IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

AURA: Advanced Unified Responsive Assistant for Medical Emergencies

Prathibha D[1], Rakshith D Achar[2], Swathi Ravindra Kodabagi[3], Tejashwini[4], Rashmi[5]

Student, Dept. of Computer Science and Engineering, East West College of Engineering, Bengaluru, India[1]

Student, Dept. of Computer Science and Engineering, East West College of Engineering, Bengaluru, India[2]

Student, Dept. of Computer Science and Engineering, East West College of Engineering, Bengaluru, India[3]

Student, Dept. of Computer Science and Engineering, East West College of Engineering, Bengaluru, India[4]

Project Guide, Dept. of Computer Science and Engineering, East West College of Engineering, Bengaluru, India[5]

Abstract: Modern healthcare systems often struggle with slow disease detection, disconnected patient care, and delayed emergency responses. To address these challenges, this paper presents AURA (Advanced Unified Responsive Assistant for Medical Emergencies), an integrated healthcare platform designed to provide intelligent, accessible, and timely medical assistance. The proposed system combines four major functionalities within a single application: a machine learning based disease prediction module, an AI powered medical chatbot for answering general healthcare queries, a real time SOS emergency feature that shares the user's live location with nearby hospitals via the Twilio API, and a medical supply chain tracking module for monitoring the distribution of essential healthcare resources. Unlike conventional healthcare applications that focus on individual services, AURA unifies these components into a comprehensive platform. To ensure patient safety, the chatbot utilizes a custom RAG pipeline with Hybrid Retrieval (PubMedBERT and BM25) and a Grounding Verifier, providing medically accurate, citation backed answers from a verified database of 16,407 medical records. Built on a robust microservices architecture using React.js for the frontend, Java Spring Boot for secure JWT authentication, and Python FastAPI for AI processing, AURA drastically reduces time to treatment and optimizes hospital resource management.

Keywords: Healthcare AI, Retrieval.Augmented Generation (RAG), Machine Learning, Emergency Alert System, Microservices Architecture, Spring Boot, React.js, Medical Supply Chain.

I. INTRODUCTION

Healthcare is an important part of everyday life, and access to timely medical assistance can make a significant difference in a patient's condition. Over the years, digital technologies have changed the way healthcare services are provided, from maintaining patient records to assisting doctors in identifying possible health conditions. Artificial Intelligence (AI) and Machine Learning (ML) have further contributed to these developments by making it possible to analyse medical information and support certain healthcare activities. However, despite the availability of these technologies, several practical difficulties still exist in the way healthcare services are accessed and managed. Patients may have to depend on different applications for medical information, disease assessment, and emergency assistance, while hospitals use separate systems to manage their operations and resources. This separation often makes it difficult to coordinate these activities when they are needed together.

One of the common challenges faced by patients is understanding the possible causes of their health symptoms. People may experience problems such as fatigue, dizziness, changes in weight, or other physical discomforts without knowing whether these symptoms require medical attention. Identifying the underlying condition may involve consulting a doctor, undergoing laboratory tests, and waiting for the examination results. Although these procedures are necessary for proper medical diagnosis, patients may not always have immediate access to them. Machine Learning provides an opportunity to analyse available clinical information, demographic details, and lifestyle-related factors to estimate the possibility of selected health conditions. Such systems can offer preliminary information that helps users become more aware of their health, while the final diagnosis and treatment decisions remain with qualified medical professionals.

Apart from identifying possible health conditions, access to reliable medical information is another important requirement. Patients often have questions about common diseases, symptoms, preventive measures, and general healthcare practices. Medical chatbots can help users obtain information through simple conversations without having to search through multiple websites. However, traditional chatbots and general-purpose Large Language Models (LLMs)

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.1511151

may sometimes generate answers that are incomplete, misleading, or not supported by reliable medical sources. This is a particular concern in healthcare because users may consider the generated information while making decisions about their health. Therefore, providing relevant information alone is not sufficient; the responses must also be checked for their connection to reliable medical evidence.

To address this issue, AURA incorporates a Retrieval-Augmented Generation (RAG)-based medical chatbot. Instead of depending entirely on the information learned by the language model during training, the system first searches a medical knowledge base for information related to the user's question. It uses PubMedBERT to generate text embeddings, ChromaDB for vector storage, and BM25 for keyword-based retrieval. The retrieved information is then provided to Gemma 2B through Ollama to generate a response. A grounding verification component checks the generated answers for unsupported medical claims and certain unsafe diagnostic or medication instructions. This process is intended to make the chatbot's responses more closely related to the available medical information and reduce the possibility of unsupported answers.

Emergency assistance is another area where timely communication and proper coordination are important. During a medical emergency, patients or their family members may find it difficult to identify a nearby hospital and contact the appropriate healthcare facility. Searching for hospitals manually can add further difficulty, particularly when immediate assistance is required. Location-based technologies can help simplify this process by identifying healthcare facilities based on the user's geographical position. AURA addresses this requirement through its SOS module, which captures the patient's GPS coordinates and uses the Haversine formula to identify the nearest hospital from the available hospital records. The system integrates Twilio to send SMS and automated voice notifications to the assigned hospital. Hospital administrators can then view the incoming emergency request and accept or reject it. When a request is rejected, the system can identify the next nearest hospital and forward the request accordingly.

Hospital resource management is equally important because the availability of essential medical supplies directly affects the ability of healthcare facilities to handle patient requirements. Medicines, beds, and other medical resources need to be maintained in sufficient quantities, but the demand for these resources can change during disease outbreaks or sudden increases in reported cases. In some situations, one hospital may face a shortage while nearby hospitals may have resources available. AURA includes a medical supply chain module to support coordination in such situations. The module follows a rule-based approach in which the number of reported disease cases is compared with a predefined threshold. When the reported cases reach or exceed this threshold, the system generates alerts for connected or nearby hospitals. Their administrators can respond to the alerts, raise supply requests, and coordinate the transfer of available resources to the affected hospital.

