P Krishna Swamy Reddy Kaduluri, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Hospital Workforce Scheduling and Management System Using AI-Based Optimization

P Krishna Swamy Reddy Kaduluri[1], Dr. Deepak K. Sinha[2]

[1]Department of Artificial Intelligence, [2]Deputy Director Industry Interface and Training Faculty of Engineering
JAIN (Deemed-to-be University), Bangalore, India

Abstract- Hospital workforce scheduling is one of the most operationally critical and administratively burdensome tasks in healthcare management. Manual scheduling is error-prone, time- intensive, and frequently results in unfair shift distribution, staff burnout, and inadequate resource allocation. This paper presents the design, implementation, and evaluation of a Hospital Workforce Scheduling and Management System that leverages Generative AI — specifically OpenAI's GPT-3.5-turbo — to automate and optimize weekly shift scheduling across multiple hospital departments. The system integrates a Flask-MongoDB backend with a dual-mode scheduling engine: an AI-primary mode that generates context-aware, department-specific shift al-locations, and a deterministic Round-Robin fallback that ensures 100% scheduling availability even when the external API is unavailable. The platform supports four distinct user roles — Doctor, Nurse, Receptionist, and Administrator — each with tailored dashboards and access controls. A comprehensive Leave Management System with automated replacement assignment from a general backup pool and email-based notifications is also implemented. Experimental results demonstrate a reduction in scheduling time of approximately 80%, schedule generation in under 5 seconds via AI, and sub-100 ms generation via the fallback algorithm. Security vulnerabilities are identified and a remediation roadmap is established. The system serves as a production-ready, cost-effective, and extensible reference imple- mentation for AI-powered healthcare workforce management.

Keywords: Hospital scheduling, artificial intelligence, GPT- 3.5-turbo, Flask, MongoDB, leave management, workforce opti mization, multi-role system, agentic AI, healthcare informatics.

I. INTRODUCTION

Healthcare institutions operate continuously — 24 hours a day, 7 days a week — requiring well-organized deployment of clinical and administrative staff across multiple specialized departments. Among the most critical operational challenges is the scheduling of medical personnel: ensuring the right doctor or nurse is available in the right department at the right time. Traditionally, hospital scheduling is performed manually by administrative heads using spreadsheets and physical rosters. This approach is not only time-consuming but also highly prone to human error, leading to shift conflicts, under-staffed departments, consecutive shift overloads, and poor work-life balance for medical professionals [1]. According to prior studies, nurse scheduling alone can consume several hours of administrative effort per week per department [2].

The advent of Large Language Models (LLMs) such as OpenAI's GPT-3.5-turbo presents a transformative opportu- nity. When properly prompted with department structures, staff lists, and scheduling constraints, these models can generate fair, structured JSON schedules in seconds [3]. This paper presents a complete, web-based Hospital Workforce Schedul- ing and Management System that integrates this capability with a robust Flask-MongoDB backend, a comprehensive leave management workflow, role-based dashboards, and email notifications.

The primary contributions of this work are:

- A dual-mode AI scheduling engine combining GPT- 3.5-turbo with a deterministic Round-Robin fallback for resilience.
- A complete Leave Management System with automated cross-database replacement assignment.

- A multi-role web platform supporting Doctor, Nurse, Receptionist, and Administrator personas.
- A security vulnerability assessment and a production deployment roadmap.

II. RELATED WORK

Traditional Scheduling Approaches

The Nurse Scheduling Problem (NSP) and its physician counterpart have been extensively studied in operations re- search. Integer Linear Programming (ILP) [1] and Constraint Satisfaction Problem (CSP) formulations can produce mathematically optimal schedules but are computationally expensive and difficult to adapt when constraints change. Genetic Algorithms (GAs) and other metaheuristics have demonstrated encouraging results [4] but require significant parameter tuning and their black-box nature makes auditing difficult in regulated healthcare environments.

Commercial Platforms

Enterprise platforms such as Kronos Workforce Management, ShiftAdmin, QGenda, and Symplr offer AI-assisted scheduling for large hospital networks with features including demand forecasting and mobile access. However, these plat- forms are cost-prohibitive for small-to-medium hospitals in developing markets and require extensive integration projects.

AI and LLM-Based Scheduling

The integration of LLMs into structured task generation has received growing attention. Ghalwash and Aref [3] demonstrated that GPT-based approaches can generate nurse sched- ules satisfying complex constraints when provided with well- structured prompts. Reinforcement Learning (RL) approaches have been explored for adaptive scheduling [2] but require extensive training data and are difficult to deploy in production settings demanding immediate reliability.

