INDUSTRIAL ORIENTED MINI PROJECT

Report

On

RHYTHM RESTORE :

A SMART SYSTEM FOR CIRCADIAN BALANCE AND SLEEP WELLNESS

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

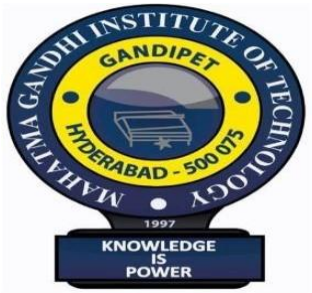
Mahareddy Megha Reddy - 22261A1238

Rohan Siddharth - 22261A1250

Under the guidance of

Dr. Ch. PremKumar

Associate Professor, Department of IT



DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY
(AUTONOMOUS)

(Affiliated to JNTUH, Hyderabad; Eight UG Programs Accredited by NBA; Accredited by NAAC with 'A++' Grade)

Gandipet, Hyderabad, Telangana, Chaitanya Bharati (P.O), RangaReddy District, Hyderabad– 500075, Telangana

2024-2025

CERTIFICATE

This is to certify that the **Industrial Oriented Mini Project** entitled **RHYTHM RESTORE A SMART SYSTEM FOR CIRCADIAN BALANCE AND SLEEP WELLNESS** submitted by **Mahareddy Megha Reddy (22261A1238), Rohan Siddharth (22261A1250)** in partial fulfillment of the requirements for the Award of the Degree of Bachelor of Technology in Information Technology as specialization is a record of the bona fide work carried out under the supervision of **Dr. Ch. PremKumar**, and this has not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Supervisor:

Dr. Ch. PremKumar
Associate Professor
Dept. of IT

IOMP Supervisor:

Dr. U. Chaitanya
Assistant Professor
Dept. of IT

EXTERNAL EXAMINAR

Dr. D. Vijaya Lakshmi
Professor and HOD
Dept. of IT

DECLARATION

We hear by declare that the **Industrial Oriented Mini Project** entitled **RHYTHM RESTORE A SMART SYSTEM FOR CIRCADIAN BALANCE AND SLEEP WELLNESS** is an original and bona fide work carried out by us as a part of fulfilment of Bachelor of Technology in Information Technology, Mahatma Gandhi Institute of Technology, Hyderabad, under the guidance of **Dr.Ch. PremKumar , Associate Professor**, Department of IT, MGIT.

No part of the project work is copied from books /journals/ internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by us and not copied from any other source.

Mahareddy Megha Reddy - 22261A1238

Rohan Siddharth - 22261A1250

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the **Industrial Oriented Mini Project**.

We would like to express our sincere gratitude and indebtedness to our Internal supervisor **Dr. Ch. PremKumar , Associate Professor**, Dept. of IT, who has supported us throughout our project with immense patience and expertise.

We are also thankful to our honourable Principal of MGIT **Prof. G. Chandramohan Reddy** and **Dr. D. Vijaya Lakshmi**, Professor and HOD, Department of IT, for providing excellent infrastructure and a conducive atmosphere for completing this **Industrial Oriented Mini Project** successfully.

We are also extremely thankful to our IOMP supervisor **Dr. U. Chaitanya**, Assistant Professor, Department of IT, and senior faculty **Mrs. B. Meenakshi** Department of IT for their valuable suggestions and guidance throughout the course of this project.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our families for their support all through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support for completion of this work.

Mahareddy Megha Reddy - 22261A1238

Rohan Siddharth -22261A1250

ABSTRACT

Effective management of sleep patterns is crucial for improving overall wellness, with individualized recommendations playing a significant role in achieving optimal health. The RhythmRestore project leverages machine learning techniques to provide personalized lifestyle recommendations based on user input, focusing on improving sleep quality, mood, stress levels, and other factors impacting well-being. By analysing data related to sleep time, screen time, meals skipped, mood, and work-related stress, the system offers tailored suggestions on diet, exercise, and stress management.

The system uses a combination of supervised learning algorithms, such as decision trees and ensemble methods, to predict users' ideal sleep routines and suggest strategies for improving health outcomes. Evaluations are based on accuracy metrics like mean squared error (MSE) and user feedback on recommendation effectiveness. The model consistently demonstrated its ability to provide personalized, actionable advice, empowering users to make informed decisions about their lifestyle.

By offering targeted guidance for wellness optimization, RhythmRestore aims to enhance users' overall health by addressing common sleep and lifestyle challenges. This approach can help users develop healthier routines, improve their productivity, and reduce the negative impacts of poor sleep and stress on daily life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.INTRODUCTION

1.1 MOTIVATION

In today’s fast-paced world, wellness and mental health have become increasingly important aspects of overall well-being. Poor sleep, stress, and unhealthy lifestyle choices can have a significant impact on a person's health, productivity, and quality of life. Despite the growing awareness around these issues, many individuals struggle to maintain a balanced lifestyle due to a lack of personalized guidance and actionable recommendations.

The RhythmRestore project was conceived to address this gap by leveraging data-driven insights to provide personalized wellness recommendations that help individuals optimize their sleep, manage stress, and improve their overall lifestyle. Current wellness apps often provide generalized advice that may not account for the unique needs of each user. This project aims to bridge that gap by using machine learning algorithms to analyse factors such as sleep patterns, mood, work stress, screen time, and meals skipped, generating tailored recommendations to help users develop healthier routines.

Given the rapid pace of life and constant changes in individual habits, a dynamic, adaptive recommendation system is crucial. The RhythmRestore system is designed to adjust based on real-time inputs, offering practical advice on sleep, diet, workouts, and stress management. This personalized approach is key to empowering users to make informed decisions about their health, ultimately improving their overall well-being and productivity.

1.2 PROBLEM STATEMENT

In today’s fast-paced world, many individuals struggle to maintain a balanced and healthy lifestyle due to factors such as poor sleep, high stress, and inconsistent habits. While there are various wellness and health tracking apps available, they often provide generalized advice that fails to account for individual differences in lifestyle, preferences, and specific health challenges. As a result, users are left with one-size-fits-all recommendations that may not be effective in improving their well-being.

The problem is further exacerbated by the lack of systems that dynamically adapt to users' changing lifestyles and provide personalized, real-time guidance. Existing solutions in the

wellness space often lack the ability to incorporate multiple user factors like mood, work stress, screen time, and meals skipped to generate tailored recommendations. This leads to a disconnect between users' needs and the advice offered, reducing the effectiveness of these systems in promoting long-term wellness.

This project seeks to address these challenges by developing a personalized wellness recommendation system, RhythmRestore, that leverages machine learning to provide actionable insights based on individual lifestyle data. The proposed system will collect inputs such as sleep patterns, stress levels, screen time, and food intake to generate customized advice on diet, exercise, and mental health management. By using real-time data and adaptive learning techniques, RhythmRestore aims to offer accurate, context-specific recommendations that can evolve with users' changing habits. Ultimately, the project seeks to empower users to make informed decisions, leading to improved health, better sleep, and reduced stress.

1.3 EXISTING SYSTEM

Existing systems for sleep tracking and wellness management primarily focus on monitoring sleep patterns and providing basic insights into sleep quality. Devices such as Fitbit, Apple Watch, and Oura are commonly used to track various sleep metrics, including sleep duration, sleep cycles, and stages such as deep sleep and REM sleep. These devices are equipped with sensors that measure movement and heart rate, offering a broad overview of the user's sleep behavior.

Mobile apps like Sleep Cycle, Pillow, and Calm also provide sleep tracking functionalities, using algorithms to analyze sleep data collected from wearables or smartphones. These apps aim to provide users with insights on their sleep quality and recommend optimal sleep times, but the recommendations are generally static and not personalized to individual needs.

In addition to sleep-tracking, several wellness apps focus on diet and exercise management, such as MyFitnessPal and Lose It!, which help users track food intake and set fitness goals. However, these systems typically work independently and do not integrate or consider other wellness factors like stress, screen time, and emotional well-being, which also play crucial roles in sleep quality.

These systems generally focus on providing data analysis and basic recommendations but lack the capability to offer personalized, dynamic wellness suggestions based on a holistic view of the user's lifestyle.

.

1.3.1 LIMITATIONS

- **Adaptability to Dynamic User Behaviour**: While RhythmRestore uses real-time data to provide personalized recommendations, fluctuations in a user's behaviour, such as sudden changes in sleep patterns, diet, or stress levels, may challenge the system's ability to consistently adapt. This could potentially impact the accuracy and effectiveness of the recommendations during periods of significant lifestyle changes.
- **Data-Driven Predictive Horizon**: The system primarily focuses on short-term recommendations, offering guidance based on immediate factors like daily sleep, mood, and stress. While this is useful for daily optimization, users looking for long-term lifestyle adjustments may find these recommendations less comprehensive over extended periods.

1.4 PROPOSED SYSTEM

The proposed system for **RhythmRestore** aims to enhance personalized wellness recommendations by dynamically adapting to user inputs and improving the accuracy of suggestions over time. A key feature of the system is its ability to continuously update user data based on daily inputs such as sleep patterns, mood, stress levels, screen time, and meals skipped. This real-time data integration ensures that the recommendations provided are always relevant and reflective of the user's current lifestyle.

To enhance the accuracy and relevance of the recommendations, RhythmRestore employs a hybrid approach using machine learning models to analyze and predict the user's ideal wellness routine. By combining techniques such as decision trees, ensemble methods, and regression models, the system aggregates insights from multiple algorithms to offer more balanced and reliable advice. This mixed-model approach allows the system to generate

personalized guidance that adjusts to the user's changing habits and well-being needs over time.

Additionally, the system utilizes a comprehensive dataset that includes historical user data to identify long-term trends and patterns in sleep and wellness. By training on diverse data sets, RhythmRestore can improve the overall performance of its recommendations and provide insights that are more accurate, taking into account both short-term and long-term health goals.

Through the use of real-time data updates, a hybrid modeling approach, and the integration of extensive user data, RhythmRestore aims to offer dynamic and personalized wellness guidance. This approach ensures that users receive relevant recommendations tailored to their evolving health needs, helping them improve sleep quality, manage stress, and achieve a balanced lifestyle.

1.4.1 ADVANTAGES

• **Real-Time Adaptability**: The system updates in real time based on daily inputs such as sleep data, mood, stress levels, and other lifestyle factors. This ensures that the recommendations stay aligned with the user's current wellness state, enhancing the system's responsiveness to immediate changes in their routine and well-being.

• **Holistic Wellness Recommendation**: RhythmRestore offers dynamic, personalized recommendations for sleep, stress management, diet, and exercise, drawing from multiple machine learning models. By integrating insights from various models, the system adapts to the user's evolving habits and health needs, providing more accurate, individualized suggestions for improving overall wellness.

• **Increased Reliability**: The system leverages a constantly updating dataset, which includes both short-term user inputs and long-term health trends. By continuously incorporating real-time data, RhythmRestore ensures that recommendations are based on the most up-to-date and comprehensive information, reducing the risk of relying on outdated or generalized advice.

1.5 OBJECTIVES

- Improve sleep quality and manage sleep disorders through personalized, actionable recommendations delivered via web and SMS notifications.
- Apply cybernetic health principles to dynamically adapt based on user behaviour and lifestyle patterns.
- Deliver tailored interventions that align with individual lifestyle habits, mood states, and stress levels.
- Enable continuous learning within the system to refine and enhance recommendations over time through feedback.
- Send personalized health tips via SMS, ensuring users receive timely guidance even without opening the app.

1.6 HARDWARE AND SOFTWARE REQUIREMENTS

1.6.1 SOFTWARE REQUIREMENTS

•**Software**:

The system is developed using **VS Code** as the integrated development environment (IDE) for Python. VS Code provides a robust, lightweight environment for writing, testing, and debugging the code required for machine learning models and backend services.

