Major Project Report

Fitness Tacking System

Aravind Praveen - MST03-0086

Table of Contents

1. Introduction

The Fitness Tracking Web Application is designed to provide users with a personalized fitness monitoring experience. By integrating machine learning models, the app offers insights into users' health based on their input data and historical trends. It is built using the Django framework, with a focus on user interaction and data analytics to improve users' fitness journeys.

---

2. Objective

The primary objective of this project is to:

- Develop a platform where users can input fitness-related data.
- Utilize machine learning to analyze user data and provide fitness recommendations.
- Store and visualize historical fitness data for users.

---

3. System Overview

This application allows users to register, log in, and input their fitness data. Based on the provided inputs, the app calculates Body Mass Index (BMI), Basal Metabolic Rate (BMR), Total Daily Energy Expenditure (TDEE), and step distance. A machine learning model determines the user's fitness status (Fit/Not Fit) and provides feedback.

---

4. Technologies Used

- **Backend Framework**: Django (Python)
- **Frontend**: HTML, CSS
- **Database**: SQLite
- **Machine Learning**: Random Forest Classifier with Joblib
- **Other Libraries**: Django Forms, Messages Framework

---

5. Modules and Functionalities

5.1 User Management

- **Registration**: Users can create accounts using the UserRegistrationForm.
- **Login**: Authentication is performed using Django's authenticate() method.
- **Logout**: Users can securely log out.

5.2 Fitness Data Input

- Users can input data such as height, weight, step count, sleep duration, stress level, hydration level, and activity level using a form.

5.3 Fitness Analysis

- **Calculations**:
  - **BMI**: BMI = Weight (kg) / (Height (m))^2
  - **BMR**: Gender-based equations for caloric needs.
  - **TDEE**: Activity-level-based caloric requirements.
- **Machine Learning**: Predicts fitness status based on input data using a pre-trained Random Forest model.

5.4 Fitness History

- Displays historical fitness data for the logged-in user.
- Data is serialized into JSON and visualized on the frontend.

---

6. Database Design

FitnessData Model

Field Name	Data Type	Description
User	ForeignKey	Links fitness data to a user.
Height	FloatField	User's height (cm).
Weight	FloatField	User's weight (kg).
BMI	FloatField	Body Mass Index.
Step Count	IntegerField	Number of steps taken.
Calculated Distance	FloatField	Distance based on step count
Sleep Duration	FloatField	Hours of sleep.
Stress Level	IntegerField	Stress level (0-2).
Hydration Level	FloatField	Hydration level (liters).
Activity Level	CharField	User's activity level.
BMR	FloatField	Metabolic Rate.
TDEE	FloatField	Total Daily Energy Expenditure.
Status	CharField	Fitness status (Fit/Not Fit).
Message	TextField	Health-related feedback.
Created At	DateTimeField	Timestamp for data entry.

---

7. Machine Learning Integration

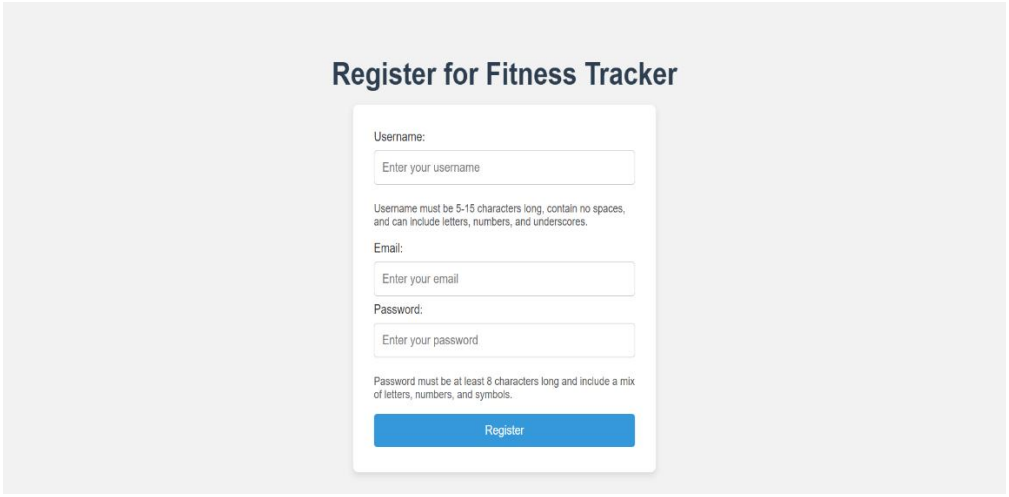
- **Model Used**: Random Forest Classifier.
- **Input Features**:
  - Height, Weight, BMI, Step Count, Calculated Distance, Sleep Duration, Stress Level, Hydration Level.
- **Output**: Fitness Status (0: Fit, 1: Not Fit).
- **Preprocessing**: Data scaled using a StandardScaler before prediction.
- **Storage**: Model and scaler saved as .pkl files using Joblib.

