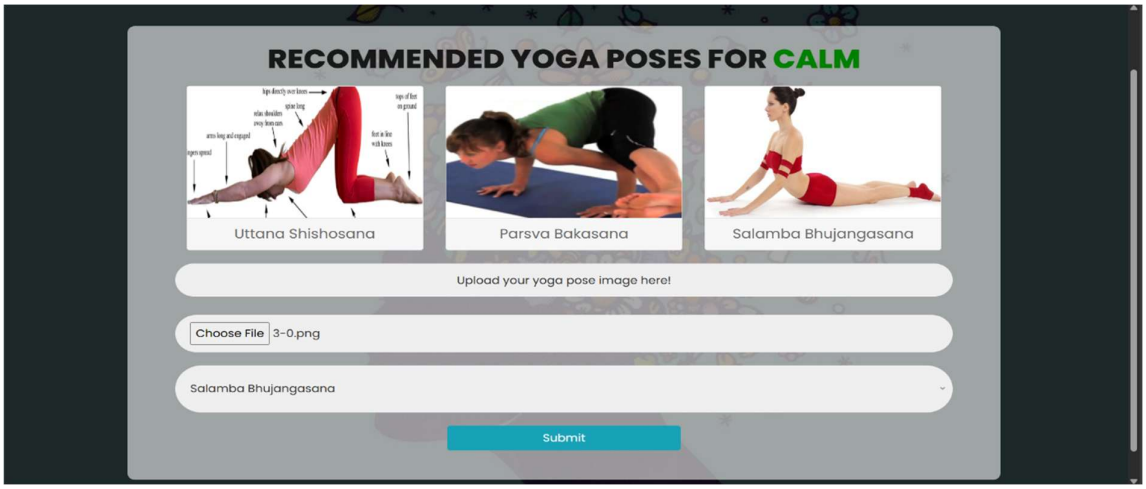

Fig 9.5 Yoga recommendation page

Correction result page: In here user can get the feedback correction.

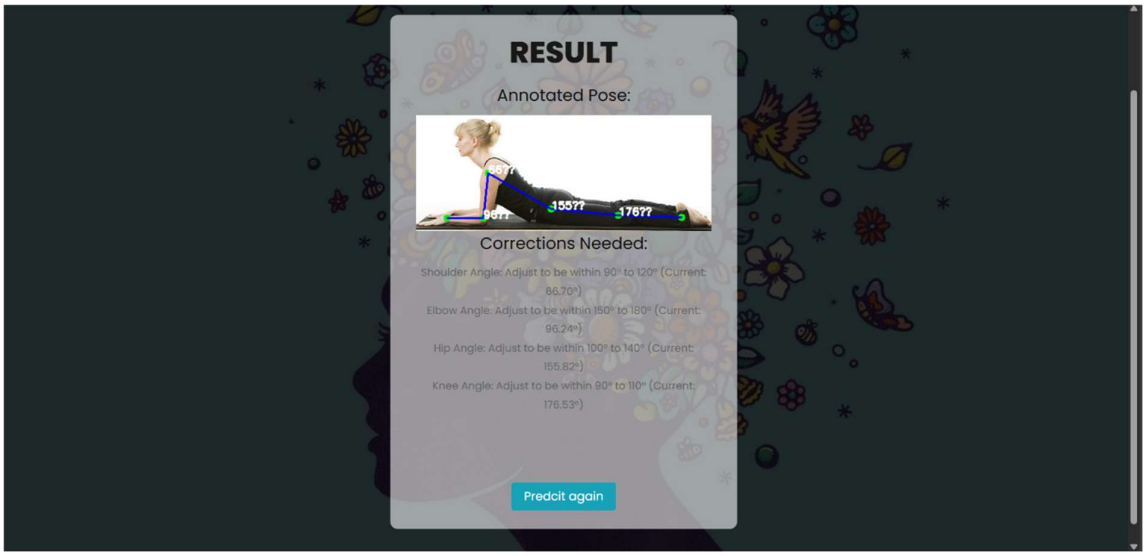

Fig 9.6 Correction result page

Dashboard Page: In here user can retrieve their own history data by date and all data he can view.