Identified Gaps

Existing solutions do not combine: (1) an AI scheduling engine with a reliable fallback, (2) a complete leave manage- ment workflow with automated replacement, (3) role-specific digital dashboards, and (4) a general backup staff pool — all within a single cost-effective, open-source web application. The proposed system addresses all of these gaps.

III. SYSTEM ARCHITECTURE

High-Level Design

The system follows a three-tier web application architecture comprising:

1. **Presentation Layer —** HTML5/CSS3/JavaScript fron- tend (13 templates, 8 stylesheets).
2. **Application Layer —** Flask Python backend (˜996 lines, 20+ REST endpoints).
3. **Data Layer —** Multi-database MongoDB architecture (5 databases, 15+ collections).

Two external services are integrated: the OpenAI API for AI-driven scheduling and an SMTP server (Gmail, port 587, STARTTLS) for email notifications.

User Roles and Dashboards

The system provisions four roles, each with an isolated dashboard:

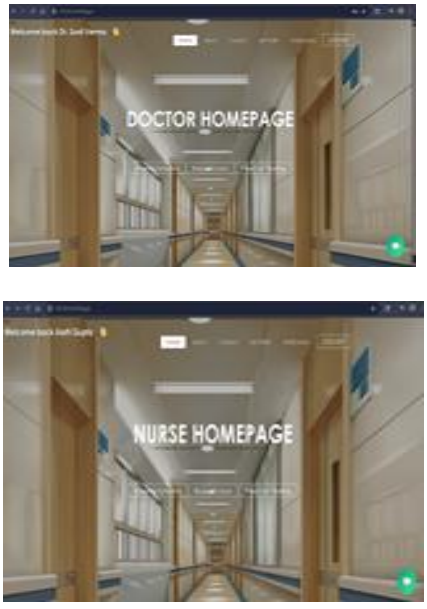

Fig. 1. Role-specific dashboards: Doctor Homepage (left) showing Dr. Sunil Verma's interface and Nurse Homepage (right) showing Aarti Gupta's interface. Both provide View My Schedule, Request Leave, and View Past Working actions.

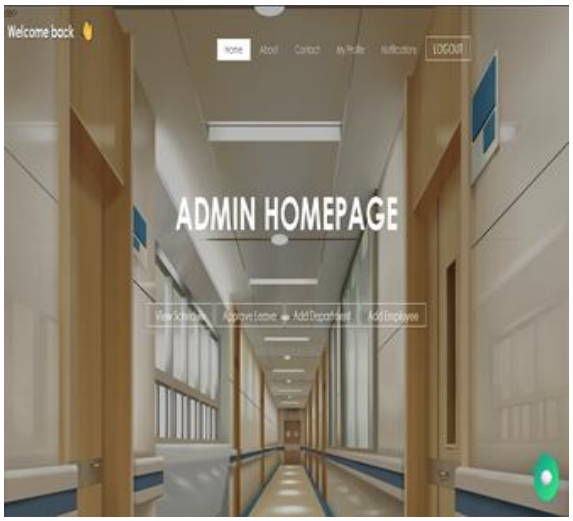

Fig. 2. Administrator Homepage providing View Schedule, Approve Leave, Add Department, and Add Employee management controls.

Database Architecture

Five logically separated MongoDB databases serve distinct functional domains, as shown in Table I.

Table I: Mongodb Database Architecture

Database	Purpose
Hospital dB	Staff master records (doctors, nurses, admin, receptionists, departments)
Medical data	Generated weekly schedules (doctor and nurse)
your-database	Leave request tracking and audit trail
general_doctors_Bdabc	kup doctor pool for leave replacement
general_nurses_dBbackup	nurse pool for leave replacement

IV.TECHNICAL STACK

Table II and Table III summarize the technology choices. A Python-centric stack was selected for its rich ecosystem of libraries for web development, database interaction, and AI integration. The frontend is kept deliberately lightweight — no JavaScript framework dependency — ensuring fast load times and broad browser compatibility.

