

SMART FITNESS: PERSONALISED WORKOUT RECOMMENDATION WEB APPLICATION USING DECISION TREE ALGORITHM

A MINI PROJECT REPORT

Submitted by

MADHU MITHA G (221801030)

in partial fulfilment for the award of the degree of

**BACHELOR OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE
AND DATA SCIENCE**



**RAJALAKSHMI ENGINEERING COLLEGE
DEPARTMENT OF ARTIFICIAL
INTELLIGENCE AND DATA SCIENCE**

ANNA UNIVERSITY, CHENNAI

NOV 2024

ANNA UNIVERSITY, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Report titled “**SMART FITNESS: PERSONALISED WORKOUT RECOMMENDATION WEB APPLICATION USING DECISION TREE ALGORITHM**” is the bonafide work of **MADHU MITHA G (221801030)**, who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Dr. J.M. Gnanasekar M.E., Ph.D.,
Professor and Head
Department of Artificial Intelligence.
and Data Science
Rajalakshmi Engineering College
Chennai – 602 105

Dr. J. MANORANJINI M.Tech.,PhD
Associate Professor,
Department of Artificial Intelligence.
and Data Science
Rajalakshmi Engineering College
Chennai – 602 105

Submitted for the project viva-voce examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavour to put forth this report. Our sincere thanks to our Chairman Mr. S. MEGANATHAN, B.E, F.I.E., our Vice Chairman Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S., and our respected Chairperson Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D., for providing us with the requisite infrastructure and sincere endeavouring in educating us in their premier institution.

Our sincere thanks to Dr. S.N. MURUGESAN, M.E., Ph.D., our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to Dr. J.M. GNANASEKAR, M.E., Ph.D, Professor and Head of the Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. We are glad to express our sincere thanks and regards to our project supervisor Mrs. J MANORANJINI, M.Tech ., phd, Associate Professor, Department of Artificial Intelligence and Data Science and coordinator Dr.P. INDIRA PRIYA, M.E., Ph.D., Professor, Department of Artificial Intelligence and Data Science, Rajalakshmi Engineering College for his valuable guidance throughout the course of the project.

Finally we express our thanks for all teaching, non-teaching, faculty and our parents for helping us with the necessary guidance during the time of our project.

ABSTRACT

In this project, we focus on developing a fitness app that tailors workout recommendations to individual users by analyzing their fitness goals and performance data. The core objective is to enhance user engagement and improve fitness outcomes by delivering personalized workout regimens. By utilizing advanced data analysis techniques, the app ensures that each user's workout plan is optimized to meet their unique needs, thereby fostering a more effective and enjoyable fitness experience. To achieve this, we employ a decision tree algorithm for personalization, which helps in classifying users based on their fitness levels, goals, and progress. This approach enables the app to dynamically adjust workout recommendations as users' performance data changes over time, ensuring that the exercise routines remain challenging yet achievable. Additionally, a content-based filtering algorithm is incorporated to recommend exercises that align with the user's preferences and past workout history, creating a well-rounded, engaging workout experience. The effectiveness of this personalized approach is evaluated through various metrics, including user engagement levels and fitness progress indicators such as improvements in strength, endurance, and overall health. By tracking these metrics, we assess the app's ability to promote healthier lifestyles and improve fitness outcomes. This research highlights the potential of personalized fitness apps in not only fostering long-term user engagement but also contributing to more effective fitness journeys.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	i
	LIST OF FIGURES	iv
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 NEED FOR THE STUDY	2
	1.3 OBJECTIVES OF THE STUDY	2
	1.4 OVERVIEW OF THE PROJECT	3
2	REVIEW OF LITERATURE	4
	2.1 INTRODUCTION	4
	2.2 LITERATURE REVIEW	4
	2.3 FRAMEWORK OF LCA	5
3	SYSTEM OVERVIEW	6
	3.1 EXISTING SYSTEM	6
	3.2 PROPOSED SYSTEM	7
	3.3 FEASIBILITY STUDY	7
4	SYSTEM REQUIREMENTS	9
	4.1 HARDWARE REQUIREMENTS	9
	4.2 SOFTWARE REQUIREMENTS	9

5	SYSTEM DESIGN	11
5.1	SYSTEM ARCHITECTURE	11
5.2	MODULE DESCRIPTION	13
5.2.1	ADMIN MODLE	13
5.2.2	ADMIN TRACKING USER	13
5.2.3	USER SIGNUP MODULE	14
5.2.4	USER LOGIN PAGE	14
5.2.5	USER INPUT MODULE	15
5.2.6	WORKOUT RECOMMENDATION	16
5.2.7	USER HISTORY MODULE	17
6	RESULT AND DISCUSSION	18
6.1	RESULT	18
6.2	DISCUSSION	18
7	CONCLUSION AND FUTURE ENHANCEMENT	
7.1	CONCLUSION	19
7.2	FUTURE ENHANCEMENT	20
	APPENDIX	22
A1.1	SAMPLE CODE	22
A1.2	SCREENSHORTS	28
	REFERENCES	32

LIST OF FIGURES

Figure No	Figure Name	Page No
1.	System Architecture	12
2.	Admin Module	28
3.	Admin tracking user dashboard	28
4.	User signup	29
5.	User loginpage	29
6.	User Input	30
7.	Workout Recommendation	30
8.	User profile and history	31

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Fitness apps have revolutionized how individuals approach their health and wellness goals, providing users with convenient access to workout plans, tracking tools, and personalized recommendations. As more people turn to digital solutions to manage their fitness, the demand for innovative features that can keep users engaged and motivated has grown. Despite the widespread availability of fitness apps, many users struggle with maintaining consistent exercise habits and staying committed to their goals over time. This gap highlights the need for advanced strategies that not only enhance user experience but also promote long-term adherence to fitness routines.

One of the most promising approaches to improving user engagement is the integration of gamification elements into fitness apps. By incorporating features such as rewards, challenges, and badges, these apps can transform workouts into interactive, enjoyable experiences. Gamification taps into users' natural desire for achievement and competition, making it easier for them to stay motivated. As users complete challenges and earn rewards, they build a sense of accomplishment that encourages continued participation, ultimately fostering healthier exercise habits.

In addition to gamification, personalization is a key factor in maintaining user motivation. Fitness apps that analyze user performance data and fitness goals can offer tailored workout recommendations, ensuring that each exercise regimen is suited to the individual's abilities and preferences. Personalized reminders and real-time progress tracking further enhance the user experience by providing ongoing encouragement and making it easier for users to monitor their improvements. This level of customization helps users stay engaged, as the app adapts to their evolving fitness needs.

Moreover, the inclusion of social and community features in fitness apps can significantly impact user engagement. By allowing users to participate in group challenges, track each other's progress, and offer support, apps create a sense of accountability and camaraderie. Users are more likely to remain committed to their fitness goals when they feel connected to a community of like-minded individuals. Additionally, providing guidance on nutrition, sleep, and mental well-being helps users adopt a more holistic approach to health, encouraging them to integrate wellness into their everyday lives.

