IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

MedHist: A Medical Unified Database Management System

Ayush Chandgude[1], Vinit Mule[2], Amol Chincholkar[3]
Department of School of Computing[1,2,3]
MIT ADT University Pune, India
ayushchandgude08@gmail.com; vinitmule2772@gmail.com;
amolchincholkar@mituniversity.edu.in

Abstract: *In today's healthcare system, the requirement for effective, secure, and convenient digital record-keeping of medical information has become more significant. Manual processes tend to be time-consuming, error-prone, and inconvenient for patients as well as healthcare providers. To counteract these issues, this project presents MedHist – a Medical Unified Database Management System, which is a light web application that has been developed using Python (Flask), MySQL, and typical web technologies including HTML and CSS. MedHist was created as an integrated system that is intended to manage patient information, medical histories, treatments, medications, and appointment scheduling.*

Keywords: Medical Records, Record System, Flask Framework, MySQL, Webpage, Patient Profile

I. INTRODUCTION

Over the past few years, the healthcare industry has witnessed a massive shift towards automation and digitization, prompted by growing needs for quality patient care, accuracy in record handling, and streamlining of administrative functions. Hospitals, clinics, and even small private medical practices are increasingly embracing electronic health record (EHR) systems to improve patient data handling. However, most available solutions in the market are costly, complex, or require a lot of technical expertise for proper installation and operation. MedHist (Medical Unified Database Management System) aims to overcome these limitations by offering a deployable, user-friendly, and lightweight solution that is best suited for small-scale healthcare providers or educational institutions. MedHist makes simple operations like patient registration, login, appointment scheduling, and management of medical records simple through an easy-to-use web interface. Contrary to conventional systems involving the use of several loosely integrated tools or paper-based processes, MedHist consolidates all patient-related information in one integrated platform, thus making it easy to access for updates and consultations by healthcare professionals like physicians, nurses, or administrators, as and when needed. This leads to fewer errors, time savings, and improved overall quality of care. MedHist is coded with Flask, a Python micro web framework deserving of lightness and versatility, and MySQL, a lightweight database engine for applications in embedded or local environments. MedHist, for its part, employs the Waitress WSGI server for production-level and hassle-free deployment in Windows environments

The purpose of MedHist is to create a simple-to-use medical database system where users are able to:

- Enrol patients with required medical data
- Log in with their Patient ID and Name
- Observe and review patient record
- Book appointments
- Secure access to all information by a web browser

MedHist is useful to display a patient's profile where the entire details are to be displayed. The medicines which are to be required are also displayed as the entire portfolio is generated. The extent of the MedHist – Medical Unified Database Management System is to design a light, scalable, and user-friendly application for efficiently handling patient data in small to mid-level healthcare facilities like clinics, medical camps, or educational institutions.

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-26467

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

This project aims to:

- Provide a core repository for the storage of patient data such as personal information, medical conditions, treatments, and drug history.
- Enable secure registration and login with a one-of-a-kind Patient ID system.
- Allow users to schedule and manage appointments via the same web interface.
- Offer transparent database connectivity and real-time updates with a slim backend.
- Preserve simple deployment and platform independence, particularly on Windows platforms with a WSGI production server such as Waitress.
- Employ modern, responsive web technologies to create a clean, accessible user interface.
- Make the system modular and extensible to enable future integration of future advanced functionalities such as AI-based diagnosis, doctor/admin panels, or remote health monitoring.

II. LITERATURE SURVEY

EHR systems such as Epic and Cerner control hospital infrastructure but have steep barriers to entry for small providers. Open-source solutions such as Open MRS are promising but need technical skill. Flask-based applications provide rapid development and integration, which is best suited for focused solutions such as MedHist. Research also highlights the need for patient-centred systems and secure data. MedHist combines the best practices from other systems and puts accessibility and ease of use first. The development and implementation of Electronic Health Record (EHR) systems have revolutionized healthcare today, streamlining patient care coordination, decreasing medical errors, and enhancing the overall outcomes of healthcare. However, there are differences in the implementation methods, scalability, accessibility, and security between systems renders it cumbersome—especially to solo practitioners and small clinics.