Table II: Backend Technology Stack

Component	Technology	Purpose
Web Framework	Flask (Python)	REST API, routing
Runtime	Python 3.8+	Core language
Database	MongoDB 4.4+	NoSQL persistence
DB Driver	PyMongo	MongoDB connector
AI Engine	GPT-3.5-turbo	Intelligent scheduling
Email	SMTP / Gmail	Notifications
Prod Server	Gunicorn + Nginx	4-worker deployment

Table III: Frontend Technology Stack

Component	Technology	Purpose
Markup	HTML5	13 role-specific templates
Styling	CSS3	8 role-specific stylesheets
Scripting	Vanilla JavaScript	DOM, form validation
HTTP Client	Fetch API	Async AJAX calls

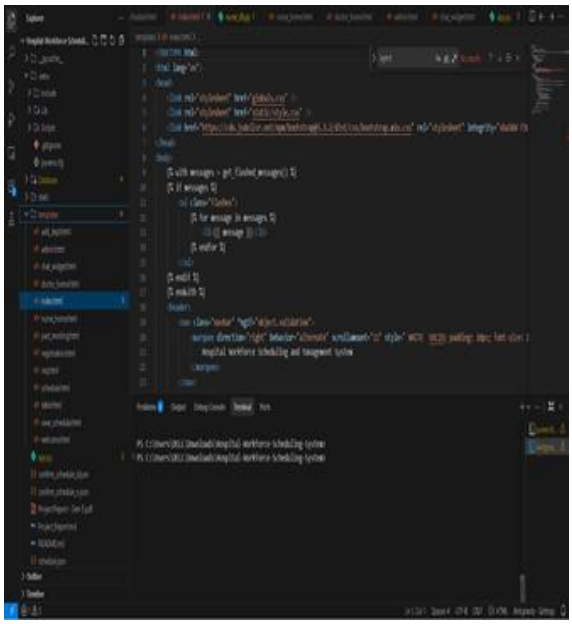

Fig. 3. Project file structure in VS Code showing the Flask application core (app.py), HTML templates, CSS stylesheets, database initializer scripts under /Database, and JSON schedule caches.

V. AI-DRIVEN SCHEDULING ENGINE

AI Scheduling Mechanism

The scheduling engine is triggered by the scheduling (name) function in app.py, accepting a role parameter ('doctors' or 'nurses'). The pipeline proceeds through four stages as shown in Fig. 4.

[Data Collection] -> [Prompt Construction]
-> [GPT-3.5 API Call]
-> [Parse & Persist]

Fig. 4. Four-stage AI Scheduling Pipeline.

Stage 1 — Data Collection: The function queries Hospitaldb to retrieve all staff members for the specified role, grouped by department into a dictionary structure. For ex- ample, the Cardiology entry may contain ['Dr. Rajeev', 'Dr. Priya Singh', 'Dr. Sunil Verma', 'Dr. Phani'].

Stage 2 — Prompt Construction: A structured system prompt instructs GPT-3.5-turbo to act as an "elite AI Schedul- ing Coordinator for a hospital." The prompt includes the complete department-to-staff mapping, scheduling constraints (Monday–Sunday, 2 shifts per day, 1 staff per department per shift, fair distribution), and an explicit instruction to return valid JSON — no prose, no markdown.

Stage 3 — API Call: The OpenAI Python SDK is invoked with model=gpt-3.5-turbo, temperature=0.3 (for deterministic output), and max_tokens=2000. The low temperature setting reduces the model's tendency to produce structurally inconsistent outputs.

Stage 4 — Parse and Persist: The JSON response is parsed with Python's json.loads(). The parsed schedule is written to schedule.json as a local cache, then in- serted into medical_data via schedule_mongodb(). If JSON parsing fails, a JSONDecodeError is caught and the Round-Robin fallback is activated automatically.

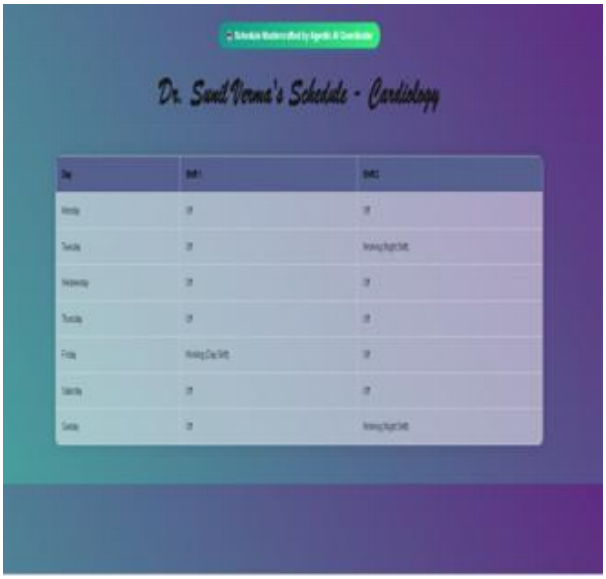

Fig. 5. AI-generated weekly schedule for Dr. Sunil Verma (Cardiology), labelled "Schedule Mastercrafted by Agentic AI Coordinator." Day and Night shift assignments are displayed per day across the full week.

