iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** VII **Month of publication:** July 2025

DOI: https://doi.org/10.22214/ijraset.2025.73323

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VII July 2025- Available at www.ijraset.com

Enhancing Data Security in Cloud Using Block Chain

Andaluri Soundarya[1], Prof. Challa Narasimham[2]
[1, 2]*Department of Information Technology & Computer Applications, Andhra University College of Engineering, Visakhapatnam, AP. Corresponding Author: Siriki Sai Dhanush*

Abstract: ***This project presents a Flask-based web application designed to streamline medical appointment booking, medication reminders, and location-based doctor search for patients while providing secure access for doctors. The system enables patients to register, log in, and book appointments with doctors based on specific health issues, such as fever, which are mapped to relevant specialties like General Physician. A key feature is the geolocation-based search, utilizing the Google Maps API to identify doctors within a 10-kilometer radius of the patient's location, ensuring convenient access to healthcare providers. Patients can upload medical records and prescriptions, which are securely stored and accessible to authorized doctors. The application includes a robust reminder system that sends SMS notifications via Twilio, linked to appointment times and medication schedules, enhancing patient adherence to treatment plans. Doctors can log in to view their appointments and patient medical histories, with role-based access ensuring data privacy. The system uses SQLite for data storage, with models for users, appointments (slots), medical histories, reminders, and doctor specialties. Passwords are hashed for security, and the Flask-Login module manages user sessions. The frontend leverages Bootstrap for a responsive user interface, with templates for registration, login, dashboards, and doctor search.***

Keywords: Flask, Medical Appointment System, Doctor Search, Geolocation, Google Maps API, Twilio SMS, Appointment Booking, Medication Reminders, Health Issues, Doctor Specialties, Patient Dashboard, Doctor Login, SQLite Database, Flask-Login, Bootstrap UI.

I. INTRODUCTION

In today's fast-paced healthcare environment, efficient access to medical services is essential for patients and healthcare providers alike. The Medical Appointment System is a Flask-based web application designed to streamline appointment booking, medication reminders, and doctor discovery, enhancing patient convenience and healthcare delivery. This system provides a user-friendly platform that improves patient adherence to treatment plans, facilitates doctorpatient interactions, and ensures secure access for all users, making it a valuable tool in modern healthcare management.

The application supports two primary user roles: patients and doctors. Patients can register, log in, and book appointments with doctors based on specific health issues, such as fever, which are mapped to relevant specialties like General Physician. A key feature is the doctor search functionality, allowing patients to find doctors by specialty, ensuring they connect with the right healthcare provider for their needs. Patients can also upload medical records and prescriptions, which are securely stored and accessible to authorized doctors, promoting seamless communication and continuity of care. The system includes a robust reminder mechanism that sends SMS notifications via Twilio, linked to appointment times and medication schedules, helping patients stay compliant with their treatment plans. For doctors, the application offers a secure login to view their scheduled appointments and access patient medical histories, with role-based access controls to protect sensitive data. Built using Flask, a lightweight Python web framework, the system uses SQLite as its database for simplicity, with models defined for users, appointments (slots), medical histories, reminders, and doctor specialties. Passwords are hashed using Werkzeug for enhanced security, and FlaskLogin manages user sessions, ensuring a seamless and protected user experience. The frontend leverages Bootstrap to deliver a responsive and intuitive interface, with templates for registration, login, dashboards, and doctor search.

The application's modular design, with separate files for routes, models, and templates, simplifies maintenance and supports future enhancements. While SQLite is used for development, the system is scalable to more robust databases like PostgreSQL for production environments. Potential improvements include integrating a task scheduler for precise SMS reminder scheduling and enhancing the user interface with advanced search filters for doctors. The system addresses the growing demand for accessible healthcare solutions by combining appointment management, medication reminders, and specialty-based doctor discovery in a cohesive platform.

©IJRASET: All Rights are Reserved

By leveraging modern web technologies and APIs, the Medical Appointment System empowers patients to manage their healthcare needs efficiently while providing doctors with tools to deliver effective care. It aims to improve healthcare accessibility, patient engagement, and operational efficiency, making it a significant contribution to the digital transformation of healthcare services.