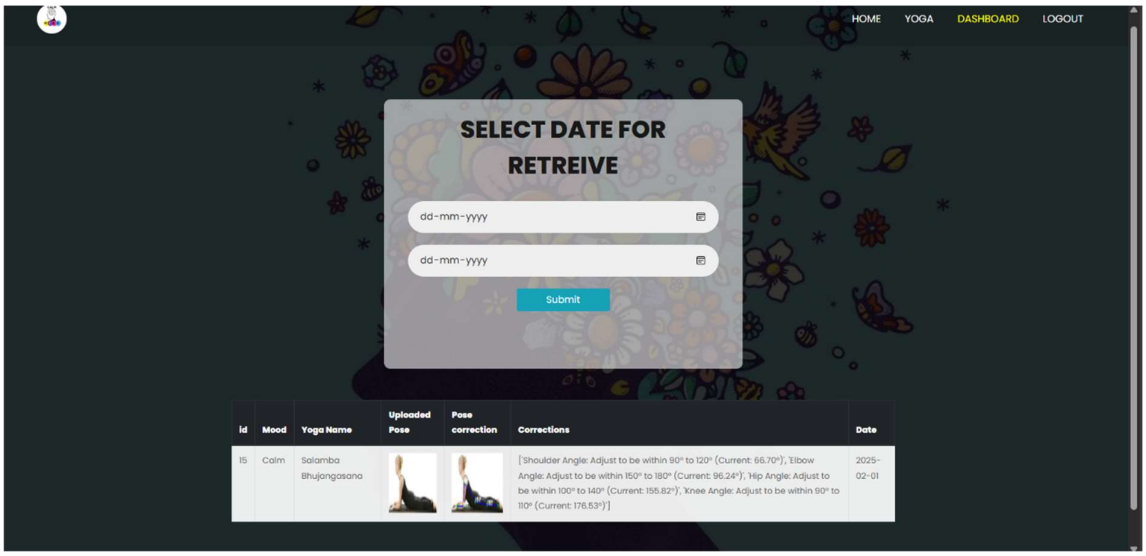

Fig 9.7 Dashboard Page

About Page: This is about section which contains information about our project

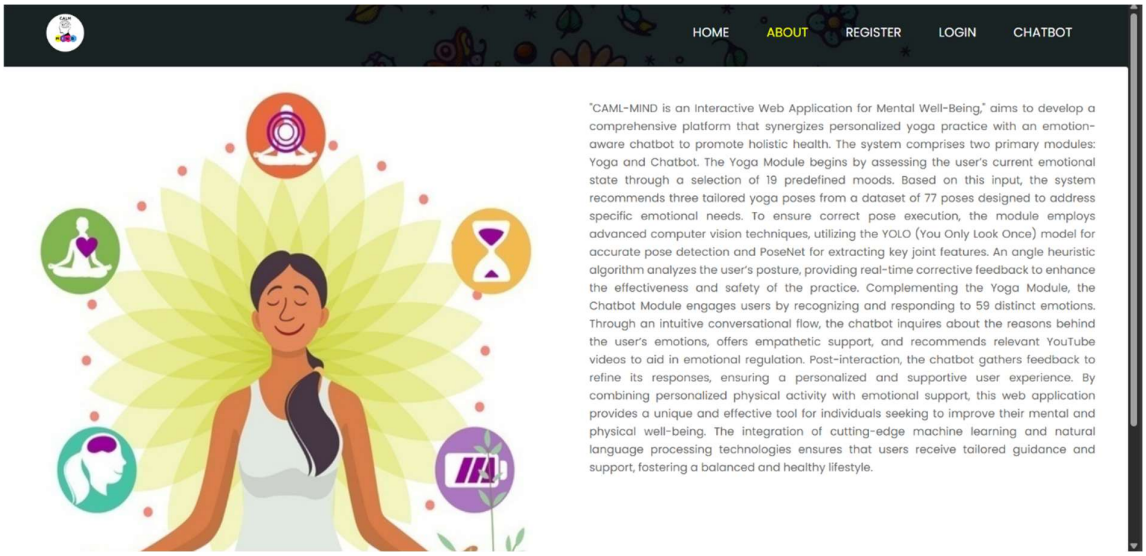

Fig 9.8 About Page

Chatbot Page: In here user can interact with chatbot.