---

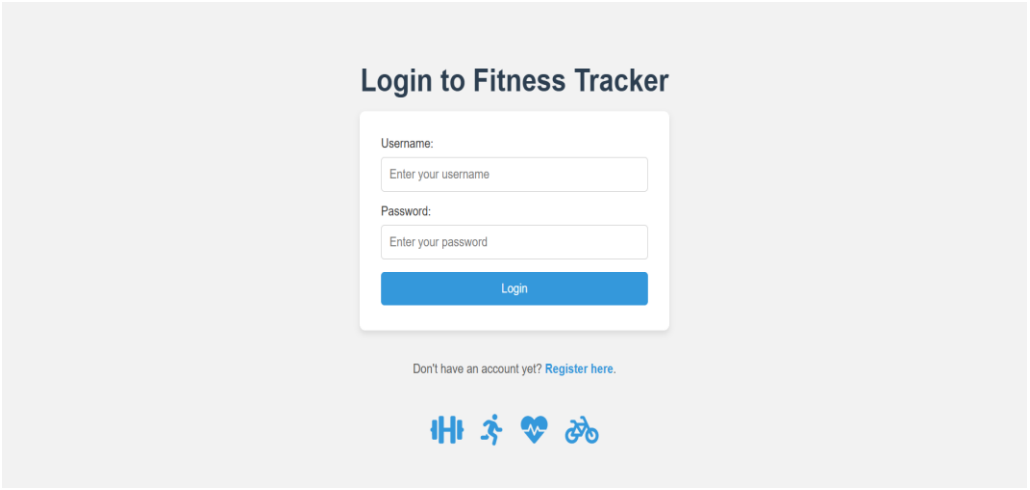
8. User Interface

Registration Page: Form for new user registration

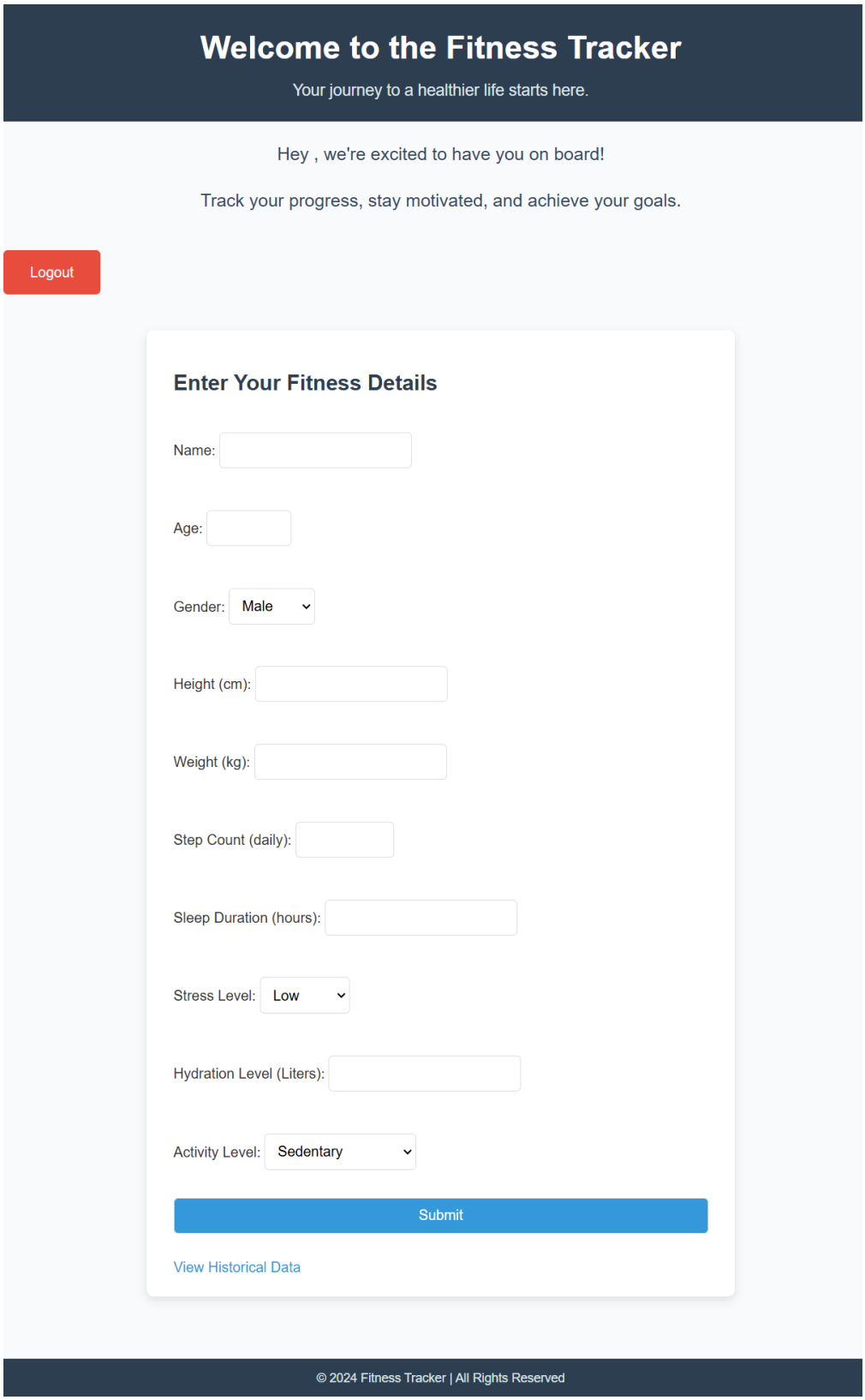
.