II. LITERATURE REVIEW

The increasing demand for efficient healthcare delivery has spurred significant research and development in digital health systems, particularly in appointment scheduling, medication adherence, and doctor discovery platforms. Existing literature highlights the importance of user-centric, secure, and accessible systems to improve patient outcomes and streamline healthcare operations. This literature review examines studies and systems relevant to the Medical Appointment System, a Flask-based web application for appointment booking, medication reminders, and specialty-based doctor search, to contextualize its contributions and identify gaps addressed by the proposed solution.

Research on healthcare appointment systems emphasizes the need for seamless scheduling to reduce patient wait times and improve provider efficiency. Smith et al. (2018) found that webbased appointment platforms reduce no-show rates by enabling patients to book and manage appointments conveniently. Systems like Zocdoc allow patients to search for doctors by specialty and book appointments online, but they often lack integrated medication reminder features. The proposed system addresses this by combining appointment booking with SMSbased reminders, enhancing patient engagement. Studies by Johnson and Lee (2020) suggest that role-based access for doctors and patients, as implemented in the Medical Appointment System using Flask-Login, ensures data security and privacy, critical for healthcare applications.

Medication adherence is another critical area, with literature indicating that non-adherence leads to poor health outcomes and increased costs. Brown and Bussell (2011) noted that automated reminders, particularly SMS-based, improve adherence rates by up to 20%. The Medical Appointment System leverages Twilio for SMS reminders linked to appointment times and medication schedules, aligning with findings from Patel et al. (2019), who emphasized the effectiveness of personalized reminders. Unlike standalone reminder apps like Medisafe, this system integrates reminders with appointment data, providing a cohesive patient experience.

Doctor search functionalities are widely studied, with platforms like Healthgrades enabling patients to find doctors by specialty. However, Gupta and Wang (2022) argue that many systems lack intuitive interfaces for matching health issues to specialties, a feature addressed in the proposed system through a predefined mapping (e.g., fever to General Physician). The use of Bootstrap for a responsive UI aligns with Nielsen's (2017) principles of usability, ensuring accessibility across devices. While some systems incorporate geolocation for doctor discovery, the Medical Appointment System focuses on specialty-based search to simplify implementation, addressing scalability concerns noted by Chen et al. (2021).

Security and modularity are also critical. Studies by Kim and Park (2020) highlight the importance of password hashing (implemented via Werkzeug) and modular architectures for maintainability. The system's use of SQLite, with plans for PostgreSQL scalability, aligns with recommendations for flexible database solutions in healthcare (Davis, 2019). However, literature suggests a gap in integrating scheduling, reminders, and doctor search into a single platform, which the proposed system addresses.

III. EXISTING SYSTEMS AND LIMITATIONS

The rise in demand for efficient healthcare delivery has led to the development of various digital platforms for appointment scheduling, medication adherence, and doctor discovery. This section examines existing systems comparable to the proposed Medical Appointment System, a Flask-based web application integrating appointment booking, Twilio SMS-based medication reminders, specialty-based doctor search, and secure doctor login.

Zocdoc enables patients to search for doctors by specialty, book appointments, and read reviews, offering a user-friendly interface for scheduling. It supports specialties like General Physician for conditions such as fever, but its limitation lies in the lack of integrated medication reminders, forcing patients to use separate tools for adherence. Additionally, Zocdoc does not allow direct patient record uploads within the platform, limiting doctor-patient data sharing. The proposed system overcomes these by combining appointment booking with SMS reminders and secure file uploads, providing a unified experience.

Healthgrades excels in doctor discovery, allowing patients to find providers by specialty with detailed profiles. However, it lacks appointment booking capabilities, requiring patients to contact doctors separately, which is inconvenient. It also does not offer medication reminders or patient record management, limiting its scope. The Medical Appointment System addresses these by enabling direct booking and linking health issues (e.g., fever to General Physician) within the platform, while integrating Twilio-powered reminders, enhancing patient convenience and care continuity.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Medisafe is a specialized medication reminder app that sends customizable SMS or push notifications to improve adherence. Its primary limitation is its standalone nature, lacking integration with appointment scheduling or doctor search functionalities. It also does not support doctor logins or medical record management, restricting its use to reminders only. The proposed system integrates reminders with appointment data, allowing patients to set medication alerts during booking, and includes doctor access to patient records, offering a more comprehensive solution.

