HOSPITAL MANAGEMENT SYSTEM

MICRO PROJECT REPORT

Object Oriented Techniques ITT-202

Submitted by

JACOB SAM JOSE
(AJC22IT039)
&
DIVIYA WILSON
(AJC22IT030)

BACHELOR OF TECHNOLOGY
in

INFORMATION TECHNOLOGY

AMAL JYOTHI COLLEGE OF ENGINEERING
KANJIRAPPALLY

MAY 2024

DEPARTMENT OF INFORMATION TECHNOLOGY

AMAL JYOTHI COLLEGE OF ENGINEERING, KANJIRAPALLY

CERTIFICATE

This is to certify that the report entitled "**Hospital Management System**" submitted by **JACOB SAM JOSE(AJC22IT039) & DIVIYA WILSON(AJC22IT030)** to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the Degree of Bachelor of Technology in Information Technology, is a bonafide record of the Micro Project work carried out by him under our guidance and supervision during the academic year 2023 - 2024. This report in any form has not been submitted to any other University or Institute for any purpose.

Thomas Varghese
Associate Professor

Project Guide

ABSTRACT

This report introduces the development of a Hospital Management System (HMS), aimed at enhancing administrative and patient-centric operations within healthcare facilities. Implemented using Java Swing, the system boasts a user-friendly interface facilitating login options for both administrators and patients.

Upon admin login, a dedicated dashboard offers functionalities including staff management, appointment scheduling, and staff availability tracking. Patients, upon login, gain access to a personalized portal for updating information and managing appointments. The system integrates a MySQL database for robust data storage and retrieval, with meticulous error handling ensuring data integrity.

Workflow and data flow diagrams depict the system's sequential operations and data interactions. The HMS adheres to coding best practices, fulfilling project requirements for creating, editing, deleting, and viewing appointments. Its development serves as a foundational platform poised for further customization and enhancement based on user feedback, promising efficient hospital operations and improved patient care.

CONTENTS

LIST OF FIGURES

INTRODUCTION

The aim of this project is to develop a comprehensive Hospital Management System (HMS) that offers intuitive functionality for efficient management of hospital operations. In response to the growing complexities of healthcare administration, our platform seeks to empower healthcare professionals and administrators in effectively overseeing various aspects of hospital management.

Users of the HMS will have access to a user-friendly interface designed to streamline administrative tasks, ensuring smooth operations within the healthcare facility. The system allows for distinct user roles, including administrators and patients, each with tailored functionalities to address their specific needs and responsibilities.

Administrators can seamlessly navigate through the system, with options to manage staff, appointments, and overall hospital resources. Key features include the ability to add, remove, or update staff information, schedule appointments, and monitor staff availability. This facilitates optimal workforce management and ensures efficient allocation of resources.

Patients, on the other hand, benefit from a personalized portal offering convenient access to their medical records, appointment schedules, and communication with healthcare providers. They can easily update their personal information, schedule appointments, and access essential healthcare services, thereby enhancing patient engagement and satisfaction.

The HMS leverages robust database integration to securely store and retrieve critical healthcare data, ensuring data integrity and confidentiality. Throughout the development process, emphasis is placed on adherence to coding best practices and the implementation of robust security measures to safeguard sensitive patient information.

In this report, we will delve into the design, development, and implementation of the Hospital Management System, outlining its underlying architecture, user interface components, and database integration. Our overarching goal is to provide healthcare professionals and administrators with a powerful tool for effectively managing hospital operations, ultimately improving patient care delivery and enhancing overall healthcare outcomes.

METHODOLOGY

2.1 Proposed System

The proposed system for the Hospital Management System (HMS) project aims to develop a robust and efficient application that utilizes Java Database Connectivity (JDBC) technology to interact with relational databases. This system will serve as a comprehensive solution for managing various aspects of hospital operations while ensuring data integrity, security, and performance**.**

The major objectives of the proposed system are as follows:

1. **Provide a User-Friendly Interface:** The system will offer an intuitive and user-friendly interface tailored to the needs of hospital administrators, staff, and patients. It will facilitate seamless navigation and execution of tasks related to hospital management.
2. **Support Multiple Database Systems:** The system will be designed to seamlessly integrate with various popular relational database management systems commonly used in hospital environments, ensuring compatibility and flexibility.
3. **Ensure Data Integrity:** The system will implement robust mechanisms to validate and enforce data integrity constraints, safeguarding the accuracy and consistency of hospital data across different modules and operations.
4. **Optimize Performance:** Employing performance optimization techniques, the system will enhance the efficiency and speed of database operations, ensuring swift retrieval and manipulation of hospital data to support real-time decision-making.
5. **Enable Error Handling and Logging:** The system will incorporate comprehensive error handling mechanisms to gracefully manage exceptions and unexpected scenarios. Additionally, it will implement logging functionality to track system activities and facilitate troubleshooting.