Considering these requirements, we developed AURA (Advanced Unified Responsive Assistant for Medical Emergencies) as a web-based platform that brings these healthcare functions together within a single application. The system is designed for three main user roles: Patients, Hospital Administrators, and Supply Administrators. Each role has a separate dashboard that provides access to the operations related to its responsibilities. Patients can use the platform for preliminary disease risk assessment, medical queries, and emergency assistance, while hospital administrators can manage incoming SOS requests and participate in supply coordination. Supply administrators can view alerts and handle requests related to medical resources. This arrangement brings different activities into a common platform rather than requiring users to depend on separate applications for each function.

The platform is developed using React.js for the frontend, Java Spring Boot for backend operations, Python FastAPI for AI-based services, and PostgreSQL for structured data storage. The backend manages communication between the frontend, database, and AI services through REST APIs. Spring Security and JSON Web Tokens (JWT) are used for authentication and role-based access control, ensuring that users can access only the operations permitted for their roles. The AI services handle disease prediction and chatbot processing independently of the main application interface. This division also makes it easier to maintain individual components and introduce improvements without requiring major changes to the entire application.

The main aim of developing AURA is to bring medical information assistance, preliminary disease risk assessment, emergency communication, and hospital supply coordination into one accessible platform. Rather than focusing on a single healthcare function, the project attempts to address several related requirements that patients and hospital administrators may encounter. The developed system demonstrates how machine learning, retrieval-based conversational AI, geographical distance calculations, and rule-based resource coordination can be used together in a practical healthcare application. Although further improvements and clinical validation are required before wider real-world use, AURA provides a foundation for developing more connected and accessible healthcare support systems.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

II. BACKGROUND CONCEPTS

A. Machine Learning for Symptom Based Prediction

Machine learning classifiers, such as Random Forest and XGBoost, are highly capable of mapping complex symptom clusters to potential diseases. While many existing systems rely on structured clinical lab data, AURA utilizes these models to predict frequently neglected, symptom heavy conditions (such as Thyroid disorders, Vitamin deficiencies, and Low Blood Pressure) based on user reported inputs, enabling accessible, non-invasive early screening.

B. Retrieval Augmented Generation (RAG)

Large Language Models (LLMs) provide conversational medical assistance but frequently generate hallucinated or inaccurate responses. Retrieval Augmented Generation (RAG) substantially improves reliability by grounding LLM outputs in trusted medical knowledge bases. AURA implements a custom RAG pipeline that converts user queries into vector embeddings, retrieves the most relevant document chunks from a vector database, and injects this context into the prompt for the language model, ensuring responses are evidence based and cited.

C. Real Time Geolocation and Emergency Dispatch

Rapid emergency response relies heavily on automated alerts. By utilizing the Haversine formula and GPS tracking, healthcare platforms can calculate the proximity of medical facilities in real time. AURA integrates this with cloud communication APIs (Twilio) to automatically dispatch SMS and voice alerts to nearby hospitals and emergency contacts when a critical symptom threshold is breached.

D. Microservices Architecture in Healthcare

To ensure scalability and separation of concerns, AURA adopts a modern 3 tier microservices architecture. A React.based frontend handles user interactions, a Java Spring Boot backend acts as the central API gateway handling secure authentication via JSON Web Tokens (JWT), and a Python FastAPI service handles the heavy computational load of the AI and ML models.

III. LITERATURE REVIEW

A. Disease Prediction Systems

Kanmani et al. [1] presented a multi-disease prediction system that uses machine learning techniques to identify conditions such as kidney disease, breast cancer, and heart disease through a common web application. Their work demonstrates how algorithms such as Random Forest and XGBoost can be used to analyse health-related data and support preliminary disease assessment. However, the study mainly focuses on disease prediction and does not extend to other healthcare services such as medical chatbot assistance, emergency response, or hospital coordination. AURA builds on this approach by integrating multiple condition-specific prediction models with other healthcare functionalities, allowing users to access different services through a single platform.

B. Medical Chatbots and RAG

Tomar et al. [2] proposed a knowledge-infused disease diagnosis assistant that combines transformer-based models with a medical knowledge graph to identify possible diseases from patient-doctor conversations. Their approach highlights the importance of using medical knowledge to improve the relevance of AI-generated responses. Similarly, Bora and Cuayáhuitl [3] explored Retrieval-Augmented Generation with medical question-answering datasets to improve the quality of responses generated by language models. AURA follows a related approach by combining PubMedBERT embeddings, BM25 retrieval, and ChromaDB to obtain relevant medical information before passing it to Gemma 2B for response generation. It also includes a grounding verification component to check for unsupported medical claims and certain unsafe medical instructions.

C. Emergency Alert Systems

Weinlich et al. [4] studied the use of smartphone-based geolocation technologies to support emergency calls and improve the transfer of location information to emergency services. Their work demonstrates how GPS and other location technologies can help identify a caller's position when requesting medical assistance. AURA adopts a similar location-based approach through its SOS module, which captures the patient's geographical coordinates and uses the Haversine formula to identify the nearest hospital. Twilio is integrated to send SMS and automated voice notifications to the assigned hospital. Hospital administrators can accept or reject incoming requests, and rejected requests can be forwarded to another nearby hospital.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

D. Medical Supply Chain

Navaei et al. [5] examined the factors affecting knowledge management in healthcare supply chains using decision-making techniques such as the Best-Worst Method and D-Numbers. Their research highlights the importance of information sharing and proper resource management in healthcare institutions. Similarly, Luo et al. [6] explored the use of K-Means clustering for the classification of medical consumables. These studies demonstrate different approaches to improving healthcare supply chain operations. AURA uses a rule-based approach in which reported disease cases are compared against predefined thresholds. When the threshold is reached, supply alerts are generated for connected or nearby hospitals, allowing administrators to raise supply requests and coordinate the transfer of essential medical resources.