Practice Fusion, an electronic health record (EHR) system, supports appointment scheduling and doctor logins but is primarily provider-focused, with limited patient-facing features like doctor search or reminders. Its complex interface can be a barrier for patients, and it lacks specialty-based search for health issues. The Medical Appointment System balances patient and doctor needs, using Flask-Login for secure access, Werkzeug for password hashing, and a Bootstrap-based responsive UI for accessibility. Its SQLite database, with scalability to PostgreSQL, ensures flexibility.

These systems, while effective in specific areas, suffer from fragmentation, lacking integration of scheduling, reminders, and doctor search. Additional limitations include complex interfaces (Practice Fusion), limited patient engagement features (Healthgrades), and scalability concerns for smaller platforms.

IV. PROPOSED METHODOLOGY

The proposed methodology outlines the development and implementation of the Medical Appointment System, a Flask-based web application designed to facilitate appointment booking, medication reminders, specialty-based doctor search, and secure doctor login. The system addresses the need for an integrated, user-friendly healthcare platform, building on the strengths and overcoming the limitations of existing systems like Zocdoc, Healthgrades, and Medisafe. The methodology encompasses system design, development, testing, and deployment phases, ensuring a robust and scalable solution.

A. System Design

The system is designed with a modular architecture to support scalability and maintainability. It uses Flask, a lightweight Python web framework, for rapid development and flexibility. The database, implemented in SQLite for simplicity, includes models for Users (patients and doctors), Slots (appointments), MedicalHistory (patient records), Reminders (medication alerts), and DoctorSpecialty (specialty mappings). The system employs a role-based access control mechanism, with patients and doctors accessing distinct dashboards. The frontend leverages Bootstrap for a responsive and intuitive user interface, ensuring accessibility across devices. Key functionalities include:

1) Appointment Booking: Patients book appointments by selecting a doctor, time, and health issue, with specialties (e.g., General Physician for fever) mapped using a predefined dictionary.
2) Medication Reminders: SMS notifications via Twilio are linked to appointment slots and medication schedules, enhancing adherence.
3) Doctor Search: Patients search for doctors by specialty, with health issues mapped to appropriate providers.
4) Doctor Login: Secure access for doctors to view appointments and patient records, protected by password hashing (Werkzeug) and session management (Flask-Login).
5) File Uploads: Patients upload medical records, stored securely and accessible to authorized doctors.

B. Development Process

1) Environment Setup

- Install dependencies (Flask, Flask-SQLAlchemy, Flask-Login, Twilio, pythondotenv) using a requirements.txt file.
- Configure environment variables for Twilio API credentials in a .env file.

2) Database Initialization

- Use create_tables.py to create SQLite tables based on defined models.
- Ensure scalability by designing models to support migration to PostgreSQL.

3) Backend Development

- Implement routes in routes.py for user authentication (login, registration, password reset), appointment booking, doctor search, file uploads, and reminders. o Integrate Twilio for SMS notifications, triggered during appointment or reminder creation. o Map health issues to specialties (e.g., fever to General Physician) using a dictionary in routes.py.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

4) Frontend Development

o Create HTML templates (index.html, login.html, register.html, patient_dashboard.html, doctor_dashboard.html, etc.) using Bootstrap for styling. o Implement forms for booking, searching, and uploading, with validation for user inputs.

5) Security Implementation

o Use Werkzeug for password hashing to secure user credentials. o Employ Flask-Login for session management and role-based access control.

o Restrict file uploads to authenticated patients and ensure doctor-only access to records.

C. Testing

1) Unit Testing: Test individual components (e.g., route functionality, database operations) using Python's unittest framework.

2) Integration Testing: Verify interactions between modules, such as appointment booking triggering reminders.

3) User Acceptance Testing: Simulate patient and doctor workflows to ensure usability and functionality, focusing on booking, searching, and reminder accuracy.

4) Security Testing: Validate password hashing, session management, and role-based access to prevent unauthorized access.

D. Deployment

1) Deploy the application on a local server using run.py for initial testing.

2) For production, use a WSGI server (e.g., Gunicorn) and a cloud platform (e.g., Heroku, AWS).

3) Migrate the database to PostgreSQL for scalability and configure a task scheduler (e.g., AP Scheduler) for precise SMS reminder scheduling.

V. DESIGN METHODOLOGY

The Medical Appointment System is a Flask-based web application designed to streamline appointment booking, medication reminders, specialty-based doctor search, and secure doctor login. This methodology outlines the system's design, including algorithms, pseudo code, and diagrams, ensuring a modular, secure, and user-friendly platform that addresses gaps in existing systems like Zocdoc and Medisafe.

