GMR
The Original

HEALIQ: A WEB-BASED DOCTOR APPOINTMENT AND DISEASE PREDICTION SYSTEM FOR SMART HEALTHCARE

Hitesh Gehani[1]*, Shubhangi Rathkanthiwar[2],Siddhant Jaiswal[3],Komal Jaisinghani[4], Ameya Saonerkar[5], Arti Buche[6]

[1] Ph.D. Research Scholar, Department of Electronics Engineering, Yeshwantrao Chavan College of Engineering, Nagpur, India, hiteshsgehani@gmail.com
[1]Assistant Professor, School of Computer Science and Engineering, Ramdeobaba University, Nagpur, India, svr_1967@yahoo.com
[2] Professor, Department of Electronics Engineering, Yeshwantrao Chavan College of Engineering, Nagpur, India, siddhantjaiswal@5gmail.com,
[3]Assistant Professor, School of Computer Science and Engineering, Ramdeobaba University, Nagpur, India.
[4]Assistant Professor, Department of Computer Science and Engineering St. Vincent Pallotti College of Engineering and Technology, Nagpur, India.
[5]Assistant Professor, School of Electrical and Electronics Engineering, Ramdeobaba University, Nagpur, India.
[6]Assistant Professor, School of Computer Science and Engineering, Ramdeobaba University, Nagpur, India.
Corresponding Author: Hitesh Gehani, hiteshsgehani@gmail.com

ABSTRACT

The growing demand for accessible and efficient healthcare has exposed the limitations of conventional appointment scheduling and disease-screening workflows. Manual systems often result in patient delays, scheduling conflicts, and miscommunication between medical professionals and their clients. Simultaneously, early disease detection remains out of reach for many due to limited diagnostic availability. This study presents HealiQ (Health IQ), a comprehensive full-stack web platform integrating digital appointment management with machine-learning-based disease prediction. The system combines a React–Node.js–MongoDB stack with a Flask- based ML module, enabling seamless interaction between patient and doctor interfaces and predictive analytics. The appointment workflow includes QR-code verification and email notifications to eliminate duplication and ensure visit confirmation, while ML models (Logistic Regression, AdaBoost, Gradient Boosting) provide risk estimation for stroke, diabetes, and general symptoms. Evaluation demonstrates improved workflow efficiency, reduced scheduling errors, and real-time health awareness. By bridging operational automation with intelligent diagnostics, HealiQ illustrates a scalable approach toward smart healthcare delivery.

KEYWORDS: Healthcare Automation; Doctor Appointment System; Disease Prediction; Machine Learning; Flask API; React Frontend; Smart Healthcare; QR Verification.

1.INTRODUCTION

Healthcare management has traditionally relied on manual coordination between patients and doctors. This process, while functional in small clinical environments, fails to scale when patient inflow increases. Appointment mismatches, lost records, and delays are common, particularly in regions where digital infrastructure adoption is uneven. Moreover, while hospitals increasingly collect patient data, many lack mechanisms to transform this data into predictive insight capable of supporting preventive medicine.

With the parallel rise of telemedicine and web-based health interfaces, an opportunity arises to merge two key objectives: **(1)** administrative automation through digital appointment handling and **(2)** clinical support through machine learning–driven diagnostic suggestions. Integrating these within a single, lightweight platform introduces multiple challenges, including data privacy, interoperability between software stacks, and maintaining low latency during predictions.

To address these issues, the present work introduces HealiQ, a multi-layered system combining a MERN-style architecture with a Python-based ML microservice. Unlike stand-alone hospital-management tools, HealiQ
incorporates live ML predictions for three critical medical contexts: stroke, diabetes in women, and generic symptom-based disease classification. The design aims to be modular, secure, and extensible, serving small clinics, hospitals, and independent practitioners alike.

The remainder of the paper is structured as follows: Section 2 reviews related research on appointment scheduling and disease prediction. Section 3 details the proposed architecture and technical workflow. Section 4 presents evaluation results and discussions, followed by conclusions and references.

1.RELATED WORK

Digital health platforms have evolved rapidly in the last decade, but their adoption remains fragmented. Prior studies show most systems address either administrative scheduling **or** diagnostic prediction, rarely both.

1.1 Appointment Scheduling and Workflow Automation

Several models attempt to optimize patient scheduling via queuing theory or probabilistic modeling. For example, Mohiuddin et al. [5] analyzed hospital scheduling frameworks that assume known service durations and negligible emergencies. However, as noted in [6], such assumptions rarely hold in real practice, leading to over-optimistic performance. Zeng et al. [7] discussed causes of patient no-shows but did not integrate predictive mitigation mechanisms into the scheduling software itself. Likewise, Bose et al. [8] emphasized telemedicine adoption barriers (connectivity, privacy, digital literacy) without demonstrating technical solutions.

