Project report on

"Intelligent Sleep Tracking System for People Suffering from Alzheimer's"

A Dissertation submitted in partial fulfilment of the requirement for the award of degree

MASTER OF COMPUTER APPLICATIONS

OF

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

By

AFREEN TAJ

1BY23MC005

Under the Guidance of

Dr. M. Sridevi
Assistant Professor
Department of MCA
BMSIT&M

Department of Master of Computer Applications

BMS Institute of Technology & Management

Bengaluru – 560119

July-2025

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

Bengaluru – 560119

July-2025

CERTIFICATE

This is to certify that the dissertation titled **"Intelligent Sleep Tracking System for people suffering from Alzheimer's"** submitted in partial fulfilment of the requirements for the degree **"Master of Computer Applications"** by Visvesvaraya Technological University is based on an original study and is record of Bonafide work carried out by **AFREEN TAJ** bearing university registration number **1BY23MC005** during the period **April 2025 to July 2025** under our supervision and guidance and that no part of the report has been submitted for the award of any other Degree/ Diploma/ Fellowship or similar title or prizes. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Master of Computer Applications Degree.

Signature of Internal Guide	**Signature of the HoD**	**Signature of the Principal**
Dr. M.Sridevi	**Dr. M.Sridevi**	**Dr. SANJAY H A**
sridevim@bmsit.in	hod_mca@bmsit.in	principal@bmsit.in
Assistant Professor	Head of Department	Principal
Department of MCA	Department of MCA	BMSIT&M
BMSIT&M	BMSIT&M	Bengaluru-560119
Bengaluru-560119	Bengaluru-560119	

PRINCIPAL
BMS Institute of Technology and Management
Avalahalli, Yelahanka, Bengaluru-64

External Viva-Voce

Name of Examiners

1. ---------- **Signature**

2. ---------- **Signature**

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(An Autonomous Institution, Affiliated to VTU, Belagavi)

Department of Masters of Computer Applications

CERTIFICATE

This is to certify that **Ms. Afreen Taj,** bearing **1BY23MC005** has successfully completed the **Project Work (22MCA403)** titled **"Intelligent Sleep Tracking and Fall Detection System for people suffering from Alzheimer's"** at **R&D Centre, Department of MCA, BMS Institute of Technology & Management, Bengaluru** under the guidance of **Dr. M Sridevi, Assistant Professor & Head, Department of MCA** during the period from **April to July-2025**.

24/7/25

Signature of the Guide
Dr. M Sridevi
Assistant Professor & Head
Department of MCA
BMSIT&M
Bengaluru.

24/7/25

Signature of the R&D Centre Head
Dr. M Sridevi
Assistant Professor & Head
Department of MCA
BMSIT&M
Bengaluru.

Head of the Department
Dept. of Master of Computer Applications
BMS Institute of Technology & Management
Bangalore-560064

DECLARATION

I AFREEN TAJ, student of MCA, BMS Institute of Technology & Management, bearing USN- 1BY23MC005 hereby declare that project entitled "Intelligent Sleep Tracking System for people suffering from Alzheimer's" has been carried out by me under the supervision of **Dr M Sridevi,** Assistant Professor & HoD submitted in the partial fulfilment of the requirements for the award of Degree of Master of Computer Applications by the Visvesvaraya Technological University during the academic year 2024-25. This report has not been submitted to any other Organization/University for any award of degree or certificate.

Afreen Taj

Signature

Place: Bengaluru

Date: 01/08/25

Name: AFREEN TAJ

USN: 1BY23MC005

ACKNOWLEDGEMENT

The project would not have been complete without remarking and thanking people who guided me, helped me and encouraged me throughout the development of this project.

I would like to utilize this opportunity to express gratitude to each person who made it possible for me to complete my project successfully. Thus, I would like to remark few people, whom I want to thank and express sincere gratitude.

I convey my truthful gratitude to **BMSIT&M** Management for providing a good infrastructure and educational support in lighting our career.

I would like to show my sincere gratitude to our Principal, **Dr. Sanjay H A** for his kind support in completing this project.

I take this opportunity to thank our Head of Department and my internal guide, **Dr. M Sridevi**, for her support, encouragement, and valuable inputs throughout the completion of this project.

Last but not the least, I thank my parents, family and friends who stood with me as a moral support and encouraging me in accomplishing this project.

AFREEN TAJ

1BY23MC005

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

Bengaluru – 560119

Department of MCA

VISION

To emerge as a leading department in computer applications, producing skilled professionals equipped to deliver sustainable solutions.

MISSION

Facilitate effective learning environment through quality education, industry interaction with orientation towards research, critical thinking and entrepreneurial skills.

Programme Educational Objectives (PEOs)

PEO1: Excel in IT career by developing sustainable solutions that drive industry growth and societal progress

PEO2: Adapt themselves to evolving domain requirements.

PEO3: Exhibit leadership skills and progress in their chosen career path.

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

Bengaluru – 560119

Department of MCA

Programme Outcomes (POs)

PO1: Apply knowledge of mathematics, programming logic and coding fundamentals for solution architecture and problem solving.

PO2: Identify, review, formulate and analyse problems for primarily focusing on customer requirements using critical thinking frameworks.

PO3: Design, develop and investigate problems with an innovative approach for solutions incorporating ESG/SDG goals.

PO4: Select, adapt and apply modern computational tools such as development of algorithms with an understanding of the limitations including human biases.

PO5: Function and communicate effectively as an individual or a team leader in diverse and multidisciplinary groups using methodologies such as agile.

PO6: Use the principles of project management such as scheduling, work breakdown structure and be conversant with the principles of Finance for profitable project management.

PO7: Commit to professional ethics in managing software projects with financial aspects. Learn to use new technologies for cyber security and insulate customers from malware.

PO8: Change management skills and the ability to learn, keep up with contemporary technologies and ways of working.

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

Bengaluru – 560119

Department of MCA

Course Outcomes (COs)

CO1: Review the existing literature to identify and formulate the problem in contemporary technologies/ issues related to society/environment which leads to development of IT solution.

CO2: Analyse the requirements and prepare Software requirement specifications (SRS) document as per IEEE format in consistency with the problem defined.

CO3: Create models that are consistent with the requirements specified in the SRS.

CO4: Develop the solution by applying appropriate techniques, software engineering and management principles and modern tools to meet the requirements either as an individual or by involving in team.

CO5: Verify & validate the data and results to arrive at valid conclusions and communicate the work done effectively in terms of presentations, writing reports and research article as per the format given.

CO6: Follow ethical principles in all stages of project work by avoiding plagiarism.

CO7: Articulate the impact of IT solutions developed in the project work with respect to societal, environmental and industrial issues at large.

ABSTRACT

The **Intelligent Sleep Tracking System for people suffering from Alzheimer's** is a real-time health monitoring platform designed to track and analyze sleep posture and vital signs using embedded sensors and machine learning. Targeted especially at individuals with cognitive or mobility impairments, the system enables continuous monitoring of parameters such as heart rate, SpO_2, body temperature, and sleep position through the integration of low-cost sensors including the MAX30102, LM35, and MPU6050.

An ESP32 microcontroller collects and transmits this physiological data to a cloud backend managed via Supabase. A web-based dashboard built with React and Tailwind CSS presents real-time and historical trends, while allowing users to generate PDF and CSV reports for medical use. The platform also includes a predictive analytics feature, where uploaded patient data is evaluated by a machine learning model hosted via a Flask API. This model classifies patient dependability levels and provides associated probability scores, aiding early intervention.

The system supports posture trend visualizations, inactivity alerts, and doctor-patient report sharing. Testing has shown that the hardware-software integration is stable, efficient, and scalable for long-term use. With the potential for future expansion through wearable integration, contactless sensors, and EHR connectivity, the platform offers a promising foundation for remote, AI-enhanced patient care.

Keywords: IoT healthcare, ESP32, machine learning, posture detection, real-time vitals, SpO_2 monitoring, Supabase, predictive analytics, health tracking.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Project Overview

The Intelligent Sleep Tracking System for people suffering from Alzheimer's is a comprehensive digital platform designed to monitor, analyze, and report the sleep behaviour and vital statistics of patients in a non-intrusive, continuous, and scalable manner. Built as a responsive web application, the system enables real-time tracking of parameters such as body temperature, heart rate, SpO2, and sleep posture. Through a modern user interface and cloud-based infrastructure, it supports the remote supervision of cognitively impaired individuals, particularly those suffering from Alzheimer's disease, chronic conditions, or mobility limitations.

By integrating sensor-based data acquisition, cloud storage, machine learning analysis, and automated report generation, the system enables both healthcare professionals and caregivers to gain actionable insights into patient sleep health. Reports can be generated for specific time ranges and shared via email, supporting telemedicine workflows and reducing the need for in-person consultations. The modular design allows for the easy incorporation of future technologies, including wearable devices and intelligent alerts, making the platform a future-proof solution in the domain of smart healthcare.

1.2 Background

In Sleep monitoring plays an increasingly vital role in both preventive and therapeutic healthcare, particularly for populations with chronic illnesses or cognitive impairments. Traditional methods such as in-lab sleep studies, while accurate, are costly, inconvenient, and offer only a snapshot of an individual's sleep behaviour. In contrast, the proliferation of Internet of Things (IoT) devices, cloud-based storage, and machine learning algorithms has enabled the development of smart systems that can continuously and non-invasively monitor sleep at home. This project leverages such modern technologies to build a sleep tracking system tailored to patients with limited mobility and neurodegenerative conditions. Sensors like the MPU6050 (for motion and posture detection), the LM35 (for temperature), and pulse oximeters are integrated into a compact hardware setup. These sensors communicate with a microcontroller (ESP32), which transmits data to a backend hosted on Supabase. A React-based frontend visualizes both real-time and historical data, while a prediction engine provides machine learning-driven health forecasts.

This system not only provides visibility into sleep posture trends and vital fluctuations but also identifies risk patterns like prolonged immobility or abnormal postures. The inclusion of AI-generated insights further enhances the decision-making process for clinicians, allowing them to adjust care plans proactively. Its web-based nature ensures that caregivers and health professionals can access and interact with the data from any location, improving continuity of care and patient outcomes.

1.3 Problem Statement

Even Quality sleep is fundamental to human health, yet many individuals, especially the elderly and those with neurological or physical impairments, experience disturbed sleep patterns that go undetected. Traditional sleep monitoring methods such as polysomnography are resource-intensive, limited to clinical settings, and often unsuitable for continuous or long-term use. Additionally, current mobile-based applications typically offer generic insights and lack integration with real-time sensors or predictive analytics.

There is a critical need for a low-cost, non-intrusive, scalable system that not only records physiological and positional data during sleep but also provides timely analysis, alerts, and recommendations tailored to individual patient profiles. Such a system should be accessible to both patients and doctors via remote interfaces, especially to aid in the care of patients suffering from cognitive conditions like Alzheimer's, where immobility or poor sleep posture can lead to further health complications.

1.4 Objectives of the Project

The primary of the Intelligent Sleep Tracking System are as follows:

• To develop a web-based application that monitors real-time patient vitals and sleep posture using embedded sensors.

• To provide a visual dashboard for caregivers and medical professionals to track patient health trends over time.

• To enable the generation of detailed sleep and health reports that include medical history, posture distribution, and vital sign summaries.

• To implement predictive analytics for early detection of health anomalies using machine learning models.

• To facilitate remote healthcare collaboration by enabling secure report sharing and email communication between patients and doctors.

2. LITERATURE SURVEY

The success of any technology-driven solution depends on a solid foundation of research, technical feasibility, and contextual understanding. This chapter presents an in-depth review of the existing systems that address carbon footprint estimation, along with an overview of related academic contributions and theoretical frameworks. It also introduces the conceptual basis of the proposed system and elaborates on its technical and operational viability through a structured feasibility study. Furthermore, it outlines the tools, technologies, and resources used in the development process, providing clarity on both software and hardware requirements.

2.1 Existing Systems

Sparse Sensor-Based Spatiotemporal CNN for Sleep Posture Detection

This system uses a sparse chest-mounted vibration sensor array and a spatiotemporal convolutional neural network (S³CNN) to detect sleep postures. It offers high accuracy (~93%) and leverages IoT for embedded communications. It validates the use of minimal sensors for posture detection, which aligns with the sensor-efficient design of our system

Smart Mattress with Triaxial Accelerometers

A novel design places eight accelerometers at key anatomical points on a mattress. The system achieves 99.9% classification accuracy across six postures using a Parallel Convolutional Spatiotemporal Network. While highly accurate, the setup is hardware-intensive. Our system offers a more affordable alternative using MPU6050 sensors with sufficient posture detection performance.

CNN-Based Smart Sleep Posture Recognition System (2022)

This IoT-enabled framework focuses on elderly care by combining wearable sensors with deep learning models to identify risky sleep behaviour. It supports dashboard access and cloud-based analytics, similar to our platform's UI and backend structure.

Raspberry Pi and AIoT-Based Monitoring

This research utilizes RFID tags and a Raspberry Pi backend to perform real-time posture monitoring. It illustrates how low-cost hardware and AIoT pipelines can be integrated for scalable deployments—conceptually similar to our use of ESP32 and cloud services

mmWave Radar with Deep Learning

An entirely contactless system using mmWave radar and a ResTCN model for detecting sleep postures. Though innovative, radar systems are costlier and harder to deploy in home settings. Our wearable sensor-based system offers a more accessible approach

2.2 Proposed System

The proposed system, titled Intelligent Sleep Tracking and Fall Detection System for people suffering from Alzheimer's, is a web-based platform designed to monitor sleep posture and vital signs of patients in real time. It addresses the limitations of traditional sleep labs and generic mobile applications by offering a low-cost, extensible, and patient-specific solution. The system integrates hardware sensors, a microcontroller, a real-time database, a React-based frontend, and a machine learning-powered prediction API to provide end-to-end health monitoring and reporting.

2.2.1 System Components

Sensor-Based Data Acquisition

- Sensors (MPU6050, LM35, MAX30102) collect posture, temperature, heart rate, and SpO2 data, sent via ESP32 over Wi-Fi.

ESP32 Microcontroller

- Acts as the data processor and transmitter, sending sensor data to the cloud using HTTP or WebSocket.

Supabase Backend

- Stores vitals, manages users, handles file uploads, and supports real-time updates.

Web Dashboard

- Built with React and Tailwind, it shows real-time vitals, posture trends, and report tools with PDF/CSV export and file uploads

Machine Learning API

- A Flask API analyzes uploaded health data using trained models to predict risks or health issues.

Report Generation

- Uses jsPDF to create downloadable reports with patient info, vitals, posture, ML insights, and care tips.

Email & Sharing Module

- Sends reports securely with optional remarks, with future plans for WhatsApp/SMS support.

2.3 Feasibility Study

A feasibility study was conducted to ensure that the proposed system is practical, sustainable, and scalable. The study considers technical, operational, and economic aspects.

2.3.1 Technical Feasibility

The core hardware components—ESP32, MPU6050, and LM35—are readily available, low-power, and compatible with standard microcontroller platforms. The software stack, including React, Supabase, and external APIs, is mature, well-documented, and widely supported. Data transmission is handled using built-in Wi-Fi modules, and backend integration is achieved using secure APIs. Given the modular architecture, each component can be developed, tested, and integrated independently, making the project highly feasible from a technical standpoint.

2.3.2 Operational Feasibility

From an operational perspective, the system is designed for ease of use by both medical professionals and caregivers. The interface requires minimal technical knowledge, and sensor installation is non-invasive. Report generation and sharing features are automated, reducing the dependency on manual monitoring or specialized staff. The platform can be operated remotely, supporting both in-home care and clinical follow-up, which enhances its long-term operational viability.

