ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Meditrack: A Gui-Based Clinic Appointment and Patient Record Management System with MYSQL Storage and Excel Backup

Ambrad, Jhasfer Moi., Bandoy, John Fred A., Bluza, Joel Jr. D., Broncano, Kyla Mae., Dongon, Ma. Janelle T., San Juan, Chzian Eric D., Engr. Fabro, Meshelle N.

Department of Computer Engineering Eulogio "Amang" Rodriguez Institute of Science and Technology (EARIST) Nagtahan, Manila, Philippines

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150600056

Received: 05 June 2026; Accepted: 10 June 2026; Published: 04 July 2026

ABSTRACT

Manual management of clinic appointments and patient records can cause delays, misplaced files, duplicate entries, inaccurate information, and difficulty in retrieving patient data. To address these problems, this study developed **MediTrack**, a GUI-based Clinic Appointment and Patient Record Management System designed to help clinic staff manage patient information and appointment records in a faster, more organized, and more reliable manner. The system was developed using Python as the main programming language, Tkinter for the graphical user interface, MySQL for permanent database storage, and OpenPyXL for automatic Excel backup. MediTrack includes important features such as secure login, adding patient records, viewing records, searching patient information, updating appointment status, deleting records, clearing input fields, and safely exiting the system. It also applies input validation to prevent common errors, such as blank fields, duplicate patient IDs, invalid names, incorrect age input, and contact numbers that do not contain exactly eleven digits. Patient data are stored in a MySQL database while a duplicate copy is automatically saved in an Excel file for backup and reporting purposes. The results of system testing showed that MediTrack successfully performed its intended functions and prevented invalid information from being saved. Overall, MediTrack provides a practical, user-friendly, and efficient solution for improving clinic appointment scheduling and patient record management.

Keywords: MediTrack, Clinic Management System, Patient Record System, Appointment System, Python, Tkinter, MySQL, Excel Backup

INTRODUCTION

Background of the Study

Medical clinics handle different types of patient information every day, including patient names, ages, contact numbers, symptoms, assigned doctors, appointment dates, and appointment status. In many clinics, these records are still managed manually using paper forms, logbooks, or separate files. Although manual recording may be useful for small-scale record keeping, it can become difficult to manage as the number of patients increases.

Manual clinic record management often leads to problems such as lost documents, unreadable handwriting, duplicate entries, slow record retrieval, and inaccurate patient information. Clinic staff may also find it difficult to update appointment status, monitor patient schedules, and organize records efficiently. These problems can affect the quality of service provided to patients because staff may spend more time searching for records instead of assisting patients.

With the advancement of technology, computerized systems have become helpful tools in improving healthcare-related record management. A computerized clinic system can store patient records in a database, retrieve information quickly, reduce paperwork, and improve the accuracy of data handling. Through a graphical user interface, clinic staff

can interact with the system using buttons, forms, tables, and input fields, making the system easier to operate

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

compared to manual record keeping or command-based programs.

This study focuses on the development of **MediTrack: A GUI-Based Clinic Appointment and Patient Record Management System with MySQL Storage and Excel Backup**. The system was designed to help clinic staff manage patient records and appointments through a desktop-based application. It allows users to add new patient records, view existing records, search patient information, update appointment status, delete records, clear input fields, and exit the system safely.

MediTrack uses MySQL as its database management system to store patient records permanently. This allows records to remain saved even after the program is closed. The system also uses OpenPyXL to automatically create an Excel backup of patient records, providing an additional copy for reporting and data safety. Furthermore, input validation is included to prevent errors such as blank fields, duplicate patient IDs, invalid names, incorrect age entries, and invalid contact numbers.

Overall, MediTrack was developed to provide a more organized, accurate, and efficient method of managing clinic appointments and patient records. It helps reduce manual work, improve data reliability, and support faster clinic transactions through the use of Python, Tkinter, MySQL, and Excel backup integration.

Statement of the Problem

This study aims to develop MediTrack, a GUI-based Clinic Appointment and Patient Record Management System that can efficiently manage and organize patient records and appointment information.

Specifically, this study seeks to answer the following questions:

1. How can MediTrack automate the process of patient appointment and record management?
2. How can the system efficiently save and retrieve patient records using a MySQL database?
3. How can the system reduce human errors commonly found in manual clinic logbooks?
4. How can a graphical user interface improve user interaction and usability for clinic staff?
5. How can Excel backup support record safety and reporting in the system?

Objectives of the Study General Objective

The general objective of this study is to develop **MediTrack**, a GUI-based Clinic Appointment and Patient Record Management System using Python, Tkinter, MySQL, and Excel backup to help clinic staff manage patient records and appointments efficiently.

Specific Objectives

Specifically, this study aims to:

1. Design the system flow using a flowchart and structured process.
2. Develop the system backend using Python programming language.
3. Create a user-friendly graphical user interface using Tkinter.
4. Integrate MySQL database for permanent and organized storage of patient records.
5. Implement automatic Excel backup using OpenPyXL for reporting and data duplication.
6. Provide essential clinic record management functions such as add, view, search, update status, delete, clear fields, and exit system.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

7. Apply input validation to prevent incomplete, duplicate, and invalid patient records.

8. Test the functionality of the system to ensure that it performs according to its intended purpose.

Significance of the Study

This study is significant because it provides a computerized solution for improving clinic appointment and patient record management. MediTrack can help reduce paperwork, improve accuracy, and make patient records easier to organize, retrieve, and update.

Clinic Staff. The system can help clinic staff manage patient records faster and more systematically. Through the graphical interface, staff can easily add, view, search, update, and delete patient information without relying on manual logbooks.

Clinics. The system can help clinics improve productivity and efficiency by reducing manual work and minimizing errors in patient record handling. It also supports better appointment monitoring through organized status updates such as Scheduled, Checked-in, and Completed.

Patients. Patients may benefit from faster and more organized clinic services. Since records can be retrieved quickly and accurately, clinic staff can provide better assistance and reduce delays during appointment processing.

Students and Developers. This study can serve as a reference for students and future developers who want to create similar systems using Python, Tkinter, MySQL, and Excel integration. It demonstrates the practical application of object-oriented programming, database management, graphical user interface design, input validation, and system testing.

Scope and Limitations

This study focuses on the development of MediTrack, a desktop-based Clinic Appointment and Patient Record Management System using Python, Tkinter, MySQL, and OpenPyXL. The system includes basic clinic management functions such as adding patient records, viewing records, searching patient information, updating appointment status, deleting records, clearing input fields, and exiting the system. It also includes login, input validation, MySQL database storage, and automatic Excel backup.

The system handles patient information such as patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and appointment status. It supports basic CRUD operations, including creating, reading, updating, and deleting patient records.

1. **Create** - Insert the information of a new patient data to the database. Only accurate and complete information should be kept.

2. **Read** - Displays patient information in a structured table, allowing consistent and reliable access to records.

3. **Update** - Allows you to change the status or details of the patient so the database always has the latest information.

4. **Delete** - Safely removes data, to keep the system accurate and to avoid build-up of outdated records.

However, the system is limited to local desktop use only. The database runs on one computer and does not yet support online access, cloud storage, multi-user network access, SMS reminders, advanced user roles, or automatic report printing. The system is also mainly developed for academic and learning purposes. Future improvements may include separate account levels for doctors and receptionists, network connectivity, patient notification features, responsive interface design, and printable medical reports.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

REVIEW OF RELEVANT THEORY, STUDIES, AND LITERATURE

This chapter presents the relevant theories, studies, and literature that support the development of **MediTrack: A GUI-Based Clinic Appointment and Patient Record Management System with MySQL Storage and Excel Backup**. It discusses concepts related to electronic medical records, medical database systems, clinic appointment management, graphical user interface design, database management, Python programming, object-oriented programming, input validation, and Excel backup. These concepts provide the foundation for understanding the importance of developing a computerized system for managing patient records and clinic appointments.