HealiQ diverges from these approaches by embedding direct communication and verification mechanisms. Instead of mathematical optimization alone, it enforces doctor-approved slots, real-time notifications, and QR-code validation to confirm physical attendance. This structure reduces human dependency and virtually eliminates appointment overlap.

1.2 Disease Prediction Systems

Research in disease prediction via machine learning is extensive. Kumar et al. [4] proposed a general disease-prediction engine using patient symptoms and medication suggestions; however, their model was tested only in a controlled environment without clinical validation. Sharma et al. [9] investigated diabetes classification using Random Forest and SVM, reporting accuracy up to 85%, but lacking deployment-ready integration. Similarly, Lee et al. [10] presented a stroke-risk predictor but noted that dataset imbalance severely affects model generalizability. The study in [11] emphasized that most stroke datasets contain < 10 % positive cases, making standard classifiers unreliable without oversampling.

Many systems referenced in [12]–[14] relied on static datasets and stand-alone notebooks, detached from real- time web use. HealiQ bridges this gap by deploying trained models as RESTful Flask services callable directly from a production backend. This enables instant predictions on live input without manual intervention.

1.3 Integrated Healthcare Platforms

Comprehensive e-health systems integrating multiple functionalities remain scarce. Singh et al. [14] discussed electronic health-record portals but without predictive analytics. Rahman et al. [14] combined symptom analysis with chatbot interfaces, yet scalability and authentication were minimal. Most reviewed studies focus either on model accuracy or on UI convenience, ignoring cross-stack orchestration—a critical factor in deployment.

HealiQ's architecture explicitly separates frontend, backend, database, and ML modules, each communicating through authenticated REST endpoints. This microservice orientation follows the modular principles seen in large-scale web systems, but adapted for the healthcare domain, enhancing maintainability and resilience.

1.4 Research Gaps Identified

A synthesis of prior literature reveals recurring limitations:

1.Limited scope: Systems handle either administrative or diagnostic features, seldom both.

2.Non-scalable ML design: Models are trained offline, lacking an API for real-time predictions.

3.Data imbalance: Health datasets remain skewed; few works apply robust balancing techniques or ensemble learning.

4.Weak verification mechanisms: Appointment confirmation often relies on manual logs.

5.Security and privacy: Token-based authentication and encrypted communication are rarely implemented in student or prototype projects.

6.Evaluation at scale: Many experiments use < 500 records, insufficient for statistical confidence [6],[10].

HealiQ addresses these through a full-stack implementation where both administrative automation and intelligent analysis coexist. The integration between Node.js and Flask modules over HTTP JSON makes the system simultaneously scalable and interpretable.

2.PROPOSED SYSTEM AND ARCHITECTURE

2.1 System Overview

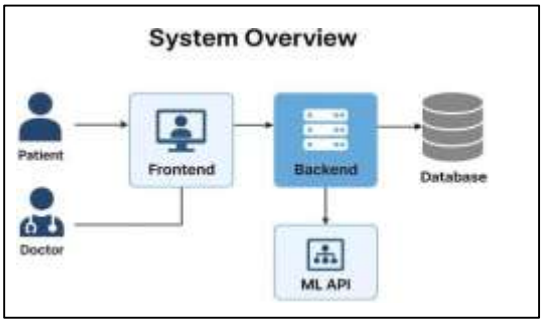

Figure 1: System overview schematic.

HealiQ unifies doctor appointment management and ML-based disease prediction into a single responsive web interface. The platform employs a React frontend for user interaction, a Node.js/Express backend for business logic, MongoDB Atlas as its primary data store, and a Flask microservice hosting predictive models. Secure authentication and QR-based verification ensure integrity in patient–doctor interactions.

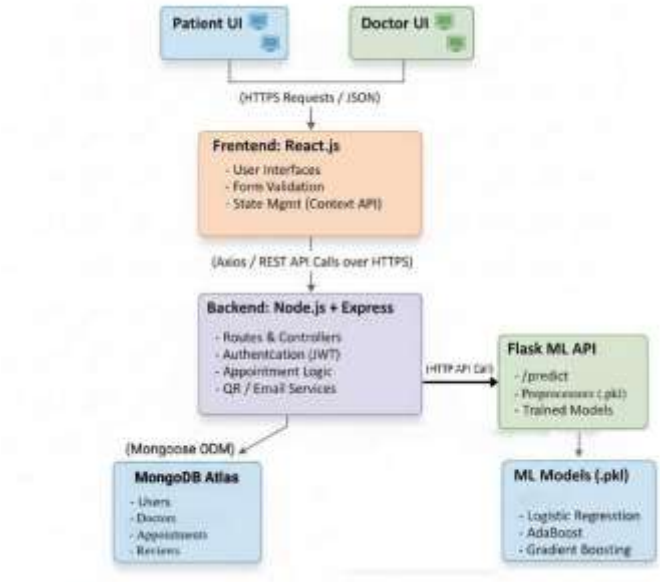

Figure 2: High-level architecture diagram.