The proposed HMS aims to deliver a comprehensive solution that provides a user-friendly interface, supports integration with multiple database systems, ensures data integrity and security, optimizes performance, and incorporates robust error handling and logging mechanisms.

2.1.1MODULE 1: GUI

GUI (Graphical User Interface) is a visual interface that allows users to interact with a software application or system using graphical elements such as windows, buttons, menus, icons, and other graphical components. It provides a user-friendly and intuitive way for users to interact with complex functionalities and perform tasks efficiently.

GUI revolutionized the way users interact with computers by replacing the command-line interfaces with visually appealing and easy-to-use graphical elements GUI simplifies user interaction and does not require extensive technical expertise. GUI offers advantages such as visual representation, interactivity, multitasking, and error prevention.

Key design principles for GUI include consistency, simplicity, responsiveness, visual hierarchy, navigation, error handling, and user feedback. In conclusion, GUI has revolutionized user interaction with software applications by providing a visual and intuitive interface. Its significance lies in its user-friendly nature, intuitive design, enhanced productivity, and accessibility features.

GUI offers advantages such as visual representation, interactivity, multi-tasking, and error prevention. Adhering to key design principles, such as consistency, simplicity, responsiveness, visual hierarchy, navigation, error handling, and user feedback, is essential for creating effective and user-friendly GUIs

2.1.2Module 2: Database Connectivity

Database Connectivity is a crucial component in software applications that enables the interaction between the application and a relational database management system (RDBMS). It allows applications to retrieve, store, update, and manage data stored in databases. This module focuses on understanding the significance of database connectivity, its advantages, and key considerations for implementation.

Significance of Database Connectivity:
Seamless Data Management: Database connectivity facilitates the seamless integration of databases with software applications, enabling efficient data management.
Data Persistence: It allows data to be stored and retrieved from databases, ensuring the persistence and availability of critical information.
Scalability: Database connectivity supports scalability by accommodating a growing amount of data and concurrent user requests.
Data Integrity: By leveraging database connectivity, applications can enforce data integrity constraints such as referential integrity, data types, and uniqueness, ensuring the consistency and reliability of data.

Database connectivity is essential for software applications to interact with relational databases. It enables efficient data management, persistence, scalability, and data integrity. The advantages of database connectivity include efficient data access, data security, data consistency, and querying capabilities. Considerations for implementation include selecting appropriate database drivers, connection management, error handling, transaction management, performance optimization, and security measures. By implementing robust database connectivity, applications can effectively leverage the power of databases to store, retrieve, and manage data efficiently.

2.1.3 Module 3: Operations

Operations in the context of a software application refer to the various tasks and functionalities performed by the system to meet user requirements. This module focuses on understanding the different types of operations, their significance, and key considerations for their implementation.

Types of Operations:
a. Data Retrieval: Operations that involve fetching data from databases based on specific criteria or queries.
b. Data Manipulation: Operations that involve modifying, inserting, or deleting data in the database.
c. Data Validation: Operations that ensure the integrity and consistency of data by validating inputs against predefined rules and constraints.
d. Data Aggregation: Operations that combine data from multiple sources or records to generate summaries or reports.
e. Data Analysis: Operations that involve performing calculations, statistical analysis, or data mining to derive insights from the data.
f. Data Presentation: Operations that involve formatting and presenting data to users through reports, charts, graphs, or visual representations.

Operations form the core functionalities of a software application, ranging from data retrieval and manipulation to data analysis and presentation. They play a significant role in meeting functional requirements, ensuring data integrity, providing a good user experience, and supporting decision-making processes. Implementing operations requires considerations such as efficiency, error handling, security, scalability, compatibility, testing, and documentation. By effectively implementing operations, an application can perform its intended tasks efficiently and provide value to its users.

2.2 WORK FLOW SYSTEM

1. User Authentication and Login:
    - Users access the HMS system through a login page.
    - Upon successful authentication, users are directed to their respective dashboards based on their roles (admin/staff or patient).
2. Admin Dashboard:
    - Admins have access to functionalities for managing staff, appointments, and available staff.
    - Admins can choose from options like "Edit Staffs," "Appointments," and "Available Staffs."
3. Edit Staffs:
    - Admins can add, remove, or display current staff members.

4. Appointments:
    - Admins can view, add, or remove appointments.

5. Available Staffs:
    - Admins can view the list of doctors currently available.
    - They can mark leave for any staff member and view staff currently on leave.