Electronic Medical Records and Digital Clinic Systems

As information technology continues to improve, healthcare institutions are gradually shifting from manual record-keeping to electronic medical records. Electronic medical records, or EMRs, are digital versions of patient information that allow healthcare workers to store, update, retrieve, and manage medical data more efficiently. In clinic settings, EMRs help reduce paper-based work and improve the organization of patient information.

Manual clinic records are often prone to errors, misplaced files, unreadable handwriting, duplicated entries, and delayed record retrieval. These problems can affect the quality of service given to patients because clinic staff may spend more time looking for files instead of assisting patients. According to the uploaded research journal, manual appointment and patient record systems can result in confusion, inefficiency, missing medical histories, double booking, overlapping appointments, and longer waiting times.

The use of computerized clinic systems helps address these problems by allowing patient records to be stored and retrieved digitally. In MediTrack, patient information such as patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and appointment status can be saved in a structured database. This improves the accuracy, speed, and organization of clinic record management.

Medical Database Systems

A medical database system is a structured digital storage system used to manage patient health information. It allows medical and administrative staff to store patient records in organized tables and retrieve information whenever needed. Unlike manual records, a database system can keep data more secure, searchable, and permanent.

The uploaded Group 3 research journal explains that a Medical Database System provides a structured way of storing digital patient health information and helps healthcare providers access the information they need conveniently. It also states that medical database systems can improve the quality of patient care while reducing operational costs in the healthcare system.

In MediTrack, MySQL is used as the main database management system. MySQL stores patient data permanently so that records remain available even after the program is closed. This is important because clinic information must not be lost when the application stops running. Through MySQL, the system can create, read, update, and delete patient records efficiently.

Clinic Appointment Management

Clinic appointment management refers to the process of organizing patient schedules, monitoring appointment dates, assigning doctors, and updating the status of patient visits. In manual systems, appointment management is usually done through logbooks, printed forms, or handwritten notes. These methods may lead to problems such as overlapping schedules, missed appointments, and difficulty in tracking patient status.

A computerized appointment system helps clinic staff manage schedules more effectively. It can organize patient appointments in one platform and allow staff to update appointment status, such as Scheduled, Checked-in, or Completed. In MediTrack, the Update Record Status feature allows clinic staff to change the current status of a

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

patient's appointment directly from the dashboard. This helps the clinic monitor patient flow more clearly and efficiently.

Appointment management is important because it affects both clinic productivity and patient experience. When appointments are organized properly, clinic staff can serve patients faster and reduce waiting time. Patients also benefit because their records can be retrieved quickly and their appointment status can be monitored more accurately.

Graphical User Interface in Clinic Systems

A graphical user interface, or GUI, allows users to interact with a computer system through visual elements such as windows, buttons, text boxes, tables, labels, and menus. GUI-based systems are easier to use than command-line systems because users do not need to memorize text commands. Instead, they can click buttons and fill out forms.

In MediTrack, the GUI was developed using Tkinter. The main dashboard contains input fields for patient details, buttons for system functions, and a table for displaying stored records. The uploaded final research journal shows that the main GUI dashboard is the primary window where patient management tools, input forms, and data tables are located.

The GUI improves usability because clinic staff can manage records through a simple and organized screen. They can add records, view patient data, search information, update status, delete records, clear fields, and log out without using complex commands. This makes the system more user-friendly, especially for users who are not highly technical.

Python Programming in System Development

Python is a high-level programming language commonly used in system development because it is simple, readable, and flexible. It supports desktop application development, database connectivity, data processing, and automation. Python is also useful for academic projects because beginners can understand its syntax more easily compared to other programming languages.

In MediTrack, Python was used as the main programming language. It handles the system logic, user input, validation, database connection, record processing, and Excel backup. The uploaded Research Journal #3 states that Python was used because it is easy to read and write, and it allows the program to connect to a MySQL database to save patient records permanently.

Python also supports the use of libraries such as Tkinter, MySQL connector, Pillow, and OpenPyXL. Tkinter is used for the graphical user interface, MySQL connector is used for database communication, Pillow is used for image processing, and OpenPyXL is used for Excel file creation and backup. These tools help make MediTrack more functional and organized.

Object-Oriented Programming Concepts

Object-Oriented Programming, or OOP, is a programming approach that organizes code using objects, classes, functions, and reusable components. Since MediTrack was developed under an Object-Oriented Programming course, the project applies programming concepts such as functions, loops, and conditional statements.

Functions help divide the program into smaller and more organized parts. For example, functions can be used for adding patient records, searching records, updating appointment status, deleting data, clearing fields, and connecting to the database. This makes the system easier to understand and maintain.

Loops allow the program to continue running until the user chooses to exit. Conditional statements allow the system to make decisions, such as checking whether a field is empty, whether the patient ID already exists, or whether the contact number contains exactly eleven digits. The uploaded Research Journal #3 explains that the system uses functions, loops, and conditional statements to process user input and produce output.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

These OOP concepts support the development of MediTrack by making the code more organized, reusable, and easier to debug.

Database Management Using MySQL

Database management is an important part of any patient record system. A database allows information to be stored in an organized and permanent way. Without a database, data may only exist temporarily while the program is running and may be lost once the system is closed.

The uploaded Research Journal #4 explains that MySQL was used to store and manage all patient information. It also explains that databases are important because they save data permanently, keep information organized, protect private data, and allow users to find records quickly.

The database table used in the system is named **patients**. It contains the following fields: patient ID, name, age, contact, symptoms, doctor, appointment date, and status. These fields allow the system to store patient information in

a structured format. Through MySQL, MediTrack can add new patient records, display stored records, update appointment status, delete records, and search patient information.

Python connects to MySQL using a connector. The connection process includes providing the database address, username, password, and database name. The system uses a cursor to send SQL commands, commits changes to save data permanently, and closes the connection after completing a task. This process ensures that patient records are properly saved and managed.

CRUD Operations in Patient Record Management

CRUD stands for Create, Read, Update, and Delete. These are the basic operations used in most record management systems. In MediTrack, CRUD operations allow clinic staff to manage patient records efficiently.

Create refers to adding new patient records into the database. This function is used when a new patient visits the clinic or needs to be scheduled for an appointment. Read refers to viewing or searching patient records from the database.

This allows clinic staff to retrieve patient information quickly. Update refers to modifying patient information or changing the appointment status. Delete refers to removing records that are no longer needed or were entered incorrectly.

The uploaded final research journal explains that the system supports CRUD operations to ensure reliable data management. It also describes the functions for adding, viewing, updating, and deleting patient records.

CRUD operations make MediTrack useful because they allow users to manage the complete life cycle of a patient record, from creation to updating and removal.

Input Validation and Error Prevention

Input validation is the process of checking user input before saving it into the system. It prevents incorrect, incomplete, or invalid data from being stored in the database. This is important in clinic systems because patient records must be accurate and reliable.

MediTrack includes input validation to prevent common errors. The system checks for blank fields, duplicate patient IDs, numbers in the name field, invalid age input, and contact numbers that are shorter or longer than eleven digits. The uploaded final research journal shows that these validation tests were conducted and all were marked as PASS.

Input validation improves the quality of data stored in the system. For example, requiring the contact number to

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

contain exactly eleven digits helps ensure that the phone number is complete. Preventing duplicate patient IDs helps avoid confusion between patient records. Blocking numbers in the name field helps maintain proper formatting of patient information.

Through validation, MediTrack reduces human errors commonly found in manual logbooks and improves the accuracy of patient record management.

Excel Backup and Reporting

Excel backup is an additional feature that allows the system to save a duplicate copy of patient records into a spreadsheet file. This is useful for reporting, record review, and backup purposes. If there is a problem with the database, the Excel file can serve as an extra copy of the saved patient records.

MediTrack uses OpenPyXL to automatically append patient records into an Excel spreadsheet. The uploaded final journal states that the system automatically duplicates and saves every new patient entry into an Excel spreadsheet, giving the clinic an easier way to view reports and keep a backup of records.