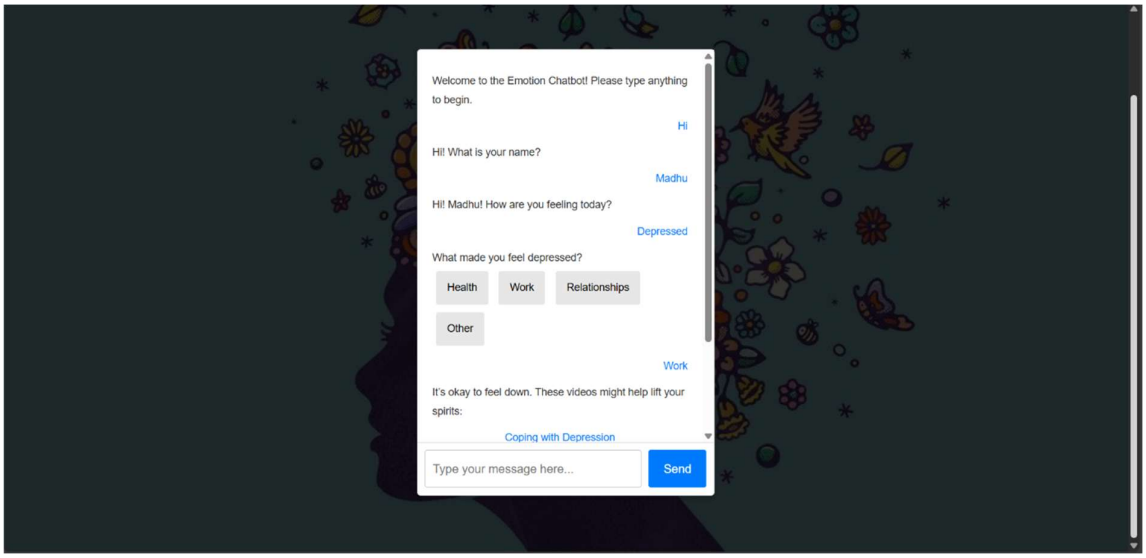

Fig 9.9 Chatbot interaction(1)

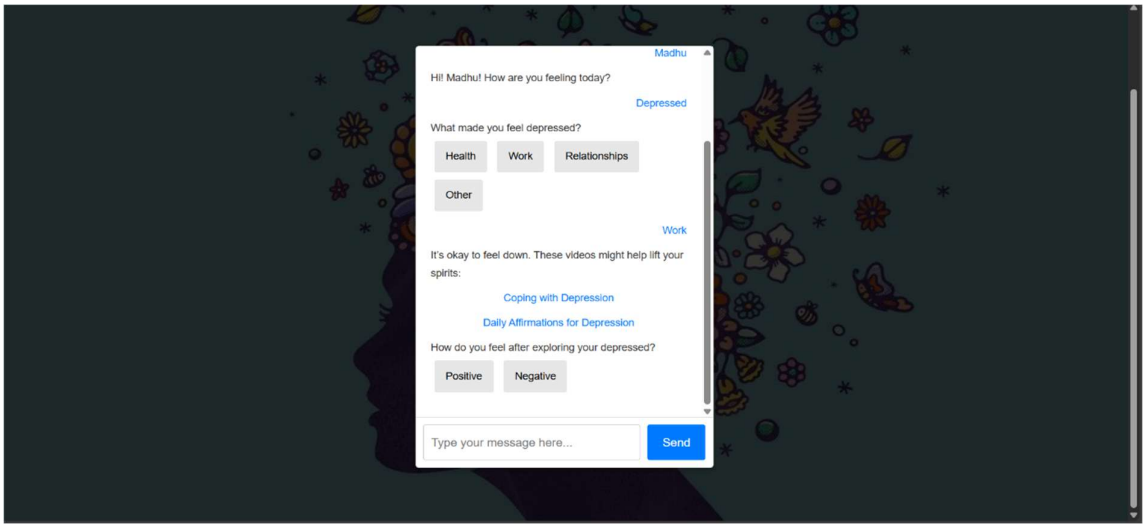

Fig 9.10 Chatbot interaction(2)