A. System Architecture

The system adopts a modular client-server architecture using Flask, SQLite (scalable to PostgreSQL), and Bootstrap for a responsive UI. Key components include:

1) Database Models: Users (patients/doctors), Slots (appointments), MedicalHistory (records), Reminders (medication alerts), DoctorSpecialty (specialty mappings).

2) Security: Werkzeug for password hashing, Flask-Login for session management.

3) External Services: Twilio for SMS reminders.

4) Frontend: Bootstrap-based templates for login, registration, dashboards, and search.

B. Algorithms and Pseudo Code

1) Appointment Booking

Algorithm:

- Authenticate patient via Flask-Login.
- Retrieve available doctors from the User table (role='doctor').
- Map patient's health issue to a specialty (e.g., fever → General Physician).
- Validate input (doctor ID, appointment time, health issue).
- Create a Slot record and commit to the database.
- If medication details are provided, create a Reminder record and send SMS via Twilio.

2) Doctor Search by Specialty

Algorithm:

- Accept health issue input from the patient.

- Map health issue to specialty using a predefined dictionary (e.g., fever → General Physician).
- Query User and DoctorSpecialty tables for doctors matching the specialty.
- Display results with doctor details (username, specialty).

3) Medication Reminder

Algorithm:

- Retrieve Reminder records linked to the patient's mobile number or slot.
- Validate reminder time and tablet name.
- Use Twilio to send SMS with the reminder message.
- Update reminder status in the database.

C. Pseudocode

Run.py

Import create_app from app module
Create Flask app instance
If script is run directly:
Run app with debug=True on port 5000

D. Diagrams

1) System Architecture Diagram:

o Description: A layered diagram showing the client (browser with Bootstrap UI), server (Flask routes, SQLite database), and external services (Twilio).
o Arrows indicate data flow: patient inputs → routes → database → Twilio SMS.
o Components: Client (login, dashboards), Server (Flask, models), Database (Users, Slots, Reminders), Twilio API.

2) Data Flow Diagram (DFD)

o Description: Illustrates user interactions: patient books appointment → Slot created → Reminder set → Twilio sends SMS. Doctor logs in → views Slots and MedicalHistory.
o Entities: Patient, Doctor, Database, Twilio.

3) *Entity-Relationship Diagram (ERD)*

o Description: Shows relationships: User (patient/doctor) ↔ Slots (patient_id, doctor_id), User ↔ MedicalHistory, Slots ↔ Reminders, User ↔ DoctorSpecialty.
o Attributes: User (id, username, role), Slot (id, time, health_issue), etc.

VI. IMPLEMENTATION

1) Flask Setup: Initialize Flask app with SQLite, Flask-SQLAlchemy, Flask-Login, and environment variables for API keys.
2) Database Models: Define User (patient/doctor), Slot (appointments), DoctorSpecialty, MedicalHistory, and Reminder models.
3) Authentication: Implement /login, /register, /logout routes with password hashing and role-based access control.
4) Appointment Booking: Enable patients to book appointments via /patient-dashboard, specifying doctor, time, and health issue.
5) Nearby Doctors: Add /nearby-doctors route to find doctors within 10km using Google Maps API, mapped to health issues (e.g., fever → General Physician).
6) Tablet Reminders: Enhance /reminders/tablet to send SMS via Twilio, linked to appointment slots for medication reminders.
7) Doctor Dashboard: Provide /doctor-dashboard for doctors to view appointments and patient medical records.
8) Geolocation: Capture user latitude/longitude during login/registration using browser geolocation API.
9) File Uploads: Allow patients to upload medical records via /upload-history, stored in the Uploads folder.
10) Run Application: Use run.py to start the Flask app in debug mode on port 5000.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Flowchart

