International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 4, 2025

MEDIQUICK – Optimized Emergency Bed allocation using AI and IoT

K Madhuravani[1], Eesha Karpuradu[2], Revadi Hema Satya Varshini[3]

[1]Assistant Professor, Department Of , Information Technology, Bhoj Reddy Engineering College For Women, India.

[2,3]B. Tech Students, Department Of , Information Technology, Bhoj Reddy Engineering College For Women, India.

eeshakarpuradu@gmail.com

ABSTRACT

MediQuick is an AI and IoT-powered system that simplifies emergency hospital bed booking. It enables patients to input symptoms through a web interface, which are analyzed using a Logistic Regression model to determine if the condition is an emergency. If confirmed, a Random Forest classifier further identifies the emergency category. The system then uses the patient's location to search for nearby hospitals with both available beds and doctors specialized in the detected condition. Once a suitable match is found, a temporary bed reservation is made to hold the spot. If the patient fails to arrive within the given time frame, the bed is automatically released for others in need. An IoT module, powered by a Force Sensing Resistor (FSR) connected to a NodeMCU, detects real-world patient presence on the bed to confirm physical admission. This allows seamless, automatic check-in without manual hospital intervention. Flask is used to build the backend server, while MySQL handles all hospital, bed, and patient data. Python's machine learning ecosystem powers the emergency prediction models. MediQuick enhances the responsiveness of emergency healthcare systems through intelligent automation and real-time decision-making. The system supports efficient triaging and optimizes resource usage in critical situations. It bridges the gap between patients and hospitals using smart technology. MediQuick ultimately delivers a scalable and life-saving solution for modern healthcare challenges.Traditional sentiment analysis tools often fall short by offering only basic polarity classification and failing to provide detailed understanding of specific customer concerns.

Keywords : *Emergency Healthcare, Bed booking system, AI in healthcare, IoT in hospitals, Logistic Regression, Random Forest Classifier, Symptom Classification, geolocation, Real-time admission, FSR Sensor, NodeMCU, Smart Triaging*

1. INTRODUCTION

In modern healthcare, especially during emergencies, timely access to hospital beds and specialist doctors is critical for saving lives. However, in many cases, patients or their families are forced to search manually for hospitals with available beds and the required medical facilities. This leads to delays, inefficiencies, and in worst cases, loss of life. Additionally, hospitals often lack real-time bed monitoring systems, which can result in inaccurate availability information being conveyed to incoming patients.

The proposed project, **"IoT-Based Smart Hospital Bed Assignment System"**, is developed to automate and optimize hospital resource allocation using an integrated web and IoT-based approach. It uses real-time sensor data to detect bed occupancy, a smart classification system to identify emergency cases, and intelligent routing of patients to the nearest suitable hospitals based on doctor availability and bed status. This system is divided into two key modules: the User Module and the Hospital Module, working together through a centralized database and API communication.

Existing System:

- Resource management in hospitals is mostly manual or relies on basic software, causing delays during emergencies.
- Families often waste critical time contacting multiple hospitals to find available beds or specialists.
- Existing hospital management systems are internal, lack public access, and have outdated or error-prone bed updates.
- There's no automated system to classify emergencies based on symptoms, leading to poor triaging and treatment delays.

Proposed System:

- The system uses AI and IoT to analyse symptoms, classify emergencies, and match patients with nearby hospitals that have available beds and doctors.
- It features real – time hospital dashboards and an IoT-enabled bed that updates occupancy automatically, creating a smart and seamless emergency response flow.

2. RELATED WORK

Title: *Mediquick – Optimized Emergency Bed Allocation using AI and IoT*

Reference: G. Sushma, B. Arundathi, and N. Lakshmi, ―Real-Time Hospital Bed Information System During Pandemic Situation,‖ 2023 7th

International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2023.
Authors: G. Sushma, B. Arundathi, N. Lakshmi

SUMMARY: MediQuick builds upon the real-time hospital bed monitoring system proposed in [1], which uses sensors and microcontrollers to track bed occupancy. While
[1] focuses on internal hospital use, MediQuick extends this by integrating patient-side emergency detection using AI [3] and enabling intelligent bed and doctor assignment [5]. The system updates bed status in real time using an FSR sensor, similar to [1], but also empowers users to initiate the booking process through a web interface—bridging the gap between hospital automation and patient accessibility.