1.2 NEED FOR THE STUDY

The need for this study stems from the widespread use of fitness apps and the challenge of maintaining long-term user engagement and adherence to workout routines. Despite providing accessible tools, many users struggle with sustaining motivation and commitment, leading to a gap between initial enthusiasm and consistent progress. Enhancing user engagement and fostering sustained commitment is crucial for maximizing the effectiveness of these apps.

This study will explore how personalized features, such as tailored workout plans and progress tracking, can improve the user experience by addressing common issues like disengagement and overly challenging routines. It will also evaluate the role of gamification and social integration in boosting user motivation by making workouts more enjoyable and providing community support.

Ultimately, the study is essential to understanding how fitness apps can effectively use personalization, gamification, and social features to promote healthier lifestyles, improving user retention and fitness outcomes.

1.3 OBJECTIVES OF THE STUDY

The objective of this study is to investigate strategies that enhance user engagement and adherence in fitness apps by leveraging personalization, gamification, and social integration. Specifically, the study aims to:

1. **Examine the impact of personalized workout recommendations:** based on user fitness goals and performance data, assessing how tailored plans influence user motivation and progress.
2. **Assess the role of social integration features:** including community support, group challenges, and leaderboards, in fostering accountability and increasing user commitment to exercise.
3. **Analyze user engagement metrics:** to determine how these advanced features contribute to better user retention and overall fitness outcomes.
4. **Provide insights into the development of more effective fitness apps:** That successfully combine personalization, gamification, and social elements to support healthier lifestyles and improved long-term fitness results.

1.4 OVERVIEW OF THE PROJECT

This study focuses on improving user engagement and long-term adherence in fitness apps through the implementation of advanced features such as personalization, gamification, and social integration. Despite the growing popularity of fitness apps, many users struggle to maintain consistent exercise routines and stay motivated over time, leading to poor fitness outcomes and user dropout. The research aims to address these challenges by exploring how tailored workout plans, interactive elements, and community support can make fitness journeys more enjoyable and effective.

A key area of exploration in the study is the role of personalized workout recommendations. By analyzing user data, including fitness goals and past performance, the app can create custom workout plans that cater to each individual's unique needs. This personalized approach ensures that users receive exercises that are both appropriately challenging and aligned with their progress, making it more likely that they will remain committed to their fitness routines. The study will assess how this level of customization impacts user motivation and long-term adherence.

In addition to personalization, the study examines the effectiveness of gamification in enhancing user engagement. Gamification features such as rewards, challenges, and achievement badges transform exercise routines into interactive, goal-oriented experiences. By introducing a sense of fun and competition, these elements encourage users to meet their workout goals, fostering a sense of accomplishment and motivating them to continue using the app. The study will evaluate how these features impact user satisfaction and retention.

Finally, the research looks at the role of social integration in fitness apps. Community features such as group challenges, leaderboards, and forums provide users with a sense of belonging and accountability. Engaging with others who share similar goals can help users stay motivated and overcome mental barriers that might otherwise lead to disengagement. The study aims to determine how the combination of personalization, gamification, and social support can create a more immersive and effective fitness app experience, ultimately promoting healthier lifestyles and better fitness outcomes.

CHAPTER 2

REVIEW OF LITERATURE

INTRODUCTION:

Fitness apps have revolutionized personal health by offering data-driven, personalized workout recommendations. However, sustaining user motivation remains challenging. This project aims to develop a workout recommendation system using decision tree algorithms to analyze user data and tailor routines to individual goals, fostering a more engaging and sustainable fitness experience.

Literature Review

1. Mobile Device Interventions for Physical Activity:

A systematic review by Muntaner et al. (2016) highlights the effectiveness of mobile interventions in increasing physical activity. This study underscores the potential of mobile applications to significantly enhance activity levels through personalized feedback and real-time interventions. However, it also points out common challenges such as maintaining user motivation and engagement over time.

2. Effectiveness of Weight Loss Apps:

Breton et al. (2011) evaluated the adherence of weight loss apps to evidence-based practices. Their research indicates that while many apps provide effective weight loss strategies, adherence to scientifically informed practices remains inconsistent. This emphasizes the importance of using advanced algorithms, like decision trees, to deliver evidence-based, personalized recommendations for sustainable fitness.

3. Perceived Motivational Affordances in Activity Trackers:

Jarrahi et al. (2018) examined the role of motivational affordances in fitness trackers, finding that personalized feedback increases user motivation to maintain physical activity. This insight aligns with the need for real-time, adaptive systems like our decision tree-based workout recommendation engine.

4. Tailoring Physical Activity Coaching Systems:

Op den Akker et al. (2014) explored real-time physical activity coaching systems and developed a model for personalized feedback. Their research indicates that dynamically adjusting recommendations based on real-time user data significantly improves the effectiveness of fitness apps. This serves as a foundational concept for our proposed system, which will rely on real-time updates of user progress to adapt workout recommendations.

Framework of Latent Class Analysis (LCA)

Latent Class Analysis (LCA) is a statistical method used to identify unobserved subgroups (or "classes") within a population based on responses to observed variables. In the context of fitness applications, LCA can segment users into distinct groups based on factors like fitness levels, goals, and past performance. These groups, or latent classes, allow for the customization of workout plans, ensuring that recommendations are relevant to users' specific needs and goals.

Steps Involved in LCA:

- 1. Data Collection :** Gather user input such as fitness goals, current activity level, and historical performance data.
- 2. Modeling Latent Classes :** Use LCA to identify distinct user profiles based on patterns in the data. For example, one group may consist of users aiming for weight loss, while another may focus on strength-building.
- 3. Classification :** Assign users to a latent class based on their characteristics. The decision tree algorithm plays a key role here by evaluating input data and classifying users.
- 4. Workout Recommendation :** For each latent class, personalized workout plans are generated, optimizing intensity, duration, and type of exercise to fit the identified class's goals.
- 5. Continuous Update :** As users progress and their fitness levels change, they can be reclassified into different latent classes, and their workout recommendations will be updated accordingly.

CHAPTER 3

SYSTEM OVERVIEW

3.1 EXISTING SYSTEM

In the existing system, fitness apps primarily offer basic features such as predefined workout routines, activity tracking, and general health tips. These apps often provide access to a wide variety of workouts, including strength training, cardio, yoga, and more. Users can select from a range of fitness plans based on their preferences, fitness levels, or specific goals (e.g., weight loss, muscle gain). However, these workout plans are often generalized and lack the adaptability required to meet individual user needs over time.

Many fitness apps also include tracking features that monitor users' steps, calories burned, and workout durations. While these features offer a degree of insight into user activity, they typically lack deeper analysis and personalized recommendations based on real-time performance data. The engagement strategies in existing systems are often limited to push notifications or reminders, which are not always effective in sustaining user motivation. Additionally, these systems rarely provide personalized reminders tailored to individual habits and progress.

Social features in current fitness apps are generally underdeveloped, with limited integration of community-building tools. Some apps allow users to share their workouts or compete on leaderboards, but these features are often passive and do not foster a strong sense of accountability or social engagement. Moreover, gamification elements—such as rewards, badges, or challenges—are either absent or minimally implemented, reducing the potential for making workouts more interactive and enjoyable. The lack of personalization, social interaction, and gamification in existing systems highlights the need for a more advanced approach to fitness apps, which this study aims to address.