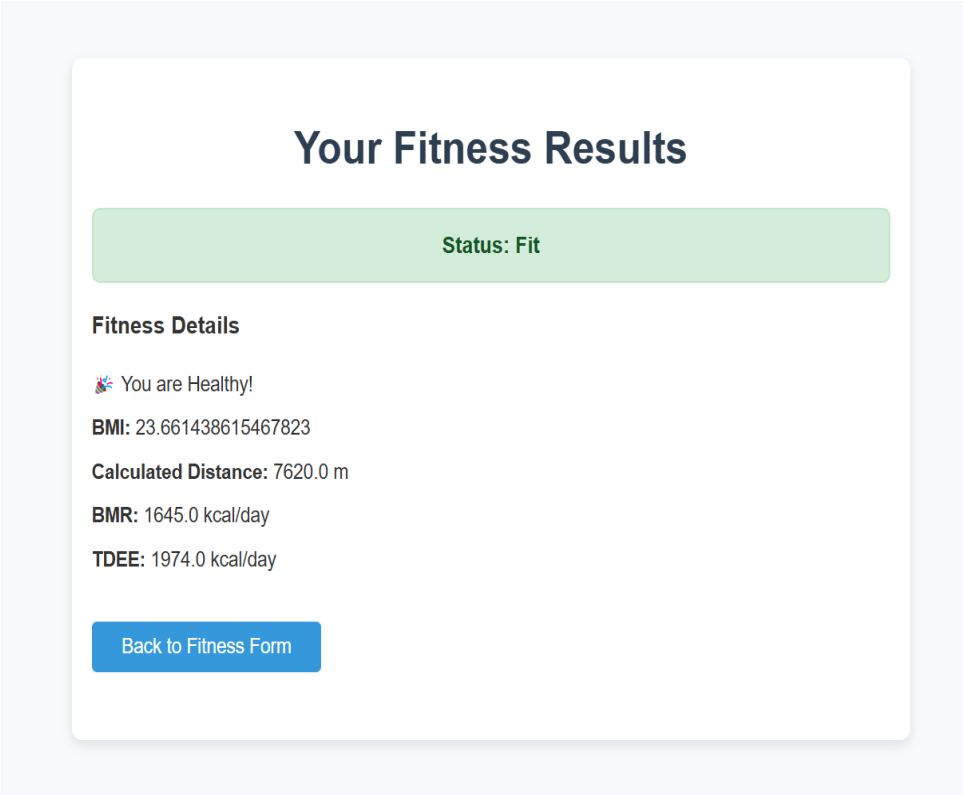
Login Page: Authentication for returning users.