Fig 9.11 Video Recommendation

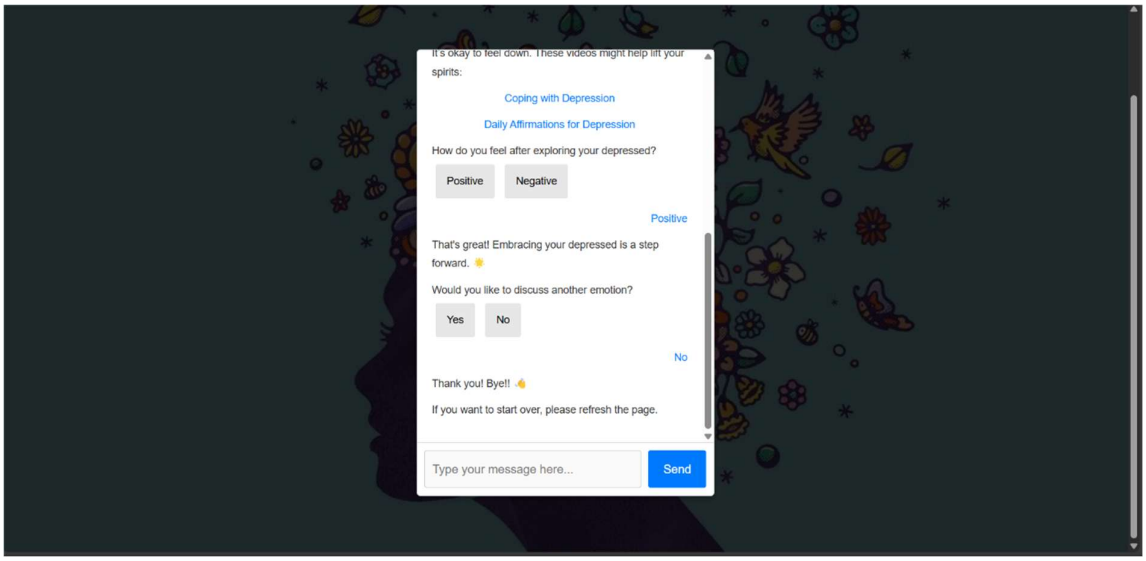

Fig 9.12 Chatbot interaction(3)

CHAPTER 10

DEPLOYMENT

The below is the step by step guide to deploy the Project:

I. INSTALL PYTHON

- Download and install Python from official Python website.
- Ensure Python is added to your system's PATH during installation.
- Optionally, install **Anaconda** for managing Python packages and environments easily.

II. Install XAMPP:

1. Download XAMPP:

- Visit the official XAMPP website: https://www.apachefriends.org/index.html.
- Choose the appropriate version for your operating system (Windows, Linux, or macOS).
- Click the "Download" button to begin downloading the XAMPP installer.

2. Install XAMPP on Windows:

- **Run the Installer:**
  - Once the XAMPP installer file is downloaded, double-click on it to run.
- **Choose Installation Language:**
  - Select the language (usually English) and click "OK."
- **Select Components to Install:**
  - The installer will ask you to select the components to install. By default, Apache, MySQL, PHP, phpMyAdmin, and others are selected. You can leave them selected if you want the full stack. Click Next.
- **Choose Installation Directory:**
  - Choose the directory where you want to install XAMPP. The default is C:\xampp. Click Next.
- **Start Menu Folder:**
  - Choose a folder in the Start Menu for XAMPP shortcuts. You can leave this as default or change it. Click Next.
- **Ready to Install:**
  - Click Install to start the installation process. This might take a few minutes.

3. Launch XAMPP:

- **Start the XAMPP Control Panel:**
  - Once the installation is finished, you can launch the XAMPP Control Panel.
  - The installer will ask if you want to start the XAMPP Control Panel immediately. Select Finish.
- **Start Apache and MySQL:**
  - Open the XAMPP Control Panel.
  - In the Control Panel, click Start next to Apache and MySQL to start the web server and database server.
  - After Apache and MySQL are started, their status should change to "Running" with green indicators.

4. Test the Installation:

- **Check the XAMPP Dashboard:**
  - Open your web browser and type http://localhost/ in the address bar.
  - If everything is installed correctly, you should see the XAMPP dashboard, which confirms that your server is up and running.

5. phpMyAdmin:

- XAMPP comes with phpMyAdmin for managing MySQL databases.
- You can access phpMyAdmin by navigating to http://localhost/phpmyadmin/ in your browser**.**

III. EXECUTION

In terminal type:

Step-1) cmd

Step-2) conda activate python3.10.8

Step-3) python app.py

CHAPTER 11
PERFORMANCE EVALUATION

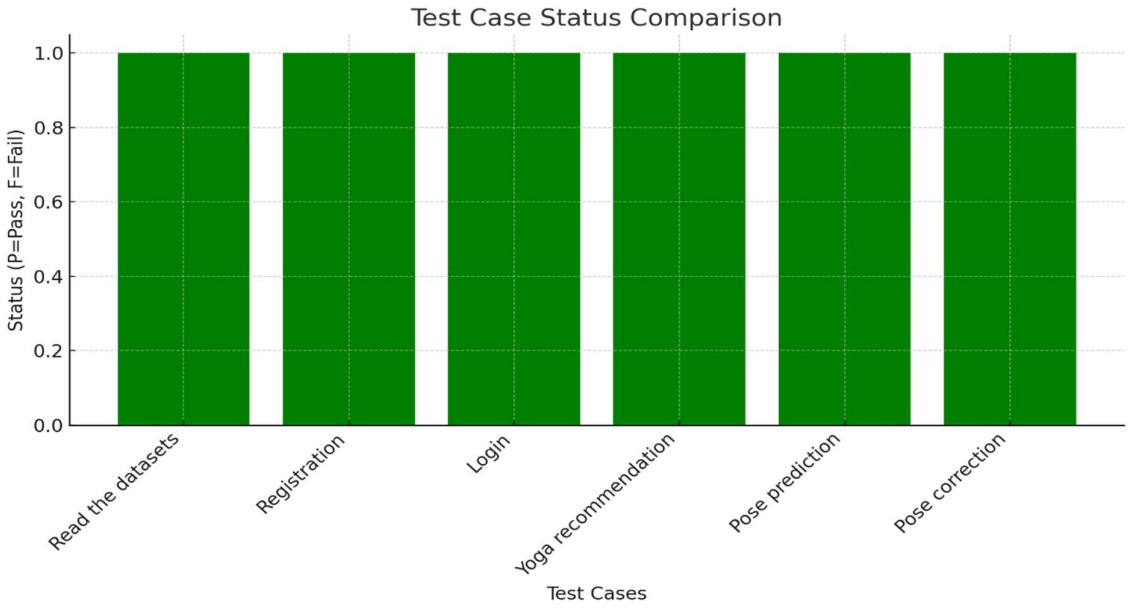

Fig 11.1 Performace Evalution

The system was evaluated based on six key test cases:

Read the datasets, Registration , Login ,Yoga Recommendation, Pose prediction, Pose correction

Results & Findings

Pass Rate: 100% (All test cases passed)

Failures: None recorded

System Stability: All modules performed as expected

Analysis & Interpretation

- The system demonstrates high reliability across core functionalities.
- No errors or failures were observed, indicating robust implementation.
- The test coverage suggests readiness for deployment, with further evaluation recommended for extreme edge cases.

CHAPTER 12
COMPARISION WITH EXISTING SYSTEM

1. **Headspace**
   - **Drawback:** Headspace primarily focuses on meditation and mindfulness but lacks personalized mood tracking and real-time interaction. It also has high subscription fees, limiting accessibility for many users.
   - **How CalmMind Fulfills It**: CalmMind offers personalized mood tracking, real-time suggestions via a chatbot, and a broader range of features like stress-relief exercises and a library of guided meditations at a lower or no cost.

2**. Calm**
   - **Drawback:** While Calm provides relaxation techniques and sleep support, it does not offer a personalized experience based on individual emotional patterns. Calm's premium subscription can also be a barrier for some users.
   - **How CalmMind Fulfills It:** CalmMind provides mood tracking to tailor suggestions for each user, integrating personalization through an AI chatbot. Additionally, it is designed to be free or low-cost, increasing accessibility.

3. Moodfit
   - **Drawback:** Moodfit offers mood tracking and mental health tools but lacks interactive real-time support and customization based on the user's daily moods. There is also a limited focus on integrating different forms of therapy, like guided meditation.
   - **How CalmMind Fulfills It:** CalmMind adds an AI chatbot for real-time, personalized feedback and integrates a more comprehensive approach by combining mood tracking, meditation, exercises, and relaxation activities.

4. Talkspace
   - **Drawback:** Talkspace focuses on virtual therapy, which can be costly and requires appointment-based interactions. It also has limited features for daily self-care and mood management.
   - **How CalmMind Fulfills It:** CalmMind provides continuous real-time support through a chatbot, alongside daily self-care tools like meditation and exercises, offering a more affordable and always-available alternative for users.

CHAPTER 13
CONCLUSION

This project successfully integrates a chatbot with a yoga pose identification and correction system, creating a holistic wellness platform that addresses both mental and physical well-being. The chatbot, developed using Python, Flask, HTML, CSS, Bootstrap, and JavaScript, offers users empathetic interactions, guiding them through their emotions with predefined conversational flows and curated resources. This user-friendly interface ensures accessibility and engagement across various devices, fostering a supportive environment for emotional health.