This feature improves data safety because records are stored in both the MySQL database and an Excel file. It also supports easier reporting because spreadsheet files can be opened, reviewed, printed, or shared when needed.

Related Systems

Several existing systems are related to MediTrack. Examples include electronic medical record systems, clinic management systems, appointment scheduling systems, and hospital information systems. These systems help healthcare providers manage patient information, appointments, medical history, and clinic workflows.

The uploaded Research Journal #1 mentioned systems such as Clinicea and CureMD EHR as examples of modern healthcare systems that support secure, standardized, and shareable patient records. These systems show how digital healthcare platforms can improve record safety, reduce physical file loss, and make clinic operations more efficient.

Compared with larger professional healthcare systems, MediTrack focuses on basic clinic record and appointment management. It is designed for academic and learning purposes, but it applies important concepts found in real healthcare systems, such as database storage, user login, patient record management, appointment monitoring, data validation, and backup.

Synthesis of Reviewed Theory, Studies, and Literature

The reviewed theories, studies, and literature show that computerized systems are important in improving clinic operations and patient record management. Manual systems can cause delays, lost files, duplicate entries, and inaccurate records. Electronic medical records and medical database systems help solve these problems by storing patient information digitally and making records easier to retrieve.

The literature also supports the use of Python, Tkinter, MySQL, and OpenPyXL in system development. Python provides the main programming logic, Tkinter supports the graphical user interface, MySQL provides permanent database storage, and OpenPyXL supports Excel backup. These tools work together to make MediTrack functional, organized, and user-friendly.

Overall, the reviewed materials support the development of MediTrack as a practical solution for managing clinic appointments and patient records. The system improves record accuracy, reduces manual work, prevents common input errors, and provides safer storage through both MySQL and Excel backup.

Conceptual Framework

The conceptual framework of this study follows the Input-Process-Output model.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

The input includes patient information such as patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and appointment status. These data are entered by the user through the graphical user interface.

The process includes user login, input validation, database connection, record creation, record viewing, searching, appointment status updating, record deletion, field clearing, and Excel backup. The system checks whether the entered data are valid before saving them into the MySQL database and Excel file.

The output includes organized patient records, updated appointment status, search results, deleted or cleared records, database-stored patient information, and Excel backup files. These outputs help clinic staff manage patient information and appointments more efficiently.

Through this framework, MediTrack transforms raw patient information into organized, accurate, and accessible clinic records.

METHODOLOGY

This chapter presents the research design, system development process, system design, database design, tools and technologies used, system features, testing procedure, and evaluation method used in the development of **MediTrack: A GUI-Based Clinic Appointment and Patient Record Management System with MySQL Storage and Excel Backup**. It explains how the system was planned, designed, developed, tested, and implemented to provide a more organized and efficient way of managing clinic appointments and patient records.

Research Design

This study used a system development approach. This design was appropriate because the main purpose of the study was to develop a functional desktop-based clinic management system that can record, organize, retrieve, update, delete, and back up patient information.

The system was developed to address common problems in manual clinic record management, such as misplaced records, duplicate patient entries, slow retrieval of patient information, inaccurate data entry, and difficulty in monitoring appointment status. Through the development of MediTrack, the researchers aimed to provide clinic staff with a computerized system that can improve the accuracy, speed, and organization of patient record handling.

MediTrack was developed using **Python** as the main programming language, **Tkinter** for the graphical user interface, **MySQL** for database storage, and **OpenPyXL** for automatic Excel backup. These tools were selected because they support desktop application development, data validation, database management, and spreadsheet generation.

System Development Process

The development of MediTrack followed several phases to ensure that the system would meet its intended purpose and function correctly.

Planning

During the planning stage, the researchers identified the problems commonly encountered in manual clinic appointment and patient record management. These problems include lost paper records, delayed searching of patient information, duplicate entries, and difficulty in updating appointment status.

The researchers also identified the needed system features, such as login, add record, view records, search patient, update appointment status, delete record, clear fields, logout, MySQL database storage, and Excel backup. These features were selected to make the system more useful and practical for basic clinic record management.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

System Design

The system design stage involved creating the flow of the application and planning how users would interact with the system. A flowchart was used to show how the system works from start to finish. The system begins with a login or registration process before allowing the user to access the main dashboard.

After logging in, the user is directed to the dashboard, where the main functions of the system are available. These functions include adding a patient record, viewing saved records, searching for patient information, updating appointment status, deleting records, clearing input fields, and logging out of the system. After completing a task, the system returns to the main dashboard so the user can perform another action.

Development

The development stage involved coding the system using Python. Tkinter was used to create the graphical interface, including labels, entry fields, buttons, dropdown menus, tables, and message boxes. The interface was designed to be simple and easy for clinic staff to use.

Python functions were created for each major task of the system. These include functions for logging in, adding patient records, displaying records, searching records, updating appointment status, deleting records, clearing fields, connecting to the database, and saving backup records to Excel.

The system also applied important programming concepts such as functions, loops, and conditional statements. Functions were used to organize the code into separate tasks. Conditional statements were used to check user inputs and determine what action should be performed. Validation checks were also included to prevent incorrect or incomplete data from being saved.

Database Integration

MySQL was integrated into the system to provide permanent storage for patient records. The system connects to the MySQL database using a Python connector. Once connected, the system can send SQL commands to insert, retrieve, update, and delete patient records.

The database allows patient information to remain saved even after the system is closed. This makes the system more reliable compared to temporary storage using only lists or dictionaries. MySQL also helps keep the records organized in rows and columns, making it easier to manage patient data.

Excel Backup Integration

OpenPyXL was used to create an automatic Excel backup feature. Every time a new patient record is saved, the system also appends the same information to an Excel spreadsheet. This provides an additional copy of the patient record for backup and reporting purposes.

The Excel backup helps improve data safety because the records are stored not only in the MySQL database but also in a spreadsheet file. This feature can also help clinic staff review or print records more easily when needed.

Testing and Debugging

After development, the system was tested to check whether all features worked correctly. The researchers tested valid and invalid inputs to determine if the system could properly handle different situations.

Testing included checking blank inputs, duplicate patient IDs, numbers entered in the name field, invalid age input, and incorrect contact numbers. The system was also tested to confirm whether it could successfully add, view, search, update, delete, and back up patient records.

Errors found during testing were corrected through debugging. The researchers reviewed the code, fixed logical errors, improved validation, and ensured that the system worked properly with the MySQL database and Excel

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

backup file.

Implementation

After testing and debugging, MediTrack was implemented as a desktop-based clinic appointment and patient record management system. The final system allows clinic staff to manage patient records using a graphical interface and store the records in both MySQL and Excel.

The implemented system provides a faster, cleaner, and more organized alternative to manual paper-based clinic records.

System Design

MediTrack starts with a login screen where the user must enter valid account credentials before accessing the system. This login process helps protect patient records from unauthorized access.

After successful login, the system opens the main dashboard. The dashboard contains the patient details form, action buttons, search bar, and patient record table. The patient details form allows the user to enter information such as patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and appointment status.

When the user clicks **Add Record**, the system checks whether the entered information is complete and valid. If the input contains errors, the system displays a warning message. If the information is valid, the record is saved to the MySQL database and backed up to an Excel file.

When the user clicks **View Records**, the system displays all saved patient records in a table. The user can also use the search function to locate a specific patient record.

When the user clicks **Update Record Status**, the system allows the user to change the appointment status of a selected patient, such as Scheduled, Checked-in, or Completed.

When the user clicks **Delete Record**, the system removes the selected patient record from the database after confirmation.

When the user clicks **Clear Fields**, the system clears all input boxes so the user can enter another patient record.