•**PrimaryLanguage**:

The project is primarily developed in **Python**, which is ideal for machine learning, data analysis, and web development. Python is used to handle data processing, model building, and recommendation logic, with libraries like **pandas**, **numpy**, **TensorFlow**, and **scikit-learn** for processing and model training.

•**FrontendFramework**:

The frontend is built using **Streamlit**, a popular Python library for creating interactive web applications. Streamlit is used to develop the user interface, allowing users to interact with the system and receive personalized recommendations based on their data inputs.

• **Backend Framework**:

The backend system and data processing tasks are written in **Python** and run through **VS Code**. The logic for personalized wellness recommendations, including machine learning model inference, is handled on the backend.

• **Database**:

For data storage, the system uses **SQLite3**, a lightweight database system that is well-suited for storing user data, historical input, and recommendation results. It's efficient for small to medium-scale applications.

• **Frontend Technologies**:

The user interface uses:

- **HTML**: For structuring the web pages.
- **CSS**: For styling and layout.
- **JavaScript**: To enable dynamic features and interactivity.
- **Bootstrap 4**: For responsive design, ensuring the application is usable across devices.

1.6.2 HARDWARE REQUIREMENTS

OperatingSystem:

The system is designed to run on **Windows** and **Mac** operating systems, providing compatibility with all the development tools used in the project.

Processor:

A processor with at least an **Intel i5** or higher is recommended to handle data processing, model execution, and real-time recommendations efficiently.

RAM:

A minimum of **8GB RAM** is required to efficiently manage the system's real-time data inputs, model inference, and multi-tasking requirements. For better performance, higher RAM capacity is recommended.

1. LITERATURE SURVEY

C. della Monica et al. explored a comprehensive protocol for monitoring sleep and circadian rhythms using digital technology, focusing on older adults and individuals with dementia. It emphasizes the need for real-time tracking of sleep patterns and circadian disruptions, which are crucial for providing personalized sleep recommendations. The protocol also highlights the importance of combining multiple technologies, such as wearables and mobile applications, for improved accuracy. However, the study's applicability is limited to specific populations, and it calls for better integration with other wellness systems for a more holistic approach to sleep improvement [1].

Wu et al. compared various commercial wearable devices for circadian rhythm monitoring, focusing on heart rate, activity levels, and core body temperature. The study shows the efficacy of wearables in capturing data on sleep and activity patterns, essential for circadian rhythm prediction. Despite this, the approach depends heavily on wearable devices, limiting its flexibility for users who cannot or prefer not to use them. Furthermore, real-time integration with other health management systems, such as diet and stress tracking, remains an underdeveloped area in this research [2].

A. Nguyen et al. research investigates the use of real-time acoustic stimulation to improve sleep. The approach involves closed-loop systems that provide personalized audio feedback during different stages of the user's sleep cycle. While this system demonstrates potential in enhancing sleep quality, it requires continuous monitoring and may not suit all users. The study suggests a need for further personalization options that account for factors like mood and work-related stress, which can significantly impact sleep patterns [3].

Rostaminia et al. introduce a novel all-textile eye mask for non-invasive sleep monitoring that senses brain activity and physiological signals. The eye mask provides detailed insights into sleep quality without causing discomfort. However, the mask's usability for extended periods is limited, and the study calls for more research on the long-term comfort and effectiveness of wearable devices. Additionally, the integration of this sleep-monitoring technology with real-time feedback systems for personalized sleep improvement is still an area of research [4].

Table 2.1 Literature Survey of RhythmRestore

Ref	Author& year of publication	Journal / Conference	Method / Approach	Merits	Limitations	Research Gap
[1]	C. della Monica et al., 2024	Digital Health (IEEE)	Protocol using wearables & mobile apps for sleep/circadian monitoring in older adults & dementia patients	Comprehensive protocol; digital integration; clinical focus	Limited to older/dementia patients; lacks general applicability	Generalize to wider population including youth and healthy individuals
[2]	F. Wu et al., 2024	IEEE Transactions on IoT	Comparative study of commercial wearable devices (smartwatches, fitness bands) for physiological tracking	Covers multiple physiological metrics; real-world device evaluation	High dependency on wearables; accessibility concerns	Explore non-wearable or passive sensing alternatives
[3]	A. Nguyen et al., 2022	Nature Scientific Reports	Closed-loop acoustic stimulation system that adapts based on sleep stages to improve sleep quality	Personalized feedback; real-time monitoring	Continuous monitoring not suitable for all; noise sensitivity concerns	Develop silent/alternative feedback systems for sensitive users
[4]	S. Rostaminia et al., 2021	IEEE Sensors Journal	All-textile smart eye mask capturing brain & physiological signals in a non-invasive way	Comfortable, non-invasive sleep tracking	Long-term comfort concerns; device availability	Improve long-term usability; make device commercially accessible

2. ANALYSIS AND DESIGN

The field of wellness and sleep improvement faces numerous challenges, particularly in offering personalized recommendations that adapt to the dynamic and diverse needs of individuals. This project aims to develop an advanced wellness recommendation system, **RhythmRestore**, designed to address these challenges by leveraging real-time data integration, machine learning techniques, and personalized recommendation models. The system is designed to provide dynamic recommendations for sleep patterns, mood management, exercise, and nutrition, enabling users to optimize their health and well-being based on data-driven insights.

The core of **RhythmRestore** lies in its ability to collect and integrate real-time data from various sources, such as user input on sleep time, mood, stress levels, screen time, and meals skipped. By continuously gathering this information, the system ensures that its recommendations are tailored and up-to-date. This real-time adaptability is crucial in wellness, as lifestyle factors like sleep and stress can change quickly, impacting an individual's health.

To provide personalized recommendations, the system employs a hybrid approach using machine learning models. The system integrates multiple algorithms, such as **Decision Trees** for identifying patterns in user behavior, **Logistic Regression** for understanding the relationships between variables, and **Ensemble Methods** (like Voting Regressors) to combine the outputs of various models for more accurate and robust predictions. This mixed-model approach ensures that the recommendations account for multiple aspects of the user's lifestyle, providing a holistic solution for improving sleep quality and overall health.

3.1 MODULES

User Interaction Module

Allows users to input data related to their sleep, mood, stress, and other lifestyle factors, and receive personalized recommendations. It enables the user to track their sleep patterns, mood variations, and other wellness data.

• **Input**: User inputs such as sleep time, mood, stress levels, screen time, meals skipped, and additional lifestyle data.
• **Output**: Personalized recommendations for improving sleep quality, managing stress, and adjusting lifestyle choices (such as diet and exercise). Displays recommendations and progress in an easy-to-read format, such as daily suggestions, weekly goals, and progress graphs.

Preprocessing Module

Collects and processes real-time user input data and historical data related to sleep and lifestyle factors. The module processes this data to generate relevant features for analysis, such as sleep duration, mood variations, and stress levels over time.
• **Input**: User data inputs (e.g., daily sleep patterns, mood, screen time, stress levels).
• **Output**: Processed data ready for model prediction, including calculated features like average sleep time, stress levels, and mood fluctuations.

Prediction Module

Uses machine learning models (e.g., decision trees, logistic regression, ensemble methods) to predict personalized wellness recommendations based on user data. The model outputs tailored suggestions to help users improve their sleep, diet, exercise, and overall wellness.
• **Input**: Processed user data (e.g., sleep patterns, stress levels, meals skipped, screen time).
• **Output**: Predicted wellness recommendations, including suggestions for sleep improvement, diet adjustments, stress management, and lifestyle changes, along with their effectiveness over a specified time period.

VisualizationModule

Displays the predicted wellness recommendations in a user-friendly interface, such as progress graphs, charts, and milestone trackers. Helps the user visualize their improvement over time and compare their progress with the system's suggestions.
• **Input**: Predicted wellness recommendations, historical user data.
• **Output**: Visual representation of wellness data through charts and graphs, showcasing progress over time (e.g., sleep improvement, stress reduction).

3.2 ARCHITECTURE

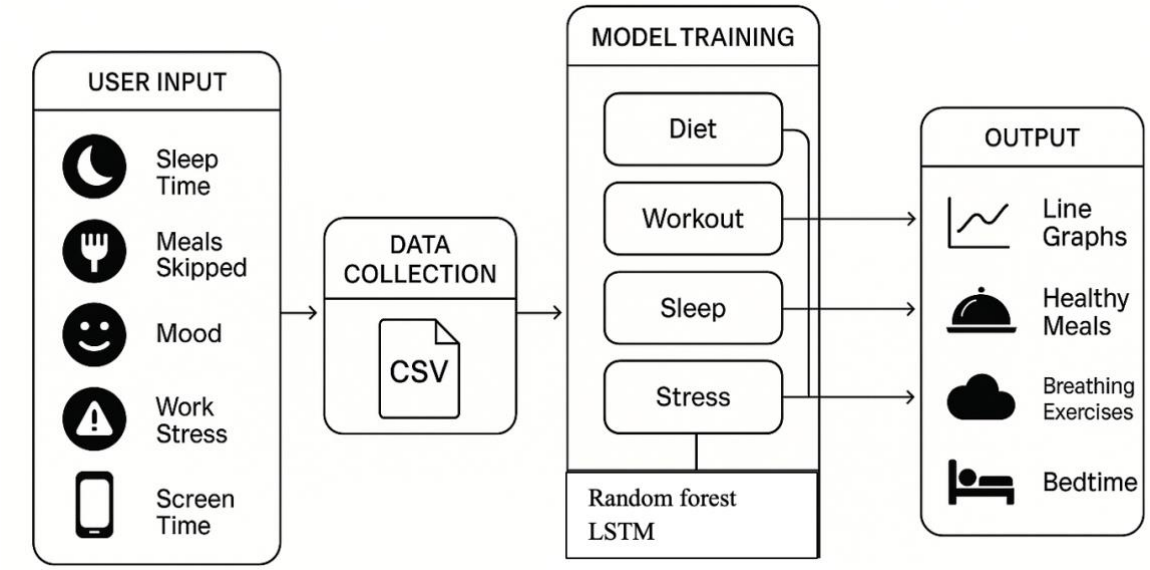


Fig. 3.2.1 Architecture of Sleep wellness Recommendation

The architecture of the system, as shown in Fig. 3.2.1, is designed for providing personalized wellness recommendations. The system begins with the Data Collection component, which collects real-time user input data, such as Sleep Time, Meals Skipped, Mood, Work Stress, and Screen Time. This data is collected through a user-friendly interface and stored in a CSV format for further analysis.

Once the data is collected, it is sent to the Main Model for training. In the Model Training phase, the system processes the collected data to generate features like average sleep time, stress levels, and mood variations. The model is then trained using this data, with the goal of providing personalized recommendations for diet, workout, sleep improvement, and stress management.

The trained model incorporates various machine learning techniques, including Decision Trees, Ensemble Methods, and the Random Forest algorithm. These models work together to predict personalized insights based on the user's lifestyle data. The system also continuously updates its predictions with new data, ensuring real-time adaptability.

The results of the trained model are then displayed in the Output section. The system provides visual representations of the recommendations in the form of line graphs, showing trends like sleep improvement, stress reduction, and the effectiveness of suggested interventions. It also provides actionable insights like meal recommendations, breathing exercises, and optimal bedtime for the user.

Users interact with the system through the User Input component, where they can provide input on their sleep, mood, stress, and other wellness-related factors. The system then provides tailored recommendations based on their individual inputs, helping users improve their overall well-being.