IV. COMPARATIVE ANALYSIS AND RESEARCH GAP

Table I compares the reviewed systems against AURA along five dimensions: core focus, prediction approach, how (if at all) chatbot guidance is handled, whether outputs are explainable, and whether a patient facing transparency signal is produced. No reviewed system addresses more than two of these dimensions at once, which motivates a single, unified design.

TABLE I COMPARATIVE ANALYSIS OF REVIEWED SYSTEMS

Study	Core Focus	Prediction Approach	Chatbot / Guidance Handling	Explainable Output	Patient-Facing
Kanmani et al. (2025) [1]	Multi.disease prediction via structured lab data	Random Forest / XGBoost ensemble	Not addressed	Partial	No
Tomar et al. (2024) [2]	Automated disease diagnosis assistant	Knowledge graph + LLM	Flags unverified outputs	Partial	No
Lee & Kim (2023) [3]	Real.time medical QA chatbot	RAG framework	Reduces hallucination	No	Yes (chat only)
Benreguia (2024) [4]	Emergency call optimization	GPS.based routing	Not addressed	No	No
ML in E.Commerce Survey [5]	Cross.cutting ML applications	Reviews multiple paradigms	Discussed conceptually only	Discussed conceptually only	No
AURA (Proposed)	Prediction + Chatbot + SOS + Supply	Stacked ensemble (DPE)	RAG anomaly + Grounding Verifier	Yes- evidence backed Guidance Score	Yes- transparency dashboard

The comparison in Table I highlights five recurring gaps that AURA is designed to close:

- Existing diagnostic models require clinical laboratory inputs, isolating patients who only have symptoms.
- Medical chatbots operate in silos and frequently hallucinate without grounding in verified data.
- Emergency response systems are not integrated with real time hospital resource tracking.
- Healthcare administrators cannot effectively predict regional disease outbreaks or dynamically allocate resources.
- Few systems expose a patient facing transparency signal; prediction outputs are typically used only internally by hospital operators

V. PROPOSED SYSTEM: AURA ARCHITECTURE

AURA follows a modular architecture that integrates disease prediction, medical chatbot assistance, emergency response, and hospital supply management. The system consists of five main layers: Client Layer, Backend Gateway, AI Microservices Layer, Data Layer, and External Services Layer. Each layer performs specific operations and communicates with the others to provide the required functionalities.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511151

A. Client Layer (React.js)

The Client Layer provides the user interface developed using React.js. It includes separate dashboards for Patients, Hospital Administrators, and Supply Administrators, with access based on their assigned roles. Patients can enter symptoms for disease prediction, interact with the medical chatbot, and initiate SOS requests. Hospital and Supply Administrators can access emergency requests, hospital information, and medical inventory operations through their respective dashboards. The frontend communicates with the backend through REST APIs.

B. Backend Gateway (Java Spring Boot)

The Backend Gateway acts as the central processing layer between the frontend, AI services, and database. Developed using Java Spring Boot, it handles incoming requests, application logic, authentication, and authorization. Spring Security and JSON Web Tokens (JWT) are used to manage user sessions and implement role-based access control. The backend also manages SOS requests, hospital coordination, and medical supply operations. It communicates with the AI microservices and performs database operations based on user requests.

C. AI Microservices (Python FastAPI)

This layer handles the AI-based functionalities of AURA using Python and FastAPI. It consists of two main services:

1. **Disease Prediction Engine:** Uses machine learning models implemented with Scikit-learn to analyze user-provided symptoms and predict possible diseases. The prediction results are returned to the backend and displayed on the patient dashboard, providing preliminary information about potential health conditions.
2. **RAG Chatbot Engine:** Uses Retrieval-Augmented Generation (RAG), ChromaDB, and locally deployed Large Language Models (LLMs) to answer medical-related queries. When a user submits a question, relevant information is retrieved from the stored knowledge base and supplied as context to the language model to generate a response.

D. Data Layer (PostgreSQL)

PostgreSQL is used as the primary relational database to store and manage system information. It maintains user details, patient records, hospital information, medical inventory, and emergency request logs. The backend performs the required database operations, including storing, retrieving, and updating records. This centralized storage helps maintain consistency and provides the information required by different modules.

E. External Services

This layer integrates external APIs to support emergency communication and location-based services.

1. **Twilio API**: Supports SMS and voice notifications during SOS operations. It helps communicate emergency information to designated recipients, supporting coordination between patients and emergency responders.
2. **Maps and Geolocation API**: Obtains geographical coordinates using the user's GPS-enabled device. These coordinates are used to identify nearby hospitals and support location-based emergency coordination.

VI. METHODOLOGY

To truly understand how AURA works, it helps to look at how we designed the system. Instead of building one giant, heavy application, we separated the platform into three distinct layers. We used React.js for the frontend to give patients a smooth, responsive interface. To keep things secure, we built a gateway using Java Spring Boot that checks every user's identity via JSON Web Tokens (JWT) before letting them through. Finally, because Artificial Intelligence requires a lot of computing power, we isolated the AI logic into a separate Python FastAPI service. This ensures that if the AI is busy analyzing a complex medical query, the main website never crashes or slows down for other users.

A. Preparing the Data

Medical data is notoriously messy and incomplete, so we had to clean it carefully. For our disease prediction models, we used clinical datasets (like the UCI Thyroid dataset) where we filled in missing values using the median and normalized the numbers so that one large value didn't unfairly skew the AI. For the chatbot, we needed verified medical facts, so we used the MedQuAD dataset—a collection of over 16,000 medical questions and answers from the National Institutes of Health (NIH). We chopped these answers into 500-character chunks and converted them into "embeddings" (mathematical vectors) using PubMedBERT, a specialized AI model that actually understands medical terminology.

