ICU Reservation & Management System

Comprehensive Project Documentation
Version: 2.0 (Updated Tech Stack)

1. Introduction

The ICU Reservation & Management System is a web-based platform designed to solve the logistical chaos often found in hospital emergency wards. By connecting ambulance staff directly with hospital reception and ICU managers, the system ensures transparency and speed in patient admission.

1.1 Problem Statement

Current ICU management often suffers from:

- **Manual Processes:** Phone calls and paper logs leading to slow information transfer.
- **Lack of Visibility:** Ambulance staff often don't know bed availability until arrival.
- **Human Error:** Double-booking beds or failing to mark beds as "clean/available" after discharge.

1.2 Proposed Solution

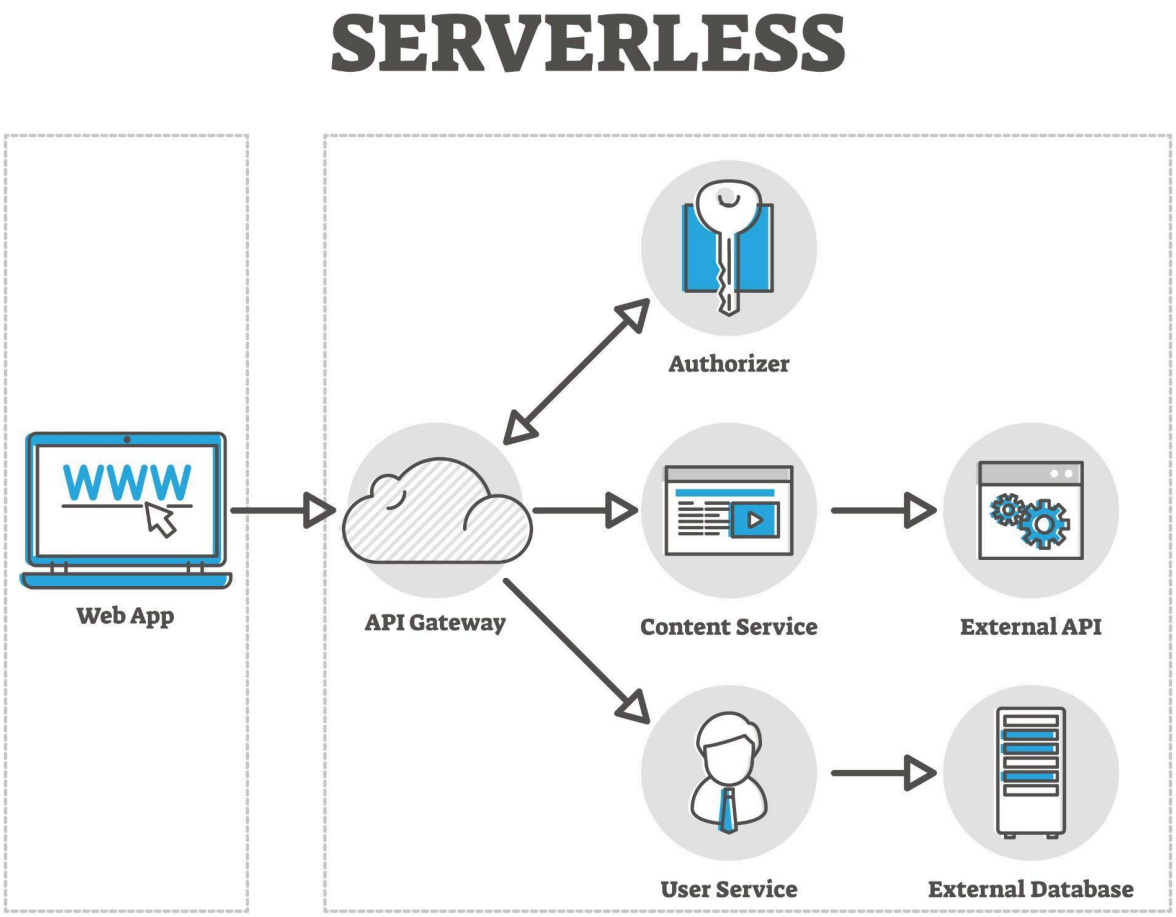
A centralized dashboard that updates in real-time. When a Receptionist marks a patient as "Arrived," the bed status instantly changes to "Occupied" across all screens. When an Ambulance starts a trip, the hospital can track their location via GPS.

2. System Architecture & Design

2.1 High-Level Architecture

The system follows a modern microservices-ready architecture:

1. **Client Layer:** React application providing different views based on the logged-in user role.
2. **API Gateway:** Express.js server handling REST API requests and WebSocket connections.
3. **Data Layer:** MongoDB for flexible storage of user profiles, logs, and bed statuses.
4. **Real-Time Layer:** Socket.io ensures that dashboard updates (e.g., bed availability) happen instantly without page reloads.