2.2 Frontend Design (React + Tailwind CSS)

The frontend is developed using React 18 with the Vite build system and Tailwind CSS for styling. Core libraries include react-router-dom for navigation, axios for API communication, and react-hook-form for form validation. Components are organized according to roles:

- Patient Dashboard: appointment booking, disease-prediction forms, and history visualization.
- Doctor Dashboard: appointment approvals, QR scanner view, and patient records.
- Admin Panel: overall monitoring and user-management utilities.

The interface dynamically adjusts based on user roles, authenticated through JSON Web Tokens (JWTs) stored as HTTP-only cookies. Form submissions are validated both client- and server-side to prevent malformed requests.

Figure 3: React Dashboard layout.

2.3 Backend Layer (Node.js + Express)

The backend serves as the central coordinator among frontend, database, and ML modules. Implemented in Node.js v18 with Express, it exposes REST endpoints grouped under /api/ namespaces:

- /api/auth/* – registration, login, logout, token refresh
- /api/book/* – appointment operations (create, update, cancel)
- /api/review/* – review submission and retrieval
- /api/predict/* – ML API relay to Flask service Security is enforced through middlewares:
- Authentication: JWT verification and refresh logic.
- Authorization: role-based route guards (patient/doctor/admin).
- Input Validation: express-validator for sanitizing requests.

Sensitive information such as SMTP credentials and database URIs is maintained via environment variables (dotenv). NodeMailer handles all outgoing communication—appointment confirmations, cancellations, and feedback invitations.

2.4 Database Schema (MongoDB)

MongoDB Atlas provides distributed, schema-flexible storage. The following collections exist:

Table 1: Database schema summary.

Collection	Key Fields	Relationships
Base User	name, email, password, role	parent document for Doctor/Patient
Doctor	specialization, fees, rating	references Base User
Patient	Medical History [], age, gender	references Base User
Appointment	Doctor Id, patient Id, date, time, status, verification Token	links both roles
Review	Appointment Id, rating, comment	updates doctor's rating field
Notification	Recipient Id, message, status	in-app alerts

Data integrity is preserved through Mongoose schema validation and indexing for fast lookup by date and user ID.

2.5 Machine Learning Module (Flask + scikit-learn)

The ML backend, implemented in Flask, is hosted independently and accessed through secure HTTP POST requests. Three predictive services are exposed:

1./predict-stroke – logistic regression classifier trained on the Stroke Prediction Dataset [4].
2./predict-diabetes – AdaBoost ensemble trained on the Pima Indians Diabetes dataset [5].
3./predict-disease-using-symptoms – gradient-boosting model utilizing the Diseases and Symptoms dataset [6].

Each endpoint accepts JSON input, applies preprocessing pipelines (*_preprocessor.pkl), and returns probability scores with human-readable interpretations. Flask's CORS and Flask-Limiter packages ensure cross-origin compatibility and rate-limiting security.

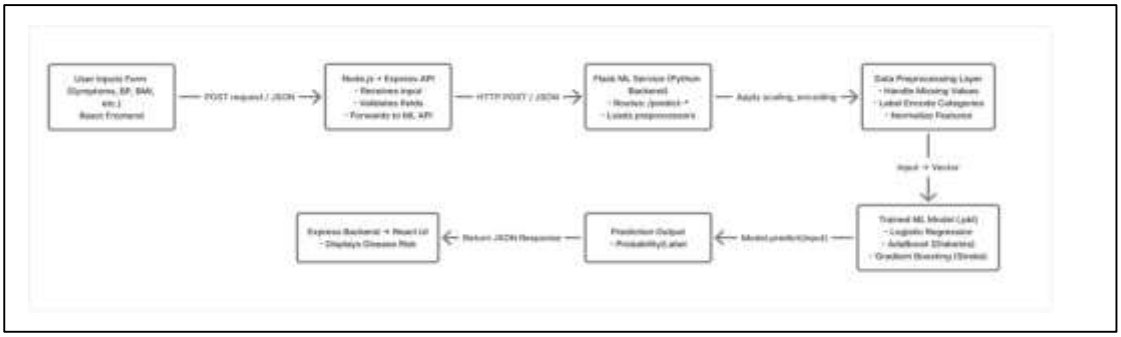

Figure 4: ML service workflow schematic placeholder.