3.2 PROPOSED SYSTEM

The proposed system utilizes advanced data analysis to deliver personalized workout recommendations via a decision tree algorithm. Input features include user data such as fitness level, goals, workout history, and performance metrics. These features are analyzed to understand the user's current fitness status and preferences. The system outputs a customized workout plan with details like workout type, intensity, and duration, tailored to the user's specific needs to optimize fitness outcomes.

The model's training process is based on supervised learning, using historical data to map user features to optimal workout recommendations. As the system learns from past user data, it identifies patterns that improve its ability to suggest personalized workout plans. Key decision nodes are built around the user's goals, preferences, and performance metrics. For example, the system considers goals like weight loss or muscle gain, alongside preferences for types of workouts, such as cardio or strength training.

Each decision path leads to a leaf node representing a personalized workout plan. These final recommendations align with the user's fitness profile, ensuring that the suggested workouts are neither too easy nor too challenging. The system continuously updates recommendations based on real-time progress and new data, adjusting workout intensity and focus dynamically. This adaptive approach helps enhance user engagement, promoting long-term adherence to fitness routines and ensuring better fitness outcomes.

3.3 FEASIBILITY STUDY

The feasibility study for this personalized fitness app evaluates the technical, operational, and economic viability of the proposed system, which aims to enhance user engagement and fitness outcomes through personalized workout recommendations, progress tracking, and gamification features.

Technical Feasibility:

The proposed system is built using advanced machine learning techniques, specifically a decision tree algorithm, to generate personalized workout recommendations. This approach is technically feasible because decision trees are well-suited for classifying users based on various inputs such as fitness goals, current fitness levels, workout preferences, and historical performance data. The algorithm learns from past data to optimize future workout suggestions, continuously improving as it gains more user-specific data. Additionally, content-

based filtering is used to suggest exercises based on the user's workout history, ensuring that the system adapts to individual preferences over time.

The technical infrastructure of the app relies on modern technologies like Flask for managing user sessions and API calls, and MongoDB for flexible data storage and efficient retrieval of user profiles, workout logs, and analytics. Flask is a lightweight and scalable framework ideal for managing secure user interactions, while MongoDB's schema-less format allows for the easy handling of dynamic data from various users. These technologies ensure the system is not only capable of handling large amounts of user data but is also scalable to support future growth and feature expansion.

Operational Feasibility:

The app is designed to enhance user engagement through personalized experiences. It provides tailored workout plans based on user profiles and fitness goals, allowing users to track their progress over time. Features like real-time updates, progress tracking, and personalized reminders aim to keep users motivated. By integrating gamification elements such as badges, rewards, and challenges, the app encourages users to engage regularly and meet their fitness goals. The social aspect of the app, including group challenges and leaderboards, fosters accountability and community engagement, which are key to long-term user retention.

The app also includes an admin module to manage user data and monitor overall system performance. Admins have access to dashboards that track user activity, engagement levels, and workout completion rates, allowing for continuous system improvements and user support. Additionally, secure login and data management protocols ensure the privacy and security of user information.

Economic Feasibility:

From an economic perspective, the app is cost-effective to develop and maintain. Using open-source technologies like Flask and MongoDB reduces initial development costs while offering scalability for future growth. The decision tree algorithm, once developed, requires minimal maintenance, further reducing ongoing expenses. The app has potential revenue streams through premium features, in-app purchases, and partnerships with fitness brands. Given the rising demand for personalized fitness solutions, there is a strong potential for user adoption and retention, ensuring a solid return on investment.

CHAPTER4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Development Workstations (Computers/Laptops):

- High-performance computers for coding, testing, and debugging.
- Minimum Specifications:
 - Processor: Intel i5 or better.
 - RAM: 8 GB or higher.
 - Storage: SSD with at least 256 GB.

Internet Router/Modem:

- Stable internet connection for downloading libraries, app testing in the cloud, and collaboration with team members.

Server or Cloud Infrastructure (if hosting data):

- Cloud service providers such as AWS, Google Cloud, or a private server for hosting the app's backend, handling user data, and managing database queries.

4.2 SOFTWARE REQUIREMENTS

1. Programming Language:

Python: The primary language used for developing the backend logic of the application.

2. Frameworks:

Flask: A micro web framework for Python used to build the web application and handle routing, user sessions, and template rendering.

3. Database:

MongoDB: A NoSQL database used to store user information, workout details, and other relevant data. It allows for flexible data storage and retrieval.

4. Libraries:

pymongo: A Python library to interact with MongoDB, allowing the application to perform CRUD (Create, Read, Update, Delete) operations on the database.

Jinja2: A templating engine used by Flask for rendering HTML templates dynamically with data from the backend.

5. HTML/CSS/JavaScript:

HTML: Used for creating the structure of web pages (e.g., login, BMI input, and result pages).

CSS: For styling the web pages (presumably included in the HTML files, although not shown in the code).

JavaScript: For adding interactivity on the client side (presumably in the HTML templates, although not explicitly shown in the code).

6. Development Tools:

Integrated Development Environment (IDE): A code editor such as Visual Studio Code, PyCharm, or any other text editor for writing and managing the Python code and templates.

Postman: For testing APIs (not explicitly mentioned in the code, but useful for any RESTful service testing).

7. Version Control:

Git: For version control, tracking changes in the codebase, and collaborating with others.

8. Web Server:

Built-in Flask Development Server: The app uses Flask's built-in server for local development (indicated by `app.run(debug=True)`).

9. Deployment Environment:

Local Development Environment: The application runs locally on a server set up using Python and Flask. Deployment can later be on cloud platforms like AWS or Heroku.

10. Optional:

Docker: (if used in future deployments) for containerizing the application to ensure consistent environments across development and production.

- **Selenium:** (optional, for testing purposes) to automate browser-based testing of the web application.

11. Security:

Flask-Login: (optional, if needed) for handling user sessions and authentication securely, though not explicitly in the provided code.

These software components form the backbone of the fitness app, allowing it to manage user interactions, store data, and provide personalized workout recommendations based on BMI calculations.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

The architecture of a Fitness App system has components organized into four main sections: Frontend, Backend, External Services, and Databases. Here's an overview of each section:

1. Frontend

Web App: This is the user interface where users interact with the fitness app. Users can access features such as workout recommendations, data visualization, and progress tracking.

2. External Services

User Authentication: This component, typically integrated with an external authentication provider (like Google, Facebook, or custom OAuth services), handles user sign-in and access control for security. It ensures only authorized users can access their data and other app features.

3. Backend

API Gateway: Acts as a central entry point for all client requests coming from the frontend. It directs each request to the appropriate backend service and provides security, routing, and load balancing.

Workout Recommendation Service: Uses user data and preferences to generate customized workout plans or exercise suggestions. It likely interacts with the Machine Learning Service to personalize recommendations based on user history and fitness goals.

User Data Service: Manages and retrieves data related to user profiles, goals, and preferences. This service pulls data from the User Data DB in the database layer to display relevant information to the user.

Analytics Service: Processes and analyzes user activity and performance data. It tracks metrics like workout frequency, progress, and other insights to provide users with meaningful analytics. This service pulls data from the Analytics DB to support data-driven insights.