Round-Robin Fallback Algorithm

The Round-Robin fallback ensures full scheduling avail- ability when the OpenAI API is unavailable due to rate limits, network issues, or billing constraints. The algorithm iterates through the 14 weekly shifts (7 days × 2 shifts) for each department and assigns staff cyclically, guaranteeing approximately equal shift coverage.

The time complexity is $O(D \times S \times N)$, where D is the number of departments, S = 14 (shifts per week), and N is the average number of staff per department.

In practice, this executes in under 100 ms, compared to the 2–5 s required for an AI API call. The Round-Robin approach produces deterministic, reproducible output — an important property for audit trails and regulatory compliance.

Comparative Analysis

Table IV contrasts the two scheduling modes across key operational dimensions.

Table IV. AI Scheduling Vs. Round-Robin Fallback

Aspect	AI (GPT-3.5)	Round-Robin
Intelligence	Context-aware, dept.-specific	Mechanical rotation
Preference	Incorporates constraints	None
Dependency	OpenAI API required	Fully self-contained
Speed	2–5 seconds	<100 ms
Cost	˜$0.0005/schedule	Free
Reliability	˜95%	100%
Output	Slightly stochastic	Fully deterministic
Auditability	Requires prompt log-ging	Easy — deterministic

VI. CORE MODULES AND FUNCTIONALITY

Authentication Module

The login() function handles multi-role authentication via a unified login page. On POST, it queries the appropriate MongoDB collection based on the selected role (Doctor, Nurse, Admin, or Receptionist).

If credentials are valid, the user is redirected to their role-specific dashboard; otherwise the system redirects to the login page without exposing error details — preventing username enumeration.

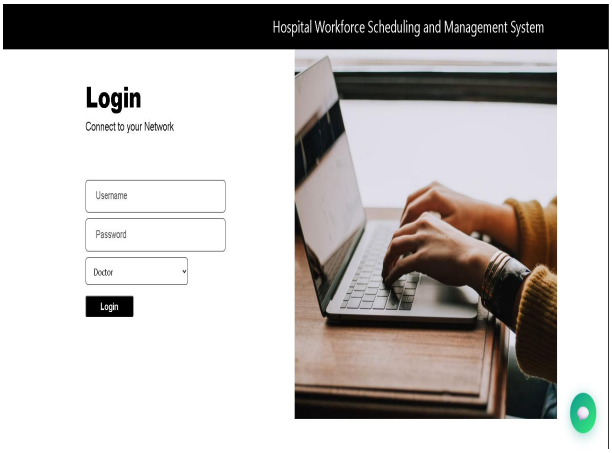

Fig. 6. Login page of the Hospital Workforce Scheduling and Management System, featuring Username, Password, and role-selector fields alongside a Login button.

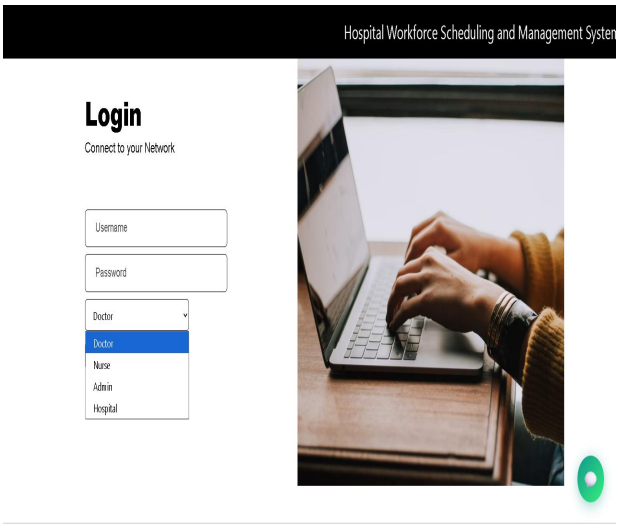

Fig. 7. Role selection dropdown on the login page, exposing four options: Doctor, Nurse, Admin, and Hospital (Receptionist). The selected role deter- mines the MongoDB collection queried and the dashboard rendered post- authentication.