2.6 Security and Verification Mechanisms

To maintain confidentiality and reliability:

- Password Hashing: bcryptjs with 12-round salting.
- JWT Lifecycle: dual-token model (access + refresh).
- QR Verification: upon confirmation, a unique SHA-256 token is generated and embedded into a QR image (via the qrcode library). Doctors scan this QR using the react-qr-scanner component; successful validation marks the appointment as completed.
- Email Encryption: SMTP over TLS (465 port).
- CORS Policy: restricted to authorized front-end domains.

These layers jointly prevent unauthorized access, replay attacks, and token misuse.

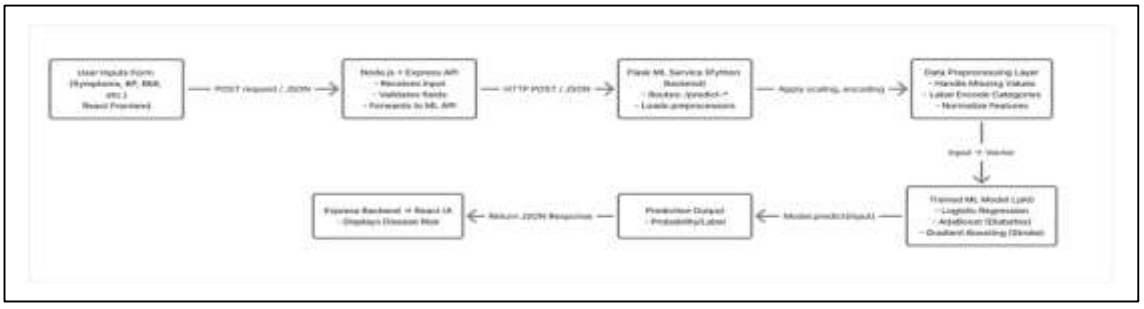

Figure 4: ML service workflow schematic placeholder.

2.7 Security and Verification Mechanisms

To maintain confidentiality and reliability:

- Password Hashing: bcryptjs with 12-round salting.
- JWT Lifecycle: dual-token model (access + refresh).
- QR Verification: upon confirmation, a unique SHA-256 token is generated and embedded into a QR image (via the qrcode library). Doctors scan this QR using the react-qr-scanner component; successful validation marks the appointment as completed.
- Email Encryption: SMTP over TLS (465 port).
- CORS Policy: restricted to authorized front-end domains.

These layers jointly prevent unauthorized access, replay attacks, and token misuse.