Commercial EHR Systems

Well-established clinical EHR systems like Epic, Cerner, and Allscripts dominate the large hospital environments. These systems have strong features, such as advanced analytics, billing integration, and compliance features. However, their implementation is expensive and usually demands considerable IT infrastructure and training. HealthIT.gov (2022) states that their complexity and high price render them unfit for low-resource settings or small-scale practices. A recurring issue in EHR adoption is poor user experience, leading to clinician burnout. Research shows that lightweight systems designed with simplicity, responsiveness, and minimal clicks can significantly improve adoption and satisfaction rates. This supports the minimalist interface design strategy used in MedHist, which prioritizes clarity and accessibility over excessive features. Although EHR technology is mature, few platforms are simultaneously free, lightweight, user-friendly, and customizable for low-scale deployment. Most clinics only need a limited subset of the features offered by enterprise EHRs but are either constrained to over-invest or resort to using paper-based solutions. This is where MedHist seeks to fit in—by providing a working system with login, tracking of patient records, and scheduling of appointments without requiring specialized hardware or staff.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

III. SYSTEM ARCHITECTURE AND DESIGN

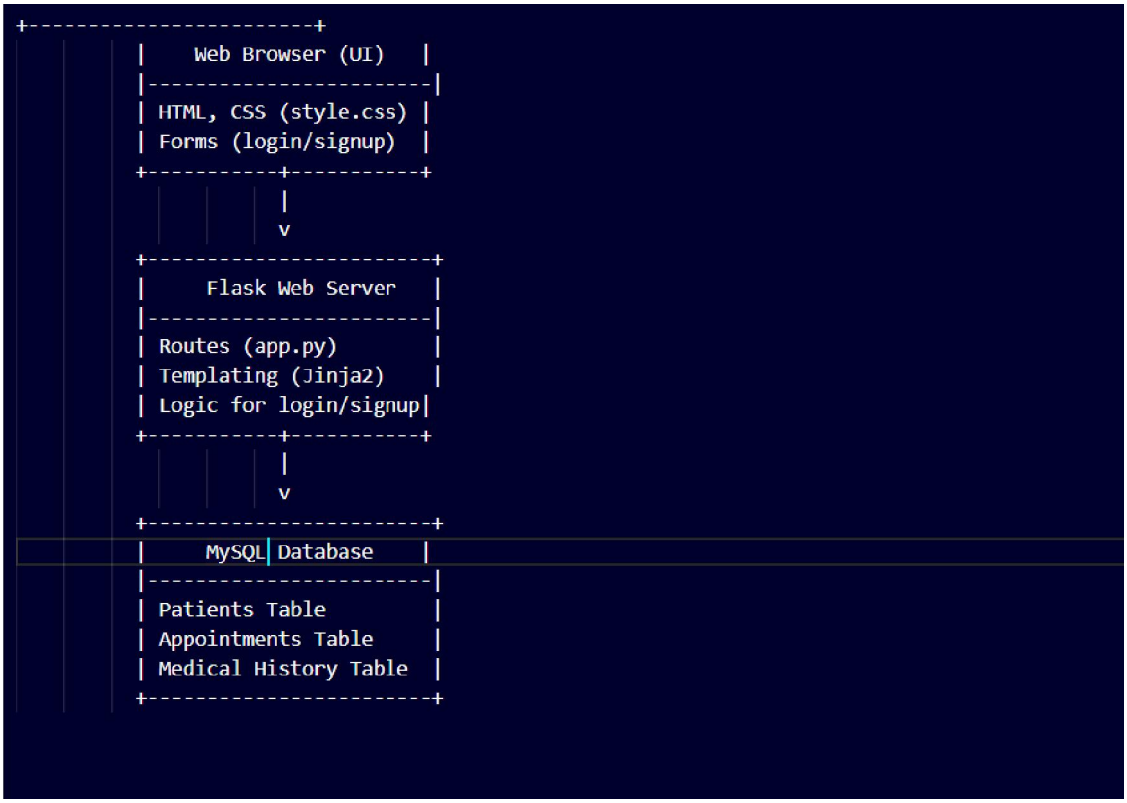

Figure 1. System Architecture Diagram

1. Frontend (Web Browser)

- Built with HTML, CSS templates
- Includes pages like login.html, signup.html, home.html, and appointments.html.