3.3 UML DIAGRAMS

3.3.1 USE CASE DIAGRAMS

A use case diagram is a visual representation that depicts the interactions between various actors and a system, capturing the ways in which users or external entities interact with the system to achieve specific goals. It is an essential tool in system analysis and design, often used in software engineering and business analysis. In a use case diagram, actors are entities external to the system that interact with it, and use cases are specific functionalities or features provided by the system as seen in Fig. 3.3.1.1. These interactions are represented by lines connecting actors to use cases. The diagram helps to illustrate the scope and functionality of a system, providing a high-level view of how users or external entities will interact with it.

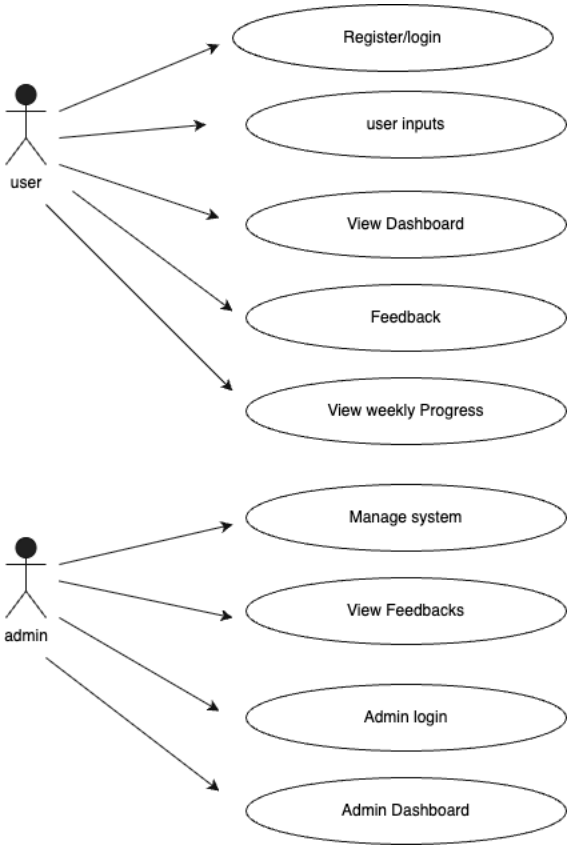


Fig. 3.3.1.1 Use Case Diagram

Actors:

1. **User**: The individual interacting with the system, providing input data (such as sleep time, mood, etc.) and viewing personalized wellness recommendations.
2. **System**: The backend system that processes user data, trains models, and generates predictions based on the user's inputs.

Use Cases:

1. **Register**: The user can create an account to access the system's features, including personalized recommendations for sleep and wellness.
2. **Login**: After registration, the user logs in to the system using their credentials to access personalized data and recommendations.
3. **Selecting Coin**: The user selects a cryptocurrency (Bitcoin or Ethereum) to predict its price, starting the prediction process.

4. **Prediction of Coin Price**: The system predicts the price of the selected cryptocurrency based on the trained models.
5. **Predicted Price in Incremental Timeframes**: The system provides predictions for different timeframes, such as hourly, daily, or weekly.
6. **Visualizing Trained Models and Predicted Price**: The system displays the trained machine learning models, showing graphs and charts of the predicted cryptocurrency price trends.

3.3.2 CLASS DIAGRAM

A class diagram is a visual representation that models the static structure of a system, showcasing the system's classes, their attributes, methods (operations), and the relationships between them as seen in Fig. 3.3.2.1. It is a key tool in object-oriented design and is commonly used in software engineering to define the blueprint of a system.

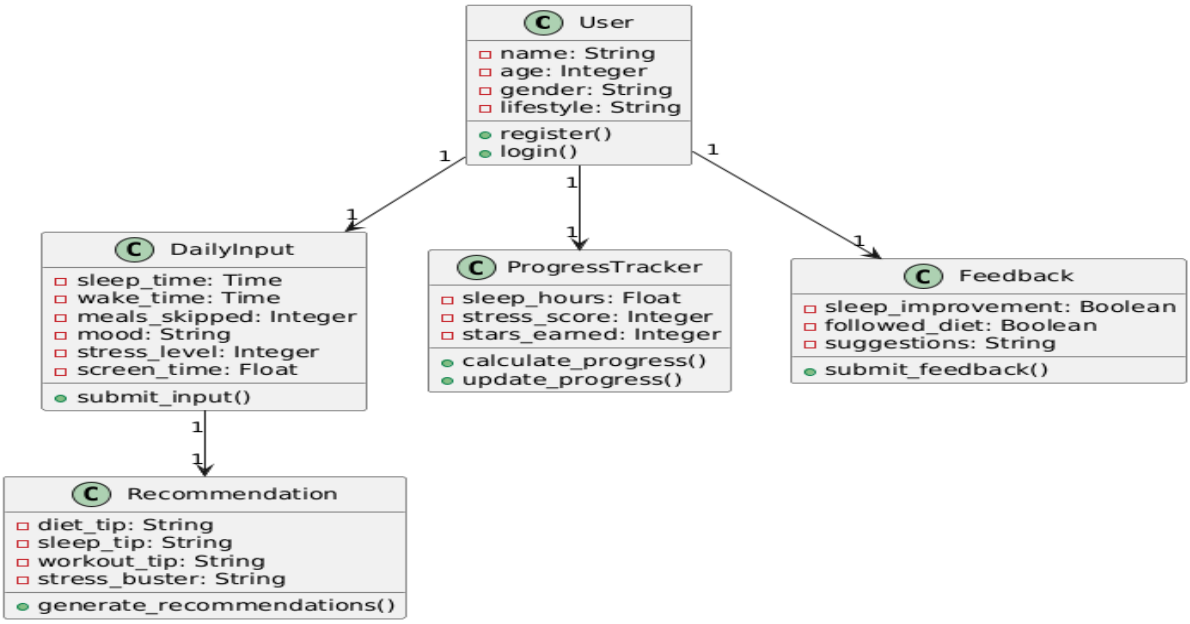


Fig. 3.3.2.1 Class Diagram

Relationships:

1. **User → Database**:

- The User interacts with the Database for login and registration**.**
- The Database handles storing user data and validating user credentials.

2. **Database → Predictor**:
   - The Predictor retrieves necessary data (like historical user input or system-related data) from the Database.
   - This data is then used by the Predictor for making predictions or training machine learning models.
3. **Predictor → Data Source**:
   - The Predictor fetches relevant data from an external Data Source (e.g., Yahoo Finance, if you're dealing with cryptocurrency data or other APIs for wellness data).
   - This data is used to train predictive models and inform future predictions.
4. **Predictor → Models**:
   - The Predictor uses different machine learning models (such as LSTM**,** Logistic Regression**,** and Voting Regressor) to analyze the data and perform predictions.
   - These models help to make accurate predictions based on historical and current data.
5. **Predictor → Output**:
   - The Predictor sends the predicted data (like predicted sleep improvement or stress levels) to the Output component.
   - It also sends plots, graphs, or any other data visualizations generated during prediction.
6. **Models → Output**:
   - The Models contribute to generating comparison charts and predictive visualizations, which are displayed in the Output section of the system.
   - These visualizations can include line graphs, bar charts, or other formats that help the user see their predicted progress and recommendations.

System Flow:

1. **User Interaction**:
    - The user logs in or registers via the User class**.**
    - The Database class handles credential verification or storing new users.
2. **Data Fetching**:
    - The Predictor fetches data from the Data Source, such as user inputs (e.g., sleep time, mood) or external data sources (e.g., historical cryptocurrency prices for model training).
3. **Prediction**:
    - The Predictor uses trained models (like LSTM, Logistic Regression, or Voting Regressor) to make predictions.
    - It uses historical and real-time data to generate the predictions for sleep improvement, stress reduction, diet recommendations, etc.
4. **Visualization**:
    - The Output class takes the predictions from the Predictor and generates visualizations (e.g., line charts, progress graphs, or comparative charts).
    - These visualizations help the user track their improvement, compare predicted outcomes, and adjust their wellness routine accordingly.

3.3.3 ACTIVITY DIAGRAM FOR RHYTHM RESTORE

An Activity Diagram is a type of behavioral diagram used in Unified Modeling Language (UML) to represent the flow of control or data through the system as seen in Fig. 3.3.3.1. It focuses on the flow of activities and actions, capturing the sequence of steps in a particular process or workflow. Activity diagrams are commonly used to model business processes, workflows, or any sequential activities in a system.

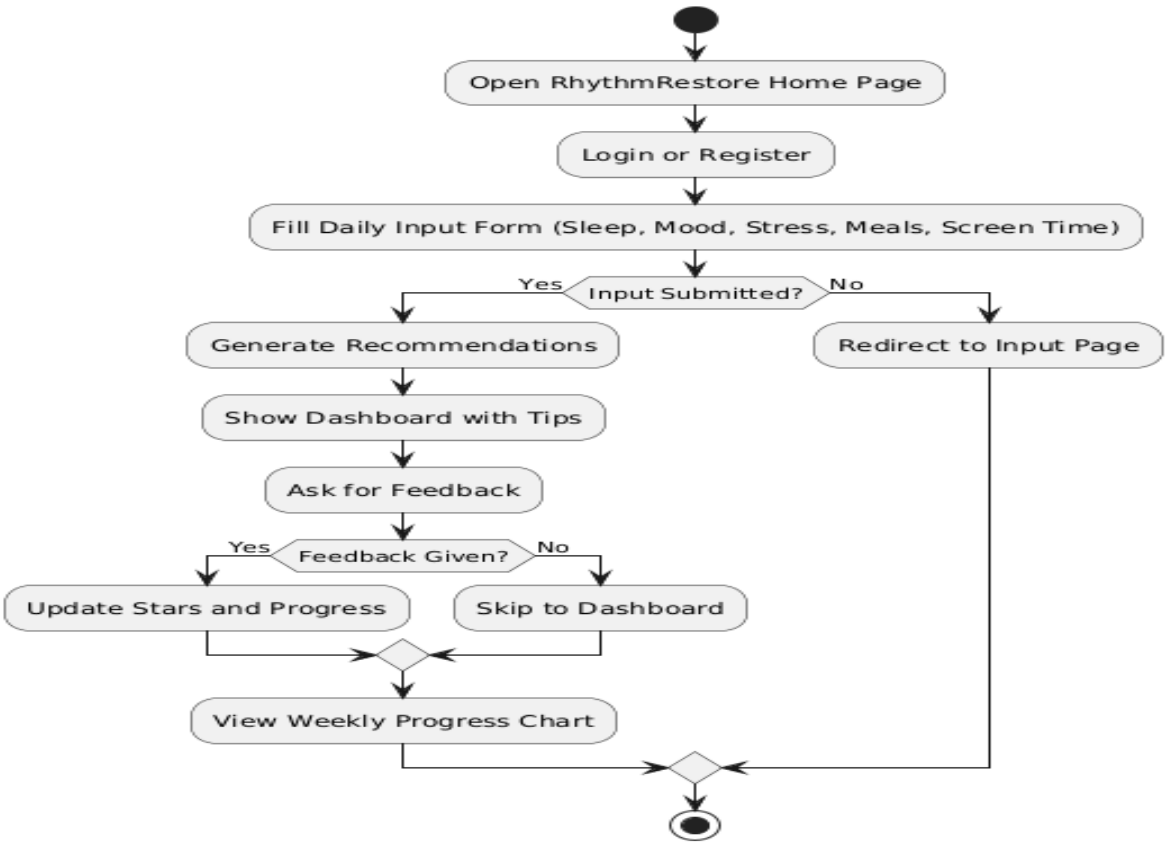


Fig. 3.3.3.1 Activity Diagram

Flow Explanation:

1. **Open Application**

   The user starts by opening the RhythmRestore application.

2. **Login or Register**

   The user is presented with two options:

   - Login: If the user already has an account, they log in by providing their credentials.
   - Register: New users need to register, which updates the database with their details (e.g., name, age, gender).

3. **Validation**

   - If the login details are valid, the user proceeds to the next step.
   - If the login details are invalid, the user is redirected back to the login page.