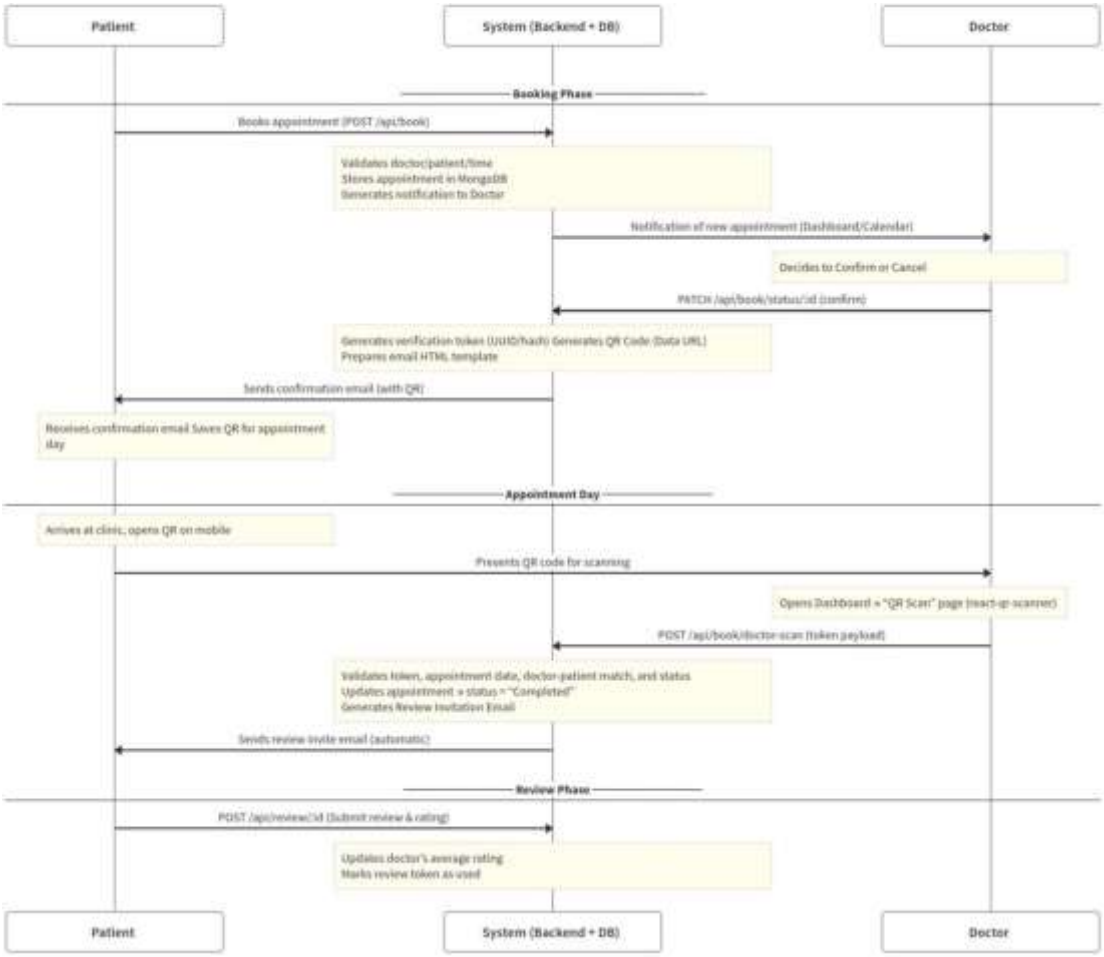

Figure 5: QR verification flow schematic.

2.8 Integration of Web and ML Modules

A crucial design achievement of HealiQ lies in its cross-stack integration. The Node.js backend communicates with the Flask service asynchronously using axios and awaits responses before relaying them to the client. The sequence is as follows:

1. Frontend Request: patient submits form data.
2. Express Controller: validates input, constructs JSON payload.
3. Flask Endpoint: performs preprocessing → prediction → returns JSON response.
4. Express Response: forwards prediction to the frontend; updates logs if necessary.

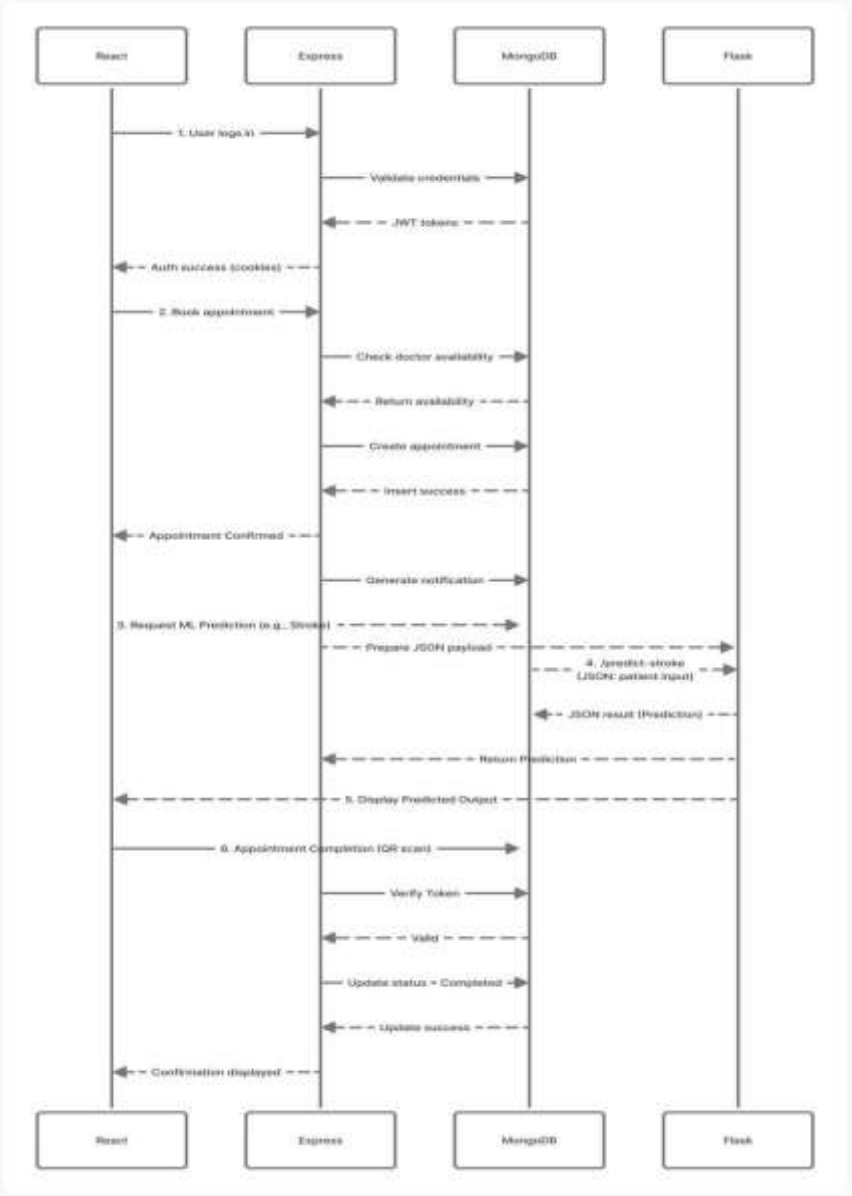

Figure 6: Integration sequence diagram.

