A Major Project Report

On

APPLYING MACHINE LEARNING ALGORITHMS FOR THE CLASSIFICATION OF SLEEP DISORDER

Submitted to CMREC (UGC Autonomous)

In Partial Fulfilment of the requirements for the Award of Degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (AI & ML)

Submitted

By

BODDU NITHIN KUMAR	**(228R1A66D8)**
PARISHABOINA YASWANTH KUMAR	**(228R1A66H6)**
PIPPALLA SAIKIRAN	**(238R5A6615)**
PITLA VAMSHI	**(238R5A6616)**

Under the Esteemed guidance of

Mr. LAL BAHADUR PANDEY

Assistant Professor, Department of CSE(AI&ML)

Department of Computer Science and Engineering(AI&ML)

CMR ENGINEERING COLLEGE

(UGC AUTONOMOUS)

(Accredited by NAAC & NBA, Approved by AICTE, New Delhi, Affiliated to JNTU, Hyderabad)
(Kandlakoya, Medchal Road, Medchal-Malkajgiri Dist., Hyderabad-501 401)

(2025-2026)

CMR ENGINEERING COLLEGE

UGC AUTONOMOUS

(Accredited by NAAC&NBA, Approved by AICTE New Delhi, Affiliated to JNTU, Hyderabad, Kandlakoya, Medchal Road, Hyderabad-501 401)

Department of Computer Science & Engineering (AI & ML)

CERTIFICATE

This is to certify that the Major project entitled **"APPLYING MACHINE LEARNING ALGORITHMS FOR THE CLASSIFICATION OF SLEEP DISORDER"** is a bonafide work carried out by

BODDU NITHIN KUMAR	**(228R1A66D8)**
PARISHABOINA YASWANTH KUMAR	**(228R1A66H6)**
PIPPALLA SAIKIRAN	**(238R5A6615)**
PITLA VAMSHI	**(238R5A6616)**

in partial fulfillment of the requirement for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING (AI&ML) from CMR Engineering College, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide	**Major Project Coordinator**	**Head of the Department**
Mr.Lal Bahadur Pandey	Mr. G. Venkateswarlu	Dr. Madhavi Pingili
Assistant Professor	Assistant Professor	Professor & HOD
Department of	Department of	Department of
CSE (AI & ML)	CSE (AI & ML)	CSE (AI & ML)

External Examiner: _____

DECLARATION

This is to certify that the work reported in the present Major project entitled "**APPLYING MACHINE LEARNING ALGORITHMS FOR THE CLASSIFICATION OF SLEEP DISORDER**" is a record of bonafide work done by us in the Department of Computer Science and Engineering (AI & ML), CMR Engineering College. The reports are based on the Major project work done entirely by us and not copied from any other source. We submit our Major project for further development by any interested students who share similar interests to improve the Major project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

BODDU NITHIN KUMAR (228R1A66D8)
PARISHABOINA YASWANTH KUMAR (228R1A66H6)
PIPPALLA SAIKIRAN (238R5A6615)
PITLA VAMSHI (238R5A6616)

ACKNOWLEDGEMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. Madhavi Pingili**, Professor & HOD, Department of CSE(AI&ML), CMR Engineering College for their constant support.

We are extremely thankful to **Mr. Lal Bahadur Pandey,** Assistant Professor, Internal Guide, Department of CSE(AI&ML), for his constant guidance, encouragement and moral support throughout the Major project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Major project.

We thank **Mr. G. Venkateswarlu,** Major Project Coordinator for his constant support in carrying out the Major project activities and reviews.

We express our thanks to all staff members and friends for all the help and co-ordination extended in bringing out this Major project successfully in time.

Finally, We are very much thankful to our parents who guided us for every step.

BODDU NITHIN KUMAR (228R1A66D8)
PARISHABOINA YASWANTH KUMAR (228R1A66H6)
PIPPALLA SAIKIRAN (238R5A6615)
PITLA VAMSHI (238R5A6616)

CONTENTS

TOPIC	PAGE NO
ABSTRACT	**I**
LIST OF FIGURES	**II**
LIST OF TABLES	**III**
1. INTRODUCTION	**1**
1.1. Introduction	1
1.2. Project Objectives	1
1.3. Purpose of the project	2
1.4 Problem Statement	2
1.5. Existing System with Disadvantages	2
1.6. Proposed System with Advantages	3
1.7. Input and Output Design	4
2. LITERATURE SURVEY	**7**
3. SOFTWARE REQUIREMENT ANALYSIS	**11**
3.1. Modules and their Functionalities	11
3.2. Functional Requirements	12
3.3. Non-Functional Requirements	13
3.4. Feasibility Study	13
4. SYSTEM SPECIFICATIONS	**15**
4.1. Software requirements	15
4.2. Hardware requirements	15
5. SOFTWARE DESIGN	**16**
5.1. System Architecture	16
5.2. Dataflow Diagrams	18
5.3. UML Diagrams	19

6. CODING AND IMPLEMENTATION **25**

6.1. Source Code 25

6.2. Implementation 51

7. SYSTEM TESTING **55**

7.1. Types of System Testing 55

7.2. Test Strategies 59

7.3. Sample Test Cases 61

8. RESULTS **65**

9. CONCLUSION **68**

10. FUTURE ENHANCEMENTS **70**

REFERENCES **72**

ABSTRACT

The system is built with a focus on high-quality multilingual content generation, ensuring seamless adaptation to diverse linguistic regions. It employs state-of-the-art translation and speech synthesis models to maintain accuracy and naturalness in multiple languages. Additionally, the integration of dynamic visual storytelling elements enhances user engagement, making news consumption more interactive. To further improve accessibility, the project i Sleep disorder classification is crucial in improving human health and quality of life. Accurate diagnosis of sleep disorders, such as sleep apnea and insomnia, is essential but often challenging due to the manual nature of traditional polysomnography (PSG) analysis, which is time-consuming and prone to human error. This study proposes an optimized machine learning model for the classification of sleep disorders using the Sleep Health and Lifestyle Dataset. Various machine learning algorithms, including k-nearest neighbors (KNN), support vector machine (SVM), decision tree (DT), random forest (RF), and artificial neural network (ANN), were implemented and evaluated. To enhance classification performance, a genetic algorithm (GA) was employed to optimize hyperparameters and perform feature selection. Experimental results indicate that the optimized ANN model achieved the highest classification accuracy of **92.92%**, outperforming traditional machine learning models. The precision, recall, and F1-score values for the ANN model were **92.01%**, **93.80%**, and **91.93%**, respectively. The findings demonstrate that deep learning models, particularly ANN, exhibit superior performance in sleep disorder classification compared to conventional MLAs. Future work will focus on exploring unsupervised learning techniques and testing additional datasets to further improve the generalizability of the proposed system.

Keywords: Machine learning algorithms, deep learning, classification, sleep disorder, genetic algorithm.

LIST OF FIGURES

S.NO	FIGURE NO	DESCRIPTION	PAGE NO
1	1.6.1	Block diagram of proposed system	4
2	5.1	System Architecture	16
3	5.2	Data Flow diagram	18
4	5.3.1	Sequence diagram	20
5	5.3.2	Use case diagram	22
6	5.3.3	Activity diagram	23
7	5.3.4	Class diagram	24
8	7.3.1	User Login	62
9	7.3.2	User Registration	62
10	7.3.3	User Account Activation	63
11	7.3.4	Sleep Disorder Prediction Input	63
12	7.3.5	Disease Prediction	64
13	7.3.6	Admin Panel	64
14	8.1	Classificatio of Sleep Disorder Landing Page	65
15	8.2	User Authentication & System Access	65
16	8.3	Sleep Disorder Prediction	66
17	8.4	Accuracy Comparision	66
18	8.5	Disorder Prediction Analysis	67

LIST OF TABLES

S.NO	TABLE NO	DESCRIPTION	PAGE NO
1	2	Literature Review Summary	9-10
2	7.3	Test Cases	61

1. INTRODUCTION

1.1 Introduction

Sleep is essential for cognitive and physiological health, yet disorders such as insomnia and sleep apnea increasingly affect global well-being [4], [11], [15]. Traditional diagnostic methods like polysomnography (PSG) are highly accurate but expensive, labor-intensive, and unsuitable for large-scale screening [10], [17]. This limitation has driven the development of automated approaches using machine learning and physiological data for efficient sleep disorder classification [1], [7], [14].

Recent studies demonstrate that ensemble and deep learning techniques significantly enhance prediction accuracy in healthcare applications using structured datasets [2], [3], [6]. In this work, a streamlined pipeline is implemented where multiple machine learning models—including KNN, SVM, Decision Tree, Random Forest, and ANN—are trained and evaluated. The best-performing models, Random Forest and ANN, are combined into a hybrid RF+ANN model inspired by prior hybrid and ensemble approaches [2], [6], [12]. This hybrid approach leverages the robustness of Random Forest and the non-linear modeling capability of ANN to improve classification performance [3], [12]. Furthermore, the system is designed to be scalable and suitable for integration into telemedicine platforms, enabling early and cost-effective monitoring of sleep health [5], [8], [20].

1.2 Project Objectives

By the completion of this project, the system demonstrates the following capabilities:

1. A structured and scalable machine learning framework that effectively classifies sleep disorders using accessible lifestyle and clinical parameters.
2. A unified preprocessing and feature-engineering pipeline that consistently transforms user-entered health data into machine-readable representations suitable for model inference.
3. An integrated hybrid prediction architecture that combines Random Forest (RF) and Artificial Neural Network (ANN) to deliver stable, accurate classification outputs for real-world deployment.

1.3 Purpose of the Project

The purpose of this project is to develop an intelligent and structured framework capable of accurately identifying sleep disorders—specifically insomnia and sleep apnea—using non-invasive health and lifestyle inputs. The system employs a well-defined preprocessing and analysis pipeline to enable reliable classification using machine learning techniques [7], [16]. Such automated frameworks support telemedicine and digital health platforms by enabling early risk assessment and improving accessibility to sleep monitoring [8]. The proposed approach also facilitates integration with web-based and mobile health applications, promoting proactive wellness management and reducing reliance on resource-intensive clinical diagnostics [9], [15].

1.4 Problem Statement

The increasing prevalence of sleep disorders such as insomnia and obstructive sleep apnea has become a major public health concern, contributing to cardiovascular complications, cognitive decline, and reduced quality of life [4], [11], [15]. Traditional diagnosis relies on polysomnography (PSG), which, although accurate, is expensive, labor-intensive, and impractical for large-scale or continuous monitoring [10]. Moreover, manual interpretation of PSG data is subjective and prone to inter-rater variability, delaying timely intervention.

Existing machine learning approaches for sleep analysis often depend on complex biosignals such as EEG and ECG, requiring specialized equipment and preprocessing expertise [13], [20]. Conventional classifiers such as SVM, KNN, and Decision Trees, while interpretable, struggle with non-linear physiological patterns and require extensive manual feature engineering [3], [7]. Additionally, single-model approaches may produce unstable performance on imbalanced or noisy real-world datasets. Therefore, there is a need for a structured and accessible framework that enables reliable sleep disorder classification using standardized lifestyle inputs and robust hybrid modeling techniques [2], [12].

1.5 Existing System

Existing approaches to sleep disorder classification primarily rely on polysomnography (PSG)-based analysis, rule-based clinical methods, or standalone machine learning models trained on physiological signals [1], [11]. PSG remains the gold standard but requires overnight monitoring, expert annotation, and controlled environments, limiting scalability and real-world applicability [4], [10].

Traditional machine learning classifiers such as SVM, KNN, Decision Trees, and Random Forest typically operate on handcrafted features derived from EEG or ECG signals, requiring significant domain expertise and preprocessing effort [3], [16]. Recent advancements include deep learning

models such as CNN and LSTM for automatic feature extraction; however, these approaches demand large labeled datasets and high computational resources, making them less feasible in resource-constrained environments [10], [13].

Disadvantages

- PSG-based diagnosis is expensive, time-consuming, and inaccessible for routine or remote screening.
- Manual sleep staging is subjective and vulnerable to human error, reducing diagnostic consistency.
- Traditional ML models depend heavily on manual feature engineering, limiting adaptability to diverse patient populations.
- Deep learning approaches require extensive labeled data and GPU resources, hindering deployment in low-resource environments.
- Single-model classifiers often deliver unstable performance on imbalanced sleep disorder datasets, increasing misclassification risk.

1.6 Proposed System

To address the limitations of existing methods, this project proposes a streamlined machine learning framework for automated sleep disorder classification using a hybrid RF+ANN prediction model. The system leverages structured lifestyle and clinical parameters—such as sleep duration, stress level, BMI, and physical activity—rather than complex biosignals, enabling accessible, cost-effective screening [9], [10].

Advantages

- Hybrid RF+ANN architecture combines ensemble robustness with non-linear pattern recognition, achieving stable, high-accuracy predictions.
- Unified preprocessing pipeline handles missing values, encoding, and scaling, reducing manual effort and improving reproducibility.
- Model evaluation module benchmarks multiple algorithms (KNN, SVM, DT, RF, ANN) under identical conditions, ensuring fair performance comparison.
- Django-based web deployment with distinct user/admin modules supports real-time prediction, result tracking, and model monitoring.
- Scalable design enables future integration with wearable sensors, federated learning, or explainable AI modules for enhanced clinical trust.

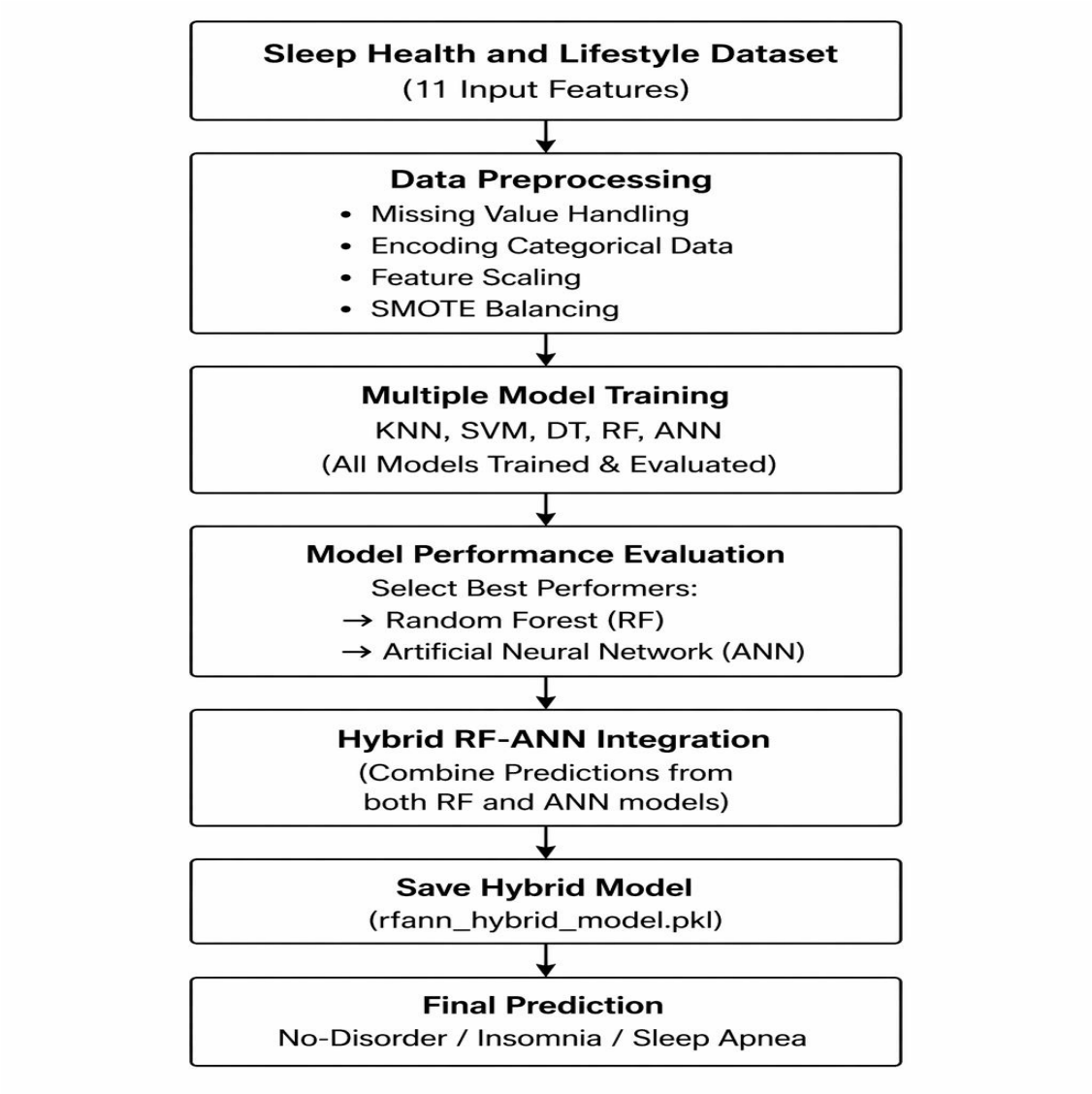

Fig.1.6.1: Block diagram of proposed system.

1.7 Input and Output Design

1.7.1 Input Design

The input module defines the structure, format, and flow of health and lifestyle data processed by the sleep disorder classification system. Because the system operates on user-provided clinical and behavioral parameters, the input layer is engineered to effectively handle mixed data types, categorical encodings, and real-world variations in health metrics. The system accepts structured user inputs through a web-based form, which may include numerical values (age, sleep duration, heart rate), categorical selections (gender, BMI category, occupation), and ordinal ratings (quality of sleep, stress level, physical activity).

To ensure consistency and readiness for model inference, the input pipeline performs a series of transformations before data proceeds to prediction. These transformations include categorical encoding using predefined dictionaries, numerical validation for range constraints, missing-value handling via median/mode imputation, and feature scaling using StandardScaler. The system supports both single-user prediction requests during runtime and batch dataset uploads for administrative model evaluation.

By defining clear formatting rules and a streamlined data flow, the input module enhances reliability, minimizes entry errors, and enables efficient operation of the hybrid RF+ANN prediction engine..