Literature on IoT-Based Bed Monitoring:
Title: *IoT-Based Smart Hospital Bed Monitoring System*
Reference: R. Sharma, A. Mehta, and N.Joshi, ―IoT-Based Smart Hospital Bed Monitoring System,‖ International Journal of Engineering Research & Technology (IJERT)
Authors: R. Sharma, A. Mehta, N. Joshi

SUMMARY: MediQuick incorporates IoT concepts similar to those in [2] and [4], which use sensors and microcontrollers like ESP8266 for smart bed monitoring. These systems focus on automating bed occupancy detection and transmitting data to hospital systems. In MediQuick, we adopt a similar approach using an FSR sensor and NodeMCU to detect real-time occupancy. However, unlike [2] and [4], which are primarily confined to internal monitoring, MediQuick connects this data to a web platform for patient interaction and booking. This fusion of IoT with user-facing functionality sets MediQuick apart as a more holistic emergency response system.

Literature on Emergency Detection Using AI:
Title: *AI-Based Health Emergency Detection Using Symptom Classifiers*
Reference: Desai, S. Patel, ―AI-Based Health Emergency Detection Using Symptom Classifiers,‖ IEEE International Conference on Healthcare Informatics, 2023.
Authors: M. Desai, S. Patel

SUMMARY: The emergency detection system in MediQuick takes inspiration from AI- based approaches like those in [3], which classify health emergencies based on user symptoms. While [3] focuses on building accurate classifiers for critical conditions, MediQuick adapts this idea into a practical web application. Using machine learning models like Logistic Regression and Random Forest, we identify both the presence and type of emergency in real time. Unlike [3], which operates in isolated environments, our system links AI predictions to immediate hospital resource allocation [5], making it actionable and life-saving.

3. REQUIREMENT ANALYSIS

Functional Requirements:

- *User Registration and Login*
  The system must allow users (patients) to create accounts, log in securely, and manage their profiles.
- *Symptom Input and Emergency Classification*
  Users can enter their symptoms via text input. The system must classify whether the entered symptoms represent an emergency using rule-based logic or AI symptom mapping.
- *Hospital and Doctor availability search*
  The system must compare the classified emergency condition with the database of available hospitals and identify those with both available beds and required specialists.

Non-Functional Requirements:
Non-functional requirements define the system's quality attributes and operational constraints that ensure it performs efficiently, securely, and reliably. The proposed system must meet the following non-functional requirements:

- ***Performance***
  The system should respond to user actions such as symptom input, hospital search, and status updates within 2–3 seconds. The IoT sensor should send updated bed status to the backend within 5 seconds of detecting a change.
- ***Scalability***
  The system should be designed to handle multiple hospitals, doctors, and patient requests without performance degradation. While the prototype uses only one IoT- enabled bed, the backend and database structure support future scaling to multiple IoT-connected beds and hospitals.
- ***Availability***
  The system should be available on a local Wi-Fi network for continuous monitoring and interaction. In future deployments, it should support 24x7 availability via a secure web-hosted server.

Software Requirements:

- Backend: Flask (Python), Flask-CORS, pymysql
- Database: MySQL
- Frontend: HTML, CSS, JavaScript
- IoT Integration: Data collection from force sensing resistors
- insights.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 4, 2025

Hardware Requirements:

- Processor: Intel i3 or higher
- RAM: 8GB or more
- Storage: 500GB HDD or SSD
- IoT Sensors: Force sensing resistors for bed occupancy tracking

4. DESIGN

System Architecture:

The system architecture of MediQuick is designed to integrate AI, web technologies, databases, and IoT into a cohesive emergency response platform. The frontend, built with HTML, CSS, and JavaScript, allows users to enter personal details and symptoms, while also providing hospital administrators with a dashboard to monitor bed and doctor availability. The backend, developed using Flask, processes user input, runs emergency classification using a Logistic Regression model, and identifies the emergency category using a Random Forest classifier. It then matches the patient with the nearest hospital that has both an available bed and the required specialist. Data is managed through a MySQL database that stores patient, hospital, bed, and doctor information. An IoT component, consisting of a Force Sensing Resistor (FSR) connected to a NodeMCU ESP8266, detects real-time bed occupancy and updates the backend via WiFi. This modular and scalable architecture enables smart, real-time hospital-patient coordination, enhancing both efficiency and response time during emergencies.