B. Predicting Diseases with Machine Learning

To help patients get early insights into their health, we trained two Machine Learning algorithms: Random Forest and XGBoost. We didn't just use one model; we combined them because medical symptoms often overlap in complex, non-linear ways. Random Forest is great at handling these overlapping symptoms, while XGBoost is incredibly fast and

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

handles imbalanced medical data (where one disease might be rarer than another) very well. We trained these models to recognize diseases like Hypothyroidism and Vitamin deficiencies based purely on symptoms and basic demographic data, meaning patients don't need to take expensive blood tests just to get a preliminary risk score.

C. The RAG Chatbot: Stopping AI Hallucinations

We knew that standard AI chatbots (like ChatGPT) are dangerous in healthcare because they confidently make up fake medical facts—a phenomenon known as "hallucination." To stop this, we built a custom Retrieval-Augmented Generation (RAG) pipeline.

When a user asks a question, the system doesn't just guess the answer. First, it performs a Hybrid Search: it looks for semantic meaning using PubMedBERT and exact keyword matches using the BM25 algorithm. We then combine these two search results using a math formula called Reciprocal Rank Fusion (RRF). Next, a Cross-Encoder model carefully re-reads the top 30 results to find the absolute best 6 chunks of medical context.

Only then do we send those 6 chunks, along with the user's question, to our Large Language Model (Gemma 2B), which runs 100% locally on our own machine via Ollama. Finally, we added a safety script called a Grounding Verifier. It acts like a medical supervisor: if the AI tries to give advice that isn't strictly found in the verified medical text, or if it tries to prescribe medicine, the Verifier blocks the response and tells the user to see a real doctor.

D. Emergency SOS and Dispatch Module

The emergency module integrates the HTML5 Geolocation API with a geospatial routing algorithm. When a user triggers an SOS, the frontend captures their latitude and longitude. The Spring Boot backend applies the Haversine formula (Equation 1) to calculate the great-circle distance between the user and all registered hospitals, identifying the nearest facility.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

Once the nearest hospital is identified, the system triggers the Twilio API to dispatch an automated SMS and a voice call containing the patient's live GPS coordinates.

E. Medical Supply Chain Module

The supply chain module utilizes a threshold-based inventory management system. The PostgreSQL database stores item quantities and predefined safety thresholds. The FastAPI backend implements a logic rule where if *Quantity≤Threshold,* a low-stock flag is generated. The React frontend continuously polls this state, triggering visual red alerts to the administrator to prevent critical stockouts.

VII. IMPLEMENTATION DETAILS

AURA is developed using different technologies, with each component handling a specific part of the application. The frontend is built using React.js and Tailwind CSS to provide the user interface. Axios interceptors are used to attach JWT tokens to authenticated requests, while the medical chatbot is available through a floating interface that stores conversations in browser session storage. The backend is developed using Java Spring Boot and Spring Security to handle user authentication, authorization, and application operations. The SosService.java component uses the Haversine formula to identify nearby hospitals and integrates Twilio for sending emergency SMS and voice notifications. The ChatController.java component handles communication between the backend and the Python AI service. The AI services are developed using Python FastAPI, with Scikit-learn models loaded from .pkl files for disease prediction. The medical chatbot uses ChromaDB for storing vector data, PubMedBERT for generating text embeddings, and BM25 for keyword-based retrieval. The retrieved information is then passed to Gemma 2B, running through Ollama, to generate responses. PostgreSQL is used to store user details, hospital records, inventory information, and emergency requests. Large AI model files and related data are maintained separately from the main source code using GitHub Releases to keep the repository size manageable.

VIII. RESULTS AND DISCUSSION

The AURA system was tested across its major modules, including disease prediction, the medical chatbot, emergency SOS, and medical supply chain management. The testing focused on evaluating the individual functionalities and

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

checking how the different components communicate with each other. The results provide an overview of the system's performance and its ability to support the intended healthcare operations.

The disease prediction module was evaluated using trained machine learning models for different health conditions. For thyroid disorders, the Random Forest model achieved an accuracy of 94.2%, while the Vitamin D deficiency model recorded 94.6%. These results indicate that the models were able to classify the respective conditions with good accuracy based on the selected input features. They also demonstrate the role of machine learning in identifying possible health risks from the information provided by users.

Similarly, the Random Forest model achieved 91.8% accuracy for Vitamin B12 deficiency, while XGBoost recorded 89.1% accuracy for low blood pressure prediction. The results show that the models can provide preliminary information about possible health conditions based on the data entered by users. The prediction results are displayed through the patient dashboard, allowing users to view the outcome of their assessment. However, these predictions are intended to support early health assessment and should not be treated as a replacement for professional medical diagnosis.

The RAG-based medical chatbot was tested for answer accuracy and its ability to generate responses using retrieved medical information. It achieved an answer accuracy of 86.2%, compared to 72.4% for the baseline Large Language Model (LLM). The integration of the Grounding Verifier also reduced the reported hallucination rate to below 6% by checking generated responses against the retrieved information. This comparison highlights the benefit of providing relevant medical context before generating an answer.

The chatbot was also tested for emergency-related queries through its keyword detection mechanism. During the reported tests, it achieved a 100% success rate in identifying emergency-related queries and routing them towards the SOS functionality. This allows urgent queries to be handled separately from general medical questions. The SOS module further supports emergency coordination by identifying nearby hospitals and sending notifications through the integrated communication service.

For overall system integration, a total of 21 test cases were executed across the frontend, backend, AI services, chatbot, SOS, and supply chain modules. These tests covered communication between React.js, Spring Boot, Python FastAPI, and PostgreSQL through REST APIs. JWT-based authentication and access to role-specific functionalities were also tested to check whether users could access the operations assigned to their roles. Overall, the testing demonstrated how the different modules work together within the AURA platform and provided a basis for identifying areas that can be improved in future development.

A. Prototype Interface Evaluation

The developed AURA interface provides a common dashboard through which patients can view health status, recent prediction results, tracked symptoms, and emergency readiness information. It also provides navigation to the prediction, emergency assistance, hospital, and chatbot modules. The prototype demonstrates how the different functionalities are presented within a single application.