4. **Fill Daily Input Form**

   The user fills out the Daily Input Form, including:

   - Sleep Time
   - Mood
   - Stress Level
   - Meals Skipped
   - Screen Time

5. **Input Submitted?**
   - If the user submits the input, the system proceeds to the next step.
   - If the user does not submit the input, they are redirected back to the input page.

6. **GenerateRecommendations**

   The system generates personalized wellness recommendations based on the user's input data (e.g., sleep tips, meal suggestions, stress management).

7. **ShowDashboardwithTips**

   The system displays the Dashboard with personalized wellness tips, including diet suggestions, exercise recommendations, and sleep improvement.

8. **Ask for Feedback**

   The system asks the user for feedback on the recommendations provided.

9. **Feedback Given?**
   - If feedback is provided, the system updates the user's progress, including stars earned for following recommendations.
   - If no feedback is given, the system skips the progress update and displays the dashboard as usual.

10. **Update Stars and Progress**

    The system updates the user's stars and progress based on the feedback.

11. **ViewWeeklyProgressChart**

    The system displays a weekly progress chart, showing improvements in sleep, stress levels, and adherence to recommendations.

12. **End of Workflow**

    The process concludes, and the user can either:

    - Close the application, or
    - Perform additional input and predictions.

3.3.4 SEQUENCE DIAGRAM

A sequence diagram illustrates the flow of interactions between actors and system components over time as seen in Fig. 3.3.4.1, emphasizing the order in which messages are exchanged to achieve specific functionalities. Actors represent external entities that interact with the system, while lifelines depict the system components involved in the process. Messages are shown as arrows, indicating the flow of information or actions between these elements. By providing a step-by-step view of workflows, sequence diagrams help in understanding and designing the dynamic behaviour of a system.

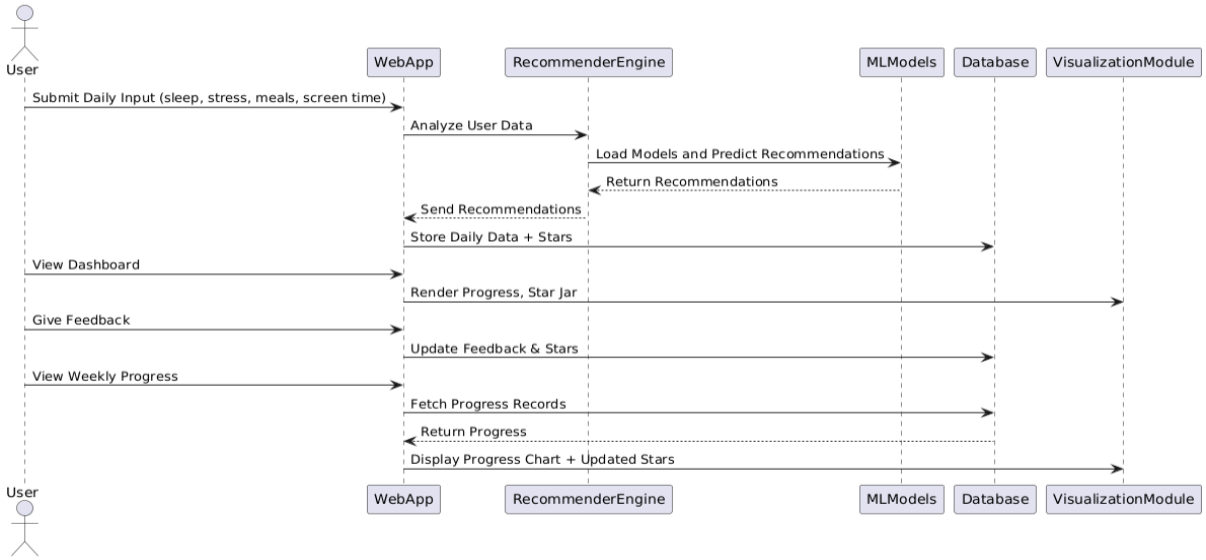


Fig. 3.3.4.1 Sequence Diagram

Key Interactions and Relationships

- **User and System**:
    - **Registration and Login**: The user registers and logs into the system using their credentials.
    - **Selecting Coin**: The user chooses which cryptocurrency (Bitcoin or Ethereum) they want predictions for.
- **System and Database**:

    - **Storing User Details**: When the user registers, the system stores their information in the database.
    - **Verifying Credentials**: During login, the system checks the user's credentials against the database to authenticate the user.
- **System and External Data Source**:
    - **Fetching Current Data**: To provide accurate predictions, the system requests the latest market data for the selected cryptocurrency from an external data source, such as a financial API.
- **Predictive Models**:
    - **Data Processing and Analysis**: The system processes the fetched data and feeds it into predictive models (e.g., LSTM, Logistic Regression, Voting Regressor) to forecast future prices.
- **System and User**:
    - **Displaying Results**: The system visualizes the predicted prices and other relevant information, presenting the results to the user through a user-friendly interface.

3.3.5 COMPONENT DIAGRAM

A Component Diagram is a type of structural diagram used in software engineering to represent the components of a system and how they interact or depend on each other. It shows how the components (which could be software modules, subsystems, or other significant parts) are organized and connected within a system. In this diagram, each component encapsulates a set of related functionalities and interfaces as shown in Fig. 3.3.5.1.

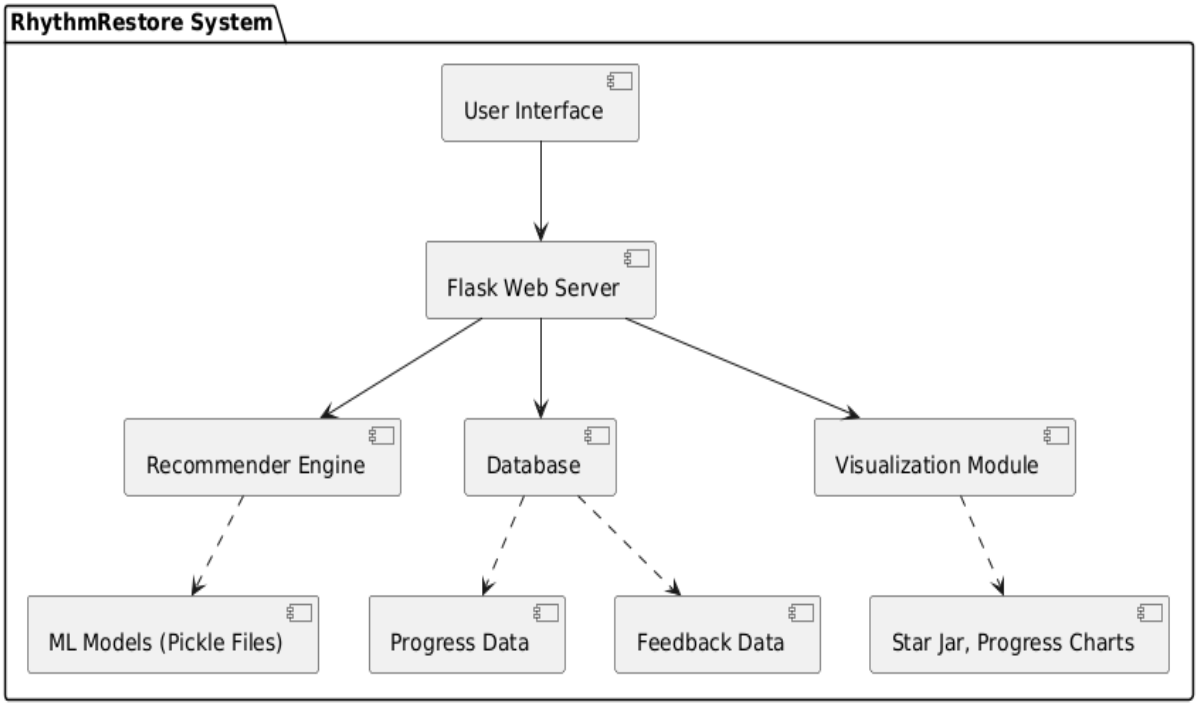


Fig. 3.3.5.1 Component Diagram

Main Components:

1. **User Interface**:
    - **Description**: The front-end component where the user interacts with the system. This includes user inputs such as mood, sleep time, screen time, meals skipped, and stress level.
2. **Flask Web Server**:
    - **Description**: The backend server built using Flask that handles requests from the User Interface and communicates with the Recommender Engine, Database, and Visualization Module.
3. **Recommender Engine**:
    - **Description**: This engine processes the data submitted by the user and generates wellness recommendations based on the trained ML Models. It sends the generated recommendations to the User Interface.
4. **Database**:
    - **Description**: The component that stores user details, progress data, feedback data, and other relevant information such as recommendations and user history. It serves as the data storage layer for the entire system.

5. **Visualization Module**:
    - **Description**: Responsible for visualizing the user's progress and displaying the results. It uses data from the Progress Data and Star Jar to display charts and graphs for the user.
6. **ML Models**:
    - **Description**: The machine learning models (e.g., LSTM, Logistic Regression, etc.) that are trained on historical data. These models are used to generate predictions and recommendations for the user based on their input.
7. **Progress Data**:
    - **Description**: Stores data related to the user's wellness progress, including sleep hours, stress levels, and mood, which is updated regularly as the user interacts with the system.
8. **Feedback Data**:
    - **Description**: Stores feedback provided by the user on the recommendations they received. This data is used to refine and improve the recommendations.
9. **Star Jar**:
    - **Description**: A component to track the user's progress by awarding stars based on their adherence to recommendations or overall improvement in their wellness.
10. **Progress Graphs**:

- **Description**: Displays the progress of the user over time, helping them track improvements in areas like sleep, stress, mood, and adherence to recommendations. These are visualized using graphs and charts.

3.3.6 DEPLOYMENT DIAGRAM

The Deployment Diagram illustrates the physical deployment of software components across hardware nodes in the **Rhythm Restore** system. It captures the interactions between the **user device**, **server**, and **database**, highlighting how system components communicate and are distributed in a real-world deployment scenario.

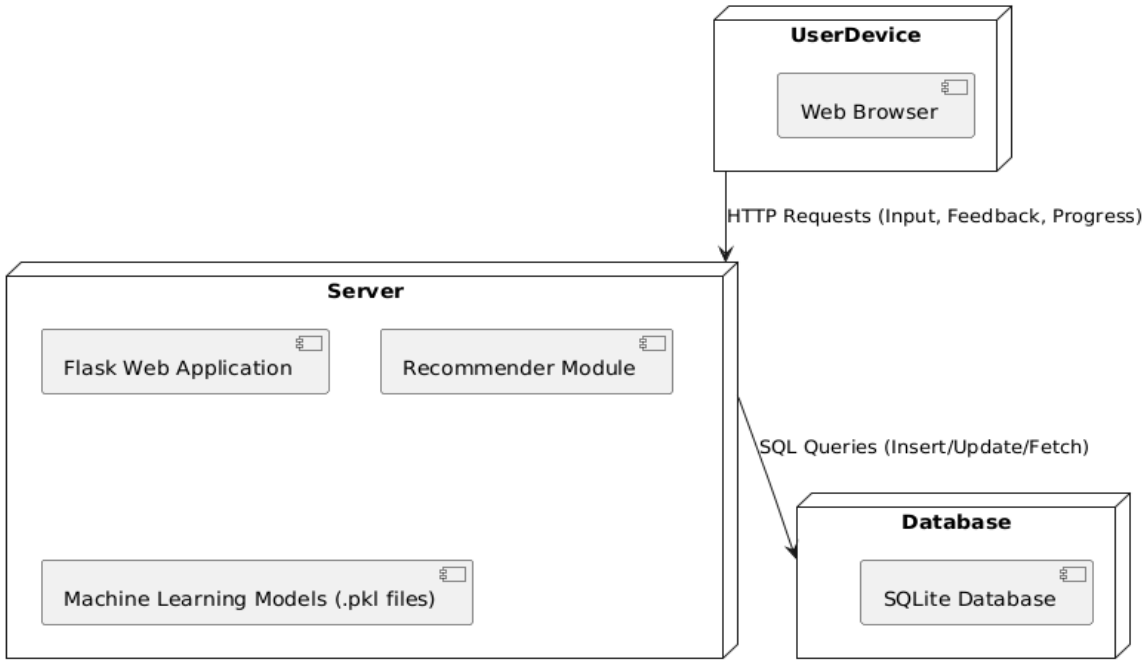


Fig. 3.3.6.1 Deployment Diagram

The deployment diagram shows how the system components are distributed across three main nodes:

- **User Device:** Runs the web browser where users input data (sleep, mood, stress, etc.) and view recommendations. Communicates with the server via HTTP requests.
- **Server:** Hosts the Flask Web Application, Recommender Module (using a **Weighted Rule-Based Algorithm**), and ML Models (.pkl files). It processes user input, generates recommendations, and manages communication with the database.
- **Database:** An SQLite database used to store user inputs, progress, and feedback. The server accesses it via SQL queries (insert/update/fetch).

