

AI Agent-Based Multi-Stage Appointment Scheduling System for Clinics

Hor Ying Huai, Suhail S.M. Alqrinawi*

Faculty of Artificial Intelligence and Cyber Security, Universiti Teknikal Malaysia Melaka (UTeM),
Malacca, Malaysia

*Corresponding Author

DOI: <https://doi.org/10.51584/IJRIAS.2026.11080145>

Received: 02 September 2026; Accepted: 07 September 2026; Published: 19 September 2026

ABSTRACT

Many clinics still use phone calls to schedule general consultations, vaccinations and blood tests. This process is slow and becomes more difficult for multi-stage vaccinations because each dose must follow the required interval. Workflow analysis shows that clinic staff spend significant time reviewing patient records before confirming appointments. In a simulated clinic environment with 100 reservations per week, manual scheduling produced appointment conflicts and increased scheduling time. This paper introduces an AI Agent-Based Multi-Stage Appointment Scheduling System for Clinics that checks appointment availability, balances doctor workloads, and validates vaccine dose schedules to address the lack of multi-stage scheduling and workload balancing in clinic appointment systems. A Telegram bot for patients, a Next.js/React web portal for clinic staff, and a Python FastAPI backend with PostgreSQL are used. An LLM race method accesses a local DeepSeek model and the Google Gemini API simultaneously to minimize response time while a LangGraph agent evaluates scheduling rules, doctor workload, and vaccination dependencies. The system was tested using 80 Pytest-based regression, smoke, monkey, scheduling, booking success rate, workload balancing, AI extraction, communication agent, and vaccine validation tests and 132 manual black-box functional test cases across 19 functional modules on the web portal and Telegram bot. All automated validation tests passed 100%, and manual functional verification passed 99.2% (131 of 132 cases). Results show that the proposed system addresses the main limitations of existing clinic scheduling systems by providing AI-assisted booking, workload balancing, and multi-stage vaccine validation, while allowing clinic staff to review and manage all appointments through the administrative dashboard.

Keywords AI agents, appointment scheduling, multi-stage dependency, Telegram bot, clinic management system

INTRODUCTION

Medical clinics depend on an efficient appointment system to manage patient visits. Many clinics still use phone calls or messaging apps to schedule general consultations, vaccinations and blood tests. This process takes time for both staff and patients. During system analysis, it was observed that clinic staff often spend a large part of their working hours checking doctor availability, updating bookings and replying to repeated patient enquiries. For example, in a simulated clinic configured with about 100 total appointments each week, manual scheduling increased the risk of double bookings and uneven doctor workloads during busy periods.

The problem becomes more serious for multi-stage vaccinations because each dose must follow a fixed time interval. Staff need to check previous vaccination records before confirming the next appointment. If this process is done manually, patients may receive the wrong appointment date or miss the correct vaccination sequence. Current clinic systems provide only basic scheduling functions, so staff and patients still need to manage these requirements manually.

This paper presents an AI agent-based appointment scheduling system for clinics. The system uses AI agents to check appointment availability, balance doctor workloads and validate vaccine dose schedules using a relational database. It also provides a Telegram bot for booking, rescheduling and cancelling appointments and a Next.js

web portal for clinic staff. The implementation showed that the integration of AI-assisted decision making with rule-based validation resulted in more dependable scheduling. This was due to the fact that each appointment followed according to predetermined clinical guidelines, while also enabling staff to evaluate the final result.

RELATED WORK

Existing commercial systems

This study reviews three types of clinic appointment systems to identify the research gap. These include Electronic Health Record (EHR) scheduling systems such as ClinicAssist [1], web-based booking systems such as BookDoc [7] and rule-based chatbot platforms such as ManyChat [9] for Telegram and WhatsApp. The literature review indicates that these systems support basic appointment booking but have limited support for multi-stage scheduling, workload balancing and vaccine dependency validation. For example, in a simulated multi-dose vaccination case, most existing systems could book appointments but could not automatically verify the required interval before scheduling the next dose. Table 1 compares the strengths and limitations of these systems.

Table 1. Comparison of Existing Scheduling Systems

Category	Strengths	Weaknesses
Legacy EHR scheduling modules	Secure and closely linked with patient records and billing systems.	Used mainly for clinic management. No patient access. Cannot manage multi-stage appointments or automatically handle linked schedule changes.
Web-based booking portals	Patients can book and manage appointments online without calling the clinic.	The booking process is simple. Each appointment is treated as individual registration. Has no support for dependency checking or multi-stage appointment validation.
Rule-based chatbots	Patients can easily navigate through platforms that are well-known to them. No requirement for an additional application.	Employs predetermined guidelines and restricts the flow of conversation. Is unable to understand complex requests or interact with a live database for intelligent scheduling.

Although existing clinic appointment systems provide online booking and appointment management, they generally lack autonomous AI decision-making, workload-balanced scheduling, vaccine dependency validation and automated appointment rescheduling. Therefore, this research proposes a multi-agent clinic scheduling architecture that integrates these capabilities into a single intelligent platform.