2. Backend (Flask Framework)

- Uses app.py to handle routing, logic, and interaction with the database.
- Communicates with HTML templates via Jinja2.
- Manages patient registration, login, form submissions, and data fetching.

3. Database (MySQL)

- A file-based lightweight database (database. dB).

Stores:

- Patient Information
- Appointment Bookings
- Medical History

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

BLOCK DIAGRAM OF MEDHIST

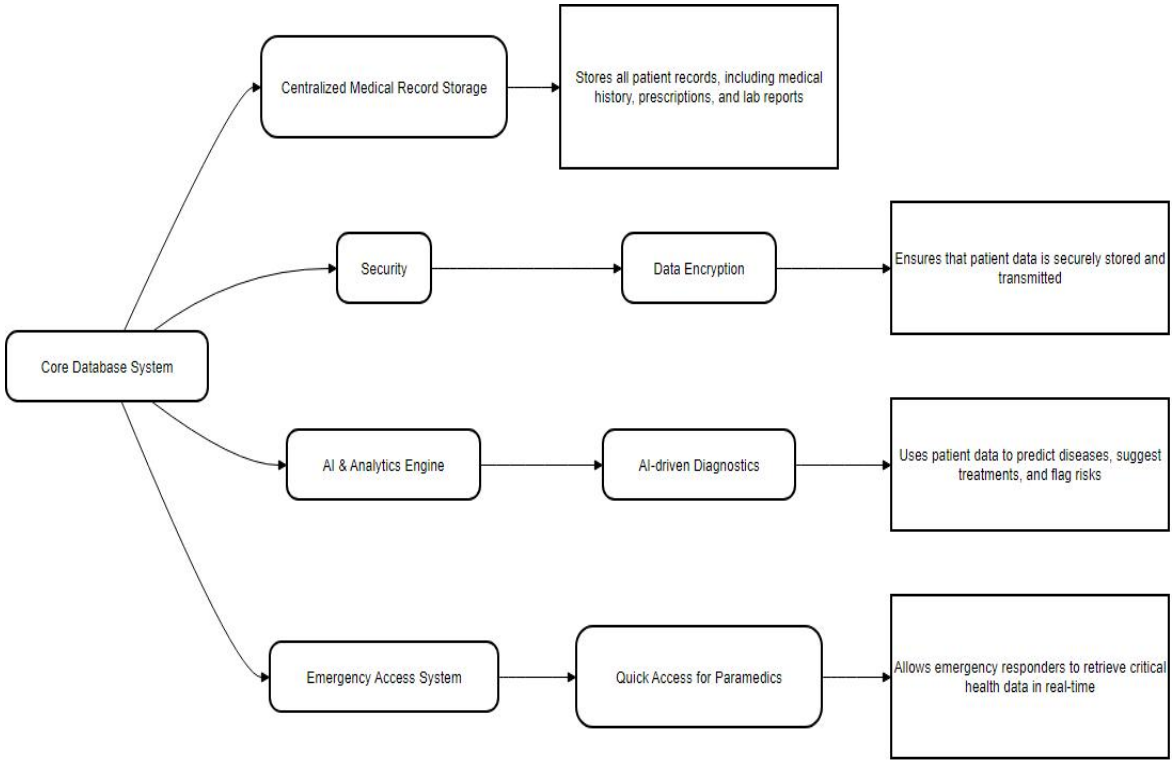

Figure 2. Block Diagram of MedHist

IV. PROPOSED METHODOLOGY

MedHist was developed using Python's Flask framework with MySQL for database management. The development process followed agile principles, incorporating user feedback through iterative testing. Core features include:

- User registration/login with secure password hashing
- Auto-generated patient IDs
- Medical history input and view functionality
- Appointment booking
- Clean, responsive UI with HTML/CSS and Bootstrap
- Deployment using Waitress for Windows-based production

The Core Features of MedHist are as follows

1. User Registration (Signup)

- Input: Name, Age, Gender, Address, Medical Condition, Treatment, Medicines.
- Generates a unique Patient ID.

2. User Login

- Requires only Name and Patient ID for secure access.

3. Medical History

- Patients' data is stored and can be retrieved or extended with new visits and treatments.

