## A PROJECT REPORT ON

Queuing Management System for Clinic

SUBMITTED TO

MIT SCHOOL OF COMPUTING, LONI, PUNE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

## BACHELOR OF TECHNOLOGY (INFORMATION TECHNOLOGY)

**BY**

**Anurag Bodkhe - ADT23SOCB1509**



**DEPARTMENT OF INFORMATION TECHNOLOGY MIT School OF COMPUTING**

## MIT Art, Design and Technology University Rajbaug Campus, Loni-Kalbhor, Pune 412201 2024-25



**MIT SCHOOL OF COMPUTING DEPARTMENT OF INFORMATION TECHNOLOGY** MIT ART, DESIGN AND TECHNOLOGY UNIVERSITY, RAJBAUG CAMPUS, LONI-KALBHOR, PUNE 412201

# CERTIFICATE

This is to certify that the project report entitled

# Queuing Management System for Clinic

Submitted by

**Anurag Bodkhe - ADT23SOCB1509**

is a Bonafide work carried out by them under the supervision of Prof. Swati Singh and it is submitted towards the partial fulfillment of the requirement of MIT ADT university, Pune for the award of the degree of Bachelor of Technology (Information Technology)

### Prof.Swati Singh Dr.Prashant Doetre

Guide Head of Department

### Dr. Vipul Dalal Dr.Rajneeshkaur Sachdeo

Director Dean

Seal/Stamp of the College Place: PUNE

Date:

# CERTIFICATE

This is to certify that the Project report entitled

# Queuing Management System In Hospital

Submitted by

**Anurag Bodkhe - ADT23SOCB1509**

is a bonafide work carried out by him/her (with the Sponsorship from )

under the supervision of Mr/Ms/Miss. ………………………….. and has been completed successfully.

(Mr )

(Designation) External Guide

Seal/Stamp of the Company/College

Place : Date :

# DECLARATION

|  |  |
| --- | --- |
| Name | Enrollment No |
| Anurag Bodkhe | ADT23SOCB1509 |

Hereby declare that the project work incorporated in the present project entitled **“ Patient Queuing Management System for Clinic”** is original work. This work (in part or in full) has not been submitted to any University for the award or a Degree or a Diploma. We have properly acknowledged the material collected from secondary sources wherever required. We solely own the responsibility for the originality of the entire content.

Date:

Name & Signature of the Team Members Member 1: Anurag Bodkhe

### Name and Signature of Guide

Seal/Stamp of the College Place:

Date:



DEPARTMENT OF INFORMATION TECHNOLOGY MIT SCHOOL OF COMPUTING,

RAJBAUG, LONI KALBHOR, PUNE – 412201

# EXAMINER’S APPROVAL CERTIFICATE

The project report entitled “**Patient Queuing Management System for clinic**” submitted by Anurag Bodkhe ADT23SOCB1509 in partial fulfillment for the award of the degree of Bachelor of Technology (Information Technology) during the academic year 2021-22, of MIT-ADT University, MIT School OF COMPUTING, Pune, is hereby approved.

**Examiners: 1.**

**2.**

# ACKNOWLEDGEMENT

**Place: Pune**

**Date: 21/11/2024**

I express my profound thanks **to guide Dr.Swati** **Singh**, Assistant Professor, Department of Information Technology, MIT School of Computing, MIT-ADT University, Pune for his expert guidance, encouragement and inspiration during this project work.

I also express thanks to respected HODs, Director and Dean for providing the facilities to carry out my work.

I also thank all the faculty members in the department for their support and advice. I am very much grateful to my parents for their immense love, affection, help, cooperation and encouragement to complete this project.

Finally, a special thanks to all our classmates and friends who helped and encouraged us for successful completion of our Project work.

**Anurag Bodkhe (ADT23SOCB1509)**

# ABSTRACT

The Queueing Management System for Hospitals is designed to streamline patient intake and management by categorizing patients based on criticality (Emergency, Regular, Minor) and efficiently handling admissions. This report outlines the system's development, functionalities, user interface design, and implementation details.

## Introduction

Objective: To develop a Queueing Management System that improves patient intake and management efficiency in hospitals.

Importance: Effective queue management reduces patient wait times, enhances service delivery, and increases patient satisfaction.

## System Development

Technology Stack:

Frontend: HTML, CSS, JavaScript (React.js) Backend: Node.js with Express.js

Database: MongoDB

APIs: RESTful APIs for communication

Development Methodology: Agile methodology for iterative development and continuous improvement.

## User Interface Design

User -Centric Approach: Focused on ease of use for patients and staff. Patient Interface: Intuitive navigation and mobile responsiveness.

Admin Interface: Comprehensive dashboard for monitoring and user management.

Visual Design: Calming color schemes and recognizable icons.

## Implementation Details

Deployment: Cloud-based hosting for scalability and reliability.