When the user clicks **Logout** or exits the system, the program closes safely while keeping all saved records stored in the database and Excel backup.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

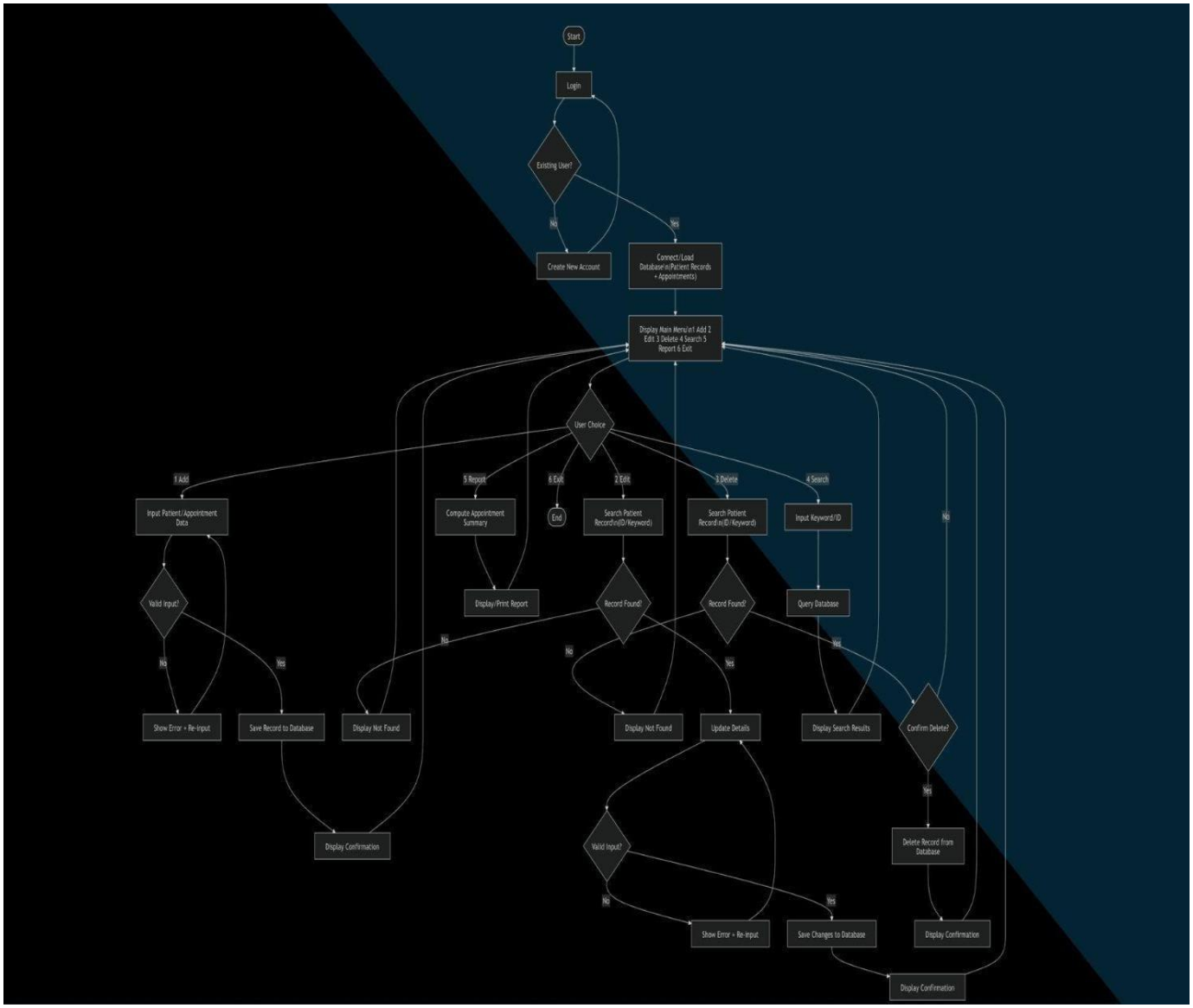


Figure 1. Flowchart of Clinic Appointment And Patient Record Management System

This flowchart maps out how your clinic program works from start to finish. It begins at the top, where the user to log in or create a new account before opening the database. Once you are logged in, you land on a central "Dashboard" that lets you choose between six different tasks: adding, editing, deleting, searching, updating patient reports, or exiting the system. Finally, all the long lines on the outside of the chart show that after the program finishes any task, it automatically loops right back to the main menu so the clinic worker can immediately start the next job.

Database Design

The system uses MySQL as its database management system. The main table used in the system is the **patients** table. This table stores the patient information needed for clinic appointment and record management.

The table includes the following fields:

Field Name	**Data Type**	**Description**
patient_id	VARCHAR	A unique identification code for each patient.
name	VARCHAR	The full name of the patient.
age	INT	The age of the patient.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

contact	VARCHAR	The patient's contact number.
symptoms	TEXT	The symptoms or reason for the patient's visit.
doctor	VARCHAR	The assigned doctor for the patient.
appointment_date	VARCHAR	The scheduled appointment date.
status	VARCHAR	The current appointment status, such as Scheduled, Checked-in, or Completed.

This database structure allows the system to store patient records in an organized way. It also supports the major CRUD operations: creating new records, reading or viewing saved records, updating appointment status, and deleting unnecessary records.

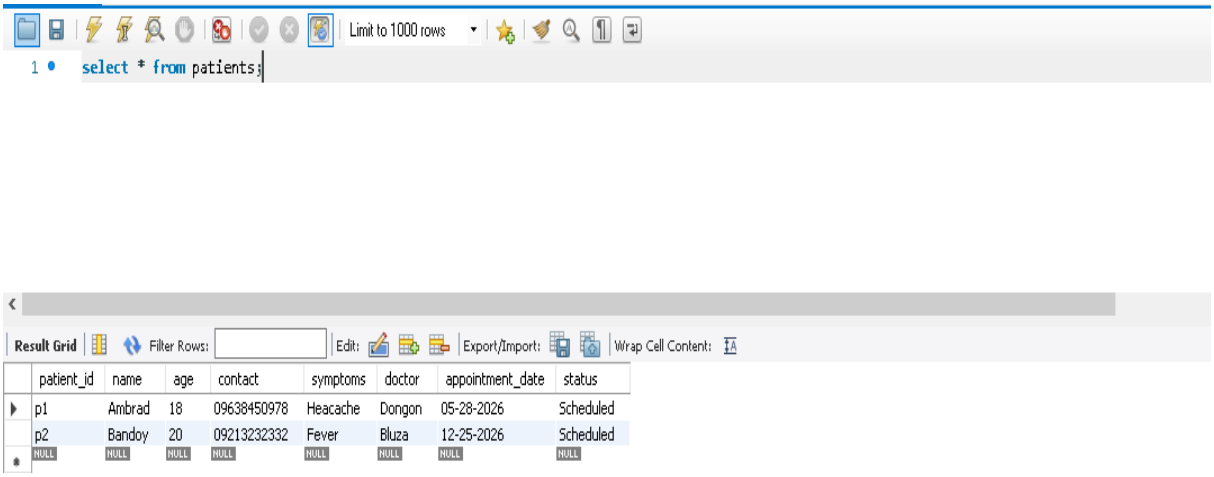


Figure 2. MySQL Database Table for Stored Patient Records

This figure shows the patients table in MySQL Workbench, where patient information is stored in the MediTrack system. The table includes important fields such as patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and appointment status. It confirms that patient records are successfully saved and retrieved from the database.

Tools and Technologies Used

The following tools and technologies were used in the development of MediTrack:

Python was used as the main programming language. It handled the system logic, input validation, database connection, record processing, and Excel backup.

Tkinter was used to create the graphical user interface. It provided the buttons, text fields, labels, dropdown menus, tables, and message boxes used in the system.

MySQL Workbench was used for database management. It stored patient records permanently and allowed the system to retrieve, update, and delete records.

OpenPyXL was used for Excel backup. It allowed the system to automatically save patient records into an Excel spreadsheet.

Pillow Library was used for image processing and interface design support.

PyCharm IDE was used as the development environment for writing, editing, testing, and debugging the program code.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

System Features

MediTrack includes the following features:

Login and Registration allows users to access the system through an account-based entry process.