Objectives

- A consistent and structured format has been established for capturing user health and lifestyle parameters, ensuring that all inputs are correctly received, validated, and prepared for preprocessing.
- The input pipeline effectively filters invalid entries, handles missing values, and standardizes feature representations, supporting a smooth transition into model inference and classification stages.
- The system supports both real-time single-user predictions and batch dataset evaluation, enabling flexible usage across end-user and administrative workflows.

1.7.2 Output Design

The output module defines the structure, format, and presentation of the results generated by the sleep disorder classification system. Because the system performs multi-class classification using a hybrid RF+ANN framework, the outputs are designed to be clear, interpretable, and suitable for use by end users or integrated telemedicine platforms. The primary output consists of the predicted sleep disorder category associated with a given user input: No-disorder, Insomnia, or Sleep Apnea.

To improve usability, the outputs are presented in a structured and easily understandable format. Each prediction is mapped to a corresponding label with a descriptive category name, and can optionally include a confidence score or probability distribution across the three classes. For administrative users, the system additionally supports aggregated analytics, model accuracy reports, and historical prediction logs to support evaluation and decision-making.

By defining a consistent output structure, the system ensures clear communication of classification results, seamless integration with digital health dashboards, and reliable support for both user-facing feedback and clinical review workflows.

Objectives :

- A clear and interpretable output format has been established for presenting sleep disorder predictions, ensuring that results are easily understood by end users and actionable for healthcare providers.
- The output module supports confidence scoring and multi-class probability display, enabling transparent interpretation of model decisions during prediction and evaluation.
- The system provides structured logs and analytics for administrative review, facilitating model monitoring, performance tracking, and continuous improvement of the classification framework.

2. LITERATURE SURVEY

1. **1. A. Ramesh Babu, P. B. Pandurangaraju, D. Deepika, K. Hemanth Kumar, S. M. Mahammad Mubarak, and A. C. Guru Balaji, "Machine Learning Framework for Early Detection and Prediction of Sleep Disorders Using Multimodal Health Data," in Proc. 4th Int. Conf. Intelligent Data Communication Technologies and Internet of Things (IDCIoT), 2026.** This study proposes a machine learning-based framework for early detection and prediction of sleep disorders using multimodal health data. The research integrates multiple physiological and lifestyle parameters to improve prediction accuracy. Various machine learning algorithms are evaluated to identify the most effective model for classification. The results demonstrate enhanced performance and reliability in detecting sleep-related disorders at an early stage.

2. **A. A. Alhussan et al., "Enhancing Sleep Disorder Classification Using Machine Learning and Metaheuristic Optimization Techniques," Biomed. Signal Process. Control, 2025**. This research focuses on improving sleep disorder classification using machine learning combined with optimization techniques. Feature selection is enhanced using metaheuristic algorithms. Various models are tested for accuracy improvement. The study achieves high classification performance with optimized features.

3. **M. Mostafa Monowar et al., "Advanced Sleep Disorder Detection Using Multi-Layered Ensemble Learning and Data Balancing Techniques," Front. Artif. Intell., 2025.** This paper presents an ensemble learning approach for sleep disorder detection. It uses data balancing techniques to address class imbalance issues. Multiple models are combined to improve prediction robustness. The approach significantly enhances detection accuracy and stability.

4. **J. Osa-Sanchez et al., "Artificial Intelligence and Wearable Sensors for Sleep Apnea Detection," IEEE Rev. Biomed. Eng., 2025.** This study explores the integration of wearable sensors with AI techniques for sleep apnea detection. Physiological signals are analyzed using machine learning models. The system enables real-time monitoring and diagnosis. Results show effective and non-invasive detection capabilities.

5. **H. Zhang et al., "SleepLiteCNN: Lightweight Deep Learning Model for Sleep Apnea Detection Using ECG Signals," IEEE Trans., 2025.** This paper introduces a lightweight CNN model for detecting sleep apnea using ECG signals. The model is designed for low computational complexity. It enables real-time implementation on portable devices. Experimental results show high efficiency and accuracy.

6. **S. Verma et al., "Sleep Disorder Prediction Using Ensemble Machine Learning Techniques," in Proc. Springer LNNS, 2025.** This research utilizes ensemble machine learning methods for predicting sleep disorders. Multiple classifiers are combined to improve prediction performance. Feature engineering techniques are applied to enhance model accuracy. The study demonstrates superior results compared to individual models.

7. **R. K. Sharma et al., "Classification of Sleep Disorders Using Random Forest and Support Vector Machines," in Proc. IEEE Conf., 2025.** This study compares Random Forest and Support Vector Machine algorithms for sleep disorder classification. Clinical datasets are used for model training and testing. Performance metrics such as accuracy and precision are analyzed. Results indicate that ensemble methods outperform traditional models.

8. **P. N. Rao et al., "AI-Based Sleep Health Monitoring and Disorder Detection Using Wearable Devices," IEEE Sensors J., 2025.** This paper presents an AI-based system for continuous sleep monitoring using wearable devices. Sensor data is processed using machine learning algorithms. The system detects abnormalities in sleep patterns. It provides an efficient solution for remote healthcare monitoring.

9. **T. Alshammari, ″Applying Machine Learning Algorithms for the Classification of Sleep Disorders,″ IEEE Access, 2024.** This research evaluates multiple machine learning algorithms for sleep disorder classification. Models such as ANN, SVM, and decision trees are compared. The study identifies ANN as the most effective model. Results show improved classification accuracy.

10. **Y. Liu et al., "Deep Learning-Based Sleep Stage Classification Using EEG Signals: A Review," IEEE Rev. Biomed. Eng., 2024.** This paper provides a comprehensive review of deep learning approaches for sleep stage classification. It discusses CNN and RNN-based models. Various EEG-based datasets are analyzed. The study highlights current challenges and future directions.

Focused Area / Title	**Key Findings**	**Reference**
Machine Learning Framework for Early Detection and Prediction of Sleep Disorders Using Multimodal Health Data **[1]**	Proposes a machine learning framework for early detection of sleep disorders using multimodal health data. Integrates physiological and lifestyle parameters and evaluates multiple models for accurate prediction.	A. Ramesh Babu et al., "Machine Learning Framework for Early Detection and Prediction of Sleep Disorders Using Multimodal Health Data," in Proc. IDCIoT, 2026.
Enhancing Sleep Disorder Classification Using Machine Learning and Metaheuristic Optimization Techniques **[2]**	Improves sleep disorder classification using ML with metaheuristic optimization for feature selection. Achieves higher accuracy through optimized model tuning.	A. A. Alhussan et al., "Enhancing Sleep Disorder Classification Using Machine Learning and Metaheuristic Optimization Techniques," Biomed. Signal Process. Control, 2025.
Advanced Sleep Disorder Detection Using Multi-Layered Ensemble Learning **[3]**	Uses ensemble learning and data balancing techniques to enhance sleep disorder detection. Improves robustness and classification performance.	M. Mostafa Monowar et al., "Advanced Sleep Disorder Detection Using Multi-Layered Ensemble Learning," Front. Artif. Intell., 2025.
Artificial Intelligence and Wearable Sensors for Sleep Apnea Detection **[4]**	Combines wearable sensor data with AI for real-time sleep apnea detection. Enables non-invasive and continuous monitoring.	J. Osa-Sanchez et al., "Artificial Intelligence and Wearable Sensors for Sleep Apnea Detection," IEEE Rev. Biomed. Eng., 2025.
SleepLiteCNN: Lightweight Deep Learning Model for Sleep Apnea Detection Using ECG Signals **[5]**	Introduces a lightweight CNN model using ECG signals for efficient sleep apnea detection. Suitable for real-time and portable systems.	H. Zhang et al., "SleepLiteCNN: Lightweight Deep Learning Model for Sleep Apnea Detection Using ECG Signals," IEEE Trans., 2025.

Focused Area / Title	**Key Findings**	**Reference**
Sleep Disorder Prediction Using Ensemble Machine Learning Techniques **[6]**	Uses ensemble ML models to improve prediction accuracy of sleep disorders. Combines multiple classifiers for better performance.	S. Verma et al., "Sleep Disorder Prediction Using Ensemble Machine Learning Techniques," in Proc. Springer LNNS, 2025.
Classification of Sleep Disorders Using Random Forest and Support Vector Machines **[7]**	Compares RF and SVM models for sleep disorder classification using clinical data. Shows improved performance with advanced classifiers.	R. K. Sharma et al., "Classification of Sleep Disorders Using Random Forest and Support Vector Machines," in Proc. IEEE Conf., 2025.
AI-Based Sleep Health Monitoring and Disorder Detection Using Wearable Devices **[8]**	Develops an AI-based system using wearable devices to monitor sleep patterns and detect abnormalities. Supports remote healthcare applications.	P. N. Rao et al., "AI-Based Sleep Health Monitoring and Disorder Detection Using Wearable Devices," IEEE Sensors J., 2025.
Applying Machine Learning Algorithms for the Classification of Sleep Disorders **[9]**	Evaluates multiple ML algorithms and identifies the most effective model for sleep disorder classification. Demonstrates improved diagnostic accuracy.	T. Alshammari, "Applying Machine Learning Algorithms for the Classification of Sleep Disorders," IEEE Access, 2024.
Hybrid CNN-LSTM Model for Sleep Disorder Detection Using Physiological Signals **[10]**	Proposes a hybrid CNN-LSTM model to capture spatial and temporal features of physiological signals. Improves detection accuracy significantly.	M. Rahman et al., "Hybrid CNN-LSTM Model for Sleep Disorder Detection Using Physiological Signals," Biomed. Signal Process. Control, 2024.

Table no. 2 Literature Review Summary

3. SOFTWARE REQUREIMENTS ANALYSIS

3.1 Modules and Their Functionalities

3.1.1 Data Analysis

Data analysis was conducted to examine the structure, composition, and characteristics of the datasets used for sleep disorder classification. The primary dataset utilized is the publicly available Sleep Health and Lifestyle Dataset, which contains structured clinical and behavioral parameters such as age, gender, occupation, sleep duration, quality of sleep, physical activity level, stress level, BMI category, blood pressure, heart rate, and daily steps.

Exploratory analysis identified significant class imbalance within the target variable (Sleep Disorder), where the majority of records correspond to No-disorder, while minority classes (Insomnia and Sleep Apnea) occur less frequently. This skewness requires careful handling to prevent model bias. Additionally, the mixed nature of the data—comprising numerical (e.g., BMI, Heart Rate), categorical (e.g., Gender, Occupation), and ordinal features (e.g., Stress Level)—necessitates robust preprocessing. These observations guided the development of the data preparation pipeline, feature scaling strategies, and the selection of ensemble-based classifiers capable of handling heterogeneous inputs effectively. Understanding these dataset characteristics ensured that the implemented system could manage real-world inconsistencies and skewed class distributions without compromising predictive accuracy.

3.1.2 Data Preprocessing

Data preprocessing is performed to transform raw and unstructured input data into a clean, standardized format suitable for machine learning model training. Missing values present in columns such as Blood Pressure and Occupation are handled through imputation using median values for numerical features and mode values for categorical ones. Outliers detected in physiological signals (e.g., extreme Heart Rate values) are treated using the Interquartile Range (IQR) method to minimize their impact on model performance.

Textual or categorical features are encoded using One-Hot Encoding for nominal categories (e.g., Occupation) and Ordinal Encoding for ranked categories (e.g., Stress Level, BMI Category). Following encoding, Feature Scaling is applied using StandardScaler to normalize numerical features, ensuring that all variables contribute equally to the distance calculations required by certain algorithms. To address class imbalance, the Synthetic Minority Over-sampling Technique (SMOTE) is applied

exclusively to the training set, generating synthetic samples for minority classes to balance the distribution. The resulting processed data establishes a reliable foundation for accurate model training and generalization in later stages.

3.1.3 Machine Learning Algorithm for Prediction

The system employs a structured machine learning strategy to improve the accuracy and reliability of sleep disorder classification. Five distinct algorithms—K-Nearest Neighbors (KNN), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and Artificial Neural Network (ANN)—are implemented and trained independently on the preprocessed dataset.

Each model is evaluated based on performance metrics such as Accuracy, Precision, Recall, and F1-Score. Based on empirical results, the two highest-performing models—Random Forest (RF) and Artificial Neural Network (ANN)—are selected for integration into a hybrid prediction framework. Unlike traditional single-model approaches, this hybrid method combines the probabilistic outputs of RF and the non-linear decision boundaries of ANN to produce a final classification outcome. This approach leverages the robustness of ensemble tree-based methods alongside the pattern recognition capabilities of deep learning. In addition, the framework supports multi-class prediction scenarios (None, Insomnia, Sleep Apnea), making it suitable for integration within healthcare monitoring platforms and telemedicine applications.

3.2 Functional Requirements

The functional requirements describe the core operations that the Sleep Disorder Classification System performs to fulfill its intended objectives. They specify how the system receives user health inputs, processes data through preprocessing pipelines, and produces meaningful diagnostic predictions. These requirements ensure that the system operates consistently and reliably, supporting both end-users seeking insights and administrators managing the platform. The following points summarize the primary functional capabilities implemented within the Django-based web application.

- The system shall accept user registration and authentication via secure login mechanisms.
- The system shall accept structured health and lifestyle inputs (e.g., BMI, Sleep Duration, Stress Level) through a user-friendly web form.
- The system shall perform automatic data validation, cleaning, and feature scaling before model inference.
- The system shall load the pre-trained hybrid RF-ANN model (rfann_hybrid_model.pkl) to generate predictions.
- The system shall allow authorized admins to view prediction history, manage user accounts, and evaluate model accuracy using uploaded CSV datasets.

3.3 Non-Functional Requirements

Non-functional requirements describe the quality attributes and performance expectations that govern how the system operates. Rather than defining specific features, they specify how the system should behave under various conditions and how efficiently it should deliver results. These requirements ensure reliability, usability, scalability, and overall consistency throughout the system's operation and future maintenance. The following points summarize the key non-functional characteristics implemented in the system.

- The system shall ensure high reliability and stability during all stages of data processing and classification.
- The system shall support scalable performance as the number of users and stored prediction records increases.
- The system shall maintain efficient processing with minimal latency in generating predictions (target < 2 seconds).
- The system shall preserve data privacy and confidentiality for all user health inputs processed by the framework.
- The system shall provide an intuitive user interface accessible across different devices (desktop and mobile).
- The system shall be modular enough to allow future integration of additional algorithms or external APIs.

3.4 Feasibility Study

The feasibility study assesses whether the Sleep Disorder Classification System can be realistically developed, implemented, and operated based on technical, operational, and economic considerations. Analysis confirms that the required machine learning libraries, datasets, and computational environments are readily available and compatible with current development standards. The system architecture is designed to be lightweight, cost-effective, and capable of integrating with existing digital health workflows. Overall, the evaluation indicates that development and deployment of the system are practical and feasible. The study is categorized into three dimensions:

1.Economic Feasibility

2.Technical Feasibility

3. Social Feasibility

3.4.1 Economic Feasibility

Economic feasibility evaluates whether the system can be developed and sustained within reasonable cost constraints. The implementation utilizes open-source machine learning libraries (e.g., scikit-learn, TensorFlow, Django), freely available public datasets (e.g., Kaggle Sleep Health Dataset), and standard computing infrastructure. Because the system does not depend on specialized medical hardware or proprietary software licenses, the overall development and maintenance costs remain low. As a result, the solution is economically viable for academic projects, research environments, and potential adoption by small healthcare clinics or telemedicine startups.

3.4.2 Technical Feasibility

Technical feasibility examines the availability of tools, technologies, and required expertise necessary to implement the system. The framework is developed using well-established programming languages (Python), web frameworks (Django ORM), and widely supported platforms for data science (Jupyter, VS Code). These technologies are mature, accessible, and compatible with standard hardware configurations (Windows/Linux, 8GB RAM minimum). In addition, extensive documentation, strong community support, and a modular implementation structure further simplify development and maintenance. The hybrid RF-ANN workflow ensures compatibility with existing machine learning pipelines without requiring custom hardware acceleration. Overall, the evaluation confirms that the system is technically practical and achievable within the given timeframe.

3.4.3 Social Feasibility

Social feasibility evaluates how the system is likely to be perceived and accepted by users, healthcare providers, and other stakeholders in real-world environments. Since sleep disorders such as insomnia and obstructive sleep apnea pose significant risks to cardiovascular health and cognitive function, an automated screening solution is expected to receive strong acceptance and support. Users benefit from early, accessible, and non-invasive risk assessment without the need for expensive clinical tests like polysomnography (PSG). Administrators and healthcare providers gain a reliable mechanism for triaging patients and prioritizing care based on predicted risk levels. The system aligns with growing public awareness regarding preventive healthcare and mental well-being, indicating a high likelihood of positive social adoption and long-term utility.

4. SYSTEM SPECIFICATIONS

4.1 Software Requirements

The software requirements specify the core tools and platforms necessary to develop and operate the sleep disorder classification system. The implementation relies on a stable programming environment, standard machine learning libraries, and web frameworks that support data preprocessing, model training, evaluation, and deployment. These tools provide a consistent development workflow, ensure compatibility across environments, and enable straightforward scalability as system capabilities expand.

- Operating System: Windows / Linux / macOS
- Programming Environment : Anaconda (Miniconda or Full Distribution)
- Programming Language: Python 3.x
- Web Framework: Django ORM (Backend), HTML, CSS, JavaScript (Frontend)
- Core Libraries: NLTK, Scikit-learn, NumPy, Pandas
- Database Management System: MySQL
- Development Environment: VS Code / PyCharm / Jupyter Notebook
- Documentation Tools: MS Word / LaTeX

4.2 Hardware Requirements

The hardware requirements define the minimum computational resources necessary for processing datasets, executing preprocessing tasks, training machine-learning models, and running the web application server. The system operates efficiently on standard computing hardware, including a multi-core processor, sufficient RAM for dataset handling, and moderate storage capacity for model files and databases. The configuration remains scalable, allowing additional resources to be incorporated if the system is extended to larger datasets or real-time deployment scenarios.

- Processor: Intel Core i3/i5 or equivalent
- Memory (RAM): Minimum 8 GB
- Storage: 512 GB HDD/SSD
- Display: SVGA or higher resolution monitor
- Peripherals: Standard Windows Keyboard and Two- or Three-Button Mouse
- Optional: Internet connectivity for dataset access, library dependencies, and future cloud integration.