This setup ensures smooth interaction between front-end, back-end, and data layers in the Rhythm Restore system.

3.4 METHODOLOGY

Data Acquisition

- **Live Data**: The system retrieves real-time data from **Yahoo Finance** using libraries like **yfinance**. This includes both up-to-date cryptocurrency price updates and historical data, which serve as the foundation for training the models.
- **Purpose**: The historical data is used to train the models, while live data ensures that predictions remain relevant and updated.

Model Steps

- **Data**: The historical data is used for training the models to demonstrate performance and facilitate comparison.
- **Time Frame Generation**: Data is divided into different time frames (e.g., 7 days, 30 days) to allow the models to forecast prices over multiple periods simultaneously, adding flexibility to the system.

Models Used

1. LSTM (Long Short-Term Memory)

- **Description**: LSTM is a type of **Recurrent Neural Network (RNN)** designed to handle time-series data. It is especially useful for predicting cryptocurrency prices because it can capture long-term dependencies and seasonal patterns in historical price movements.
- **How it Works**: LSTM models are trained on time-series datasets containing historical prices of Bitcoin and Ethereum. They learn to recognize recurring patterns and fluctuations in price movements. After training, the model can predict future prices by analyzing past trends, trading volumes, and other relevant factors.

- **Why it's Used**: LSTM's ability to store and manage information over long periods allows it to capture gradual, long-term trends in cryptocurrency prices, making it particularly effective for long-term predictions in volatile markets.

2. Linear Regression

- **Description**: Linear Regression is a statistical model used for predicting continuous numerical outcomes. In this case, it is adapted for forecasting cryptocurrency prices over a given period (e.g., next day, week, or month).
- **How it Works**: Linear Regression uses input features like past prices, market volume, and other technical indicators to predict the future price of a cryptocurrency. It fits a linear equation to the historical data and minimizes the difference between predicted and actual values.
- **Why it's Used**: Linear Regression is simple, fast, and interpretable, making it ideal for short-term price predictions. Its simplicity helps with decision-making in investments, although it might lack the ability to account for complex patterns compared to more advanced models.

3. Voting Regressor

- **Description**: The **Voting Regressor** is an ensemble learning technique that combines the predictions of multiple models to improve accuracy and robustness. In this system, it uses a combination of three base models:
    - **Random Forest Regressor**
    - **Decision Tree Regressor**
    - **Gradient Boosting Regressor**
- **How it Works**: The Voting Regressor averages the predictions of the base models to provide a final prediction. Each model makes an individual prediction, and the final output is an average of these predictions. This technique helps reduce overfitting or extreme predictions from any one model.
    - **Random Forest Regressor**: It is an ensemble of decision trees that reduces overfitting by averaging the predictions from multiple trees.

    - **Decision Tree Regressor**: This model splits the data at each node based on feature thresholds to predict continuous output values.
    - **Gradient Boosting Regressor**: Builds a sequence of models, where each new model tries to correct the errors made by the previous models.
- **Why it's Used**: The **Voting Regressor** helps to balance the strengths and weaknesses of each individual model:
    - Random Forest offers stability and resistance to overfitting.
    - Decision Trees can easily interpret and capture simple relationships in data.
    - Gradient Boosting improves predictive accuracy by focusing on correcting errors in previous models.

4. CODE AND IMPLEMENTATION

4.1 CODE

app.py

```
from flask import Flask, render_template, request, redirect, url_for, session
import sqlite3
import pandas as pd
from src.recommender import get_recommendations

app = Flask(__name__)
app.secret_key = 'super_secret_key'  # Needed for session management

# Connect to DB
def get_db_connection():
    conn = sqlite3.connect('data/rhythm_restore.db')
    conn.row_factory = sqlite3.Row
    return conn

# Home Page
@app.route('/')
def home():
    return render_template('home.html')

# Register Page
from src.send_sms import send_sms  # import this at top if not already

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        phone = request.form.get('phone')
        welcome_message = "Welcome to Rhythm Restore!"

        # Comment out SMS sending for now
        send_sms(phone, welcome_message)

        # Continue normal flow
        return redirect(url_for('input_page'))
    return render_template('register.html')
```

```
# Input Daily Data Page
@app.route('/input')
def input_page():
    return render_template('input.html')

# After Form Submit - Recommendations
@app.route('/recommend', methods=['POST'])
def recommend():
    user_input = {
        "sleep_time": request.form.get("sleep_time"),
        "wake_time": request.form.get("wake_time"),
        "meals_skipped": request.form.get("meals_skipped"),
        "stress": int(request.form.get("stress")),
        "mood": request.form.get("mood"),
        "traffic": request.form.get("traffic"),
        "screen_time": request.form.get("screen_time")  # Don't cast to int here
    }
    results = get_recommendations(user_input)

    # Save today's input
    session['today_input'] = user_input
    session['today_recommendations'] = results

    # Calculate sleep duration
    try:
        sleep = pd.to_datetime(user_input['sleep_time'])
        wake = pd.to_datetime(user_input['wake_time'])
        sleep_hours = (wake - sleep).seconds / 3600
        if sleep_hours < 0:
            sleep_hours += 24
    except:
        sleep_hours = 7.0  # default

    # Insert today's initial progress (stars will be updated after feedback)
    conn = get_db_connection()
    conn.execute('INSERT INTO progress (sleep, stress, stars) VALUES (?, ?, ?)',
                 (sleep_hours, user_input['stress'], 0))
    conn.commit()
    conn.close()

    return redirect('/dashboard')

# Dashboard
@app.route('/dashboard')
def dashboard():
    if 'today_recommendations' not in session:
        return redirect('/input')
    today = session['today_recommendations']
    return render_template('dashboard.html', today=today)
```

```
# Feedback Page
@app.route('/feedback')
def feedback():
    return render_template('feedback.html')

# Submit Feedback
@app.route('/submit_feedback', methods=['POST'])
def submit_feedback():
    sleep_improved = request.form.get('sleep_improved')
    followed_diet = request.form.get('followed_diet')

    stars = 0
    if sleep_improved == 'yes':
        stars += 1
    if followed_diet == 'yes':
        stars += 1

    conn = get_db_connection()
    conn.execute('UPDATE progress SET stars = ? WHERE id = (SELECT MAX(id)
FROM progress)', (stars,))

    # Insert into feedbacks table
    conn.execute('INSERT INTO feedbacks (sleep_improved, followed_diet, stars_earned)
VALUES (?, ?, ?)',
                 (sleep_improved, followed_diet, stars))
    conn.commit()
    conn.close()

    session['stars_earned'] = stars
    return redirect('/starjar')

# Star Jar Page
@app.route('/starjar')
def starjar():
    stars = session.get('stars_earned', 0)
    return render_template('starjar.html', total_stars=stars)

# Weekly Progress Page
@app.route('/progress')
def progress():
    conn = get_db_connection()
    rows = conn.execute('SELECT sleep, stress, stars FROM progress ORDER BY id
DESC LIMIT 7').fetchall()
    conn.close()

    progress = []
    total_stars = 0
    day = len(rows)

    for row in reversed(rows):  # show from Day 1 to Day 7
```

```
        sleep = float(row['sleep'])
        stress = int(row['stress'])
        stars = int(row['stars'])
        progress.append((day, sleep, stress, stars))
        total_stars += stars
        day -= 1

    return render_template('progress.html', progress=progress, total_stars=total_stars)
@app.route('/progress_graph')
def progress_graph():
    import matplotlib
    matplotlib.use('Agg')  # Force non-GUI backend

    import matplotlib.pyplot as plt
    import io
    import base64
    from flask import Response
    import sqlite3
    import pandas as pd

    conn = sqlite3.connect('data/rhythm_restore.db')
    query = "SELECT sleep, stress FROM progress ORDER BY id DESC LIMIT 7"
    df = pd.read_sql_query(query, conn)
    conn.close()

    df = df[::-1].reset_index(drop=True)
    df.index = [f"Day {i+1}" for i in range(len(df))]

    fig, ax1 = plt.subplots(figsize=(8, 4))
    ax1.plot(df.index, df['sleep'], marker='o', color='blue', label='Sleep')
    ax1.set_ylabel('Sleep Hours', color='blue')
    ax1.set_ylim(0, 10)
    ax1.tick_params(axis='y', labelcolor='blue')

    ax2 = ax1.twinx()
    ax2.plot(df.index, df['stress'], marker='s', color='red', label='Stress')
    ax2.set_ylabel('Stress Level (1–5)', color='red')
    ax2.set_ylim(0, 6)
    ax2.tick_params(axis='y', labelcolor='red')

    plt.title('📊 Sleep & Stress Trend (Past 7 Days)')
    plt.xticks(rotation=45)
    plt.tight_layout()

    buf = io.BytesIO()
    plt.savefig(buf, format='png')  #  Save to buffer
    buf.seek(0)

    return Response(buf.getvalue(), mimetype='image/png')
```

```
# Admin Login Page
@app.route('/admin_login', methods=['GET', 'POST'])
def admin_login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')
        if username == 'admin' and password == 'admin123':
            session['admin_logged_in'] = True
            return redirect('/admin_dashboard')
        else:
            return "Invalid credentials. Try again."
    return render_template('admin_login.html')

# Admin Dashboard
@app.route('/admin_dashboard')
def admin_dashboard():
    if not session.get('admin_logged_in'):
        return redirect('/admin_login')

    conn = get_db_connection()
    feedbacks = conn.execute('SELECT * FROM feedbacks').fetchall()

    # Count sleep improvements
    sleep_yes = conn.execute('SELECT COUNT(*) FROM feedbacks WHERE
sleep_improved = "yes"').fetchone()[0]
    sleep_no = conn.execute('SELECT COUNT(*) FROM feedbacks WHERE
sleep_improved = "no"').fetchone()[0]
    conn.close()

    return render_template('admin_dashboard.html', feedbacks=feedbacks,
sleep_yes=sleep_yes, sleep_no=sleep_no)

# Admin Logout
@app.route('/admin_logout')
def admin_logout():
    session.pop('admin_logged_in', None)
    return redirect('/')

# Contact Page
@app.route('/contact')
def contact():
    return render_template('contact.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001, debug=True)
```