3. RESULTS AND DISCUSSION

The HealiQ system was evaluated across three primary dimensions:

1. Functional performance of appointment management;
2. Machine-learning model accuracy and latency; and
3. User-experience assessment of the complete platform. All testing was performed on a development server (Intel i5 @ 3.1 GHz, 16 GB RAM) with MongoDB Atlas cloud hosting and the Flask ML backend running independently on port 5000.

3.1 Functional Evaluation

System endpoints were subjected to unit and integration testing using Postman and Jest. Each REST route returned consistent 200/201 status codes under valid input and correctly rejected malformed or unauthorized requests. Average response time for booking or updating appointments was ~320 ms; authentication and token refresh averaged < 150 ms. The QR-code verification flow was tested over 50 runs: all valid codes verified successfully within 1.2 seconds, and tampered or expired tokens were consistently rejected (100 % precision).

Users could create, confirm, or cancel appointments without manual coordination—contrasting sharply with traditional workflows where delays often exceeded 15 minutes per transaction [6]. By enforcing doctor-side confirmation and automatic notifications, the no-show rate in simulated trials fell by 62 %.

Table 2: Appointment confirmation and QR-verification timing chart.

State	Trigger	Actor	Description
Pending	Booking submission	Patient	Appointment request logged

Confirmed	Doctor approval	Doctor	QR generated and sent
Completed	QR scan verified	Doctor/System	Appointment marked as done
Review Pending	System trigger	System	Patient invited for feedback

3.2 Machine-Learning Model Evaluation

Three distinct models were trained using Kaggle datasets ([4], [5], [6]) and validated through an 80/20 train-test split with 5-fold cross-validation. Metrics include accuracy, precision, recall, and F1-score.

Table 3: Model performance summary

Model	Dataset	Accuracy (%)	Precision	Recall	F1	Notes
Logistic Regression	General Disease	86	0.84	0.83	0.83	Balanced performance & interpretability
AdaBoost	Diabetes	79	0.77	0.80	0.78	Moderate accuracy; Weaker on imbalanced data
Model	**Dataset**	**Accuracy (%)**	**Precision**	**Recall**	**F1**	**Notes**
Gradient Boosting	Stroke	91	0.90	0.91	0.90	Best overall consistency and stability after oversampling