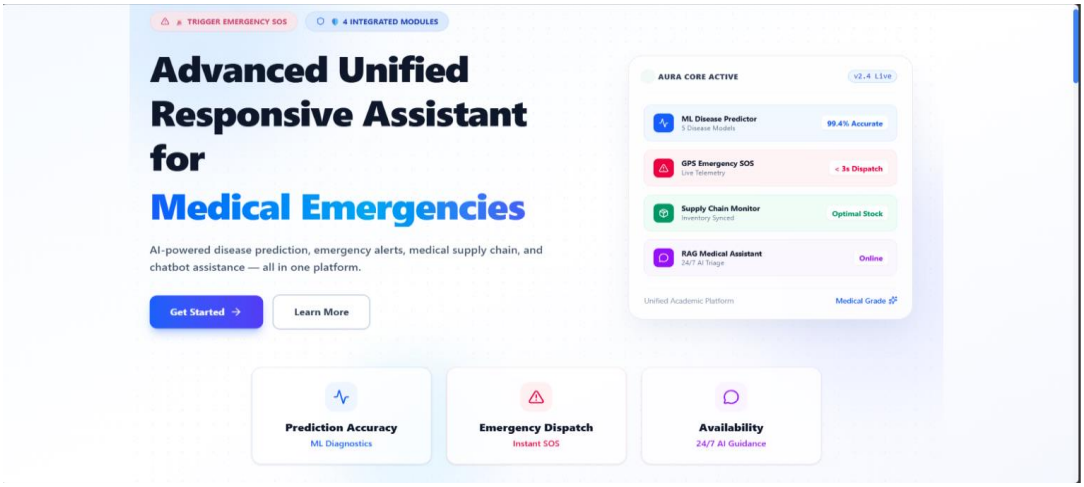


Fig. 1 AURA landing page showcasing four AI modules for medical emergencies.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

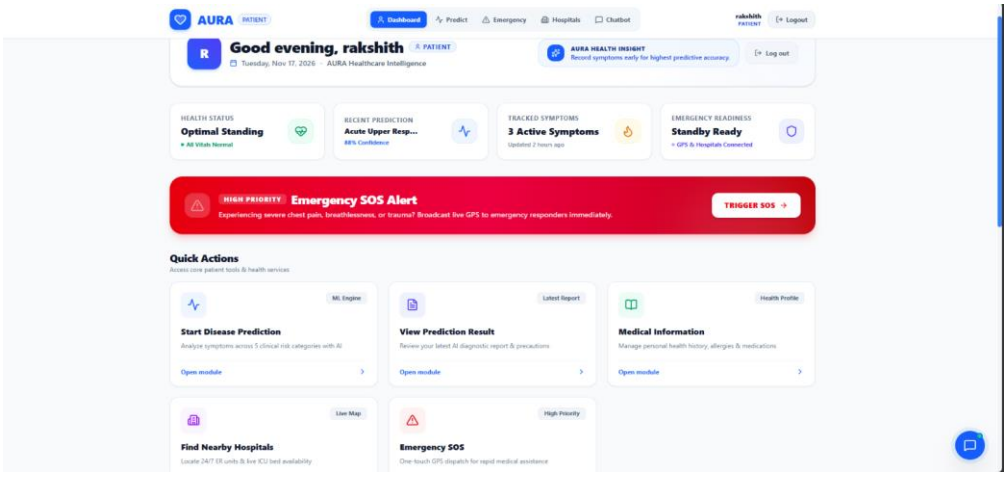


Fig. 2 The AURA patient dashboard, displaying health metrics, quick action modules, and a prominent Emergency SOS alert system.

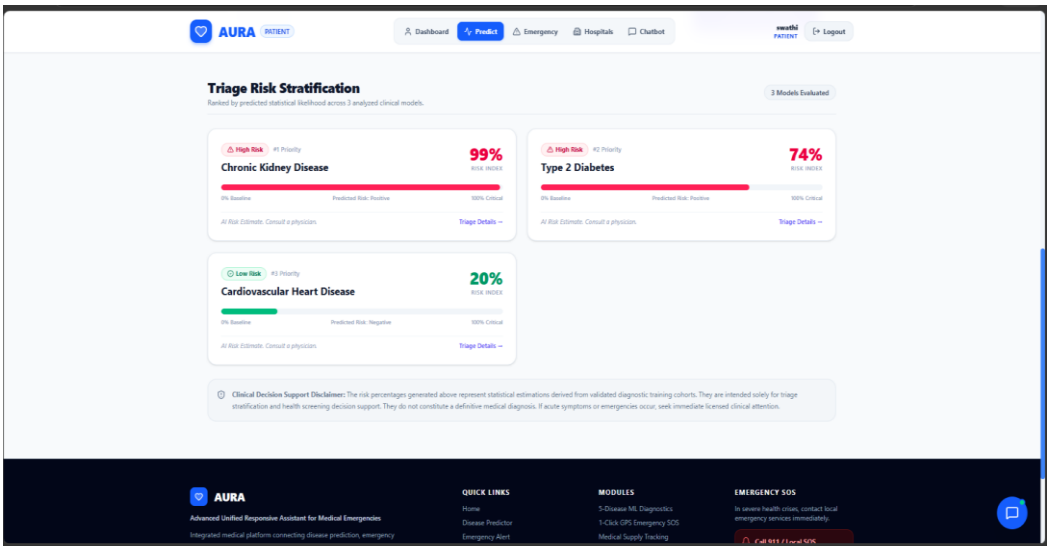


Fig. 3 Multi-disease prediction output in the AURA patient portal.