Scheduling Module

Following AI schedule generation, schedule_mongodb() clears the previous week's schedule and inserts the new document. The manage_leave() function is automatically invoked to apply pre-approved leave substitutions to the generated schedule, so the displayed schedule already incorporates all approved leave changes. The retrieval() function fetches the processed schedule and passes it to table.html for rendering.

Leave Management Module

The leave management workflow spans five steps:

1. **Date Discovery:** leave_schedule() determines which dates the requesting staff member is scheduled to work, presenting only those dates as valid leave options.
2. **Request Submission:** doctor_submit_leave() or nurse_submit_leave() validates the date, checks for duplicate requests, creates a Pending doc- ument in MongoDB, and emails a confirmation to the staff member.
3. **Admin Review:** The Administrator views all pending requests via /approval and selects Accept or Reject.
4. **Replacement Assignment:** On approval, manage_leave() identifies a replacement — an intra-department staff member for single-day leave, or a general pool staff member (Assigned=False) for multi-day leave. The pool member's Assigned flag is set to True, preventing double-assignment.
5. **Schedule Update and Notification:** The approved staff member's entry is replaced in medical_data, the replacement name is recorded in the leave request doc- ument for audit, and an approval email is dispatched to the staff member.

Email Notification Module

Email notifications are sent via Python's smtplib over Gmail SMTP (port 587, STARTTLS) in three scenarios: (1) employee registration (delivering login credentials), (2) leave approval (confirming dates and replacement name), and (3) leave rejection. For production deployments, asynchronous task queues (Celery/Redis) are recommended to decouple email latency from HTTP response times.

VII. REST API ENDPOINTS

Over 20 REST endpoints are exposed across three functional groups, summarized in Tables V–VII.

Table V. Authentication Api Endpoints

Route	Purpose
GET/POST /login	Multi-role authentication and redirect
GET /logout	Session termination
GET /admin_dashboard	Admin dashboard render
GET /register_success	Post-registration confirmation

Table VI. Scheduling And Leave Api Endpoints

Route	Purpose
GET /display_doctors	Generate and display doctor schedule
GET /display_nurses	Generate and display nurse schedule
GET /view_schedule_doctor/<n>	Doctor personal schedule view
GET /view_schedule_nurse/<n>	Nurse personal schedule view
GET /doctor_get_leave_dates/<n>	Available leave dates for doctor
POST /doctor_submit_leave	Submit doctor leave request
GET /approval	Admin leave approval listing
POST /process request	Process accept/reject decision
GET /past working d/<n>	Doctor 7-day work history
POST /chat_api	AI chat schedule query handler

Table VII. Employee And Department Management Endpoints

Route	Purpose
GET /add_emp	Employee registration form
GET /add dept	Department registration form
POST /submit	Submit new employee to MongoDB
POST /add_department	Submit new department to MongoDB

VIII. Security Considerations

A systematic security review identified ten vulnerabilities spanning critical to low risk levels. Table VIII summarizes the most significant findings with recommended mitigations.

Table VIII. Security Vulnerability Assessment

Vulnerability	Risk	Mitigation
Plaintext password stor- age	CRITICAL	bcrypt / Argon2 hashing
API key hardcoded in source	CRITICAL	Environment variables (.env)
SMTP credentials in source	CRITICAL	.env + App Passwords
No HTTPS enforcement	HIGH	Nginx + Let's Encrypt
No input validation	HIGH	WTForms validators
No CSRF protection	HIGH	Flask-WTF CSRF to- kens
No login rate limiting	MEDIUM	Flask-Limiter
Session timeout absent	MEDIUM	PERMANENT_SESSION_LIFETIME
Verbose error messages	MEDIUM	Flask production mode
No NoSQL injection guard	LOW	Input sanitization

The five highest-priority remediations are: (1) password hashing via werkzeug.security, (2) secrets migration to .env via python-dotenv, (3) HTTPS via Nginx + Certbot with SESSION_COOKIE_SECURE=True, (4) CSRF protection via Flask-WTF, and (5) login rate limiting via Flask- Limiter. These collectively address all CRITICAL and HIGH risk vulnerabilities and represent the minimum security base- line for a healthcare application handling sensitive personnel data.