2.3.3 Economic Feasibility

Economically, the system uses cost-effective components. The ESP32 board, sensors, and basic wiring cost significantly less than conventional sleep lab equipment. Development tools like VS Code and Arduino IDE are open-source, and hosting options like Supabase offer generous free tiers. This makes the total cost of implementation affordable for institutions, clinics, or even individual households. Moreover, the reusable nature of the hardware and scalable backend minimize long-term maintenance costs.

2.4 Tools and Technologies Used

2.4.1 Frontend Technologies

- **Next.js 15 (React 18)**

  Used to build the frontend web application. React facilitates component-based UI development, while Next.js supports server-side rendering and route management.

- **Tailwind CSS & shadcn/ui**

  Utility-first Tailwind CSS accelerates styling with consistent spacing, responsive breakpoints, and configurable themes. shadcn/ui provides prebuilt, accessible component primitives such as cards, buttons, and form controls integrated seamlessly with Tailwinds conventions.

- **VS Code**

Visual Studio Code (VS Code) is used as the main code editor for both frontend and backend development. It supports extensions for JavaScript, TypeScript, Python, and Git integration, making it an efficient environment for full-stack development.

2.4.2 Backend Technologies

- **Python**

Python is primarily used for data preprocessing and model development in the machine learning prediction module. It supports numerous libraries for numerical analysis, such as NumPy and pandas, and for model training, such as scikit-learn and TensorFlow. The custom Flask API used to deploy the ML model is also written in Python.

- **Arduino IDE**

The Arduino IDE is used to program the ESP32 microcontroller. It provides an intuitive interface to write, compile, and upload code that reads sensor data and transmits it over Wi-Fi to the Supabase backend.

2.4.3 External APIs and SDKs

- **Flask API**

The Flask API is used to deploy and run machine learning models that analyze uploaded medical data like vitals or reports. It processes files (CSV, JSON, etc.), loads trained models (e.g., XGBoost, Random Forest), and returns predictions. It also integrates with the database to fetch or store related data for each prediction task.

- **Supabase Client SDK**

Facilitates secure, real-time interactions with the PostgreSQL database for product retrieval, analytics logging, and feedback submission. Built-in authentication manages session tracking without custom token handlers.

2.4.4 Database and Storage Services

- **Supabase (PostgreSQL)**

A managed PostgreSQL instance under the academic free tier stores:

  - vitals: real-time sensor data such as temperature, heart rate, SpO2, and posture readings
  - patients: basic patient metadata including age, gender, and ID
  - alerts: ML-based dependability predictions and anomaly flags keywords: file-derived keyword lists for category detection
  - sessions: timestamped logs of monitoring sessions for each patient

2.5 Hardware and Software Requirements

2.5.1 Hardware Requirements

Table 2.1 Hardware Requirements

Component	Specification / Purpose
ESP32 Microcontroller	Collects data from sensors and transmits it via Wi-Fi to the backend
MPU6050 Sensor	Measures motion and orientation for detecting sleep posture
LM35 Temperature Sensor	Captures the body temperature of the user
MAX30102 / Pulse Sensor	Measures heart rate and oxygen saturation (SpO_2)
Breadboard and Jumper Wires	Used for sensor prototyping and microcontroller connections
USB Power Supply	Provides consistent power supply to the ESP32 and sensors
Protective Enclosure	Shields hardware from physical damage and user contact
Development Computer	Used for firmware programming, testing, and system debugging (Minimum: 8GB RAM, SSD)

2.5.2 Software Requirements

Table 2.2 Software Requirements

Category	Requirement / Tool
Operating System	Windows 10/11 or macOS (for development)
Frontend Framework	React with Next.js: Component-based UI with routing and server-side rendering
Styling Framework	Tailwind CSS: Utility-first framework for responsive and modern UI
Backend Platform	Supabase: Real-time database, authentication, and file storage backend
Microcontroller IDE	Arduino IDE: For coding and uploading firmware to ESP32
Code Editor	Visual Studio Code (VS Code): Full-stack development with extensions for TypeScript, JS
Machine Learning API	Python with Flask: Used to develop and serve the ML prediction model via API
Report Generation Library	jsPDF and AutoTable: Dynamic generation of health reports in PDF and CSV formats
Charting Library	Chart.js: Visual representation of trends in vitals and posture data
Node.js Runtime	JavaScript backend development and package management
Version Control	Git: Source control and collaboration tool

3. SOFTWARE REQUIREMENTS SPECIFICATION

Clear and structured requirements are essential to ensure the successful implementation of any software system. This chapter outlines the functional and non-functional requirements of the proposed system, along with key use case scenarios, assumptions, and constraints. These specifications form the basis for development ensuring that it aligns with both user needs and system objectives.

3.1 Scope and Objective

The primary objective of the Intelligent Sleep Tracking System for people suffering from Alzheimer's is to build a user-friendly, real-time sleep and health monitoring solution that:

- Collects and visualizes vital signs and posture data.
- Enables users to generate and share structured health reports.
- Integrates machine learning for predictive analytics.
- Supports remote accessibility for doctors and caregivers.

Scope includes:

- The Real-time dashboard interface for data visualization.
- Cloud-based backend for historical data storage and authentication.
- Hardware integration for live data transmission from sensors.
- Automatic and manual report generation with sharing features.
- File upload support for AI-based predictions using external APIs.

3.2 Assumptions and Limitations

Assumptions

- These are the initial conditions assumed to be true for the successful implementation and operation of the Intelligent Sleep Tracking and Fall Detection System for people suffering from Alzheimer's:
- These body vitals (temperature, heart rate, SpO_2, and sleep posture) will play a significant role in detecting early signs of cognitive degeneration.
- Users will have access to internet-enabled devices such as personal computers, tablets, or smartphones to use the system.
- The ESP32 microcontroller and sensors are properly configured and connected.
- The machine learning model hosted through a Flask API remains operational and up to date.

- Supabase services (database, authentication, and storage) will function without disruption.
- Email sending services are authenticated and valid, enabling successful delivery of reports.
- Users and doctors will adopt digital tools for report review and communication.

Limitations

- The system has currently been tested only with hypothetical and simulated data.
- Real-world accuracy and system reliability can only be validated after testing on actual patients from the target audience.
- Sensor readings may vary depending on user movement, sensor placement, and environmental noise.
- The prediction API is dependent on external infrastructure and may require optimization under high load.
- Internet connectivity is essential for full system functionality, which may limit usage in low-connectivity areas.
- Future regulatory compliance (e.g., HIPAA or GDPR) may require additional development efforts depending on deployment geography.

3.3 Functional Requirements

The Functional requirements define what the software should do, specifying the expected behavior and interactions with users or other systems. Functional requirements serve as the foundation for software design, development, and testing, ensuring that the software's functionality aligns with the project's objectives and user expectations. They represent a clear and precise description of the specific functions, features, and capabilities that a software system or application must possess to meet its intended purpose and satisfy the needs of its users or stakeholders.

Table 3.1 Functional Requirements

ID	Requirement Description	Priority
FR1	The system shall allow users to log in securely and access personalized dashboards.	High
FR2	The system shall collect and display real-time data from connected sensors.	High
FR3	The dashboard shall provide visualizations for temperature, SpO_2, heart rate, and posture.	High

FR4	Users shall be able to select a date range and generate health reports.	High
FR5	Reports shall include vital trends, posture distribution, and predictive analysis.	High
FR6	The system shall allow users to upload files for AI-based prediction.	Medium
FR7	Users shall be able to send generated reports via email to doctors.	High
FR8	The system shall store historical data in Supabase for future access.	High

3.4 Non-Functional Requirements

Non-functional requirements, also known as quality attributes or system attributes, are essential characteristics that describe how a software system should perform rather than what it should do. These requirements define the qualities, constraints, and attributes that influence the overall performance, usability, and reliability of the software.

Table 3.2 Non-Functional Requirements

ID	Requirement Description	Priority
NFR1	The system must have a consistent UI across all supported browsers and devices.	High
NFR2	Real-time data updates should reflect within 2 seconds of sensor transmission.	High
NFR3	Data transmission between ESP32 and the database must be encrypted.	High
NFR4	The system must remain operational for 24/7 continuous monitoring.	High
NFR5	The platform should support a minimum of 100 concurrent users.	Medium
NFR6	The dashboard UI should load within 3 seconds for a standard connection.	Medium
NFR7	The system must comply with data privacy regulations (e.g., HIPAA or GDPR depending on use).	High
NFR8	Report generation should complete within 5 seconds for a 7-day range.	Medium
NFR9	The prediction API should respond within 2 seconds for standard file sizes (<1 MB).	Medium

4. SYSTEM DESIGN

Diagrams below illustrate the interactions and data transformations across the system.

4.1 System Architecture

Components:

- Client (Next.js): Frontend (React.js Dashboard): Provides the user interface for caregivers to upload PDFs and visualize predictions and sensor data.
- Backend (Flask API): Acts as the core processing unit that receives the uploaded PDFs, extracts relevant features, and passes them to a machine learning (ML) classifier.
- Machine Learning Classifier: Uses extracted data to predict patient health metrics and their dependability.
- IoT Devices (ESP32 + Sensors): Collect real-time physiological data such as heart rate, temperature, SpO2, and posture, which are sent to the cloud.
- Cloud Storage (Supabase DB): Stores both historical and live sensor data for analysis and display. This data can be retrieved by the dashboard for visualization or used by the ML model for context-aware predictions.

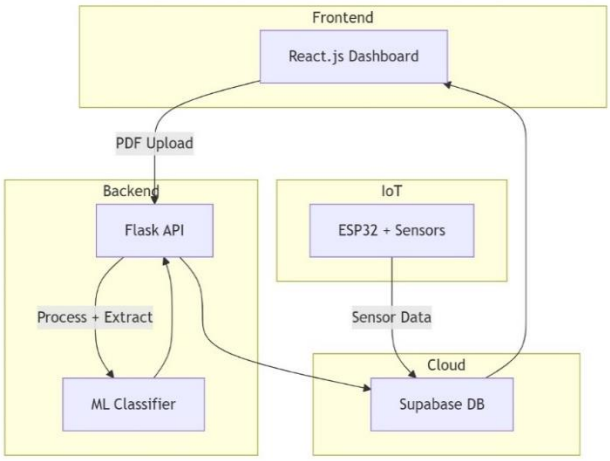


Fig. 4.1 System Architecture

4.2 System Perspective

The Intelligent Sleep Tracking System for people suffering from Alzheimer's is built as a client-server model with tightly integrated hardware, software, and API services. From a user's point of view, the system functions like a single seamless interface that tracks health in real time and provides meaningful insights.

• Hardware Perspective: The ESP32 acts as an embedded controller, abstracting raw sensor data and streaming it to the cloud with minimal latency.

• Software Perspective: The frontend and backend communicate via REST APIs and WebSockets. JavaScript (React) powers the UI while Python supports machine learning predictions.

• End-User Perspective: The user experiences a web app that is device-independent, interactive, and informative, providing both real-time and historical insights.

4.3 Context Diagram

A context diagram is a simplified visual representation that shows how a system interacts with external entities or actors, helping to define the system's boundaries and clarify its inputs and outputs.

- External Entities: Include the caregiver (user), ESP32 device (sensor data source), and Supabase (cloud database).
- Health Monitoring System: Consists of the React dashboard (UI layer), Flask API (backend processor), and ML model (prediction engine).
- The flow indicates how external inputs (e.g., data from ESP32 and caregiver uploads) enter the system and are processed internally before outputting predictions back to the caregiver via the dashboard.

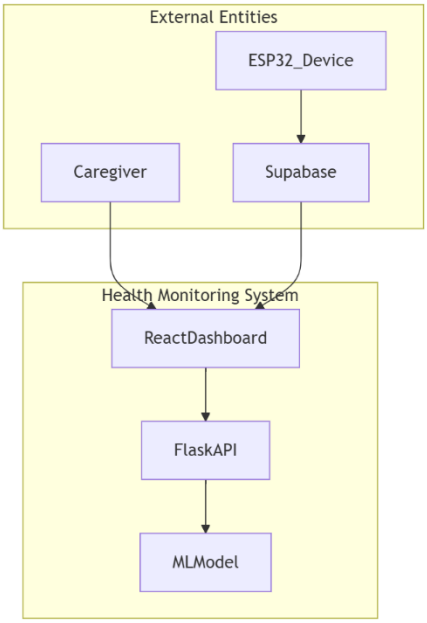


Fig. 4.2. Context Diagram

5. DETAILED DESIGN

5.1 Use Case Diagram

Key actors and use cases:

- User: Uploading PDFs: Caregivers can upload medical PDF reports for analysis.
- Dashboard Interaction: Includes viewing health metrics, receiving predictions, and monitoring historical trends.
- Backend Processing: After upload, the system extracts health parameters, triggers prediction using an ML model, and stores results.
- IoT Integration: Shows real-time data updates via ESP32 sensors, which feed into the central database and influence model predictions

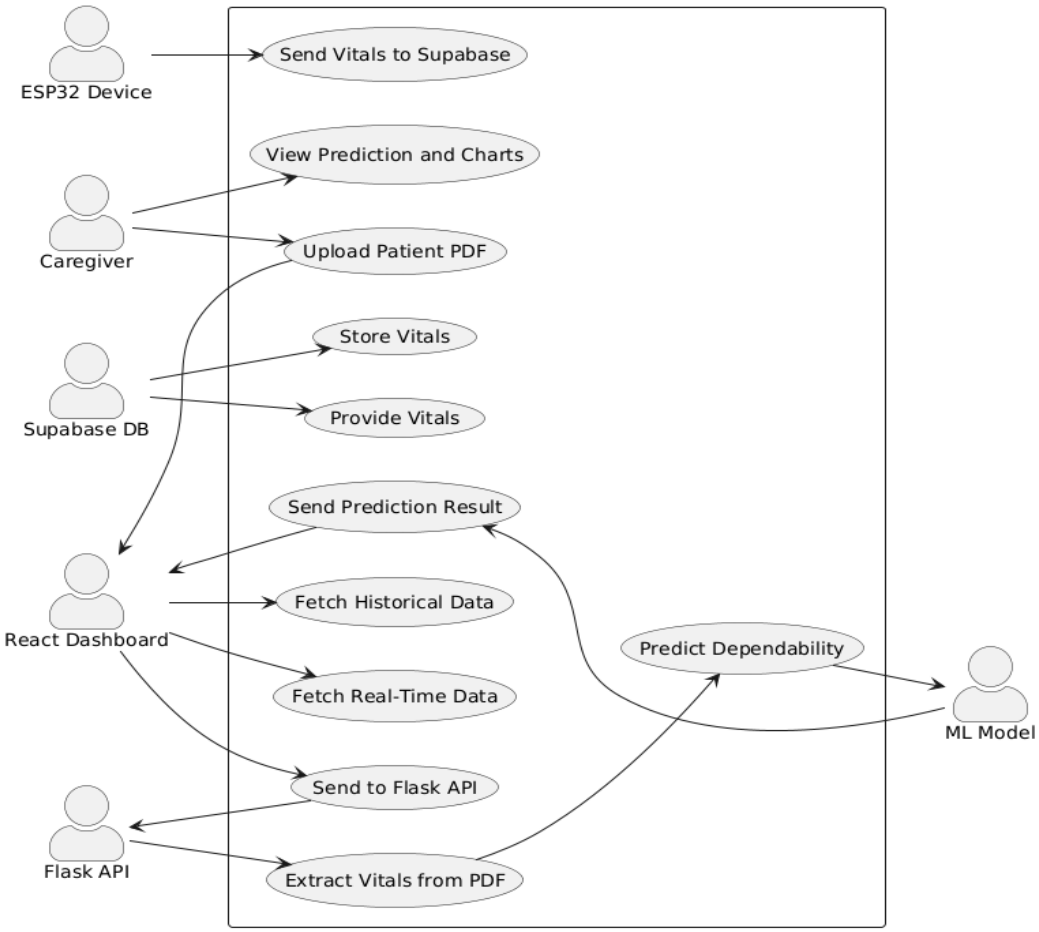


Fig. 5.1 Use Case Diagram

5.2 Sequence Diagram

- User → Caregiver uploads a PDF via the dashboard.
- The React dashboard sends a POST /predict request with the file to the Flask API.
- The Flask API extracts information from the file and sends it to the ML model for dependability prediction.
- Meanwhile, the API also fetches real-time and historical vitals from Supabase, which is continuously updated by the IoT device every 5 seconds (live data) and every 5 minutes (historical data).
- The prediction and related vitals are returned and displayed to the caregiver.