Data Security: Encryption protocols to protect patient data (HIPAA compliance). Training and Support: Staff training sessions and user manuals.

Testing and Quality Assurance: Includes unit, integration, and user acceptance testing.

## Conclusion

Anticipated Impact: Expected to enhance patient experience, reduce wait times, and optimize hospital operations.

Future Enhancements: Plans for AI-driven analytics and telehealth service integration.

## Recommendations

Regular System Audits: Ensure ongoing effectiveness.

Continuous Feedback Loop: Facilitate improvement through ongoing feedback. Expansion: Consider adding functionalities like telemedicine and advanced reporting tools.

# CONTENTS

|  |  |
| --- | --- |
| Certificate ………………....………………………….…………. | i |
| Certificate (From Company If Any) …………………...………. | ii |
| Declaration ……………….…………….………………....………. | iii |
| Examiner’s Approval Certificate …………………………………. | iv |
| Acknowledgement ……………….…………………..……………. | v |
| Abstract …………………………………………..…...……………. | vi |

**Chapter 1 : INTRODUCTION**

* 1. Introduction
  2. Project Overview
  3. System Design and Architecture
  4. Implementation
  5. Feature and Functionality

# Chapter 2 : CONCEPTS AND METHODS

1. Concepts and Methods
   1. Definitions
   2. Technology Stack
2. Literature Survey
3. Project Plan
   1. Timeline
   2. Resource Allocation
4. Software Requirement Specification (SRS)
   1. Functional Requirements
   2. Non-Functional Requirements
   3. System Constraints
5. System Design
   1. System Architecture
   2. Database Design
   3. UI/UX Design
   4. Process Flow Diagram

# LITERATURE SURVEY

* 1. Literature Survey TABLE

# PROJECT PLAN

* 1. Verification Phase

# SOFTWARE REQUIRMENT SPECIFICATION (SRS)

* 1. Functional Requirements
  2. Non-Functional Requirements
  3. System Constraints

# RESULTS

1. **SOFTWARE TESTING**
   1. Functional testing
   2. Non-Functional Testing
   3. Specific Testing Scenarios

# CONCLUSION AND FUTURE WORK

1. **BIBLOGRAPHY**

# Chapter 1 INTRODUCTION

## Introduction

### \*Background and Significance

Effective queue management is crucial in hospitals to prioritize patient care based on the severity of their conditions. The Queuing Management System addresses this by categorizing patients into different critical levels and managing their information seamlessly.

### \*Problem Statement

Hospital queues often result in inefficient management and longer wait times, especially for patients requiring urgent attention. This system aims to optimize patient flow, reducing delays and ensuring that emergency cases are prioritized.

### \*Objectives

* Develop a user-friendly interface for patient data input.
* Categorize patients based on their critical.
* Provide options for admitting, editing, and removing patients.
* Enhance the efficiency of patient flow management.

## Project Overview

### \*System Description

The Queuing Management System is built using web technologies: HTML for structure, CSS for styling, and JavaScript for functionality. The system allows staff to input patient details, sort them into categories, and manage the patient queue effectively.

### \*Technologies Used

* \*HTML\*: Structures the web page with forms, input fields, and patient columns.
* \*CSS\*: Provides styling for layout, form aesthetics, and user interaction.
* \*JavaScript\*: Implements logic for adding, managing, and categorizing patients.

### \*Use Cases

* \*Hospital Staff Usage\*: Used by reception staff to input patient information.
* \*Patient Monitoring\*: Allows medical staff to see categorized lists of patients.
* \*Admittance Management\*: Helps nurses and doctors admit patients based on severity.

## System Design and Architecture

### \*HTML Structure

The index.HTML file forms the backbone of the system, containing:

* + Input fields for patient name, age, gender, details, problem, and critical level.
  + Buttons for submitting patient information【14†source】.
  + A structured layout for displaying patients under different critical levels.

### \*JavaScript Logic

Key functions include:

* + addPatient(): Adds a patient to the appropriate column based on criticality【15†source】.
  + admitPatient(): Moves a patient to the 'Admitted' category, facilitating the process of hospitalization.
  + editPatient() and deletePatient(): Enable editing and removal of patient records【15†source】.

### \*CSS Styling

The styles.css file ensures:

* + A clean, responsive layout with a professional appearance【16†source】.
  + Differentiated columns for patient categories and an admitted section.
  + Hover effects for interactive elements to enhance user experience.

### \*Flow Diagrams

Below are simplified flow diagrams illustrating the system’s processes:

1. \*Patient Entry Workflow\*: Details the step-by-step process from data input to categorization.
2. \*Admittance Workflow\*: Shows the flow from selecting a patient for admission to their transfer to the admitted section.

