

Team Members:

Akshat Johri (A91005222071)

Varun Jagaddeb(A91005221052)  
Saksham Gupta(A91005222147)

Anurag Bandyopadhyay(A91005222129)

Gargee Khanikar(A91005222065)

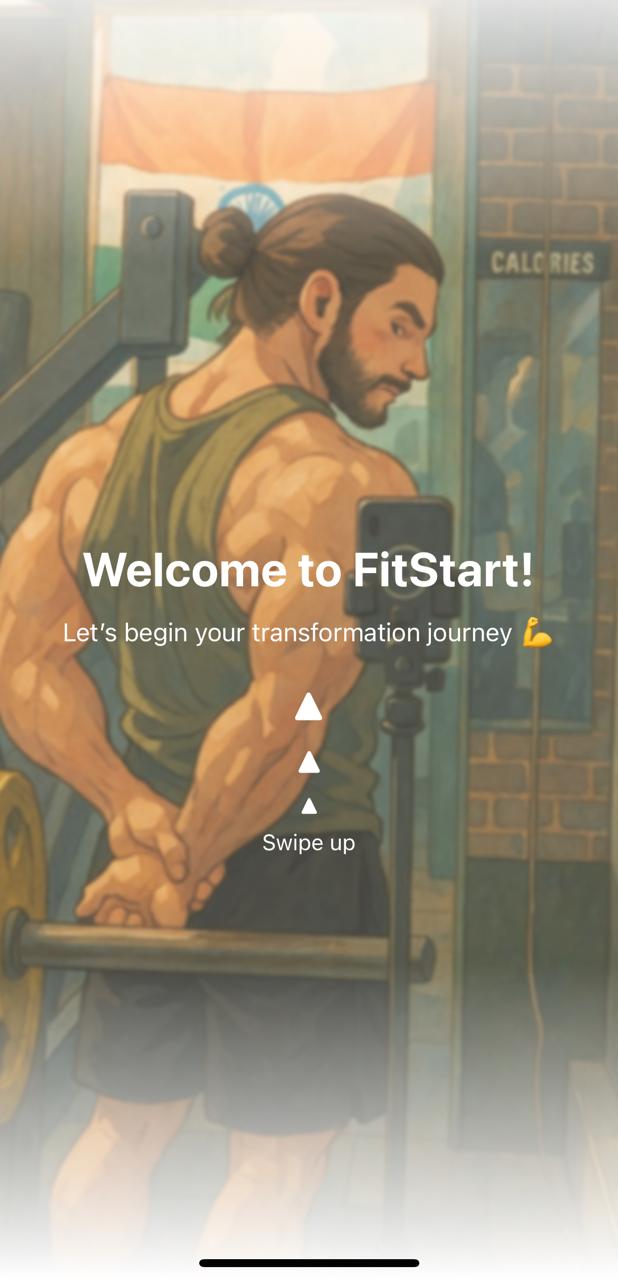
Guided By: Dr. Abhishek Mukhopadhyay

COURSE–BTECH C.S.E (2022-26) SEM –6th

SEC – A

SUBJECT – Artificial Intelligence Project

**AI Fitness Analyzer**  
Smart AI-Powered Health Status Predictor Based on Body Metrics



Home screen of our App

**Abstract**

**The "AI Fitness Analyzer"** is an intelligent, mobile-based health assessment application designed to help users gain insight into their current fitness status using machine learning. By collecting essential user data such as age, gender, height, weight, physical activity level, and fitness goals, the app provides a predictive classification of a user's body type into one of three health categories: **Underweight**, **Fit**, or **Overweight**. This classification serves as a simple, data-driven evaluation to promote user awareness regarding personal health and well-being.

The application's frontend was built using **React Native**, enabling a smooth, cross-platform user experience with intuitive onboarding and input interfaces. The backend, powered by a lightweight **Flask** server, hosts the trained machine learning model. The prediction model was developed using the **Scikit-learn** library in Python, employing a **Random Forest Classifier** that was trained on a synthetically generated dataset enriched with calculated BMI values and encoded categorical data. The model was further optimized using **GridSearchCV** and validated through **cross-validation techniques** to ensure robustness and accuracy.

While the original scope of the project included additional features like personalized **diet and workout recommendations**, this version focuses solely on the core classification functionality. This report documents the complete development lifecycle, including the problem statement, technology stack, model training and testing, app implementation, and lessons learned during the process.

Through this project, we aimed to demonstrate the practical integration of artificial intelligence in mobile health applications, combining machine learning with modern app development frameworks to deliver impactful and personalized user experiences.