Add Record allows clinic staff to save new patient information, including patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and status.

View Records displays all saved patient records in a structured table.

Search Patient allows users to find specific patient records quickly.

Update Record Status allows the user to change the appointment status of a selected patient.

Delete Record removes selected patient records from the database after confirmation.

Clear Fields clears all input fields to prepare the form for a new entry.

Excel Backup automatically saves a duplicate copy of patient records in an Excel spreadsheet.

Input Validation checks blank fields, duplicate patient IDs, invalid names, incorrect age input, and invalid contact numbers.

Exit or Logout allows the user to safely close or leave the system.

Testing Procedure

The system was tested using functional testing. Each major feature was tested to determine whether it produced the expected result. The researchers tested both valid and invalid inputs to ensure that the system could prevent common errors before saving records.

The following test conditions were used:

Test Case	Expected Result
Blank input fields	The system blocks saving and shows a warning message.
Duplicate patient ID	The system prevents the entry to avoid duplicate records.
Numbers entered in the name field	The system shows an error asking for letters only.
Letters entered in the age field	The system shows an error asking for a whole number.
Contact number shorter or longer than 11 digits	The system shows an error asking for exactly 11 digits.
Add valid patient record	The record is saved successfully in MySQL and Excel.
View patient records	The system displays all saved records in the table.
Search patient record	The system displays the matching patient record.
Update appointment status	The selected patient’s status is updated successfully.
Delete patient record	The selected record is removed from the database.
Clear fields	All input fields are cleared.
Logout or exit system	The system closes or returns safely without data loss.

The testing procedure helped confirm whether MediTrack could perform its intended functions and handle incorrect user inputs properly.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Evaluation of the System

The system was evaluated based on functionality, accuracy, usability, reliability, and data management.

Functionality was evaluated by checking whether the system could successfully add, view, search, update, delete, clear, and back up patient records.

Accuracy was evaluated by checking whether the system prevented invalid information from being saved, such as blank fields, duplicate patient IDs, invalid names, incorrect age input, and invalid contact numbers.

Usability was evaluated based on the clarity and simplicity of the graphical user interface. The use of forms, buttons, tables, and message boxes made the system easier to operate.

Reliability was evaluated by checking whether the system consistently saved records in the MySQL database and Excel backup file.

Data management was evaluated by checking whether records were stored permanently, displayed properly, updated correctly, and deleted when needed.

Summary

The methodology of this study followed a system development process consisting of planning, system design, development, database integration, Excel backup integration, testing, debugging, and implementation. MediTrack was developed using Python, Tkinter, MySQL, OpenPyXL, Pillow, and PyCharm.

Through this methodology, the researchers were able to create a functional desktop-based clinic appointment and patient record management system. The system supports login, patient record management, appointment status updates, input validation, MySQL storage, and Excel backup. It demonstrates the practical application of object-oriented programming, graphical user interface development, database management, and system testing.

RESULTS AND DISCUSSION

This chapter presents the results and discussion of the developed **MediTrack: A GUI-Based Clinic Appointment and Patient Record Management System with MySQL Storage and Excel Backup**. It discusses the system overview, completed system features, graphical user interface, testing results, and overall performance of the system. The results show how MediTrack was able to manage patient records and clinic appointments through Python, Tkinter, MySQL, and Excel backup integration.

System Overview

MediTrack was developed as a desktop-based clinic appointment and patient record management system. The system was designed to help clinic staff manage patient information and appointment records in a faster, more organized, and more accurate way compared to manual paper-based record keeping.

The system uses **Python** as the main programming language, **Tkinter** for the graphical user interface, **MySQL** for database storage, and **OpenPyXL** for automatic Excel backup. Through these tools, the system provides a functional platform where clinic staff can add, view, search, update, delete, and back up patient records.

The system begins with a login or registration screen before allowing the user to access the main dashboard. After logging in, the user can manage patient records through the available buttons and input fields. The dashboard serves as the main working area where patient details are entered and where saved records are displayed in a table.

Overall, MediTrack provides an organized and user-friendly clinic management tool. It helps reduce manual work, prevents common encoding errors, and keeps patient information stored in both a MySQL database and an Excel backup file.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

System Features and Results

The developed MediTrack system includes several features that support clinic appointment and patient record management. Each feature was designed to improve the accuracy, speed, and organization of clinic data handling.

Login and Registration

The login and registration feature provides an entry point for users before accessing the system dashboard. This feature helps protect patient records by allowing only registered users to enter the system.

The result showed that the login and registration feature successfully directed authorized users to the main dashboard. This adds a basic layer of security and helps prevent unauthorized access to patient records.

Add Record

The Add Record feature allows the user to encode and save new patient information. The system accepts details such as patient ID, name, age, contact number, symptoms, assigned doctor, appointment date, and appointment status.

The result showed that the Add Record feature successfully saved valid patient information into the MySQL database. At the same time, the system automatically appended the saved information into an Excel file for backup. This means that every new patient entry is stored in two locations: the database and the spreadsheet backup.

The system also checked the user's input before saving. If a required field was empty, if the patient ID already existed, if the name contained numbers, if the age was not a whole number, or if the contact number was not exactly eleven digits, the system displayed an error message and prevented the record from being saved.

View Records

The View Records feature displays all saved patient records in an on-screen table. This allows clinic staff to see patient information in an organized format without checking paper files or logbooks.

The result showed that the View Records feature successfully loaded stored patient records from the database and displayed them in clear columns. This made it easier for users to review patient information, check appointments, and monitor records.

Search Patient Record

The Search feature allows users to locate a specific patient record more quickly. Instead of manually scanning through all records, the user can search for patient information using the available search field.

The result showed that the system was able to retrieve matching patient records and display them in the table. This feature improves efficiency because clinic staff can find patient information faster.

Update Record Status

The Update Record Status feature allows the user to change the current appointment status of a selected patient. Examples of appointment status include **Scheduled**, **Checked-in**, and **Completed**.

The result showed that the system successfully updated the appointment status of selected patient records. This feature helps clinic staff monitor patient progress and appointment flow more effectively.

Delete Record

The Delete Record feature allows users to remove selected patient records from the system. This is useful when

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

a record is incorrect, duplicated, or no longer needed.

The result showed that the Delete Record feature successfully removed selected patient records from the database. Since deletion affects stored data, the system should be used carefully to avoid removing important patient information.

Clear Fields

The Clear Fields feature empties all input boxes in the form. This allows the user to prepare the form for a new patient entry without manually deleting each field.

The result showed that the Clear Fields feature worked properly. It helped users save time and reduced the chance of accidentally mixing old and new patient information.

Excel Backup

The Excel Backup feature automatically saves a duplicate copy of patient information into an Excel spreadsheet. This feature was made possible through OpenPyXL.

The result showed that the system successfully backed up newly added patient records into an Excel file. This provides additional data safety and makes it easier to view, print, or use patient records for reporting purposes.

Exit System

The Exit System feature allows the user to close the program safely. The result showed that the system closed properly while keeping saved records stored in the MySQL database and Excel backup file.

System Interface

The MediTrack system uses a graphical user interface developed with Tkinter. The interface includes a login screen, a main dashboard, input fields, action buttons, search functions, and a table for displaying records.

Main GUI Dashboard: The primary window where all patient management tools, input forms, and data tables are located.

