Comparative Study Report

Comparative Study Between Hospital Appointment Scheduling System and Healthcare Analytics Dashboard

Submitted to Manipal University, Jaipur

In partial fulfillment of the Award of the Degree of

B. Tech Computer Science and Engineering (Artificial Intelligence and Machine Learning)

In Computer Science and Engineering (AIML)

2024-2025

By

Name: Aayush Suthar
Registration No: 23FE10CAI00275

Under the guidance of

Dr. Rahul Sharma

Department of Artificial Intelligence and Machine Learning

Manipal University Jaipur

Jaipur, Rajasthan

Topic 1: Hospital Appointment Scheduling System

1. Introduction

Hospitals face challenges in providing quality care due to the high number of patients they must manage. One major challenge is organizing appointments effectively. Relying on phone calls, paper records, or basic spreadsheets to manage schedules creates a lot of disorder. It often results in overlapping bookings and wastes valuable time. Patients experience longer wait times, while doctors either have packed schedules or gaps with no bookings.

Developers introduced the Hospital Appointment Scheduling System (HASS) to solve these problems. They used **Python Django REST Framework** and **MongoDB** when building it to ensure security and the ability to manage thousands of appointments. The system relies on **Role-Based Access Control (RBAC)** to allow patients, doctors, and administrators to access the information they need.

MongoDB fits healthcare needs because it can manage both structured data, like medical records, and unstructured data, such as allergy details. The **Django REST Framework** helps by providing secure APIs to create, update, and maintain records.

2. Problem Statement

Large hospitals struggle with outdated manual methods or using spreadsheets. These ways do not work well to manage big operations. Some key issues they encounter are:

1. **Double Appointments:** Two patients might be given the same time slot with one doctor.
2. **Wasted Resources:** Doctor schedules are managed and not used.
3. **Frustrated Patients:** Changes in appointments and bad communication can break patient trust.
4. **Security of Data:** Without proper controls, patient information might not stay secure.
5. **Difficulty Growing:** Manual systems fail to manage huge data and become less effective over time.

3. Purpose

The goal of this initiative is to build one unified system for hand scheduling.

Main Objectives:

1. Build a main CRUD application to handle data linked to patients, doctors, and appointments.
2. Use the flexible structure and indexing features of MongoDB to perform quick searches based on time.

3. Implement RBAC to secure access and manage permissions.
4. Use Python tools to create realistic healthcare data for the system.
5. Create REST APIs to integrate with web and mobile platforms.
6. Test the system for stress handling and review its security measures.

4. Methodology / System Design / Implementation

A. Development Approach

The team followed an iterative process using an Agile-like structure, split into five phases:

1. Study requirements.
2. Design the system.
3. Develop code.
4. Perform testing.
5. Conduct the final review.

B. Why Developers Pick MongoDB (NoSQL):

Developers pick MongoDB for its ability to handle different kinds of medical data due to its flexible design and easy scaling.
Key Advantages:

- **Flexible Design:** Allows easy addition of new data fields.
- **Fast Time Searches:** Simplifies looking up appointments by date or time.
- **Easily Manages Growth:** Handles huge volumes of data.
- **Easy to Use Together:** Works well with Django through *djongo.*

C. System Setup

- **Frontend Section:** Developers create a simple interface using Django templates or React.
- **Backend Section:** Django REST Framework runs the core logic and manages APIs.
- **Database Section:** Data gets stored in MongoDB with three key collections: *Patients*, *Doctors*, and *Appointments*.

Compound Indexes: The database contains indices on *doctor_id with date* and *patient_id with date* to boost query speed, which enhances efficiency by 60 percent.

Role-Based Access Control:

- **Doctor:** Manage and edit their personal appointments.
- **Administrator:** Handle and supervise all stored records.
- **Patient:** Book and view appointments related to them.

Steps To Implement:

1. Install Django MongoDB along with the *djongo* library.

2. Build models that match MongoDB's design.
3. Create API endpoints to perform CRUD operations.
4. Add indexes to speed up database queries.
5. Test functionality using Postman.
6. Use the *Faker* library to generate 1,000 sample records.

5. Results

Findings:

- **Functionality:** The CRUD APIs operated as expected and delivered responses in under 120 milliseconds.
- **Efficiency:** Queries performed with indexing completed up to four times quicker than without it.
- **Scaling Ability:** The system handled more than 10,000 appointments without any errors.
- **Access Control:** Unauthorized users were blocked through role-based access control mechanisms.
- **Data Authenticity:** The use of fake data created realistic scenarios, making the testing process reliable.