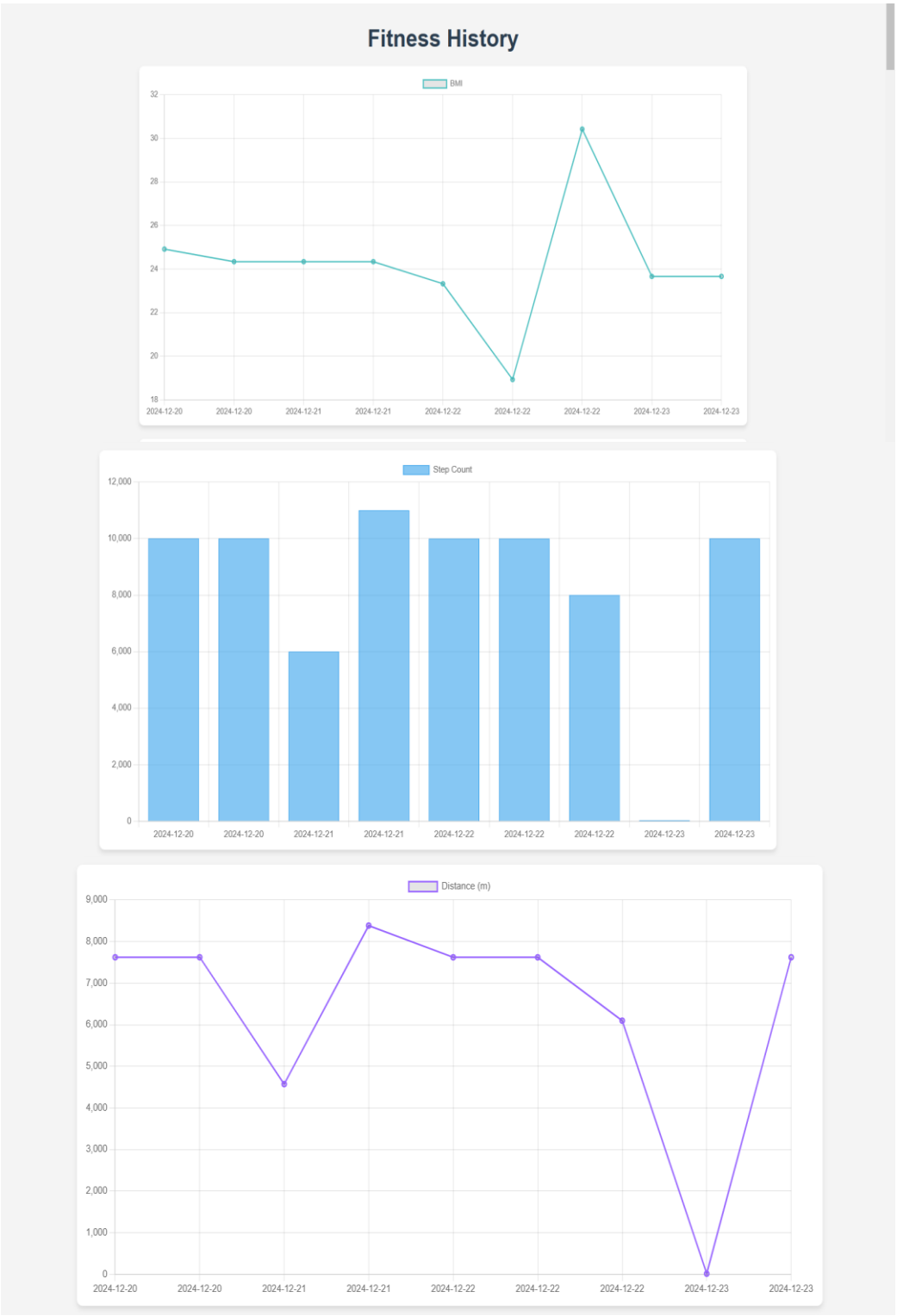
Fitness Form: Allows users to input fitness-related data.