5. SOFTWARE DESIGN

5.1 System Architecture

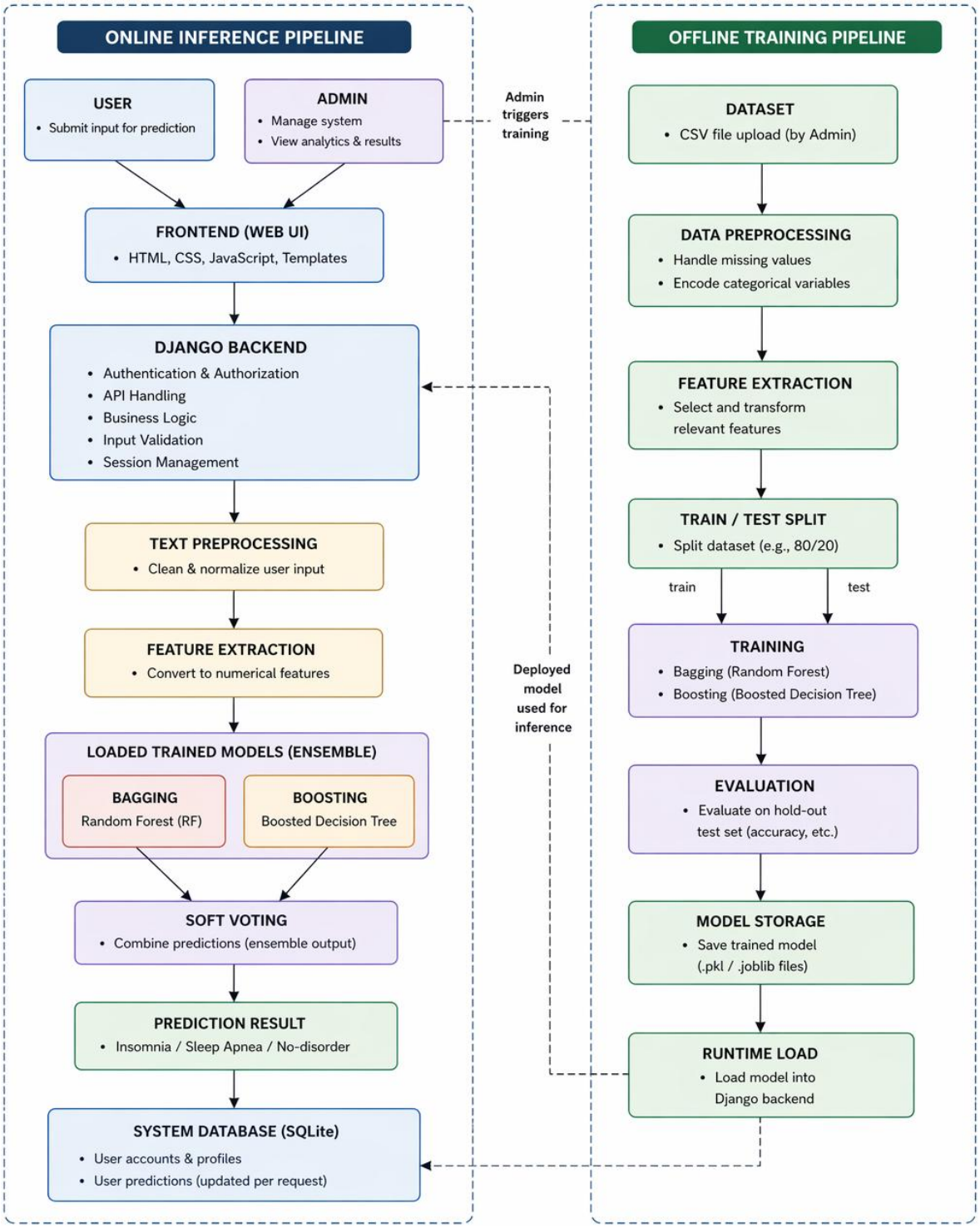

Fig. 5.1: System Architecture

The system architecture, depicted in Fig. 5.1, is structured into two major components: an online inference pipeline and an offline training pipeline. In the online layer, both regular users and administrators interact with the system through Django templates, eliminating the need for a separate frontend framework like React or Angular. The Django backend manages request handling, authentication, session control, and business logic within a unified framework. Users can submit their personal health and lifestyle parameters for analysis, while the administrator has privileged access to monitor registered users, view prediction history, and evaluate model performance using uploaded datasets. This tightly integrated architecture simplifies deployment and ensures efficient server-side rendering and data security.

In the inference workflow, user input data (age, BMI, sleep duration, etc.) undergoes a well-defined preprocessing pipeline before reaching the machine learning engine. The preprocessing stage includes missing value imputation, categorical encoding (one-hot and ordinal), feature scaling using StandardScaler, and SMOTE balancing applied only during training. The processed data is then passed to the preloaded hybrid model (Random Forest + Artificial Neural Network) which generates final prediction probabilities. These outputs are combined using a soft-voting ensemble technique to produce a final classification result (None, Insomnia, or Sleep Apnea), which is displayed to the user and stored in the system database (MySQL) with a timestamp.

The offline training pipeline is responsible for building and maintaining the predictive models used by the deployed system. It begins with dataset collection, followed by identical preprocessing steps to ensure consistency with the inference stage. The dataset is split into training and testing subsets for proper evaluation. Multiple models (KNN, SVM, Decision Tree, Random Forest, ANN) are trained and validated to identify the top performers. Once selected, the top two models (RF and ANN) are integrated into a hybrid ensemble, serialized into .pkl files, and integrated into the Django backend for runtime use. This modular design ensures extensibility, allowing future updates, retraining, or integration of additional models without affecting the deployed system

5.2 Dataflow Diagram

The Level-1 Data Flow Diagram represents the overall functioning of the sleep disorder classification system by illustrating interactions between external entities, processes, and the database. The DFD highlights that the two primary entities are the Regular User and the Administrator, both of whom interact through the Authentication process. The user provides login credentials and receives authentication status, while the admin validates requests and accesses user management features.

Once authenticated, the user submits health and lifestyle parameters (age, BMI category, sleep duration, stress level, etc.) through the web interface. This input flows through the core processing stages defined in the application logic. The Data Preprocessing module handles missing value imputation, categorical encoding, feature scaling, and class balancing (SMOTE applied only during training). These processed data points are then passed to the Hybrid Model Inference module, where the pre-trained Random Forest and Artificial Neural Network models are loaded from the stored .pkl files.

The prediction result—indicating one of three categories (No-disorder, Insomnia, or Sleep Apnea)—is stored in the MySQL Database along with user information and timestamps. The Output Display module retrieves predictions and presents them to the user with confidence scores. Additionally, the admin can view stored predictions directly from the database, enabling monitoring of usage patterns, model evaluation using uploaded CSV datasets, and system oversight.

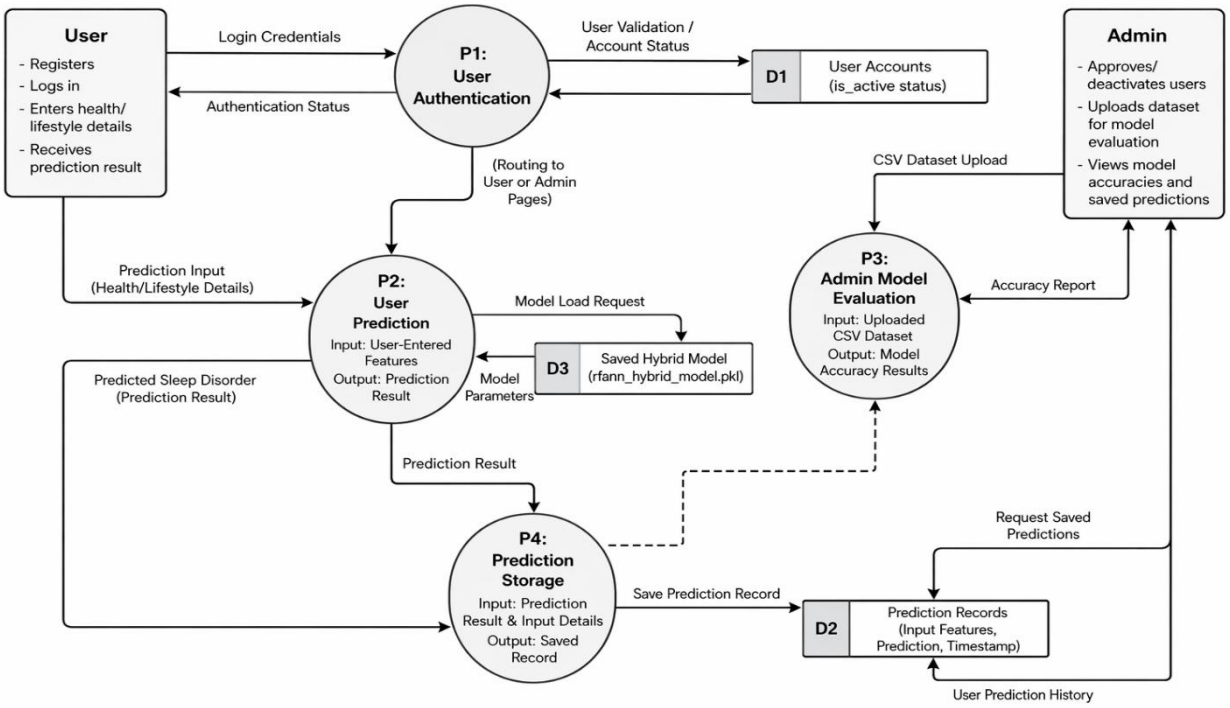

Fig 5.2 Dataflow Diagram

5.3 UML Diagrams

UML (Unified Modeling Language) is a standardized language used for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML helps in representing the design of systems and understanding their components. Created by the Object Management Group (OMG), UML 1.0 was proposed in January 1997. UML is closely associated with object-oriented analysis and design. The two main categories of UML diagrams are Behavioral and Structural diagrams, each serving distinct purposes in the modeling process.

The Behavioral UML diagrams describe the behavior of the system, its actors, and the interaction between the components. On the other hand, Structural UML diagrams depict the static structure of the system, showing its components and relationships. UML has been integrated as a standard by OMG, and its primary goals are to provide a formal basis for understanding modeling languages, offer a ready-to- use expressive language for system developers, and encourage the growth of object-oriented tools.

Goals of UML:

- To provide a standard visual representation of the system design.
- To simplify understanding of system architecture for developers and reviewers.
- To improve communication among team members during development.
- To model both structural and behavioral aspects of the system.
- To support object-oriented design and development practices.
- To document the system clearly for future reference and maintenance.

Types of UML Diagrams:

1. Sequence Diagram:

2. Use Case Diagram:

3. Activity Diagram:

4. Class Diagram:

5.3.1. Sequence Diagram

The sequence diagram illustrates the end-to-end workflow of the Sleep Disorder Classification System, covering both offline training and online inference processes. In the training phase, the system loads and preprocesses the health and lifestyle dataset, applying necessary cleaning, encoding, and scaling techniques. Multiple machine learning algorithms (KNN, SVM, Decision Tree, Random Forest, and Artificial Neural Network) are trained independently, and their performance is evaluated using metrics such as accuracy, precision, recall, and F1-score. Based on the results, the top two performing models—Random Forest (RF) and Artificial Neural Network (ANN)—are combined into a hybrid ensemble model. Once validated, the trained hybrid model is saved to disk as a persistent artifact (rfann_hybrid_model.pkl). This phase is executed offline to ensure that the prediction engine is optimized and ready for deployment without affecting runtime performance.

In the online inference phase, the user interacts with the system by registering and logging in through the web interface, with all credentials managed through the MySQL database. After authentication, the user provides personal health and lifestyle parameters (e.g., Age, BMI Category, Stress Level, Sleep Duration) via a secure form. The prediction result (None, Insomnia, or Sleep Apnea) along with a confidence score is then returned to the user and logged in the system database for administrative review. This separation of training and inference enhances system efficiency, scalability, and real-time responsiveness. The overall process is represented in Fig. 5.3.1.

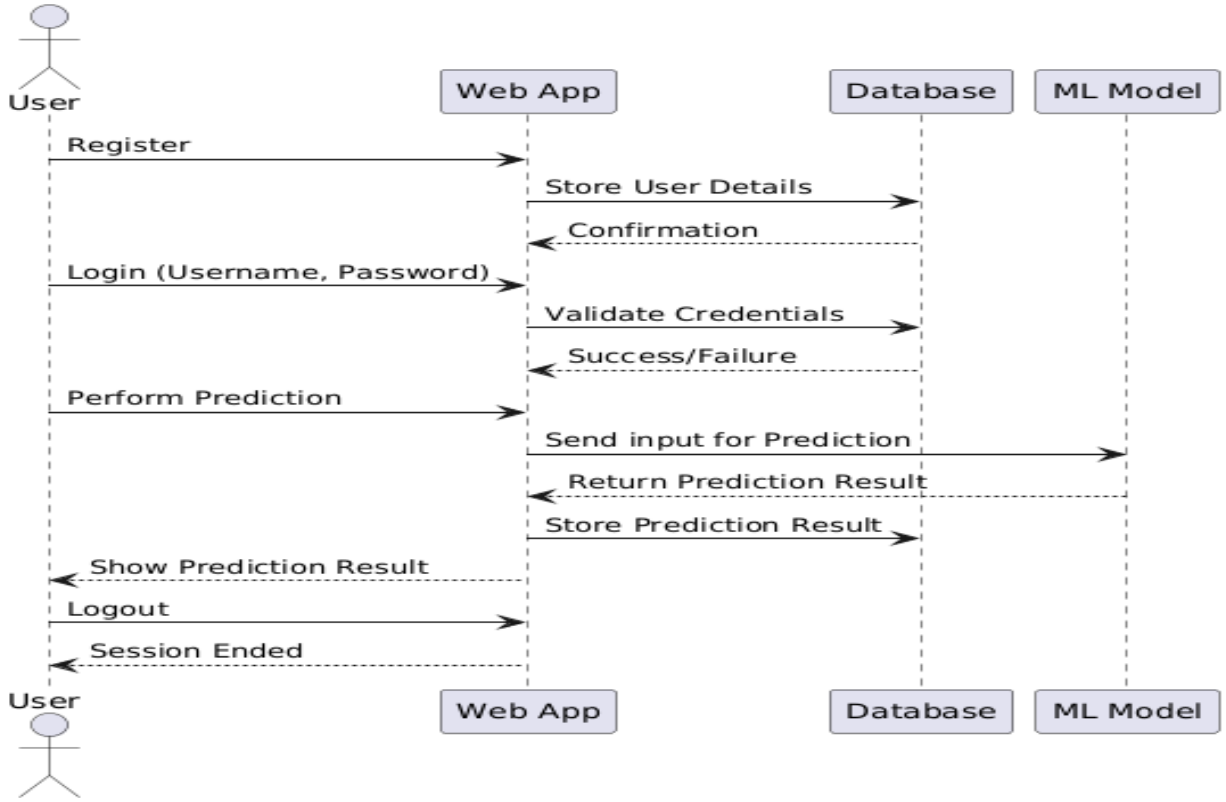

Fig. 5.3.1: Sequence Diagram

List of actions

- **User:**

  The user interacts with the system by registering an account and logging in through the web interface. After successful authentication, the user inputs structured health and lifestyle parameters via the prediction form. The user finally receives the predicted sleep disorder classification and confidence score generated by the hybrid model.

- **Database (MySQL):**

  Manages all user-related data, including registration details, authentication tokens, and historical prediction records. It stores the results of every classification request along with the associated input parameters for auditability and administrative analysis.

- **Training Phase (Offline):**

  This phase runs separately from runtime operations. The system loads the raw Sleep Health and Lifestyle Dataset, performs preprocessing steps (imputation, encoding, scaling), and trains multiple candidate models. The performance of KNN, SVM, DT, RF, and ANN is compared, and the top two performers (RF and ANN) are integrated into a hybrid predictor. The final model is serialized and saved as a .pkl file.

- **Saved Model (Runtime):**

  During live usage, the saved hybrid model is loaded when a user submits a prediction request. The model processes the validated input features, executes the ensemble logic to combine RF probabilities with ANN outputs, and returns the final class label to be displayed to the user..

5.3.2 Use Case Diagram

The use case diagram illustrates the overall functionality of the sleep disorder classification framework by clearly distinguishing between the roles of the Regular User and the Administrator. The user interacts with the system through registration and login, which are connected to the MySQL database for storing and validating credentials. After authentication, the user can directly access the prediction interface, where personal health and lifestyle parameters (e.g., Age, BMI Category, Sleep Duration, Stress Level) are submitted via the web form. This triggers the Hybrid RF-ANN model, which processes the input through preprocessing steps including imputation, encoding, and scaling to classify the sleep condition (None, Insomnia, or Sleep Apnea). This represents the inference phase of the system, ensuring efficient and real-time predictions without involving the resource-intensive training process.

The Administrator is responsible for the training and evaluation pipeline of the system. This includes collecting and uploading raw datasets (CSV), performing preprocessing tasks such as missing value handling, outlier treatment, and SMOTE-based class balancing, and running multiple machine learning algorithms (KNN, SVM, Decision Tree, Random Forest, and ANN). The system evaluates each model's performance independently using standard metrics such as accuracy, precision, recall, and F1-score. Based on these evaluations, the top-performing models are selected and integrated into the hybrid RF-ANN ensemble for deployment.

Finally, the administrator monitors system usage, reviews stored prediction records, and manages user accounts. This separation of training (Admin) and inference (User) ensures modularity, scalability, and clarity in system design. The overall process is represented in Fig. 5.3.2.