Simultaneously, the yoga pose identification and correction system empowers users to enhance their physical health through accurate pose recognition and real-time feedback, leveraging advanced image processing techniques. By providing personalized yoga recommendations based on user emotions, the project bridges the gap between mental and physical wellness, promoting a balanced lifestyle.

The seamless integration of these components not only enhances user experience but also reinforces the platform's mission to offer comprehensive support. Challenges such as ensuring accurate pose detection and maintaining conversational relevance were adeptly managed through meticulous data structuring and responsive design principles.
Looking forward, the project holds significant potential for expansion, including incorporating dynamic chatbot capabilities, expanding the range of yoga poses, and integrating additional wellness tools. Continuous user feedback and iterative improvements will further refine the platform, solidifying its role as an indispensable tool for holistic health and well-being.

CHAPTER 14
FUTURE ENHANCEMENTS

Building upon the successful integration of a static chatbot with a yoga pose identification and correction system, several avenues exist to enhance and expand the project's capabilities. These future developments aim to enrich user experience, increase functionality, and leverage advanced technologies to provide a more personalized and comprehensive wellness platform.

1. Advanced Yoga Pose Recognition

- **3D Pose Estimation:** Enhance the yoga pose correction system by incorporating 3D pose estimation for more accurate and detailed feedback, helping users achieve precise form and reduce injury risks.
- **Real-Time Feedback:** Implement real-time pose correction using augmented reality (AR), allowing users to receive instant visual feedback on their performance.

2. Multilingual and Multicultural Support

- **Language Expansion:** Develop multilingual support to cater to a global audience, ensuring accessibility for non-English speaking users.
- **Cultural Sensitivity:** Adapt content and recommendations to respect and incorporate diverse cultural practices and preferences related to yoga and wellness.

3. Mobile Application Development

- **Cross-Platform App:** Create mobile applications for iOS and Android to provide users with on-the-go access to the chatbot and yoga pose system, enhancing convenience and user engagement.

- **Offline Functionality:** Enable certain features to function offline, ensuring usability in environments with limited internet connectivity.

The envisioned future work aims to transform the static chatbot and yoga pose system into a dynamic, personalized, and comprehensive wellness platform.

CHAPTER 15
REFERENCES

[1]. Andersson & Titov (2014) - Web-Based Cognitive Behavioral Therapy and Chatbot Support https://www.sciencedirect.com/science/article/pii/B9780128034576000210

[2]. Calvo & Peters (2014) - Ethical Considerations in Emotional AI and Recommender Systems https://philarchive.org/archive/CALSHA-2

[3]. Bakker et al. (2016) - Mobile Apps for Mindfulness and Stress Reduction https://www.researchgate.net/publication/328635736_Mobile_Mindfulness_Meditation_a_Randomised_Controlled_Trial_of_the_Effect_of_Two_Popular_Apps_on_Mental_Health

[4]. Mittelstadt et al. (2016) - Ethical Issues in AI-Driven Mood Detection and Recommendations https://www.sciencedirect.com/science/article/pii/S2589004221002170

[5]. Mohr et al. (2017) - Integrating Lifestyle Changes with Digital Mental Health Tools https://pmc.ncbi.nlm.nih.gov/articles/PMC8414307/

[6]. Papandreou et al. (2018) - Pose Estimation Techniques for Fitness and Sports Applications https://www.researchgate.net/publication/381666579_Efficient_Human_Pose_Estimation_Leveraging_Advanced_Techniques_with_MediaPipe

[7]. Sharma et al. (2019) - Yoga Alignment Evaluation Using Pose Estimation Models https://www.researchgate.net/publication/384966073_Yoga_pose_detection_and_feedback_generation_A_review

[8]. Laamarti et al. (2020) - Multi-Modal Feedback Systems for Fitness and Well-Being https://pmc.ncbi.nlm.nih.gov/articles/PMC11603803/

[9]. Abbas et al. (2022) - Hybrid Recommender Systems for Personalized Wellness https://www.researchgate.net/publication/304291833_A_Framework_of_Hybrid_Recommender_System_for_Personalized_Clinical_Prescription

[10]. Chen et al. (2022) - Deep Learning-Based Pose Estimation in Fitness and Yoga https://www.sciencedirect.com/science/article/pii/S2405844024126205