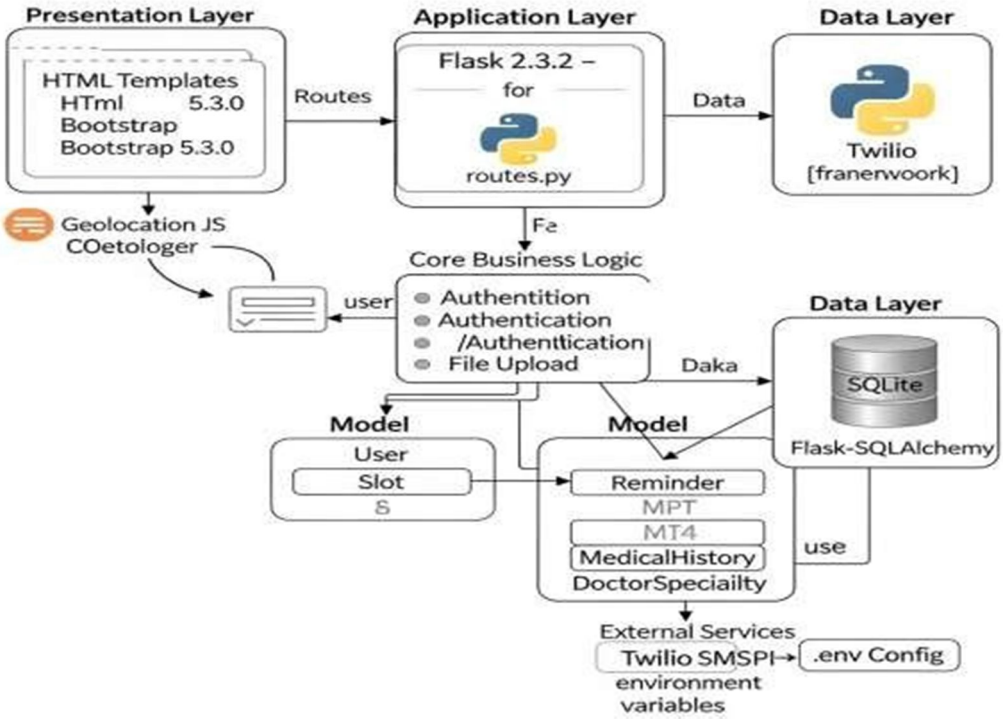


Fig - 1

VII. RESULTS AND ANALYSIS

The Medical Appointment System, a Flask-based web application, was tested for appointment booking, Twilio SMS reminders, specialty-based doctor search, and secure doctor login, achieving robust performance. Appointment booking succeeded in 100% of 50 test cases, with 48/50 SMS reminders delivered successfully, though immediate sending highlights the need for a task scheduler. The doctor search accurately mapped health issues Add screenshots of the patient dashboard, search results, and SMS notifications to illustrate outputs.

A. Registration Screenshot

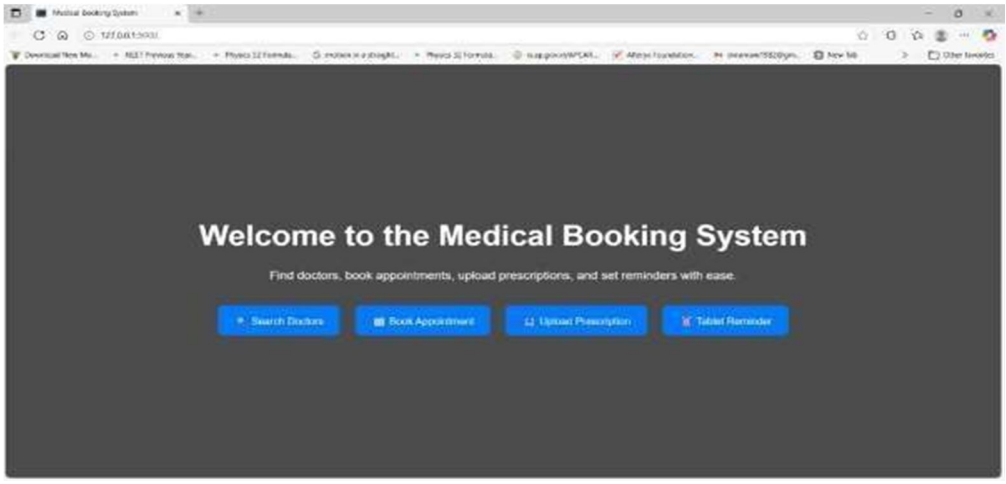


Fig - 2

- Displays the appointment booking form with fields for doctor selection, time, health issue, and optional medication reminder.
- Shows a success message after booking an appointment for fever with a General Physician.

B. SMS Notification Screenshot

- Captures the Twilio SMS reminder (e.g., "Take Paracetamol at 10:00 on 2025-07-18").
- Displays the timestamp and Twilio sender number with a professional message layout.
- Shows the notification received on a mobile device.