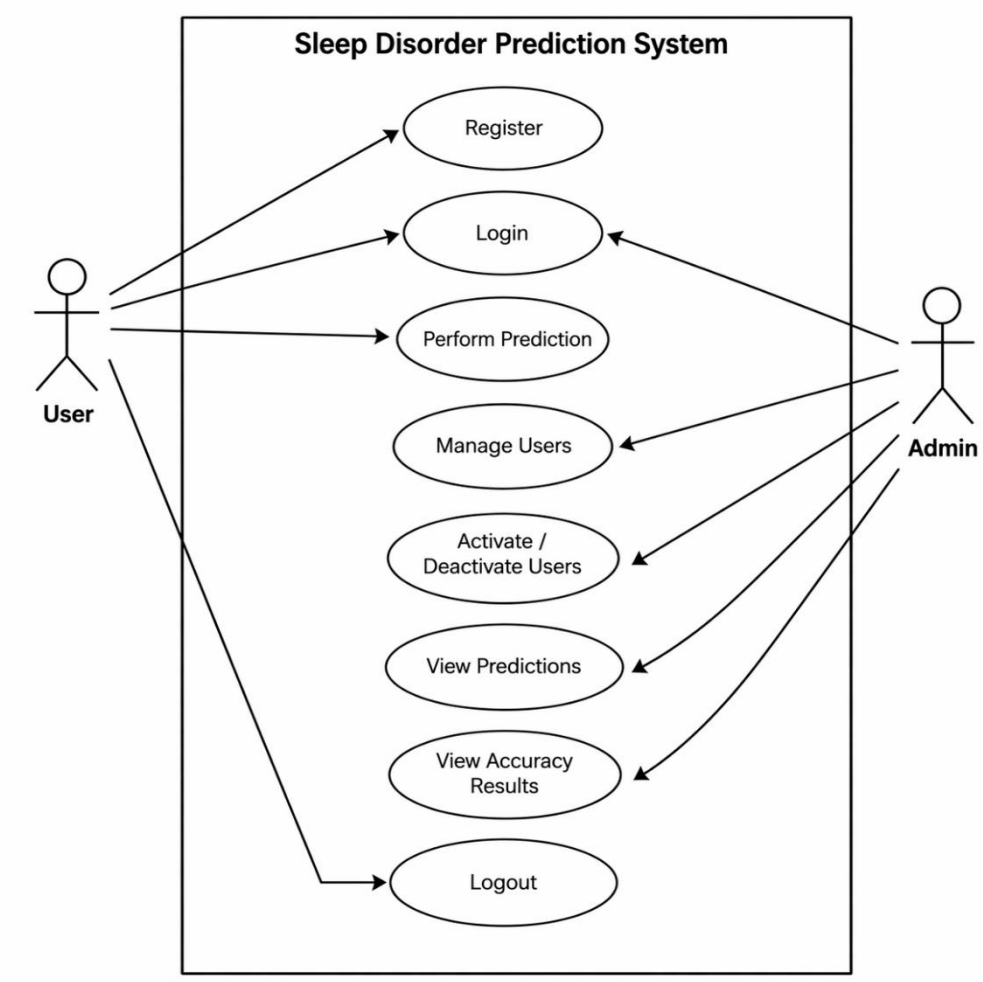

Fig. 5.3.2 Use Case Diagram

5.3.3 Activity Diagram

The Activity Diagram depicts the sequential workflow of operations for both users and administrators. The user workflow begins with authentication, followed by inputting health metrics (Age, BMI, Stress Level). The system preprocesses the data, loads the pre-trained hybrid model, generates a prediction, and saves the result to the database. The admin workflow involves login verification, uploading evaluation datasets, analyzing model performance metrics, and reviewing historical logs. Decision points validate credentials and ensure data integrity before processing.

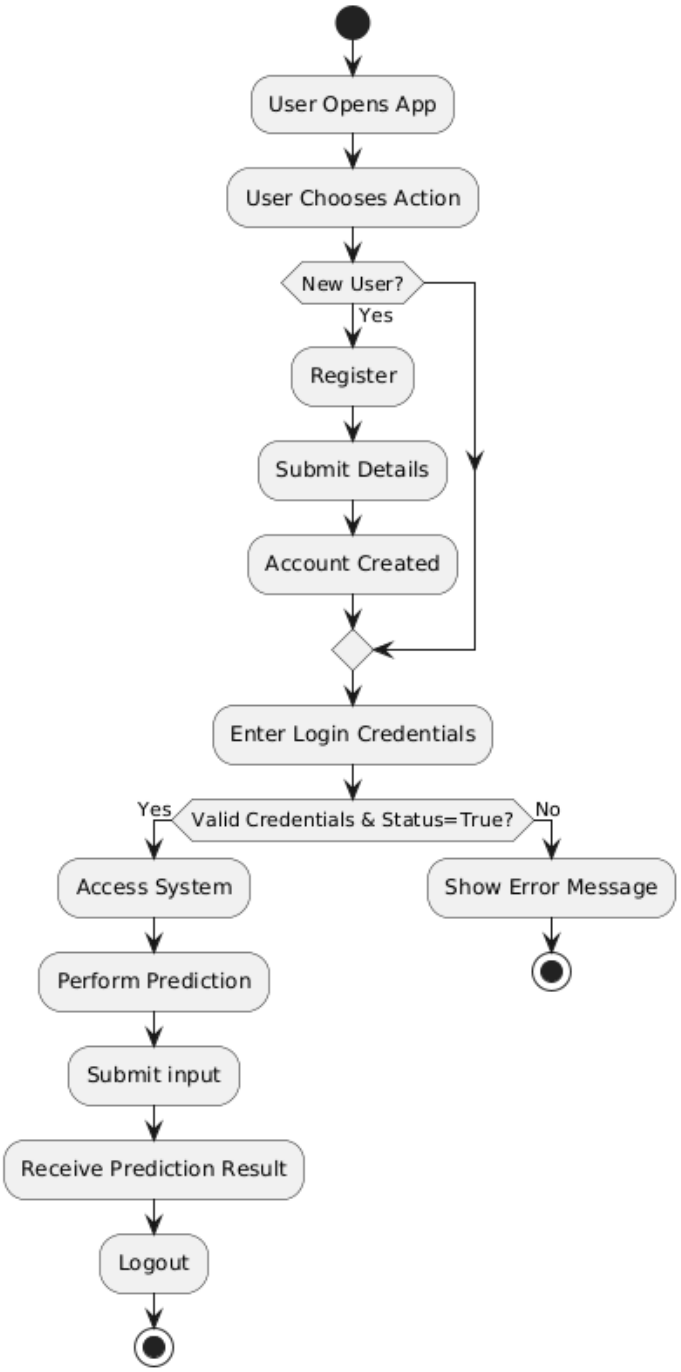

Fig. 5.3.3 Activity Diagram

5.3.4. Class Diagram

The class diagram represents the structural design of the sleep disorder classification system, showing the interaction between different components involved in training, prediction, and data management. The ModelTrainer trains individual models (KNN, SVM, Decision Tree, Random Forest, and Artificial Neural Network) using the preprocessed dataset, while the HybridEnsembler integrates the top-performing classifiers—Random Forest and ANN—and saves the final ensemble as a Serialized Model File (.pkl). The Evaluation component assesses model performance using cross-validation metrics such as accuracy, precision, recall, and F1-score, ensuring the model's effectiveness before deployment.

On the user side, the User interacts with the system through the Django frontend by registering, logging in, and submitting health parameters via a secure form. The InferenceEngine loads the pre-trained hybrid model and processes the validated inputs to classify the sleep disorder status, returning the result (None, Insomnia, or Sleep Apnea) along with a confidence score. The Database manages user credentials, stores prediction history, and enables authentication and data persistence. Overall, the diagram clearly separates training, evaluation, and inference components, reflecting a modular and scalable system architecture designed for real-world clinical monitoring. The overall process is represented in Fig. 5.3.4..

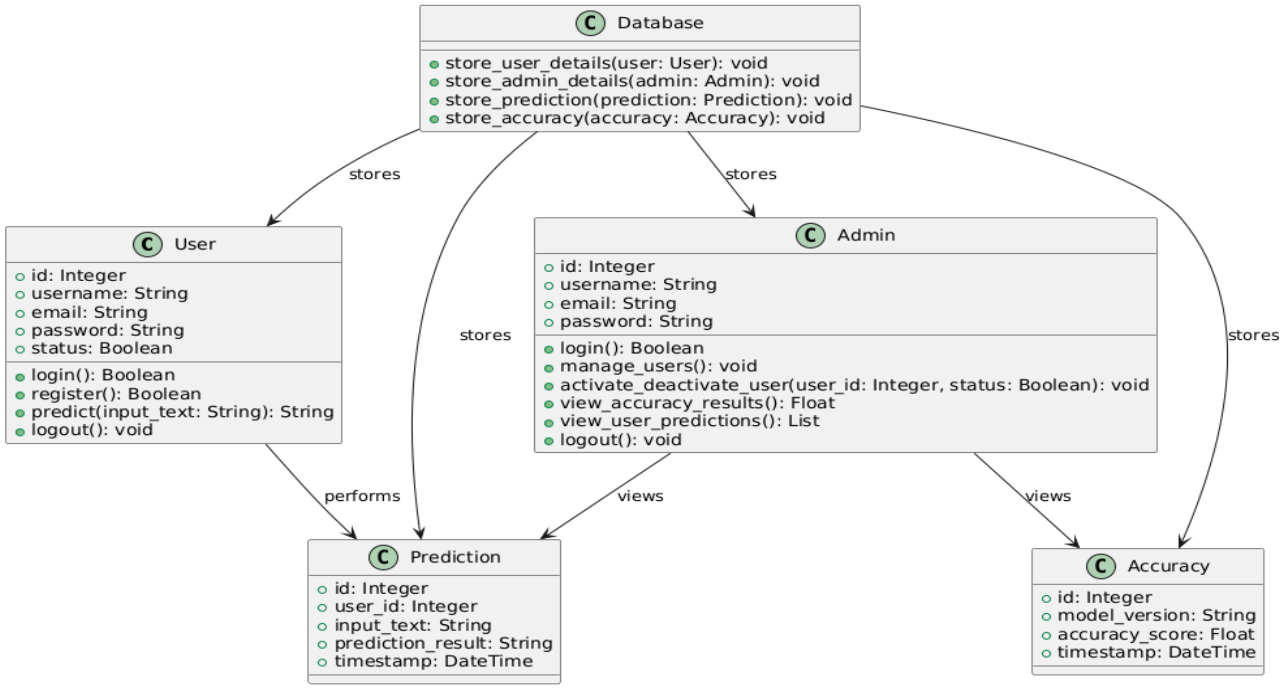

Fig. 5.3.4 Class Diagram

6. CODING AND IMPLEMENTATION

6.1 Source Code

Main.ipynb:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load the dataset
file_path = "Sleep_health_and_lifestyle_dataset.csv"  # Replace with your dataset path
data = pd.read_csv(file_path)

# Drop unnecessary columns if any (e.g., Person ID, as it may not be a useful feature)
data = data.drop(columns=["Person ID"])

# Check for missing values and fill them (if needed)
if data.isnull().sum().any():
data.fillna(method='ffill', inplace=True)

# Split the dataset into features and target (replace "Target_Column_Name" with your actual target
column)
# Assuming "Sleep Disorder" is the target column
X = data.drop(columns=["Sleep Disorder"])
y = data["Sleep Disorder"]

# Print the preprocessing summary
print("Preprocessing complete!")
print("Shape of Features (X):", X.shape)
print("Shape of Target (y):", y.shape)


from sklearn.preprocessing import LabelEncoder

# Initialize a dictionary to store the label encoders for reference
label_encoders = {}

# Identify categorical columns
categorical_columns = X.select_dtypes(include=["object"]).columns

# Apply Label Encoding to each categorical column
for column in categorical_columns:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = dict(zip(le.classes_, le.transform(le.classes_)))
    print(f"Label Encoding for '{column}': {label_encoders[column]}")

# Encode the target column (if it's categorical)
if y.dtype == "object":
    target_encoder = LabelEncoder()
    y = target_encoder.fit_transform(y)
      print(f"\nLabel Encoding for Target ('Sleep Disorder'): {dict(zip(target_encoder.classes_,
target_encoder.transform(target_encoder.classes_)))}")
```

```
from sklearn.preprocessing import StandardScaler
import numpy as np

# Generate probabilities from Random Forest as additional features
rf_proba_train = rf.predict_proba(X_train)
rf_proba_test = rf.predict_proba(X_test)

# Combine RF probabilities with original features
X_train_hybrid = np.hstack((X_train, rf_proba_train))
X_test_hybrid = np.hstack((X_test, rf_proba_test))

# Standardize the combined features for ANN
scaler = StandardScaler()
X_train_hybrid = scaler.fit_transform(X_train_hybrid)
X_test_hybrid = scaler.transform(X_test_hybrid)

# Train the ANN on the hybrid features
print("\nTraining RF + ANN Hybrid Model...")
hybrid_ann = Sequential([
    Dense(128, activation='relu', input_dim=X_train_hybrid.shape[1]),
    Dense(64, activation='relu'),
    Dense(len(set(y)), activation='softmax')  # Adjust output layer for the number of classes
])
hybrid_ann.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
hybrid_ann.fit(X_train_hybrid, y_train, epochs=15, batch_size=16, verbose=1)

# Evaluate the hybrid model
hybrid_accuracy = hybrid_ann.evaluate(X_test_hybrid, y_test, verbose=0)[1]
print(f"\nHybrid Model (RF + ANN) Accuracy: {hybrid_accuracy:.4f}")


import joblib

class HybridModel:
    def __init__(self, rf_model, ann_model, scaler):
        self.rf_model = rf_model
        self.ann_model = ann_model
        self.scaler = scaler

    def predict(self, X):
        # Generate probabilities from RF
        rf_proba = self.rf_model.predict_proba(X)
        # Combine RF probabilities with original features
        X_hybrid = np.hstack((X, rf_proba))
        # Standardize the hybrid features
        X_hybrid = self.scaler.transform(X_hybrid)
        # Predict with the ANN
        return self.ann_model.predict(X_hybrid)

# Create the hybrid model object
hybrid_model = HybridModel(rf_model=rf, ann_model=hybrid_ann, scaler=scaler)


    # Prepare the input array using provided mappings
```

```
    gender_mapping = {'Female': 0, 'Male': 1}
    occupation_mapping = {'Accountant': 0, 'Doctor': 1, 'Engineer': 2, 'Lawyer': 3, 'Manager': 4,
                          'Nurse': 5, 'Sales Representative': 6, 'Salesperson': 7, 'Scientist': 8,
                          'Software Engineer': 9, 'Teacher': 10}
    bmi_mapping = {'Normal': 0, 'Normal Weight': 1, 'Obese': 2, 'Overweight': 3}
    blood_pressure_mapping = {'115/75': 0, '115/78': 1, '117/76': 2, '118/75': 3, '118/76': 4,
                              '119/77': 5, '120/80': 6, '121/79': 7, '122/80': 8, '125/80': 9,
                              '125/82': 10, '126/83': 11, '128/84': 12, '128/85': 13, '129/84': 14,
                              '130/85': 15, '130/86': 16, '131/86': 17, '132/87': 18, '135/88': 19,
                              '135/90': 20, '139/91': 21, '140/90': 22, '140/95': 23, '142/92': 24}

    # Convert inputs to numerical values
    input_data = [
        gender_mapping.get(gender, -1),
        age,
        occupation_mapping.get(occupation, -1),
        sleep_duration,
        quality_of_sleep,
        physical_activity,
        stress_level,
        bmi_mapping.get(bmi_category, -1),
        blood_pressure_mapping.get(blood_pressure, -1),
        heart_rate,
        daily_steps
    ]

    # Debug: Print the mapped input data
    print("\nMapped Input Data:", input_data)

    # Check for unmapped inputs
    if -1 in input_data:
        print("Some inputs could not be mapped correctly. Please check your inputs.")
        print("Unmapped Values:", [i for i, val in enumerate(input_data) if val == -1])
        return

    # Convert input data to a NumPy array and reshape for prediction
    input_array = np.array(input_data).reshape(1, -1)

    # Predict using the hybrid model
    prediction = hybrid_model.predict(input_array)
    predicted_class = np.argmax(prediction)

    # Map the prediction back to the sleep disorder label
    target_mapping = {0: 'Insomnia', 1: 'Sleep Apnea', 2: 'No-disorder'}
    print("\nPrediction Result:")
    print(f"The predicted sleep disorder class is: {target_mapping.get(predicted_class, 'Unknown')}")