AI Agents and LLM-Based Scheduling in the Literature

Recent studies have applied LLM-based agents to healthcare scheduling and clinical conversations. CareConnect, which is developed Hasan et al. [6], employs rule-based safety tests before the LLM processing user requests. The system in this study is quite similar to this approach, in which scheduling rules are verified before AI's decision-making. Zhi et al. [17] demonstrated that healthcare systems are evolving from basic chatbots to AI agents that have advanced memory and planning capabilities. This supports the use of AI agents that follow database rules instead of relying only on LLM responses. Ruiz Mejia and Rawat [14] proposed MedScrubCrew, which combines multiple LLM agents with matching algorithms to improve appointment scheduling. Hannebauer and Müller [5] used multi-agent systems to solve appointment scheduling problems and Marynissen and Demeulemeester [10] found that multi-agent methods and heuristic algorithms remain effective for healthcare scheduling. The primary objective of current research is to enhance scheduling algorithms. However, a relatively small number of clinic management platforms integrate AI agents with a complete system that includes a staff web portal, a Telegram bot and database-driven medical validation. This gap motivated the development of the proposed system.

Computational Technique Comparison

This study compares two methods for appointment scheduling. The first method uses rule-based constraint

checking with if-else logic. It is efficient and produces consistent outcomes; however, it is limited to understanding natural language and requires preset input formats. The second method uses an AI agent framework with LLMs managed by LangGraph [8]. It can understand natural language, apply scheduling rules and balance doctor's workloads more effectively. Experimental evaluation showed that the AI agent handled different patient requests more accurately than the rule-based method, especially when users entered incomplete or conversational messages. Therefore, the proposed system implements an AI agent framework that is monitored by LangGraph. The architecture implements "LLM race strategy" to optimize system reliability and processing speed. It synchronously queries the Google Gemini API and a local DeepSeek model, which was selected for its cost-effectiveness and reduced cloud dependency. The system rapidly provides patients with feedback, regardless of whether one service experiences latency, by executing scheduling decisions based on the model that returns a valid response first.

System Architecture

The platform adopts a decoupled three-tier client-server architecture, illustrated in Figure 1.

Three-Tier System Architecture for Clinic Management

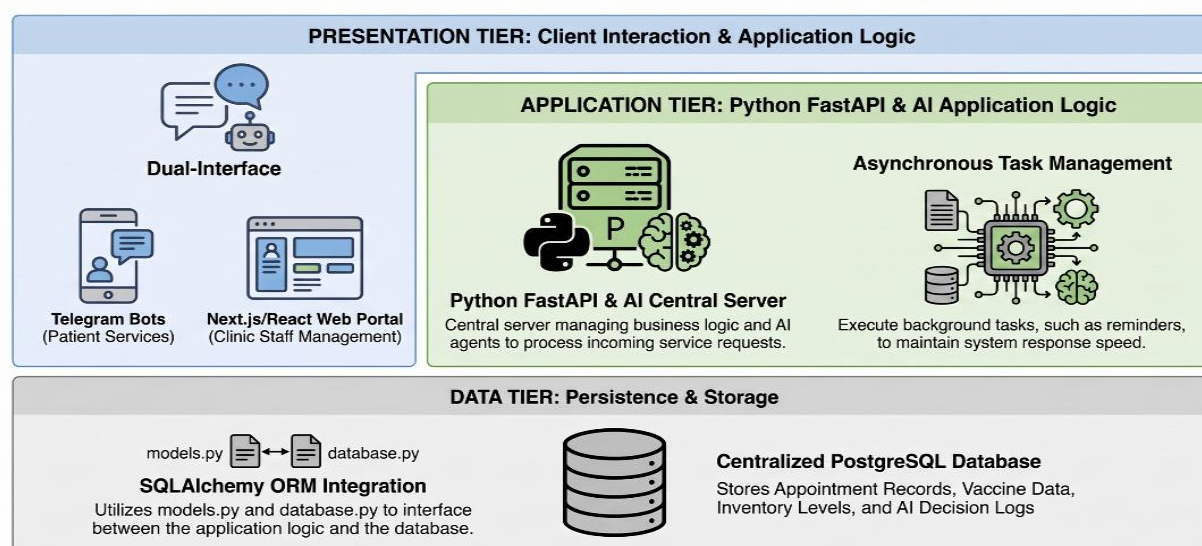


Figure 1. Three-Tier System Architecture: Presentation, Application and Data Layers

- **Presentation Tier:** The system offers a Next.js/React web portal for clinic staff and a Telegram bot for patients. The web portal enables staff to manage appointments and monitor system activities, while the Telegram bot supports appointment services.
- **Application Tier:** AI agent is deployed, business logic is managed and requests are processed by a Python FastAPI [4] server. Celery [2] with Redis [13] executes background tasks, including appointment reminders, without affecting the system's response time.
- **Data Tier:** The system relies on the SQLAlchemy [15] ORM (`models.py` and `database.py`) together with a PostgreSQL [11] database. In a central location, the database maintains appointment records, vaccine information, inventory data and AI decision logs.