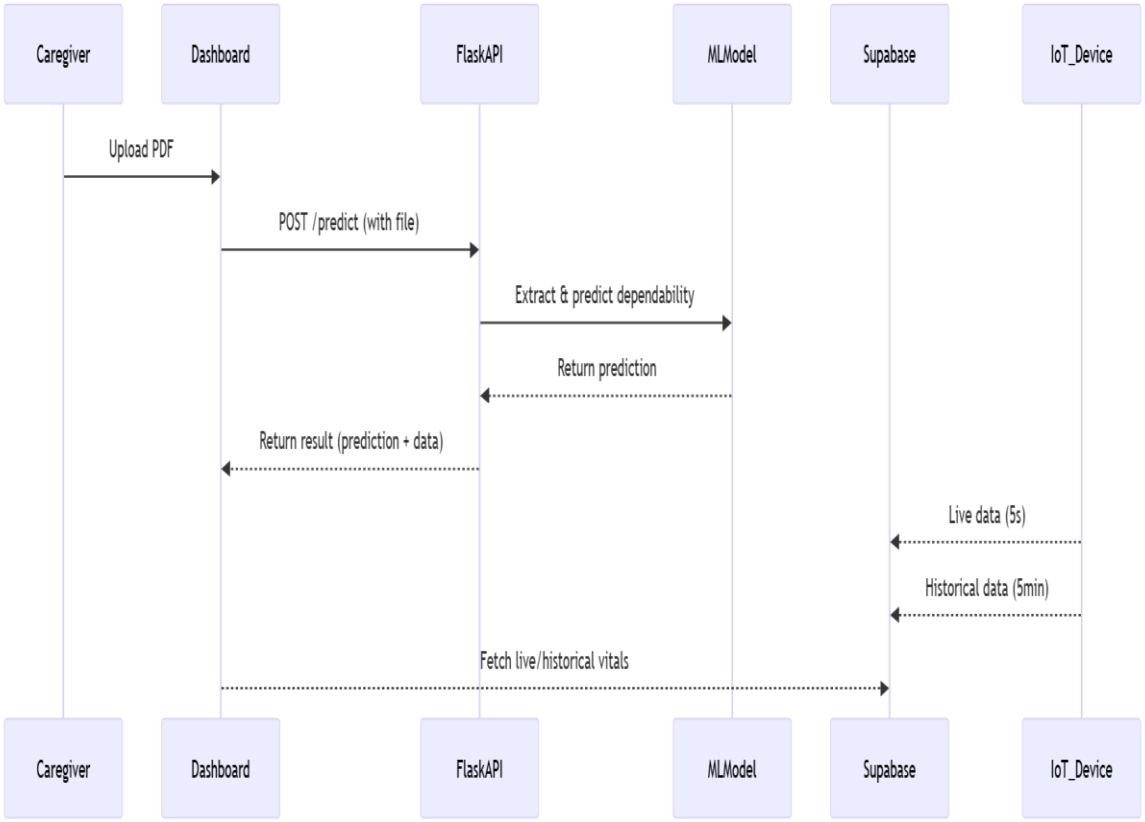


Fig. 5.2 Sequence Diagram

5.3 Activity Diagram

- Caregiver uploads a patient's medical report (PDF) via the dashboard, which is sent to the Flask API for processing.
- The Flask API extracts vital parameters from the PDF and passes the data to an ML model for dependability prediction.
- The prediction result is returned and displayed on the dashboard; alongside live and historical vitals fetched from the Supabase database.
- Simultaneously, the ESP32 device sends live sensor data to the Supabase cloud database every few seconds.
- ESP32 also periodically batches and sends historical data to Supabase for long-term analysis.
- The process ends with a complete display of predictions and visualized data on the dashboard.

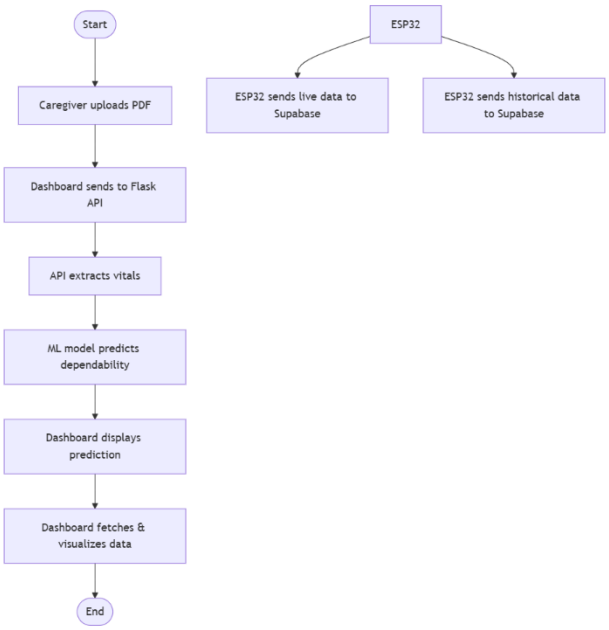


Fig 5.3 Activity Diagram

5.4 Data Flow Diagrams

- Data flows ESP32 sensors collect health parameters (HR, SpO2, Temp, Posture) and send them to Supabase.
- Caregivers interact with the React Dashboard to upload PDF reports.
- Upon upload, the dashboard calls the Flask API, which processes the file, extracts features, and sends them to the ML model.
- The ML model returns predictions and dependability scores, which are stored and displayed.

- Supabase stores both raw sensor data and derived predictions, enabling data-driven insights and historical analysis.

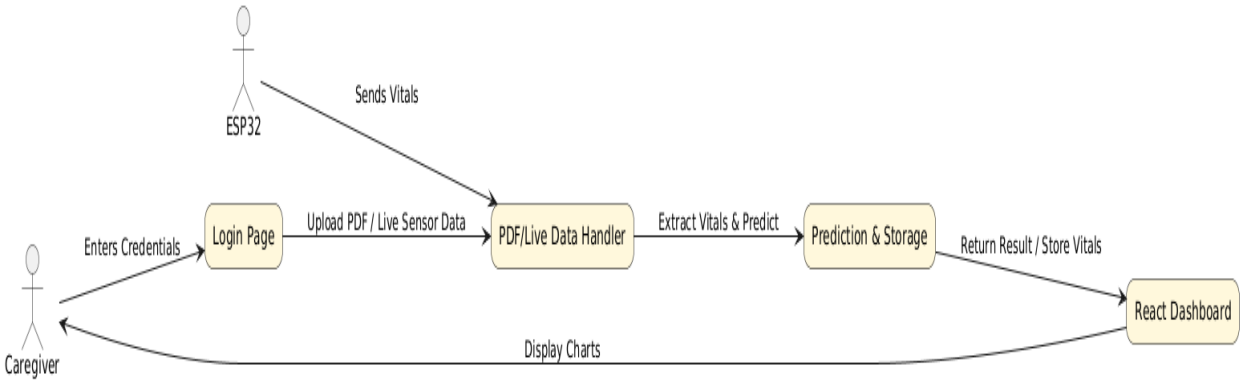


Fig. 5.4 Data Flow – Level 0

The caregiver logs in through the login page and can upload a PDF report or live sensor data. This data, along with vitals sent by the ESP32 device, is processed by the PDF/Live Data Handler. The handler extracts the necessary information and forwards it to the prediction and storage module, which predicts dependability and stores the vitals. Finally, the results and visualizations are displayed back to the caregiver via the React Dashboard

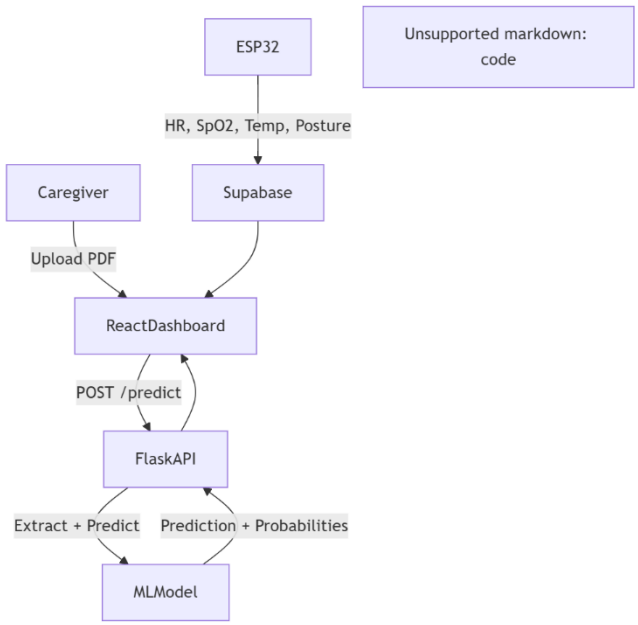


Fig. 5.5 Data Flow – Level 1

This diagram provides a detailed breakdown of the system processes. The ESP32 device continuously sends health data (HR, SpO2, temperature, and posture) to the Supabase database. Caregivers can also upload PDF reports via the React Dashboard, which sends

data to the Flask API for processing. The Flask API extracts the vitals and interacts with the ML model to perform predictions, returning both prediction results and probabilities. The React Dashboard retrieves predictions and live data from Supabase, displaying a complete view of the patient's health to the caregiver.

5.5 Entity Relationship Diagram (ERD)

Major entities and relationships:

- Caregiver: The primary user who uploads PDF health reports and views patient data.
- UPLOAD: Represents uploaded PDF files, capturing both the file path and upload timestamp.
- PREDICTION: Triggered by a PDF upload, it stores predicted vitals and a qualitative "dependability" score output by the ML model.
- HISTORICAL_DATA: Automatically populated by the ESP32 IoT sensors, this table stores actual patient health data recorded over time, such as heart rate, SpO2, temperature, posture, and timestamp.
- The relationships between entities reflect real-world interactions: a caregiver uploads files, views historical records, and triggers predictions. Relationships:

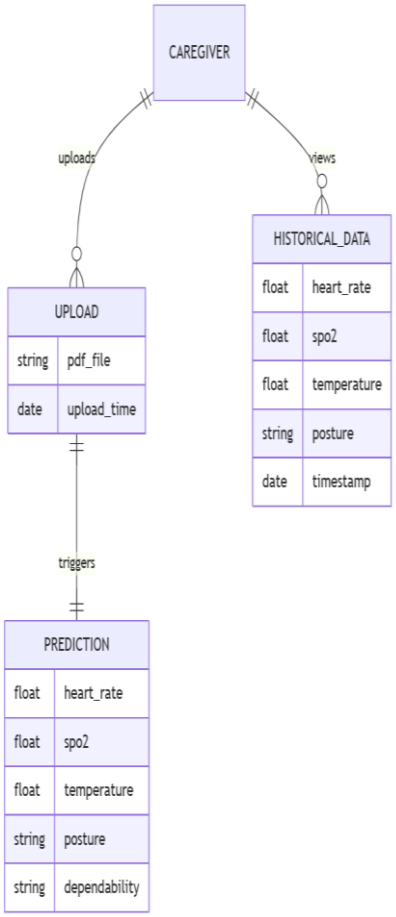


Fig. 5.6 ER Diagram

6. IMPLEMENTATION

6.1 Coding Standard

To ensure the codebase for the Intelligent Sleep Tracking System for people suffering from Alzheimer's remains clean, scalable, and maintainable, the following coding standards and best practices were applied across the three major components: embedded firmware, backend API, and frontend dashboard.

Embedded C++ (ESP32 / Arduino)

- Use descriptive variable names and maintain consistent indentation (2 or 4 spaces).
- Group configuration constants (#define or const) at the top for clarity and reuse.
- Encapsulate sensor logic in modular functions such as sendLiveData() or checkWiFiStatus() to enhance readability.
- Minimize delay-based timing and use millis() for non-blocking time tracking.
- Avoid hardcoding credentials or keys in the main sketch. Use secrets.h where applicable.
- Use meaningful debug Serial.print() messages for traceability during testing.

Python (Flask Machine Learning API)

- Follow PEP 8 guidelines for consistent formatting, spacing, and imports.
- Separate model logic (e.g., prediction, data extraction) into class methods or utility functions.
- Use try-except blocks to catch and handle errors gracefully when reading files or parsing input.
- Use joblib for model and encoder serialization, and load them once at startup for performance.
- Define consistent API responses with appropriate HTTP status codes and application/json headers.
- Avoid exposing internal errors to the client; log exceptions to console or a log file instead.

React (Frontend Dashboard)

- Use TypeScript for component-level type safety and maintain strong typing for props and API responses.
- Follow strict folder naming conventions: lowercase for folders (/components, /pages) and PascalCase for files (VitalsCard.tsx).

- Maintain a centralized theme and design system using Tailwind CSS for spacing, typography, and color consistency.
- Organize API calls in separate service files to separate logic from presentation.
- Use React Hooks (useEffect, useState) for managing side effects and state within functional components.
- Apply role-based rendering and conditional routing to maintain access control (e.g., doctor vs. patient view).
- Structure Supabase queries cleanly using async/await and handle loading/error states gracefully.

General Best Practices

- Keep all environment variables and keys in .env files or configuration constants, not hardcoded in source.
- Perform unit testing for critical modules (e.g., PDF generation, prediction endpoint).
- Use consistent file structures across the frontend and backend to simplify onboarding.
- Apply version control rigorously with meaningful Git commit messages and feature branches.

By enforcing these standards across all layers of the system, the project remains organized, testable, and extensible for future enhancements or team-based development.