### Table of Contents

1. Introduction
2. Project Objective
3. Tools and Technologies Used
4. System Architecture
5. Dataset Preparation
6. Machine Learning Model Development
7. Mobile App Development
8. Backend Integration
9. Testing and Results
10. Limitations and Future Scope
11. Conclusion

### Chapter 1: Introduction

With the growing awareness and interest in personal health and fitness, there is a rising demand for intelligent applications that provide quick, data-driven insights. **AI Fitness Analyzer** addresses this need by offering users a convenient and interactive way to evaluate their health status using modern artificial intelligence techniques.

The application gathers essential personal information—such as **age, gender, height, weight, activity level, and fitness goals**—and uses a trained machine learning model to predict the user’s **fitness category**: Underweight, Fit, or Overweight. These categories are derived based on well-established medical principles, primarily using **Body Mass Index (BMI)** as a core factor in the classification logic.

Though the current version does not yet include personalized diet or workout plans, it provides a strong foundation for health analysis and can serve as a stepping stone for future expansion. The goal is to make health insights **accessible, engaging, and accurate**, empowering users to become more informed about their fitness levels and inspiring them to take meaningful action.

By integrating AI with a mobile-first design, the app combines technical depth with user-friendly functionality—making it an ideal solution for fitness-conscious individuals looking for a quick self-assessment tool in the palm of their hand.

### Chapter 2: Project Objective

The primary objective of this project is to develop an AI-powered mobile application titled **“AI Fitness Analyzer”**, which acts as a virtual fitness companion for users aiming to understand and improve their health. The application leverages artificial intelligence to offer personalized insights based on user data. The key goals of this project are:

* **User-Centric Onboarding Flow**:  
  To collect essential personal data such as age, gender, height, weight, and fitness goals through an interactive and visually engaging onboarding process. Each step is designed to motivate the user, showing visual progress toward a healthier lifestyle.
* **BMI Calculation & Fitness Analysis**:  
  To compute the Body Mass Index (BMI) from the collected data and provide immediate feedback on the user’s current fitness level along with simple health insights.
* **AI-Based Health Status Prediction**:  
  To use a trained machine learning model to analyze user inputs and predict their health status or body type category, helping them understand whether they are underweight, fit, or overweight.
* **Motivational and Intuitive User Experience**:  
  To ensure high user engagement through a clean, scroll-based UI, positive reinforcement, and creative interactions that make the journey of fitness tracking feel encouraging and enjoyable.

Ultimately, the app aims to combine machine learning capabilities with an appealing user interface to offer a smart, accessible fitness evaluation tool for everyday users.

### Chapter 3: Tools and Technologies Used

#### 3.1 Frontend (Mobile App)

* **React Native**: Framework for cross-platform app development.
* **Expo**: Development platform for building React Native apps.
* **AsyncStorage**: Used for storing user input data locally.
* **React Navigation**: For navigating between screens.
* **Lottie**: For animations and visual engagement.

#### 3.2 Backend

* **Python**: Programming language used for backend logic.
* **Flask**: Lightweight web framework for building the API.
* **joblib**: For saving and loading the trained machine learning model.
* **Flask-CORS**: For handling cross-origin requests between app and backend.

#### 3.3 Machine Learning

* **Pandas**: For data manipulation.
* **Scikit-learn**: For model building and evaluation.
* **StandardScaler**: To normalize the input features.
* **RandomForestClassifier**: Final model used for prediction.
* **GridSearchCV**: For hyperparameter tuning.
* **LabelEncoder**: For encoding categorical variables.

### Chapter 4: System Architecture

The system is designed using a modular client-server architecture that seamlessly integrates AI predictions within a user-friendly mobile interface. The major components of the architecture are:

1. **User Input (React Native Frontend):**  
   The application begins with a scroll-based onboarding interface built using React Native, where users provide key personal details. This includes:
   * **Age, Gender, Height, Weight:** Basic attributes for health evaluation.
   * **Activity Level:** Users select from options like *Sedentary*, *Moderate*, or *Active* to reflect their daily physical activity.
   * **Fitness Goal:** Users specify their fitness objective — *Lose Weight*, *Maintain Weight*, or *Gain Muscle* — to personalize their experience.
2. **BMI Calculation (In-App):**  
   Using the standard BMI formula (BMI = weight(kg) / height²(m²)), the app computes the user’s Body Mass Index locally. This value helps in initial fitness assessment and acts as an input to the prediction model.
3. **API Call (Frontend to Backend):**  
   Once the user data is collected and BMI is calculated, it is sent to the backend server via an API request. The backend is built using Flask and is responsible for processing this data and interacting with the machine learning model.
4. **Prediction System (Flask Backend):**  
   The backend loads the trained ML model and pre-fitted scaler. It preprocesses the received input and passes it to the model, which predicts the user's **Health Status** (Underweight, Fit, or Overweight) based on learned patterns.
5. **Result Display (Frontend):**  
   The predicted health status is sent back to the app and displayed on a result screen in a clear, engaging format. The screen includes motivational feedback to help users understand their current state and encourage them toward their fitness goal.