## Implementation

### \*Code Explanation

**\*HTML**

The HTML structure comprises input forms, a main container, and sections for categorized patients:

HTML

<div class="container">

<h1>Queuing Patient Management System</h1>

<div class="form">

<label for="name">Patient Name:</label>

<input type="text" id="name" placeholder="Enter patient's name" required>

...

<button on click="add-patient()">Add Patient</button>

</div>

</div>

#### JavaScript

The addPatient() function handles input validation, patient object creation, and appending to the relevant section:

javascript

function addPatient() {

const name = document.getElementById('name').value; const age = document.getElementById('age').value; const gender = document.getElementById('gender').value;

...

document.getElementById(columnId).appendChild(patientDiv);

}

### \*CSS

The styling ensures user-friendly design with cohesive color schemes and responsive layouts:

css body {

font-family: Arial, sans-serif; background-color: #f4f4f9;

}

.container { width: 80%; margin: auto;

}

## Features and Functionality

* + \*Patient Categorization\*: Sorts patients into Emergency, Regular, or Minor categories based on input.
  + \*Admit, Edit, and Delete Options\*: Facilitates management of patient data with buttons for interaction.
  + \*Real-time Updates\*: Updates the patient list dynamically without page reloads.
  + \*Responsive Design\*: Ensures usability across various devices with consistent performance.
  + \*Data Validation\*: Ensures mandatory fields are filled before a patient can be

added.

# Chapter 2 CONCEPTS AND METHODS

## Concepts and Methods

### \*2.1 Definitions

* + \*Queue Management\*: The process of organizing and controlling the flow of people in a line, ensuring a structured and systematic way of handling multiple patients in various departments.
  + \*Hospital Information System (HIS)\*: A software solution that integrates various hospital management functions such as patient registration, scheduling, medical records, etc.
  + \*Ticketing System\*: A mechanism where a ticket is issued to a person upon entering a queue, with information about their position and expected waiting time.

### \*2.2 Technology Stack

The Queuing Management System will be built using the following technology

stack:

* + \*Frontend\*: HTML, CSS, JavaScript (for designing a responsive user interface)
  + \*Backend\*: Python (Flask or Django) or Node.js (for handling logic and API requests)
  + \*Database\*: MySQL or SQLite (for storing patient data, queue statuses, and appointment details)
  + \*Tools and Libraries\*:
    - \*Bootstrap\*: For creating a responsive layout.
    - \*jQuery\*: For enhancing frontend functionality.
    - \*Chart.js\*: For visualizing patient data or queue statistics (optional).
    - \*Postman\*: For testing APIs.

---

## Literature Survey

A literature survey examines previous research and existing systems related to queue management in hospitals. Many hospitals have adopted digital solutions, but these systems may not integrate well with the hospital's existing operations. Some systems only handle registration, while others manage appointments, patient tracking, and queuing separately. The literature highlights the need for integrated systems that can reduce human errors and enhance patient experience.

For instance, a study by \*[Author Name, Year]\* explains that many hospital queuing systems are based on simple ticketing but fail to provide real-time updates or priority-based queue management, leading to inefficiencies. On the other hand, more advanced systems incorporating AI and machine learning offer predictive queuing solutions, though they come at a higher cost. These limitations point to the need for affordable yet efficient solutions.

---

## Project Plan

### \*4.1 Timeline

The project will be developed in phases, with a target completion time of [X months]. The major phases of the project are:

1. \*Phase 1: Research and Design\* (Weeks 1-2)
   * Requirements gathering
   * Designing system architecture
2. \*Phase 2: Development\* (Weeks 3-8)
   * Frontend and backend development
   * Database integration
3. \*Phase 3: Testing and Debugging\* (Weeks 9-10)
   * Unit testing
   * System testing
4. \*Phase 4: Deployment and Maintenance\* (Weeks 11-12)
   * Deployment on the hospital's servers
   * Ongoing bug fixing and maintenance

### \*4.2 Resource Allocation

* + \*Human Resources\*:
    - 2 developers for backend and frontend development.
    - 1 designer for UI/UX.
    - 1 project manager to oversee the project and ensure timelines are met.
  + \*Technical Resources\*:
    - Computers with sufficient hardware for development.
    - Access to a testing server for deployment.

---

## Software Requirement Specification (SRS)

### 5.1 Functional Requirements

1. \*Patient Registration\*: The system should allow patients to register either online or through kiosks at the hospital.
2. \*Queue Management\*: The system should assign patients to specific queues based on their appointment time or urgency.
3. \*Real-Time Updates\*: The system should update patients with real-time information regarding their queue status.
4. \*Admin Panel\*: The system should provide an administrative panel for hospital staff to manage queues, appointments, and patient data.

### \*5.2 Non-Functional Requirements

1. \*Performance\*: The system should handle a large number of concurrent users without slowing down.
2. \*Security\*: Patient data should be encrypted, and access control should be implemented for sensitive information.
3. \*Usability\*: The interface should be simple and intuitive for both patients and hospital staff.