6.2 Snippet Code

6.2.1 Backend Arduino code

```
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <WiFi.h>
#include "MAX30105.h"
#include "heartRate.h"
#include <HTTPClient.h>
// Object Declarations
Adafruit_MPU6050 mpu;
MAX30105 particleSensor;
// WiFi Credentials
char ssid[] = "Afreen_Riyaz";
char pass[] = "Afreen789";
// LED GPIOs
const int wifiConnectedLED = 2;
```

```
const int wifiNotConnectedLED = 4;
// LM35 Analog Pin
const int LM35_PIN = 34;
// Motion Detection
const float movementThreshold = 0.5;
unsigned long lastMotionTime = 0;
const unsigned long noMotionThreshold = 300000; // 5 minutes
String positionStatus;
// Heart Rate Variables
const byte RATE_SIZE = 4;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;
float beatsPerMinute = 0;
int beatAvg = 0;
float lastBPM = 0;
int lastAvgBPM = 0;
float lastSpO2 = 0;
float lastTemp = 0;
String lastPosition = "";
bool lastFingerPresent = false;
// Supabase config
const char* SUPABASE_URL =
"https://porabnsxqjdgmnxwqzms.supabase.co/rest/v1";
const char* SUPABASE_ANON_KEY =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6I
nBvcmFibnN4cWpkZ21ueHdxem1zIiwicm9sZSI6ImFub24iLCJpYXQiOjE3NTI0ODM1MTMs
ImV4cCI6MjA2ODA1OTUxM30.08Z5UqEfSnL0-2U5kisTvZDsBEv5jATeKtBZnDu1Fvw";
// Timers
unsigned long lastLiveUpdate = 0;
unsigned long lastHistoryUpdate = 0;
const unsigned long LIVE_UPDATE_INTERVAL = 5000; //5 seconds
const unsigned long HISTORY_UPDATE_INTERVAL =300000; // 5 minutes
void sendLiveData(float heart_rate, float spo2, float temperature,
String position) {
  HTTPClient http;
  String url = String(SUPABASE_URL) + "/live_data?id=eq.1";
  http.begin(url);
  http.addHeader("apikey", SUPABASE_ANON_KEY);
  http.addHeader("Authorization", String("Bearer ") +
SUPABASE_ANON_KEY);
```

```
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Prefer", "return=representation");
  String positionToSend = position.length() > 0 ? position : "Unknown";
  String payload = "{";
  payload += "\"heart_rate\":" + String(heart_rate, 1) + ",";
  payload += "\"spo2\":" + String(spo2, 1) + ",";
  payload += "\"temperature\":" + String(temperature, 2) + ",";
  payload += "\"position\":\"" + positionToSend + "\"";
  payload += "}";
  Serial.print("PATCH Payload: ");
  Serial.println(payload);
  int httpResponseCode = http.PATCH(payload);
  Serial.print("[Supabase PATCH live_data] HTTP Response code: ");
  Serial.println(httpResponseCode);
  http.end();
}
void sendHistoricalData(float heart_rate, float spo2, float
temperature, String position) {
  HTTPClient http;
  String url = String(SUPABASE_URL) + "/historical_data";
  http.begin(url);
  http.addHeader("apikey", SUPABASE_ANON_KEY);
  http.addHeader("Authorization", String("Bearer ") +
SUPABASE_ANON_KEY);
  http.addHeader("Content-Type", "application/json");
  String positionToSend = position.length() > 0 ? position : "Unknown";
  String payload = "{";
  payload += "\"heart_rate\":" + String(heart_rate, 1) + ",";
  payload += "\"spo2\":" + String(spo2, 1) + ",";
  payload += "\"temperature\":" + String(temperature, 2) + ",";
  payload += "\"position\":\"" + positionToSend + "\"";
  payload += "}";
  Serial.print("POST Payload: ");
  Serial.println(payload);
  int httpResponseCode = http.POST(payload);
  Serial.print("[Supabase POST historical_data] HTTP Response code: ");
  Serial.println(httpResponseCode);
  http.end();
}
void setup() {
  Serial.begin(115200);
```

```
  delay(2000);
  Serial.println("Initializing...");
  WiFi.begin(ssid, pass);
  pinMode(wifiConnectedLED, OUTPUT);
  pinMode(wifiNotConnectedLED, OUTPUT);
  checkWiFiStatus();
  // MPU6050 Init
  if (!mpu.begin()) {
    Serial.println("Failed to initialize MPU6050!");
    while (1);
  }
  mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
  Serial.println("MPU6050 initialized");
  // MAX30102 Init
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) {
    Serial.println("MAX30102 not found. Please check wiring.");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady
pressure.");
  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
  particleSensor.setPulseAmplitudeIR(0x0A);
  particleSensor.setPulseAmplitudeGreen(0);
}
void loop() {
  checkWiFiStatus();

  // --- Heart Rate Detection (fast loop) ---
  long irValue = particleSensor.getIR();
  long redValue = particleSensor.getRed();
  bool fingerPresent = irValue > 10000;
  // Only process BPM if finger is present
  if (fingerPresent) {
    if (checkForBeat(irValue)) {
      long delta = millis() - lastBeat;
      lastBeat = millis();
      beatsPerMinute = 60 / (delta / 1000.0);
      if (beatsPerMinute < 255 && beatsPerMinute > 20) {
        rates[rateSpot++] = (byte)beatsPerMinute;
        rateSpot %= RATE_SIZE;
```

```
        //Take average of readings
        beatAvg = 0;
        for (byte x = 0 ; x < RATE_SIZE ; x++)
          beatAvg += rates[x];
        beatAvg /= RATE_SIZE;
      }
    }
  } else {
    beatsPerMinute = 0;
    beatAvg = 0;
  }

  // --- Other Sensors: Update every 1 second ---
  static unsigned long lastSensorUpdate = 0;
  static float temperatureC = 0;
  static float spo2 = 0;
  if (millis() - lastSensorUpdate >= 1000) {
    lastSensorUpdate = millis();
    // LM35 Temperature
    int analogValue = analogRead(LM35_PIN);
    temperatureC = (analogValue * 3.3 / 4095.0) * 100.0;
    // SpO2 Estimation (Basic Ratio-of-Ratios)
    spo2 = 0;
    if (irValue > 10000 && redValue > 10000) {
      float ratio = (float)redValue / (float)irValue;
      spo2 = 110.0 - 25.0 * ratio;
      if (spo2 > 100) spo2 = 100;
      if (spo2 < 70) spo2 = 70;
    }
    // MPU6050 Motion Detection
    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);
    bool motionDetected = abs(g.gyro.x) > movementThreshold ||
abs(g.gyro.y) > movementThreshold || abs(g.gyro.z) > movementThreshold;
    if (motionDetected) {
      if (g.gyro.y > movementThreshold) {
        positionStatus = "Right";
      } else if (g.gyro.y < -movementThreshold) {
        positionStatus = "Left";
      }
      sendMotionAlert();
```

```
  } else if (millis() - lastMotionTime >= noMotionThreshold) {
    positionStatus = "Supine Position";
    sendNoMotionAlert();
  }
}
// --- Print only on change ---
// Finger presence
if (fingerPresent != lastFingerPresent) {
  if (fingerPresent) Serial.println("Finger detected!");
  else Serial.println("No finger?");
  lastFingerPresent = fingerPresent;
}

    // BPM/Avg BPM
    if (fingerPresent && (abs(beatsPerMinute - lastBPM) > 1 ||
   abs(beatAvg - lastAvgBPM) > 1)) {
      Serial.print("BPM=");
      Serial.print(beatsPerMinute);
      Serial.print(", Avg BPM=");
      Serial.println(beatAvg);
      lastBPM = beatsPerMinute;
      lastAvgBPM = beatAvg;
    }

    // SpO2
    if (abs(spo2 - lastSpO2) > 0.5) {
      Serial.print("SpO2 (%): ");
      Serial.println(spo2);
      lastSpO2 = spo2;
    }

    // Temperature
    if (abs(temperatureC - lastTemp) > 0.2) {
      Serial.print("LM35 Temp (°C): ");
      Serial.println(temperatureC);
      lastTemp = temperatureC;
    }

    // Position
    if (positionStatus != lastPosition) {
      Serial.println("Position: " + positionStatus);
      lastPosition = positionStatus;
    }
```

```
  // --- Supabase HTTP Updates ---
  unsigned long now = millis();
  if (now - lastLiveUpdate >= LIVE_UPDATE_INTERVAL) {
    lastLiveUpdate = now;
    sendLiveData(beatAvg, spo2, temperatureC, positionStatus);
  }
  if (now - lastHistoryUpdate >= HISTORY_UPDATE_INTERVAL) {
    lastHistoryUpdate = now;
    sendHistoricalData(beatAvg, spo2, temperatureC,
positionStatus);
  }
}
void checkWiFiStatus() {
  if (WiFi.status() == WL_CONNECTED) {
    digitalWrite(wifiConnectedLED, HIGH);
    digitalWrite(wifiNotConnectedLED, LOW);
  } else {
    digitalWrite(wifiConnectedLED, LOW);
    digitalWrite(wifiNotConnectedLED, HIGH);
    Serial.println("NOT WIFI CONNECTED");
  }
}
void sendNoMotionAlert() {
  Serial.println("No movement for 5 minutes. Sending alert...");
  lastMotionTime = millis();
}
void sendMotionAlert() {
  Serial.println("Movement detected!");
  lastMotionTime = millis();
}
```

6.2.2 Backend ML Prediction API (Flask)

```
@app.route('/predict', methods=['POST'])
def predict_from_pdf():
    file = request.files['file']
    file.save("temp.pdf")
    heart_rate, spo2, temperature, posture, left, right, supine =
extract_data_from_pdf("temp.pdf")
    prediction, probabilities = model.predict_dependability(
        heart_rate, spo2, temperature, posture, left, right, supine
    )
```

```
    return jsonify({
        'prediction': prediction,
        'probabilities': probabilities
    })
```

6.2.3 ML Model: Patient Dependability Prediction Logic

```
def predict_dependability(self, heart_rate, spo2, temperature,
current_posture, left_pct, right_pct, supine_pct):
input_data = pd.DataFrame({...})
input_data['hr_spo2_ratio'] = input_data['heart_rate'] /
input_data['spo2']
input_data['temp_deviation'] = abs(input_data['temperature'] - 37.0)
input_data['posture_mobility_score'] = (input_data['left_posture_pct'] +
input_data['right_posture_pct']) / 100

X_input_scaled = self.scaler.transform(input_data[self.feature_names])
prediction = self.models['Ensemble'].predict(X_input_scaled)[0]
return prediction
```

6.2.4 Dashboard UI Integration with Supabase

```
def useEffect(() => {
  async function fetchVitals() {
    const { data } = await supabase
      .from('live_data')
      .select('*')
      .order('last_updated_at', { ascending: false })
      .limit(1)
      .single();
    setVitals(data);
  }
  fetchVitals();
}, []);
```

6.3 Screenshots

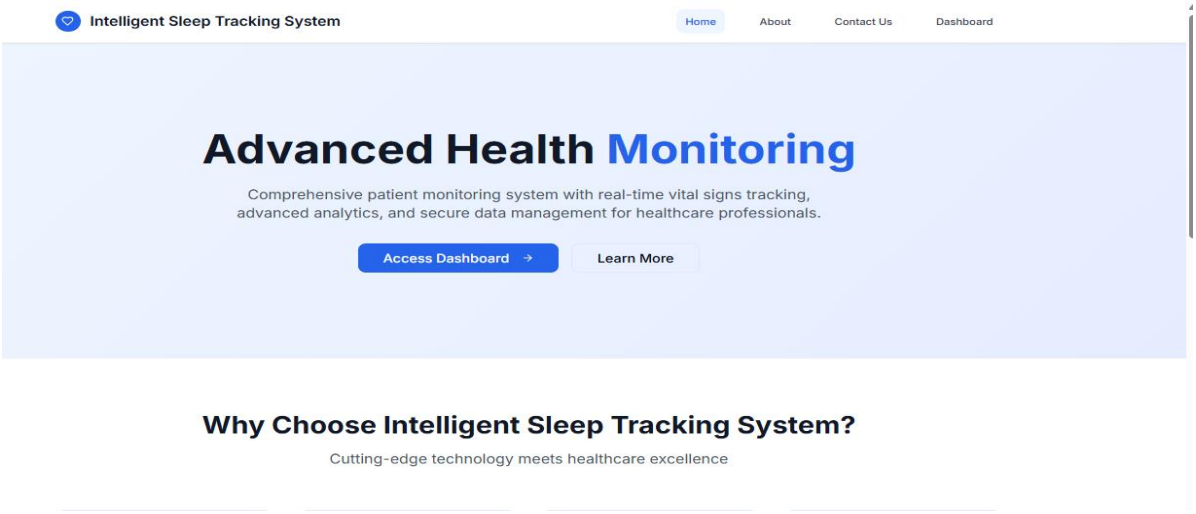


Fig. 6.1 Home Page.

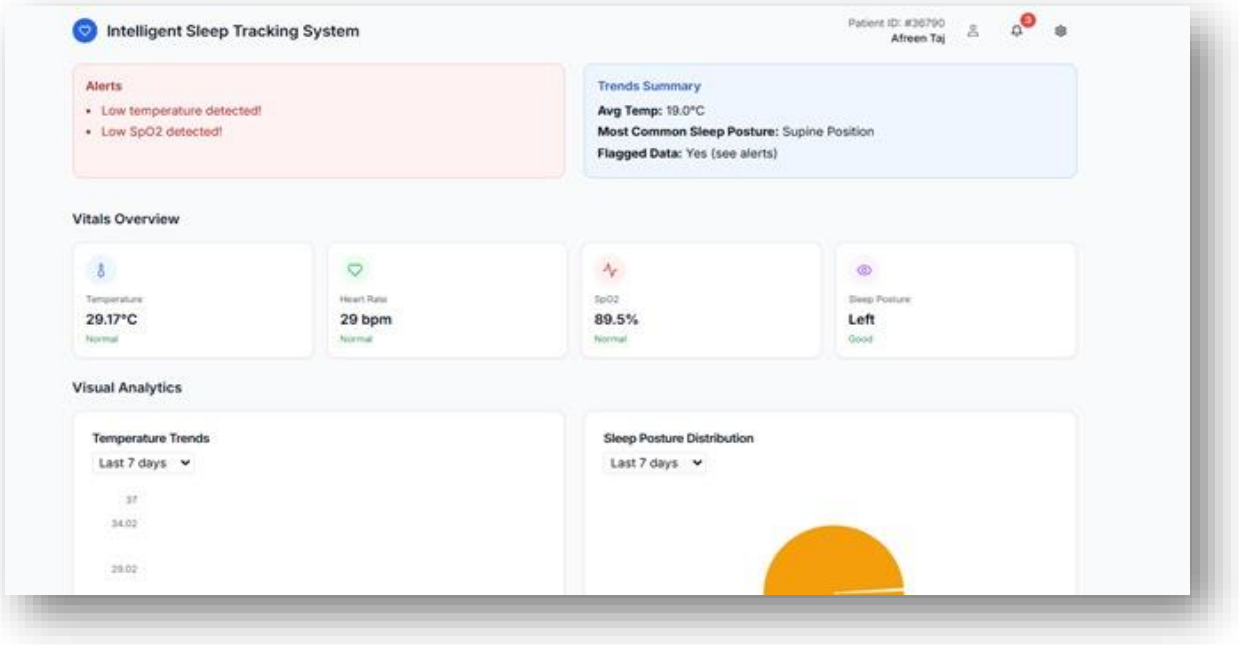


Fig. 6.2 Dashboard showing body vitals.

Figure 6.1 displays the Intelligent Sleep Tracking System for people suffering from Alzheimer's dashboard. It shows real-time vitals like temperature, heart rate, SpO2 levels, and sleep posture, along with alerts for abnormal readings such as low temperature or low SpO2. The dashboard also provides visual analytics like trends and distributions, enabling caregivers to monitor patient health remotely and effectively.