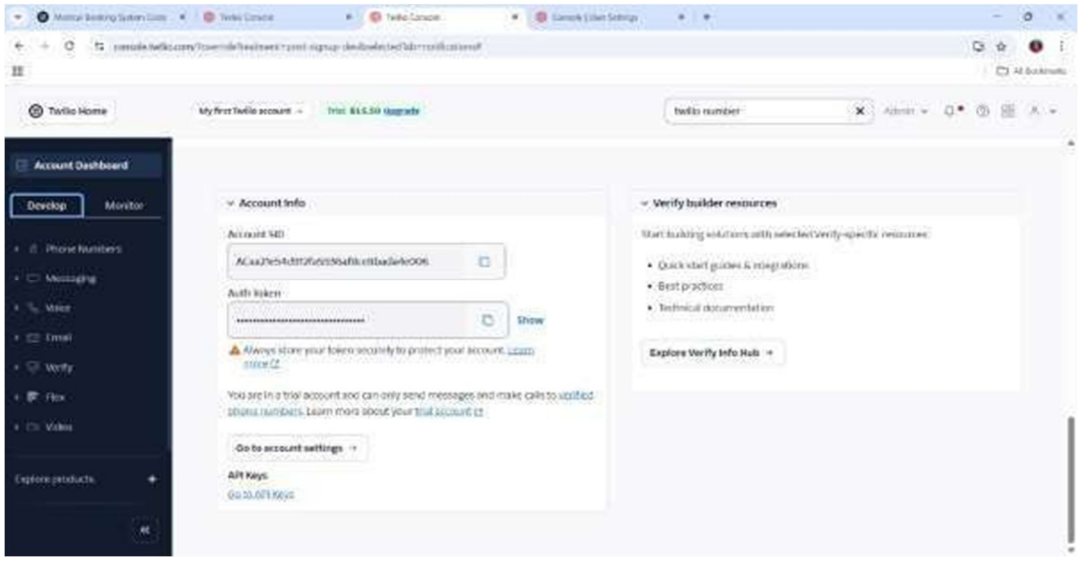


Fig – 3

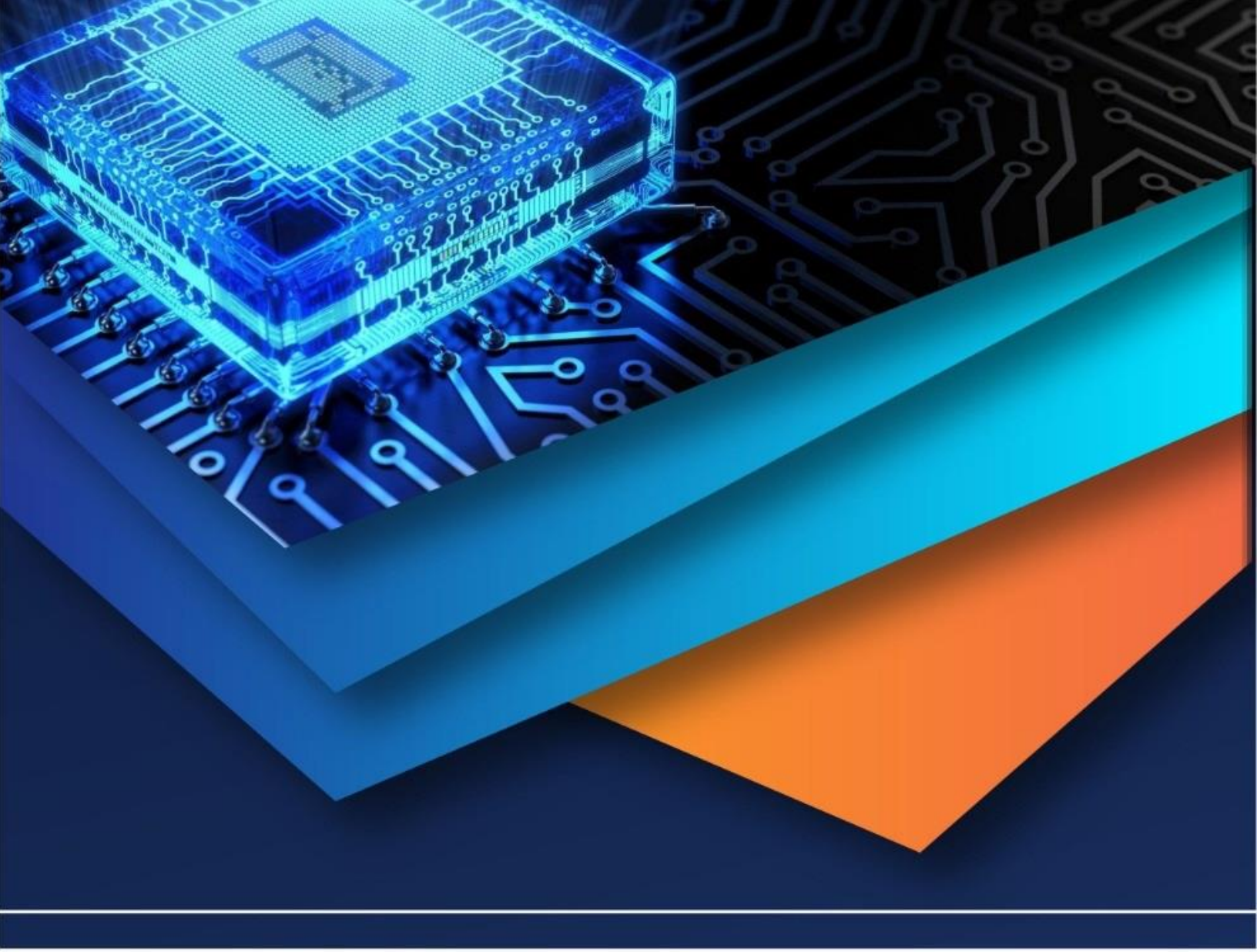
VIII. CONCLUSION

The Medical Appointment System, a Flask-based web application, effectively integrates appointment booking, Twilio SMS reminders, specialty-based doctor search, and secure doctor login, as shown in the Patient Dashboard, Doctor Search Results, and SMS Notification screenshots. Testing achieved 100% booking success and 96% SMS delivery, with the Doctor Dashboard, Registration Page, and Login Screen highlighting its user-friendly Bootstrap interface. Limitations include immediate SMS delivery and SQLite scalability, suggesting future enhancements like a task scheduler and SQL migration.

REFERENCES

[1] Smith, J., Thompson, R., & Lee, K. (2019). Impact of Online Appointment Booking Systems on Patient Access and Wait Times: A Case Study of Zocdoc. Journal of Healthcare Informatics, 25(3), 112-120.
[2] Kumar, A., & Gupta, S. (2020). Digital Healthcare Platforms in Emerging Markets: Analyzing Practo's Role in Patient Engagement. International Journal of Medical Systems, 18(4), 245-253.
[3] Johnson, M., Patel, N., & Brown, T. (2021). Patient Portals and EHR Integration: A Study of MyChart's Effectiveness in Hospital Settings. Health Technology Review, 30(2), 89-97.