index.html

```
<!DOCTYPE html>
<html>
```

```
<head>
  <title>RhythmRestore</title>
  <style>
    body {
      font-family: 'Segoe UI', sans-serif;
      background-color: #f5f5f5;
      text-align: center;
      margin: 0;
      padding: 0;
    }
    h1 {
      margin-top: 30px;
      font-size: 36px;
      color: #ff4c4c;
    }
    .form-container {
      background-color: white;
      max-width: 500px;
      margin: 30px auto;
      padding: 30px;
      border-radius: 20px;
      box-shadow: 0 0 20px rgba(0,0,0,0.1);
    }
    label {
      display: block;
      margin-top: 15px;
      font-weight: bold;
      text-align: left;
    }
    input, select {
      width: 100%;
      padding: 10px;
      margin-top: 5px;
      border: 1px solid #ccc;
      border-radius: 10px;
    }
    .btn {
      background-color: #ff4c4c;
      color: white;
      font-weight: bold;
      padding: 12px 30px;
      margin-top: 20px;
      border: none;
      border-radius: 12px;
      cursor: pointer;
      transition: 0.3s;
    }
    .btn:hover {
      background-color: #e63946;
    }
```

```
        .categories {
            margin-top: 40px;
        }
        .categories div {
            display: inline-block;
            margin: 8px;
            padding: 12px 18px;
            border-radius: 15px;
            font-weight: bold;
            color: white;
            font-size: 14px;
        }
        .yellow { background-color: #f4b400; }
        .green { background-color: #34a853; }
        .blue { background-color: #4285f4; }
        .purple { background-color: #9c27b0; }
        .pink { background-color: #e91e63; }
        .orange { background-color: #fb8c00; }
        .gray { background-color: #607d8b; }
    </style>
</head>
<body>

    <h1>RhythmRestore</h1>

    <form method="POST" action="/recommend" class="form-container">

        <label>Sleep Time:</label>
        <input type="time" name="sleep_time">

        <label>Wake-up Time:</label>
        <input type="time" name="wake_time">

        <label>Meals Skipped:</label>
        <select name="meals_skipped">
            <option value="Never">Never</option>
            <option value="Sometimes">Sometimes</option>
            <option value="Always">Always</option>
        </select>

        <label>Mood:</label>
        <select name="mood">
            <option value="Happy">Happy</option>
            <option value="Sad">Sad</option>
            <option value="Anxious">Anxious</option>
            <option value="Tired">Tired</option>
            <option value="Calm">Calm</option>
            <option value="Angry">Angry</option>
        </select>
```

```
        <label>Work Pressure Level:</label>
        <select name="work_stress">
            <option value="Low">Low</option>
            <option value="Medium">Medium</option>
            <option value="High">High</option>
        </select>

        <label>Stress Level:</label>
        <select name="stress">
            <option value="1">1 - No Stress</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - Extremely Stressed</option>
        </select>

        <label>Traffic Delay Impact:</label>
        <select name="traffic">
            <option value="Never">Never</option>
            <option value="Rarely">Rarely</option>
            <option value="Sometimes">Sometimes</option>
            <option value="Always">Always</option>
        </select>

        <button class="btn" type="submit">Get Recommendations</button>
    </form>



</body>
</html>
```

Trainmodels.py:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
import pickle

# Load your dataset
df = pd.read_csv('src/Rhythm_Restore_Refined.csv')

# Rename columns properly
df = df.rename(columns={
    'what time do you usually sleep': 'sleep_time',
    'what time do you usually wake up': 'wake_time',
    'How often do you skip meals?': 'meals_skipped',
```

```
    'How would you rate your stress levels during the day?Scale: 1 = No Stress, 5 =
Extremely Stressed)': 'stress',
    'Do you feel mentally exhausted before bedtime? ': 'mood',
    'How often are you stuck in traffic?': 'traffic',
    'How much screen time do you get before bed?': 'screen_time'
})

# 🛠 Convert sleep and wake time into sleep_duration_hours
df['sleep_time'] = pd.to_datetime(df['sleep_time'], format='%H:%M:%S', errors='coerce')
df['wake_time'] = pd.to_datetime(df['wake_time'], format='%H:%M:%S', errors='coerce')
# Fix screen_time values: convert '1Hr', '2Hr' -> 1, 2
df['screen_time'] = df['screen_time'].astype(str).str.extract('(\d+)').astype(float)
df['screen_time'] = df['screen_time'].fillna(1.0)  # fallback

df['sleep_duration_hours'] = (df['wake_time'] - df['sleep_time']).dt.total_seconds() / 3600
df['sleep_duration_hours'] = df['sleep_duration_hours'].apply(lambda x: x+24 if x<0 else
x)
df['sleep_duration_hours'] = df['sleep_duration_hours'].fillna(7.0)

# Final selected features
features = ['sleep_duration_hours', 'meals_skipped', 'stress', 'mood', 'traffic', 'screen_time']

# Fill missing values
df = df.fillna({
    'meals_skipped': 'No',
    'stress': 3,
    'mood': 'Neutral',
    'traffic': 'No',
    'screen_time': 1
})

# 🛠 Label Encode categorical columns
label_encoders = {}

for col in ['meals_skipped', 'mood', 'traffic']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le  # Save encoder if you need reverse mapping later

# Function to train and save models
def train_and_save_model(target_column, output_file):
    X = df[features]
    y = df[target_column]

    model = DecisionTreeClassifier()
    model.fit(X, y)

    with open(f'src/{output_file}', 'wb') as f:
        pickle.dump(model, f)
```

```
# Train and save models
train_and_save_model('Diet Suggestions', 'diet_model.pkl')
train_and_save_model('Workout Suggestions', 'workout_model.pkl')
train_and_save_model('Sleep Habit Suggestions', 'sleep_habit_model.pkl')
train_and_save_model('Stress Busters', 'stress_busters_model.pkl')

print("✅ All models trained and saved correctly!")
```

Input.html:

```
{% extends 'base.html' %}

{% block content %}
<div class="form-container mx-auto" style="max-width: 600px; background: white;
padding: 30px; border-radius: 20px; box-shadow: 0px 10px 20px rgba(0,0,0,0.2);">
  <h2 class="text-center mb-4 text-primary">🛏️ Daily Rhythm Input 🛏️</h2>

  <form action="/recommend" method="POST">

    <div class="mb-3">
      <label class="form-label">Sleep Time</label>
      <input type="time" class="form-control" name="sleep_time" required>
    </div>

    <div class="mb-3">
      <label class="form-label">Wake Up Time</label>
      <input type="time" class="form-control" name="wake_time" required>
    </div>

    <div class="mb-3">
      <label class="form-label">Did You Skip Meals?</label>
      <select class="form-select" name="meals_skipped" required>
        <option value="">Select</option>
        <option value="Yes">Yes</option>
        <option value="No">No</option>
      </select>
    </div>

    <div class="mb-3">
      <label class="form-label">Stress Level (1-5)</label>
      <select class="form-select" name="stress" required>
        <option value="">Select</option>
        <option value="1">1 - No Stress</option>
        <option value="2">2 - Low Stress</option>
        <option value="3">3 - Moderate Stress</option>
        <option value="4">4 - High Stress</option>
        <option value="5">5 - Extremely High Stress</option>
      </select>
    </div>
```

```
      <div class="mb-3">
        <label class="form-label">Mood</label>
        <select class="form-select" name="mood" required>
          <option value="">Select</option>
          <option value="Happy">Happy</option>
          <option value="Sad">Sad</option>
          <option value="Anxious">Anxious</option>
          <option value="Neutral">Neutral</option>
        </select>
      </div>

      <div class="mb-3">
        <label class="form-label">Did You Get Stuck in Traffic?</label>
        <select class="form-select" name="traffic" required>
          <option value="">Select</option>
          <option value="Yes">Yes</option>
          <option value="No">No</option>
        </select>
      </div>

      <div class="mb-3">
        <label class="form-label">Phone Screen Time Before Bed</label>
        <select class="form-select" name="screen_time" required>
          <option value="">Select</option>
          <option value="0-1 Hr">0-1 Hr</option>
          <option value="1-2 Hr">1-2 Hr</option>
          <option value="2-3 Hr">2-3 Hr</option>
          <option value="More than 3 Hr">More than 3 Hr</option>
        </select>
      </div>

      <div class="d-grid">
        <button type="submit" class="btn btn-success btn-lg mt-3">✨ Restore My
Rhythm ✨</button>
      </div>

    </form>
</div>
{% endblock %}
```

Register.html

```
{% extends 'base.html' %}

{% block content %}
<div class="form-container mx-auto" style="max-width: 600px; background: white;
padding: 30px; border-radius: 20px; box-shadow: 0px 10px 20px rgba(0,0,0,0.2);">
    <h2 class="text-center mb-4 text-success">Create Your Profile ✨</h2>

    <form action="/register" method="POST">
      <div class="mb-3">
```

```
        <label class="form-label">Name</label>
        <input type="text" class="form-control" name="name" required>
      </div>

      <div class="mb-3">
        <label class="form-label">Phone Number</label>
        <input type="text" class="form-control" name="phone" required>
      </div>

      <div class="d-grid">
        <button type="submit" class="btn btn-primary btn-lg">Continue to Restore
Rhythm 🚀</button>
      </div>
    </form>
</div>
{% endblock %}
```

Progress.html

```
{% extends "base.html" %}

{% block content %}
<div class="container text-center mt-5 fade-in">
  <h1 class="mb-4">📈 Your Weekly Progress</h1>

  <!-- 📊 Graph Section -->
  <div class="mb-5">
    <h4 class="mb-3">Sleep & Stress Graph</h4>
    <img src="{{ url_for('progress_graph') }}" alt="Progress Graph" class="img-fluid
rounded shadow">
  </div>

  <!-- 📅 Daily Stats Cards -->
  <div class="row justify-content-center">
    {% for day, sleep, stress, stars in progress %}
    <div class="col-md-3 mb-4">
      <div class="card shadow p-3 rounded-4">
        <h5 class="card-title">Day {{ day }}</h5>
        <p>😴 Sleep: <strong>{{ sleep }} hours</strong></p>
        <p>🔥 Stress: <strong>{{ stress }}/5</strong></p>
        <p>⭐ Stars Earned:</p>
        <p>
          {% for i in range(stars) %}
            <img src="{{ url_for('static', filename='img/star.png') }}" width="20px">
          {% endfor %}
        </p>
      </div>
    </div>
    {% endfor %}
  </div>
```

```
    <!-- 🏆 Star Jar -->
    <h3 class="mt-5">☆ Collected Stars ☆</h3>
    <div style="position: relative; width: 200px; margin: 20px auto;">
        <img src="{{ url_for('static', filename='img/jar.png') }}" style="width: 200px;">

        <div id="stars-container" style="position: absolute; top: 20px; left: 20px; width:
160px; height: 160px;">
            {% for i in range(total_stars) %}
                <img src="{{ url_for('static', filename='img/star.png') }}" class="floating-star"
style="top: {{ (i*10)%140 }}px; left: {{ (i*15)%140 }}px;">
            {% endfor %}
        </div>
    </div>
</div>

<style>
    .floating-star {
        width: 20px;
        position: absolute;
        animation: floatUp 2s ease-in-out infinite alternate;
    }
    @keyframes floatUp {
        from { transform: translateY(0); }
        to { transform: translateY(-10px); }
    }
    .fade-in {
        animation: fadeIn 2s;
    }
    @keyframes fadeIn {
        from { opacity: 0; }
        to { opacity: 1; }
    }
</style>
{% endblock %}
```

style.css

```
body {
    background: linear-gradient(to bottom right, #ccffcc, #ccffff);
    font-family: 'Poppins', sans-serif;
    margin: 0;
    padding: 0;
}

nav {
    background-color: #333;
    color: white;
    padding: 10px 20px;
    display: flex;
```

```
    justify-content: space-between;
    align-items: center;
}

nav a {
    color: white;
    margin-left: 20px;
    text-decoration: none;
    font-weight: 500;
}

.container {
    padding: 20px;
}

h1, h2 {
    text-align: center;
    margin: 20px;
}

.recommendations {
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
    margin: 30px 0;
}

