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 2, February 2026

DOI: 10.17148/IJARCCE.2026.15249

HopeStream – Intelligent Hospital Queue Management System with Priority-Based Scheduling

MATTHEWLYNN M[1], Dr. C. DANIEL NESAKUMAR[2]

Student, Department of Computer Applications, Sri Ramakrishna College of Arts and Science (Autonomous),

Coimbatore – 641006, Tamil Nadu, India[1]

Assistant Professor, Department of Computer Applications,

Sri Ramakrishna College of Arts and Science (Autonomous), Coimbatore – 641006, Tamil Nadu, India[2]

Abstract: The problem becomes clear when you enter a government hospital that experiences heavy traffic on Mondays. The issue exists since long ago but continues to create problems. The majority of hospitals use either paper registers or basic digital token dispensers which operate on a first-come, first-served basis. The two methods fail to handle urgent medical needs which leads to actual patient care delays and creates unnecessary waiting periods for essential medical treatment.

HopeStream was developed to fill this requirement. The software functions as a complete web-based hospital queue management system which unifies patient registration with triage-based priority classification and intelligent scheduling into a single system. Rather than assigning tokens in the order patients arrive, HopeStream asks a few targeted questions — body temperature, reported symptoms, and a short checklist of emergency indicators — and uses that information to assign each patient a priority level of High, Medium, or Low. The system performs this task without requiring the receptionist to decide about patient cases.

The scheduling logic uses a weighted round-robin algorithm which maintains a 2:1 rule that High priority patients receive treatment after two High priority patients have been treated. The simple rule enables urgent cases to advance while still allowing doctors to treat patients who scheduled routine check-ups. The system operates department-wise to generate tokens while displaying a waiting-area display that updates every five seconds which includes a doctor panel and automated visit history functions that developers built using Python Flask and SQLite and Bootstrap 5.

Keywords: Hospital Queue Management, Priority-Based Scheduling, Weighted Round-Robin Algorithm, Triage Classification, Python Flask, SQLite, Real-Time Display, Role-Based Access Control

I. INTRODUCTION

Emergency departments serve as the most intricate operational spaces found within hospital buildings. The hospital must handle patient arrivals who present with various medical situations while moving through their waiting area at different speeds throughout the day. The administrative side of this process, particularly queue management, is often treated as a secondary concern. Yet it is precisely where many of the most frustrating and consequential inefficiencies occur.

The typical approach in most hospitals, especially smaller ones, still relies heavily on manual work. The reception staff performs their duties by jotting down patient names, providing paper tokens and then calling patients according to their arrival sequence. The digital token dispensers now used in some modern facilities automatically create number sequences through their digital system yet follow the same basic principle that first arrivals receive first service. The system lacks a way to determine whether the person who should be served next has a simple headache or a serious chest pain emergency. The system requires all patients to wait their turn which results in equal treatment for everyone who came in.

High patient volumes create serious operational challenges for hospitals. Critically ill patients must face long waiting times during peak hours because multiple routine patients arrived before them. Priority queue systems that send urgent cases to treatment first create an alternative issue because low-priority patients experience prolonged waiting times when urgent cases keep reaching the system. The two situations which exist both fail to meet requirements.

DOI: 10.17148/IJARCCE.2026.15249

The project introduces HopeStream which functions as an intelligent hospital queue management system that addresses this operational conflict. The system streamlines patient registration while it uses standardized triage data to determine patient priority and it employs a weighted round-robin system to organize queues based on both urgent cases and equitable treatment. The system directs patients to their designated medical departments which include General Medicine, Paediatrics, and Cardiology and describes their respective department tokens. A live TV display keeps patients informed while doctors interact with the queue through a secure panel. The system uses Python Flask for its backend development and SQLite for its database and Bootstrap 5 provides a responsive frontend design.

II. OBJECTIVE

The central objective of this work is to replace the manual, inefficient, and inequitable processes that govern patient queuing in most hospitals with an automated system that is smarter, fairer, and easier for staff to use on a daily basis.

The project will create a system which registers new patients and retrieves existing patient records while preventing duplicate records and uses actual triage data for assigning priority levels and establishes queue sequences based on urgency and fairness requirements. The system should support multiple hospital departments simultaneously, generate meaningful tokens that direct patients to the right place, and give doctors everything they need to handle consultations efficiently — all from a single web interface.

This project aims to demonstrate through its technical achievements that intelligent scheduling algorithms can be implemented practically in real-world healthcare settings using accessible web technologies. The tools used here — Flask, SQLite, Bootstrap — are not exotic or expensive. The barrier to building something meaningfully better than a paper token system is lower than it might appear, and HopeStream is an attempt to make that argument concretely.

III. EXISTING SYSTEM

Understanding the improvements of HopeStream requires an honest evaluation of current operational procedures. Hospitals that treat many patients while dealing with budget constraints typically use two main methods for queue management which they sometimes need to combine imperfectly.