6. Patient Dashboard:
    - Patients can choose from options like "Edit patient Information" and “Appointments”.
7. Edit Patient Information:
    - Patients can update their personal details such as name, contact information, etc.
8. Appointments (Patient):
    - Patients can add new appointments, or edit existing ones.
9. Logout:
    - Both admins and patients have the option to securely logout from their accounts.

SYSTEM IMPLEMENTATION & TESTING

Implementation of Module 1 (GUI)

The Testing module in the Hospital Management System (HMS) is responsible for ensuring the reliability and functionality of the system's graphical user interface (GUI). This involves testing the various features such as creating, editing, deleting, and viewing patient records, appointments, and staff information. The testing process encompasses both manual and automated testing methods to validate the accuracy and responsiveness of the GUI across different scenarios and user interactions.

Java Swing Framework:
The GUI implementation in the HMS utilizes the Java Swing framework. Java Swing offers a robust toolkit for building interactive and platform-independent GUI applications. It provides a wide array of components, layout managers, and event handling mechanisms, enabling developers to create intuitive and visually appealing user interfaces for hospital administrators, staff, and patients.

Design Considerations:
In designing the GUI for the Hospital Management System, several key considerations are taken into account:
a. User-Centric Layout: The interface layout prioritizes ease of use and intuitive navigation, ensuring that common tasks such as managing patient records, appointments, and staff information are easily accessible.
b. Responsive Design: The GUI is designed to adapt seamlessly to various screen sizes and resolutions, providing a consistent and optimized user experience across different devices and platforms.
c. Professional Appearance: The interface incorporates a professional and clean design aesthetic, utilizing appropriate colour schemes, typography, and icons to enhance visual appeal and usability.
d. Error Handling: Robust error handling mechanisms are integrated into the GUI to effectively communicate error messages and guide users in resolving issues related to input validation errors, system failures, or other unexpected scenarios.

GUI Components:

The graphical user interface (GUI) implementation in the Hospital Management System (HMS) utilizing Java Swing comprises the following components:

a. Main Frame: Serving as the central window of the application, the main frame features the system title, logo, and a menu bar or toolbar for convenient access to various functionalities such as managing staff, appointments, and patient records.

b. Input Forms: Input forms are provided for capturing essential information such as patient details, appointment schedules, and staff records. These forms incorporate relevant labels, text fields, and validation mechanisms to ensure accurate data entry and maintain data integrity.

c. Buttons and Menus: Strategically placed buttons and menus within the interface facilitate users in executing actions related to hospital management tasks. These components are associated with event handlers to trigger functionalities such as adding, editing, or deleting patient records, appointments, and staff information.

d. Record Viewer: The GUI includes components for viewing patient records, appointment schedules, and staff details in a visually organized manner. Each record is displayed with pertinent information, enabling users to navigate through records efficiently and interact with them as necessary.

Technologies Used:

a. Java Swing Framework: Leveraging the Java Swing framework provides the foundational infrastructure for creating the user interface, managing GUI components, and handling user interactions within the HMS application.

b. Java Programming Language: The implementation utilizes Java as the primary programming language for developing the GUI functionalities, ensuring compatibility, reliability, and robustness in managing hospital operations effectively.

Implementation of Module 2 (Database Connectivity)

The Database Connectivity module in the Hospital Management System (HMS) is tasked with facilitating seamless interaction with the database, ensuring efficient storage, retrieval, and management of hospital-related data while upholding data integrity.

Database Management System: The Database Connectivity in the Hospital Management System project utilizes a database management system (DBMS) to store and manage the hospital data. The choice of the DBMS depends on the specific requirements of the project, such as data structure, scalability, and performance. Commonly used DBMS options include MySQL, Oracle, or SQLite. We Have Used MySQL here.

Key Features:

1. **Connection Establishment:** The module is responsible for establishing a secure and reliable connection with the database server, enabling seamless communication for data operations.
2. **Data Storage and Retrieval:** It facilitates the storage and retrieval of various hospital-related data, including patient records, appointment schedules, staff information, and administrative data.
3. **Data Integrity Management:** The module implements mechanisms to enforce data integrity constraints, ensuring the accuracy, consistency, and reliability of data stored in the database.

Technologies Used:

1. **Java Database Connectivity (JDBC):** JDBC is utilized to establish a connection with the database server and execute SQL queries to perform database operations seamlessly within the HMS application.

2. **Relational Database Management System (RDBMS):** The module interacts with a relational database management system (e.g., MySQL, PostgreSQL) to store and retrieve hospital-related data efficiently, leveraging the structured query language (SQL) for data manipulation.