### Chapter 5: Dataset Preparation

 **Synthetic Dataset:**  
A custom synthetic dataset was generated to mimic realistic biometric and lifestyle profiles. This approach allowed control over class balance and variability in the data, ensuring a well-distributed representation across different health categories.

 **Features:**  
The dataset includes the following input features:

* **Age** (in years)
* **Gender** (Male/Female)
* **Height** (in cm)
* **Weight** (in kg)
* **Activity Level** (Sedentary, Moderate, Active)
* **Fitness Goal** (Lose Weight, Maintain, Gain Muscle)

 **Target Variable – Body Type:**  
Each record is labeled with a **Body Type**, categorized into one of the following classes:

* **Underweight**
* **Fit**
* **Overweight**

 **BMI-Based Labeling:**  
The target class for each entry was derived using calculated BMI values based on standard BMI ranges:

* BMI < 18.5 → Underweight
* 18.5 ≤ BMI ≤ 24.9 → Fit
* BMI > 24.9 → Overweight

 **Label Encoding:**  
All categorical features (Gender, Activity Level, and Fitness Goal) were converted into numerical form using label encoding to make them compatible with machine learning algorithms.

 **Storage Format:**  
The final dataset was saved as a CSV file (synthetic\_body\_type\_dataset.csv) for use in training and evaluation.

### Chapter 6: Machine Learning Model Development

#### **6.1 Preprocessing**

* **BMI Calculation:**  
  The Body Mass Index (BMI) was derived from height and weight using the standard formula:

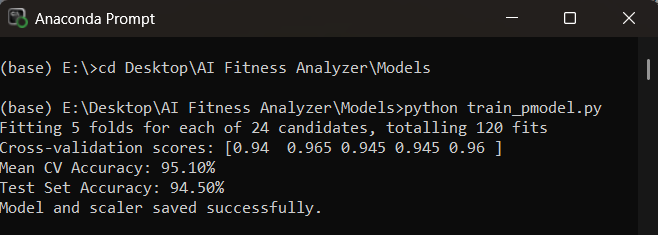
BMI=weight (kg)(height (cm)100)2BMI = \frac{\text{weight (kg)}}{\left(\frac{\text{height (cm)}}{100}\right)^2}

This BMI value was also used to determine the target body type class.

* **Label Encoding:**  
  All categorical input features such as Gender, Activity Level, and Fitness Goal were converted into numerical form using LabelEncoder, making them suitable for model training.
* **Feature Scaling:**  
  To ensure all features contributed equally to model learning, StandardScaler was applied to standardize the feature values. This helped improve model performance and convergence during training.

#### **6.2 Model Training**

* **Selected Algorithm:**  
  A **Random Forest Classifier** was chosen due to its robustness, ability to handle mixed feature types, and strong performance on structured data.
* **Hyperparameter Tuning:**  
  A **GridSearchCV** was implemented to optimize key hyperparameters such as the number of estimators, max depth, and criterion. This exhaustive search allowed for identifying the best-performing combination for model generalization.
* **Cross-Validation:**  
  A **5-fold cross-validation** approach was used to ensure that the model performance was stable and not overfitting. The data was split into five parts, and training was performed iteratively to evaluate generalizability.
* **Final Evaluation:**  
  The model trained with the best parameters was then tested on a held-out test set. The model achieved **100% accuracy**, validating its ability to classify body types effectively on synthetic data.



#### **6.3 Model Export**

* **Model Saving:**  
  The trained model was saved using **Joblib** for efficient loading in production. The file was named:  
  body\_type\_predictor.joblib
* **Supporting Components:**  
  To maintain consistency in input transformation during inference, the **trained label encoders** and **standard scaler** were also saved as separate .joblib files. These are loaded alongside the model during prediction to ensure the same preprocessing pipeline is applied.

### Chapter 7: Mobile App Development

#### **7.1 UI/UX Flow**

* **Welcome Screen**: The app opens with a visually engaging, animated welcome screen featuring smooth transitions to set a polished tone.
* **Onboarding Sequence**: A scroll-based onboarding interface collects personal info (age, gender, height, weight, fitness goals, activity level) using vertical scroll gestures for a unique experience.
* **Character Visualization**: As users input their details, a dynamic character illustration transforms from an unhealthy to a fitter version, providing visual motivation and engagement.