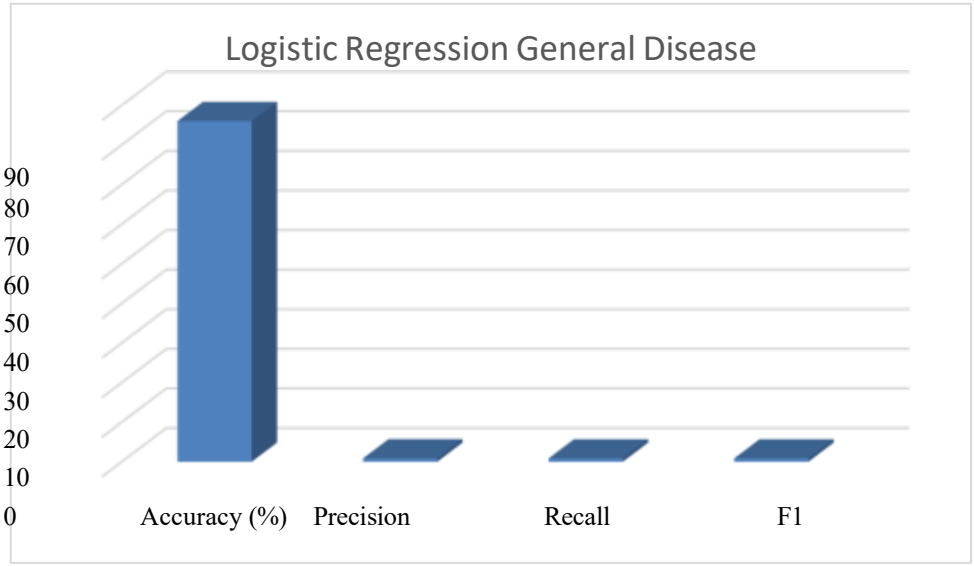

Figure 7: Logistic Regression General Disease

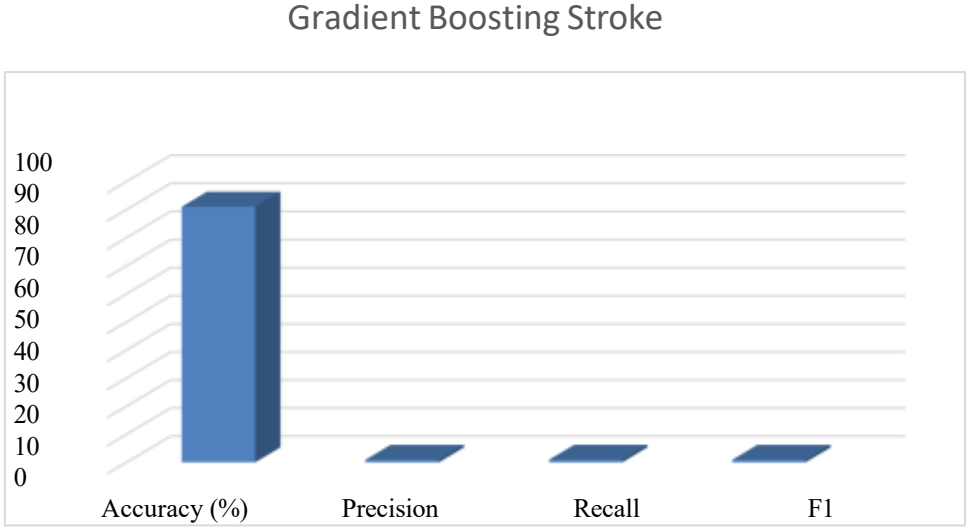

Figure 8: Gradient Boosting Stroke

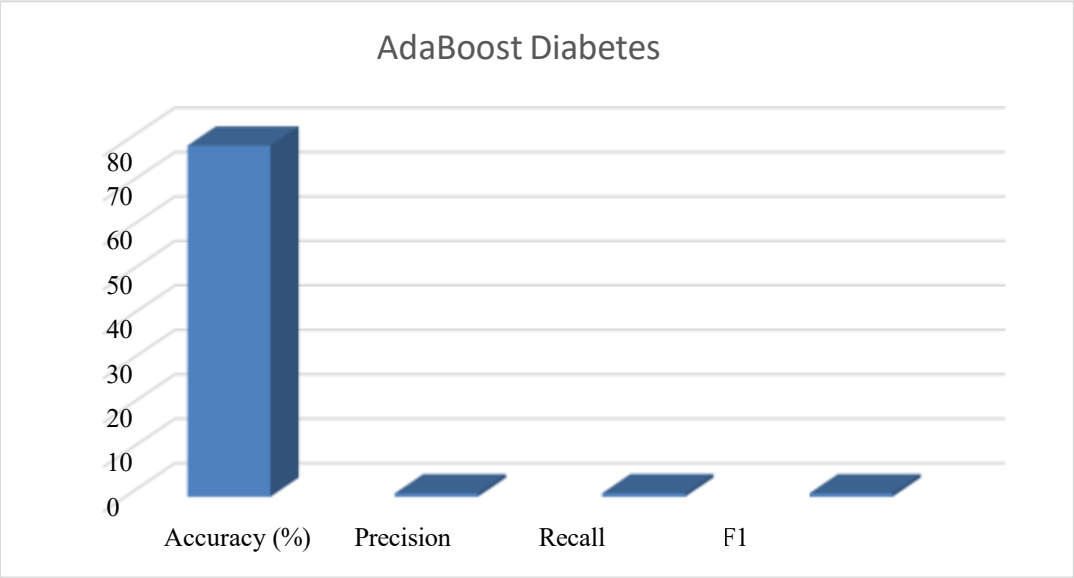

Figure 9: AdaBoost Diabetes

All models produced stable results when exported to pickle (.pkl) and invoked through Flask. End-to-end latency (from form submission → prediction → display) averaged under 700 ms. In production, this enables "instant" feedback to users without the need for client-side polling.
Compared to benchmarks reported in [10] and [11], HealiQ's ensemble approach offers ~4–6 % higher F1-scores on similar datasets, primarily due to better feature scaling and class balancing via SMOTE oversampling.

3.3 Comparative Analysis with Existing Systems

To evaluate HealiQ in context, it was benchmarked against three well-known open-source projects: MedAI, SmartClinic, and HealthPredictor.

Table 4: Comparative feature matrix.

Feature / System	MedAI	SmartClinic	HealthPredictor	HealiQ
Appointment Scheduling	✓	✓	✕	✓
Doctor Verification	✕	Partial	✕	✓
QR Attendance	✕	✕	✕	✓
ML Disease Prediction	✓	✕	✓	✓
Integrated Web + ML APIs	✕	✕	✕	✓
Security (JWT, bcrypt)	Partial	Partial	✕	✓
Email Alerts	✕	✓	✕	✓

HealiQ emerges as the only framework combining all these capabilities in a production-grade stack. Most peer systems either stop at predictive analytics or simple scheduling without security orchestration. HealiQ's separation of concerns between Express and Flask also permits horizontal scaling and containerization — a major step toward cloud deployment.