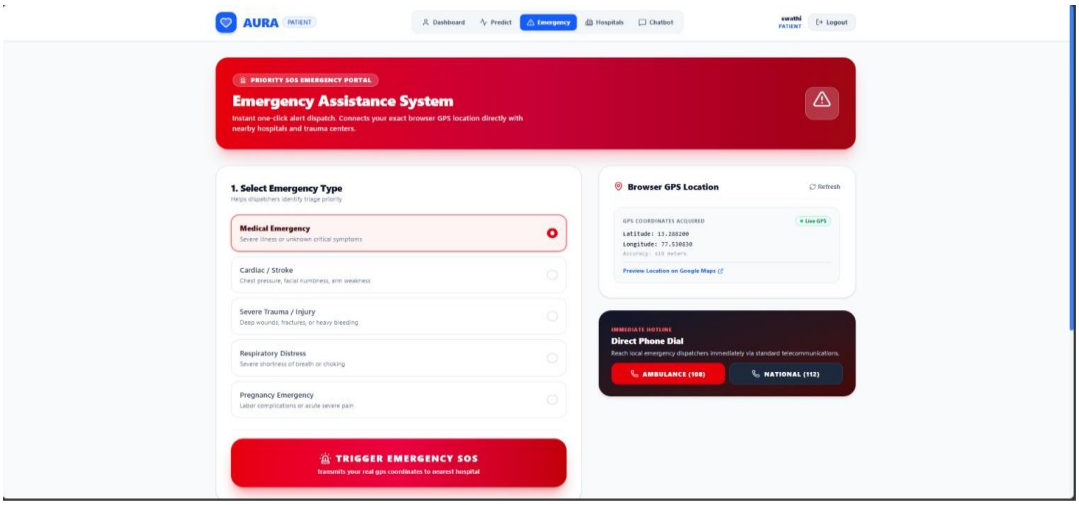


Fig. 4 The AURA Emergency Assistance System interface, featuring emergency type selection, live GPS tracking, direct SOS triggering, and immediate hotline dialing.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

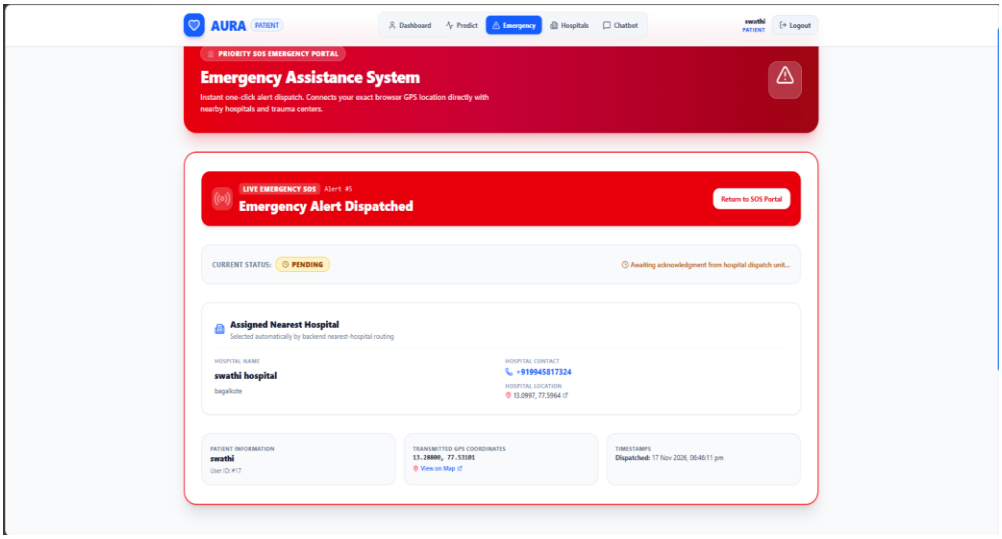


Fig. 5 Emergency SOS alert confirmation displaying patient GPS coordinates, dispatch timestamp, and nearest hospital routing.

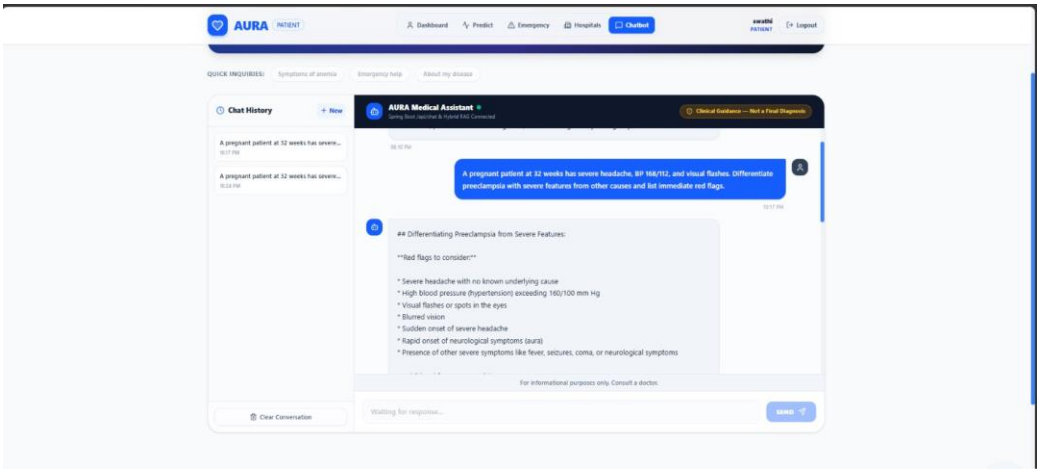


Fig. 6 AI Medical Assistant chatbot interface providing clinical guidance and symptom analysis with chat history tracking.