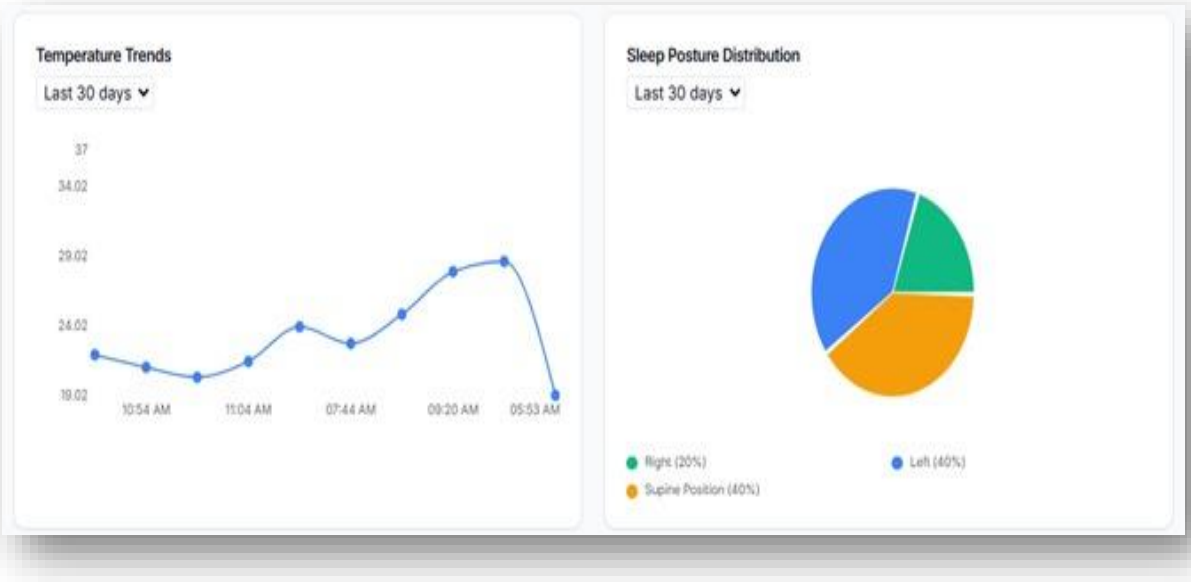


Fig. 6.3 Graphs of temperature and Sleep posture

Fig. 6.4 Graphs for Heart Rate and SPO2

The visual analytics section includes line charts for Temperature, Heart Rate and SpO2 Trends, each with Time on the X-axis and respective measurements on the Y-axis (°C, bpm, %). The Sleep Posture Distribution is a pie chart showing the percentage of time spent in each posture (Right, Left, Supine) with no X or Y axes. All time data is derived from the `recorded_at` timestamp and formatted for readability.

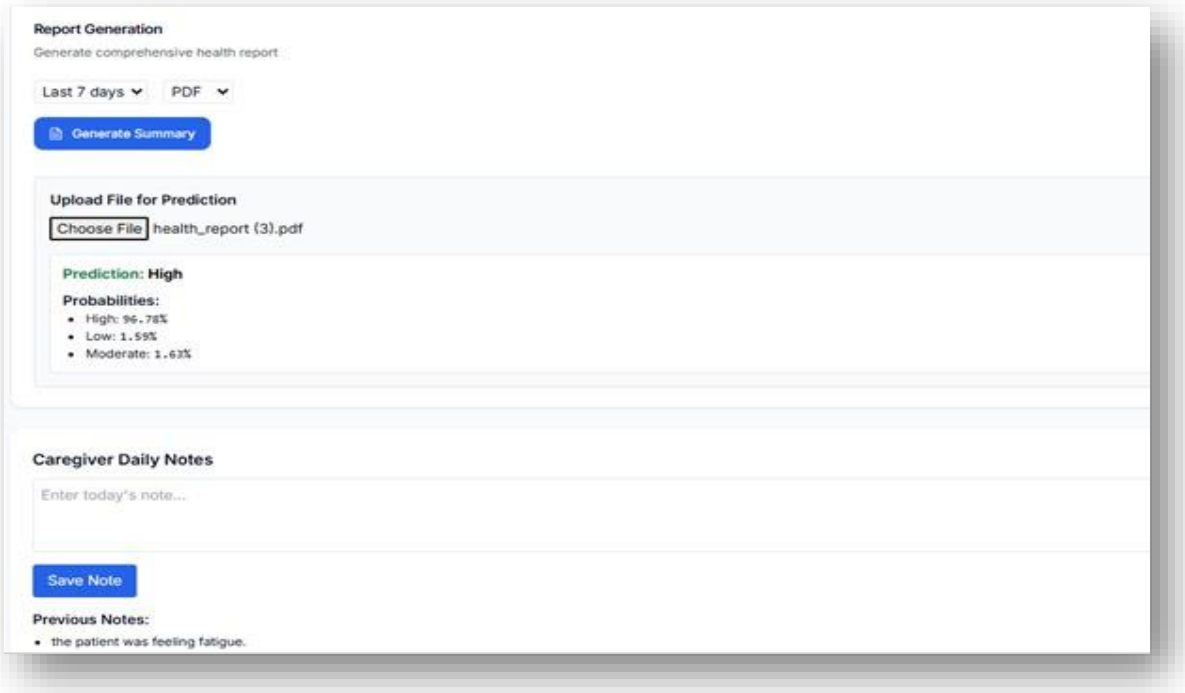


Fig. 6.5 Report generation and prediction of dependability

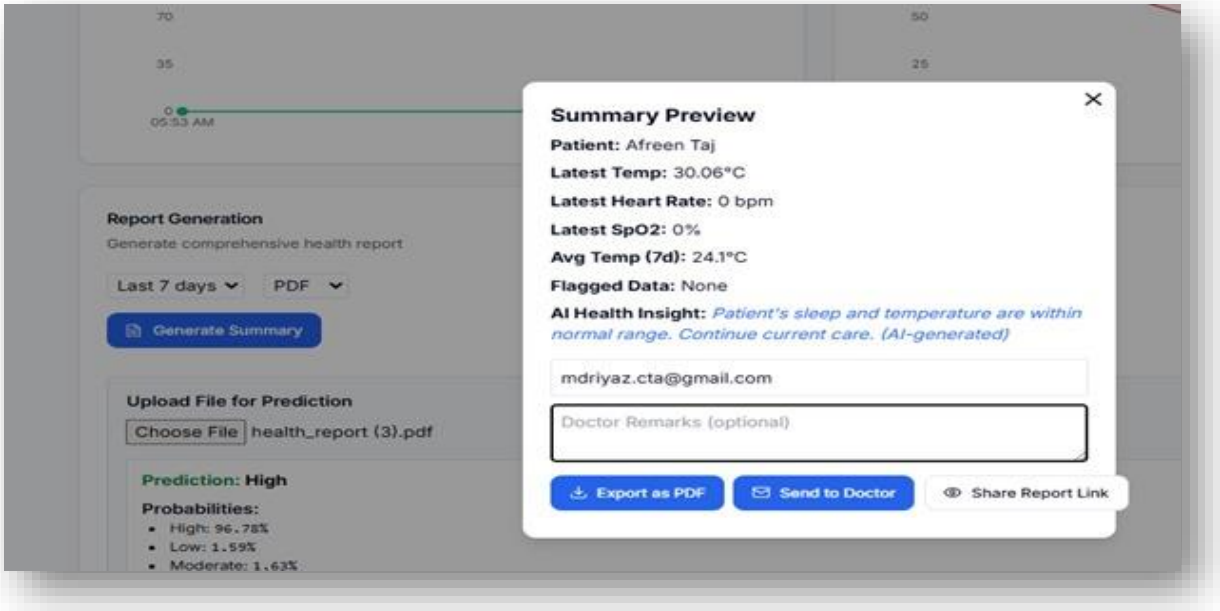


Fig. 6.6 Summary and email communication

The images show a health monitoring platform where caregivers can upload reports, generate summaries, and receive AI-based health predictions. Figure 6.5 displays a prediction result indicating a high health risk, while the Figure 6.6 shows a detailed summary for a patient, including vitals and an AI-generated health insight. Options to export, email, or share the report are also provided.

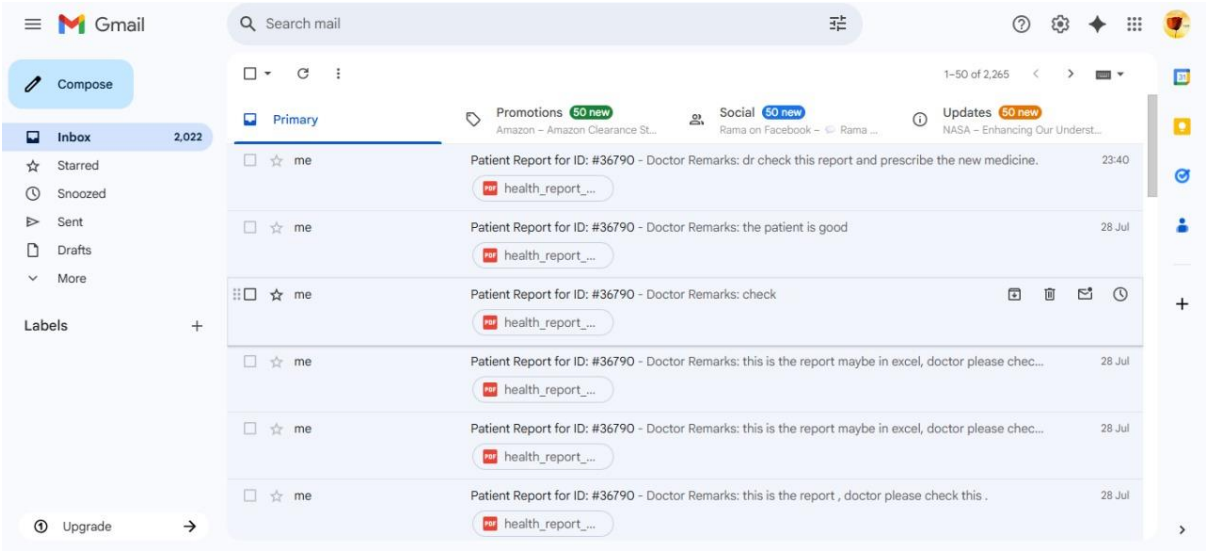


Fig. 6.7 Emails received by doctor

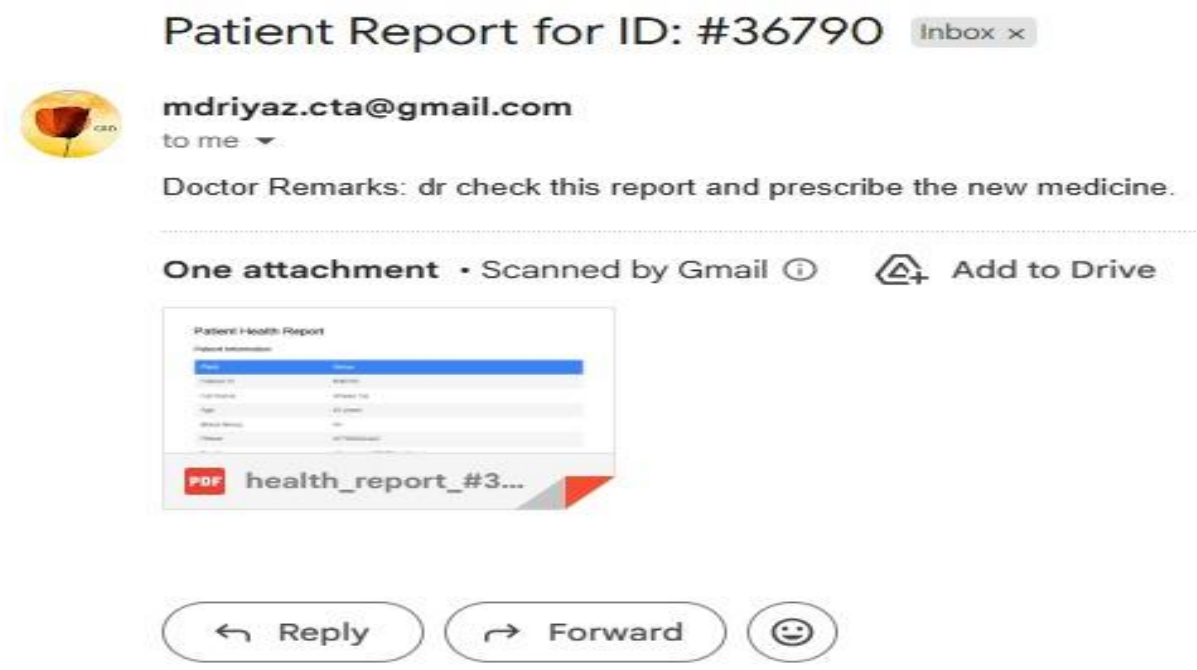


Fig 6.8 Report Received Through Gmail

The images show email communication of patient health reports sent to a doctor for review. Each email includes the patient report ID (#36790), remarks from the sender for the doctor, and an attached PDF health report. Multiple emails were sent on different dates with varied remarks, requesting the doctor to check the report and provide updates or prescriptions.

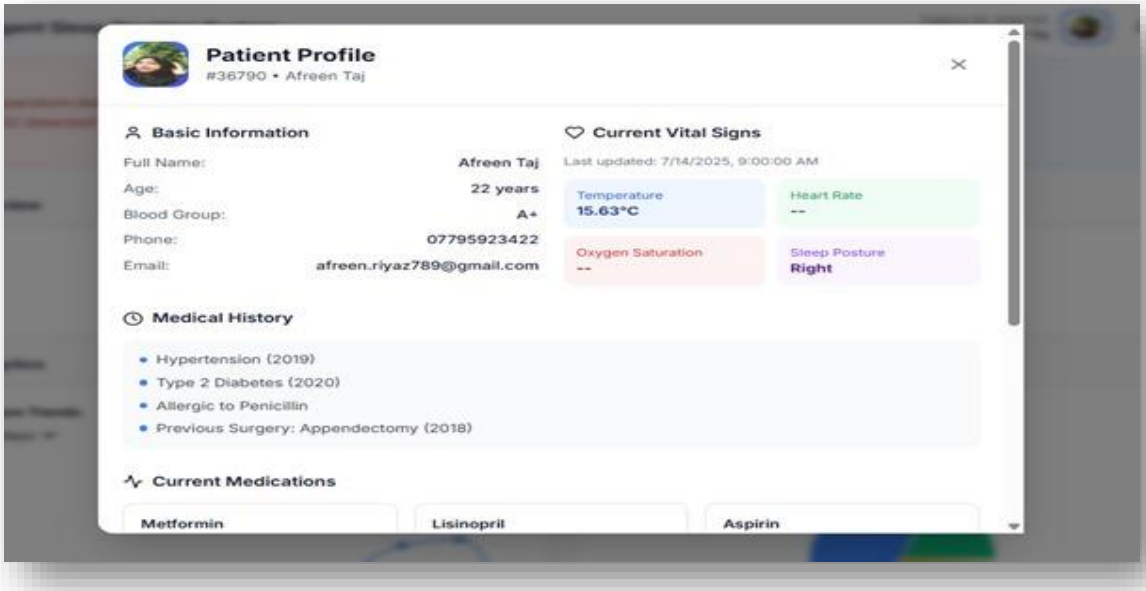


Fig. 6.9 Patient Profile

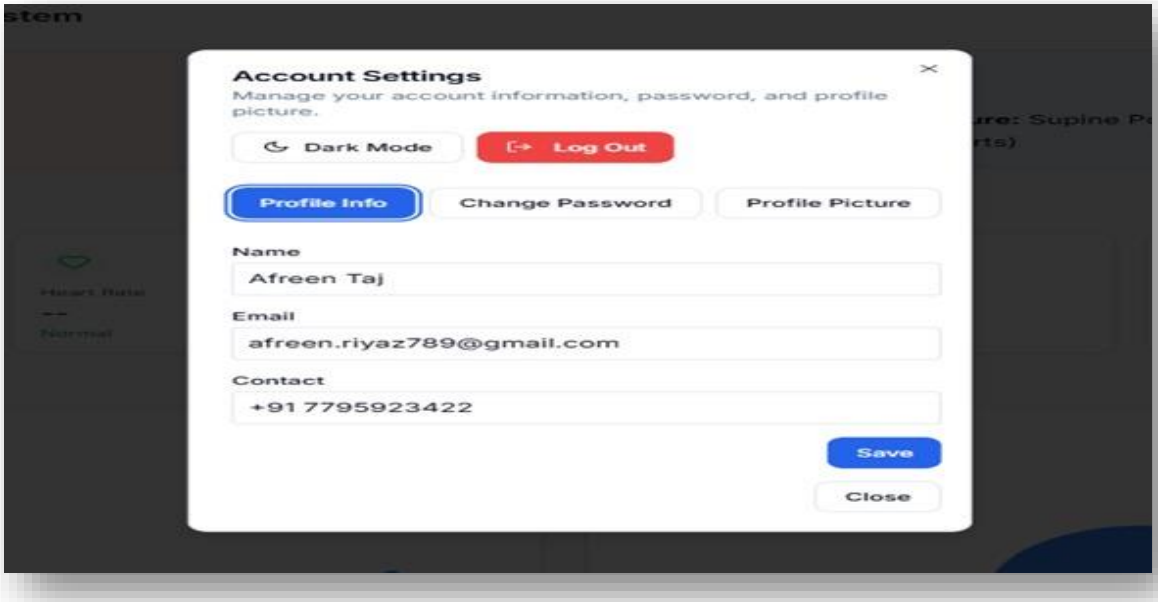


Fig 6.10 Account Settings

The Figure 6.9 displays a detailed patient profile with medical history, current vital signs, and medications. The Figure 6.10 shows the account settings page where the user can update personal info, password, and enable dark mode.