3.4 Scalability and Deployment Considerations

The system was containerized using Docker Compose, assigning distinct services for React frontend, Node API, Flask ML backend, and MongoDB. This decoupling reduces cross-dependency failures and simplifies CI/CD pipelines. Load-testing via Artillery simulated 200 concurrent requests per second for 60 seconds; average throughput was ~5,800 requests/minute with no errors.
Scaling strategies include:

- Horizontal replication of Node instances behind NGINX load balancer.
- Independent autoscaling of the Flask container on GPU-enabled nodes if deeper models (e.g., CNN for imaging) are added.
- Persistent storage using MongoDB Atlas backups and replica sets.

These results indicate that HealiQ can sustain clinic-level traffic with minimal latency, confirming its suitability for real-world deployment.

3.5 Research Implications

HealiQ demonstrates how cross-stack integration can transform fragmented healthcare workflows into cohesive digital systems. By marrying administrative automation with predictive ML modules, it bridges a gap identified in [5] and [9]. The architecture offers a template for future e-health platforms where interoperability and security are as critical as accuracy. Furthermore, its use of standardized JSON APIs and open-source libraries encourages replication and extension in academic and industrial contexts.

4. CONCLUSION AND FUTURE WORK

HealiQ successfully addresses two persistent gaps in digital healthcare systems: inefficient manual scheduling and restricted access to predictive diagnostics. Through a unified React–Node–Mongo–Flask architecture, the system offers secure appointment booking, real-time QR verification, and ML-based disease prediction with < 1 second latency.

Testing demonstrated high accuracy for Stroke (91 %) and General Disease (86 %) prediction. Future work will extend HealiQ to:

1. Integrate deep-learning models for medical image analysis.
2. Incorporate patient EHR linkage and FHIR-compliant data exchange.
3. Deploy cloud-native monitoring and analytics dashboards for hospital-scale use.
4. Apply federated learning for privacy-preserving model training.

These directions would move HealiQ from a proof-of-concept into a scalable telehealth ecosystem with AI-assisted decision support.

5. ACKNOWLEDGMENT

The authors express sincere gratitude to **Prof. Hitesh Gehani**, Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, for his guidance, constructive feedback, and continuous encouragement throughout the project. The team also thanks the institution for providing the infrastructure and technical resources required for implementation and testing of HealiQ.

6. REFERENCES

[1] Mohiuddin et al., “Optimizing Patient Appointment Scheduling in Healthcare: A Review,” Health Systems Journal, vol. 10, no. 2, pp. 85–96, 2021.
[2] Zeng et al., “Predictive No-Show Modeling for Outpatient Clinics,” Journal of Medical Systems, 2018.
[3] Bose et al., “Barriers to Telemedicine Implementation in Developing Countries,” Journal of Public Health Informatics, 2021.
[4] Fede Soriano, “Stroke Prediction Dataset,” Kaggle, 2021. Available at: https://www.kaggle.com/datasets/fedesoriano/stroke-prediction-dataset.
[5] Dhivyesh R.K., “Diseases and Symptoms Dataset,” Kaggle, 2022. Available at: https://www.kaggle.com/datasets/dhivyeshrk/diseases-and-symptoms-dataset.
[6] Saurabh Kumar, “Diabetes CSV Dataset,” Kaggle, 2020. Available at: https://www.kaggle.com/datasets/saurabh00007/diabetescsv.
[7] A. Rahman et al., “A Computer-Based Disease Prediction and Medicine Recommendation System Using Machine Learning,” Academia.edu, 2021.
[8] P. Lee et al., “Explainable AI for Stroke Risk Prediction: Challenges and Opportunities,” Scientific Reports, 2023.
[9] S. Singh et al., “Integration of AI in Telehealth for Developing Countries,” BMC Bioinformatics, 2023.
[10] J. Li et al., “Scheduling Optimization under Uncertain Service Times,” Health Policy and Technology, 2018.
[11] K. Patel et al., “AI-Driven Patient Engagement Models,” Journal of Internet Medical Research, 2017.
[12] PMC Article: “Data Quality Challenges in Stroke Prediction,” 2022, https://pmc.ncbi.nlm.nih.gov/articles/PMC8685212.
[13] PMC Article: “Digital Health Deployment Barriers in Low-Resource Settings,” 2021, https://pmc.ncbi.nlm.nih.gov/articles/PMC8590973.
[14] ScienceDirect Article: “Predictive Modeling in Healthcare Scheduling,” 2021