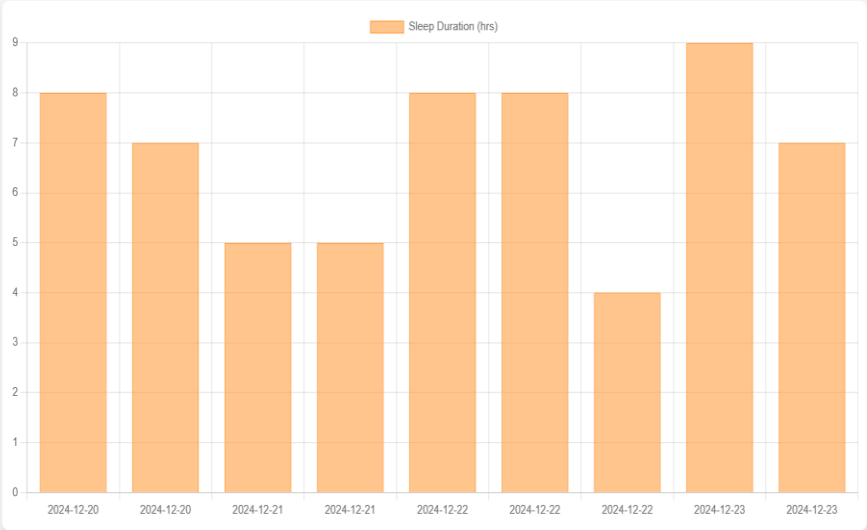


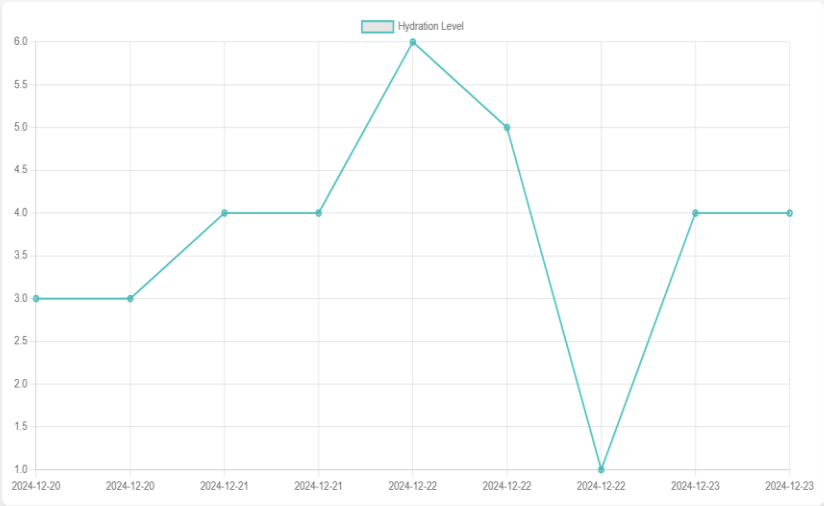
Results Page: Displays predictions and insights.