Fig. 1 shows how an appointment request moves through the system. The patient sends a message through the Telegram Bot API [16]. The message is sent to the FastAPI server and processed by the LangGraph [8] AI agent. The agent checks appointment availability, doctor workload and vaccine scheduling rules using the PostgreSQL database through SQLAlchemy [15]. A booking confirmation, a rejection message and an alternative appointment slot is then sent to the Telegram bot. This workflow reduced the need for manual verification by validating scheduling rules before storing appointments in the database, as demonstrated by system testing. The Clinic Staff Web Portal utilises the same FastAPI services and database, enabling staff to observe and manage appointments in real time. Background tasks, including appointment reminders are managed by Celery [2] and Redis [13]. The AI agent also converts patient messages into structured JSON data and the system validates the data before creating or updating any appointment record.

AI Agent Design

The scheduling system uses several specialized AI agents and support modules instead of one large algorithm. Each component is responsible for a specific function, such as verifying database records and scheduling rules and interpreting patient requests. The modular implementation allowed each component to be independently tested, maintained and extended without affecting other modules.

Scheduling Agent

The Scheduling Agent validates appointment requests before AI processing by employing a single-node LangGraph StateGraph (scheduling_agent_node). It compares the requested appointment date and time with the availability of the selected doctor, rejects appointments that are scheduled during unavailable time slots or in the past and suggests the nearest available appointment slot when necessary. When multiple doctors are available, the system assigns the doctor with the lowest future workload. The system suggests the best available appointment together with 3 alternative time slots. Experimental evaluation demonstrated that this approach reduced invalid booking requests, improved scheduling accuracy and distributed appointments more evenly across doctors. The workload-balanced appointment displayed in the Telegram bot is illustrated in Fig. 2, which includes the selected doctor, the recommendation reason and alternative appointment slots.

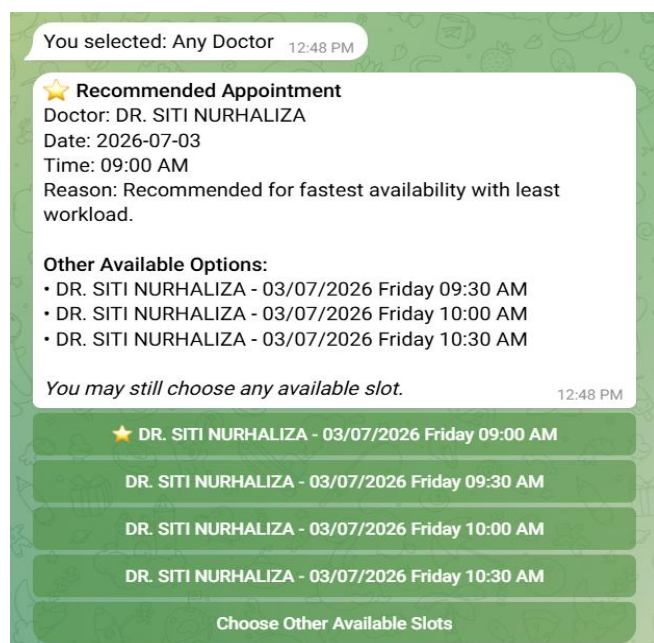


Figure 2. Telegram Bot Output of the Workload-Balanced Appointment Recommendation

Multi-Stage Vaccine Dependency Agent

The /validate-vaccine-booking routine checks a patient's completed doses, scheduled doses and external vaccination records before booking the next appointment. The next necessary dose is automatically determined by the system, which also verifies vaccination brand continuity, missing doses, repeat series eligibility, duplicate bookings, interruption rules and the necessary time between doses. The system presents the earliest available appointment date and rejects the booking if any rules are violated. Verification using simulated vaccination records demonstrated that this approach prevented incorrect dose sequences and reduced manual verification by clinic staff. All vaccine rules, such as dose intervals, repeat series and interruption settings, are stored in the database, so new vaccines can be added without changing the system logic.

Rescheduling Agent and Vaccine Search Agent

The Rescheduling Agent helps patients change their appointment date, time or doctor. Before updating the database, it uses the Scheduling Agent to verify the new appointment. Then, the previous appointment time slot is released and all related appointment records stay the same. Functional verification showed that this procedure avoided duplicate bookings and decreased schedule conflicts.

Natural language queries about diseases or vaccine names are accepted by the Vaccine Search Agent. It returns structured vaccine data, such as the type of vaccine, total doses, booster availability, repetition rules, interruption rules and dose intervals, using the LLM race strategy. In order to ensure accurate appointment scheduling, this data is sent straight away to the vaccine validation process.

Intelligent Patient-Admin Communication Agent

This agent is responsible for managing conversations that are beyond the scope of the other agents' automatic resolution responsibilities. The agent requests that the patient confirm a hand-off to clinic admin while preserving the in-progress booking draft when a patient's message during a booking flow is classified as clinic-related but unrelated to the current action. Upon confirmation, the message is forwarded to the clinic administrator and an `is_live_chat` flag directs any subsequent patient messages to the administrator dashboard. The patient receives real-time responses from the administrator via Telegram.

Database Design

As shown in Fig. 3, the system maintains data consistency by using a PostgreSQL database. The initial appointment data is stored in the appointments table, while the status and scheduled time of each appointment stage are stored in the appointment_stages table. To maintain the proper sequence for multi-stage appointments, the `depends_on_stage_id` field connects relevant stages. The experimental verification demonstrated that their relationships stored in the database improved validation accuracy and reduced scheduling errors. New vaccine schedules can be introduced without altering the system logic since vaccine rules, like dose intervals, are kept in the vaccines and vaccine_dose_schedules tables. When necessary, clinic staff may review appointment activities by referring to the chat_messages and agent_logs tables, which keep track of AI decisions and patient messages.