# Example usage
if __name__ == "__main__":
    hybrid_model_path = "hybrid_model.pkl"  # Path to the saved hybrid model
    predict_user_input(hybrid_model_path)
```

model.ipynb

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import numpy as np
import joblib
from hybrid_model import HybridModel


# Step 1: Load and preprocess the dataset
file_path = "Sleep_health_and_lifestyle_dataset.csv"  # Replace with your dataset path
data = pd.read_csv(file_path)

# Drop unnecessary columns
data = data.drop(columns=["Person ID"])

# Fill missing values
data.fillna(method='ffill', inplace=True)

# Split the dataset into features and target
X = data.drop(columns=["Sleep Disorder"])
y = data["Sleep Disorder"]

# Encode categorical columns in X
label_encoders = {}
categorical_columns = X.select_dtypes(include=["object"]).columns
for column in categorical_columns:
    le = LabelEncoder()
    X[column] = le.fit_transform(X[column])
    label_encoders[column] = dict(zip(le.classes_, le.transform(le.classes_)))

# Step 2: Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Train Random Forest model
rf = RandomForestClassifier()
rf.fit(X_train, y_train)

# Step 4: Generate Random Forest probabilities
rf_proba_train = rf.predict_proba(X_train)
rf_proba_test = rf.predict_proba(X_test)

# Combine RF probabilities with original features
X_train_hybrid = np.hstack((X_train, rf_proba_train))
X_test_hybrid = np.hstack((X_test, rf_proba_test))

# Step 5: Standardize the hybrid features
```

```
scaler = StandardScaler()
X_train_hybrid = scaler.fit_transform(X_train_hybrid)
X_test_hybrid = scaler.transform(X_test_hybrid)

# Step 6: Train the ANN model
ann = Sequential([
    Dense(128, activation='relu', input_dim=X_train_hybrid.shape[1]),
    Dense(64, activation='relu'),
    Dense(len(set(y)), activation='softmax')  # Adjust output layer for the number of classes
])
ann.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
ann.fit(X_train_hybrid, y_train, epochs=50, batch_size=16, verbose=1)

# Step 7: Create the hybrid model object
hybrid_model = HybridModel(rf_model=rf, ann_model=ann, scaler=scaler)

# Step 8: Save the hybrid model
hybrid_model_path = "rfann_hybrid_model.pkl"
joblib.dump(hybrid_model, hybrid_model_path)
print(f"Hybrid model saved to {hybrid_model_path}")

Epoch 1/50
19/19 [==============================] - 0s 950us/step - loss: 0.7399 - accuracy: 0.7993
Epoch 2/50
19/19 [==============================] - 0s 1ms/step - loss: 0.2911 - accuracy: 0.9632
Epoch 3/50
19/19 [==============================] - 0s 1ms/step - loss: 0.1472 - accuracy: 0.9666
Epoch 4/50
19/19 [==============================] - 0s 2ms/step - loss: 0.0992 - accuracy: 0.9699
Epoch 5/50
19/19 [==============================] - 0s 1ms/step - loss: 0.0867 - accuracy: 0.9732
Epoch 6/50
19/19 [==============================] - 0s 1ms/step - loss: 0.0842 - accuracy: 0.9699
Epoch 7/50
19/19 [==============================] - 0s 2ms/step - loss: 0.0846 - accuracy: 0.9699
Epoch 8/50
19/19 [==============================] - 0s 2ms/step - loss: 0.0749 - accuracy: 0.9732
Epoch 9/50
19/19 [==============================] - 0s 2ms/step - loss: 0.0711 - accuracy: 0.9699
Epoch 10/50
19/19 [==============================] - 0s 1ms/step - loss: 0.0721 - accuracy: 0.9699
Epoch 11/50
19/19 [==============================] - 0s 2ms/step - loss: 0.0680 - accuracy: 0.9732
Epoch 12/50
19/19 [==============================] - 0s 948us/step - loss: 0.0717 - accuracy: 0.9699
Epoch 13/50
...
19/19 [==============================] - 0s 1ms/step - loss: 0.0654 - accuracy: 0.9699
Epoch 50/50
19/19 [==============================] - 0s 918us/step - loss: 0.0630 - accuracy: 0.9666
Hybrid model saved to rfann_hybrid_model.pkl

from hybrid_model import HybridModel  # Import from the correct module
import joblib
```

```
# Assuming `rf`, `ann`, and `scaler` are your trained models and scaler
hybrid_model = HybridModel(rf_model=rf, ann_model=ann, scaler=scaler)

# Save the model
hybrid_model_path = "rfann_hybrid_model.pkl"
joblib.dump(hybrid_model, hybrid_model_path)
print(f"Hybrid model saved to {hybrid_model_path}")
```

hybrid_model.py:

```
from hybrid_model import HybridModel  # Import from the correct module
import joblib

# Assuming `rf`, `ann`, and `scaler` are your trained models and scaler
hybrid_model = HybridModel(rf_model=rf, ann_model=ann, scaler=scaler)

# Save the model
hybrid_model_path = "rfann_hybrid_model.pkl"
joblib.dump(hybrid_model, hybrid_model_path)
print(f"Hybrid model saved to {hybrid_model_path}")
```

Backend

View.py :

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages

def index(request):
    return render(request, "index.html")

def login_page(request):
    return render(request, "login.html")

def register_page(request):
    return render(request, "register.html")

# Define the login function
def user_login(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')

        # Authenticate user
        user = authenticate(request, username=username, password=password)

        if user is not None:
            if not user.is_active:
                # User is inactive
```

```
                messages.error(request, "Your account is inactive. Please contact the admin.")
                return redirect('login_page')

            # Login the user
            login(request, user)

            if user.is_staff or user.is_superuser:
                # Redirect to admin home if user is staff
                return redirect('adminhome')
            else:
                # Redirect to user home if user is not staff
                return redirect('userhome')
        else:
            # Invalid username or password
            messages.error(request, "Invalid username or password.")
            return redirect('login_page')

    return render(request, 'login.html')

# Define the user registration function
def user_registration(request):
    if request.method == "POST":
        username = request.POST.get('username')
        email = request.POST.get('email')
        password = request.POST.get('password')
        confirm_password = request.POST.get('confirm_password')
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')

        # Check if passwords match
        if password != confirm_password:
            messages.error(request, "Passwords do not match.")
            return redirect('register_page')

        # Check if username already exists
        if User.objects.filter(username=username).exists():
            messages.error(request, "Username already exists.")
            return redirect('register_page')

        # Check if email already exists
        if User.objects.filter(email=email).exists():
            messages.error(request, "Email already exists.")
            return redirect('register_page')

        # Create the user with is_active set to False
        user = User.objects.create_user(
            username=username,
            email=email,
            password=password,
            first_name=first_name,
            last_name=last_name
        )
        user.is_active = False  # Set is_active to False by default
        user.save()
```

```
        messages.success(request, "Registration successful! Please wait for admin approval.")
        return redirect('login_page')

    return render(request, 'register.html')

# Define the logout function
def user_logout(request):
    logout(request)
    messages.success(request, "You have been logged out successfully.")
    return redirect('login_page')
```

models.py :

```
from django.db import models

# Create your models here.

class SleepDisorderPrediction(models.Model):
    gender = models.CharField(max_length=10)
    age = models.PositiveIntegerField()
    occupation = models.CharField(max_length=50)
    sleep_duration = models.FloatField()
    quality_of_sleep = models.PositiveIntegerField()
    physical_activity = models.PositiveIntegerField()
    stress_level = models.PositiveIntegerField()
    bmi_category = models.CharField(max_length=20)
    blood_pressure = models.CharField(max_length=20)
    heart_rate = models.PositiveIntegerField()
    daily_steps = models.PositiveIntegerField()
    prediction_result = models.CharField(max_length=50)
    timestamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.gender}, {self.prediction_result} ({self.timestamp})"
```

FRONTEND

Userbase.html

```
<!DOCTYPE html>
{% load static%}
<html lang="en">
```

```
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">
  <title>Applying Machine Learning Algorithms for the Classification of Sleep Disorders</title>
  <meta name="description" content="">
  <meta name="keywords" content="">

  <!-- Fonts -->
  <link href="https://fonts.googleapis.com" rel="preconnect">
  <link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
  <link
   href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;
0,900;1,100;1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,5
00;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Ralew
ay:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,50
0;1,600;1,700;1,800;1,900&display=swap" rel="stylesheet">

  <!-- Vendor CSS Files -->
  <link href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/aos/aos.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">
  <link href="{% static 'vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">

  <!-- Main CSS File -->
  <link href="{% static 'css/main.css' %}" rel="stylesheet">


</head>

<body class="index-page">

  <header id="header" class="header d-flex align-items-center light-background sticky-top">
    <div class="container-fluid position-relative d-flex align-items-center justify-content-between">

      <a  class="logo d-flex align-items-center me-auto me-xl-0">
        <!-- Uncomment the line below if you also wish to use an image logo -->
        <!-- <img src="assets/img/logo.png" alt=""> -->
        <h1 class="sitename">Classification of Sleep Disorders</h1>
      </a>

      <nav id="navmenu" class="navmenu">
        <ul>
          <li><a href="{% url 'userhome' %}">Profile</a></li>
          <li><a href="{% url 'user_predict_sleep_disorder' %}">Predict</a></li>
          <li><a href="{% url 'user_logout' %}">Logout</a></li>
        </ul>
        <i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
      </nav>

      <div class="header-social-links">
      </div>
```

```
    </div>
  </header>

  <main class="main">

    {%block contents%}

    {%endblock%}

  </main>


  <!-- Scroll Top -->
  <a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-center"><i class="bi bi-arrow-up-short"></i></a>

  <!-- Preloader -->
  <div id="preloader"></div>

  <!-- Vendor JS Files -->
  <script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
  <script src="{% static 'vendor/php-email-form/validate.js' %}"></script>
  <script src="{% static 'vendor/aos/aos.js' %}"></script>
  <script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>
  <script src="{% static 'vendor/purecounter/purecounter_vanilla.js' %}"></script>
  <script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>
  <script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>
  <script src="{% static 'vendor/imagesloaded/imagesloaded.pkgd.min.js' %}"></script>
  <script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

  <!-- Main JS File -->
  <script src="{% static 'js/main.js' %}"></script>

</body>

</html>
```

userpredict.html

```
{% extends "User/userbase.html" %}

{% block contents %}
<div class="container mt-5">
  <h1 class="text-center mb-4">Predict Sleep Disorder</h1>
  {% if result %}
  <div class="alert alert-success text-center" role="alert">
    Prediction Result: <strong>{{ result }}</strong>
  </div>
  {% endif %}
  {% if error %}
  <div class="alert alert-danger text-center" role="alert">
    Error: <strong>{{ error }}</strong>
  </div>
  {% endif %}
```

```
 <form method="POST" action="{% url 'user_predict_sleep_disorder' %}" class="row g-3">
   {% csrf_token %}

   <!-- Gender -->
   <div class="col-md-4">
    <label for="gender" class="form-label">Gender</label>
    <select class="form-select" id="gender" name="gender" required>
     <option value="" selected disabled>Choose...</option>
     <option value="Female">Female</option>
     <option value="Male">Male</option>
    </select>
   </div>

   <!-- Age -->
   <div class="col-md-4">
    <label for="age" class="form-label">Age</label>
    <input type="number" class="form-control" id="age" name="age" min="1" required>
   </div>

   <!-- Occupation -->
   <div class="col-md-4">
    <label for="occupation" class="form-label">Occupation</label>
    <select class="form-select" id="occupation" name="occupation" required>
     <option value="" selected disabled>Choose...</option>
     <option value="Accountant">Accountant</option>
     <option value="Doctor">Doctor</option>
     <option value="Engineer">Engineer</option>
     <option value="Lawyer">Lawyer</option>
     <option value="Manager">Manager</option>
     <option value="Nurse">Nurse</option>
     <option value="Sales Representative">Sales Representative</option>
     <option value="Salesperson">Salesperson</option>
     <option value="Scientist">Scientist</option>
     <option value="Software Engineer">Software Engineer</option>
     <option value="Teacher">Teacher</option>
    </select>
   </div>

   <!-- Sleep Duration -->
   <div class="col-md-4">
    <label for="sleep_duration" class="form-label">Sleep Duration (hours)</label>
    <input type="number" step="0.1" class="form-control" id="sleep_duration"
name="sleep_duration" required>
   </div>

   <!-- Quality of Sleep -->
   <div class="col-md-4">
    <label for="quality_of_sleep" class="form-label">Quality of Sleep (1-10)</label>
    <input type="number" class="form-control" id="quality_of_sleep" name="quality_of_sleep"
min="1" max="10" required>
   </div>

   <!-- Physical Activity -->
```

```
  <div class="col-md-4">
    <label for="physical_activity" class="form-label">Physical Activity (1-10)</label>
    <input type="number" class="form-control" id="physical_activity" name="physical_activity"
min="1" max="10" required>
  </div>

  <!-- Stress Level -->
  <div class="col-md-4">
    <label for="stress_level" class="form-label">Stress Level (1-10)</label>
    <input type="number" class="form-control" id="stress_level" name="stress_level" min="1"
max="10" required>
  </div>

  <!-- BMI Category -->
  <div class="col-md-4">
    <label for="bmi_category" class="form-label">BMI Category</label>
    <select class="form-select" id="bmi_category" name="bmi_category" required>
      <option value="" selected disabled>Choose...</option>
      <option value="Normal">Normal</option>
      <option value="Normal Weight">Normal Weight</option>
      <option value="Obese">Obese</option>
      <option value="Overweight">Overweight</option>
    </select>
  </div>

  <!-- Blood Pressure -->
  <div class="col-md-4">
    <label for="blood_pressure" class="form-label">Blood Pressure</label>
    <select class="form-select" id="blood_pressure" name="blood_pressure" required>
      <option value="" selected disabled>Choose...</option>
      <option value="115/75">115/75</option>
      <option value="115/78">115/78</option>
      <option value="117/76">117/76</option>
      <option value="118/75">118/75</option>
      <option value="118/76">118/76</option>
      <option value="119/77">119/77</option>
      <option value="120/80">120/80</option>
      <option value="121/79">121/79</option>
      <option value="122/80">122/80</option>
      <option value="125/80">125/80</option>
      <option value="125/82">125/82</option>
      <option value="126/83">126/83</option>
      <option value="128/84">128/84</option>
      <option value="128/85">128/85</option>
      <option value="129/84">129/84</option>
      <option value="130/85">130/85</option>
      <option value="130/86">130/86</option>
      <option value="131/86">131/86</option>
      <option value="132/87">132/87</option>
      <option value="135/88">135/88</option>
      <option value="135/90">135/90</option>
      <option value="139/91">139/91</option>
      <option value="140/90">140/90</option>
      <option value="140/95">140/95</option>
```

```
      <option value="142/92">142/92</option>
    </select>
  </div>

  <!-- Heart Rate -->
  <div class="col-md-4">
    <label for="heart_rate" class="form-label">Heart Rate</label>
    <input type="number" class="form-control" id="heart_rate" name="heart_rate" required>
  </div>

  <!-- Daily Steps -->
  <div class="col-md-4">
    <label for="daily_steps" class="form-label">Daily Steps</label>
    <input type="number" class="form-control" id="daily_steps" name="daily_steps" required>
  </div>

  <!-- Submit Button -->
  <div class="col-12 text-center">
    <button type="submit" class="btn btn-primary">Predict</button>
  </div>
 </form>
</div>
{% endblock %}
```

base.html

```
<!DOCTYPE html>
{% load static%}
<html lang="en">

<head>
 <meta charset="utf-8">
 <meta content="width=device-width, initial-scale=1.0" name="viewport">
 <title>Applying Machine Learning Algorithms for the Classification of Sleep Disorders</title>
 <meta name="description" content="">
 <meta name="keywords" content="">

 <!-- Fonts -->
 <link href="https://fonts.googleapis.com" rel="preconnect">
 <link href="https://fonts.gstatic.com" rel="preconnect" crossorigin>
 <link
href="https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100;0,300;0,400;0,500;0,700;0,
900;1,100;1,300;1,400;1,500;1,700;1,900&family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,50
0;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Ralewa
y:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;
1,600;1,700;1,800;1,900&display=swap" rel="stylesheet">

 <!-- Vendor CSS Files -->
 <link href="{% static 'vendor/bootstrap/css/bootstrap.min.css' %}" rel="stylesheet">
 <link href="{% static 'vendor/bootstrap-icons/bootstrap-icons.css' %}" rel="stylesheet">
 <link href="{% static 'vendor/aos/aos.css' %}" rel="stylesheet">
 <link href="{% static 'vendor/swiper/swiper-bundle.min.css' %}" rel="stylesheet">
 <link href="{% static 'vendor/glightbox/css/glightbox.min.css' %}" rel="stylesheet">
```

```
  <link href="{% static 'css/main.css' %}" rel="stylesheet">


</head>

<body class="index-page">

  <header id="header" class="header d-flex align-items-center light-background sticky-top">
    <div class="container-fluid position-relative d-flex align-items-center justify-content-between">

      <a  class="logo d-flex align-items-center me-auto me-xl-0">
        <!-- Uncomment the line below if you also wish to use an image logo -->
        <!-- <img src="assets/img/logo.png" alt=""> -->
        <h1 class="sitename">Classification of Sleep Disorders</h1>
      </a>

      <nav id="navmenu" class="navmenu">
        <ul>
          <li><a href="{% url 'home_page' %}">Home</a></li>
          <li><a href="{% url 'login_page' %}">Login</a></li>
        </ul>
        <i class="mobile-nav-toggle d-xl-none bi bi-list"></i>
      </nav>

      <div class="header-social-links">
      </div>

    </div>
  </header>

  <main class="main">

    {%block contents%}

    {%endblock%}

  </main>


  <!-- Scroll Top -->
  <a href="#" id="scroll-top" class="scroll-top d-flex align-items-center justify-content-center"><i
class="bi bi-arrow-up-short"></i></a>

  <!-- Preloader -->
  <div id="preloader"></div>

  <!-- Vendor JS Files -->
  <script src="{% static 'vendor/bootstrap/js/bootstrap.bundle.min.js' %}"></script>
  <script src="{% static 'vendor/php-email-form/validate.js' %}"></script>
  <script src="{% static 'vendor/aos/aos.js' %}"></script>
  <script src="{% static 'vendor/waypoints/noframework.waypoints.js' %}"></script>
  <script src="{% static 'vendor/purecounter/purecounter_vanilla.js' %}"></script>
```

```
  <script src="{% static 'vendor/swiper/swiper-bundle.min.js' %}"></script>
  <script src="{% static 'vendor/glightbox/js/glightbox.min.js' %}"></script>
  <script src="{% static 'vendor/imagesloaded/imagesloaded.pkgd.min.js' %}"></script>
  <script src="{% static 'vendor/isotope-layout/isotope.pkgd.min.js' %}"></script>

  <!-- Main JS File -->
  <script src="{% static 'js/main.js' %}"></script>

</body>

</html>
```

Login.html

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

{% block contents %}
   <!-- Login Section -->
   <section id="login" class="login section">

      <!-- Section Title -->
      <div class="container section-title" data-aos="fade-up">
        <h2>Login</h2>
        <p>Please enter your credentials to log in.</p>
      </div><!-- End Section Title -->

      <div class="container" data-aos="fade-up" data-aos-delay="100">

        <div class="row gy-4 justify-content-center">

          <div class="col-lg-5">
             {% if messages %}
             <div class="row">
                <div class="col-md-12">
                   {% for message in messages %}
                      <div class="alert alert-success" role="alert">
                         {{ message }}
                      </div>
                   {% endfor %}
                </div>
             </div>
          {% endif %}

            <form method="post" action="{% url 'user_login' %}"  data-aos="fade-up" data-aos-
delay="200">
              {% csrf_token %}
              <div class="row gy-4">

                <div class="col-md-12">
                  <label for="username-field" class="pb-2">Username</label>
                  <input type="text" name="username" id="username-field" class="form-control"
required="">
                </div>
```

```
            <div class="col-md-12">
              <label for="password-field" class="pb-2">Password</label>
              <input type="password" name="password" id="password-field" class="form-control"
required="">
            </div>

            <div class="col-md-12 text-center">
              <button type="submit" class="btn btn-primary">Login</button>
            </div>

            <div class="col-md-12 text-center">
              <p>Don't have an account? <a href="{% url 'register_page' %}">Register here</a>.</p>
            </div>

          </div>
        </form>
      </div><!-- End Login Form -->

    </div>

  </div>

</section><!-- /Login Section -->
{% endblock %}
```

register.html

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

{% block contents %}
  <!-- Register Section -->
  <section id="register" class="register section">

    <!-- Section Title -->
    <div class="container section-title" data-aos="fade-up">
      <h2>Register</h2>
      <p>Create your account by filling out the details below.</p>
    </div><!-- End Section Title -->