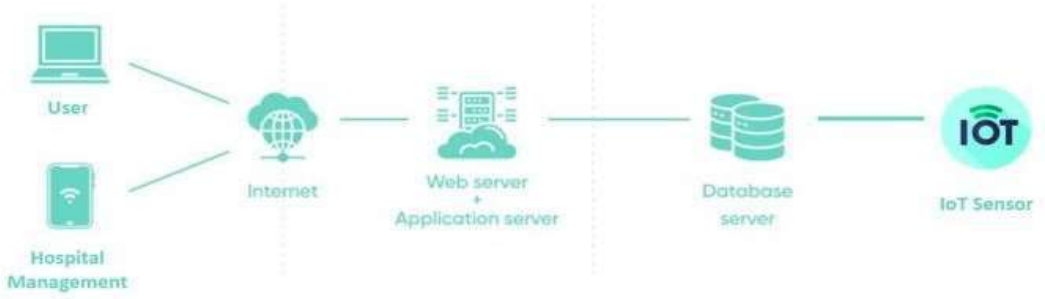


Fig. 1 System Architecture

Technical Architecture:

Technical Architecture refers to the structural process of designing and building the technological backbone of the Mediquick application with a strong focus on usability, flexibility, and integration. MediQuick's technical architecture is a multi-layered system that combines machine learning, IoT, and web technologies for intelligent emergency healthcare management. The frontend is developed using HTML, CSS, and JavaScript to enable user interaction and hospital dashboard access. The backend is powered by Flask, which acts as the central controller for handling user requests, running ML inference, and managing database operations. Two pre-trained models—Logistic Regression for emergency detection and Random Forest for emergency type classification—are integrated using joblib. The system relies on a MySQL database for storing real-time information on hospitals, doctors, beds, and patient bookings. For physical bed monitoring, a NodeMCU ESP8266 microcontroller with an FSR sensor is used to detect bed occupancy and send updates over WiFi to the Flask backend. This real-time data flow ensures accurate bed tracking and efficient resource allocation, creating a seamless and responsive healthcare experience.

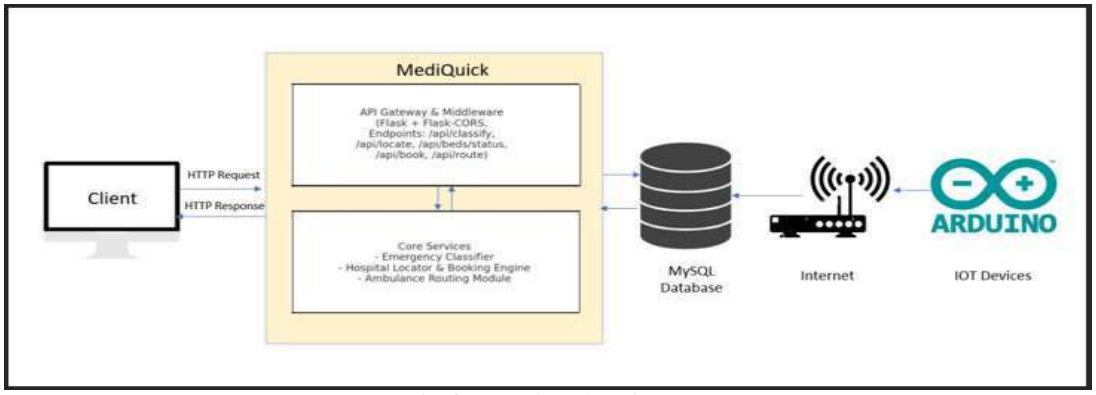


Fig. 2 Technical Architecture

International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 4, 2025

Methodology:

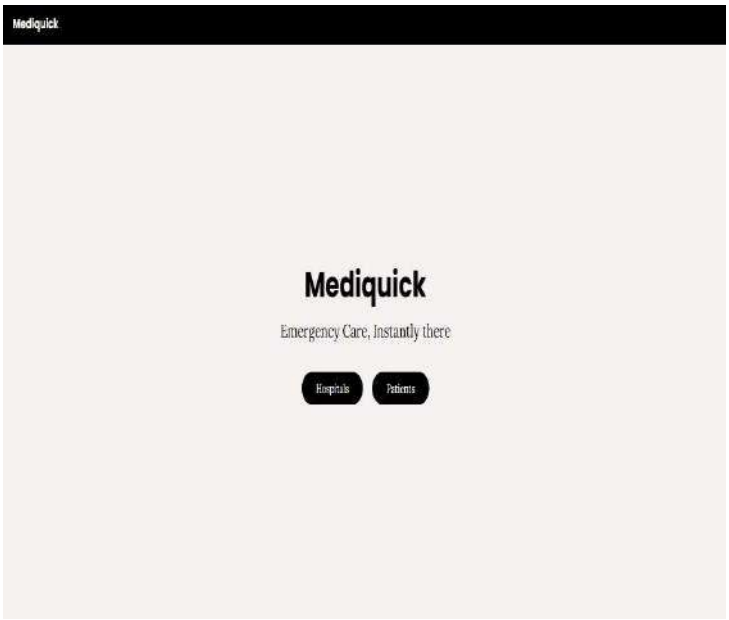
The development of MediQuick followed a modular and iterative methodology, starting with problem identification and requirement analysis focused on delays in emergency healthcare access. A rule-based machine learning approach was implemented, where user symptoms are classified using a Logistic Regression model to detect emergencies and a Random Forest classifier to determine the specific emergency category. Based on this, the system dynamically searches a MySQL database for nearby hospitals with available beds and matching specialists using geolocation logic called Harversine Formula. The web interface was developed using HTML, CSS, and JavaScript for intuitive interaction, while Flask was used as the backend framework to manage routing, API endpoints, and communication between modules. IoT integration was achieved using a NodeMCU ESP8266 and Force Sensing Resistor (FSR) to simulate bed occupancy and automatically update the backend.

5. IMPLEMENTATION

Libraries

- **Flask -** It is a lightweight web framework used to build the backend of MediQuick. It handles API routes, connects the frontend to machine learning models, and manages communication with the database. Its simplicity and flexibility make it ideal for rapid prototyping and RESTful API development.
- **Numpy** - It is a general-purpose array-processing package. It provides a high- performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various features including these important ones: A powerful N-dimensional array object, Sophisticated (broadcasting) functions, Tools for integrating C/C++ and Fortran code.
- **Pandas** - It is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze.
- **Scikit-learn** - It is used to implement and train the machine learning models in MediQuick. Logistic Regression and Random Forest Classifier from this library are used for emergency detection and category classification. It provides easy-to-use tools for model training, prediction, and evaluation.
- **Threading** - The threading module allows parts of the code to run in the background without blocking the main Flask app. In MediQuick, Thread is used to handle timeout logic in parallel — for example, when a bed is reserved, a separate thread can wait for a few minutes and then auto-release the bed if the patient hasn't arrived, all while the main app keeps running