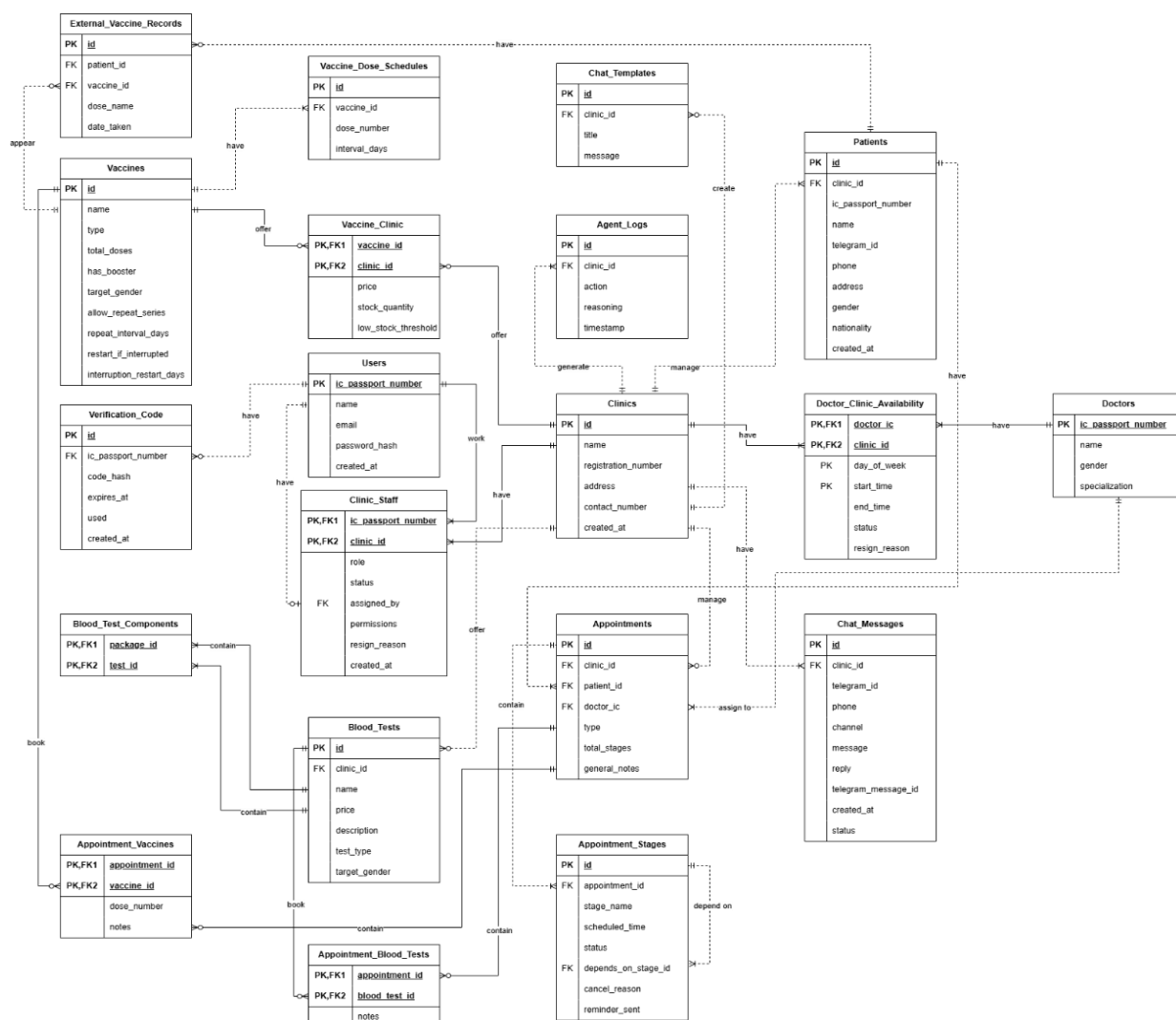


Figure 3. Entity-Relationship Diagram of the Clinic Scheduling Database

Implementation

The backend is developed using Python 3, FastAPI and Uvicorn. It offers over 70 RESTful APIs for clinic management, appointment booking, AI processing, vaccine validation, patient registration and clinic administration. The primary database is PostgreSQL, which is connected via secure SSL connections and the system employs SQLAlchemy [15] to manage database operations. The modular API design simplifies future system enhancements and improves maintainability.

The Telegram bot assists patients in the selection of a clinic, registration, service, appointment booking, rescheduling and cancellation. It is fully integrated with MyKad OCR, manual registration, inline date and time selection and an Exit command that enables users to terminate the booking process at any point. The evaluation conducted with simulated users confirmed that the menu-driven workflow reduced input errors and simplified the booking process for first-time users.