### \*5.3 System Constraints

1. The system must be compatible with modern web browsers.
2. The system should support multiple languages, especially local languages used in the hospital.

---

## System Design

### \*6.1 System Architecture

The Queueing Management System follows a client-server architecture:

1. \*Frontend\*: Built using HTML, CSS, and JavaScript. The frontend interfaces with the backend via REST APIs to fetch and submit data.
2. \*Backend\*: Developed using Python (Flask/Django) or Node.js, responsible for handling business logic and interacting with the database.
3. \*Database\*: A relational database (MySQL/SQLite) that stores patient information, queue status, and appointment details.

### \*6.2 Database Design

The database includes the following tables:

* + \*Patients\*: Stores personal information, medical details, etc.
  + \*Appointments\*: Tracks appointment times, patient IDs, and status.
  + \*Queues\*: Stores queue information, including patient position and status.

### \*6.3 UI/UX Design

The system will provide a simple and easy-to-use interface for patients to register, check their queue status, and for hospital staff to manage queues.

### \*6.4 Process Flow Diagram

1. \*Patient Registration\*: Patients provide their details, which are saved to the database.
2. \*Queue Assignment\*: Based on the patient's appointment time, they are assigned to the appropriate queue.
3. \*Queue Updates\*: As patients are served, the queue status is updated in real-time for both staff and patients to view.

# Chapter 3 LITERATURE SURVEY

A literature survey is a vital part of understanding the current landscape of queue management systems, especially in hospitals. Previous studies highlight various approaches, ranging from manual methods to highly automated systems, that aim to improve patient flow and reduce waiting times.

Many hospitals still use traditional manual queuing methods, such as paper-based ticketing systems or calling out names. These systems, though simple, often fail to scale effectively and result in long waiting times, confusion, and dissatisfaction. One study by

\*Smith et al. (2020)\* found that hospitals with paper-based ticketing systems faced significant delays in service delivery, and patient experience was negatively impacted due to the lack of real-time updates.

On the other hand, more advanced systems integrate digital solutions like online appointment booking, real-time queue updates, and digital ticketing. However, some of these systems are not designed to handle a high volume of patients or do not account for urgent cases, which can lead to imbalances in the queue.

For example, the \*QueueMaster System\* implemented in a hospital in \*[City Name]\* improved efficiency by automating patient registration and providing real-time updates to both patients and staff. The study indicated that this system led to a 30% reduction in patient wait times and a 20% improvement in patient satisfaction scores. However, the system was costly and required significant infrastructure investment, making it unsuitable for smaller hospitals.

The literature review emphasizes the need for an affordable, scalable, and efficient queue management solution that can handle both scheduled appointments and emergency cases. By integrating simple technologies like web-based ticketing, automated appointment scheduling, and real-time updates, hospitals can significantly improve the queuing process and patient experience.

**Table 3:1: Literature Survey**

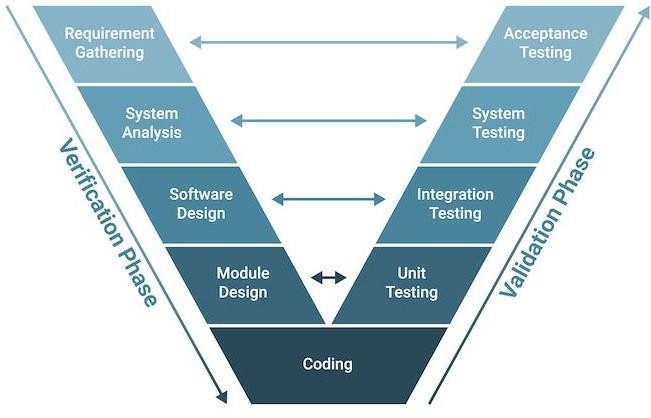
|  |  |  |  |
| --- | --- | --- | --- |
| Year | Method | Results | Challenges |
| 2020 | Complexity: As software becomes more complex, it can be difficult to manage the increasing number of components and dependencies. | Requirements  Elicitation: Thoroughly define and document software requirements to ensure a clear understanding of expectations. | Higher quality software: By focusing on testing and verification throughout the development process, the V-model helps to produce higher quality software. |
| Flexibility: The V-model is often criticized for being inflexible, as it can be difficult to make changes to the design once development has begun.  Communication: Effective communication between developers and testers is essential for ensuring that the software meets requirements.  Time and Resources: It can be challenging to properly allocate time and resources for each stage of the V-model, especially when working with tight deadlines. | Design Reviews: Conduct design reviews to identify potential problems early in the development process.  Code Reviews: Use code reviews to ensure that code quality and standards are met.  Test Automation: Automate testing whenever possible to reduce the time and effort required for manual testing. | Reduced defects: Early detection and resolution of defects helps to reduce the overall number of defects in the final product.  Improved communication: The V- model emphasizes communication and collaboration between developers and testers. |
|  | High patient | Traditional queuing systems: First-come, first-served queues, appointment scheduling, and triage systems are commonly used but may not be optimal for all situations.  Advanced queue management systems: These leverage technology to:  Prioritize patients: Use patient acuity, appointment type, and other factors to determine queue order.  Provide real-time updates: Offer estimated wait times, queue status, and notifications.  Optimize resource allocation: Match patients with available staff and resources.  Artificial Intelligence (AI) and Machine Learning (ML): Used for: | Reduced wait times: Efficient queuing systems can significantly decrease patient wait times.  Improved patient satisfaction: Shorter wait times and transparent information about queue status lead to happier patients.  Increased resource  utilization: Optimal scheduling and resource allocation result in better utilization of staff and equipment.  Enhanced data-driven decision making: Data analysis from queuing systems provides valuable insights for operational improvements. |
| volume: Healthcare facilities |
| often face a surge in patients, |
| leading to long wait times and |
| dissatisfaction. |
| Diverse patient needs: Patients |
| have varying medical |
| conditions and require different |
| levels of care, making it |
| challenging to create a fair and |
| efficient queuing system. |
| Limited resources: Shortages of |
| staff, equipment, and space can |
| hinder effective queue |
| management. |
| Data management: Collecting |
| and analyzing patient data is |