Reception staff perform manual operations by keeping registers and notebooks to record patient arrivals who they will call in the order of their arrival. The system lacks a formal structure which results in no patient deduplication and requires searching through old registers to find patients who visited months ago. Basic digital token dispensers function as the second method by using machines or software to issue numbered tickets while showing the currently active ticket number. The systems decrease desk confusion through their operation but they cannot solve the core problem of deciding which patient should receive service next.

The two methods fail to consider the urgent medical needs of patients. The system treats both a mild cough and a suspected cardiac event as two medical conditions requiring the same routine administrative handling. Patients who wait on the queue system cannot see real-time updates about their current standing in line. Patients who return for follow-up appointments cannot access their previous medical visit records through the system. The system has no user role definitions because anyone who can access the reception desk in various settings already has complete access to patient records for both viewing and editing purposes.

IV. METHODOLOGY

The development of HopeStream followed a method that built the system through incremental progress and modular construction. The system developers created testing procedures to evaluate each functional component registration triage scheduling doctor panel TV display and visit history before they combined all functions into the finished system. The method enabled the team to detect and resolve problems during the early development stage while building the final product from verified individual components.

The system operates through a configuration that uses three distinct tiers. The frontend system handles all user interactions through three components which include registration forms for reception staff and a consultation panel for doctors and a public TV display for the waiting area. The backend built with Python Flask controls all application functions which include authentication and session management and queue processes and the scheduling system. The database system uses SQLite to store patient records and queue entries and visit history and department configuration and user credentials information.

Patient Registration and Lookup
The receptionist begins the registration process by filling out a short registration form which includes the patient's name and phone number and blood group and address and emergency contact information. The system uses phone numbers as its main identification method which checks for duplicate records before adding new entries to avoid creating duplicate records. Patients can have their records accessed by using only their phone number which allows staff to retrieve their entire medical history within two seconds. The system first performs a lookup which allows users to access information without reentering it while maintaining accurate data between different visits.

Triage and Priority Assignment
The process of adding a patient to the queue requires more steps than selecting a department. The receptionist documents the patient temperature and their symptoms before she starts to complete the emergency checklist which includes chest pain or breathing difficulty and severe bleeding or injury and unconsciousness or disorientation and severe pain that reaches 8 or higher and symptoms that have continued for more than three days. The system uses these inputs to create an automatic priority assessment. The patient receives High classification when critical flags are checked or their temperature exceeds 39°C. The system assigns medium status when the patient temperature falls between 37.5°C and 39°C and there are no emergency flags present. The system assigns Low status to all remaining situations. The system operates through automated logic which eliminates the need for manual control while it maintains consistent results and lessens the need for reception staff to make decisions.

Weighted Round-Robin Scheduling
The scheduling algorithm partitions waiting patients in the department into two distinct categories which are High priority and Medium or Low combined and it uses a 2:1 rule to establish their treatment sequence. The queue table stores this sequence as a sequence order field which updates whenever new patients enter or existing patients exit the system. The system creates a queue that advances urgent patients while preventing permanent delays for lower priority patients which neither strict FIFO nor pure priority queuing methods can accomplish.

Token Generation and Department Management
The system creates tokens which are distributed to each department on a daily basis. The General Medicine department provides patients with GM-1 and GM-2 and additional designations. The Paediatrics department uses designations which start with PED to identify its patients. The system identifies the current highest token for the department and increases that value by one; at midnight, the count begins again. The hospital system allows new departments to be added to the departments table without needing to alter application code which enables ongoing system maintenance as the hospital expands.

System Architecture Overview

Layer	Technology	Key Responsibilities
Presentation Layer	HTML, CSS, Bootstrap 5, JavaScript	Reception forms, doctor panel, TV display, client-side validation
Application Layer	Python Flask	Authentication, session control, scheduling algorithm, route management
Database Layer	SQLite	Patient records, queue state, visit history, user credentials, departments

Table 1: Three-Tier System Architecture

Doctor Panel and Visit History
Doctors work through a role-restricted panel showing the current patient's token, name, blood group, temperature, symptoms, priority, and department. Clicking 'Call Next Patient' simultaneously writes the consultation to the visits table and removes the patient from the queue, triggering a re-sequencing of remaining patients. Doctors can also search visit history by phone number to review a patient's prior records during the same appointment.

TV Display and Security
The waiting area display is a public page showing the current token and the next one, with patient name, department code, and priority visible. A JavaScript snippet reloads the page every five seconds, keeping the display current without requiring websocket infrastructure. All other routes are protected by session-based authentication. Role checks on

sensitive routes return a 403 for unauthorized access attempts. All database operations use parameterized queries throughout to prevent SQL injection.

V. RESULT AND DISCUSSION

The testing process included four testing types which consisted of unit tests for single modules and system tests which checked complete workflows and performance tests which evaluated system capacity under multiple users and validation tests which examined priority and scheduling functions. All test cases passed. The subsequent discussion centers around the findings which will determine if the system meets its predetermined performance criteria.