6. SCREENSHOTS

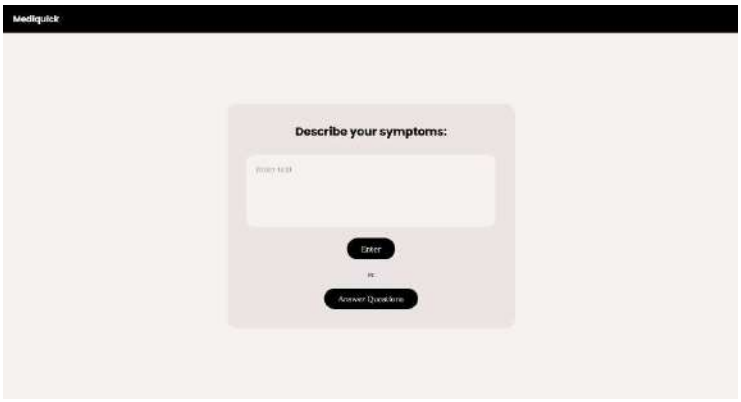


1. **Home Page**

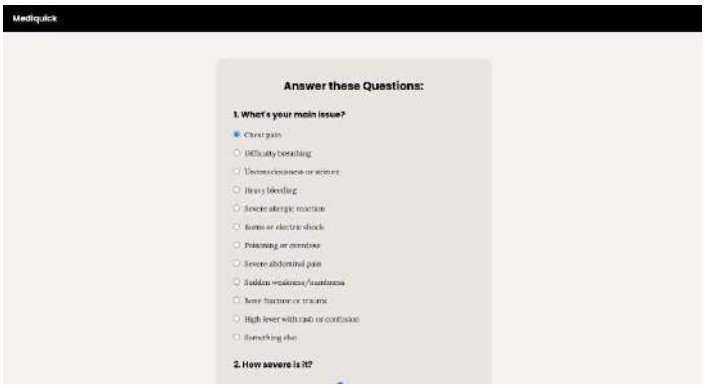
International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