.recommendation {
    background: #fff9db;
    padding: 20px;
    border-radius: 10px;
    width: 250px;
    text-align: center;
    margin: 15px;
    box-shadow: 0 4px 8px rgba(0,0,0,0.2);
}

.recommendation img {
    width: 100px;
    height: 100px;
    margin-bottom: 10px;
}

#bunny-message {
    font-size: 20px;
    margin-top: 20px;
    text-align: center;
}

#star-jar {
```

```
    width: 150px;
    display: block;
    margin: 20px auto;
}

.hidden {
    display: none;
}

/* Bunny Animation */
@keyframes hop {
  0%, 100% { transform: translateY(0); }
  50% { transform: translateY(-20px); }
}

#bunny {
    animation: hop 1s infinite;
}

/* Stars Animation */
.star {
  animation: fall 2s forwards;
  position: absolute;
}

@keyframes fall {
  0% { top: 0; opacity: 0; }
  50% { top: 150px; opacity: 1; }
  100% { top: 250px; opacity: 0; }
}
```

Dashboard.html:

```
{% extends "base.html" %}

{% block content %}
<div class="text-center">
    <h2>🌟 Your Daily Progress 🌟</h2>

    <h4 style="margin-top: 30px;">📋 Today's Recommendations</h4>
    <div class="row justify-content-center">
        <div class="col-md-3 m-2">
            <img src="{{ url_for('static', filename='img/diet_checkin.png') }}"
style="width:100px;">
            <p><strong>Diet:</strong> {{ today["Diet"] }}</p>
        </div>
        <div class="col-md-3 m-2">
            <img src="{{ url_for('static', filename='img/workout.png') }}"
style="width:100px;">
            <p><strong>Workout:</strong> {{ today["Workout"] }}</p>
```

```
      </div>
      <div class="col-md-3 m-2">
        <img src="{{ url_for('static', filename='img/sleep_improvement.png') }}"
style="width:100px;">
        <p><strong>Sleep:</strong> {{ today["Sleep Habits"] }}</p>
      </div>
      <div class="col-md-3 m-2">
        <img src="{{ url_for('static', filename='img/stress_relief.png') }}"
style="width:100px;">
        <p><strong>Stress:</strong> {{ today["Stress Busters"] }}</p>
      </div>
    </div>
</div>
{% endblock %}
```

4.2 IMPLEMENTATION

Create a directory folder as following and copy relevant pieces of code into the required parts: -

```
RhythmRestore/
│
├── app.py                    # Main Flask application
├── requirements.txt          # Python dependencies (if any)
├── rhythm_restore.db         # SQLite database (inside /data)
│
├── data/
│   └── rhythm_restore.db     # Stores progress and feedback
│
├── src/
│   ├── recommender.py        # Logic for ML model prediction
│   ├── diet_model.pkl        # Trained model for Diet
│   ├── workout_model.pkl     # Trained model for Workout
│   ├── sleep_habit_model.pkl # Trained model for Sleep
│   └── stress_busters_model.pkl  # Trained model for Stress
│
├── static/
│   ├── css/
│   │   └── style.css         # Custom styles
│   └── img/
│       ├── jar.png
│       ├── star.png
│       ├── sleep_improvement.png
│       ├── stress_relief.png
│       ├── workout.png
│       └── diet_checkin.png
│
├── templates/
```

```
│   ├── base.html              # Base layout
│   ├── home.html               # Home page
│   ├── register.html           # User registration (name, phone)
│   ├── input.html              # Daily input form
│   ├── dashboard.html          # Recommendations display
│   ├── feedback.html           # Feedback form
│   ├── starjar.html            # Star visualization
│   ├── progress.html           # Weekly progress
│   ├── admin_login.html        # Admin login form
│   └── admin_dashboard.html    # Admin analytics
│
└── README.md (optional)
```

Installing Python Packages

Open your terminal and run the following command to install all required packages:

pip install flask pandas scikit-learn sqlite3 matplotlib twilio

If you're using a requirements.txt, you can also run:

pip install -r requirements.txt

Optional: If you're using VS Code, make sure your virtual environment is activated before installing.

Setting Up SQLite

RhythmRestore uses **SQLite**, a lightweight embedded database — no external installation required.

To explore or edit the database manually:

1. Install **DB Browser for SQLite**
   📥 Download: https://sqlitebrowser.org/dl
2. After installation:
   - Open data/rhythm_restore.db
   - You can view or update progress, feedbacks, or users tables.

Creating the Required Tables

Open the terminal and enter SQLite shell:

sqlite3 data/rhythm_restore.db

Then, create the required tables:

```
-- Table to store user progress
CREATE TABLE IF NOT EXISTS progress (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    sleep REAL,
    stress INTEGER,
    stars INTEGER
);

-- Table to store user feedback
CREATE TABLE IF NOT EXISTS feedbacks (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    sleep_improved TEXT,
    followed_diet TEXT,
    stars_earned INTEGER
);