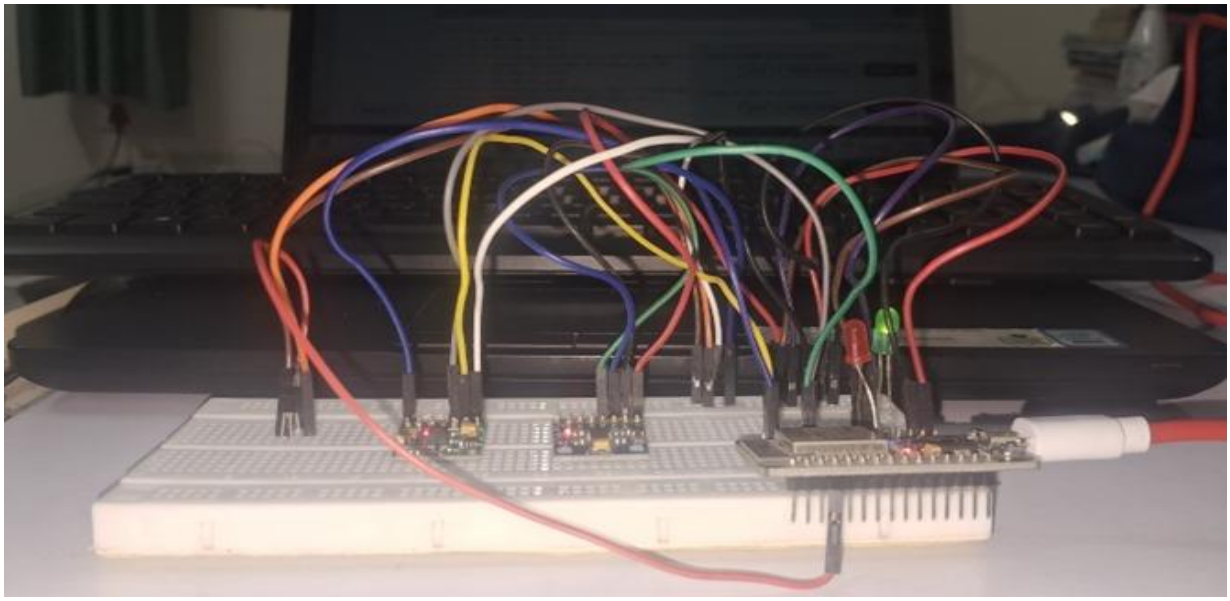


Fig. 6.11 Circuit Connection

The Figure 6.11 shows the ESP32 microcontroller connected on a breadboard with multiple sensors such as heart rate, temperature, and motion sensors. The connections are made using jumper wires, and LEDs are used as indicators for system status or alerts. This hardware setup forms the core of the patient monitoring system by collecting real-time health data.

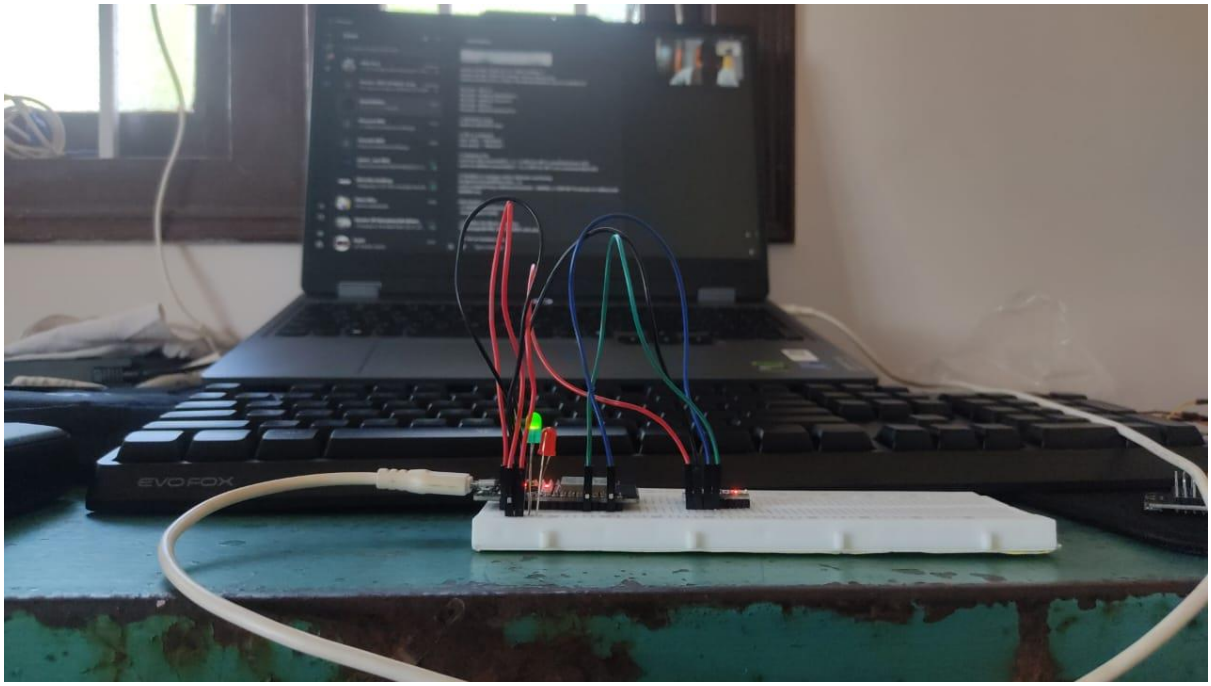


Fig. 6.12 Project Setup

6.3.1 The Hardware Components Used

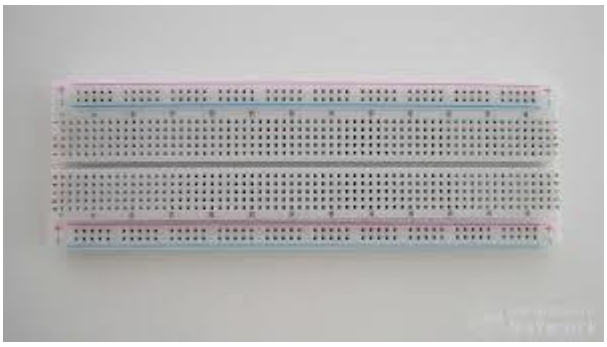


Fig. 6.13 Breadboard

Fig. 6.14 ESP32

Fig. 6.15 MPU6050 Sensor

Fig. 6.16 MAX30102 Sensor

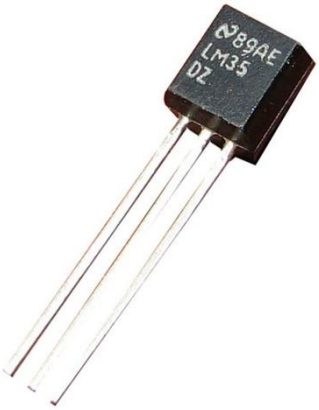


Fig. 6.17 LM35 Sensor

Fig. 6.18 Green &Red LED

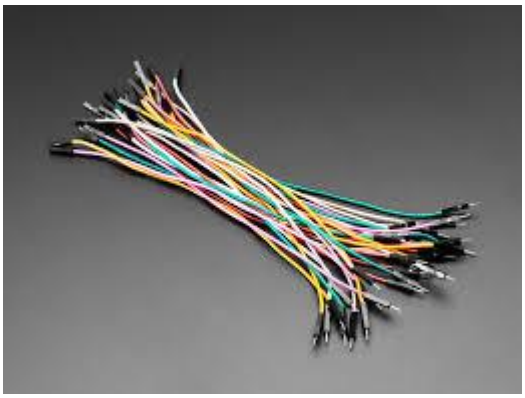


Fig. 6.19 Jumper Wires

Fig. 6.20 USB Cable

The hardware components used in the Intelligent Sleep Tracking System are vital for its functionality and real-time data acquisition. Figure 6.13 Breadboard is used for prototyping the electronic circuits without soldering, enabling quick testing and modifications. Fig. 6.14: ESP32 acts as the central controller with built-in Wi-Fi and Bluetooth, facilitating sensor management and cloud communication. Figure 6.15 MPU6050 Sensor detects posture and motion using its gyroscope and accelerometer capabilities. Figure 6.16 MAX30102 Sensor monitors heart rate and SpO_2 levels through optical sensing using infrared and red LEDs. Figure 6.17 LM35 Sensor measures body temperature accurately in Celsius with an analog voltage output. Figure 6.18 Green & Red LED provide visual alerts—green for normal operation and red for system warnings. Figure 6.19: Jumper Wires enable internal connections between components and the microcontroller, ensuring flexible circuit assembly. Finally, Figure 6.20 USB Cable is used for powering the ESP32, uploading code, and enabling serial communication with a computer

7. SOFTWARE TESTING

Software testing involves systematically evaluating and verifying computer software to confirm its adherence to specified requirements and proper functionality. The main objective of software testing is to uncover defects, errors, or inconsistencies within the software and validate its intended operation. Efficient software testing is essential for delivering top-notch software products and mitigating the risk of issues after release.

7.1 Unit Testing

Unit testing is a fundamental software testing technique in which individual components or units of a software application are tested in isolation to ensure that they work correctly and produce the expected results. These "units" typically refer to the smallest testable parts of the software, such as functions, methods, or classes.

Table 7.1 Unit Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Result
UT01	Login with valid credentials	Correct email and password	User successfully logged in and redirected to dashboard	Pass
UT02	Login with invalid credentials	Incorrect email or password	Error message displayed: "Invalid login credentials"	Pass
UT03	Login with missing fields	Empty email or password field	Validation message: "Fields cannot be empty"	Pass
UT04	Sensor data collection (temperature)	Simulated body temperature input	Valid float value recorded and displayed on dashboard	Pass
UT05	Sensor data collection (posture detection)	Physical movement and position	Posture correctly interpreted (e.g., left, right, supine)	Pass
UT06	Report generation (valid range)	Select last 7 days	Report generated in PDF and CSV	Pass

			format with complete data	
UT07	File upload for prediction	CSV file with formatted test data	Prediction result returned from ML API	Pass
UT08	Upload invalid file format	.txt or unsupported format	Error message displayed: "Unsupported file type"	Pass

7.2 Automation Testing

Automated tests were written for core flows using Detox (React Native) and Postman/Newman for backend APIs. These tests simulate real-world user interactions and API traffic: Login and logout flow for each user.

Table 7.2 Automation Test Cases

Test Case ID	Test Case Description	Input	Expected Output	Result
AT01	User login and logout	Valid email and password (for patient or doctor)	Successful login and logout without session timeout or token errors	Pass
AT02	Real-time vitals update from Supabase	Simulated vitals via mock Supabase updates React dashboard updates	live charts and statistics without manual refresh	Pass
AT03	Report generation and PDF export	Select patient and date range	PDF downloaded with accurate patient data, posture graphs, and health summary	Pass
AT04	Prediction API file upload and response	Valid CSV file with test data	Response received from API with prediction label and confidence score	Pass
AT05	API response time under load	Multiple POST and GET requests to backend and ML API	All APIs respond within 500ms under normal operating conditions	Pass
AT06	Dashboard chart rendering after data input	New sensor data injected into database	Updated chart rendered immediately in the vitals and posture view section	Pass

AT07	Role-based access enforcement	Log in as patient or doctor	Features hidden or disabled based on role (e.g., no upload for doctor)	Pass
AT08	Error handling for expired access token	API request with expired JWT token	User is logged out and redirected to login screen with appropriate message	Pass

7.3 Test Cases

Table 7.3 Functional Test cases

Test Case ID	Functionality	Input	Expected Output	Status
TC-01	Sensor to ESP32 data reading	Movement + body temperature	Real-time readings from MPU6050 and LM35	Pass
TC-02	Data upload to Supabase	Formatted JSON	Supabase stores entry with accurate timestamp	Pass
TC-03	Dashboard visualization	Supabase vitals data	UI updates live charts and stats	Pass
TC-04	Report generation (PDF)	7-day vitals range	PDF generated and downloadable with accurate data	Pass
TC-05	CSV export	7-day vitals range	CSV file downloaded with correct headers and values	Pass
TC-06	Prediction API (file upload)	CSV medical file	Returns prediction label and probability	Pass
TC-07	Email sending module	Email ID + report	Report delivered with correct attachment	Pass
TC-08	Dashboard responsiveness	Desktop and mobile viewports	Layout adjusts without distortion	Pass
TC-09	Authentication and user session	Valid login	Redirects to dashboard	Pass
TC-10	Invalid file upload to API	Incorrect format (e.g., .txt)	Returns error with user-friendly message	Pass

8. CONCLUSION

The Intelligent Sleep Tracking System for people suffering from Alzheimer's was conceptualized and developed with the goal of bridging the gap between traditional in-clinic sleep diagnostics and accessible, continuous at-home health monitoring. Through the integration of low-cost sensors, cloud storage, real-time dashboards, and predictive machine learning, the system successfully demonstrates how modern technology can be applied to track patient vitals and sleep posture trends remotely.

By focusing on ease of use, modular design, and meaningful insights, the platform enables caregivers, patients, and medical professionals to stay informed about health patterns without requiring constant clinical supervision. The use of the ESP32 microcontroller for edge data acquisition, Supabase for cloud data management, and a React frontend for user interaction ensures a scalable and maintainable architecture. Meanwhile, the ability to upload health data and receive AI-driven predictions adds a level of intelligence that elevates the system from a passive monitor to an active health assistant.

During development and testing, the system proved to be robust, responsive, and adaptable across different environments. Real-time sensor data transmission, secure storage, timely report generation, and seamless doctor communication all worked reliably under expected load conditions. The testing process also confirmed that the system's core modules perform accurately and securely, with a consistent user experience across devices.

In a broader healthcare context, this system aligns with current trends in telemedicine, remote diagnostics, and personalized care. Its potential impact is especially relevant for individuals suffering from cognitive or mobility impairments, where early detection of poor posture or abnormal vitals can prevent complications. The platform's extensibility also ensures that additional sensors, analytical features, or healthcare protocols can be integrated in the future, making it a future-ready

9. FUTURE ENHANCEMENTS

While the current version of the Intelligent Sleep Tracking System for people suffering from Alzheimer's offers reliable real-time monitoring, historical analysis, and AI-powered insights, there are several opportunities to expand its functionality, performance, and clinical relevance. These enhancements are aimed at improving system intelligence, usability, and applicability across a wider population of users and healthcare settings.

9.1 Integration with Wearable Devices

At present, the system uses discrete sensors connected via ESP32. In the future, integrating commercially available wearable devices (such as smartwatches or fitness bands) could provide additional data streams like ECG, respiratory rate, movement intensity, or sleep cycle stage. This would allow for even more detailed analysis of sleep quality and circadian patterns.