Machine Learning Service: Powers the intelligence behind the workout recommendations and analytics. It likely uses algorithms to process large amounts of user data, providing personalized suggestions and insights.

4. Databases

Workout Data DB: Stores workout-related data, such as exercises, routines, and completed workouts for each user.

User Data DB: Holds user-specific information, including profile details, fitness goals, and preferences.

Analytics DB: Stores analyzed data and insights for individual users, including workout progress, performance stats, and other analytics.

This architecture enables the app to deliver personalized workout recommendations, track user progress, and provide insightful analytics, all while maintaining scalability and security.

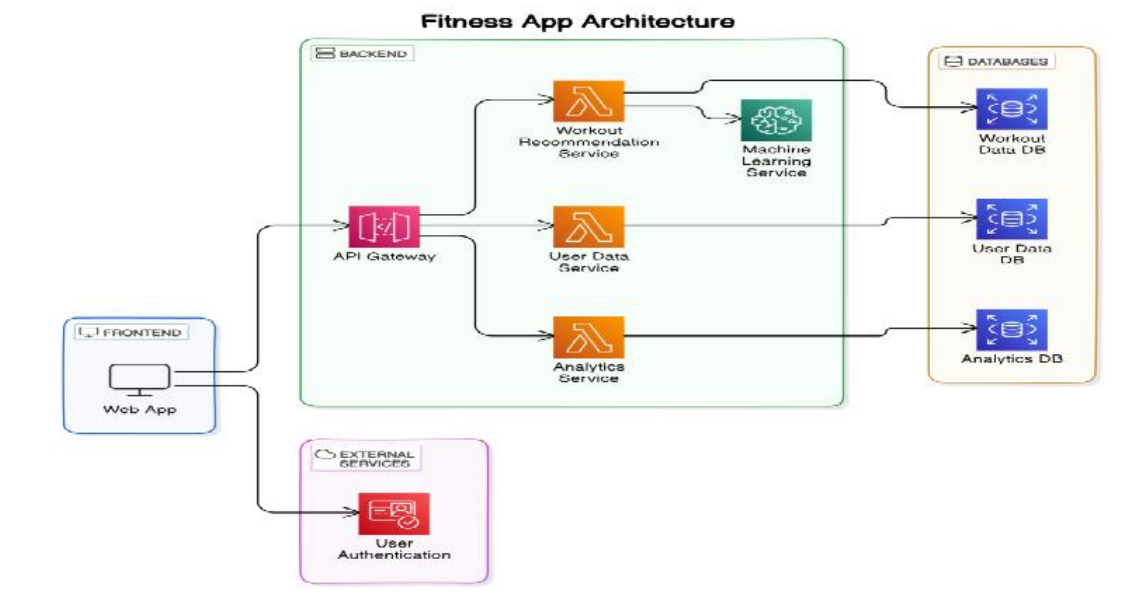


Figure:1

5.2 MODULE DESCRIPTION

5.2.1 ADMIN MODULE

Admin Secure Login and Management System

A secure admin login system, built using Flask's authentication and session handling, ensures efficient control over user data and app operations. Admins authenticate with encrypted, hashed credentials, minimizing unauthorized access risks. Flask's session management provides continuous, secure access while expiring sessions after inactivity.

Role-Based Access Control : limits portal access to authorized admins, granting them exclusive rights to view and manage user accounts, monitor app engagement, and perform system maintenance tasks. User activity logs and metrics, including login frequencies and workout engagement, are available in real-time, enabling admins to identify trends and improve user experience.

Technology : Flask's lightweight framework provides secure, modular, and scalable authentication for handling admin-only functionalities, while efficiently supporting future growth in data and user volume.

5.2.2 ADMIN TRACKING USER DASHBOARD MODULE

Admin Dashboard for User Management and Engagement Monitoring

The admin dashboard provides a comprehensive platform for administrators to oversee user activity, engagement, and overall app usage. This centralized system allows admins to monitor and manage users efficiently, ensuring the app's goals of enhancing fitness engagement and personal wellness are met.

1. User Management :

Overview of Registered Users : The dashboard displays an organized list of all registered users, showing essential information such as name, email, and registration date. This overview allows administrators to quickly access user details, streamlining communication and support efforts.

User Status : Each user is categorized based on their account status—active, inactive, or pending verification—enabling admins to monitor the user life cycle and identify inactive users who may need engagement strategies.

2. Activity Tracking :

User Activity Log : The system captures and logs each user's actions within the app, including login frequency and completed workout sessions. This log serves as a behavioral record, helping admins understand user engagement patterns.

Engagement Metrics : Data visualization tools, such as charts and graphs, illustrate engagement trends over time, including daily activity levels and workout consistency. These insights provide a clear view of how users are interacting with the app, guiding improvements in content and feature offerings to better meet user needs.

This admin dashboard empowers administrators to make informed decisions based on real-time user data, enhancing the app's ability to deliver a tailored fitness experience.

5.2.3 USER SIGNUP MODULE

User Signup Module :

The User Signup Module enables a secure, efficient registration process for new users. During signup, users provide essential information—such as name, email, and fitness goals to create accounts tailored to their needs.

Form Validation : ensures accurate data input by verifying email format and highlighting any errors before submission, enhancing data reliability. This validated data is then securely stored in a database, like MongoDB, which supports scalability and data protection through encryption.

Backend API Interaction : securely transfers user data to the database, enabling integration with other app components, such as authentication and workout recommendation modules. By leveraging API calls, the signup process remains fast, with seamless backend integration.

Overall, this module provides users with a straightforward onboarding experience, setting the stage for personalized fitness recommendations and a smooth app journey.

5.2.4 USER LOGIN PAGE MODULE

User Login Module:

The User Login Module provides users with secure access to personalized features. During login, users enter their credentials, which the system authenticates by checking them against stored records, ensuring only authorized access.

Session Management :

Using Flask's session management, the app securely maintains user sessions, allowing continuous access until the user logs out or a set inactivity period ends. Each session is uniquely identified and encrypted for security.

Password Hashing :

Passwords are stored as hashed values using Flask's hashing function, making them irreversible and adding security by safeguarding plain text data. During login, the app compares the hashed version of the entered password to the stored hash.

Error Handling :

For incorrect credentials, the system provides immediate feedback and limits login attempts, locking accounts after multiple failed attempts for additional security. Flask's session and hashing capabilities ensure a secure, efficient login process integrated with the app's personalized features.

5.2.5 USER INPUT MODULE

Algorithm for BMI Calculation and Workout Recommendation

This algorithm calculates a user's Body Mass Index (BMI) and generates personalized workout plans based on their BMI category. Here's a detailed breakdown:

1. Input User Details :

The system prompts the user to enter basic information such as gender, age, height (in centimeters), and weight (in kilograms). These inputs are essential for calculating the BMI, which will then inform the workout recommendation.

2. Convert Height from Centimeters to Meters :

To calculate BMI, height must be in meters. The system converts the user's height from centimeters to meters by dividing the entered value by 100.

Example : If a user's height is 170 cm,

then height in meters is $(170 \div 100 = 1.7, \text{m})$.