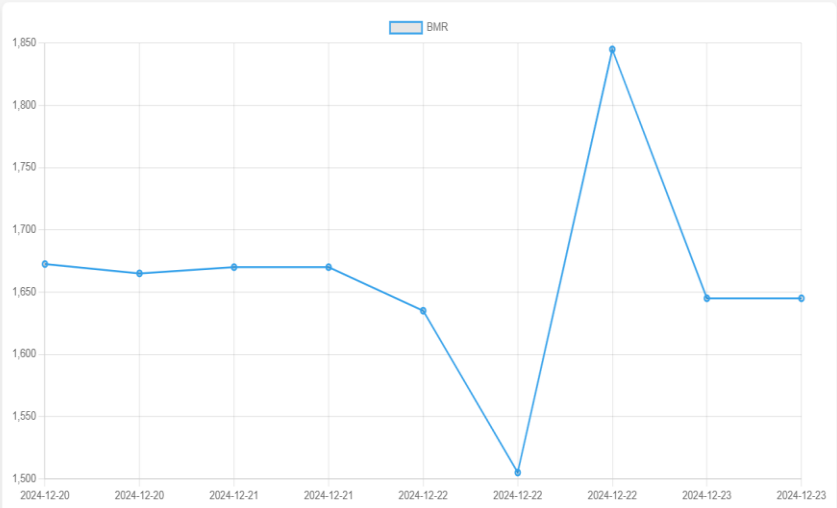


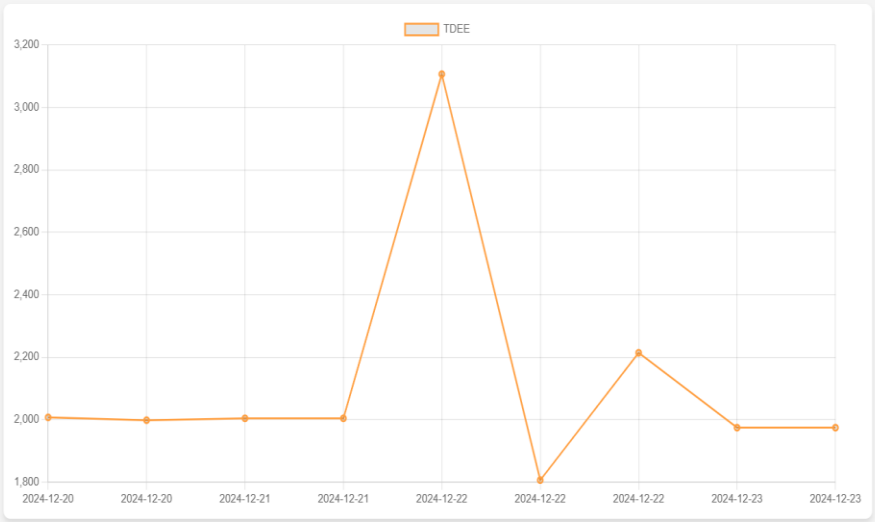
Fitness History: Visualizes historical data.