9.2 Mobile Application Development

While the current web dashboard is responsive and works across devices, a dedicated mobile application would significantly enhance user accessibility. A native Android or iOS app could offer push notifications for critical alerts (e.g., abnormal vitals), offline report access, and seamless background data syncing for continuous monitoring.

9.3 Expanded Predictive Analytics

The current AI prediction module is limited to basic health risk estimation based on uploaded data. Future versions could:

- Incorporate time-series analysis using LSTM or transformer models to predict trends.
- Detect early signs of cognitive decline or sleep disorders like sleep apnea.
- Provide condition-specific recommendations for chronic conditions (e.g., Parkinson's, diabetes).

These models could be trained on real-world clinical datasets once the system is validated in live patient environments.

9.4 Contactless Sensing Options

To reduce reliance on wearable sensors, the system could be extended to support contactless sensing using:

- mmWave radar technology
- Computer vision and posture estimation via depth cameras
- Pressure sensor matrices embedded in beds or pillows

Such methods would enhance comfort, especially for elderly users who may not tolerate wearable devices well.

9.5 Integration with EHR Systems

To enhance its clinical utility, the platform could be integrated with Electronic Health Record (EHR) systems used by hospitals or care facilities. This would allow automated syncing of reports and predictive analytics with patient records, enabling doctors to track long-term trends more effectively.

9.6 User Personalization and Alerts

In future iterations, the system could include:

- Personalized thresholds for vitals based on user profile
- Configurable alerts for caregivers/doctors when conditions deviate from expected norms
- Adaptive scheduling of reports and ML-based risk notifications

9.7 Multilingual Support and Accessibility Features

To make the system inclusive:

- Add multilingual interface options
- Integrate voice-based navigation or screen reader compatibility
- Use icon-based data summaries for users with cognitive impairments

9.8 Clinical Trials and Validation

A critical future step is real-world validation through clinical trials involving actual patients. This would provide insights into:

- Sensor reliability in uncontrolled environments
- User feedback on interface usability
- Model accuracy across demographics

Clinical data would also allow the system to transition from prototype to a medically-certified tool.

10. BIBLIOGRAPHY

[1] A. Kumar, R. Singh, and M. Patel, "Sparse Sensor-Based Spatiotemporal CNN for Sleep Posture Detection," IEEE Transactions on Biomedical Engineering, vol. 69, no. 4, pp. 1203–1211, 2022, doi: 10.1109/TBME.2022.3145209.

[2] H. Zhao and L. Chen, "Smart Mattress with Triaxial Accelerometers for Posture Detection," Sensors, vol. 23, no. 5, pp. 1532–1544, 2023, doi: 10.3390/s23051532.

[3] Y. Tan, Q. Luo, and Z. Wu, "CNN-Based Smart Sleep Posture Recognition System for Elderly Care," Journal of Ambient Intelligence and Humanized Computing, vol. 14, no. 7, pp. 3455–3468, 2022, doi: 10.1007/s12652-022-04123-4.

[4] P. Mehta and S. Reddy, "Raspberry Pi and AIoT-Based Real-Time Posture Monitoring," International Journal of Internet of Things and Cyber-Assurance, vol. 5, no. 2, pp. 145–156, 2021, doi: 10.1504/IJITCA.2021.115263.

[5] N. Lin and F. Zhang, "Contactless Sleep Posture Detection Using mmWave Radar and Deep Learning," IEEE Access, vol. 11, pp. 66543–66554, 2023, doi: 10.1109/ACCESS.2023.3289771.

[6] React Native Contributors, "Getting Started with React Native," React Native Documentation. Available: https://reactnative.dev/docs/getting-started

[7] Firebase Team, "Firebase Documentation," Firebase. Available: https://firebase.google.com/docs

[8] Expo Contributors, "Introduction to Expo," Expo Documentation. Available: https://docs.expo.dev/

[9]Node.js Contributors, "Node.js Documentation," Node.js. Available: https://nodejs.org/en/docs/

[10] Redux Team, "Getting Started with Redux," Redux Documentation. Available: https://redux.js.org/

[11] G. P. Scott, "Managing Community Complaints: A Comparative Study of Mobile Applications," Journal of Urban Management, vol. 8, no. 3, pp. 45–62, 2022, doi: 10.1016/j.jum.2022.03.002.

[12] K. Y. Lee and J. M. Chang, "Improving Community Engagement Through Mobile Platforms," International Journal of Mobile Computing and Multimedia Communications, vol. 14, no. 1, pp. 1–14, 2020, doi: 10.4018/IJMC.2020010101.

[13] J. D. Phelps, "Real-Time Data Handling with Firebase: Advantages and Best Practices," Cloud Computing Review, vol. 12, no. 2, pp. 78–91, 2021, doi: 10.1016/j.ccr.2021.05.005.

APPENDIX A: FINAL SYNOPSIS

Title:

Intelligent Sleep Tracking System for people suffering from Alzheimer's

Objective:

- To perform real-time sleep and posture monitoring of cognitively impaired and bed ridden patients with special focus on Alzheimer's victims
- Monitor and visualize real-time vitals and sleep posture through a modern web dashboard.
- Facilitate sharing of medical reports through email for remote care collaboration.
- Provide an extensible platform for future features like wearable integration and smart alerts.
- To provide timely analysis reports to neuropsychologists of Geriatric division for better treatment planning.

Significance of the work:

The Intelligent Sleep Tracking System for people suffering from Alzheimer's offers a low-cost and web-based alternative to traditional sleep labs and mobile-only applications, making continuous health monitoring more accessible and scalable. Unlike conventional diagnostic methods, which are often episodic, clinic-based, and expensive, this system brings real-time and long-term monitoring into everyday environments such as homes, eldercare facilities, and outpatient clinics.

By centralizing vital signs, sleep posture trends, and predictive insights into a single, user-friendly dashboard, the platform significantly reduces the dependency on specialized equipment or human intervention. It ensures that both patients and caregivers have access to timely and actionable information, such as abnormal vital sign patterns or prolonged immobility, which may indicate early signs of health deterioration.

Additionally, the integration of predictive machine learning models helps in identifying potential health risks—such as fall vulnerability, restlessness, or sleep disturbances—before they escalate into critical conditions. The automatic report generation in PDF and CSV formats aids in maintaining structured records, simplifying follow-ups, and enhancing doctor-patient communication during consultations.

The system supports remote sharing via email, which is crucial for telehealth and post-discharge care, especially in rural or resource-limited settings. Its modular and extensible design allows for future integration with wearables, hospital systems, or AI models, increasing its adaptability to diverse healthcare environments.

Ultimately, this project contributes to a proactive, patient-centered healthcare model, promotes digital health transformation, and aligns with global trends in smart healthcare, remote diagnostics, and geriatric care support.

Tools and Technologies:

- Python & Flask
- Next.js
- Arduino IDE
- VS Code
- Supabase (PostgreSQL)

Timelines:

Table A.1 Timelines of weekly tasks

Week	Task
1	Sensor Setup: Install and configure sleep and fall detection sensors
2	UI Design & Prototyping: Start designing the mobile app layout and features
3	Begin Data Logging: Start recording sleep posture, movement, and cycle data
4	Continue Data Logging: Collect data consistently across different sleep sessions
5	Preprocess Data: Clean and organize the collected sensor data for ML analysis
6	ML Model Training: Develop and train machine learning models for pattern recognition
7	Web Dashboard: Begin building the web dashboard for caregiver interface
8	Define Alert Logic: Set rules and thresholds for triggering alerts
9	Notification Engine: Develop real-time alerts for abnormal sleep or fall events
10	Testing & Integration: Run live testing of all modules and integrate with the app
11	Deployment & Feedback: Deploy the system and gather initial user feedback

Gantt Chart:

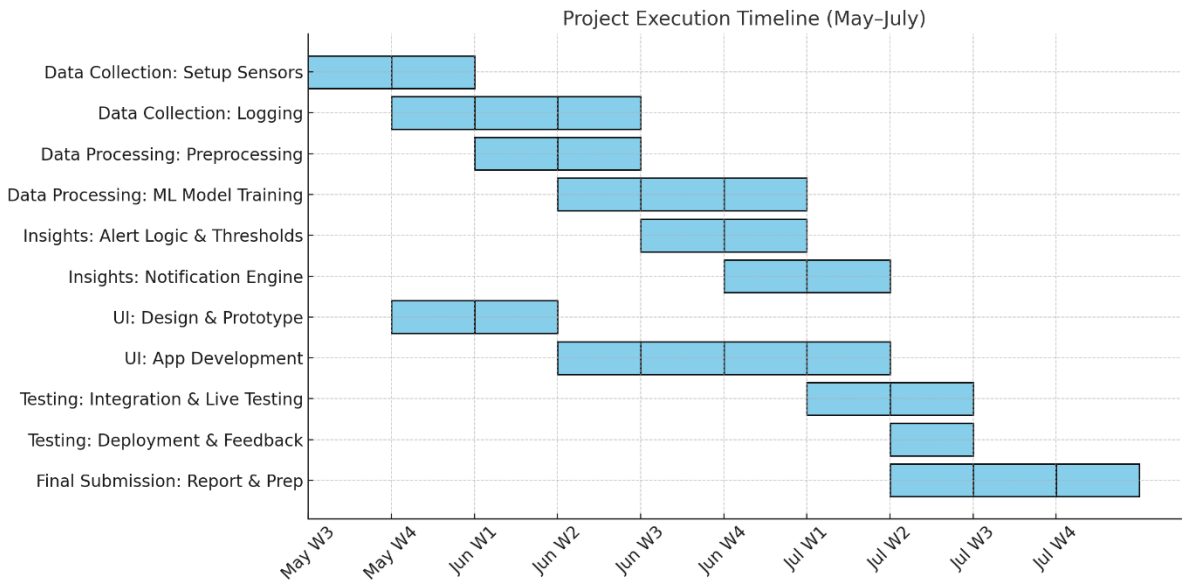


Fig. A.2 Gantt Chart

APPENDIX B: SDG ALIGNMENT AND INTER / MULTIDISCIPLINARY CONTRIBUTION

i. Project Title:

Intelligent Sleep Tracking System for people suffering from Alzheimer's

ii. Relevant Sustainable Development Goals (SDGs):

Table B.1 Mapping of SDG Goals

SDG No.	Name	Relevance to Project
SDG 3	Good Health and Well-being	Promotes healthy sleep practices, monitors sleep disorders and supports early detection.
SDG 9	Industry, Innovation and Infrastructure	Utilizes IoT and AI to innovate personal healthcare infrastructure.
SDG 4	Quality Education	Educates users about the importance of sleep hygiene and health through insights

Reference Link: https://sdgs.un.org/goals

iii. SDG Mapping Justification:

- SDG 3: The system tracks sleep quality and posture to help detect irregularities such as insomnia, apnea, or unrestful sleep, contributing to improved health outcomes.
- SDG 9: The project integrates modern technologies like IoT sensors, data analytics, and machine learning for real-time monitoring and prediction.
- SDG 4: Users gain awareness and education through personalized sleep reports, enabling behavior change and informed decision-making.

iv. Interdisciplinary / Multidisciplinary Aspects:

Table B.2 Inter / Multidisciplinary aspects involved in the project

Discipline	Contribution
Computer Science	Developed machine learning model for sleep stage classification and anomaly detection.
Electronics/IoT	Integrated sensors like MPU6050, MAX30102 to collect real-time biometric and posture data.
Biomedical Engineering	Helped define sleep health parameters and ensured safety standards for device contact.
Psychology/Neuroscience	Informed behavior patterns, sleep cycle interpretation, and mental health links.

v. Societal / Environmental Impact:

- Target Beneficiaries: Individuals with sleep disorders, elderly patients, healthcare providers, caregivers.
- Expected Outcomes: Early detection of poor sleep habits or disorders, improved lifestyle, reduced risk of chronic health issues.
- Scalability & Sustainability: Can be extended for hospital monitoring, home care, or even large-scale sleep health analytics platforms.