4. Appointment Management

- Patients can book appointments through a simple form.
- Stored in the database with date, time, and patient reference.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

V. RESULTS AND DISCUSSION

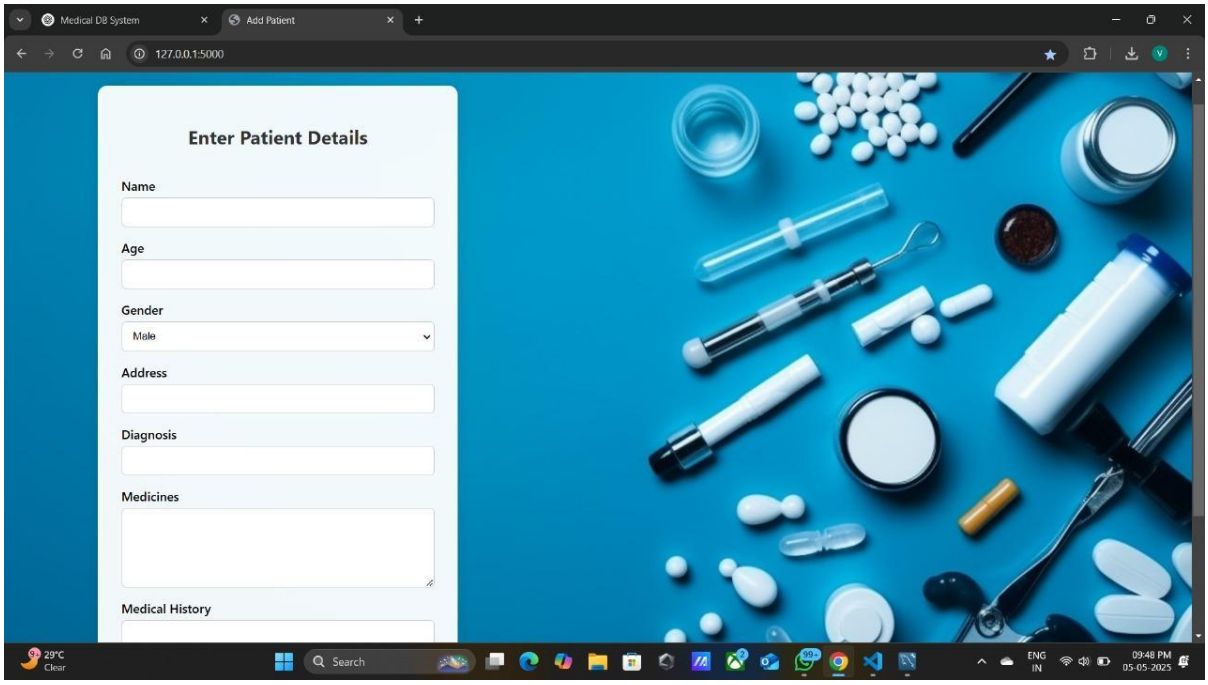

Figure 3. Patient Details on a Webpage

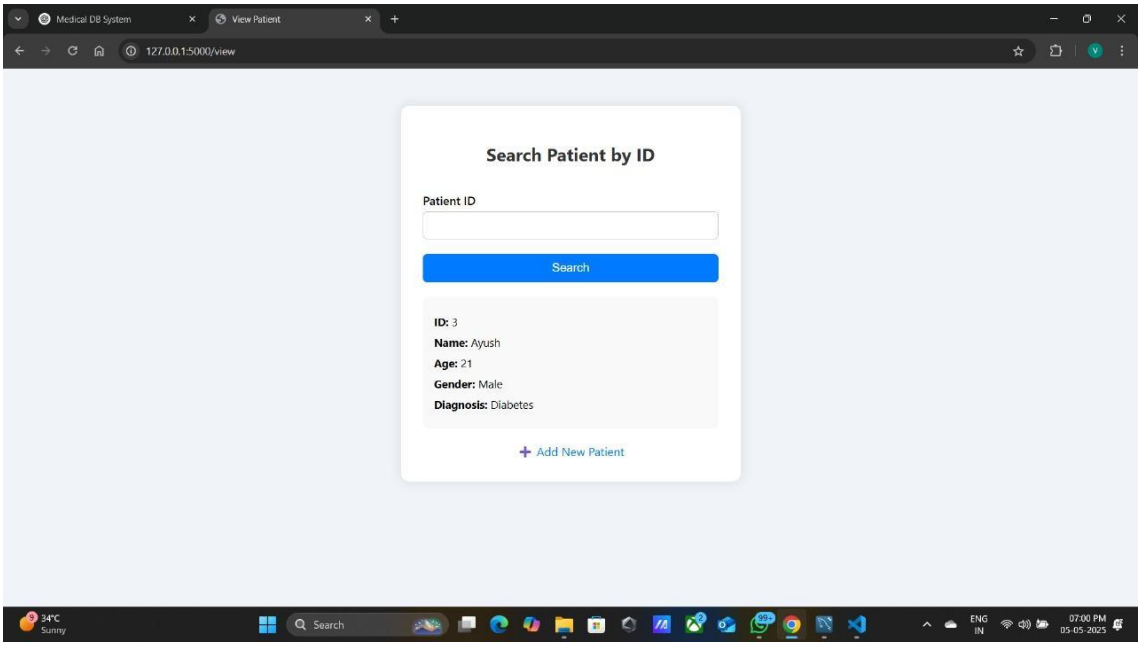

Figure 4. Patient Profile thru Patient ID

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

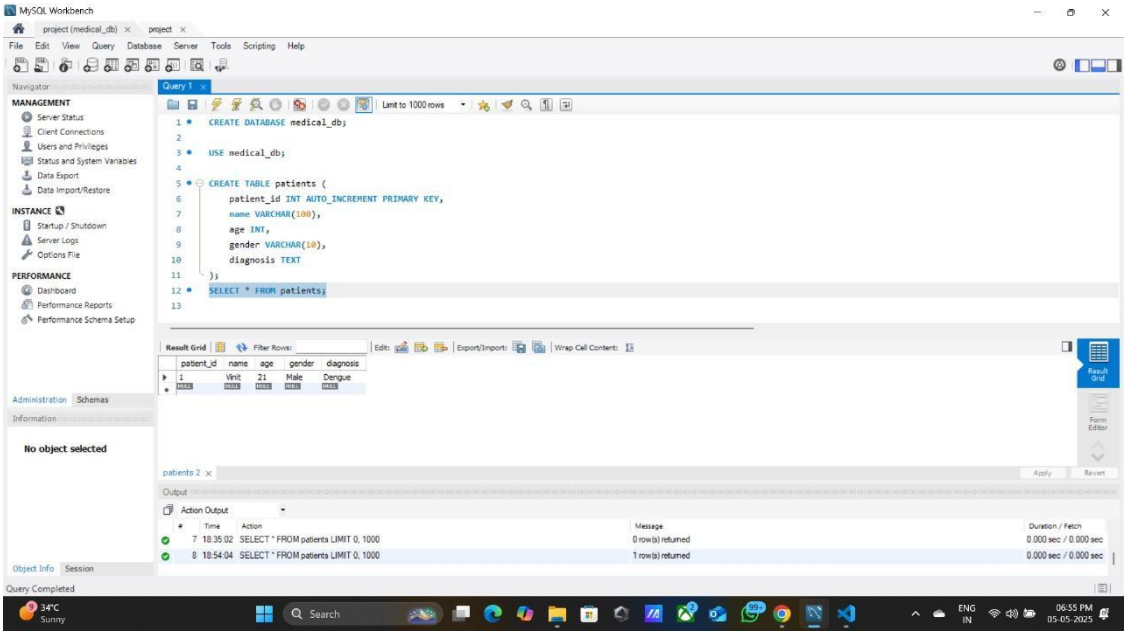

Figure 5. Data is Stored in MySQL Database

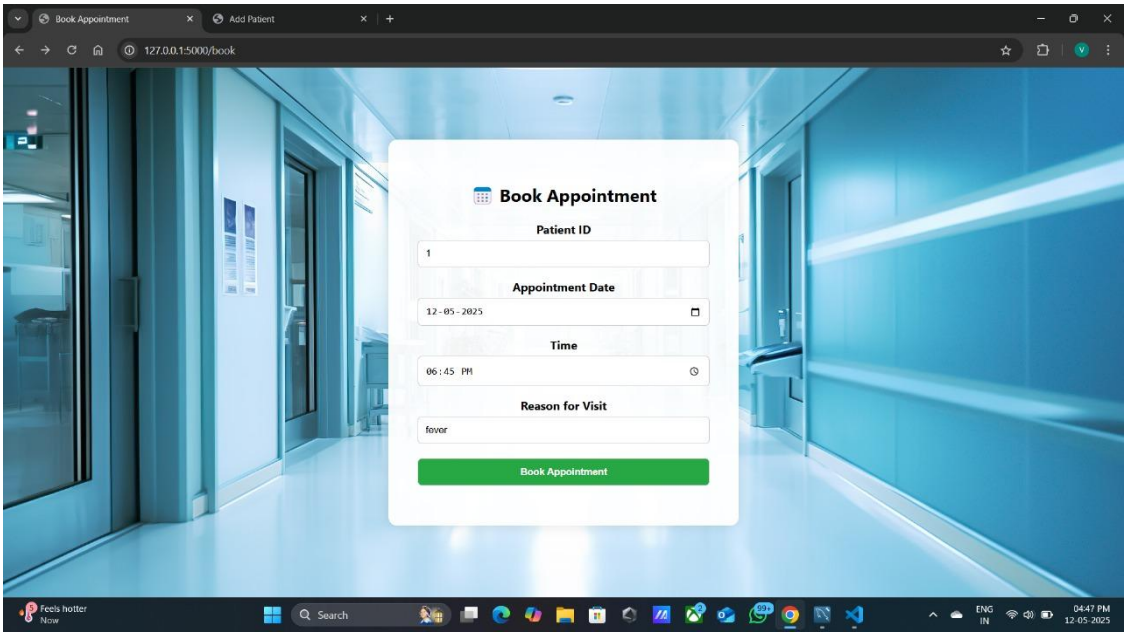

Figure 6. Appointment Scheduling