3. **Data Access Objects (DAOs):** DAO design pattern is employed to encapsulate database access logic, promoting modularity, and facilitating maintainability of the database connectivity module.

The Database Connectivity module in the Hospital Management System plays a critical role in ensuring the seamless integration of hospital data management functionalities. By adhering to design considerations and leveraging relevant technologies, the module contributes to the overall efficiency, reliability, and security of the HMS application. Continuous refinement and optimization of the database connectivity module are essential to meet evolving requirements and ensure optimal performance in managing hospital operations.

Implementation of Module 3 (Operations):

The Operations module in the Hospital Management System (HMS) is dedicated to executing various essential operations related to hospital management, ensuring seamless functionality and efficient handling of hospital-related tasks. This module encompasses key features, design considerations, and the utilization of pertinent technologies to enable comprehensive operations within the HMS application.

Functionality Overview:
The Operations module in the HMS project offers the following functionalities:

a. **Create New Records:** Hospital administrators can create new records for patients, staff members, and appointments by providing relevant information such as patient details, staff credentials, and appointment schedules. Each record is assigned a unique identifier to facilitate easy retrieval and management.
b. **Update Records:** Administrators have the capability to update existing records, including patient information, staff details, and appointment schedules. This feature allows for the modification of data to reflect changes in patient status, staff credentials, or appointment timings.
c. **Delete Records:** Administrators are empowered to delete specific records from the system, such as outdated patient records, staff profiles, or cancelled appointments. This functionality ensures the efficient management of data and helps maintain a clean and organized database.
d. **Retrieve Records:** The module provides the ability to retrieve records based on specified criteria, such as searching by patient name, staff ID, or appointment date. This functionality facilitates quick and accurate access to relevant information, enhancing workflow efficiency and decision-making processes.

Design Considerations:

During the implementation of the Operations module in the Hospital Management System, the following design considerations were prioritized:

a. Modular and Extensible Design: The module is crafted with a modular and extensible architecture to facilitate seamless integration of future enhancements and additional functionalities. This ensures scalability and adaptability to evolving requirements without disrupting the existing codebase.

b. Robust Error Handling: Comprehensive error handling mechanisms are implemented to gracefully manage exceptions and provide informative error messages to users. This promotes a smooth user experience and facilitates efficient debugging and troubleshooting processes.

c. Data Validation: Rigorous input validation is enforced to maintain the integrity and security of the hospital data. All user inputs undergo thorough validation to mitigate potential risks such as SQL injection attacks or invalid data entries, ensuring the reliability and consistency of the database.

Technologies and Libraries Used:

a. Java Programming Language: The module is developed using the Java programming language, offering a robust and platform-independent environment conducive to building scalable and reliable software solutions.

b. Java Database Connectivity (JDBC): JDBC is employed to establish a seamless connection with the underlying database system and execute SQL statements for performing a variety of database operations, including data manipulation and retrieval.

c. Database Connectivity Module: The Operations module seamlessly integrates with the Database Connectivity module, facilitating the establishment of a secure and efficient connection with the database server and enabling streamlined data-related operations within the HMS application.

The successful implementation of the Operations module in the Hospital Management System empowers administrators and staff to perform essential operations seamlessly. By offering functionalities such as creating, updating, deleting, and retrieving patient records, appointments, and staff module enhances the efficiency and effectiveness of hospital management processes.

System Integration:

System integration in the Hospital Management System (HMS) refers to the process of harmoniously combining various modules, components, and subsystems to create a cohesive and functional system that fulfils the project requirements. The primary objective of system integration is to establish seamless interactions among different components, ensuring smooth operation and efficient data flow throughout the HMS application.

Integration Activities:
The system integration process for the Hospital Management System encompassed the following key activities:

a. Integration Planning: A comprehensive integration plan was devised to outline the approach for integrating different modules and components of the HMS. This plan delineated the sequence of integration, identified dependencies, and specified interfaces between various subsystems.

b. Component Integration: Individual modules such as GUI, Database Connectivity, and Operations were integrated meticulously to ensure their seamless interaction and cohesive operation. This involved linking the GUI module with the Database Connectivity module and integrating it with the Operations module to facilitate essential data operations.

c. Data Flow Integration: Efforts were made to establish smooth data flow between different modules to facilitate efficient communication and exchange of information. For instance, user inputs captured through the GUI were transmitted to the Operations module for processing and subsequently persisted in the database via the Database Connectivity module.

d. Error Handling and Exception Management: Robust error handling mechanisms were implemented to effectively manage exceptions and error scenarios encountered during the integration process. This entailed the implementation of appropriate error messages and handling strategies to ensure graceful recovery from errors and maintain system stability.

e. Integration Testing: Rigorous testing procedures were employed to validate the integration of various modules and components within the HMS. This included comprehensive end-to-end testing to verify the functionality of the integrated system, ensure data consistency, and validate the effectiveness of error handling capabilities.