The Clinic Staff Web Portal, which was created using Next.js, React, TypeScript and Tailwind CSS, contains a clinic discovery page (Fig. 4) that offers a Telegram QR code and deep link for each clinic. Additionally, it provides an authenticated dashboard (Fig. 5) that facilitates the management of inventory, chat records, appointments, clinical schedules and account settings. The dashboard enables clinic staff to monitor AI-generated bookings and manually update appointments as necessary, thus enhancing system reliability and maintaining staff control.

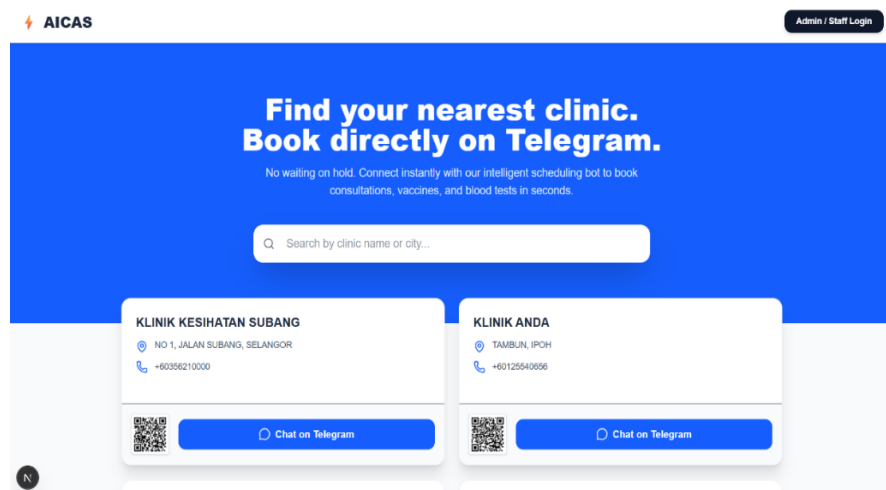


Figure 4. Public Clinic-Discovery Web Page Linking to the Telegram Bot

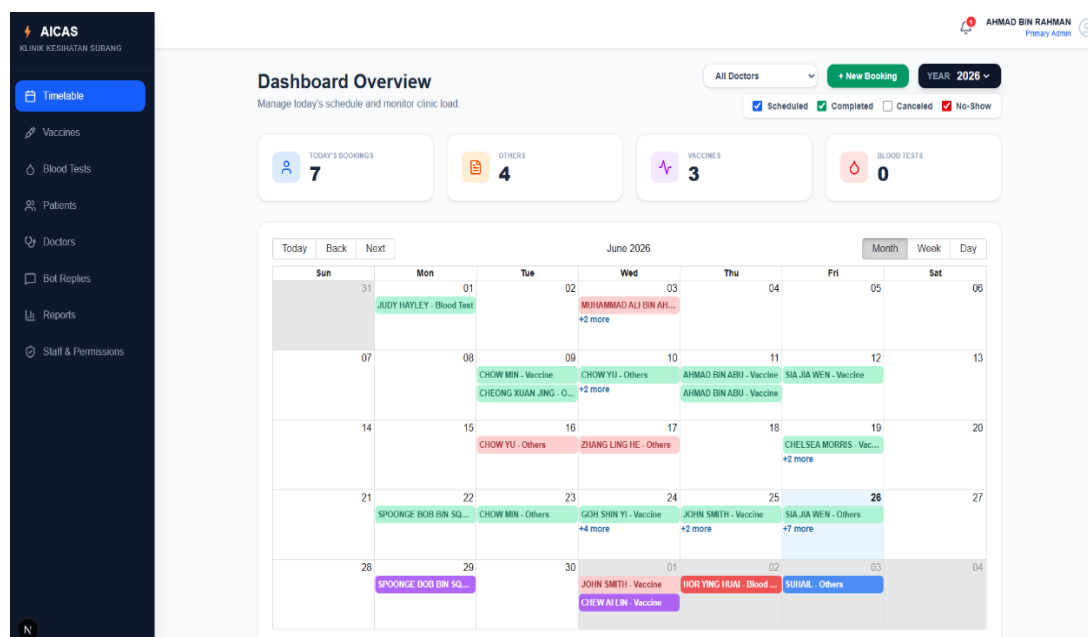


Figure 5. Clinic Staff Web Portal Dashboard/Timetable View

RESULTS AND DISCUSSION

Test Methodology

Two verification methods were used to evaluate the system. The first method tested database-independent functions, including the LangGraph Scheduling Agent and the natural language date and time parser in agent.py. These functions were tested with predefined inputs and the results are shown in Tables 2 and 3. The second approach evaluated functions that require constant data, such as workload balancing, vaccine validation and appointment slot recommendation, using automated pytest [12] test suites with fixture-based, simulated database records. The automated test results were verified against manually traced expected outputs. This method verified that before the full deployment, the system logic aligned to the expected scheduling requirements. Performance measures such as response time and OCR accuracy will be evaluated during the final User Acceptance Testing and deployment stages.

Scheduling Agent — Executed Test Results

The `scheduling_agent_node` function was tested using 5 representative `requested_time` inputs to evaluate the scheduling validation rules. The test cases covered valid requests, past dates, invalid date formats, unavailable doctor time slots and non-working day bookings. The expected results were achieved in all test cases according to the scheduling rules that were implemented. The verification results are summarized in Table 2, using the execution date of 2026-08-12 (Wednesday).