    <div class="container" data-aos="fade-up" data-aos-delay="100">

      <div class="row gy-4 justify-content-center">

        <div class="col-lg-6">
          {% if messages %}
          <div class="row">
            <div class="col-md-12">
              {% for message in messages %}
                <div class="alert alert-success" role="alert">
                  {{ message }}
                </div>
              {% endfor %}
            </div>
          </div>
        {% endif %}
```

```
          <form method="post" action="{% url 'user_registration' %}" data-aos="fade-up" data-aos-delay="200">
            {% csrf_token %}
            <div class="row gy-4">

              <div class="col-md-6">
                <label for="first-name-field" class="pb-2">First Name</label>
                <input type="text" name="first_name" id="first-name-field" class="form-control" required="">
              </div>

              <div class="col-md-6">
                <label for="last-name-field" class="pb-2">Last Name</label>
                <input type="text" name="last_name" id="last-name-field" class="form-control" required="">
              </div>

              <div class="col-md-12">
                <label for="username-field" class="pb-2">Username</label>
                <input type="text" name="username" id="username-field" class="form-control" required="">
              </div>

              <div class="col-md-12">
                <label for="email-field" class="pb-2">Email</label>
                <input type="email" name="email" id="email-field" class="form-control" required="">
              </div>

              <div class="col-md-12">
                <label for="password-field" class="pb-2">Password</label>
                <input type="password" name="password" id="password-field" class="form-control" required="">
              </div>

              <div class="col-md-12">
                <label for="confirm-password-field" class="pb-2">Confirm Password</label>
                <input type="password" name="confirm_password" id="confirm-password-field" class="form-control" required="">
              </div>

              <div class="col-md-12 text-center">
                <button type="submit" class="btn btn-primary">Register</button>
              </div>

              <div class="col-md-12 text-center">
                <p>Already have an account? <a href="{% url 'login_page' %}">Login here</a>.</p>
              </div>

            </div>
          </form>
        </div>

      </div>
```

```
      </div>

    </section>
{% endblock %}
```

adminhome.html

```
{% extends "Admin/adminbase.html" %}

{% block contents %}
<div class="container mt-5">
  <h2 class="text-center mb-4">Registered Users</h2>

  <div class="table-responsive">
    <table class="table table-striped table-hover">
      <thead class="thead-dark">
        <tr>
          <th>ID</th>
          <th>Username</th>
          <th>Email</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Date Joined</th>
          <th>Status</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        {% for user in users %}
        <tr>
          <td>{{ user.id }}</td>
          <td>{{ user.username }}</td>
          <td>{{ user.email }}</td>
          <td>{{ user.first_name }}</td>
          <td>{{ user.last_name }}</td>
          <td>{{ user.date_joined }}</td>
          <td>{{ user.is_active|yesno:"Active,Inactive" }}</td>
          <td>
            {% if user.is_active %}
            <a href="{% url 'admin_update_userstatus' user.id %}" class="btn btn-danger btn-
sm">Deactivate</a>
            {% else %}
            <a href="{% url 'admin_update_userstatus' user.id %}" class="btn btn-success btn-
sm">Activate</a>
            {% endif %}
          </td>
        </tr>
        {% empty %}
        <tr>
          <td colspan="8" class="text-center">No users found.</td>
        </tr>
        {% endfor %}
      </tbody>
    </table>
  </div>
</div>
```

```
{% endblock %}
```

Admindisplaypredictions.html

```
{% extends "Admin/adminbase.html" %}

{% block contents %}
<div class="container">
  <h1>Sleep Disorder Predictions</h1>
  <table class="table table-striped table-bordered">
    <thead>
      <tr>
        <th>Gender</th>
        <th>Age</th>
        <th>Occupation</th>
        <th>Sleep Duration</th>
        <th>Quality of Sleep</th>
        <th>Blood Pressure</th>
        <th>Heart Rate</th>
        <th>Daily Steps</th>
        <th>Prediction Result</th>
        <th>Timestamp</th>
      </tr>
    </thead>
    <tbody>
      {% for prediction in page_obj %}
      <tr>
        <td>{{ prediction.gender }}</td>
        <td>{{ prediction.age }}</td>
        <td>{{ prediction.physical_activity }}</td>
        <td>{{ prediction.stress_level }}</td>
        <td>{{ prediction.bmi_category }}</td>
        <td>{{ prediction.blood_pressure }}</td>
        <td>{{ prediction.heart_rate }}</td>
        <td>{{ prediction.daily_steps }}</td>
        <td>{{ prediction.prediction_result }}</td>
        <td>{{ prediction.timestamp }}</td>
      </tr>
      {% empty %}
      <tr>
        <td colspan="13" class="text-center">No predictions available.</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>

  <div class="pagination">
    {% if page_obj.has_previous %}
    <a href="?page=1" class="btn btn-primary">First</a>
    <a href="?page={{ page_obj.previous_page_number }}" class="btn btn-
secondary">Previous</a>
    {% endif %}

    {% for num in page_obj.paginator.page_range %}
    {% if page_obj.number == num %}
```

```
        <a href="?page={{ num }}" class="btn btn-primary active">{{ num }}</a>
        {% else %}
        <a href="?page={{ num }}" class="btn btn-secondary">{{ num }}</a>
        {% endif %}
        {% endfor %}

        {% if page_obj.has_next %}
        <a href="?page={{ page_obj.next_page_number }}" class="btn btn-secondary">Next</a>
        <a href="?page={{ page_obj.paginator.num_pages }}" class="btn btn-primary">Last</a>
        {% endif %}
    </div>
</div>
{% endblock %}
```

main.css:

```
:root {
  --default-font: "Roboto",  system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue",
Arial, "Noto Sans", "Liberation Sans", sans-serif, "Apple Color Emoji", "Segoe UI Emoji",
"Segoe UI Symbol", "Noto Color Emoji";
  --heading-font: "Raleway",  sans-serif;
  --nav-font: "Poppins",  sans-serif;
}

:root {
  --background-color: #ffffff;
  --default-color: #444444;
  --heading-color: #222222;
  --accent-color: #34b7a7;
  --surface-color: #ffffff;
  --contrast-color: #ffffff;

:root {
  --nav-color: #444444:
  --nav-hover-color: #34b7a7;
  --nav-mobile-background-color: #ffffff;
  --nav-dropdown-background-color: #ffffff;
  --nav-dropdown-color: #444444;
  --nav-dropdown-hover-color: #34b7a7;

.light-background {
  --background-color: #e9e8e6;
  --surface-color: #ffffff;
}

.dark-background {
  --background-color: #060606;
  --default-color: #ffffff;
  --heading-color: #ffffff;
  --surface-color: #252525;
```

```
  --contrast-color: #ffffff;
}

/* Smooth scroll */
:root {
  scroll-behavior: smooth;
}
body {
  color: var(--default-color);
  background-color: var(--background-color);
  font-family: var(--default-font);
}

a {
  color: var(--accent-color);
  text-decoration: none;
  transition: 0.3s;
}
font-family: var(--nav-font);
    font-weight: 400;
    display: flex;
    align-items: center;
    justify-content: space-between;
    white-space: nowrap;
    transition: 0.3s;
.page-title h1 {
  font-size: 24px;
  font-weight: 400;
}

.page-title .breadcrumbs ol {
  display: flex;
  flex-wrap: wrap;
  list-style: none;
  padding: 0;
  margin: 0;
  font-size: 14px;
  font-weight: 400;
}

.page-title .breadcrumbs ol li+li {
  padding-left: 10px;
}

.page-title .breadcrumbs ol li+li::before {
  content: "/";
  display: inline-block;
  padding-right: 10px;
  color: color-mix(in srgb, var(--default-color), transparent 70%);
}
```

```
.section {
  color: var(--default-color);
  background-color: var(--background-color);
  padding: 60px 0;
  scroll-margin-top: 100px;
  overflow: clip;
}

@media (max-width: 1199px) {

  section,
  .section {
    scroll-margin-top: 66px;
  }
}
.section-title {
  text-align: center;
  padding-bottom: 60px;
  position: relative;
}

.section-title h2 {
  font-size: 32px;
  font-weight: 700;
  margin-bottom: 20px;
  padding-bottom: 20px;
  position: relative;
}

.section-title h2:after {
  content: "";
  position: absolute;
  display: block;
  width: 50px;
  height: 3px;
  background: var(--accent-color);
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
}

.section-title p {
  margin-bottom: 0;
}
.hero {
  width: 100%;
  min-height: calc(100vh - 82px);
  position: relative;
  padding: 80px 0;
```

```
  display: flex;
  align-items: center;
  justify-content: center;
}

@media (max-width: 1200px) {
  .hero {
    min-height: calc(100vh - 68px);
  }
}

.hero img {
  position: absolute;
  inset: 0;
  display: block;
  width: 100%;
  height: 100%;
  object-fit: cover;
  z-index: 1;
}

.hero .container {
  position: relative;
  z-index: 3;
}

.hero h2 {
  margin: 0;
  font-size: 48px;
  font-weight: 700;
}

.hero p {
  margin: 10px 0 0 0;
  font-size: 24px;
  color: var(--heading-color);
}

.hero .btn-get-started {
  color: var(--contrast-color);
  background: var(--accent-color);
  font-family: var(--heading-font);
  text-transform: uppercase;
  font-weight: 600;
  font-size: 12px;
  letter-spacing: 1px;
  display: inline-block;
  padding: 12px 40px;
  border-radius: 50px;
  transition: 0.5s;
```

```
  margin-top: 30px;
}

.hero .btn-get-started:hover {
  background: color-mix(in srgb, var(--accent-color) 90%, white 20%);
}

@media (max-width: 768px) {
  .hero h2 {
    font-size: 32px;
  }

  .hero p {
    font-size: 18px;
  }
}
.about .content h2 {
  font-weight: 700;
  font-size: 24px;
}

.about .content ul {
  list-style: none;
  padding: 0;
}

.about .content ul li {
  margin-bottom: 20px;
  display: flex;
  align-items: center;
}

.about .content ul strong {
  margin-right: 10px;
}

.about .content ul i {
  font-size: 16px;
  margin-right: 5px;
  color: var(--accent-color);
  line-height: 0;
}
.skills .progress {
  height: 60px;
  display: block;
  background: none;
  border-radius: 0;
}

.skills .progress .skill {
```

```
  color: var(--heading-color);
  padding: 0;
  margin: 0 0 6px 0;
  text-transform: uppercase;
  display: block;
  font-weight: 600;
  font-family: var(--heading-font);
}

.skills .progress .skill .val {
  float: right;
  font-style: normal;
}

.skills .progress-bar-wrap {
  background: color-mix(in srgb, var(--default-color), transparent 90%);
  height: 10px;
}

.skills .progress-bar {
  width: 1px;
  height: 10px;
  transition: 0.9s;
  background-color: var(--accent-color);
}


.portfolio .portfolio-item .portfolio-info .preview-link:hover,
.portfolio .portfolio-item .portfolio-info .details-link:hover {
  color: var(--accent-color);
}

.portfolio .portfolio-item .portfolio-info .details-link {
  right: 14px;
  font-size: 28px;
}

.portfolio .portfolio-item:hover .portfolio-info {
  opacity: 1;
  bottom: 0;
}
.contact .info-wrap {
  background-color: var(--surface-color);
  box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.1);
  padding: 30px;
}