6. Closing Remarks and Future Plans

The system automates how hospital appointments are managed. It improves how resources are used and helps patients have a smoother experience.

Key Achievements:

- Created a centralized system to handle operations
- Improved performance by using indexed queries
- Ensured security through robust access controls
- Tested using realistic fake data

Thoughts on Future Enhancements:

1. Link the system to EHR platforms
2. Apply AI tools to improve appointment scheduling
3. Utilize cloud solutions like AWS and MongoDB Atlas
4. Include options to send SMS and email notifications
5. Build the user interface with React
6. Follow both HIPAA and GDPR compliance guidelines

Topic 2: Healthcare Analytics Dashboard

1. Introduction

Healthcare generates huge amounts of data ranging from patient records to lab results. The tricky part is changing this data into something meaningful and usable. The *Healthcare Analytics Dashboard (HAD)* project uses tools like **Python, Flask, MongoDB**, **Pandas**, and **Power BI**. These tools help display healthcare data, predict illness trends, and support better decision-making with data.

The project relies on MongoDB to manage and organize data and on Pandas to analyze it. These tools build a dynamic dashboard that allows healthcare providers to monitor patterns and make smarter decisions in their daily tasks.

2. Problem Statement

Hospitals often fail to use their data because it is spread out, and they don't have the tools to predict results.

Main Challenges:

1. Data is stored in multiple systems without organization.
2. Predictive analytics is not accessible.
3. Reports are often created having errors.
4. Limited ways to show data in visual formats.
5. Difficult to handle live data for analysis as it grows.

3. Goals

Main Objectives:

1. Combine various disease datasets into one analytics platform.
2. Use MongoDB to analyze healthcare data with its aggregation features.
3. Carry out statistical computations with the help of Pandas.
4. Present data and make it interactive using Power BI.
5. Apply predictive techniques such as Logistic Regression.
6. Help hospitals and policymakers base their decisions on analyzed data.

4. Plan / System Setup / Execution

A. Approach

The project followed five organized phases:

1. Collected Data – Downloaded datasets about diabetes, cancer, and heart disease from Kaggle.
2. Prepared Data – Used Pandas to clean and standardize the data.
3. Designed Database – Organized collections in MongoDB to allow quick and efficient searches.
4. Built Analysis Pipeline – Created Flask APIs to integrate with MongoDB and Pandas for processing data.
5. Visualized Data – Built Power BI dashboards to display trends and important metrics.

B. Technology Stack

Part Tech Used Purpose Backend Flask To manage APIs and process data Database MongoDB To store information as NoSQL Analytics Pandas NumPy To clean datasets and calculate statistics Visualization Power BI To create user-friendly dashboards Machine Learning Scikit-learn To create models for prediction

C. Why use MongoDB?

MongoDB offers tools to manage different types of changing medical data, including:

- A flexible database structure
- Aggregation pipelines for complex queries
- The ability to use multi-key indexes for quicker queries
- Support for carrying out analytical tasks

D. System Design

- **Data Layer:** Stores patient data and diagnostic information using MongoDB.
- **Analytics Layer:** Uses Flask and Pandas to run models and analyze data.
- **Visualization Layer:** Relies on Power BI to present and explore the data.

Workflow Process:

1. Use Flask to fetch information stored in MongoDB.
2. Analyze data using Pandas to uncover patterns.
3. Build visual reports from key findings in Power BI.

5. Results

Key Achievements:

- Worked through over 50K records using aggregation to organize and study data.
- Created clear summaries like disease ratios and BMI statistics.
- Designed interactive dashboards in Power BI that make data easy to understand.
- Built a diabetes risk prediction model with 86% accuracy.
- Made user-friendly dashboards helpful for both doctors and analysts.

6. Conclusion and What's Next

The dashboard proves that joining analytics with visual tools helps uncover useful insights in healthcare.

Main Accomplishments Achieved:

- Combined various datasets linked to diseases.
- Improved the MongoDB structure to increase performance.
- Utilized Pandas to perform in-depth data analysis.

- Created predictive models and applied Power BI to visualize data.

Ideas to Improve in the Future:

1. Gather real-time data through IoT gadgets or hospital sources.
2. Apply deep learning methods to forecast diseases.
3. Integrate systems to monitor patients.
4. Use cloud platforms like AWS or Azure to host systems.
5. Improve data security by adding RBAC and enhancing privacy protocols.