Copyright to IJARSCT
www.ijarsct.co.in
DOI: 10.48175/IJARSCT-26467

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

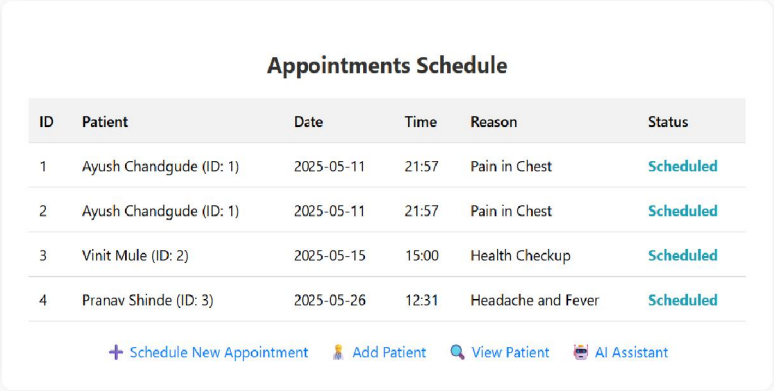

Figure 7. Appointments Scheduled for Patients

The MedHist system was tested locally in a simulated clinic environment. It demonstrated:

- High loading speed and interactive user activity
- Proper retrieval and updating of patient records
- Efficient appointment scheduling and confirmation
- Secure login/logout sessions

MedHist is designed with a combination of simplicity and required functionality. It is optimized for single practitioners or small clinics with a requirement for low-cost, private setup. Scalability is, however, restrained by MySQL database, and MedHist does not have sophisticated analytics or billing functionality and also the application of AI which is work in progress. These can be added in future versions.

VI. TESTBED SETUP AND CONFIGURATION

To verify the functionality, performance, and usability of the MedHist system, a local testbed environment was established simulating a small clinic environment. This section describes the hardware, software, and setup utilized during development and testing.