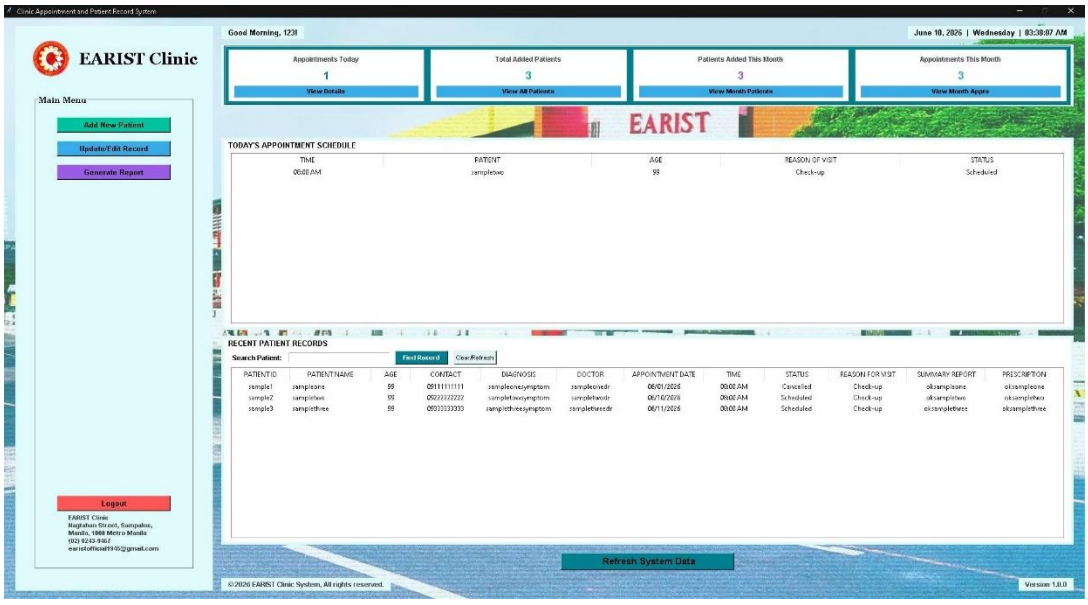


Figure 4.1. Main GUI Dashboard

The Main GUI Dashboard serves as the primary window of the system. It contains the patient management tools, input forms, buttons, and data table. Through this dashboard, the user can add, view, search, update, delete, and

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

clear patient records

Add Record: Automatically parses and tests raw input data fields, running execution scripts that write to active SQL tables and append rows to an Excel workbook.

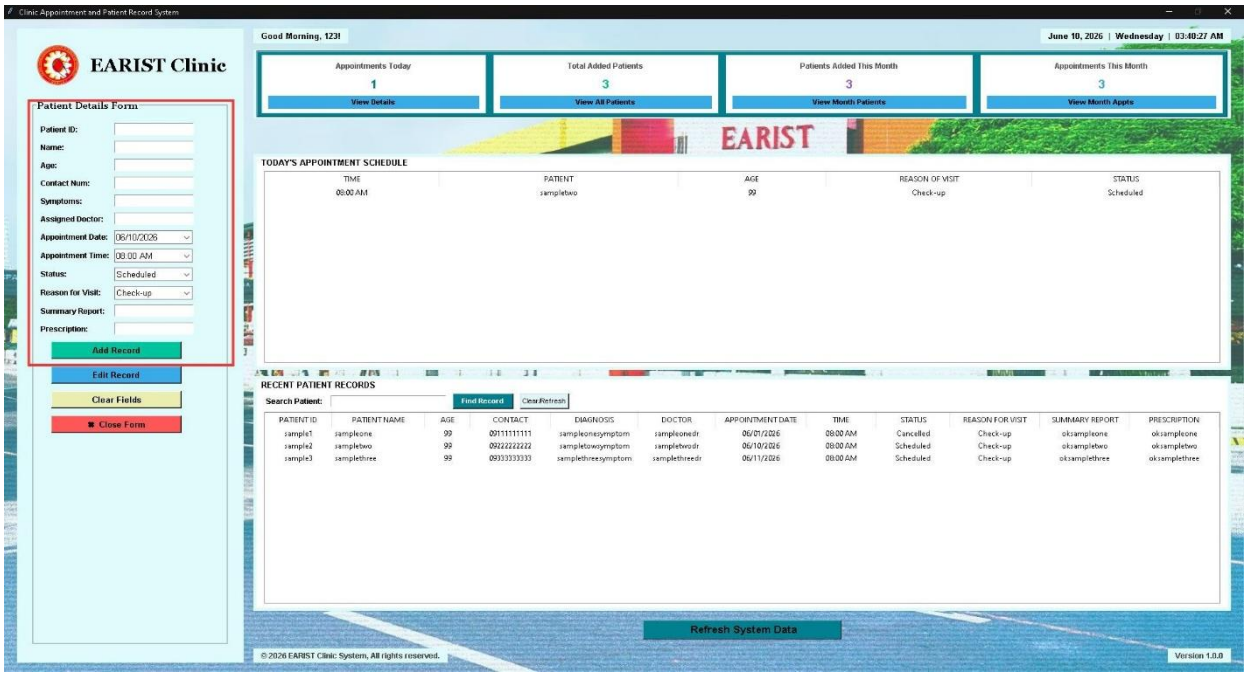
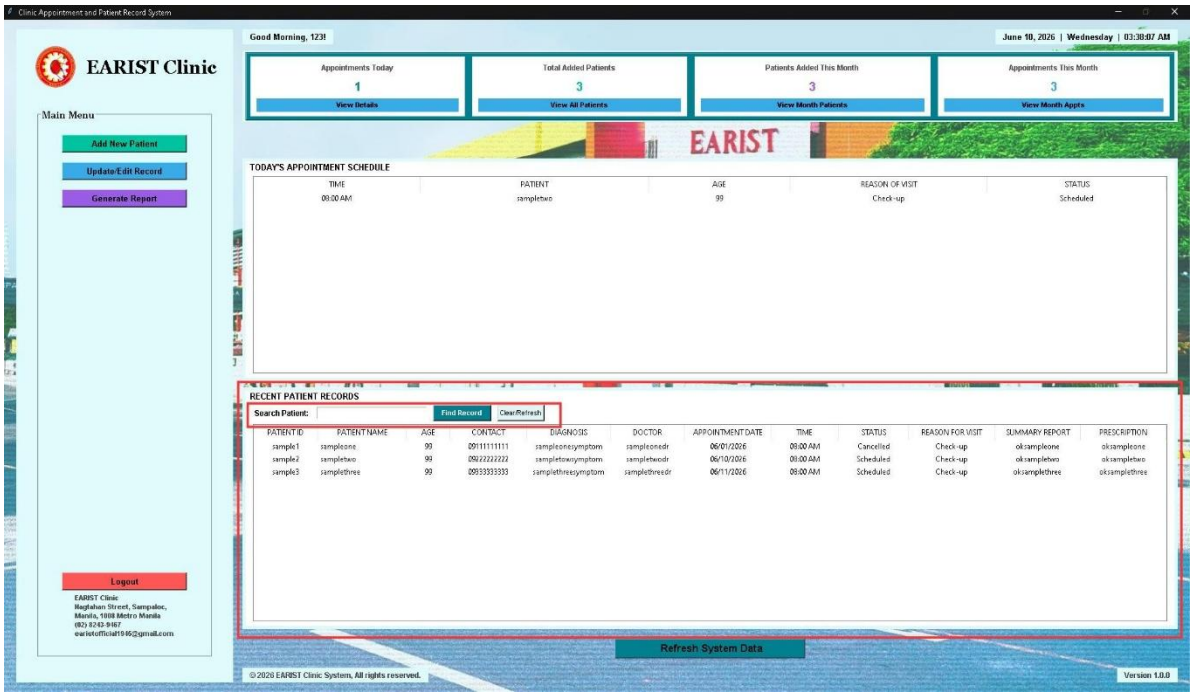


Figure 4.2. Add Record Interface

The Add Record interface allows users to encode new patient details. It includes input fields for patient ID, name, age, contact number, symptoms, doctor, appointment date, and status. Once the user submits valid data, the system saves the record into MySQL and automatically creates an Excel backup.