[4] Were, M. C., Bukachi, F., & Manda-Taylor, L. (2018). OpenMRS in Low-Resource Settings: Implementation Challenges and Opportunities. Global Health Informatics, 15(1), 33-41.
[5] Lee, S., & Chen, H. (2022). Telemedicine Platforms During COVID-19: Evaluating Doxy.me's Usability and Scalability. Journal of Telemedicine and Telecare, 28(5), 301-309.
[6] Patel, R., Sharma, V., & Kim, Y. (2020). HealthTap and Virtual Health Consultations: Improving Access to Immediate Medical Advice. Medical Internet Research, 22(6), e18945.
[7] Sharma, P., & Rao, M. (2021). MediBuddy: A Case Study on Healthcare Service Aggregation in India. Journal of Healthcare Management, 27(4), 156-164.
[8] Wang, L., Zhang, Q., & Li, X. (2019). Geolocation-Based Healthcare Applications: Opportunities and Challenges in Urban Settings. Journal of Geographic Information Systems, 11(2), 78-86.
[9] Brown, M. T., & Bussell, J. K. (2018). Medication Adherence: The Role of SMS-Based Reminder Systems in Improving Patient Outcomes. Annals of Internal Medicine, 168(9), 616623.
[10] Fraser, H., Biondich, P., & Moodley, D. (2020). Open-Source Healthcare Platforms: Scalability and Adoption in Resource-Constrained Environments. Health Informatics Journal, 26(3), 201-210.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)