-- Table to store registered users and message preferences
CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT,
    phone_number TEXT,
    sleep_time TEXT,
    diet_tip TEXT,
    stress_tip TEXT
);
```

Exit the SQLite shell:

.exit

Running the Application

1. Navigate to the project root directory:

   cd RhythmRestore

2. Start the Flask server:

   python app.py

3. Open your browser and go to:

   http://127.0.0.1:5000

SMS Reminder Integration (Optional)

To enable SMS reminders:

- Create a Twilio account
- Generate your **Account SID**, **Auth Token**, and **Phone Number**
- Update these credentials in app.py

SMS reminders will be automatically sent based on the user's behaviour (sleep time, diet, stress) after registration and form submission.

5. TESTING

5.1 INTRODUCTION TO TESTING

Testing is an essential phase in software development that verifies the correct functionality, usability, performance, and reliability of the application under varying conditions. It ensures the final system meets the desired specifications and behaves consistently under expected and unexpected inputs. Effective testing minimizes the risk of system failures and enhances the user experience by identifying bugs or inconsistencies early in the lifecycle.

In this project, **RhythmRestore**, testing was crucial to ensure that personalized recommendations, user feedback tracking, gamification (star collection), and SMS-based reminders all functioned as intended. The goal was to validate that each module—from user registration to feedback analysis and administrative monitoring—performed seamlessly and accurately contributed to behavioral health tracking.

The test cases were crafted to verify the major functionalities and flow of the application, including:

- Correct transition between pages (home, input, dashboard, feedback, etc.)
- Accurate generation of lifestyle-based recommendations
- Proper recording and display of user progress and stars earned
- Reliable storage and retrieval of feedback entries
- SMS reminders being triggered for personalized schedules
- Admin login and visual analytics on user improvement

5.2 TEST CASES:

Table 5.1 Test Cases of Rhythm Restore

Test Case ID	Test case Name	Test Description	Expected Output	Actual Output	Remarks
TC_01	Home → Register	Navigate to registration page	Registration form should be displayed	Registration form displayed	Success
TC_02	Register → Input	Submit name and phone number	Redirect to daily input page	Redirected to daily input page	Success

TC_03	Input Form	Submit all lifestyle fields	Dashboard with personalized recommendations is shown	Dashboard with personalized recommendations is shown	Success
TC_04	Dashboard	Verify recommendation texts (diet, workout, stress, sleep)	Content reflects the behaviour of the user	Content reflects the behaviour of the user	Success
TC_05	SMS Reminders	Based on user's sleep time, verify SMS schedule	User receives SMS at scheduled morning, afternoon, and evening slots	User receives SMS at scheduled morning, afternoon, and evening slots	Success
TC_06	Feedback Page	Submit responses for improvement and diet adherence	User redirected to Star Jar and stars correctly calculated	User redirected to Star Jar and stars correctly calculated	Success
TC_07	Progress Chart	Visit progress page after multiple days	Progress bars, star counts reflect correct data	Progress bars, star counts reflect correct data	Success
TC_08	Admin Login	Admin login with credentials	Admin dashboard loads with feedback statistics	Admin dashboard loads with feedback statistics	Success
TC_09	Data Integrity	Open SQLite database and verify entries	Tables (users, progress, feedbacks) updated with accurate values	Tables (users, progress, feedbacks) updated with accurate values	Success
TC_10	Input Validation	Leave a field blank during input	Validation prompts the user to complete required fields	Validation prompts the user to complete required fields	Success

6. RESULTS

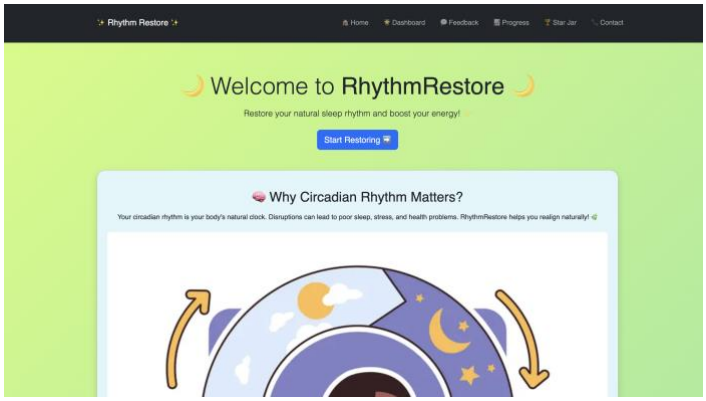


Fig. 6.1 Initial page

Fig. 6.1 shows the initial home page of the RhythmRestore web application when accessed through the local host address. It introduces the purpose of the app—helping users restore their natural sleep rhythm—and contains a "Start Restoring" button that navigates the user to the registration or input page.

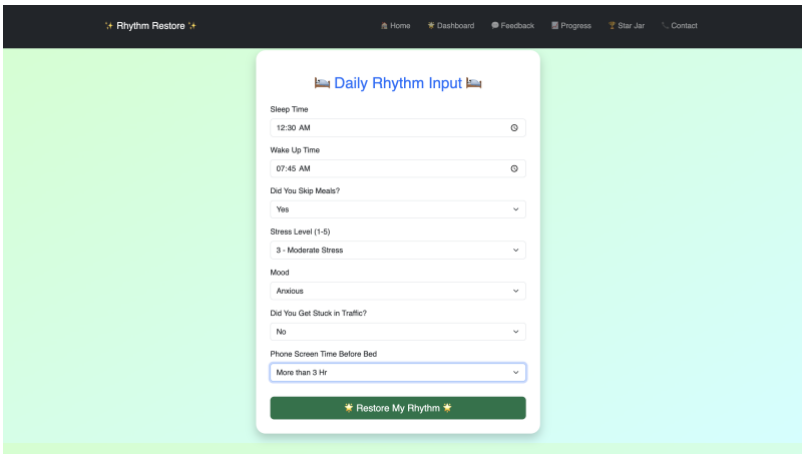


Fig. 6.2 Input page

Fig. 6.2 displays the Daily Rhythm Input form where users enter key lifestyle details such as sleep time, wake-up time, skipped meals, stress levels, mood, traffic delays, and phone screen time before bed. This information is used to generate personalized health recommendations.

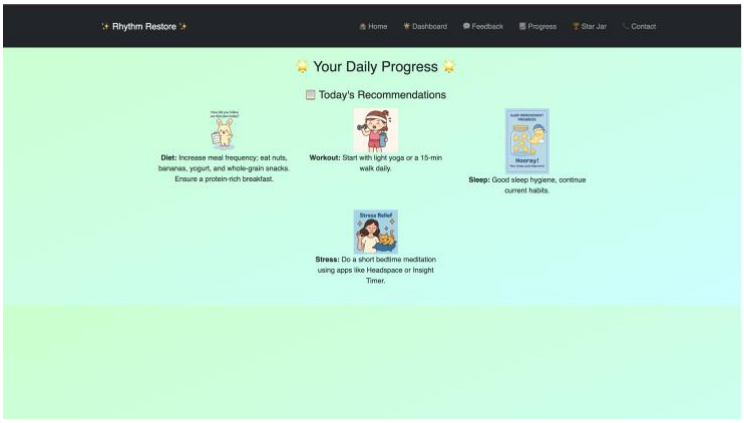


Fig. 6.3 User Dashboard page(1)

Fig. 6.3 presents the daily dashboard, displaying personalized recommendations based on the user's input. It includes tailored suggestions for diet, workout, stress relief, and sleep improvement, each accompanied by relevant icons or illustrations to enhance user engagement.

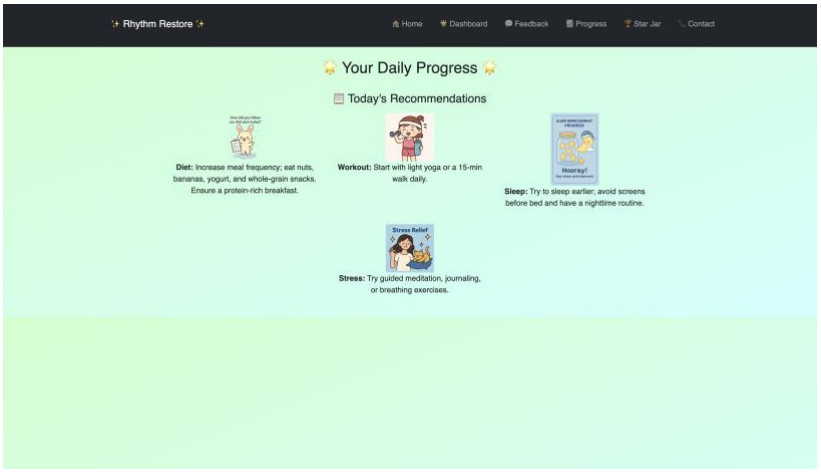


Fig. 6.4 User Dashboard page(2)

Fig. 6.4 presents the daily dashboard, displaying personalized recommendations based on the user's input. It includes tailored suggestions for diet, workout, stress relief, and sleep improvement, each accompanied by relevant icons or illustrations to enhance user engagement.

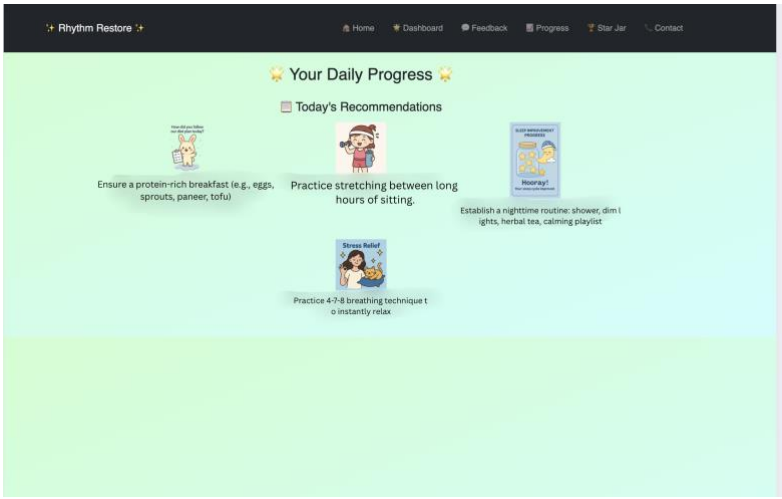


Fig. 6.5 User Dashboard page(3)

Fig. 6.5 presents the daily dashboard, displaying personalized recommendations based on the user's input. It includes tailored suggestions for diet, workout, stress relief, and sleep improvement, each accompanied by relevant icons or illustrations to enhance user engagement.

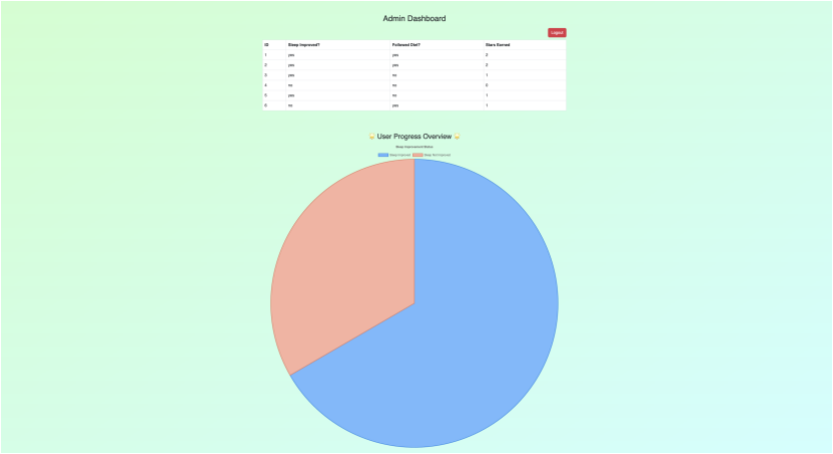


Fig. 6.6 Admin Dashboard page(3)

Fig. 6.6 displays the Admin Dashboard, providing a tabular overview of user feedback and performance. It lists key entries such as sleep improvement, diet adherence, and total stars earned by users. Below the table, a pie chart visualizes user progress by categorizing feedback data into two groups: positive (e.g., improved sleep or followed diet) and neutral/negative. This visualization enables administrators to monitor collective behavioural trends and assess the effectiveness of the recommendation system in real time.

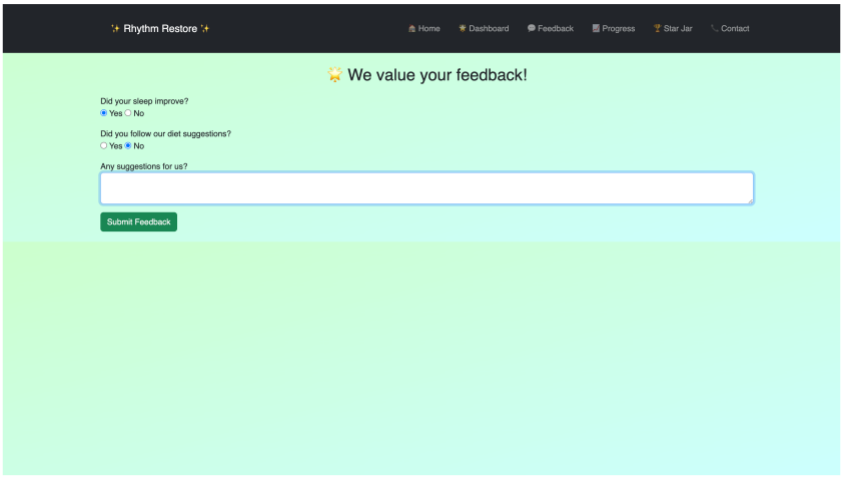


Fig. 6.7 Feedback page

Fig. 6.7 illustrates the feedback submission page, where users are asked to confirm whether their sleep improved and if they followed the recommended diet. A text box is provided for additional suggestions or comments, followed by a submission button to record the feedback.

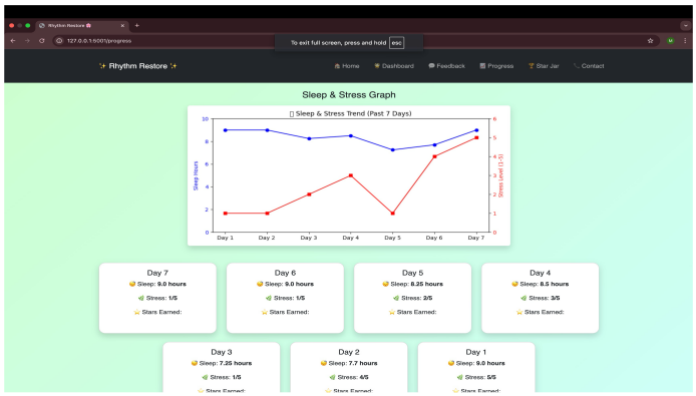


Fig. 6.5 Progress page

Fig. 6.8 shows the Weekly Progress section that visualizes trends in sleep duration and stress levels over the past 7 days using a line graph. Below the graph, daily statistics including hours slept, stress levels, and stars earned are listed for easy comparison and motivation

7. CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

The **RhythmRestore** project successfully demonstrates the integration of behavioral analytics, lifestyle monitoring, and AI-driven recommendations to promote healthier daily routines. By collecting key user inputs such as sleep patterns, mood, stress levels, and screen time, the system intelligently delivers personalized advice on diet, workout, stress relief, and sleep improvement.

A standout feature of the project is its **gamified progress tracking system**, where users earn stars based on behavioral improvement—creating a sense of motivation and self-accountability. The feedback loop, integrated with visual dashboards and an admin analytics panel, allows both users and supervisors to assess the system's effectiveness. Furthermore, the integration of **SMS-based reminders** provides real-time nudges, reinforcing healthy habits in an accessible and low-cost manner.

Overall, the system is intuitive, lightweight, and user-friendly—catering to individuals who may lack access to expensive health tracking devices or professional interventions.

7.2 FUTURE ENHANCEMENTS

Although the current version of RhythmRestore is effective, several enhancements can improve its scalability, accuracy, and user engagement:

- **Intelligent SMS Scheduling**: Extend the reminder system to dynamically adjust based on trends in user behaviour over weeks, not just based on a single day’s input.
- **Mobile App Development**: Create a mobile-friendly version or Android/iOS app for better accessibility and background notifications.
- **Advanced AI Models**: Integrate deep learning (e.g., Recurrent Neural Networks) to better predict behavioural risk zones based on multi-day input sequences.

- **Emotional Wellness Insights**: Include modules for mindfulness, journaling, or mood reflection prompts to further enhance psychological support.
- **Wearable Integration**: Incorporate APIs from smartwatches (Fitbit, Apple Watch) to gather real-time sleep and stress data.
- **Community and Sharing Features**: Enable users to share progress or participate in challenges for collective motivation.

REFERENCES

[1] C. della Monica et al., 2024. "A Protocol for Evaluating Digital Technology for Monitoring Sleep and Circadian Rhythms in Older People and People Living with Dementia in the Community," *Clocks & Sleep*, vol. 6, no. 1, pp. 129-155,2024, doi: 10.3390/clockssleep6010010.

[2] F. Wu et al., 2024. "Comparative Efficacy of Commercial Wearables for Circadian Rhythm Home Monitoring from Activity, Heart Rate, and CoreBody Temperature," *arXiv preprint*, doi: 2404.03408, 2024.

[3] A. Nguyen et al., 2022. "A Large-Scale Study of a Sleep Tracking and Improving Device with Closed-loop and Personalized Real-time Acoustic Stimulation," *arXiv preprint*, doi: 2211.02592, 2022.

[4] S. Rostaminia et al., 2021. "PhyMask: Robust Sensing of Brain Activity and Physiological Signals During Sleep with an All-textile Eye Mask," *arXiv preprint*, doi: 2106.07645, 2021

[5] S. H. Cho et al., 2023. "Wearable Devices for Monitoring Circadian Rhythms and Sleep Patterns: A Review," *Journal of Biomedical Science andEngineering*,vol.16,no.4,pp.1-15,2023 doi: 10.4236/jbise.2023.164005.

[6] D. S. Martinez et al., 2023. "Improvement of Sleep Quality through Smart Wearables: A Comparative Study of Devices," *Sleep Health*, vol. 9, no. 2, pp.102-111, 2023, doi: 10.1016/j.sleh.2023.02.005.

[7] J. R. Lee et al., 2022. "Smartphone-based Sleep Tracking and Its Impact on Sleep Improvement: A Review of Current Solutions," *Computers in Biology and Medicine*, vol. 148, pp. 1058-1065, 2022, doi: 10.1016/j.compbiomed.2022.1058.