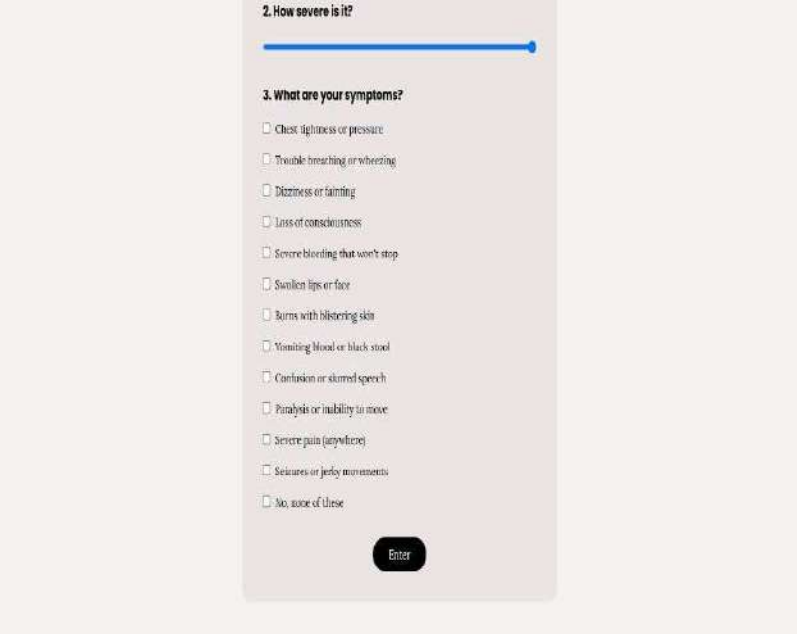
Volume 13, Issue 4, 2025



2. Text Input Area



3. Questionnaire

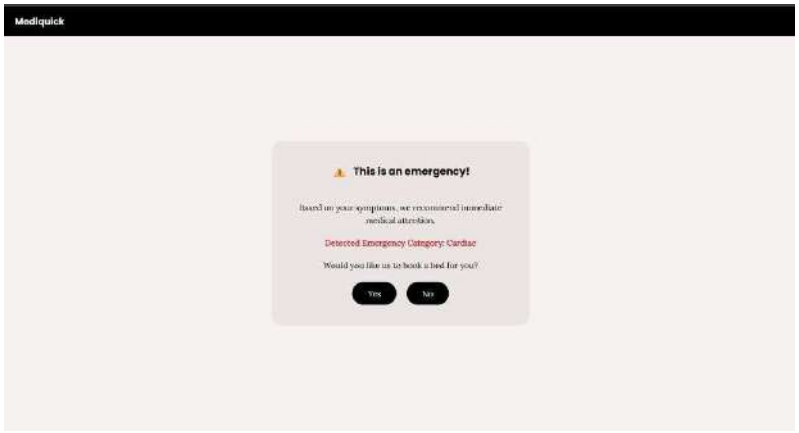


International Journal of Information Technology & Computer Engineering

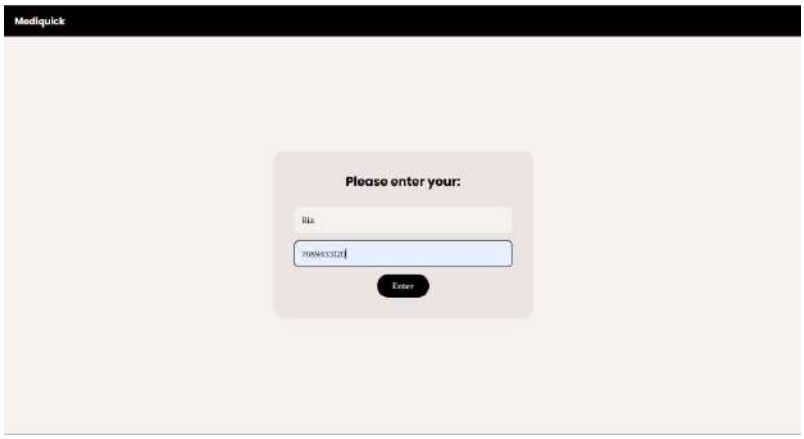
ISSN 2347–3657

Volume 13, Issue 4, 2025

4. Questionnaire



5. Emergency



6. Enter patient details

7. CONCLUSION

The proposed system, Mediquick, successfully integrates Internet of Things (IoT) and intelligent rule-based logic to provide an optimized solution for emergency hospital bed assignment. By enabling users to input symptoms and automatically classify whether the situation is an emergency, the system facilitates faster medical intervention. The backend logic identifies the nearest hospital with both available beds and appropriate specialists, thus improving the efficiency and accuracy of patient-to-hospital mapping. Furthermore, the integration of a Force Sensing Resistor (FSR) sensor with a NodeMCU microcontroller ensures real-time tracking of bed occupancy status. The sensor system automatically updates the hospital's database without manual intervention. A manual discharge mechanism is included in the hospital dashboard to reset bed availability securely. This combination of automated and manual control provides a reliable and responsive solution within a student-level budget, demonstrating the potential for scalable smart healthcare systems. Overall, the system reduces delay in critical situations, supports hospitals in managing resources effectively, and enhances patient experience during emergency medical needs.

8. REFERENCES

[1] G. Sushma, B. Arundathi, and N. Lakshmi, **"Real-Time Hospital Bed Information System During Pandemic Situation,"** 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2021.

[2] R. Sharma, A. Mehta, and N. Joshi, **"IoT-Based Smart Hospital Bed Monitoring System,"**

International Journal of Engineering Research & Technology (IJERT).

[3] M. Desai, S. Patel, **"AI-Based Health Emergency Detection Using Symptom Classifiers,"** IEEE International Conference on Healthcare Informatics, 2023.

[4] S. Bhatnagar et al., **"Smart Healthcare Monitoring Using ESP8266 and Sensors,"** International Journal of Advanced Research in Computer Science, vol. 14, no. 2, 2023.

[5] N. Kulkarni and T. Lakshmi, **"Hospital Resource Allocation Portal for Emergency Beds and Doctor Assignment,"** ACM Digital Library, 2023.

[6] Chaitra H , Sanjay , Jayaprakash G T, Magadi Achyutha, Arunagoud Gokul "Advanced IOT Technology And Machine Learning Techniques For Monitoring Waterlogging In Underpass", *IJMEC*, vol. 10, no. 4, pp. 1–6, Apr. 2025, Accessed: Oct. 30, 2025. [Online]. Available: https://ijmec.com/index.php/multidisciplinary/article/view/582