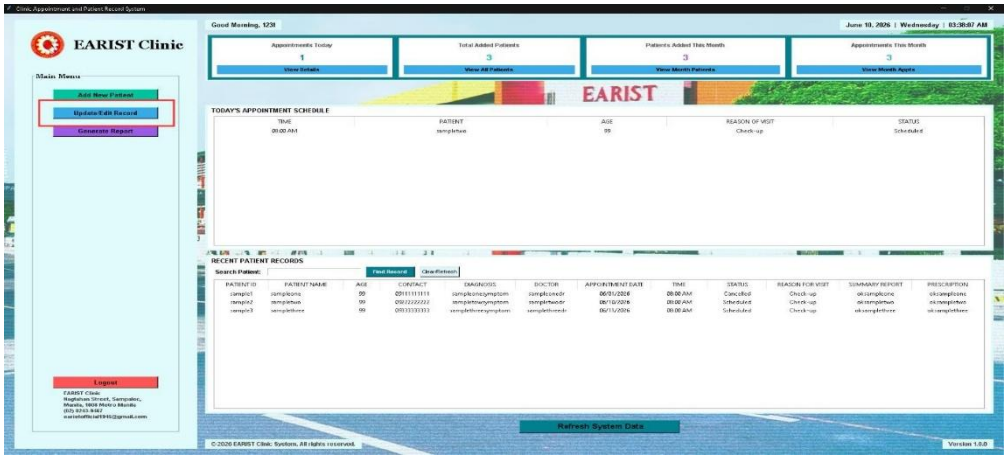


View Records: Loads records into an interactive layout, complete with clear data columns.

Figure 4.3. View Records Interface

The View Records interface displays all stored patient rows in a table. This helps clinic staff review patient records in a clear and organized format.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026



Update Record Status: Changes a patient's current appointment status (such as Scheduled, Checked-in, or completed) directly from the dashboard.

Figure 4.4. Update Record Status Interface

The Update Record Status interface allows the user to change the appointment status of a selected patient. This helps the clinic monitor whether a patient is scheduled, checked in, or completed

The graphical user interface improved the usability of the system because users can perform tasks through buttons, fields, and tables instead of using command-line instructions. This makes the system easier to understand and operate, especially for clinic staff who may not be familiar with programming commands.

System Testing Results

The system was tested to determine whether its main functions worked correctly. Functional testing was conducted using both valid and invalid inputs. The purpose of the testing was to check if MediTrack could process correct information, block incorrect entries, save records to the database, and maintain accurate patient information.

Test Case	What Was Done	Expected Result	Actual Result	Status
Blank Inputs	Left a required text box empty	The system should block saving and display a warning message.	The system blocked saving and showed a warning message.	PASS
Duplicate Patient ID	Entered a patient ID that already exists	The system should prevent duplicate patient records.	The system blocked the entry to avoid duplicate errors.	PASS
Numbers in Name Field	Entered numbers in the name field	The system should reject the input and ask for letters only.	The system showed an error asking for letters only.	PASS
Invalid Age Input	Entered letters in the age field	The system should reject the input and ask for a whole number.	The system showed an error asking for a whole number.	PASS
Invalid Contact Number	Entered a contact number shorter or longer than 11 digits	The system should reject the input and ask for exactly 11 digits.	The system showed an error asking for exactly 11 digits.	PASS
Add Valid Patient Record	Entered complete and valid patient information	The system should save the record in MySQL and Excel.	The record was saved successfully in the database and backup file.	PASS

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

View Patient Records	Clicked the View Records function	The system should display all saved records.	The system displayed patient records in the table.	PASS
Search Patient Record	Searched for an existing patient record	The system should display the matching record.	The matching record was shown in the table.	PASS
Update Appointment Status	Selected a patient and changed the status	The system should update the selected patient's status.	The appointment status was updated successfully.	PASS
Delete Patient Record	Selected a patient record and deleted it	The system should remove the selected record.	The selected record was deleted successfully.	PASS
Clear Fields	Clicked the Clear Fields button	All input fields should become empty.	The input fields were cleared successfully.	PASS
Exit System	Clicked the Exit button	The system should close safely.	The system closed properly without data loss.	PASS

The testing results showed that MediTrack successfully performed its intended functions. The system was able to add, view, search, update, delete, clear, and save patient records. It also prevented invalid entries from being stored in the system.

DISCUSSION OF FINDINGS

The results show that MediTrack successfully addressed the problems commonly found in manual clinic record management. Manual recording can lead to lost records, duplicate entries, inaccurate information, and slow retrieval of patient files. Through MediTrack, patient records are stored digitally and can be accessed more efficiently.

The use of MySQL improved the reliability of the system because patient data are saved permanently in a structured database. This ensures that records remain available even after the program is closed. The Excel backup also provides additional safety because every new record is duplicated into a spreadsheet file.

The system also improved accuracy through input validation. By checking blank fields, duplicate IDs, invalid names, incorrect age input, and invalid contact numbers, the system prevents common user errors before they are saved. This helps maintain cleaner and more reliable clinic records.

The update status function also supports better appointment monitoring. Clinic staff can easily update whether a patient is scheduled, checked in, or completed. This makes patient flow easier to manage and reduces confusion during clinic operations.

The graphical user interface contributed to the system's usability. Since users can interact with the system through buttons, input fields, and tables, the system is easier to operate than manual logbooks or command-line programs.

Overall, the findings show that MediTrack is functional, organized, and useful for basic clinic appointment and patient record management.

Strengths of the System

MediTrack has several strengths based on the results of development and testing. First, it provides a user-friendly dashboard for managing patient records. Second, it uses MySQL for permanent record storage. Third, it automatically creates an Excel backup for additional data safety. Fourth, it includes input validation that prevents invalid or incomplete records. Fifth, it supports basic clinic management functions such as add, view, search, update, delete, clear fields, and exit.

These strengths show that the system can help clinic staff work faster, reduce paperwork, and manage patient

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

information more accurately.

Limitations Observed

Although MediTrack successfully performed its intended functions, some limitations were observed. The system is designed for local desktop use only, which means the database runs on one computer. It does not yet support online access, cloud storage, or multi-user network access.

The system also has limited update functionality because it mainly allows users to update appointment status rather than editing all patient details. In addition, it does not yet include advanced features such as SMS reminders, role-based accounts for doctors and receptionists, automatic report printing, or cloud backup.

These limitations show that the system is functional within its current scope but can still be improved in future versions.

Summary

The results showed that MediTrack successfully performed its major functions, including login, adding patient records, viewing records, searching patient information, updating appointment status, deleting records, clearing fields, saving data to MySQL, and creating Excel backups. The system also passed important validation tests, including blank inputs, duplicate patient IDs, invalid names, invalid age input, and incorrect contact numbers.

The discussion showed that the system improved the accuracy, organization, and efficiency of clinic record management. Through its graphical interface, database storage, and Excel backup feature, MediTrack provides a practical and reliable solution for managing clinic appointments and patient records within the scope of the study.

CONCLUSIONS AND RECOMMENDATIONS

Conclusions

Based on the development, testing, and evaluation of **MediTrack: A GUI-Based Clinic Appointment and Patient Record Management System with MySQL Storage and Excel Backup**, the researchers concluded that the system successfully achieved its main objective of creating a desktop-based application that can help clinic staff manage patient records and appointments in a more organized, accurate, and efficient manner.

The system effectively addressed common problems encountered in manual clinic record management, such as misplaced files, duplicate records, slow retrieval of patient information, inaccurate entries, and difficulty in monitoring appointment status. Through the use of a computerized system, clinic staff can add, view, search, update, and delete patient records faster compared to paper-based methods.

MediTrack also proved useful in improving data storage and record safety. The integration of **MySQL** allowed patient records to be stored permanently in a structured database. This ensured that patient information remained available even after the program was closed. In addition, the use of **OpenPyXL** provided an automatic Excel backup, allowing every new patient entry to be duplicated into a spreadsheet file for reporting and backup purposes.

The system also improved the accuracy of patient records through input validation. It was able to detect blank fields, duplicate patient IDs, invalid names, incorrect age input, and contact numbers that did not contain exactly eleven digits. These validation features helped prevent incorrect or incomplete data from being saved into the system.