3. Calculate BMI :

The BMI is calculated using the following formula:

$$\text{BMI} = \{\text{Weight (kg)}\} / \{\{\text{Height (m)}\}^2\}$$

Example : For a user who weighs 65 kg and has a height of 1.7 m, the calculation is: $\{\text{BMI}\} = \{65\} / \{1.7^2\} = \{65\} / \{2.89\}$ approx 22.5

4. BMI Classification :

The calculated BMI is then classified into one of four categories,

- Underweight : $\text{BMI} < 18.5$
- Normal weight : $18.5 \leq \text{BMI} < 24.9$
- Overweight : $25 \leq \text{BMI} < 29.9$
- Obese : $\text{BMI} \geq 30$

5.2.6 WORKOUT RECOMMENDATION MODLUE

Workout Recommendation Using Decision Tree :

The BMI category serves as input to a decision tree that recommends workout plans. For example:

Underweight : Suggests strength-building exercises to increase muscle mass (e.g., weight training, moderate cardio).

Normal weight : Recommends balanced routines, combining cardio, flexibility, and strength training.

Overweight : Focuses on high-cardio and calorie-burning exercises to promote fat loss.

Obese : Recommends low-impact exercises to reduce strain on joints, like walking and light resistance exercises, progressing as fitness improves.

Output :

The algorithm provides the user with a customized workout plan tailored to their BMI category, offering exercise types, intensities, and durations that best support their fitness goals.

This BMI-based workout recommendation algorithm uses simple arithmetic and decision-tree classification to provide a tailored fitness experience.

5.2.7 USER PROFILE AND HISTORY MODULE**1. User Data Management :**

Displays and allows updates to personal details (name, age, fitness level) stored securely in the database. Updates trigger database queries to ensure real-time reflection of data.

2. Fitness History Tracking :

Records each workout session, including details like date, exercises, and outcomes (e.g., calories burned). This provides a timeline of activities to monitor improvements.

3. Progress Monitoring :

Users set fitness goals, such as weight loss targets or strength gain. Progress is automatically tracked and visualized using charts or graphs, allowing users to see improvements over time.

4. Technology and Database Interaction :

The backend uses CRUD (Create, Read, Update, Delete) operations for managing user data, workout records, and goals. Secure database interactions ensure accurate, up-to-date information and protect user privacy.

CHAPTER 6

RESULT & DISCUSSION

6.1 RESULT

The personalized workout recommendation web application leverages a combination of BMI calculations, decision tree algorithms, and content-based filtering to create workout plans that are uniquely tailored to each user's fitness goals, level, and history. By analyzing key data inputs such as height, weight, age, and fitness objectives, the app categorizes users into different BMI classes, including underweight, normal, overweight, and obese, and then offers specific exercise recommendations accordingly. The decision tree classifier enables the app to group users effectively, allowing for workout regimens that align closely with individual needs. Content-based filtering further refines this by suggesting personalized exercises, sets, and repetitions to suit each user's preferences. This tailored approach enhances the likelihood of users achieving their desired health outcomes, as it continually adapts based on user feedback and progress. The system's adaptability means it doesn't simply provide an initial set of recommendations but evolves to maintain relevance over time, creating a workout experience that grows in effectiveness and engagement. With its data-driven personalization, this fitness app prioritizes user satisfaction and fitness progress, offering a unique solution compared to traditional one-size-fits-all programs.

6.2 DISCUSSION

The incorporation of machine learning, particularly the decision tree classifier, significantly boosts the app's ability to deliver precise, individualized workout plans, setting it apart from conventional static fitness routines. This algorithm's classification of users based on BMI and other metrics provides a clear and accurate foundation for personalizing fitness guidance. The app's dynamic adaptability, driven by ongoing user data, enables a responsive workout experience that better aligns with evolving user needs. Decision tree algorithms, recognized for their transparency and interpretability, make the recommendations easy to understand and follow, enhancing user engagement. Despite these strengths, the app may encounter challenges related to data privacy and user motivation, both of which are essential for maintaining user engagement and effectiveness. Privacy concerns could deter some users from fully participating, and users who struggle with motivation may find it hard to adhere to their workout plans. Addressing these potential issues with enhanced gamification, strong privacy protections, and reward-based engagement features could help retain users and build trust. By providing a user-centered, adaptive, and privacy-conscious platform, this app offers a forward-thinking approach to fitness, aiming to deliver .

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 Conclusion:

The personalized fitness app developed using Flask, MongoDB, and a decision tree algorithm provides a robust solution for enhancing user engagement and fitness outcomes. By capturing and analyzing user-specific data such as BMI, fitness goals, workout history, and preferences, the app tailors workout recommendations that adapt over time with the user's progress. This personalized approach helps users stay motivated, offering a variety of exercises and goal-oriented plans that cater to individual fitness levels and objectives, from weight loss to strength training. Key features such as user authentication and secure access ensure that user data is protected, while the workout history tracking functionality allows users to see their progress over time, reinforcing commitment and a sense of accomplishment. The decision tree algorithm enables dynamic recommendation adjustments, making the workouts more relevant and personalized as the user's fitness journey evolves. MongoDB serves as the database backbone, efficiently managing user profiles, workout logs, and historical data, which supports scalability as more users engage with the app. Flask handles the server-side functionality, making the app responsive and fast. The app's integration of data-driven insights and user-specific recommendations addresses common challenges in fitness, particularly motivation and adherence, by providing a user-centric, goal-driven approach. This approach ensures that users receive relevant guidance and feedback, fostering a positive, engaging experience that supports their commitment to fitness. Overall, the app succeeds in delivering a flexible, scalable, and personalized fitness solution, effectively bridging the gap between fitness goals and achievable results through technology and user-centered design.

7.2 Future Enhancements:

1. Mobile Application Development : Developing a mobile version of the app would allow users to access features conveniently on their smartphones, meeting the needs of a digitally mobile audience. A mobile app would provide flexibility for users to track their workout routines, log data in real-time, receive workout recommendations, and monitor their progress seamlessly. Accessibility on mobile devices would likely increase user engagement and satisfaction, as users can interact with the app from any location, whether at the gym, home, or outdoors.

2. Integration with Wearable Devices : To provide more accurate and real-time fitness insights, integrating the app with wearable devices, such as Fitbit or Apple Watch, would be a significant enhancement. Wearable devices can capture data like heart rate, steps taken, distance covered, and calories burned. By accessing this data, the app could deliver highly personalized workout recommendations and adjust them based on the user's real-time fitness metrics. This integration would offer a data-driven approach to fitness, allowing users to make informed decisions and follow workout plans that are tailored to their daily activity levels and overall health metrics.

3. Enhanced Gamification Features : Adding advanced gamification elements, such as badges, leaderboards, and social sharing options, can make workouts more enjoyable and competitive. Badges for achievements (like completing a week of workouts) or leaderboards for various challenges can motivate users to reach specific goals. Social sharing features would allow users to share their progress or achievements with friends, creating a sense of community and accountability. This competitive and reward-based structure can drive users to adhere to their fitness plans more consistently, ultimately enhancing their overall fitness journey and long-term retention in the app.