Table 2. Executed Test Cases - Scheduling Agent

Test Input	Rule Exercised	Actual Output
Date/time in the past	Past-time rejection	<code>is_valid=False</code> , "You cannot book an appointment in the past."
Weekday, 11:00, 7 days ahead	Available doctor time slot	<code>is_valid=True</code> , "Slot available."
Weekday, requested time unavailable	Doctor availability validation	<code>is_valid=False</code> ; suggested alternative = nearest available appointment slot
Sunday, 10:00	Non-working day rejection	<code>is_valid=False</code> , "No doctors are available on the selected day."; suggested alternative = next available working day
Unparsable string	Malformed-input handling	<code>is_valid=False</code> , "Invalid format."

All 5 test cases produced the expected results without any system errors. The Scheduling Agent successfully rejected invalid appointment requests and returned a clear rejection message, a valid alternative appointment time or both. Experimental evaluation demonstrated that the rule-based validation consistently enforced clinic scheduling rules before the AI processed the booking request, which improved the reliability of the appointment scheduling process.

Natural-Language Date/Time Extraction — Executed Test Results

The `calculate_exact_datetime` function was tested using five representative natural language date and time expressions. In order to ensure consistent and repeatable outcomes, the `current_time_str` parameter was fixed at 09:00:00 on Wednesday, 2026-08-12. These test cases handled frequent patient expressions that are employed during appointment scheduling. The function accurately converted each expression into the specified date and time format during the testing process. The verification results are presented in Table 3.

Table 3. Executed Test Cases - Date/Time Parser

Input (date, time)	Parsed Output
"tomorrow", "2pm"	2026-08-13; 14:00:00

“next monday”, “9.30am”	2026-08-17; 09:30:00
“in 3 days”, “17:00”	2026-08-15; 17:00:00
“2026-12-25”, “noon”	2026-12-25; time_period = “noon” (12:00 PM – 1:59 PM)
“25/12”, (none)	2026-12-25; time = None

Five out of five test cases produced the expected results. The `calculate_exact_datetime` function recognizes both number-based time expressions, such as "2 pm", "14:00" and "9:30 am", and word-based time-of-day expressions, such as "noon", "midnight", "morning" and "evening". For a word-based expression, the parser returns a matching `time_period` instead of a single clock time, and the scheduling flow then lists every available slot that falls within that period's clock-time window (e.g. 12:00 PM – 1:59 PM for "noon") for the requested date.

Multi-Stage Vaccine Dependency — Automated Test Verification

The `/validate-vaccine-booking` function was validated using an automated pytest [12] suite (`test_vaccine_dependency_evaluation.py`) with fixture-based patient and vaccine history records. All 10 representative vaccination scenarios that were executed, which included brand switching, dose intervals, repeat series and interruption rules completed successfully. Throughout the testing process, the system accurately rejected requests that were submitted before the required dose interval and returned the earliest valid appointment date. It also identified cases where the interruption period exceeded the allowed limit and instructed the patient to restart the vaccination series from Dose 1. Since all vaccine rules are stored in the `vaccines` and `vaccine_dose_schedules` tables, the same validation process can be applied to newly added vaccines without changing the system code.

Workload-Balanced Recommendation

In the live example shown in Fig. 2, the system selected the doctor with the lowest workload for the requested appointment and suggested three alternative time slots. The evaluation revealed that the recommendation process produced clear and consistent results because every recommendation included the reason for the selected doctor. The system also applied the same tie-breaking rule for identical requests, which ensured that scheduling decisions were consistent and repeatable.

Post-Implementation System Comparison

Table 4 maps the weaknesses of existing systems identified in Table 1 to the specific mechanism this implementation uses to address them, based on the executed and traced behaviour described above.

Table 4. How the Implemented System Addresses Previously Identified Weaknesses

Identified Weakness (Table 1)	Addressing Mechanism in This System
No patient access and manual data entry in legacy EHR systems	Menu-driven Telegram bot with EasyOCR [3] MyKad registration to simplify patient access and data entry.
No support for multi-stage appointment dependencies in legacy EHR systems	<code>depends_on_stage_id</code> relationship and <code>/validate-vaccine-booking</code> routine to manage multi-stage.
Form-based booking systems treat each appointment separately	Conversational booking process with automatic vaccine dose sequencing using <code>vaccine_dose_schedules</code> .
Rule-based chatbots use fixed rules and cannot access live database information	LangGraph AI agent with the LLM race strategy and real-time PostgreSQL access for appointment validation and conflict resolution.

Automated Accuracy Evaluation Suite

To complement the functional verification, a pytest [12] test suite was developed to measure system performance. The suite evaluates regression testing, smoke testing, scheduling accuracy, AI extraction accuracy,

vaccine dependency validation, workload balancing, communication agent accuracy, booking success rate and monkey testing.