6.1 Hardware Installation

The MedHist testbed was implemented on a typical personal computer of a typical small clinic workstation:

- Processor: Intel Core i5 (2.40 GHz)
- RAM: 8 GB
- Storage: 256 GB SSD
- Operating System: Windows 11 Pro 64-bit
- Network: Localhost (127.0.0.1) via a private LAN

This setup mimics a low-resource environment to enable MedHist to function optimally without advanced infrastructure.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

6.2 Software Stack

The software technology stack used across the implementation and deployment of MedHist was:

- Programming Language: Python 3.11
- Web Framework: Flask 2.3
- Database: MySQL
- Deployment Server: Waitress 2.1 (for running Flask app on Windows)
- Front-end: HTML5, CSS3, Bootstrap 5
- Development Tools: Visual Studio Code, Git for version control
- Test Browser: Google Chrome v124, Mozilla Firefox v115

6.3 User Simulation and Roles

To simulate the real-life clinic setting, a variety of user categories used the system:

- Admin (test environment): Used to review databases manually.
- Medical Staff: Utilized the interface to view/enter patient history and appointments.
- Patient Simulation: Utilized limited access to verify UI response time and clarity.

6.4 Security and Session Management

- All user passwords were hashed through salted hashing through werkzeug.security.
- Flask session management was implemented using secure cookies.
- Basic form validation and input sanitization were employed to avoid SQL injection and XSS.

6.5 Limitations

The testbed now employs a single-machine setup without concurrency simulation. Future tests will include multi-user access simulation and deployment on cloud-based test environments to test performance under load.

VII. CONCLUSIONS AND FUTURE WORK

MedHist offers a promising solution for decentralized, light-weight patient information management. It has the necessary EHR functionality combined with simplicity of deployment and usage. Future includes the integration of advanced security mechanisms, cloud deployment, and data visualization.

FUTURE WORK

While MedHist readily offers an entire system for managing patient medical history, there are several areas are amenable to improvement:

Cloud Deployment and Access Control:

Having MedHist in cloud computer systems such as AWS, Azure, or Google Cloud would provide remote access, data backup and system scalability are key characteristics. Additionally, role-based access control (e.g., administrator, physician, and receptionist) would introduce multi-user capabilities and improve the security controls on the data.

Enhanced Security Frameworks:

Future versions will have to accommodate features such as 2FA, HTTPS encryption, and audit logs so that it will be HIPAA or GDPR compliant for medical data privacy laws.

Integration with Medical Devices and APIs:

Integrating device APIs such as for blood pressure monitoring devices or fitness trackers would enable real-time updating of patient histories. Also, integration with national healthcare systems on HL7 or FHIR standards would enable exchange of more complete data.

Patient Portal and Mobile Interface:

Adding a patient portal or mobile app would enhance user engagement, allowing patients to see their medical records, schedule appointments, and get reminders or drug reminders.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 5, Issue 4, May 2025
Impact Factor: 7.67

Data Visualization and Reporting Tools:

Adding charts, graphs, and summary dashboards via libraries such as Chart.js or D3.js can offer insights to physicians and administrators to support enhanced clinical and operational decision-making.

AI-Informed Insights:

Future versions will use machine learning algorithms to identify patterns in patient data, identify anomalies, or suggest diagnoses based on past patterns, thus significantly improving clinical support.

REFERENCES

[1]. Caine, K., & Hanania, R. (2013) Patients desire fine-grained control of health data privacy in electronic health records. Journal of the American Medical Informatics Association, 20(1), 7-15.

[2]. Kuo, A. M. (2011). Opportunities and challenges of cloud computing to improve health care services. Journal of Medical Internet Research, 13(3), e67.

[3]. Mamlin, B. W., et al. (2006). Open MRS: an open-source platform for developing medical record systems. AMIA Annual Symposium Proceedings.

[4]. HealthIT.gov. (2022). Benefits of EHRs. https://www.healthit.gov/topic/health-it-basics/benefits-ehrs

[5]. Zhao, L., et al. (2021). Building lightweight healthcare applications using Flask: A case study. International Journal of Medical Informatics, 149, 104417.

[6]. R Tamblyn, L Poissant, A Huang et al., "Estimating the information gap between emergency department records on community medication versus online access to community pharmacy records", Journal of the American Medical Informatics Association, vol. 21, no. 3, pp. 391-398, 2014.

[7]. L. Zhang, G. Ahn, and B. Chu. A role-based delegation framework for healthcare information systems. In ACM Symposium on Access Control Models And Technologies (SACMAT), page, 2002