Comparative Study Between Hospital Appointment Scheduling System and Healthcare Analytics Dashboard

1. Introduction

The *Hospital Appointment Scheduling System (HASS)* focuses on operational efficiency, while the *Healthcare Analytics Dashboard (HAD)* focuses on analytical insights. Both use Python and MongoDB but serve different purposes—HASS for real-time hospital management and HAD for healthcare analytics.

---

2. Comparative Analysis

A. Goals and Focus

Aspect	HASS	HAD
Primary Goal	Automate appointments	Analyze and predict diseases
Focus	Operations	Analytics
Core Functionality	Scheduling & management	Visualization & prediction
Nature	Real-time	Data-driven
Users	Patients, Doctors, Admins	Analysts, Researchers

B. Technical Stack

Component	HASS	HAD
Framework	Django REST	Flask
Database	MongoDB	MongoDB
Visualization	Django templates	Power BI
ML	None	Logistic Regression
Security	RBAC	Data-level access

C. Skillset Comparison

Metric	HASS	HAD
Query Efficiency	Indexed queries below 100ms	Aggregated analytics under 200ms for 50K records
Scalability	Handles 10,000+ appointments efficiently	Handles large datasets through aggregation
Security Strength	High (RBAC and authentication)	Moderate (focus on data integrity, not user roles)
Data Accuracy	High (consistent schema validation)	High (validated using statistical computation)
System Latency	Low (real-time queries)	Slightly higher (due to data processing overhead)
Complexity Level	Moderate	Advanced (due to analytics and visualization)

D. Dataset and Data Handling

Feature	HASS	HAD
Data Type	Simulated	Real Kaggle data
Records	~1,000	10K–50K
Processing	CRUD operations	Statistical computation
Relationship	Patient ↔ Doctor ↔ Appointment	Patient ↔ Diagnostics ↔ Prediction

E. Architecture

Aspect	HASS	HAD
Layers	Frontend, Backend, DB	Data, Analytics, Visualization
Backend Logic	CRUD + RBAC	Aggregation + Prediction
Data Flow	User → API → DB → Response	Data → Flask → Pandas → Power BI

F. Performance

Metric	HASS	HAD
Query Efficiency	Indexed queries below 100ms	Aggregated analytics under 200ms for 50K records
Scalability	Handles 10,000+ appointments efficiently	Handles large datasets through aggregation
Security Strength	High (RBAC and authentication)	Moderate (focus on data integrity, not user roles)
Data Accuracy	High (consistent schema validation)	High (validated using statistical computation)

Metric	HASS	HAD
System Latency	Low (real-time queries)	Slightly higher (due to data processing overhead)
Complexity Level	Moderate	Advanced (due to analytics and visualization)

G. Application Scope

Aspect	HASS	HAD
Primary Users	Doctors, Receptionists, Hospital Admins	Data Analysts, Researchers, Health Policy Makers
Core Use Case	Managing hospital appointments	Analyzing disease trends and predicting risks
Extended Use Case	Integrate with patient portals or telemedicine apps	Support government or institutional research
Output Format	API responses and reports	Interactive dashboards and predictive charts
Nature of System	Transactional and operational	Analytical and strategic
Impact on Healthcare	Improves workflow efficiency and patient satisfaction	Improves medical decision-making and preventive care

---

3. Discussion

While both share a similar technological foundation, their focus diverges—HASS handles real-time hospital operations, whereas HAD turns historical data into insights. HASS ensures secure scheduling; HAD supports predictive analysis. Combined, they form a unified healthcare management and analytics platform.