Sleep Duration (hrs)
9
8
7
6
5
4
3
2
1
0
2024-12-20
2024-12-20
2024-12-21
2024-12-21
2024-12-22
2024-12-22
2024-12-22
2024-12-23
2024-12-23


Hydration Level
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
2024-12-20
2024-12-20
2024-12-21
2024-12-21
2024-12-22
2024-12-22
2024-12-22
2024-12-23
2024-12-23


BMR
1,850
1,800
1,750
1,700
1,650
1,600
1,550
1,500
2024-12-20
2024-12-20
2024-12-21
2024-12-21
2024-12-22
2024-12-22
2024-12-22
2024-12-23
2024-12-23



Date	BMI	Step Count	Distance (m)	Sleep Duration (hrs)	Stress Level	Hydration Level	BMR	TDEE	Status
2024-12-20	24.913494809688583	10000	7620	8	Low	3	1672.5	2007	Fit
2024-12-20	24.337479718766904	10000	7620	7	Low	3	1665	1998	Fit
2024-12-21	24.337479718766904	6000	4572	5	Low	4	1670	2004	Not Fit
2024-12-21	24.337479718766904	11000	8382	5	Low	4	1670	2004	Not Fit
2024-12-22	23.323418063818284	9999	7619.238	8	High	6	1635	3106.5	Not Fit
2024-12-22	18.92915089237426	9999	7619.238	8	Low	5	1505	1806	Fit
2024-12-22	30.42184964845863	8000	6096	4	Low	1	1845	2214	Not Fit
2024-12-23	23.661438615467823	20	15.24	9	Low	4	1645	1974	Not Fit
2024-12-23	23.661438615467823	10000	7620	7	Low	4	1645	1974	Fit

Go Back

9. Testing and Validation

- **Unit Testing**: Tested individual functionalities like user registration, login, and data saving.
- **Integration Testing**: Ensured seamless interaction between forms, database, and machine learning model.
- **Validation**: Verified model predictions against known inputs.

---

10. Challenges and Solutions

Challenges

1. **Handling Invalid Inputs**: Users may input incorrect data.
2. **Model Integration**: Compatibility issues between the ML model and Django.
3. **Stress Level Encoding**: Converting categorical stress levels into numerical values.

Solutions

1. Implemented form validation to handle invalid inputs.
2. Used standardized formats for saving and loading ML models.
3. Mapped stress levels to numerical values (High: 0, Medium: 1, Low: 2).

---

11. Deployment

The Fitness Tracking Web Application is deployed on **Railway**, which allows for seamless deployment and scaling of the application. Railway provides a simple platform to deploy the Django app and manage its services efficiently. Key steps for deployment include:

1. **Railway Setup**: Created a project on Railway, linked it to the GitHub repository.
2. **Environment Variables**: Configured necessary environment variables for database access, Django settings, and other configurations.
3. **Deployment**: Deployed the app using Railway's simple push-to-deploy process, which automatically builds and hosts the application.
4. **SSL Setup**: SSL certificates were automatically handled by Railway for secure HTTPS connections.
5. **Performance Monitoring**: Railway provides tools to monitor app performance and scalability as traffic grows.

---

12. Code Overview

The codebase of the project is structured as follows:

- **Django Views**: Handles requests and responses, including fitness data input, analysis, and rendering of results.
- **Machine Learning Integration**: The Random Forest model is loaded, preprocessed, and used to predict fitness status based on user data.
- **Forms**: Utilizes Django forms for data input, ensuring proper validation.
- **Models**: Manages user and fitness data through Django models, which are stored in the SQLite database.

---

13. Conclusion

The Fitness Tracking Web Application successfully integrates Django and machine learning to provide personalized fitness insights. The app enables users to track their fitness journey and receive actionable feedback to improve their health. The use of machine learning adds an intelligent layer to the application, enhancing user engagement and providing deeper insights into personal health data.

---

14. Future Scope

- **Enhanced Predictions**: Train the ML model with a larger dataset for better accuracy.
- **Data Visualization**: Add graphical representations of historical trends.
- **Mobile App Integration**: Develop a companion mobile app for easier data entry.
- **Real-Time Monitoring**: Integrate wearable devices for live fitness tracking.

---

15. Application Link

You can access the Fitness Tracking Web Application using the following link:

APP_LINK