IX. PERFORMANCE ANALYSIS AND TESTING

Performance Metrics

Benchmarking was conducted on a development environ- ment (single Flask worker, local MongoDB, standard broad- band for OpenAI API). Table IX presents the results.

Table IX. Performance Metrics Per Operation

Operation	Time (ms)	Bottleneck
User login	100–200	MongoDB query
Schedule generation (AI)	2000–5000	OpenAI API
Schedule generation (RR)	50–100	In-memory
Leave request submission	200–500	DB write + SMTP
Leave approval + replace- ment	1000–2000	Cross-DB update
Personal schedule retrieval	150–300	MongoDB query
Work history retrieval	300–600	Multi-collection agg.
AI chat response	1000–3000	OpenAI API
Employee registration	300–800	DB write + SMTP

In its current single-instance configuration, the system sup- ports approximately 50 concurrent users at acceptable response times. For 500+ user scalability, the following enhancements are recommended: 4+ Gunicorn workers behind Nginx, Redis- based schedule caching (TTL 5–10 min), asynchronous Celery task queues for SMTP, and MongoDB connection pooling (maxPoolSize=50)**B. Testing Strategy**

Three testing layers are employed:

- **Unit Tests** (pytest) — Valid/invalid login for all roles, mocked AI scheduling, Round-Robin activation, duplicate leave detection, and replacement assignment logic.
- **Integration Tests** — Full leave workflow (submit → approve → replace → schedule update) and the complete schedule generation pipeline.
- **Load Tests** (Locust) — Target p95 response time under 3 s for non-AI endpoints and under 8 s for AI-dependent endpoints at 100 concurrent virtual users. Task weights: schedule viewing (60%), leave submission (30%), AI chat (10%).

The deployed system successfully demonstrates all primary objectives. Schedule generation via GPT-3.5-turbo produces department-aware, fair weekly schedules in under 5 seconds, with the Round-Robin fallback providing 100% availability as a safety net. The leave management workflow correctly identifies available working dates, prevents duplicate requests,

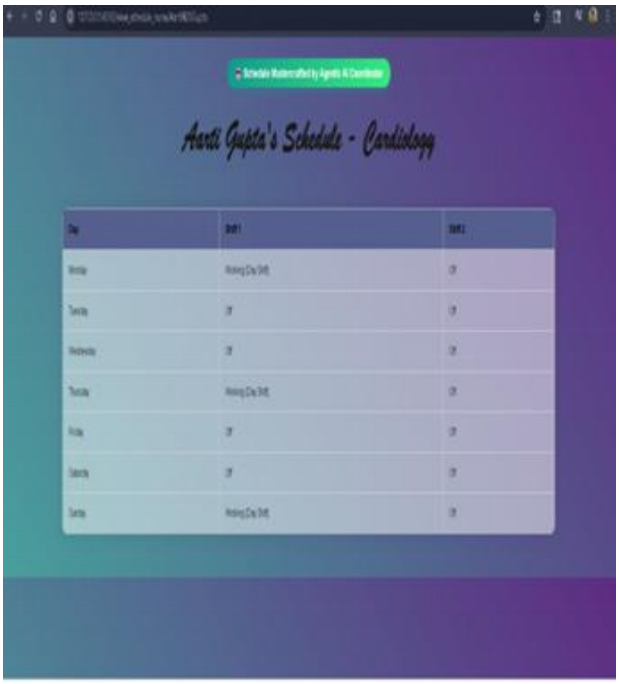

Fig. 8. AI-generated weekly schedule for Nurse Aarti Gupta (Cardiology Department). The schedule correctly differentiates Working (Day Shift) and Off days, generated and labelled by the Agentic AI Coordinator.

Role-based dashboards — Doctor Homepage, Nurse Home- page, and Admin Homepage — are fully operational with correct navigation to schedule viewing, leave requesting, and administrative functions. Personal schedule views (e.g., "Aarti Gupta's Schedule — Cardiology" and "Dr. Sunil Verma's

Schedule — Cardiology") correctly display Day and Night shift assignments per day across the week, as shown in Fig. 5 and Fig. 8.

The system's final assessment scorecard is presented in Table X.