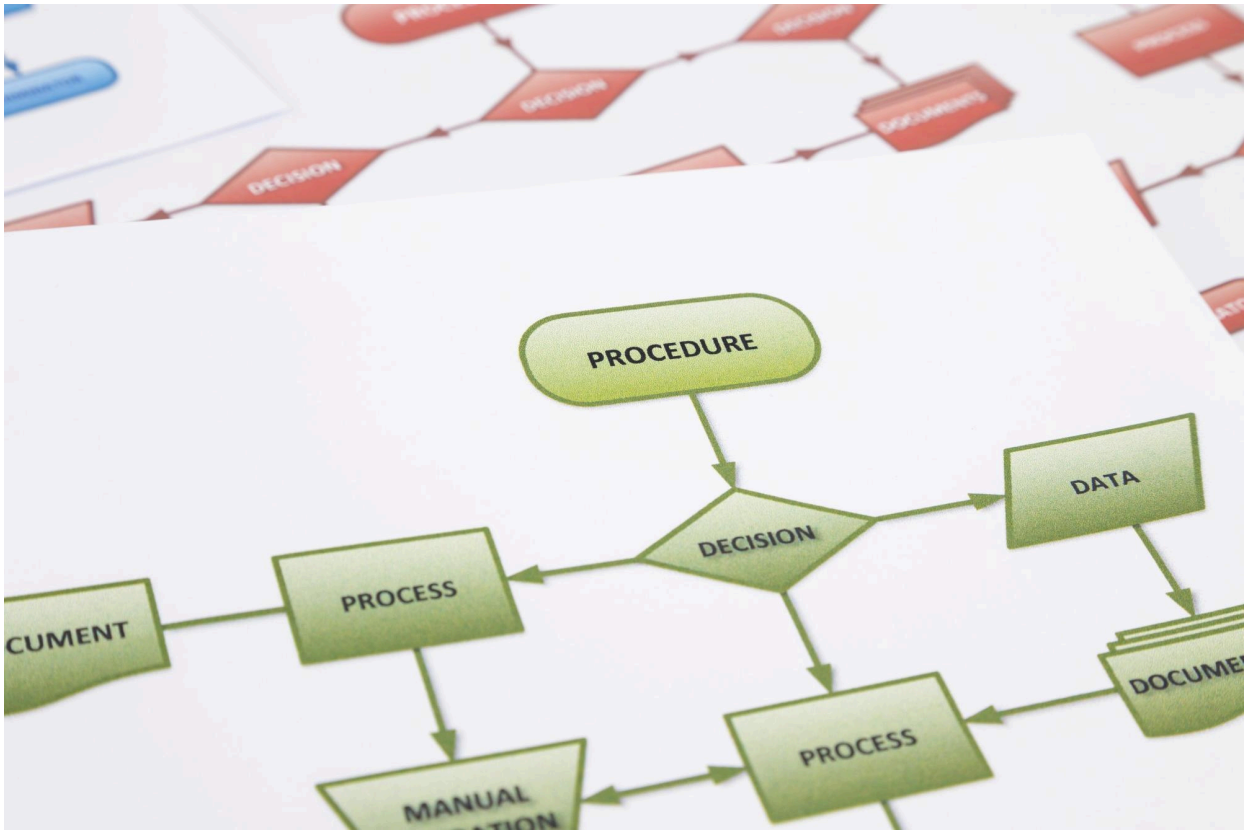
Shutterstock

Explore

2.2 Database Design (Schema Concepts)

We utilize a NoSQL document structure (MongoDB) to handle the dynamic nature of logs and real-time data.

- **Users Collection:** Stores profiles for Admins, Managers, Receptionists, and Drivers.
- **ICU_Beds Collection:**
  - bed_number (String)
  - status (Enum: Available, Occupied, Maintenance)
  - last_cleaned (Timestamp)
- **Reservations Collection:** Links a Patient to a Bed and tracks the approval workflow.
- **Ambulances Collection:** Tracks driver status and current GPS coordinates.
- **Logs Collection:** Immutable records of every action (e.g., "User X reserved Bed Y at 10:00 AM") for safety auditing.



Shutterstock

Explore

3. User Roles & Functionality

🏥 Hospital Admin & Manager

- **Responsibilities:** System oversight, user management, reviewing analytical reports.
- **Capabilities:** Can override bed status in emergencies, add/remove staff accounts, and view system health via Grafana dashboards.

💁 Receptionist

- **Responsibilities:** Patient intake and discharge.
- **Capabilities:**
  - **Check-In:** Verifies patient arrival; system auto-updates bed to "Occupied".
  - **Check-Out:** Processes discharge; system auto-updates bed to "Available" (or "Needs Cleaning").

🚑 Ambulance Staff

- **Responsibilities:** Patient transport.
- **Capabilities:** View assigned hospital destination, broadcast "En Route" status, share real-time location.

4. Visualizations & Wireframes

4.1 The Dashboard

The central hub of the application is the **Live Bed Board**.

- **Green Tiles:** Available Beds.
- **Red Tiles:** Occupied Beds (showing Patient ID).
- **Yellow Tiles:** Maintenance/Cleaning.

4.2 Analytics

Using **Prometheus and Grafana**, the system visualizes:

- Current Occupancy Rates (%)
- Average Reservation Duration
- Peak Admission Hours

5. Development & Deployment

5.1 Tech Stack

- **Language:** TypeScript (Frontend & Backend)
- **Containerization:** Docker & Kubernetes for orchestration.
- **Infrastructure as Code:** Terraform for provisioning AWS resources.

5.2 CI/CD Pipeline

We utilize **GitHub Actions** to automate our workflow:

1. **Push:** Code is committed to the repository.
2. **Test:** Unit tests run automatically.
3. **Build:** Docker images are built and pushed to the registry.
4. **Deploy:** Kubernetes clusters are updated with the new image.

6. User Tutorials (Standard Operating Procedures)

Scenario A: New Emergency Reservation

1. **Receptionist** receives a call or system alert.
2. Clicks **"New Reservation"** on the dashboard.
3. Selects **"Critical Priority"**.
4. System highlights available beds; Receptionist selects **Bed 04**.
5. **Ambulance Driver** receives notification: "Proceed to Bed 04".

Scenario B: Patient Discharge

1. Doctor clears patient for discharge.
2. **Receptionist** locates patient on the dashboard.
3. Clicks **"Process Discharge"**.
4. System updates Bed 04 status to **"Cleaning Required"**.
5. Janitorial staff is notified (Future Feature).