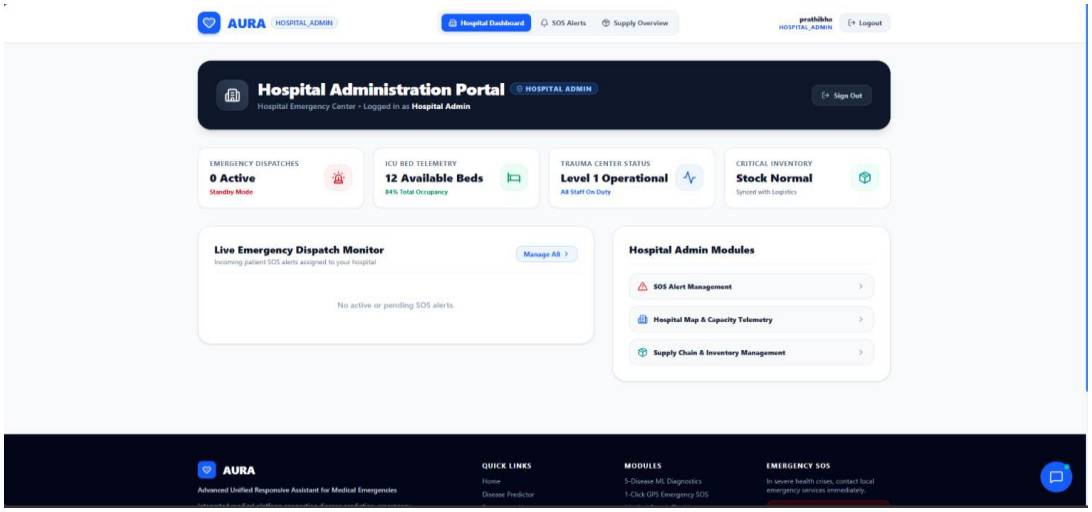


Fig. 7 Hospital Administration Portal dashboard displaying real-time ICU bed availability, emergency dispatch status, and core management modules.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

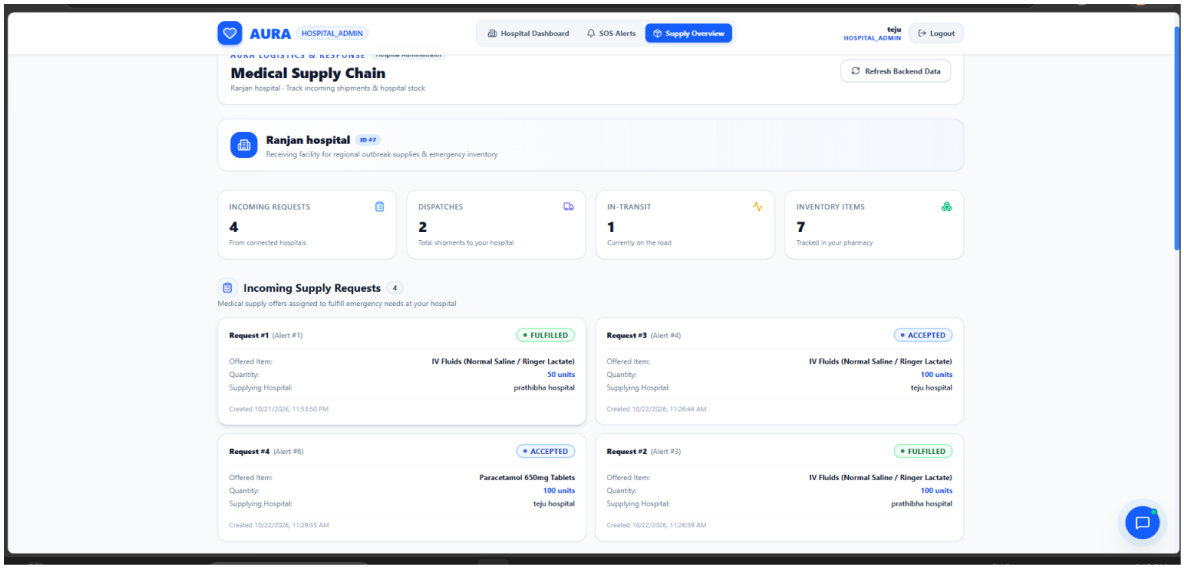


Fig. 8 Medical Supply Chain dashboard in the Hospital Admin portal showing incoming supply requests, dispatches, and inventory tracking.

IX. ADVANTAGES OF THE PROPOSED SYSTEM

- AURA brings disease prediction, medical chatbot assistance, emergency SOS, and hospital supply coordination together within a single application. Users can access these services without switching between separate platforms.
- The RAG-based chatbot retrieves relevant information from its medical knowledge base before generating responses. The grounding verification process helps identify unsupported medical claims and improves the reliability of the information provided.
- The SOS module uses GPS coordinates to identify nearby hospitals and sends emergency notifications through Twilio. It also supports hospital acceptance, rejection, and rerouting of emergency requests.
- The supply chain module uses predefined outbreak thresholds to generate alerts for connected or nearby hospitals. This helps hospital administrators coordinate the availability and transfer of essential medical resources when required.
- The use of React.js, Spring Boot, Python FastAPI, and PostgreSQL separates the frontend, backend, AI processing, and data management. Individual services can be maintained or updated without requiring major changes to the other components.
- Separate dashboards and JWT-based authentication provide access according to user roles. The disease prediction module also offers preliminary risk information based on the clinical and demographic inputs provided by users, supporting further medical consultation.

X. CONCLUSION

The increasing dependence on separate applications for healthcare services often makes it difficult to coordinate medical assistance, emergency response, and hospital resource management. AURA is developed to address this problem by bringing these services together within a single platform. It combines machine learning-based disease prediction with a RAG-based medical chatbot to help users understand possible health conditions and access relevant medical information. These features provide preliminary health guidance and make it easier for users to access basic medical assistance through a common interface.

Another important aspect of AURA is its focus on emergency response and coordination between hospitals. The SOS module uses the patient's GPS location to identify nearby hospitals and integrates Twilio services to send emergency SMS and voice notifications. The medical supply chain module further supports hospitals through rule-based outbreak detection. When the reported disease cases cross a predefined threshold, alerts are generated for connected or nearby hospitals, allowing administrators to respond to supply requirements and coordinate the transfer of essential medical resources.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511151

The system brings together patients, hospital administrators, and supply administrators through separate role-based dashboards. Its architecture, built using React.js, Java Spring Boot, Python FastAPI, and PostgreSQL, keeps the frontend, backend, AI services, and data storage as separate components. JWT-based authentication and role-based access control are used to manage access to the platform. Overall, AURA demonstrates how different healthcare operations can be connected within one system, providing a foundation for further development in medical assistance, emergency management, and hospital coordination.