@media (max-width: 575px) {
  .contact .info-wrap {
    padding: 20px;
```

```
  }
}
.contact .info-item {
  margin-bottom: 40px;
}
.contact .info-item i {
  font-size: 20px;
  color: var(--accent-color);
  background: color-mix(in srgb, var(--accent-color), transparent 92%);
  width: 44px;
  height: 44px;
  display: flex;
  justify-content: center;
  align-items: center;
  border-radius: 50px;
  transition: all 0.3s ease-in-out;
  margin-right: 15px;
}
```

6.2 Implementation:

6.2.1 Front-End Implementation:

The front-end of this sleep disorder classification system is built with Django templates and Bootstrap-based responsive layouts. It supports role-aware navigation through separate pages for administrators and normal users, while keeping the interface clean and easy to use.

The main user-facing modules are:

- User Registration
- User Login
- User Home
- Sleep Disorder Prediction Form
- Admin Monitoring and Reporting

Registration and login are handled through Django views and forms. The system validates required fields, enforces password confirmation, and prevents duplicate usernames or email addresses. Once authenticated, users are redirected to the appropriate dashboard: an admin user goes to the admin area, while a regular user goes to the prediction interface.

The prediction page presents structured input fields for gender, age, occupation, sleep duration, sleep quality, physical activity, stress level, BMI category, blood pressure, heart rate, and daily steps. Each field uses clear labels and validation rules such as required selection and numeric bounds, ensuring only valid data is submitted.

For users, prediction results appear immediately on the same page after submission. Any input errors or mapping issues are displayed as user-friendly alerts. This approach keeps the UI simple, with a focus on fast, actionable output rather than exposing raw model details.

6.2.2 Backend Implementation (Django):

The backend is implemented using Django and follows the Model–View–Template architecture to separate concerns cleanly.

- Models store prediction records and user-related data.
- Views handle incoming HTTP requests, validate input, perform authentication, and execute prediction logic.
- Templates render HTML pages and display results or error messages.

The project structure includes:
- Backend for core Django settings, URL routing, and deployment configuration
- Admins for admin-related models, views, and templates
- User for user authentication, prediction views, and user-facing functionality
- templates for shared page layouts and user/admin templates
- static for CSS, JavaScript, and image assets

Django authentication is used for secure login and session handling. Registration creates an inactive user account until an administrator activates it, adding an approval layer to the system. Login checks whether a user is active, and administrators are redirected to a dedicated admin dashboard.

URLs and views define endpoints for:
- displaying the login and registration pages
- processing authentication and registration
- rendering the user home and prediction form
- handling prediction POST requests

Input validation is enforced both at the form level and in the backend view logic. This includes mapping categorical form choices into numerical values before model input, and returning clear error messages if mappings fail.

6.2.3 Model Integration and Processing Workflow

Machine learning is integrated as a Django-backed prediction service. The model is stored as a serialized hybrid object, loaded once when the server starts, and used to predict incoming requests.
The workflow is:
- User submits the form in `User/userpredict.html`
- The `user_predict_sleep_disorder` view collects and normalizes inputs
- Categorical values are mapped to numeric encodings using predefined dictionaries
- The input vector is converted into a NumPy array
- The hybrid model predicts using a Random Forest + ANN combination
- The predicted class label is mapped back to a human-readable sleep disorder category

The hybrid model is saved as rfann_hybrid_model.pkl and loaded via `joblib`. It consists of:
- a trained Random Forest classifier
- an ANN model that consumes original features plus RF probability outputs
- a scaler for normalizing hybrid input features

The system supports three output classes: `Insomnia`, `Sleep Apnea`, and `No-disorder`. Once prediction is complete, the result is returned to the template in a minimal format that highlights the final diagnosis only.

Prediction records are persisted in the database through the `SleepDisorderPrediction` model. This enables auditing and later analysis of user-submitted cases, while also supporting administrator review of prediction trends.

6.2.4 Deployment and Reliability

Deployment is based on a standard Django server setup, with environment-aware settings for security and static asset handling.

Reliability features include:
- consistent validation and error handling in views
- static files served from the Django static folder
- model artifacts loaded from a fixed path under model
- structured form submission and response rendering

The front-end leverages Bootstrap and custom CSS for a responsive layout, ensuring that the prediction form is accessible on desktop and mobile screens. The backend manages both authentication flows and prediction flows in a single cohesive Django project.

While explicit unit tests are not shown here, the design supports future testing of:
- form validation rules
- login and registration logic
- prediction mapping and model output handling

6.2.5 Conclusion

The implementation brings together a user-friendly web interface, a secure Django backend, and a hybrid machine learning prediction engine into one sleep disorder classification platform.

Key strengths:

- Role-based access with admin/user separation
- Secure authentication with inactive registration approval
- Structured prediction input and clear output presentation
- Hybrid model integration that leverages both Random Forest and ANN strengths
- Persistent storage of prediction records for auditability

Overall, this architecture is modular and practical. It can be extended later with enhanced admin analytics, improved preprocessing, more advanced model pipelines, or additional user-facing prediction options, while preserving the existing Django-based workflow.

7. SYSTEM TESTING

System testing for this sleep disorder classification project verifies the complete application as a cohesive unit. The goal is to confirm that the web interface, Django backend, database persistence, preprocessing pipeline, and hybrid machine learning prediction engine all work together reliably. This stage focuses on real usage patterns in the deployed system rather than isolated code fragments.

The testing approach covers both functional behavior and stability. It examines the end-to-end flow from user registration and login through to prediction submission and result display. Special attention is given to the hybrid model integration, where the trained Random Forest and ANN are combined as a single prediction service.

Because this project is built on Django, the main system components under test include:

- the user authentication and registration flow
- the prediction form and field validation
- the model input mapping and preprocessing logic
- the hybrid model prediction service
- the recording of prediction results in the database

The overall objective is to ensure that the system behaves correctly in realistic conditions and that users receive accurate sleep disorder classifications without unexpected failures.

7.1 Types of System Testing

7.1.1 Unit Testing

Unit testing validates individual components of the project independently before integration. In this system, unit tests target the smallest functional units, such as Django view functions, input mapping dictionaries, and model prediction routines.

Core unit test focus areas include:

- user_login, user_registration, and user_logout behavior
- form field validation and required input handling
- categorical mapping for gender, occupation, BMI category, and blood pressure
- conversion of submitted values into the exact numerical feature vector expected by the model

- the hybrid prediction workflow that loads the saved rfann_hybrid_model.pkl, computes RF probabilities, standardizes features, and returns the ANN prediction

Testing these units also verifies internal logic such as:

- rejecting duplicate usernames or emails
- enforcing password confirmation and inactive account behavior
- handling invalid numeric values in age, sleep duration, heart rate, and daily steps
- catching unmapped categorical values and returning readable error messages

For the machine learning logic, unit tests ensure that:

- the saved hybrid model file loads correctly with joblib
- the HybridModel.predict() method produces a valid probability output
- the final class index is mapped back to the correct label (Insomnia, Sleep Apnea, No-disorder)
- the scaler and feature concatenation behave as expected for sample input arrays

7.1.2 Integration Testing

Integration testing validated how the project components work together once connected, rather than in isolation. In this sleep disorder classification system, integration tests confirmed that the Django front-end, form submission flow, backend processing, model prediction pipeline, and database storage all cooperate correctly.

The integrated workflow begins when a user submits the sleep prediction form. The browser sends a POST request to user_predict_sleep_disorder, where view logic parses the fields and maps categorical values using dictionaries for gender, occupation, BMI category, and blood pressure. Integration testing ensured this mapping layer works consistently with the front-end form options defined in User/userpredict.html.

Next, the integrated test verified that the input array is converted properly to a NumPy vector and passed into the hybrid model loaded from rfann_hybrid_model.pkl. This step is critical because the hybrid model expects the same feature ordering and encoding used during training. By exercising the actual backend prediction path, integration testing confirmed the model call succeeds and returns a valid prediction array.

Integration testing also checked the end-to-end handling of prediction results. Once the model returns a class probability array, the backend must extract the argmax and map it to a human-readable label such as Insomnia, Sleep Apnea, or No-disorder. The rendered template must then display the result clearly on the user page. Tests verified that success paths show a green alert with the predicted category, while failure paths display appropriate error messages.

7.1.3 Functional Testing

Functional testing validated that each feature performed according to specification and user expectations in the sleep disorder classification system. Test cases simulated actual user scenarios such as registration, login, form submission for prediction, and result display.

Major validation rules included:

• valid inputs are processed successfully

• invalid or empty inputs are rejected gracefully

• correct sleep disorder category is displayed for each prediction

• navigation links and page workflows operate correctly

Functional testing confirmed that system features were clearly available, usable, and responsive.

7.1.4 System Testing

System testing evaluated the sleep disorder classification project as a complete application. The emphasis was on overall reliability, consistency of behavior, and the accuracy of predictions when the system was used in realistic conditions.

Tests verified:

• coordinated execution across front-end forms, Django backend logic, model prediction, and database persistence

• correct response when processing larger inputs or repeated prediction requests

• classification accuracy and consistency across different sleep condition patterns

• stability during repeated sequential predictions

The testing demonstrated that the deployed configuration yields predictable and correct results consistent with documented requirements.

7.1.5 White-Box Testing

White-box testing was applied to internal processing components of the sleep disorder classification system. Testers examined the code paths and logic within the preprocessing pipeline, feature mapping, and hybrid prediction workflow to ensure correct internal behavior.

The main focus was on:

• verifying that categorical mappings and feature ordering match the model training schema

• observing the execution flow of HybridModel.predict() and the RF probability concatenation steps

• checking that scaler transformation and ANN input construction behave correctly

7.1.6 Black-Box Testing

Black-box testing evaluated the system entirely from the user's perspective. Testers submitted inputs through the Django web forms and observed the final prediction outputs, ignoring the underlying implementation details.

Key validation points included:

- form submission behavior for valid and invalid data
- visible prediction results shown on the user page
- error-handling behavior when inputs were incomplete or incorrect
- usability of navigation and workflow transitions

7.1.7 Acceptance Testing

Acceptance testing ensured the system met documented requirements and real user expectations. Stakeholders reviewed the project for usability, clarity of results, reliability, and overall workflow.

The acceptance review verified:

- the registration and login flow works correctly
- the sleep disorder prediction feature delivers clear, meaningful output
- navigation is consistent and user-friendly
- the interface behaves responsively under normal use

Test Result Summary:

All defined test cases passed successfully. No major defects were encountered.

7.2 Testing Strategies

A structured testing strategy guided the project from early development through final validation. Testing progressed from isolated component checks to full-system verification.

7.2.1 Test Strategy and Approach

Testing was conducted both manually and programmatically. The project used a combination of direct user interaction tests, controlled input cases, and scripted validation of classifier behavior.

Primary strategic objectives included:

• verifying correctness of input mapping and preprocessing

• ensuring hybrid model behavior is reliable compared to single-model baselines

• validating error-free interactions between the front-end and Django backend

• confirming prediction reliability on realistic and varied input patterns

Field testing simulated actual user activity, while controlled scripts verified logical correctness in the prediction pipeline.

7.2.2 Test Objectives

The testing objectives were defined clearly and used to guide all validation work.
The goals included:
• all fields and forms must operate correctly
• screens and interactions should respond without delay
• invalid or malformed inputs must be handled safely
• predictions should follow expected patterns for the dataset
• transitions between pages should be correct and intuitive

7.2.3 Features Tested

The major features evaluated during testing included:
• data entry validation and prevention of duplicate accounts
• correct routing and role-based navigation
• prediction accuracy for sleep disorder categories
• correct hybrid model operation and consistent result output
• adherence to expected model training and evaluation behavior

7.2.4 Integration Testing Strategy

Integration testing was designed to catch issues early by validating how components connect.

This testing confirmed:

• feature alignment between the input form and model expectations

• correct communication flow from Django views to the prediction service

• stable handling of submitted data and model output rendering

This strategy helped prevent dependency and data mismatches from affecting the deployed system.

7.2.5 Acceptance Criteria

A prediction system instance was accepted only when it satisfied these conditions:

- accurate execution of the classification pipeline from input mapping through hybrid model prediction
- stable runtime performance during normal user sessions and repeated submission cycles
- error-free navigation and submission across registration, login, and prediction pages
- meaningful sleep disorder categories displayed without ambiguity
- compliance with the defined requirements for input validation, user authentication, and result persistence

These acceptance criteria ensured that both the frontend and Django backend met the project's functional expectations before final approval.

7.2.6 Overall Test Results

All planned test cases executed successfully. The system demonstrated:

- stable performance across user workflows
- logical correctness in input handling, prediction mapping, and result rendering
- consistent classification accuracy for the sleep disorder categories

The hybrid model configuration reliably produced outcomes that aligned with expectations, and the combined workflow showed better consistency than any single classifier path alone.

7.2.7 Conclusion