The main test verified whether the weighted round-robin algorithm functioned properly with various queue configurations. It did. The system correctly placed Medium and Low priority patients at every third position after five High priority patients were consecutively added. The specific case of three High, one Medium, and one Low priority patients produced the sequence High, High, Medium, High, High, Low — precisely as the algorithm is designed to produce. This pattern remained consistent throughout all multiple test runs which were conducted using different department arrangements.

All test inputs led to accurate priority classifications. A patient presenting with a temperature of 39.5°C and chest pain was correctly assigned High priority. A temperature of 38°C with no emergency indicators gave Medium. The combination of normal temperature and mild symptoms resulted in Low. The logic is deterministic — the same inputs always yield the same result — which is important for staff trust. The receptionist will receive identical priority results when entering the same information on two separate occasions.

Token generation worked correctly across departments. The main medical department and the paediatric department each operated their own token sequences which advanced throughout the day before returning to their starting point at midnight. The token counts of both departments remained unaffected by simultaneous additions to different departments. All registration and queue operations completed within two seconds in sequential testing. The doctor panel loading time reached three seconds when two users tried to access it at the same time. Visit history searches took less than one second to complete.

User acceptance testing identified multiple functions which did not perform as expected. The TV display initially showed no department code which resulted in patients not knowing their designated department when their token was displayed. The issue was fixed. A search function was added to visit history after feedback indicated that scrolling through all records was impractical in a real clinical setting. Duplicate phone number detection was tightened after a test created an unintended duplicate. The doctor panel was also redesigned to give symptoms greater visual prominence.

The combined results demonstrate that a lightweight and accessible system achieves significantly better queue management than current hospital systems. The 2:1 rule in particular addresses a problem that is hard to solve with simpler approaches — it prevents patient starvation under sustained high-priority demand without sacrificing urgency-based ordering. The project outcome demonstrated the implementation process required only a few dozen lines of Python code which stands out as one of the more fascinating project results.

Validation Test Results

Scenario	Input Conditions	Expected Output	Result
High priority – vitals	Temp 39.5°C, chest pain flagged	Priority = High	Pass
High priority – emergency	Severe bleeding checkbox selected	Priority = High	Pass
Medium priority	Temp 38.0°C, no emergency flags	Priority = Medium	Pass
Low priority	Normal temperature, mild symptoms	Priority = Low	Pass
2:1 scheduling rule	3 High, 1 Medium, 1 Low in queue	Sequence: H, H, M, H, H, L	Pass

DOI: 10.17148/IJARCCE.2026.15249

VI. CONCLUSION

Queue management in hospitals sounds like a minor administrative concern until you are the patient sitting in a crowded waiting room, unsure of when you will be seen or whether anyone has noticed that you are in more pain than your token number suggests. The staff members experience a different kind of stress because they must handle a system which lacks built-in mechanisms for urgent situations and fair processes.

HopeStream was built with both of those realities in mind. The system does not attempt to resolve each administrative issue which hospitals face. The system improves queue management through intelligent queue handling which results in faster emergency response times and better patient treatment and reduced mental workload for front-desk employees. The weighted round-robin algorithm is, in some ways, the simplest part of the system. The rule establishes a boundary which protects patients from waiting endlessly while it ensures that urgent situations receive proper attention. The hospital system depends on registration and triage and token generation and live display and doctor panel and visit history and role-based access to create operational logic which maintains hospital functions.

The system proved to function correctly during testing, which included testing of priority assignment, scheduling, token generation, access control, and concurrent performance testing. User acceptance testing provided feedback which resulted in system improvements that brought the system closer to hospital staff operational needs. HopeStream functions as a complete hospital queue management system which uses open-source tools to create a fair operating system. The system creates clear paths of development which include SMS notifications and appointment booking and mobile patient access and cloud deployment.

REFERENCES

[1]. Grinberg, M., Flask Web Development: Developing Web Applications with Python, O'Reilly Media, 2nd Edition, 2018.
[2]. Owens, M. and Allen, G., The Definitive Guide to SQLite, Apress, 2nd Edition, 2010.
[3]. Lakshmi, C. and Iyer, S.A., "Application of queueing theory in health care: A literature review," Operations Research for Health Care, vol. 2, no. 1-2, pp. 25-39, 2013.
[4]. Soremekun, O.A., Takayesu, J.K. and Bohan, S.J., "Framework for analyzing wait times and patient flows in emergency departments," Journal of Emergency Medicine, vol. 41, no. 6, pp. 710-716, 2011.
[5]. Bhattacharjee, P. and Ray, P.K., "Simulation modelling and analysis of appointment system performance for multiple classes of patients in a hospital: A case study," Operations Research for Health Care, vol. 8, pp. 71-84, 2016.