4. AI-Driven Recommendations : The app could greatly benefit from more advanced AI techniques beyond decision trees, such as reinforcement learning or neural networks. These methods can enable the app to make workout recommendations that evolve based on user performance, feedback, and engagement patterns. Reinforcement learning, for example, allows the app to learn from the user's progress and adapt recommendations to better meet their goals over time. This continuous optimization would ensure that the workout plans remain challenging, relevant, and aligned with the user's growth, keeping users motivated and avoiding stagnation.

5. Nutrition Tracking : Integrating nutrition tracking within the app could provide a more comprehensive fitness solution by addressing both exercise and dietary habits. With a nutrition feature, users could log their meals, track calorie intake, and get recommendations for meals that align with their fitness goals, whether they aim to lose weight, build muscle, or maintain health. The app could offer dietary suggestions based on the user's workout intensity, goals, and even preferences, thereby creating a balanced approach to health and fitness. This feature would be particularly valuable as nutrition plays a critical role in achieving and maintaining fitness results.

6. Community and Social Features : Building a community aspect within the app by introducing group challenges, forums, or events could significantly enhance user retention and engagement. Group challenges, for instance, could motivate users by allowing them to compete or collaborate on achieving fitness milestones. Forums and community groups provide users with a space to discuss workouts, share tips, and celebrate achievements, fostering a supportive environment. These social features can create a sense of camaraderie and accountability, which often helps users stay committed to their goals over the long term.

APPENDIX

A1.1) SAMPLE CODE:

```
from flask import Flask, request, redirect, render_template
import pymongo

app = Flask(__name__)
userName = ''
password = ''
template = 'login.html'

def login_failed():
    return 'Login Failed!'

def validate_user(userName, password):
    loginsuccess = False
    dbClient = pymongo.MongoClient("mongodb://localhost:27017/")
    dbCon = dbClient["local"]
    users = dbCon["users"]
    for user in users.find():
        if userName == user["name"] and password == user["password"]:
            loginsuccess = True
    return loginsuccess

def recommend_workouts(bmi_category, fitness_level):
    # Define BMI categories and workout counts (sets and reps) for
    # each activity
    workouts = {
        "underweight": {
            "pushups": {"sets": 2, "reps": 5},
            "pullups": {"sets": 1, "reps": 3},
            "walking": {"sets": 1, "reps": 30},    # minutes of
walking
            "squats": {"sets": 2, "reps": 5}
        },
        "normal": {
            "pushups": {"sets": 3, "reps": 10},
            "pullups": {"sets": 2, "reps": 5},
            "walking": {"sets": 1, "reps": 20},    # minutes of
walking
            "squats": {"sets": 3, "reps": 10}
        },
        "overweight": {
            "pushups": {"sets": 2, "reps": 8},
```

```

        "pullups": {"sets": 1, "reps": 4},
        "walking": {"sets": 1, "reps": 40},      # minutes of
walking
        "squats": {"sets": 2, "reps": 8}
    },
    "obese": {
        "pushups": {"sets": 1, "reps": 5},
        "pullups": {"sets": 1, "reps": 2},
        "walking": {"sets": 1, "reps": 60},      # minutes of
walking
        "squats": {"sets": 1, "reps": 5}
    }
}

# Adjust workouts based on fitness level (beginner, intermediate,
advanced)
fitness_adjustments = {
    "beginner": 0.8,
    "intermediate": 1,
    "advanced": 1.2
}

# Get adjustment factor based on fitness level
adjustment_factor = fitness_adjustments.get(fitness_level, 1)

# Adjust the sets and reps based on fitness level
adjusted_workouts = {}
for activity, counts in workouts[bmi_category].items():
    adjusted_sets = max(1, int(counts["sets"] *
adjustment_factor)) # Ensure at least 1 set
    adjusted_reps = max(1, int(counts["reps"] *
adjustment_factor)) # Ensure at least 1 rep
    adjusted_workouts[activity] = {"sets": adjusted_sets, "reps":
adjusted_reps}

return adjusted_workouts

@app.route('/')
def my_form():
    return render_template(template)

userName = request.form['inputUserName']
password = request.form['inputPassword']
if validate_user(userName, password) == False:
    return redirect(url_for('failed'))

```

```

elif (userName == "madhu"):
    return redirect(url_for('admin', userName=userName))
else:
    return redirect(url_for('bmi', userName=userName,
editUser=userName))

@app.route('/bmi')
def bmi():
    userName = request.args.get('userName')
    editUser = request.args.get('editUser')
    if userName is None or userName == '':
        return 'Not authorized!'
    else:
        return render_template('bmi.html', userName=userName,
editUser=editUser)

@app.route('/bmi', methods=['POST'])
def bmi_post():
    height = int(request.form['height'])
    weight = int(request.form['weight'])
    fitness = request.form['fitness']
    userName = request.form['userName']
    editUser = request.form['editUser']
    height_m = height / 100
    bmi_value = round(weight / (height_m ** 2), 2)
    if bmi_value < 18.5:
        bmi_category = "underweight"
    elif 18.5 <= bmi_value < 24.9:
        bmi_category = "normal"
    elif 25 <= bmi_value < 29.9:
        bmi_category = "overweight"
    else:
        bmi_category = "obese"

    workout_plan = recommend_workouts(bmi_category, fitness)

    dbClient = pymongo.MongoClient("mongodb://localhost:27017/")
    dbCon = dbClient["local"]
    users = dbCon["users"]
    userdetail = dbCon["userdetail"]
    query = { "name": editUser }
    newvalues = { "$set": { "height": height, "weight": weight,
"fitness": fitness, "bmi": bmi_value, "bmicategory": bmi_category,
"workouts": workout_plan } }
    users.update_one(query, newvalues)

```

```

        userInfo = { "name": userName, "height": height, "weight":
weight, "fitness": fitness, "bmi": bmi_value, "bmicategory":
bmi_category, "workOuts": workout_plan }
        userdetail.insert_one(userInfo)

workoutList = []
workOutDetail = ""
        for activity, counts in workout_plan.items():
            if activity == 'walking':
                workoutList.append(f"{activity.capitalize()}:
{counts['reps']} minutes")
            else:
                workoutList.append(f"{activity.capitalize()}:
{counts['sets']} sets of {counts['reps']} reps")

@app.route('/bmiresult')
def bmiresult():
    userName = request.args.get('userName')
    if userName is None or userName == '':
        return 'Not authorized!'
    else:
        return render_template('bmiresult.html',
bmi_value=request.args.get('data'),
bmi_category=request.args.get('bmi_category'),
fitness_level=request.args.get('fitness'),
workout_plan=request.args.get('workout_plan'),
workOutDetail=request.args.get('workOutDetail'),userName=userName)

@app.route('/bmiresult', methods=['POST'])
def bmiresult_post():
    userName = request.form['userName']
    editUser = request.form['editUser']
    if userName is None or userName == '':
        return 'Not authorized!'
    else:
        return render_template('bmi.html', userName=userName,
editUser=editUser)

@app.route('/failed')
def failed():
    return 'Login Failed!'

@app.route('/userhistory')