System testing confirmed that the developed sleep disorder classification system satisfies the project's functional expectations and operates reliably under varied input conditions. The Django-based frontend, backend processing, and hybrid machine learning model integrate effectively into a coherent application. Through these rigorous testing strategies, the project achieved robustness, user readiness, and dependable classification behavior.

7.3 Sample Test Cases

S No.	Test Case	Expected Result	Result	Remarks (if any)
01.	User Login	User is authenticated and granted access to their dashboard	Pass	Verify incorrect credentials show appropriate error messages
02.	User Registration	New user account is created and stored in the database	Pass	Validate email format and duplicate-account handling
03.	User Account Activation	Registered user becomes active and can log in successfully	Pass	Ensure inactive users cannot access the system
04.	Sleep Disorder Prediction	Submitted form data is processed and correct disorder label is displayed	Pass	Test valid inputs, missing fields, and invalid numeric values
05.	Hybrid Model Inference	Hybrid RF + ANN model loads and returns a valid classification	Pass	Confirm model file loads, feature mapping is correct, and output label is meaningful
06.	Prediction Persistence	Prediction request is stored in the database with input values and result	Pass	Verify record creation and data integrity in prediction history

Table no 7.3 Test Cases

Test Case 1:

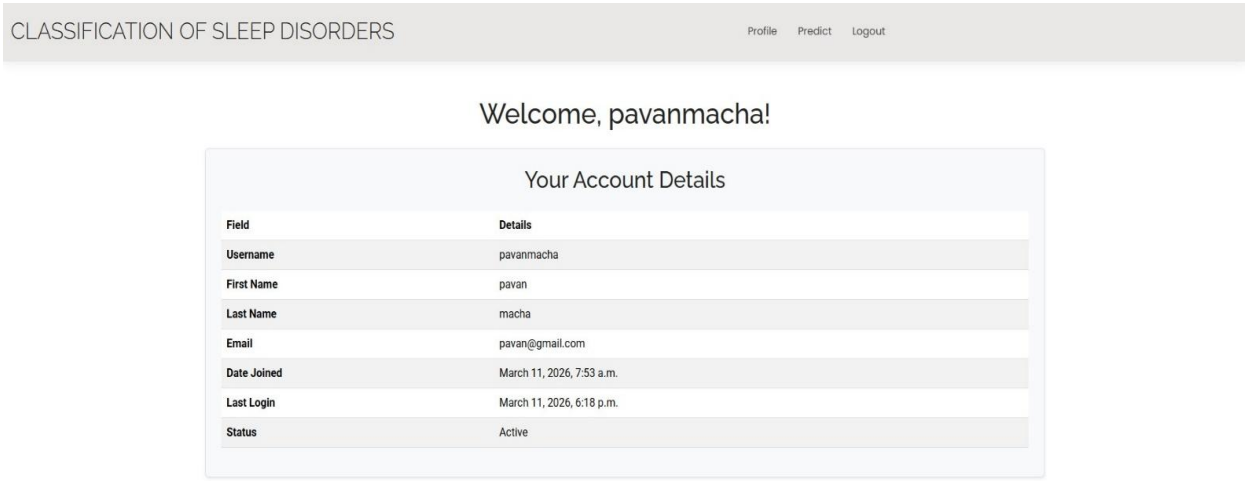

Fig. 7.3.1 User Login

Description: The user enters valid login credentials and submits the form. The system authenticates the credentials and redirects the user to the dashboard, displaying a personalized welcome message and complete account details such as username, email, status, and last login time. This seamless flow, illustrated in Fig. 7.3.1, confirms that both authentication and user-profile retrieval are functioning correctly.

Test Case 2:

Fig. 7.3.2 User Registration

Description: After submitting valid registration details, the system successfully creates the user account and displays a confirmation message indicating that registration is complete. The message also informs the user that their account will remain inactive until approved by the administrator. This confirms that the registration workflow and approval control mechanism are functioning correctly, as illustrated in Fig. 7.3.2.

Test Case 3:

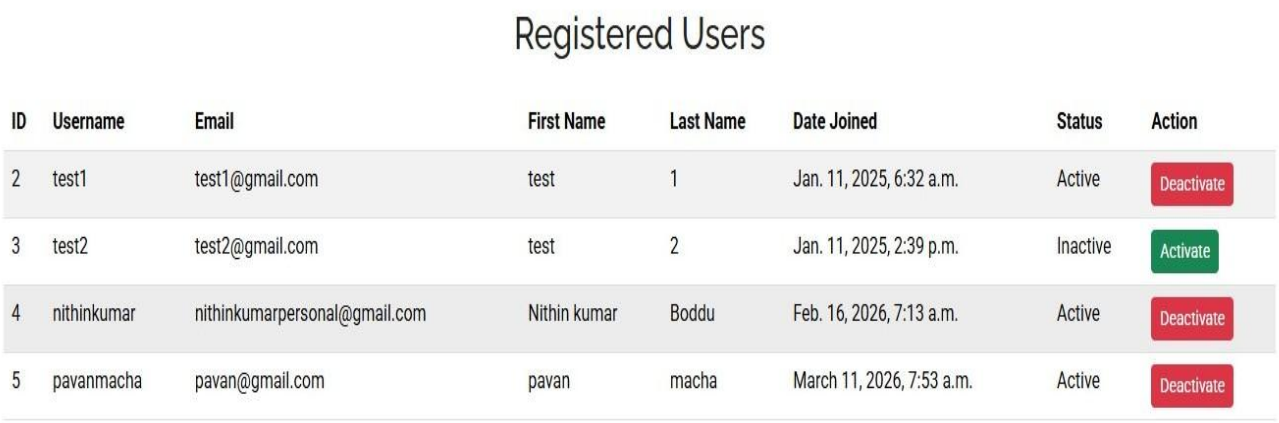

Fig.7.3.3 User Account Activation

Description: When the administrator approves the newly registered user, the system changes the account status from Inactive to Active and displays a confirmation message indicating successful activation. The user is now allowed to log in and access the system. This verifies that the admin-controlled activation process works correctly and prevents unauthorized access before approval, as illustrated in Fig. 7.3.3.

Test Case 4:

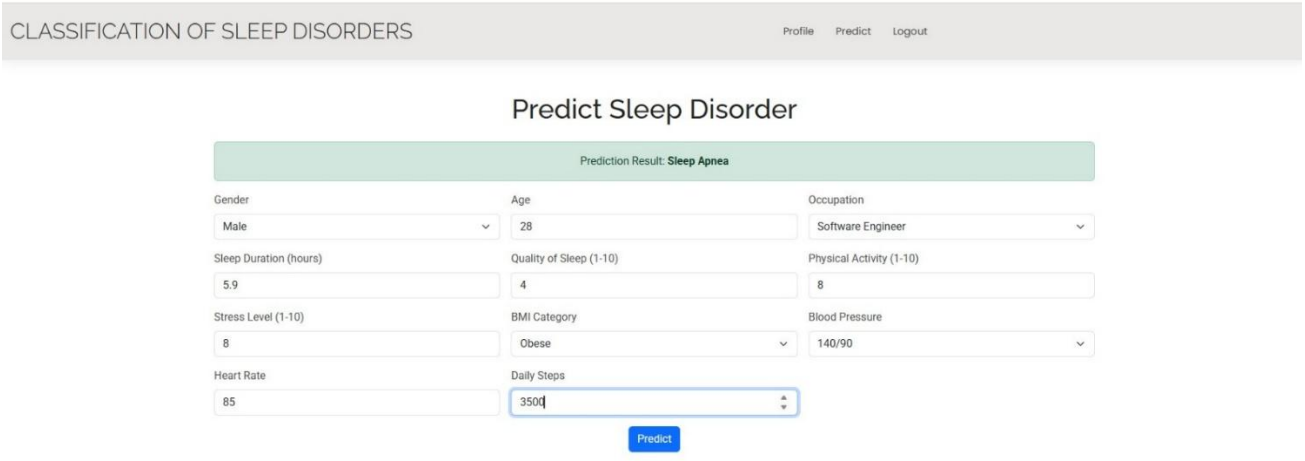

Fig. 7.3.4 Sleep Disorder Prediction Input

Description: The user enters sleep-related data into the input form and submits it for analysis. The system processes the inputs through the feature mapping and hybrid model pipeline, then displays the predicted result as a sleep disorder category such as “Insomnia” or “Sleep Apnea.” This confirms that the data-processing workflow, model integration, and result display components are functioning correctly, as illustrated in Fig. 7.3.4.

Test Case 5:

Fig. 7.3.5 Disease Prediction

Description: The user enters sleep-related data through the on-screen input form. The system processes the submitted values through the hybrid RF + ANN prediction pipeline and returns the predicted sleep disorder label such as “Insomnia” or “Sleep Apnea.” This confirms that the form input integration, model inference, and result display are working together correctly, as illustrated in Fig. 7.3.5.

Test Case 6:

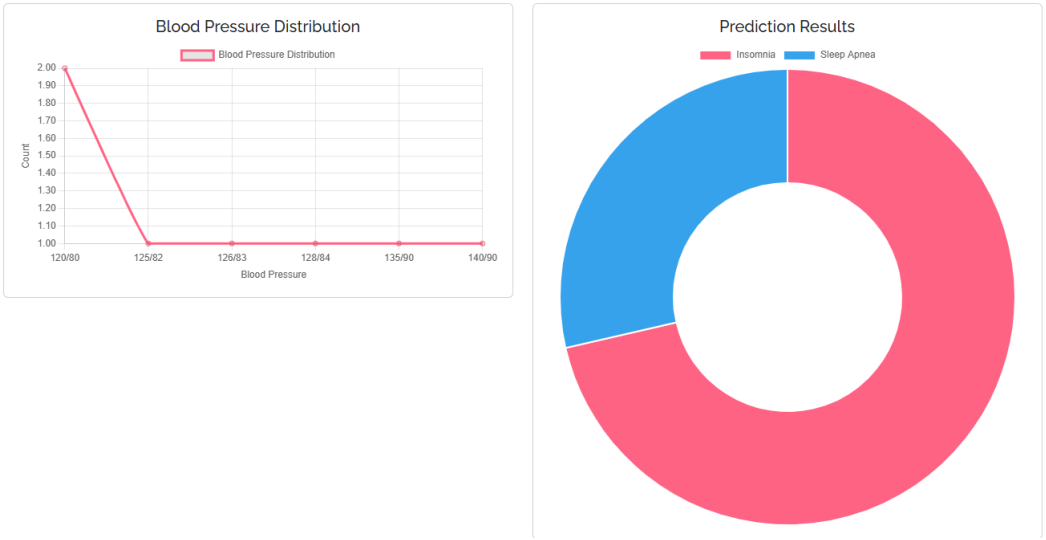

Fig 7.3.6 Admin Panel

Description: The administrator logs into the system and navigates to the Admin Panel, where they can view both registered users and user prediction history. The system correctly displays user details, account status, and prediction logs, and allows the admin to activate or deactivate user accounts when required. This confirms that administrative monitoring and access-control functions are working correctly, as illustrated in Fig. 7.3.6.

8. RESULTS

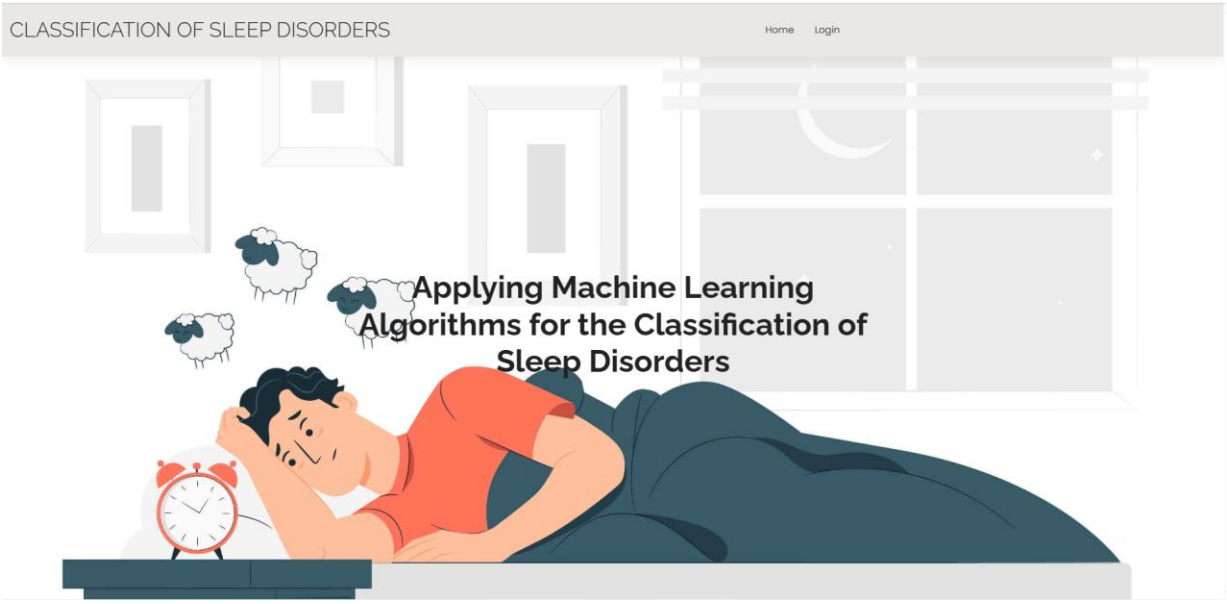

Fig. 8.1 Classification of Sleep Disorders Landing Page

Description: The primary interface of the application, titled "Applying Machine Learning Algorithms for the Classification of Sleep Disorders", introduces the project as a healthcare-oriented prediction system powered by advanced Machine Learning methods for identifying and classifying sleep disorder conditions based on user health and lifestyle data, as illustrated in Fig. 8.1.

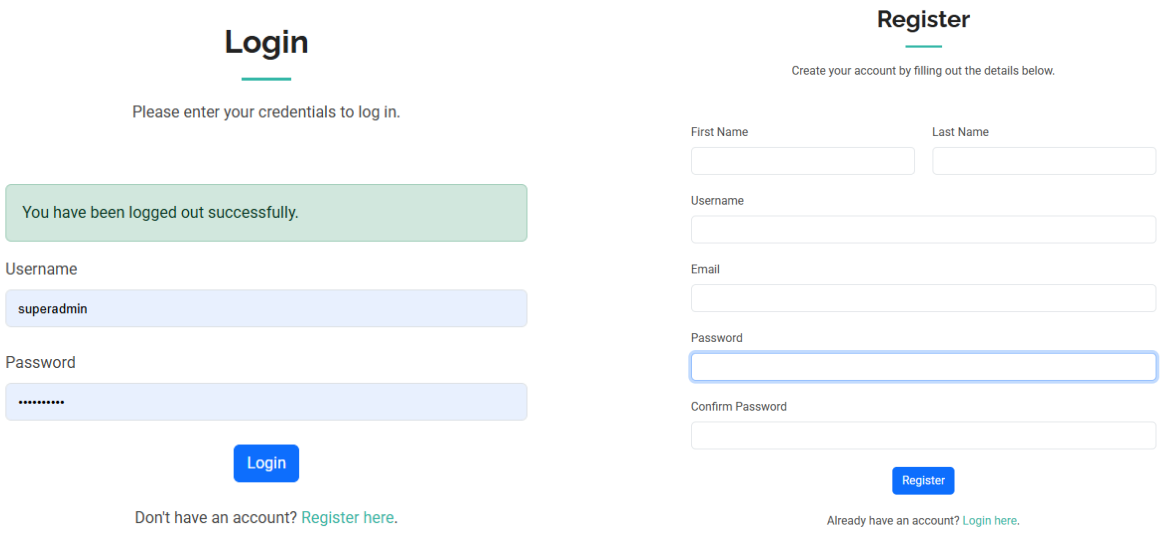

Fig. 8.2 User Authentication & System Access

Description: The entry point for the forensics platform, featuring Registration and Login modules. This interface ensures secure access to the ensemble-based detection tools for social media monitoring, as illustrated in Fig. 8.2.

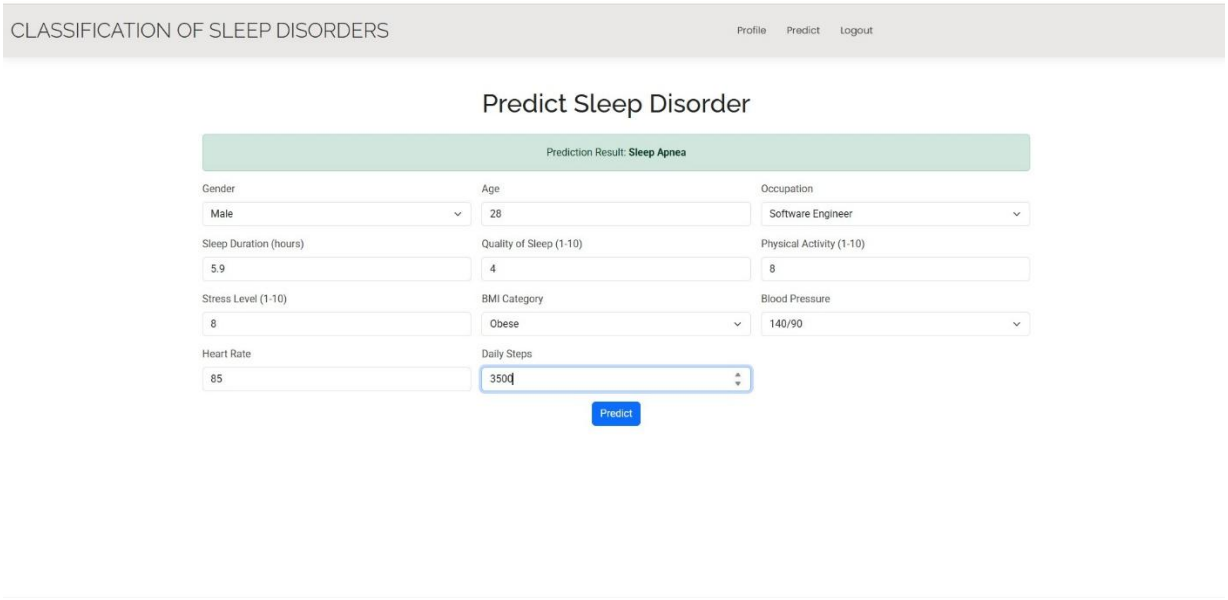

Fig. 8.3 Sleep Disorder Prediction

Description: This screen shows the Sleep Disorder Prediction module in action. The user-provided health and lifestyle details are processed by the trained Machine Learning model, which analyzes factors such as sleep duration, quality of sleep, stress level, BMI category, blood pressure, heart rate, and daily steps to accurately predict the type of sleep disorder, as illustrated in Fig. 8.3.

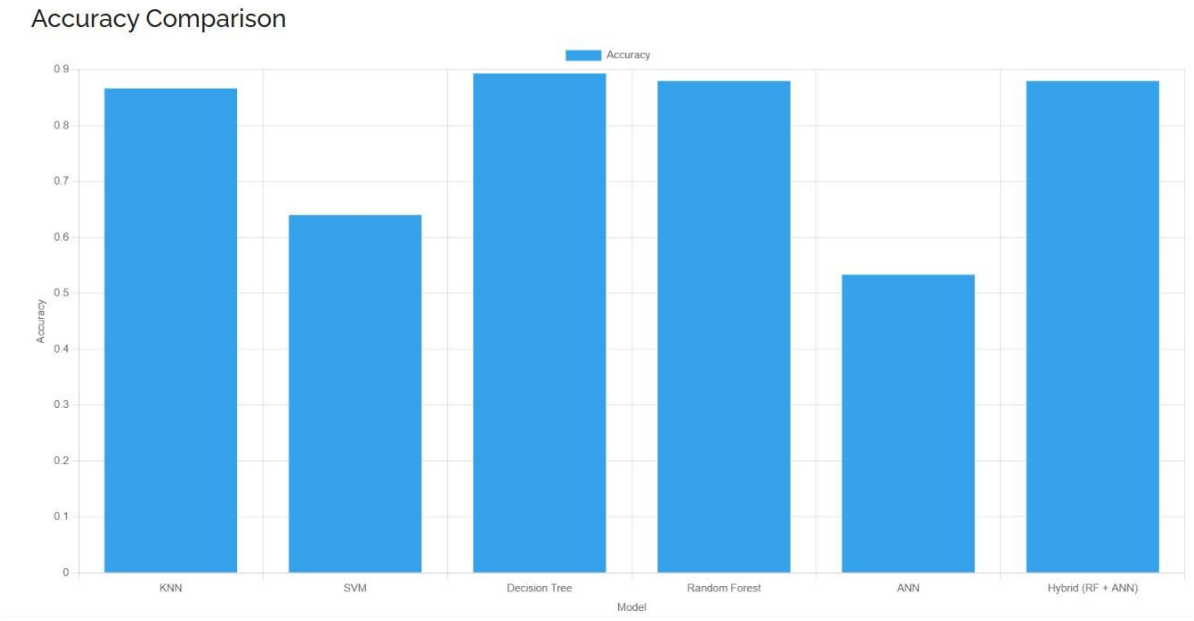

Fig. 8.4 Accuracy Comparision

Description: This interface demonstrates the system classifying a user's sleep condition based on the entered health and lifestyle parameters. By leveraging Machine Learning techniques, the model is able to analyze multiple factors together and predict possible sleep disorder categories that may not be easily identified through individual attributes alone, as shown in Fig. 8.4.

Fig. 8.5 Disorder prediction Analysis

Description: This module analyzes the health and lifestyle details entered by the user to predict the possibility of a sleep disorder. The trained Machine Learning model evaluates factors such as sleep duration, quality of sleep, stress level, BMI category, blood pressure, heart rate, and daily steps, and classifies the result as Insomnia, Sleep Apnea, or No Disorder, as shown in Fig. 8.5.

9. CONCLUSION

The present project introduces an intelligent and healthcare-oriented system for the classification of sleep disorders using Machine Learning algorithms. The main objective of the system is to analyze user health and lifestyle-related parameters and predict possible sleep disorder conditions such as Insomnia, Sleep Apnea, or No Disorder. By applying Machine Learning techniques, the project demonstrates how computational models can support early identification of sleep-related health issues and assist users in understanding their sleep condition more effectively.

During the implementation phase, the sleep health dataset was processed and prepared for model training. The dataset includes important attributes such as gender, age, occupation, sleep duration, quality of sleep, physical activity level, stress level, BMI category, blood pressure, heart rate, and daily steps. These features were used to train the system so that it could learn meaningful patterns associated with different sleep disorder categories.

Various Machine Learning algorithms were applied and evaluated, including K-Nearest Neighbors, Support Vector Machine, Decision Tree, Random Forest, Artificial Neural Network, and a Hybrid model. Each algorithm was tested to understand its performance in classifying sleep disorders. The use of multiple models helped compare their accuracy and effectiveness, while the hybrid approach improved prediction capability by combining the strengths of different learning techniques.

The developed system provides a user-friendly web interface where users can enter their health and lifestyle details and receive a predicted sleep disorder result. The system also includes an admin module for managing users, viewing prediction records, analyzing graphical reports, and evaluating model accuracy using datasets. This makes the application both practical and easy to use from a system management perspective.

From a software design point of view, the project follows a structured development approach, including requirement analysis, system design, database design, implementation, testing, and evaluation. The modular architecture of the system makes it maintainable and allows future improvements, such as adding more medical parameters, improving prediction accuracy, integrating larger datasets, or including doctor recommendation features.

Beyond its technical contribution, the project highlights the importance of using Machine Learning in the healthcare domain. Sleep disorders can affect physical health, mental well-being, productivity, and overall quality of life. An automated prediction system can help create awareness and encourage users to take timely medical advice when symptoms indicate possible sleep-related problems.

In conclusion, the project successfully demonstrates that Machine Learning algorithms can be effectively applied for the classification of sleep disorders. The system achieves its intended objective by analyzing user inputs, predicting sleep disorder categories, and presenting the results through an interactive web application. It provides a strong foundation for further research and future enhancement in intelligent healthcare prediction systems.

10. FUTURE ENHANCEMENTS

Although the developed framework provides a strong and effective foundation for cyberbullying detection, several enhancements can be implemented to further improve its performance, scalability, adaptability, and real-world applicability. As cyberbullying continues to evolve in complexity across different platforms, continuous improvements in both modeling techniques and system architecture are essential to maintain high detection accuracy and robustness.

One of the primary directions for future enhancement is the integration of advanced deep learning models such as BERT (Bidirectional Encoder Representations from Transformers), RoBERTa, and LSTM-based architectures. These models are capable of capturing deeper contextual and semantic relationships within textual data, enabling more accurate identification of subtle, implicit, and context-dependent abusive expressions. Transformer-based architectures, in particular, can significantly improve the system's ability to understand sarcasm, slang, and evolving language patterns commonly observed in social media communication. A hybrid approach that combines these deep learning models with the existing ensemble framework can further enhance prediction performance and robustness.

Another important enhancement involves expanding the dataset to include multilingual and cross-domain content. Most current implementations are limited to English-language datasets, which restricts usability in diverse linguistic environments. Extending the system to support multiple languages using multilingual embeddings or transformer models would increase its applicability across different regions and user groups. Additionally, incorporating multimodal data such as images, videos, audio, and combined text-media inputs would allow the system to detect cyberbullying in more complex scenarios, such as memes and visual content, which are increasingly prevalent in online platforms.

The system can also be extended to support real-time monitoring and streaming capabilities. By integrating with live data sources such as social media APIs or message streams, the framework can continuously analyze incoming content and provide instant detection of harmful interactions. This would enable proactive moderation and immediate intervention, making the system suitable for deployment in real-world, large-scale environments such as social media platforms, online forums, and educational systems.

Furthermore, the existing Django-based RESTful backend can be enhanced to improve scalability and performance for production-level deployment. This includes optimizing API endpoints, implementing caching mechanisms, enabling pagination for large datasets, and introducing rate limiting to handle high volumes of requests efficiently. The incorporation of asynchronous processing techniques, such as background task queues, can further improve system responsiveness and reduce latency in real-time detection scenarios.

In addition, introducing adaptive learning mechanisms can significantly improve the system over time. By incorporating feedback loops where users or moderators validate predictions, the system can iteratively retrain and refine its models. This human-in-the-loop approach helps reduce misclassifications, improves generalization, and ensures that the system adapts to evolving patterns of cyberbullying behavior. Periodic retraining with updated datasets can also maintain long-term effectiveness.

From an analytical and usability perspective, the integration of interactive dashboards and visualization tools can enhance the practical utility of the system. These dashboards can provide insights such as frequency of cyberbullying incidents, category-wise distribution, severity analysis, and temporal trends. Such features would assist administrators, researchers, and policymakers in making informed decisions and developing effective prevention strategies.

Finally, strengthening privacy, security, and ethical considerations is essential for real-world deployment. Future improvements can include implementing data anonymization, secure data storage, and compliance with data protection standards. Incorporating fairness-aware machine learning techniques can help minimize bias in predictions, while explainable AI (XAI) methods can provide transparency into model decisions, thereby increasing user trust and system accountability.

Collectively, these enhancements will transform the current framework into a more intelligent, scalable, and user-centered solution. By integrating advanced modeling techniques, expanding data diversity, enabling real-time capabilities, optimizing backend performance, and ensuring ethical deployment, the system can evolve into a comprehensive platform for effective cyberbullying prevention and safer digital communication.

REFERENCES

1. A. Ramesh Babu, P. B. Pandurangaraju, D. Deepika, K. Hemanth Kumar, S. M. Mahammad Mubarak, and A. C. Guru Balaji, "Machine Learning Framework for Early Detection and Prediction of Sleep Disorders Using Multimodal Health Data," in Proc. 4th Int. Conf. Intelligent Data Communication Technologies and Internet of Things (IDCIoT), 2026.
2. A. A. Alhussan et al., "Enhancing Sleep Disorder Classification Using Machine Learning and Metaheuristic Optimization Techniques," Biomed. Signal Process. Control, 2025.
3. M. Mostafa Monowar et al., "Advanced Sleep Disorder Detection Using Multi-Layered Ensemble Learning and Data Balancing Techniques," Front. Artif. Intell., 2025.
4. J. Osa-Sanchez et al., "Artificial Intelligence and Wearable Sensors for Sleep Apnea Detection," IEEE Rev. Biomed. Eng., 2025.
5. H. Zhang et al., "SleepLiteCNN: Lightweight Deep Learning Model for Sleep Apnea Detection Using ECG Signals," IEEE Trans., 2025.
6. S. Verma et al., "Sleep Disorder Prediction Using Ensemble Machine Learning Techniques," in Proc. Springer LNNS, 2025.
7. R. K. Sharma et al., "Classification of Sleep Disorders Using Random Forest and Support Vector Machines," in Proc. IEEE Conf., 2025.
8. P. N. Rao et al., "AI-Based Sleep Health Monitoring and Disorder Detection Using Wearable Devices," IEEE Sensors J., 2025.
9. T. Alshammari, "Applying Machine Learning Algorithms for the Classification of Sleep Disorders," IEEE Access, 2024.
10. Y. Liu et al., "Deep Learning-Based Sleep Stage Classification Using EEG Signals: A Review," IEEE Rev. Biomed. Eng., 2024.
11. K. Singh et al., "Sleep Apnea Detection Using Machine Learning Models and Clinical Data," MDPI Healthcare, 2024.
12. M. Rahman et al., "Hybrid CNN-LSTM Model for Sleep Disorder Detection Using Physiological Signals," Biomed. Signal Process. Control, 2024.
13. F. Li et al., "A Deep Learning Model for Sleep Apnea Detection Using EEG Signals," Biomed. Signal Process. Control, 2023.
14. A. Khan et al., "Automated Sleep Apnea Detection Using Machine Learning Techniques," in Proc. IEEE ICCCNT, 2023.
15. S. Arya et al., "Detection of Insomnia Using Machine Learning Classifiers," in Proc. IEEE Conf., 2023.

16. H. Phan et al., "SeqSleepNet: End-to-End Hierarchical Recurrent Neural Network for Sleep Stage Classification," IEEE Trans. Neural Syst., 2023.
17. D. Zhuang et al., "Automatic Classification of Multiple Sleep Disorders Using Deep Learning," 2022.
18. A. Supratak et al., "DeepSleepNet: A Model for Automatic Sleep Stage Scoring Based on Raw EEG Data," IEEE Trans. Neural Syst., 2022.
19. S. Biswal et al., "SleepNet: Automated Sleep Stage Classification Using Deep Neural Networks," in Proc. IEEE EMBC, 2022.
20. M. Hassan et al., "Sleep Apnea Detection Using ECG Signals and Machine Learning Algorithms," IEEE Access, 2022.