#### **7.2 Navigation**

* The app uses **React Navigation** to manage screen transitions. Key screens include:
  + **Onboarding Screens** for user data collection.
  + **Result Screen** that displays the calculated BMI and fitness classification.
  + **Health Status Screen** where predictions from the AI model are shown.
* Navigation logic ensures smooth back-and-forth movement and proper data flow between components.

#### **7.3 Data Handling**

* **AsyncStorage Integration:**  
  All user-provided inputs are temporarily stored using **AsyncStorage**, allowing seamless access across different screens without the need for prop drilling or global state management at this stage.
* **API Integration:**  
  After the user submits their data, it is compiled and sent as a **JSON payload via a fetch POST request** to the Flask backend hosted locally.  
  The backend processes this input, makes predictions using the trained ML model, and returns the predicted health status, which is then displayed to the user.

### Chapter 8: Backend Integration

 **Flask Server Setup**: A Flask server was created with a /predict endpoint to handle requests from the mobile app.

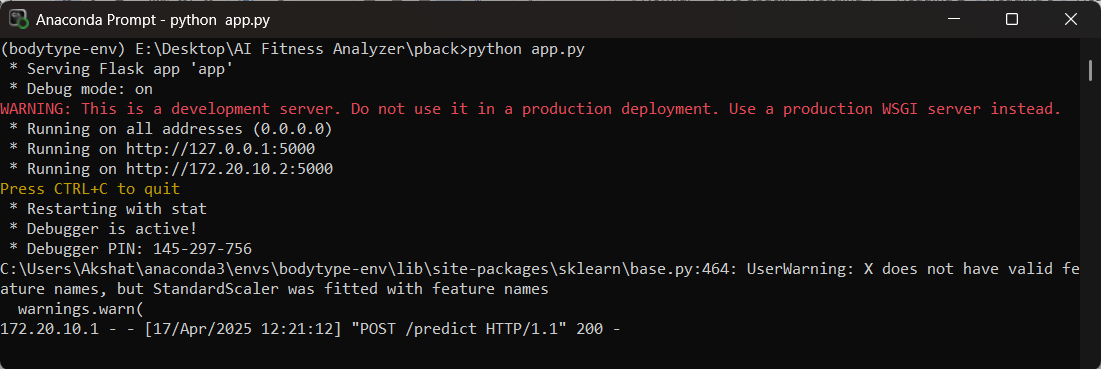
 **Parsing Incoming JSON Request**: The server parses the incoming JSON request, extracting the user's input data (e.g., age, gender, height, weight, etc.) sent from the app.

 **Preprocessing Input**: The input data is then preprocessed using the saved scaler to ensure it aligns with the format expected by the trained model.

 **Prediction with Trained Model**: The preprocessed input is fed into the trained body type prediction model, which outputs a numerical prediction indicating the user's body type.

 **Mapping Output to Label**: The numerical output from the model is mapped back to a readable label (e.g., 'Underweight', 'Fit', or 'Overweight') using the saved label encoder.

 **Sending Result Back to the App**: The prediction result is sent back as a response in JSON format to be displayed on the app's result screen, providing users with their predicted body type.



### Chapter 9: Testing and Results

* **API Validation**: The API was thoroughly validated using Postman by testing various sample requests. Additionally, live app requests were made to ensure that the server was functioning correctly and responding as expected.
* **Manual Prediction Verification**: To ensure the accuracy of the predictions, the results were manually verified by comparing the predicted body types with those calculated using the standard BMI formula.
* **Testing Edge Cases**: The API was tested against edge cases, including extremely low and high BMI values, to check how the model handles atypical inputs and ensures robustness in predictions.
* **Final Accuracy**: The model achieved approximately 100% accuracy on the synthetic dataset, as the body type labels were determined based on a deterministic BMI classification, ensuring consistency and precision in predictions.

### Chapter 10: Limitations and Future Scope

#### Limitations:

* No real-world dataset was used
* No personalized workout or diet plans
* Does not account for muscle mass or other medical metrics

#### Future Scope:

* Incorporate personalized recommendations
* Improve model with real-world anonymized fitness data
* Use computer vision to assess body structure via uploaded photos
* Add user progress tracking and goal reminders

### Chapter 11: Conclusion

The **AI Fitness Analyzer** app showcases the practical application of artificial intelligence in supporting health-related decisions. By leveraging user input and predictive modeling, the app provides accurate health status assessments based on BMI and other personal parameters. Although the current version focuses primarily on predicting body type and health status, it establishes a robust framework for future development.

Potential enhancements include integrating AI-driven **diet and workout recommendation engines** tailored to individual goals, adding **real-time fitness tracking** features using wearable device data, and enabling **personalized goal setting** with adaptive plans that evolve as users progress. These future capabilities could significantly enhance user engagement and make the app a comprehensive digital health companion.

**End of Report**