```



```

def userhistory():
    userName = request.args.get('userName')
    editUser = request.args.get('editUser')

    if userName is None or userName == '':
        return 'Not authorized!'
    else:
        if editUser == "" and userName == "madhu":
            editUser = userName
            dbClient = pymongo.MongoClient("mongodb://localhost:27017/")
            dbCon = dbClient["local"]
            users = dbCon["userdetail"]
            userStr = ""
            for user in users.find():
                if editUser == user["name"]:
                    userStr += str(user["bmi"]) + '|' + user["bmicategory"]
                    + '|' + user["fitness"] + '|' + str(user["height"]) + '|' +
                    str(user["weight"]) + '~'
            return render_template('userhistory.html', userName=userName,
userDetail=userStr)

@app.route('/signup')
def signup():
    return render_template('signup.html')

@app.route('/signup', methods=['POST'])
def signup_post():
    userName = request.form['inputUserName']
    password = request.form['inputPassword']
    firstName = request.form['inputFirstName']
    lastName = request.form['inputLastName']
    gender = request.form['inputGender']
    dob = request.form['inputDob']
    phone = request.form['inputPhone']
    dbClient = pymongo.MongoClient("mongodb://localhost:27017/")
    dbCon = dbClient["local"]
    users = dbCon["users"]
    for user in users.find():
        if userName == user["name"]:
            return 'User already exists!'

    userInfo = { "name": userName, "password": password, "firstName":
firstName, "lastName": lastName, "gender": gender, "dob": dob,
"phone": phone, "bmi": "-", "bmicategory": "-", "fitness": "-",
"height": "-", "weight": "-", "workOuts": "-" }

```

```

        users.insert_one(userInfo)
        return redirect('/')

@app.route('/admin')
def admin():
    userName = request.args.get('userName')
    if userName is None or userName == '' or userName != "madhu":
        return 'Not authorized!'
    else:
        dbClient = pymongo.MongoClient("mongodb://localhost:27017/")
        dbCon = dbClient["local"]
        users = dbCon["users"]
        userStr = ""
        for user in users.find():
            userStr += user["name"] + '|' + user["firstName"] + '|' +
            user["lastName"] + '|' + user["gender"] + '|' + user["dob"] + '|' +
            user["phone"] + '|' + str(user["bmi"]) + '|' + user["bmicategory"]
            + '|' + user["fitness"] + '|' + str(user["height"]) + '|' +
            str(user["weight"]) + '~'

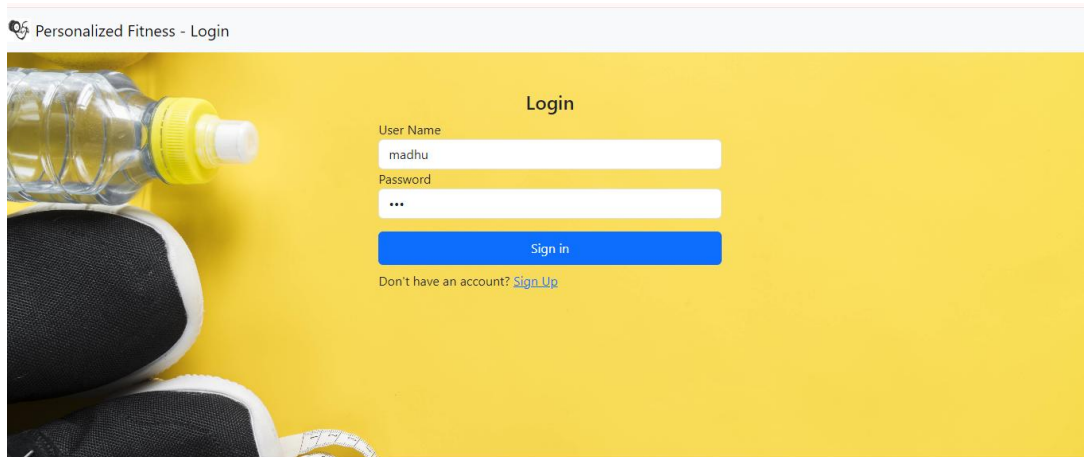
        return render_template('users.html',      userName=userName,
userList=userStr)

if __name__ == '__main__':
    app.run(debug=True)

```

A.1.2) SCREENSHOT:

ADMIN MODULE:



Personalized Fitness - Login

Login

User Name
madhu

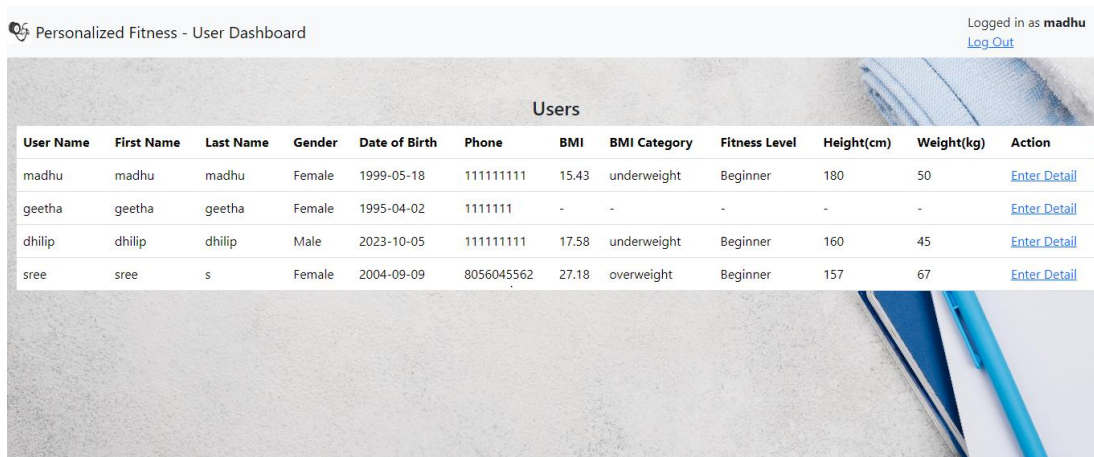
Password

[Sign in](#)

Don't have an account? [Sign Up](#)

Figure:2

ADMIN TRACKING USER DASHBOARD MODULE:



Personalized Fitness - User Dashboard

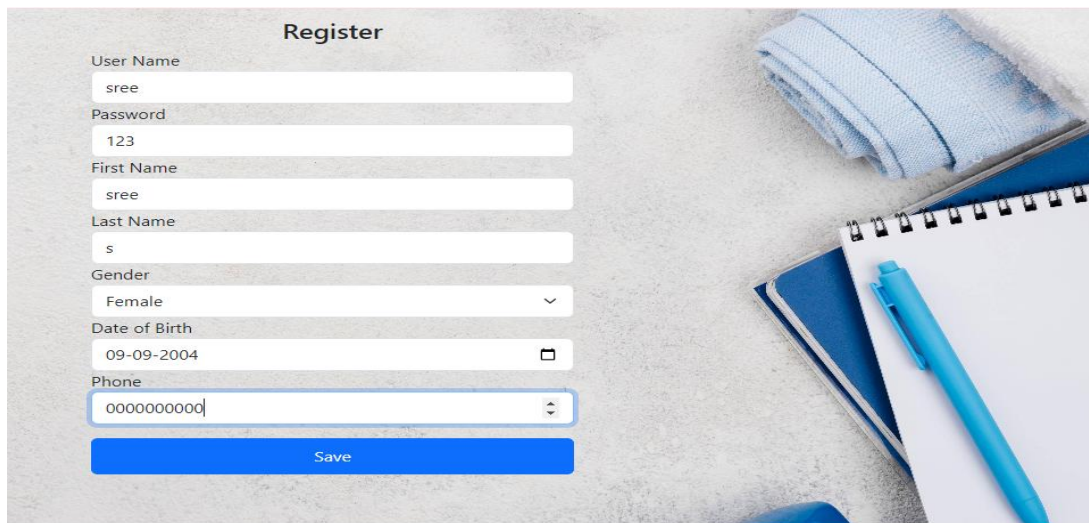
Logged in as **madhu**
[Log Out](#)