The graphical user interface developed using **Tkinter** made the system easier to use and understand. Through the use of buttons, input fields, tables, and message boxes, users were able to interact with the system without relying on manual logbooks or command-based instructions. The dashboard provided a simple and organized workspace for managing clinic records.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

The results of functional testing showed that the system performed its intended functions successfully. It was able to add patient records, view saved records, search patient information, update appointment status, delete records, clear input fields, save data to MySQL, and create Excel backups. The testing also showed that the system properly blocked invalid inputs and displayed warning messages when necessary.

Overall, MediTrack can be considered a functional and reliable system within the scope of the study. It provides a practical solution for basic clinic appointment and patient record management while demonstrating the application of Python programming, graphical user interface development, MySQL database integration, Excel backup, input validation, and object-oriented programming concepts.

Recommendations

Based on the findings and limitations of the study, the following recommendations are proposed for the improvement of MediTrack:

1. Future developers should add separate account levels for different users, such as doctors, receptionists, and administrators. This would improve data security by limiting access based on the role of each user.

2. The login and registration data should be stored permanently in the MySQL database instead of temporary storage. This would prevent newly registered accounts from being deleted when the program restarts.

3. The system should allow full editing of patient details, not only the appointment status. This would help clinic staff correct or update information such as contact number, symptoms, doctor, or appointment date when needed.

4. A network-based or online version of the system may be developed so that different clinic computers or rooms can access the same patient records. This would make the system more useful for clinics with multiple staff members.

5. The system may include patient notification features, such as SMS or email reminders, to remind patients about their scheduled appointments and reduce missed visits.

6. Future versions should include automatic report printing or report export options. This would allow clinic staff to generate printed summaries, appointment lists, or patient reports more conveniently.

7. The graphical user interface may be improved by making the layout more modern, responsive, and adjustable to different screen sizes. A more flexible design would improve user experience.

8. The system should include automatic backup and recovery features to further protect patient data in case of database errors, accidental deletion, or computer problems.

9. Search and filtering features may be improved by allowing users to search records by doctor, appointment date, status, or symptoms. This would make record retrieval faster and more specific.

10. Future researchers may conduct usability testing with actual clinic staff to gather feedback about the system’s interface, performance, and usefulness. Their feedback can guide further improvements in the system.

In conclusion, MediTrack already performs its intended functions effectively, but it can still be enhanced through stronger security, permanent user account storage, full record editing, network access, patient alerts, improved reporting, and better interface design. These improvements would make the system more efficient, secure, and suitable for broader clinic use.

ACKNOWLEDGEMENT

The researchers would like to extend their deepest appreciation to **Engr. Meshelle N. Fabro, PCpE**, for her

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

invaluable guidance, patient mentorship, and profound expertise throughout the development of this study. Her critical insights, unwavering encouragement, and professional direction played a pivotal role in refining the quality and completion of this research.

The researchers also express their profound gratitude to the **Eulogio "Amang" Rodriguez Institute of Science and Technology (EARIST)**, specifically the **Department of Computer Engineering**, along with its dedicated faculty and staff. The institutional support, academic foundation, and technical facilities provided by the university were indispensable to the realization of this project. The department's commitment to cultivating innovation and excellence served as a constant inspiration.

Special thanks are also given to the families and friends of the researchers for their endless love, understanding, and emotional support. Their presence and motivation provided a vital source of strength and resilience during the most challenging periods of this academic journey.

Above all, the researchers offer their ultimate praise and thanksgiving to the **Almighty God** for granting the intellect, clarity of mind, and physical strength required to overcome every obstacle. This achievement is humbly dedicated to Him, in recognition that all breakthroughs and successes are gifts of His divine grace.

REFERENCES

1. Asih, H., & Indrayadi, I. (2024). Transformation from manual medical records to electronic medical records: A phenomenological study. *Contagion: Scientific Periodical Journal of Public Health and Coastal Health*, *6*, 25. https://doi.org/10.30829/contagion.v6i1.19188
2. MySQL. (n.d.). *MySQL documentation*. Oracle. OpenPyXL. (n.d.). *OpenPyXL documentation*.
3. Python Software Foundation. (n.d.). *Python documentation*.
4. Python Software Foundation. (n.d.). *Tkinter — Python interface to Tcl/Tk*. Tkinter Python. (n.d.). *Tkinter Python tutorials* [YouTube channel]. YouTube.
5. Xhafa, F., Li, J., Zhao, G., Li, J., Chen, X., & Wong, D. S. (2015). Designing cloud-based electronic health record system with attribute-based encryption, 3441–3458.
6. **ABOUT THE AUTHORS**
7. **Jhasfer Moi Ambrad** is a first-year college student at Eulogio "Amang" Rodriguez Institute of Science and Technology (EARIST). He is currently studying for his Bachelor of Science in Computer Engineering (BSCpE). He has already learned foundational skills in C++ programming, Object-Oriented Programming (OOP), and Computer-Aided Design (CAD). He is dedicated to building strong technical and engineering skills at EARIST to design and develop smart technologies in the future.
8. **Joel Jr. D. Bluza** is a first-year student taking a Bachelor of Science in Computer Engineering at the Eulogio "Amang" Rodriguez Institute of Science and Technology (EARIST). He is very interested in technology, computers, and learning how software works. As a freshman, he is currently focusing on his basic engineering courses, mathematics, and programming logic. Aside from his college studies, he also enjoys creative digital work and content creation. Joel hopes to use what he learns at EARIST to build helpful tech projects, improve his skills in coding and computer systems, and become a successful computer engineer in the future.
9. **John Fred A. Bandoy** is a Computer Engineering student at the Eulogio "Amang" Rodriguez Institute of Science and Technology. He is passionate about technology and interested in learning more about computers and programming. He is a responsible and hardworking student who continues to improve his skills and knowledge in his studies. Despite challenges, he remains focused on achieving his goals and completing his education. He dreams of becoming a successful computer engineer in the future.
10. **Chzian Eric D. San** Juan is currently pursuing a degree in Computer Engineering at Eulogio "Amang" Rodriguez Institute of Science and Technology. He is trying his best to learn more about coding and other languange and willing to explore everything about coding, he's willing to learn and improve his knowledge about computer engineering fields, he's also passionate about coding, he's amaze how coding works and how different things work in fields of computer engineering, he's determined to finish this course and pursue this career to become a successful computer engineer.
11. **Kyla Mae Broncano** is a first-year college student at Eulogio "Amang" Rodriguez Institute of Science and Technology (EARIST). She is currently studying for her Bachelor of Science in Computer

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Engineering (BSCpE). She has already learned foundational skills in C++ programming and system logic. She is dedicated to building strong technical and engineering skills at EARIST to design and develop smart technologies in the future.

12. **Ma.Janelle T. Dongon** is currently pursuing a degree in Computer engineering at Eulogio "Amang" Rodriguez Institute of Science and Technology. She is dedicated to achieving academic excellence and continuously improving her knowledge and skills in the field of technology and innovation. Known for being hardworking, responsible, and determined,she always strives to overcome challenges and reach personal and professional goals. She enjoys learning new things, exploring ideas, and developing abilities that can contribute to future success. Through perseverance, discipline, and continuous self-improvement, she aspires to become successful in her chosen career in the future.
13. **Engr. Meshelle N. Fabro** is a licensed Professional Computer Engineer and educator at the Eulogio “Amang” Rodriguez Institute of Science and Technology (EARIST). She is currently pursuing a Master of Science in Computer Engineering at Bulacan State University and has professional industry experience as a Technical Engineer at Hewlett-Packard (HP) and an IT Engineer at IBM.She is also a TESDA CSS NC II and Trainers Methodology Level I (TM1) holder, demonstrating her commitment to technical excellence and competency-based education. Passionate about teaching and innovation, she is dedicated to guiding Computer Engineering students in becoming industry-ready, competent, and highly skilled individuals equipped for the evolving demands of the technology industry.