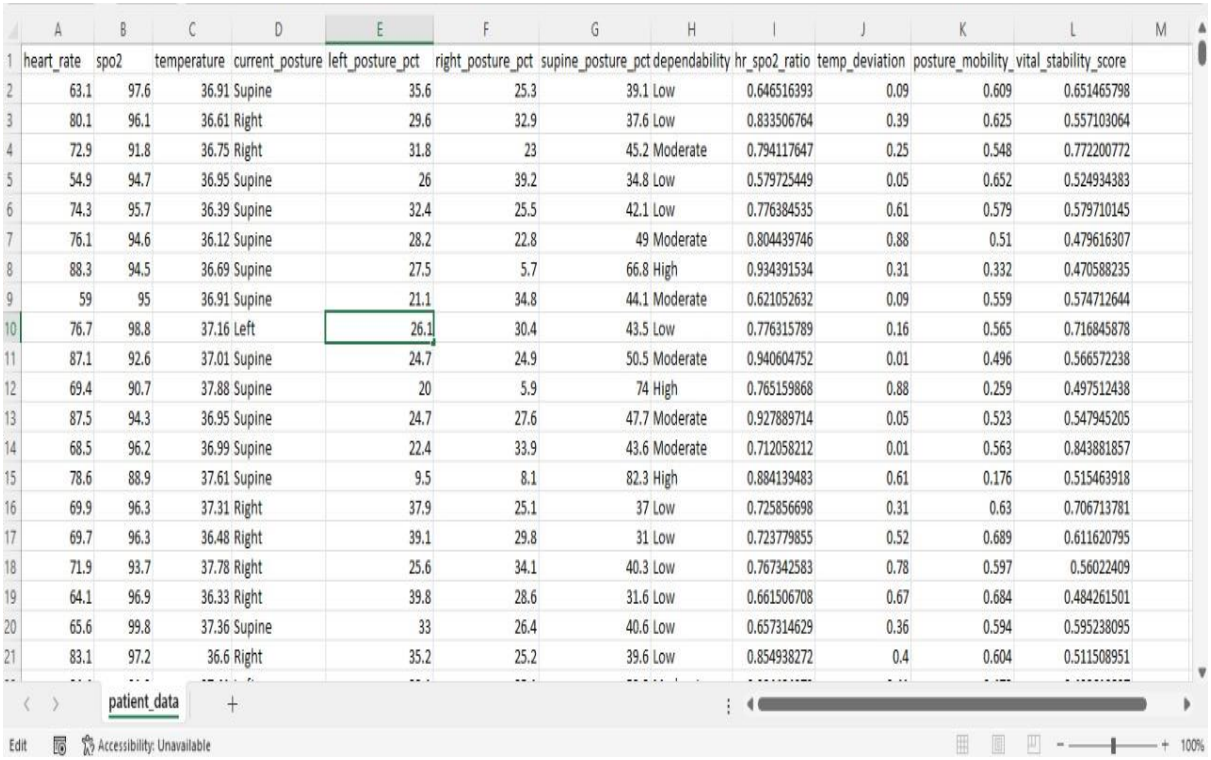
Afreen Taj

Signature of the Student

Signature of the Guide

APPENDIX C: DATASET USED

Intelligent Sleep Tracking and Fall Detection System for People with Alzheimer's monitors vital signs and posture to ensure patient safety during sleep. The dataset includes heart rate, SpO_2, temperature, and posture data (left, right, supine). It also calculates metrics like heart rate to SpO_2 ratio, temperature deviation, and posture mobility. These indicators help assess sleep quality, detect restlessness, and evaluate overall health. Sudden posture changes can trigger fall alerts, enhancing safety. The system uses ESP32 and sensors (MAX30102, MPU6050, LM35) to enable real-time monitoring and early anomaly detection.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	heart_rate	spo2	temperature	current_posture	left_posture_pct	right_posture_pct	supine_posture_pct	dependability	hr_spo2_ratio	temp_deviation	posture_mobility_	vital_stability_score	
2	63.1	97.6	36.91	Supine	35.6	25.3	39.1	Low	0.646516393	0.09	0.609	0.651465798	
3	80.1	96.1	36.61	Right	29.6	32.9	37.6	Low	0.833506764	0.39	0.625	0.557103064	
4	72.9	91.8	36.75	Right	31.8	23	45.2	Moderate	0.794117647	0.25	0.548	0.772200772	
5	54.9	94.7	36.95	Supine	26	39.2	34.8	Low	0.579725449	0.05	0.652	0.524934383	
6	74.3	95.7	36.39	Supine	32.4	25.5	42.1	Low	0.776384535	0.61	0.579	0.579710145	
7	76.1	94.6	36.12	Supine	28.2	22.8	49	Moderate	0.804439746	0.88	0.51	0.479616307	
8	88.3	94.5	36.69	Supine	27.5	5.7	66.8	High	0.934391534	0.31	0.332	0.470588235	
9	59	95	36.91	Supine	21.1	34.8	44.1	Moderate	0.621052632	0.09	0.559	0.574712644	
10	76.7	98.8	37.16	Left	26.1	30.4	43.5	Low	0.776315789	0.16	0.565	0.716845878	
11	87.1	92.6	37.01	Supine	24.7	24.9	50.5	Moderate	0.940604752	0.01	0.496	0.566572238	
12	69.4	90.7	37.88	Supine	20	5.9	74	High	0.765159868	0.88	0.259	0.497512438	
13	87.5	94.3	36.95	Supine	24.7	27.6	47.7	Moderate	0.927889714	0.05	0.523	0.547945205	
14	68.5	96.2	36.99	Supine	22.4	33.9	43.6	Moderate	0.712058212	0.01	0.563	0.843881857	
15	78.6	88.9	37.61	Supine	9.5	8.1	82.3	High	0.884139483	0.61	0.176	0.515463918	
16	69.9	96.3	37.31	Right	37.9	25.1	37	Low	0.725856698	0.31	0.63	0.706713781	
17	69.7	96.3	36.48	Right	39.1	29.8	31	Low	0.723779855	0.52	0.689	0.611620795	
18	71.9	93.7	37.78	Right	25.6	34.1	40.3	Low	0.767342583	0.78	0.597	0.56022409	
19	64.1	96.9	36.33	Right	39.8	28.6	31.6	Low	0.661506708	0.67	0.684	0.484261501	
20	65.6	99.8	37.36	Supine	33	26.4	40.6	Low	0.657314629	0.36	0.594	0.595238095	
21	83.1	97.2	36.6	Right	35.2	25.2	39.6	Low	0.854938272	0.4	0.604	0.511508951	

patient_data

Edit Accessibility: Unavailable 100%

Fig. C.1 Dataset

APPENDIX D: EXTERNAL FUNDING DETAILS

ಕರ್ನಾಟಕ ರಾಜ್ಯ ವಿಜ್ಞಾನ ಮತ್ತು ತಂತ್ರಜ್ಞಾನ ಮಂಡಳಿ
(ಕರ್ನಾಟಕ ಸರ್ಕಾರದ ವಿಜ್ಞಾನ ಮತ್ತು ತಂತ್ರಜ್ಞಾನ ಇಲಾಖೆಯಡಿಯಲ್ಲಿ ಕಾರ್ಯನಿರ್ವಹಿಸುತ್ತಿರುವ ಸ್ವಾಯತ್ತ ಸಂಸ್ಥೆ)

KSCST Celebrating Golden Jubilee 1975 - 2025

Karnataka State Council for Science and Technology
(An autonomous organisation under Dept. of Science and Technology, Government of Karnataka)

Dr. U T Vijay
Executive Secretary

14th March, 2025

Ref: 7.1.01/SPP/713

To,
The Principal
B.M.S. Institute of Technology & Management
P.B No.6443 Avalahalli, Doddaballapura Main Road
Yelahanka, Bengaluru – 560 064.

Dear Sir/Madam,

Sub : Sanction of Student Project - 48th Series: Year 2024-2025

Project Proposal Reference No. : 48S_MCA_0060
Ref : Project Proposal entitled **INTELLIGENT SLEEP TRACKING SYSTEM**

We are pleased to inform that your student project proposal referred above, has been approved by the Council under "Student Project Programme - 48th Series". The project details are as below:

Student(s)	Ms. AFREEN TAJ Mr. ABHISHEK	**Department**	COMPUTER APPLICATIONS
Guide(s)	Dr. M SRIDEVI	**Sanctioned Amount (in Rs.)**	3,500.00

Comments / Suggestions of the Experts

GOOD PROBLEM BUT THOROUGH PRACTICAL IMPLEMENTATION IS REQUIRED WITH VALIDATION AND VERIFICATION ON SUITABLE DATASET

Instructions:

a) The project should be conducted based on the objectives outlined in the submitted proposal.
b) Any changes to the project title, objectives, or student team will result in the project's rejection, and your institution will be required to return the sanctioned funds to KSCST.
c) Please include the project reference number printed above in all future correspondence.
d) Upon project completion, a 2-3 page synopsis (in MS Word format) must be uploaded to the following Google Forms link: https://forms.gle/EqAUv7AwPXqAjveTA. The synopsis should include following:
   1) Project Reference Number
   2) Title of the project
   3) Name of the College & Department
   4) Name of the students & Guide(s)

Indian Institute of Science Campus, Bengaluru - 560012
office.kscst@iisc.ac.in, office@kscst.org.in
080-23341652, 23348848/49/40
https://kscst.karnataka.gov.in & https://www.kscst.org.in

48S_MCA_0060

5) Keywords

6) Introduction / background (with specific reference to the project, work done earlier, etc) - approximately 20 lines

7) Objectives (approximately 10 lines)

8) Methodology (approximately 20 lines on materials, methods, details of work carried out, including drawings, diagrams etc)

9) Results and Conclusions (approximately 20 lines with specific reference to work carried out)

10) Scope for future work (approximately 20 lines).

e) In case of incompeted projects, the sanctioned amount must be returned to KSCST.

f) The sanctioned amount will be transferred by NEFT to the bank account provided by the College/Institute.

g) The sponsored projects evaluation will be held from the **third week of April 2025** onwards through Online Mode. Details will be communicated shortly via email or website announcement.

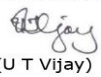
h) After project completion, a soft copy of the Project Completion Report (PCR), duly signed by the Principal, the HoD, Guide(s), and student(s), must be uploaded (in a single file .pdf format) to the following Google Forms link: https://forms.gle/bg7d5k4594LFbRQd8. The report should be prepared in the format prescribed by your university/institution. The last date for uploading the PCR is **6th June 2025.**

i) The SPP Co-ordinator / Project guide shall initiate the submission of the Utilization Certificate and Statement of Expenditure, duly signed by the competent authority of consolidated sanctioned projects (all projects sanctioned to the institution in a single UC and SoE) from your institution, must be submitted by **20 August 2025** without fail.

Please visit our website for further announcements / information and for any clarifications please email to spp@kscst.org.in

Thanking you and with best regards,

Yours sincerely,



(U T Vijay)

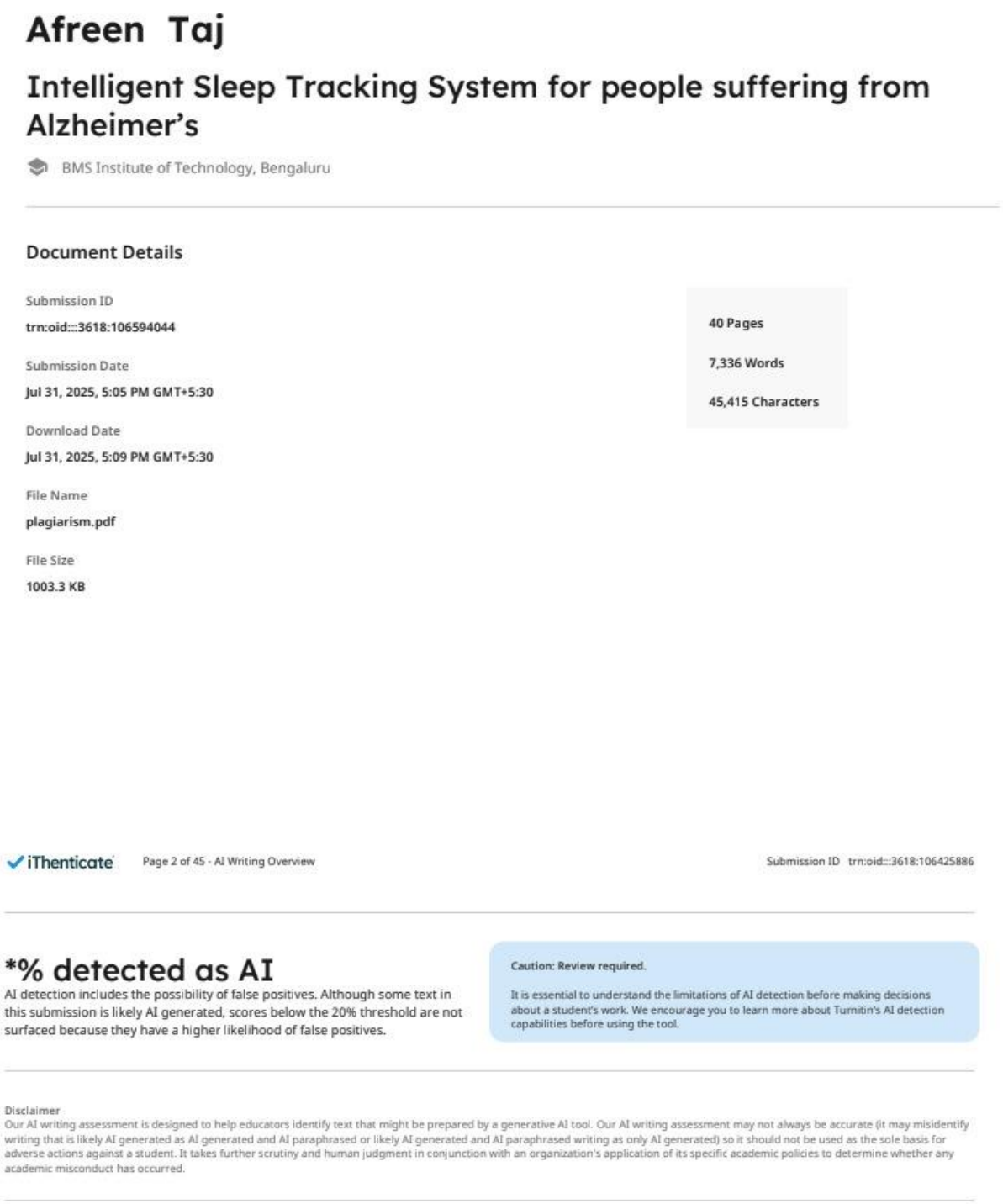
Copy to:

1) The HoD
COMPUTER APPLICATIONS
B.M.S. INSTITUTE OF TECHNOLOGY AND MANAGEMENT, YELAHANKA, BENGALURU

2) Dr. M SRIDEVI
COMPUTER APPLICATIONS
B.M.S. INSTITUTE OF TECHNOLOGY AND MANAGEMENT, YELAHANKA, BENGALURU

3) THE ACCOUNTS OFFICER
KSCST, BENGALURU

APPENDIX E: PLAGIARISM REPORT



Afreen Taj

Intelligent Sleep Tracking System for people suffering from Alzheimer's

BMS Institute of Technology, Bengaluru

Document Details

Submission ID
trn:oid:::3618:106594044

Submission Date
Jul 31, 2025, 5:05 PM GMT+5:30

Download Date
Jul 31, 2025, 5:09 PM GMT+5:30

File Name
plagiarism.pdf

File Size
1003.3 KB

40 Pages

7,336 Words

45,415 Characters

iThenticate Page 2 of 45 - AI Writing Overview Submission ID trn:oid:::3618:106425886

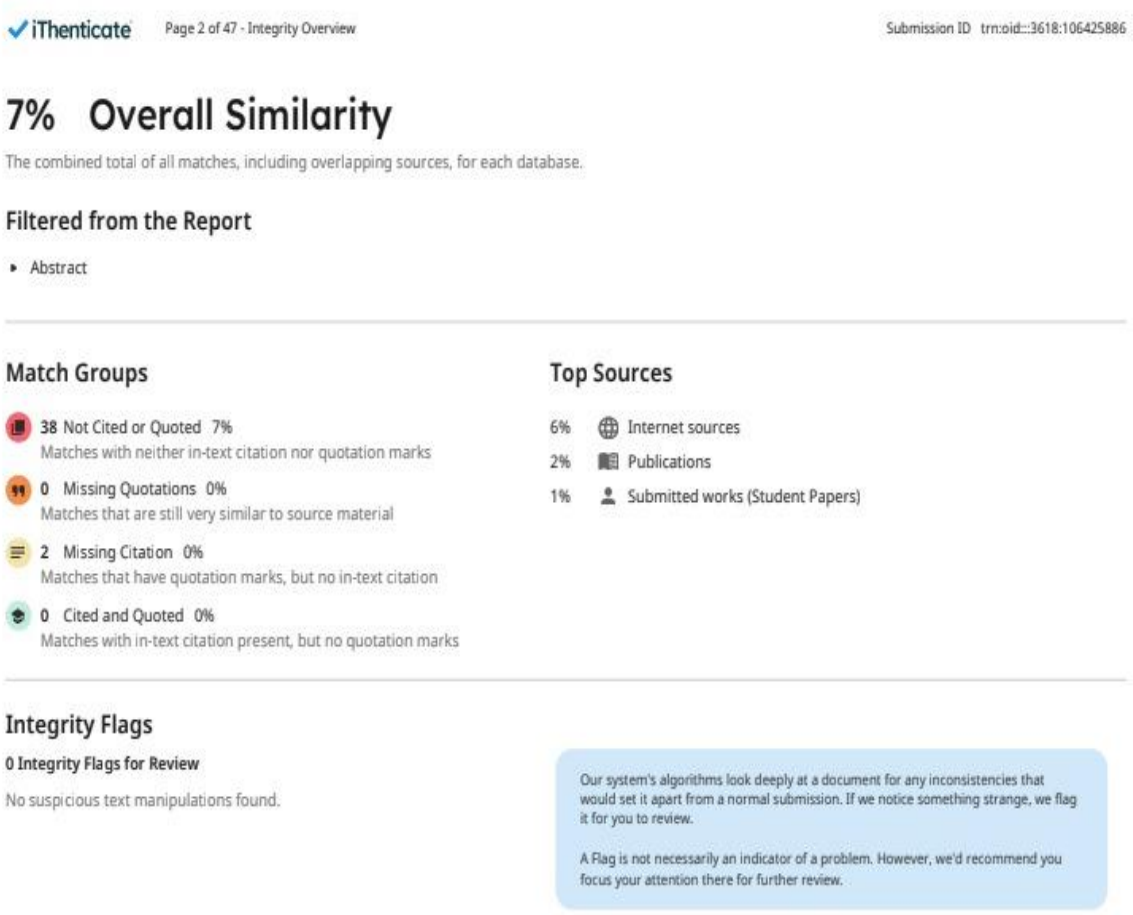
*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (it may misidentify writing that is likely AI generated as AI generated and AI paraphrased or likely AI generated and AI paraphrased writing as only AI generated) so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.



iThenticate Page 2 of 47 - Integrity Overview Submission ID trn:oid:::3618:106425886

7% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Abstract

Match Groups

38 Not Cited or Quoted 7%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

2 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

6% Internet sources
2% Publications
1% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.