Users

User Name	First Name	Last Name	Gender	Date of Birth	Phone	BMI	BMI Category	Fitness Level	Height(cm)	Weight(kg)	Action
madhu	madhu	madhu	Female	1999-05-18	111111111	15.43	underweight	Beginner	180	50	Enter Detail
geetha	geetha	geetha	Female	1995-04-02	11111111	-	-	-	-	-	Enter Detail
dhilip	dhilip	dhilip	Male	2023-10-05	111111111	17.58	underweight	Beginner	160	45	Enter Detail
sree	sree	s	Female	2004-09-09	8056045562	27.18	overweight	Beginner	157	67	Enter Detail

Figure:3

USER SIGNUP MODULE:

A registration form titled "Register" is overlaid on a background image of a desk with a blue notebook, a blue pen, and a folded blue cloth. The form contains the following fields: "User Name" with the value "sree", "Password" with the value "123", "First Name" with the value "sree", "Last Name" with the value "s", "Gender" with a dropdown menu showing "Female", "Date of Birth" with the value "09-09-2004", and "Phone" with the value "0000000000". A blue "Save" button is at the bottom of the form.

Register

User Name
sree

Password
123

First Name
sree

Last Name
s

Gender
Female

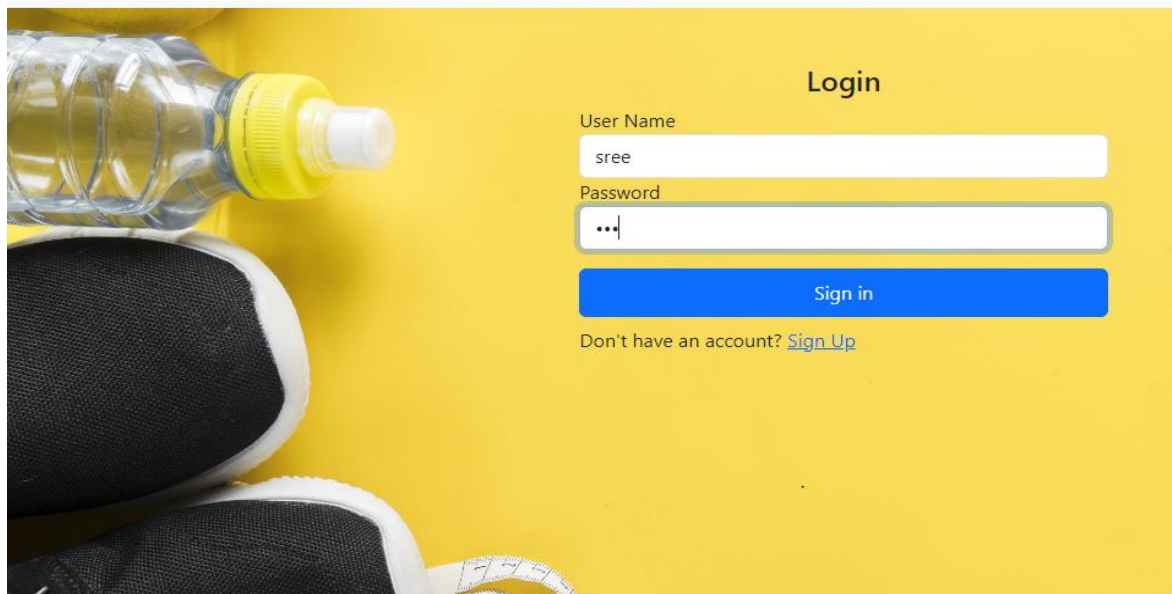
Date of Birth
09-09-2004

Phone
0000000000

Save

Figure:4

USER LOGIN PAGE

A login form titled "Login" is overlaid on a bright yellow background. To the left of the form are images of a clear water bottle with a yellow cap, a black sneaker, and a white measuring tape. The form contains the following fields: "User Name" with the value "sree" and "Password" with masked characters "...". A blue "Sign in" button is below the password field. Below the button is a link that says "Don't have an account? [Sign Up](#)".

Login

User Name
sree

Password
...

Sign in

Don't have an account? [Sign Up](#)

Figure:5

USER INPUT MODULE:

Personalized Fitness - Fill Detail

Logged in as **sree**
[Log Out](#)

Enter Detail

Height (cm)

157

Weight (kg)

50

Fitness Level

Beginner

View History

Save & Proceed

Back to Dashboard

© 2024 Copyright: **Personalized Fitness**

Figure:6

WORKOUT RECOMMENDATION MODLUE

Personalized Fitness - Recommended Workouts

Logged in as **sree**
[Log Out](#)

Recommended Workouts

Your BMI is: 27.18

You are overweight. It is recommended to lose weight.

Recommended sets and reps for BMI 27.18 (Beginner level):

- Pushups- 2 sets of 8 reps
- Pullups- 1 sets of 4 reps
- Walking- 40 minutes
- Squats- 2 sets of 8 reps

Go Back

Figure:7

USER PROFILE AND HISTORY MODULE:

Personalized Fitness - User Detail History

Logged in as **sree**
[Log Out](#)

History

BMI	BMI Category	Fitness Level	Height(cm)	Weight(kg)
20.28	normal	Beginner	157	50
27.18	overweight	Beginner	157	67

Figure:8

REFERENCE:

- S. Sadhasivam, M. S. Sarvesvaran, P. Prasanth and L. Latha, "Diet and Workout Recommendation Using 5ML", 2023 2nd International Conference on Advancements in Electrical Electronics Communication Computing and Automation (ICAECA), 2023.
- Deepanjali Chowdhury, Ahana Roy, Sreenivasan Ramasamy Ramamurthy and Nirmalya Roy, "CHARLIE: A Chatbot That Recommends Daily Fitness and Diet Plans", IEEE International Conference on Pervasive Computing and Communications Workshops, 2023.
- R. Sharma, S. Gupta and A. Kumar, "Machine Learning Approaches for Predicting Fitness Outcomes: A Review", IEEE Transactions on Emerging Topics in Computational Intelligence, 2022.
- T. Brown, J. Wilson and D. Smith, "Behavior Change Techniques in Mobile Fitness Applications: An Analysis of IEEE Xplore Literature", IEEE Consumer Electronics Magazine, 2022.
- M. H. Jarrahi, N. Gafinowitz, and G. Shin, "Activity trackers, prior motivation, and perceived informational and motivational affordances," Pers. Ubiquitous Comput., vol. 22, no. 2, pp. 433–448, 2020.