- Regression Testing: Tests the core system functions after code modifications to verify that existing features continue to operate correctly without introducing functional regressions.
- Smoke Testing: Tests the core system components, including server startup, database connectivity, API services, AI agent initialization and appointment processing, to verify that the system is stable before detailed testing.
- Scheduling Accuracy: Tests the `scheduling_agent_node` with valid and invalid appointment requests to measure scheduling accuracy.
- AI Extraction Accuracy: The test `extract_appointment_details` utilises labelled patient messages to assess the accuracy of intent and appointment information extraction. The test `classify_general_message` evaluates the classification of message-intent into the categories of create/check/modify/delete/other/unrelated.
- Communication Agent Relay Accuracy: Tests the bi-directional message relay endpoints (`/ask-admin`, `/admin/chat-reply` and its edit/delete counterparts) to ensure that admin replies are accurately preserved and forwarded to patients via Telegram or SMS, including message edits and deletions.
- Vaccine Dependency Validation: Tests `validate_vaccine_booking` and `get_next_vaccine_dose` with representative vaccination scenarios, including brand changes, dose intervals, repeat series and interrupted schedules.
- Workload Balancing Accuracy: Tests `recommend_slots` to verify doctor selection based on workload, doctor preference and appointment availability.
- Booking Success Rate: Tests the `/book-appointment` endpoint against an actual disposable test database, in which a mixed batch of valid and intentionally invalid requests is used to confirm that valid bookings are successful and the invalid ones are accurately rejected.
- Monkey Testing: To ensure that the system degrades gracefully rather than collapsing, the system is fed large volumes of randomised and malformed input into the `admin chat-reply relay`, `classify_general_message`, `scheduling_agent_node` and `calculate_exact_datetime`.

Manual Functional Verification

On the React Clinic Staff Web Portal and the Telegram bot, 132 manual black-box functional test cases were executed across 19 functional modules. These test cases covered patient management, appointment management, doctor management, vaccine management, blood test management, clinic provisioning, staff permissions, account settings and the AI Agent Engine. In addition to automated testing, the test cases were executed manually. Both recorded failures were traced to a single defect described in the Limitation section, with 131 of 132 cases passing (99.2%).

Quantitative Performance Evaluation

Table 5. Performance Evaluation Summary

Evaluation Category	Test Cases	Passed	Failed	Result
Regression Testing	20	20	0	100.00%
Smoke Testing	7	7	0	100.00%
Monkey Testing	6	6	0	100.00%
Scheduling Accuracy	9	9	0	100.00%
Booking Success Rate	1	1	0	100.00%
Workload Balancing Accuracy	3	3	0	100.00%
AI Extraction Accuracy	15	15	0	100.00%
Communication Agent Accuracy	9	9	0	100.00%
Vaccine Validation Accuracy	10	10	0	100.00%
Overall Pytest Execution	80 tests	80	0	100.00%

The proposed system was evaluated using 80 automated test cases executed through the Pytest [12] framework. The evaluation included regression, smoke, monkey, scheduling, booking success rate, workload balancing, AI extraction, communication agent and vaccine validation tests. All 80 automated tests passed successfully without execution failures.

Functional regressions were not implemented by recent system modifications, as evidenced by the 100% pass rate of regression testing. Smoke testing also achieved a 100% pass rate, which indicates that all core services, API routes and application modules were correctly initialized.

Monkey testing also obtained a 100% pass rate. This evaluation verified that the system degraded gracefully, returning validation errors or rejection messages, rather than crashing or producing unhandled exceptions, by providing large volumes of randomised and malformed input into `calculate_exact_datetime`, `scheduling_agent_node`, `classify_general_message` and the admin chat-reply relay.

The Scheduling Agent effectively implemented doctor availability restrictions and appointment validation rules, achieving 100% scheduling accuracy across all evaluated scheduling scenarios.

Additionally, the Booking Success Rate assessment obtained a perfect score of 100%. The `/book-appointment` endpoint was tested against a real disposable test database with a mixed batch of valid and intentionally invalid requests. The results confirmed that valid bookings were consistently accepted and stored, while invalid requests were correctly rejected without corrupting existing appointment records.

The workload balancing evaluation achieved 100% accuracy across all evaluated scenarios by consistently assigning appointments to the available doctor with the lowest workload while satisfying clinic scheduling constraints. The evaluation confirmed that the workload balancing strategy followed the predefined scheduling rules, resulting in a fair distribution of appointments among the available doctors.

The AI Intent Extraction evaluation achieved 100% extraction accuracy and accurately classified general enquiries into booking-related and non-booking-related categories for hand-off routing by accurately identifying user intent, appointment date and appointment time from natural language requests.

The Vaccine Dependency Agent achieved a 100% validation accuracy by accurately enforcing vaccine interval requirements, brand consistency, repeat vaccination rules and interrupted vaccination policies.

The Patient-Admin Communication Agent achieved 100% relay accuracy by accurately saving admin replies to the database and forwarding them to patients via Telegram or SMS, including message edits without losing hand-off messages.