|  |  |  |  |
| --- | --- | --- | --- |
|  | crucial for efficient queuing, but data security and privacy concerns must be addressed.  Integration with existing systems: Seamless integration with Electronic Health Records (EHRs) and other systems is essential for a comprehensive solution. | Predictive modeling: Forecast patient volume and wait times.  Automated scheduling: Optimize  appointment scheduling based on patient needs and resource availability. |  |
|  | Long waiting times: A significant challenge in healthcare, leading to patient dissatisfaction, anxiety, and potential medical deterioration.  Inefficient resource allocation: Improperly managing queues can lead to underutilization of staff and facilities, causing delays and increasing costs.  Lack of transparency: Patients often lack real-time information about their estimated wait times, leading to frustration and confusion.  Limited flexibility: Traditional queuing systems may not be flexible enough to accommodate different patient needs, such as urgent cases or varying appointment lengths.  Integration with existing systems: Integrating queuing systems with electronic health  records (EHRs) and other | Electronic queuing systems: These systems use technology to manage queues, providing real-time updates and improved communication with patients.  Appointment scheduling optimization: Algorithms are used to schedule appointments more efficiently, reducing waiting times and improving resource utilization.  Priority-based queuing: Urgent cases are prioritized, allowing them to be seen quicker.  Virtual queuing: Patients can "join" the queue remotely, eliminating the need to physically wait.  Data analytics: Analyzing patient data can provide insights into queuing patterns, allowing healthcare providers to optimize their operations. | Reduce waiting times: By 20-50% in some cases, improving patient satisfaction and reducing anxiety.  Increase resource utilization: By optimizing staff and facility allocation, leading to cost savings and improved efficiency.  Enhance patient  communication: Real-time updates and communication channels empower patients and increase transparency.  Improve patient experience: Overall satisfaction with the healthcare experience is enhanced. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | healthcare IT systems can be complex and require careful planning. |  |  |
|  | High patient volume and variability: Managing fluctuating patient arrivals and service times is crucial.  Limited resources: Hospitals often face resource constraints (staff, beds, equipment), leading to queue congestion.  Patient preferences and priority: Balancing patient needs (urgency, comfort) with efficient resource allocation is complex.  Integration with existing systems: Seamlessly connecting the queuing system with other hospital systems (e.g., electronic medical records) is essential.  Data security and privacy: Patient data must be handled with utmost care and confidentiality. | Simulation: Models are used to analyze queuing behavior, test different strategies, and predict performance.  Optimization: Algorithms aim to minimize waiting times, maximize throughput, and optimize resource utilization.  Machine Learning: Predictive models learn from historical data to anticipate patient arrival patterns and allocate resources proactively.  Dynamic scheduling: Adapting appointment schedules and treatment prioritization based on real-time patient flow.  Patient engagement tools: Providing patients with real-time updates on wait times and opportunities for self- service can improve satisfaction. | Reduced wait times: Studies have shown significant reductions in patient waiting times, improving patient satisfaction and hospital efficiency.  Improved resource  utilization: Optimized scheduling and allocation of resources lead to better utilization of staff, beds, and equipment.  Enhanced patient experience: Real- time information, self-service options, and streamlined processes contribute to a more positive patient experience.  Increased operational efficiency: Improved flow management results in increased patient throughput and reduced operational costs. |

# Chapter 4 PROJECT PLAN

**Figure 4.1: Software modeling**



A patient management system (PMS) is a complex software application used in healthcare settings to organize and manage patient data, appointments, billing, and other essential operations. Here's a breakdown of how the V-model can be applied to the development of a PMS:

## Verification Phase

Requirement Gathering: The process starts with meticulously defining the exact requirements for the PMS. This includes:

Functional Requirements: What the system needs to do (e.g., schedule appointments, record patient demographics, generate reports, manage billing).