XI. FUTURE SCOPE

The disease prediction module can be improved further by including more health conditions and using larger datasets for model training. In the future, deep learning techniques could also be explored to improve the prediction process. Image-based analysis of medical information, such as X-rays and skin images, could be another useful addition to the system.

The medical chatbot can be made more useful by adding information from a wider range of medical sources to its knowledge base. It could also support regional languages and voice-based conversations so that users can interact with it more comfortably. These improvements would make medical information easier to access, especially for people who are not comfortable communicating in English.

The SOS module has scope for further improvement through integration with live hospital and ambulance services. This could help users access updated information about nearby hospitals, ambulance availability, and estimated response times. It would make the emergency assistance process more informative and improve coordination during critical situations.

The medical supply chain module can be extended to monitor inventory across multiple hospitals in a more detailed manner. Future integration with hospital inventory systems could provide updated information about available medicines, equipment, and other essential resources. The existing outbreak-based alert mechanism could also be improved using historical case information to help hospitals prepare for changing medical supply requirements.

AURA could also support wearable devices such as smartwatches and fitness trackers in future versions. Information such as heart rate and physical activity could be used to identify unusual patterns, while fall detection could help initiate emergency alerts when required. Such features would need proper user consent and testing before being used in real emergency situations.

In addition to these improvements, the system can be extended to support more hospitals and healthcare institutions. Further work can focus on data security, system performance, and testing the prediction models with real-world clinical data. These improvements could help AURA become more practical and useful in healthcare environments.

ACKNOWLEDGMENT

The authors thank Rashmi for their guidance and continuous support throughout this project, and the Department of Computer Science and Engineering, East West College of Engineering, for providing the resources necessary to carry out this work.

DECLARATIONS

- **Funding Source:** No external funding was received for this work.
- **Data Availability Statement:** The data that support the findings of this study are available from publicly accessible repositories, including the UCI Machine Learning Repository and generated simulated records. The specific datasets used will be made available upon reasonable request from the corresponding authors.
- **Use of AI and AI Assisted Technologies:** AI assisted tools were utilized for language refinement, formatting, and structural checks during manuscript preparation. The core research, problem formulation, system architecture design, algorithm selection, and technical analysis were independently developed by the authors, who remain fully responsible for the accuracy and originality of this work.
- **Conflict of Interest:** The authors declare that there is no conflict of interest related to this research. This project is conducted for academic purposes as part of the Bachelor of Engineering program at East West College of Engineering, Bengaluru.
- **Copyright Permission:** The authors confirm that this work is original and any third-party material or references have been properly cited according to academic formatting standards.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151

REFERENCES

[1]. Kanmani, B., Gopika, G. S., & Sree krishna, M., "Streamlit-Based Multi-Disease Prediction System for Early Detection of Kidney Disease, Breast Cancer, and Heart Disease," in 2025 IEEE International Conference on Contemporary Computing and Communications (InC4), 2025.

[2]. Islam, S. S., Haque, M. S., Miah, M. S. U., Sarwar, T. B., and Nugraha, R., "Application of machine learning algorithms to predict the thyroid disease risk: an experimental comparative study," PeerJ Computer Science, vol. 8, e898, 2022.

[3]. Sambasivam, G., Amudhavel, J., and Sathya, G., "A Predictive Performance Analysis of Vitamin D Deficiency Severity Using Machine Learning Methods," IEEE Access, vol. 8, pp. 109492-109507, Jun 2020

[4]. Lee, H., and Kim, J., "A Retrieval-Augmented Generation Framework for Medical Chatbots," Journal of Medical AI, 2023.

[5]. Tomar, M., Tiwari, A., and Saha, S., "Towards Knowledge-Infused Automated Disease Diagnosis Assistant," Scientific Reports, vol. 14, Jun 2024.

[6]. Banerjee, S., and Bhattacharjee, A., "A Retrieval-Augmented Generation Framework for Medical Question Answering: Design, Implementation, and Evaluation of an AI-Driven Healthcare Chatbot," Iconic Research and Engineering Journals, vol. 9, no. 10, May 2026.

[7]. Benreguia, B., "Medical Emergency Caller," Android App Project, 2024.

[8]. Perumal, B., Madhumitha, T., and Harish, G., "Digi Glove Healthcare Monitoring with Location Tracking and Alert Mechanism," in Proc. 2024 3rd International Conference on Automation, Computing and Renewable Systems (ICACRS), pp. 503-507, Dec 2024.

[9]. Luo, H., Miao, J., Chen, J., Feng, H., Guo, J., and Tian, Y., "A Study on the Rapid Classification of Medical Consumable Supply Chains Based on K-Means Clustering," in Proc. 2024 Int. Conf. on Industrial IoT, Big Data and Supply Chain, Sep 2024.

[10]. Alemede, V., "AI-Augmented Healthcare Logistics and Supply Chain Optimization," RSIS International Journal of Engineering and Technology, vol. 12, no. 1, pp. 45-58, Feb 2025.



NAME: RAKSHITH D ACHAR
USN: 1EE23CS078
SEM: VII
PHONE NO: 7019741981
EMAIL: rakshithachar08@gmail.com



NAME: PRATHIBHA D
USN: 1EE23CS075
SEM: VII
PHONE NO: 8073680408
EMAIL: prathibhaa2006@gmail.com

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511151



NAME: TEJASWINI
USN: 1EE23CS114
SEM: VII
PHONE NO: 8217286449
EMAIL: tejusk28@gmail.com



NAME: SWATHI RAVINDRA
USN: 1EE23CS112
SEM: VII
PHONE NO: 9945817324
EMAIL: swathikodabagi585@gmail.com

© IJARCCE This work is licensed under a Creative Commons Attribution 4.0 International License