Limitations

- **Infrastructure Dependency:** The system requires a stable internet connection for the Gemini API and appropriate hardware to operate the local DeepSeek model. The system is unable to proceed with AI extraction and keeps the patient's message if both services are unavailable.
- **Region-Specific Registration:** The MyKad OCR and gender detection functions are specifically designed for the 12-digit identity card format for Malaysia. To facilitate the processing of identity documents from other countries, these functions require additional development.
- **Response Time Variation:** The LLM race strategy utilizes the first response from either the Gemini API or DeepSeek. Experimental evaluation demonstrated that this approach improved system availability. However, response time can still increase if both AI models respond slowly at the same time.
- **Message-Deletion Synchronisation:** The message-deletion handler now verifies the Telegram `deleteMessage` API response before deleting the corresponding local database record, so an admin-initiated deletion request for a message outside Telegram's 48-hour deletion window is rejected with a clear error instead of silently desynchronising the local chat record. A related gap remains unresolved: the Telegram Bot API does not notify bots when a patient deletes their own message directly within

Telegram, so such deletions succeed on Telegram but are not reflected in the clinic's local chat records. This does not affect the correctness of vaccine-dependency or scheduling.

CONCLUSION AND FUTURE WORK

This paper presents an AI agent-based multi-stage appointment scheduling system for clinics. The system combines a Telegram bot, a Next.js web portal, a PostgreSQL database and AI agents to support appointment scheduling. During system testing, it was found that the Scheduling Agent consistently enforced clinic rules, including doctor availability, past-date validation and non-working day restrictions. The date and time parser also handled most common patient requests correctly, including word-based time-of-day expressions such as "noon" and "midnight", which are resolved to a bookable time window rather than a single exact time. The vaccine validation process and workload balancing logic also followed the expected scheduling rules during code verification. The proposed system addresses the main limitations identified in existing clinic scheduling systems by providing AI-assisted booking, workload balancing and multi-stage vaccine validation while allowing clinic staff to review and manage all appointments through the administrative dashboard.

Future work will include the following: extending the message-deletion synchronisation mechanism to reconcile messages that patients delete directly within Telegram, since the Telegram Bot API does not notify bots of user-initiated deletions; generalising the OCR and identity-verification components to support flexible document schemas beyond the Malaysian MyKad format for broader geographic compatibility; the development of a live-LLM evaluation harness to directly measure the semantic accuracy of the DeepSeek and Gemini outputs (rather than the mocked deterministic wrapper); the extension of fuzz testing to the /book-appointment and /validate-vaccine-booking endpoints; and the addition of multi-language support (Bahasa Malaysia and Mandarin) to the conversational flow of the Telegram bot.

ACKNOWLEDGMENT

The authors would like to thank Fakulti Kecerdasan Buatan dan Keselamatan Siber (FAIX) and Centre for Research and Innovation Management (CRIM), Universiti Teknikal Malaysia Melaka (UTeM) for providing the facilities and support for this research, The author would like to thank her supervisor, Dr. Suhail S. M. Alqrinawi, for his guidance and support throughout this project.

REFERENCES

1. Assurance Technology Pte Ltd. (2026). Clinic Assist: Clinic management software. <https://clinicassist.co>
2. Celery Project. (2025). Celery: Distributed task queue documentation. <https://docs.celeryq.dev>
3. EasyOCR (JaidedAI). (2025). EasyOCR: Ready-to-use OCR documentation. <https://github.com/JaidedAI/EasyOCR>
4. FastAPI Team. (2025). FastAPI documentation. <https://fastapi.tiangolo.com>
5. Hannebauer, M., & Müller, S. (2001). Distributed constraint optimization for medical appointment scheduling. In Proceedings of the 5th International Conference on Autonomous Agents (AGENTS) (pp. 139–140). <https://doi.org/10.1145/375735.376026>
6. Hasan, H., Salman, S., Abou Dargham, A. T., Mohanna, A., & Chehab, A. (2026). Toward trustworthy large language model agents in healthcare. arXiv. <https://doi.org/10.48550/arXiv.2607.05055>
7. Health4U Solutions Sdn. Bhd. (2026). BookDoc: Healthcare appointment booking platform. <https://www.bookdoc.com/>
8. LangChain Inc. (2025). LangGraph documentation. <https://langchain-ai.github.io/langgraph>
9. ManyChat. (2025). ManyChat: Chat marketing platform for Messenger, Instagram and WhatsApp. <https://manychat.com>
10. Marynissen, J., & Demeulemeester, E. (2019). Literature review on multi-appointment scheduling problems in hospitals. *European Journal of Operational Research*, 272(2), 407–419. <https://doi.org/10.1016/j.ejor.2018.03.001>
11. PostgreSQL Global Development Group. (2025). PostgreSQL documentation. <https://www.postgresql.org/docs>

12. pytest-dev Team. (2025). pytest documentation. <https://docs.pytest.org>
13. Redis Ltd. (2025). Redis documentation. <https://redis.io/docs>
14. Ruiz Mejia, J. M., & Rawat, D. B. (2025). MedScrubCrew: A medical multi-agent framework for automating appointment scheduling based on patient-provider profile resource matching. *Healthcare*, 13(14), 1649. <https://doi.org/10.3390/healthcare13141649>
15. SQLAlchemy Authors. (2025). SQLAlchemy documentation. <https://docs.sqlalchemy.org>
16. Telegram FZ-LLC. (2025). Telegram Bot API documentation. <https://core.telegram.org/bots/api>
17. Zhi, X., Zhao, H., Wu, L., Zhao, C., & Zhu, H. (2025). Reinventing clinical dialogue: Agentic paradigms for LLM-enabled healthcare communication. *arXiv*. <https://doi.org/10.48550/arXiv.2512.01453>