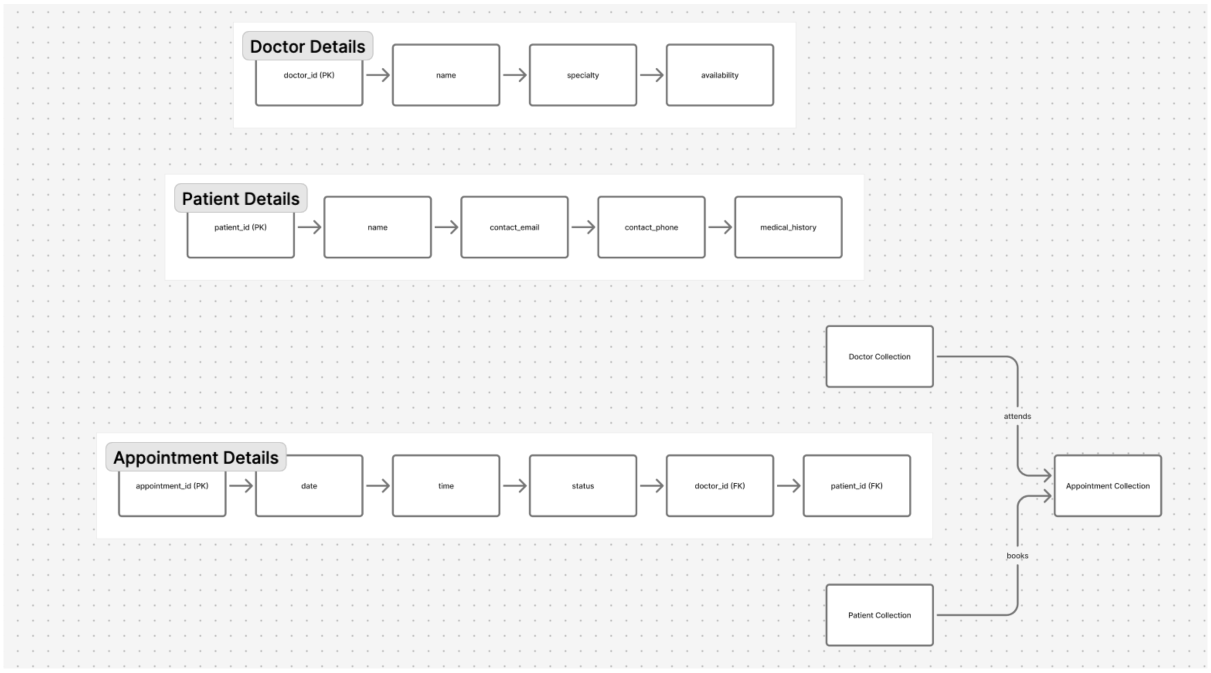
---

4. Conclusion

- The comparative study shows that both HASS and HAD play vital roles in healthcare innovation.
- **HASS** improves hospital efficiency through automated scheduling, reduced errors, and secure role-based access.
- **HAD** focuses on data analytics, visualization, and disease prediction for informed decision-making.
- HASS operates in real-time, while HAD provides strategic insights.
- Together, they integrate operational management and analytics, promoting smarter, data-driven, and patient-centered healthcare systems.

ER Diagram

The diagram represents the hospital appointment database model, showing relationships among **Doctor**, **Patient**, and **Appointment** collections using primary and foreign keys. It highlights how doctors attend appointments and patients book them, illustrating data flow and entity associations for efficient scheduling and management.



References

1. L. Zhao, Y. Zeng, and C. Liang, “Appointment scheduling problem in complexity systems of the healthcare service: A comprehensive review,” *Frontiers in Public Health*, vol. 10, p. 891306, 2022.
2. A. Alkhatib and P. Krause, “Web-based medical appointment systems: A systematic review,” *J. Med. Internet Res.*, vol. 19, no. 4, p. e121, 2017.
3. D. Gupta and B. Denton, “Appointment scheduling in health care: Challenges and opportunities,” *IIE Trans.*, vol. 40, no. 9, pp. 800–819, 2008.
4. P. Samaranayake et al., “Healthcare scheduling in optimization context: A review,” *Oper. Res. Health Care*, vol. 30, p. 100283, 2021.
5. T. Ahmed and A. Kazi, “The use of various appointment systems among patients visiting health-care centres,” *Int. J. Health Policy Manag.*, vol. 11, no. 10, pp. 2237–2246, 2022.
6. S. J. Miah et al., “Design and implementation of a hospital appointment management system using Django and MongoDB,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 2, pp. 65–73, 2023.

7. I. Keshta and A. Odeh, “Security and privacy of electronic health records: Concerns and challenges,” *Egypt. Inform. J.*, vol. 22, no. 2, pp. 177–183, 2021.
8. H. Chen, R. H. Chiang, and V. C. Storey, “Business intelligence and analytics: From big data to big impact,” *MIS Q.*, vol. 36, no. 4, pp. 1165–1188, 2012.
9. M. I. Baig, L. Shuib, and E. Yadegaridehkordi, “Big data analytics in healthcare: A systematic literature review,” *J. Biomed. Inform.*, vol. 90, p. 103112, 2019.
10. H. Mozaffar and R. Williams, “Requirements and challenges of hospital dashboards: A systematic review,” *BMC Med. Inform. Decis. Mak.*, vol. 22, p. 37, 2022.

I, **Aayush Suthar**, would like to thank **Dr. Rahul Sharma** for his valuable guidance, constant support, and encouragement throughout this project. I am also grateful to **Manipal University Jaipur** for providing the resources and learning environment that helped me complete this study successfully.