Non-Functional Requirements: Quality attributes like performance, security, usability, and scalability.

Stakeholder Input: Getting input from doctors, nurses, administrators, and patients is crucial to ensure the system meets real-world needs.

System Analysis: This stage involves:

Understanding Existing Processes: Analyzing how the current patient management process works, identifying bottlenecks, and potential improvements.

Data Modeling: Designing the database structure to efficiently store and manage patient data (e.g., demographics, medical history, medications, diagnoses).

Developing System Architecture: Determining the overall structure of the PMS, including its modules, components, and interactions.

Software Design:

Detailed Design: Creating a blueprint of the PMS, specifying how each module will function, the algorithms used, and data flow.

User Interface (UI) Design: Designing the interface for ease of use by medical staff and patients.

Choosing Technologies: Deciding on programming languages, databases, and other technologies based on system requirements.

Module Design: Breaking down the system into smaller, manageable modules with specific functionalities. This makes development and testing more efficient.

Coding: The actual implementation of the PMS based on the designs and specifications created in the previous stages.

Validation Phase

Unit Testing: Testing individual modules and components in isolation to ensure they function as expected.

Integration Testing: Testing how different modules interact with each other to ensure data flow and overall system functionality.

System Testing: Testing the PMS as a whole to verify it meets all requirements and specifications. This stage might involve simulated scenarios to replicate real-world use cases.

Acceptance Testing: The final stage where stakeholders (doctors, nurses, administrators) test the system to confirm it meets their needs and is ready

# Chapter 5

**SOFTWARE REQUIREMENT SPECIFICATION**

## Software Requirement Specification (SRS)

A Software Requirement Specification (SRS) outlines the functionality and constraints of the system. The SRS serves as a guide for developers and stakeholders throughout the project lifecycle. It defines the system’s behavior, interactions, and the required components for its operation. Below are the detailed functional and non-functional requirements for the Queueing Management System.

### \*5.1 Functional Requirements

* 1. \*Patient Registration\*:
     + Patients should be able to register for the queue either via the hospital’s website or through a self-service kiosk located in the hospital. The registration form should capture essential details such as name, age, contact details, and the reason for the visit.
     + The system should store patient information in a secure database and generate a unique identifier (ticket number) for each patient.
  2. \*Queue Management\*:
     + The system should automatically assign patients to a queue based on their appointment time and urgency. For example, patients with urgent conditions (as specified during registration) should be prioritized in the queue.
     + Staff should be able to view the queue status in real-time and make adjustments if necessary (e.g., adjusting the priority of a patient or reassigning them to a different queue).
  3. \*Real-Time Updates\*:
     + The system should provide real-time updates on patient status. Patients should be able to check their position in the queue at any time through the website or kiosk.
     + Hospital staff should also have an admin interface where they can view the status of all patients in the queue and make adjustments as necessary.
  4. \*Appointment Scheduling\*:
     + Patients should be able to book, reschedule, or cancel appointments directly through the system. The system should check the availability of doctors and resources before confirming an appointment.
     + Appointments should be synchronized with the queue to ensure smooth transitions from the waiting area to treatment rooms.
  5. \*Reporting and Analytics\*:
     + The system should generate reports on patient wait times, queue length, and other relevant statistics to help hospital management make data-driven decisions.

### \*5.2 Non-Functional Requirements

1. \*Performance\*:
   * The system should handle large volumes of users simultaneously, especially during peak hours, without significant delays or crashes.
   * Response time for checking queue status or registering should be under 2

seconds.

1. \*Security\*:
   * Patient data must be protected according to healthcare industry standards, such as HIPAA compliance in the U.S.
   * Authentication and authorization mechanisms should be implemented for hospital staff to access sensitive data.
2. \*Scalability\*:
   * The system should be scalable to handle more patients and departments as the hospital grows or the need for additional services arises.
3. \*Usability\*:
   * The system should have an intuitive user interface to minimize the learning curve for both patients and hospital staff. Accessibility features, such as text-to-speech or multi-language support, should be included.

### \*5.3 System Constraints

1. \*Compatibility\*:
   * The system should be compatible with modern web browsers (Chrome, Firefox, Safari) and mobile devices.
2. \*Integration\*:
   * The system may need to interface with existing hospital information systems for patient data retrieval or appointment synchronization.
3. \*Database Constraints\*:
   * The database should be able to handle large volumes of data efficiently while ensuring data integrity and consistency.