Table X. Final System Assessment Scorecard

Criterion	Score	Note
Core Functionality	8/10	All primary features complete
AI Integration Quality	9/10	GPT-3.5 + fallback robust
Database Architecture	8/10	Clean separation of concerns
Code Quality	6/10	Needs security hardening
Security Posture	4/10	Remediation plan established
Scalability	7/10	Async improvements needed
User Experience	7/10	Mobile not yet addressed
Documentation	9/10	Comprehensive report
Deployment Readiness	5/10	Security hardening required
Overall	**7/10**	Strong foundation

XI. DEPLOYMENT AND FUTURE WORK

The recommended production deployment targets Ubuntu 22.04 LTS with Python 3.10+, MongoDB 6.0+ (replica set), Redis 7.0+, Gunicorn (4+ workers), and Nginx 1.24+ with SSL/TLS via Let's Encrypt. The deployment pipeline com- prises five steps: environment setup (.env secrets), database initialization (scripts in /Database), Gunicorn launch, Ng- inx reverse proxy configuration, and monitoring setup.

Planned Phase 2 enhancements include:

- Upgrade to GPT-4o for enhanced constraint handling and preference learning.
- Native iOS/Android mobile apps via React Native with WebSocket push notifications.
- HL7/FHIR integration with Hospital Information Systems and OAuth 2.0 SSO.
- Docker/Kubernetes containerization with CI/CD via GitHub Actions.Predictive leave analytics dashboard for proactive backup staffing.
- Predictive leave analytics dashboard for proactive backup staffing.

XII. CONCLUSION

This paper presented the Hospital Workforce Schedul- ing and Management System — a production-oriented, AI- powered web application that automates weekly shift schedul- ing, leave management, and staff administration across mul- tiple hospital departments. The dual-mode scheduling engine, combining GPT-3.5-turbo with a deterministic Round-Robin fallback, ensures both intelligent and resilient operation.

The system achieves an estimated 80% reduction in admin- istrative scheduling effort, generates department-aware sched- ules in under 5 seconds, and delivers a complete leave man- agement lifecycle from submission through replacement and notification. The role-based multi-dashboard interface provides each user type with precisely the tools and data they require, as validated through the deployed screenshots.

With the identified security enhancements and Phase 2 features implemented, this system has significant potential as a reference implementation for AI-powered healthcare work- force management in mid-size hospital environments — par- ticularly in developing markets where commercial alternatives are cost-prohibitive.

REFERENCES

1. E. K. Burke, P. De Causmaecker, G. V. Berghe, and H. Van Landeghem, "The state of the art of nurse rostering," Journal of Scheduling, vol. 7, no. 6, pp. 441–499, 2004.
2. G. K. Palshikar and R. Srinivasan, "Intelligent scheduling in hospitals: A review of AI approaches," Journal of Healthcare Informatics Research, vol. 4, no. 2, pp. 112–138, 2020.
3. A. Z. Ghalwash and M. M. Aref, "AI-driven nurse scheduling: A GPT- based approach," International Journal of Advanced Computer Science and Applications, vol. 12, no. 8, pp. 214–222, 2021.
4. G. K. Palshikar, "Application of machine learning to hospital workforce planning," Proc. International Conference on Healthcare Informatics, pp. 45–52, 2019.
5. OpenAI, "GPT-3.5-turbo model documentation," OpenAI API Ref- erence, 2023. [Online]. Available: https:/ /platform.openai.com/docs/ models/gpt-3-5
6. Pallets Projects, "Flask — A Python microframework," 2024. [Online]. Available: http s://flask.palletsprojects.com/
7. MongoDB Inc., "MongoDB manual — Data modeling," 2024. [Online]. Available: https ://www.mongodb.com/docs/manual/dat a-modeling/
8. PyMongo Contributors, "PyMongo 4.x documentation," 2024. [Online]. Available: https:// pymongo.readthedocs.io/
9. OWASP Foundation, "OWASP top ten security risks," 2023. [Online]. Available: https: //owasp.org/www-project-top-ten/
10. Gunicorn Contributors, "Gunicorn — Python WSGI HTTP server for UNIX," 2024. [Online]. Available: https: //gunicorn.org/
11. Nginx Inc., "Nginx documentation — Reverse proxy guide," 2024. [Online]. Available: https:/ /nginx.org/en/docs/
12. Pallets Projects, "Werkzeug security — Password hashing utilities," 2024. [Online]. Available:https: //werkzeug.palletsprojects.com/
13. Celery Contributors, "Celery — Distributed task queue," 2024. [Online]. Available: https://docs.celeryq.dev/
14. Locust Contributors, "Locust — Open source load testing tool," 2024. [Online]. Available: https:/ /locust.io/