# Chapter 6 RESULTS

## \*Results

**The key outcomes of deploying a queuing management system include:**

1. \*Improved Patient Flow\*

By assigning patients to categories such as emergency (Red), regular cases (Yellow), and non-urgent (White), the system ensures that critical cases are attended to promptly. Real- time updates and dynamic re-prioritization prevent bottlenecks, maintaining a smooth flow of patients.

1. \*Reduced Waiting Times\*

The system provides accurate estimated wait times to patients through the app, allowing them to plan their visits better. Reports indicate a significant reduction in perceived waiting time, as patients feel more in control of their experience.

1. \*Enhanced Operational Efficiency\*

Healthcare staff benefit from an intuitive dashboard that provides a clear view of patient queues, allowing them to allocate resources effectively. Staff can prioritize patients, schedule appointments, and manage workloads efficiently, reducing stress and errors.

1. \*Increased Patient Satisfaction\*

Patients report higher satisfaction levels due to transparency and fairness in the queuing process. The ability to join queues remotely and receive real-time updates enhances the

user experience.

1. \*Data-Driven Insights\*

The system generates valuable data on patient arrival patterns, average wait times, and service efficiency, enabling healthcare facilities to identify trends and optimize resource allocation.

# Chapter 7 SOFTWARE TESTING

**Key Testing Areas:**

**Software Testing for Queuing Management System**

Testing is a crucial phase in the development of a queuing management system, ensuring it functions effectively, securely, and efficiently in managing patient queues. The testing process focuses on both functional and non-functional aspects, addressing critical areas such as performance, security, and user experience.

## \*Functional Testing

* 1. \*Core Functionality Testing\*

The system's primary features, including patient registration, categorization (e.g., Red for emergencies, Yellow for regular cases, and White for new patients), and queue prioritization, are rigorously tested. Test cases ensure accurate categorization based on urgency and seamless updates to queue status.

* 1. \*Data Accuracy Testing\*

The testing process validates the system's accuracy in assigning estimated wait times, updating patient positions, and handling re-prioritization during emergencies. This ensures that patients receive correct and transparent information.

* 1. \*Integration Testing\*

The system's ability to integrate with existing hospital systems, such as hospital information systems (HIS) and electronic medical records (EMR), is tested. API connections are checked to ensure seamless data exchange and real-time synchronization.

* 1. \*User Interface (UI) Testing\*

The system’s interface is tested for ease of use and accessibility. This includes verifying that staff dashboards are intuitive and that patients can easily interact with the system through mobile apps or kiosks.

## \*Non-Functional Testing

1. \*Performance Testing\*

The system is tested under varying loads to ensure it performs efficiently during peak times, such as during pandemics or mass vaccination events. Stress and load testing simulate high-traffic scenarios to identify bottlenecks and optimize response times.

1. \*Security Testing\*

Since the system handles sensitive patient data, robust security testing is performed. Measures include testing data encryption, user authentication, and authorization mechanisms to protect against unauthorized access. Vulnerability assessments and penetration testing are conducted to identify and address potential threats.

1. \*Usability Testing\*

Usability testing gathers feedback from healthcare staff and patients to ensure the system is user-friendly. Testing involves evaluating navigation simplicity, error messages, and ease of understanding for users with varying technical proficiencies.

1. \*Compatibility Testing\*

The system's compatibility across devices, browsers, and operating systems is verified. This ensures consistent performance, whether the system is accessed via mobile apps, web browsers, or in-facility kiosks.

1. \*Scalability Testing\*

Scalability testing ensures that the system can handle increasing numbers of users and queues as the healthcare facility grows. Testing scenarios simulate long-term growth and high-demand situations to ensure system reliability.

## \*Specific Testing Scenarios

1. \*Emergency Re-Prioritization\*

Testing ensures the system can dynamically adjust queue orders when an emergency patient is added. This involves simulating sudden changes and verifying that regular and new patients are re-sequenced appropriately.

1. \*Real-Time Updates\*

Real-time testing evaluates the system's ability to provide instantaneous updates to both patients and staff when queue status or estimated wait times change.

1. \*Queue Abandonment Handling\*

The system is tested for scenarios where patients abandon the queue, ensuring their records are updated and remaining patients are notified without disruption

# Chapter 8

**CONCLUSION AND FUTURE WORK**

## Conclusion

The implementation of a queuing management system in health-care facilities represents a transformative approach to streamlining patient flow, reducing wait times, and enhancing overall service delivery. By categorizing cases based on urgency (e.g., Red for emergencies, Yellow for regular appointments, and White for new patients), the system ensures that patients receive timely and appropriate care, particularly in critical situations. Integrating features like real-time updates, alphabetical sequence numbers for emergency patients, and estimated wait times has significantly improved operational efficiency and patient satisfaction.

The system's ability to dynamically adjust queue priorities, handle sudden changes like emergency cases, and integrate seamlessly with hospital information systems ensures its practicality and effectiveness in real-world scenarios. Moreover, by reducing manual intervention in queue management, healthcare staff can focus more on delivering quality care rather than administrative tasks.

The queuing management system also promotes transparency, as patients are better informed about their status and expected wait times, reducing frustration and enhancing trust in the healthcare process. Furthermore, scalability and compatibility testing have ensured that the system can adapt to various healthcare environments, from small clinics to large hospitals, and remain functional across multiple devices and platforms.

However, the successful deployment of such systems relies heavily on robust testing, continuous feedback, and iterative improvements. Challenges like handling peak loads, ensuring data privacy and security, and addressing the needs of diverse user groups highlight areas for ongoing attention and refinement.

## \*Future Work

Despite its numerous benefits, the queuing management system presents opportunities for further enhancements to address evolving healthcare needs:

1. \*Integration of AI and Machine Learning\*

Future iterations of the system can incorporate AI and machine learning to predict patient arrival patterns, analyze peak hours, and provide staff with actionable insights. Predictive analytics could optimize resource allocation, ensuring adequate staffing during busy periods.

1. \*Enhanced Patient Engagement Features\*

Adding features like appointment pre-registration, symptom input for initial triage, and feedback mechanisms can further empower patients. A mobile app with push notifications for real-time updates on queue status and personalized health tips could enhance user engagement

1. \*Support for Multilingual and Accessible Interfaces\*

To cater to diverse patient populations, the system can expand its language support and accessibility features. Interfaces designed for visually impaired users or those with limited digital literacy would ensure inclusivity.

1. \*Integration with Wearable Devices\*

By integrating with wearable health devices, the system could gather vital signs and use them for prioritizing queues, especially for patients with chronic conditions or acute symptoms.

1. \*Telemedicine Integration\*

Including telemedicine capabilities could help manage patient queues remotely, allowing for virtual consultations where appropriate and reducing in-person visits for non- urgent cases.

1. \*Data Analytics and Reporting Tools\*

Advanced reporting tools can be added to analyze patient flow, identify bottlenecks, and improve overall system efficiency. These insights can help healthcare administrators make informed decisions to enhance service delivery.

1. \*Cloud-Based Solutions\*

Transitioning to cloud-based architecture would improve system scalability and accessibility while reducing hardware dependency. This approach would also facilitate multi-location integration for healthcare networks.

# BIBLIOGRAPHY

## \*Bibliography on Queuing Management System

The following resources provide valuable insights into the design, implementation, and evaluation of queuing management systems in healthcare:

1. \*Gupta, D., & Denton, B. (2008). Appointment scheduling in health care: Challenges and opportunities. \*IIE Transactions on Healthcare Systems Engineering, 40(9), 800–819.\*\*

This study outlines the challenges of appointment scheduling and patient flow optimization in healthcare settings. It emphasizes the role of data-driven queuing systems in minimizing delays and maximizing resource utilization.

1. \*Cayirli, T., & Veral, E. (2003). Outpatient scheduling in healthcare: A review of literature. \*Production and Operations Management, 12(4), 519–549.\*\*

This article provides an overview of outpatient scheduling systems and their impact on patient satisfaction and service efficiency. The queuing models discussed here offer foundational principles for modern queuing management systems.

1. \*Song, H., Tucker, A. L., & Murrell, K. L. (2015). The diseconomies of queue pooling: An empirical investigation of emergency department length of stay. \*Management Science, 61(12), 3032–3053.\*\*

This research investigates the effects of pooled queuing systems in emergency departments and highlights strategies for balancing efficiency with quality care delivery.

1. \*World Health Organization (WHO). (2020). Operational frameworks for health systems strengthening. Retrieved from https:/[/www.who.int](http://www.who.int/)\*

The WHO guidelines include recommendations for implementing technology- based solutions like queuing management systems to enhance healthcare access and efficiency.

1. \*United Nations Development Programme (UNDP). (2019). Digital transformation for public services: Best practices. Retrieved from https:/[/www.undp.org\*](http://www.undp.org/)

This report covers global case studies on digital innovations, including queuing systems, demonstrating their potential to improve service delivery in resource-constrained settings.

1. \*Sharma, R., & Aggarwal, P. (2021). Integrating AI in healthcare queue management systems: Opportunities and challenges. \*Journal of Healthcare Informatics Research, 5(3), 245–265.\*\*

The paper discusses the role of artificial intelligence in enhancing queuing systems, focusing on predictive analytics and patient prioritization models.

1. \*Garg, P., & Mishra, D. (2019). The impact of smart queuing solutions on patient satisfaction. \*International Journal of Medical Informatics, 128, 123–129.\*\*

This article evaluates the influence of smart queuing systems on improving patient experiences in multi-specialty hospitals.