Challenges Encountered:

Throughout the system integration process, several challenges were encountered, including:

a. Interoperability Issues: Ensuring compatibility and seamless interaction among diverse modules and components posed significant challenges, particularly in integrating third-party systems or legacy software.

b. Data Consistency: Ensuring consistent data flow and integrity across different modules required meticulous attention to detail and thorough testing to identify and rectify inconsistencies.

c. Complex Dependencies: Managing complex dependencies between modules and components necessitated careful planning and coordination to mitigate potential conflicts and ensure smooth integration.

Overall Outcome:

Despite the challenges encountered, the system integration process culminated in the successful creation of a unified and functional Hospital Management System (HMS). The integrated system effectively facilitates essential hospital management operations, including patient record management, appointment scheduling, and staff administration, thereby enhancing operational efficiency and facilitating superior healthcare delivery. By seamlessly integrating diverse modules and components, the HMS provides a robust platform for managing hospital resources and delivering quality healthcare services.

Testing the System:

Testing is a critical phase in software development, ensuring the quality, reliability, and functionality of the system. This section provides an overview of the testing process conducted for the Hospital Management System (HMS), covering various types of testing, their objectives, and the overall outcome of the testing phase.

<u>Types of Testing:</u>
The testing process for the Hospital Management System involved the following types of testing:

a. **Unit Testing:** Unit testing focused on meticulously testing individual modules or components in isolation. It aimed to validate the correctness of functionalities implemented in each module, including the GUI, Database Connectivity, and Operations. Unit tests were designed to encompass all possible scenarios and edge cases, ensuring the seamless functioning of each module.
b. **Integration Testing:** Integration testing was conducted to validate the interaction between different modules upon integration. This testing phase aimed to ensure the seamless communication and data exchange between modules, verifying that they produced the expected results when combined. Integration tests extensively covered scenarios involving interaction between the GUI, Database Connectivity, and Operations modules, ensuring smooth data flow between them.
c. **System Testing:** System testing was performed to evaluate the system as a cohesive entity, comprising all integrated modules. Its objective was to validate the system against both functional and non-functional requirements, ensuring adherence to desired specifications. System tests encompassed end-to-end scenarios, simulating real-world usage of the Hospital Management System and assessing its overall performance and functionality.
d. **User Acceptance Testing (UAT):** UAT involved engaging end users or stakeholders to evaluate the system from their perspective. The primary aim was to ensure that the HMS met their expectations, was user-friendly, and fulfilled their specific requirements. UAT provided valuable insights and feedback, guiding any necessary refinements or enhancements before the system's final deployment.

Testing Objectives:

The testing phase for the Hospital Management System (HMS) had the following objectives:

a. Functional Testing: Validate that the system functions in accordance with the specified requirements. This includes verifying essential operations such as patient record management, appointment scheduling, and staff administration.

b. Usability Testing: Evaluate the user-friendliness of the system's interface, ensuring it is intuitive, easy to navigate, and provides a pleasant user experience. This involves assessing the GUI's responsiveness, layout, and overall usability to enhance user interaction.

c. Performance Testing: Assess the system's performance under various conditions, including high volumes of concurrent users or large datasets. This testing aims to identify performance bottlenecks, optimize system resources, and ensure efficient response times to maintain system scalability.

d. Security Testing: Verify the system's security measures, including authentication, authorization, and data protection. This involves testing for vulnerabilities, potential breaches, and adherence to security best practices to safeguard sensitive hospital data.

Outcome:

The testing phase for the Hospital Management System yielded the following outcomes:

a. Defect Identification and Resolution: Through rigorous testing, defects, bugs, or issues were identified and addressed by the development team. These were logged, prioritized, and resolved, ensuring a stable and reliable system.

b. Functional Validation: The system's functionalities were successfully validated, ensuring seamless operations such as patient data management, appointment scheduling, and staff administration. The system met functional requirements and provided the intended features.

c. Usability and Performance Validation: Usability testing improved the GUI's user experience, while performance testing verified the system's ability to handle expected loads and maintain acceptable response times.

d. User Acceptance: Feedback from end users or stakeholders was solicited through user acceptance testing. Their input was considered, and necessary adjustments were made to align the system with their expectations, ensuring user satisfaction and acceptance

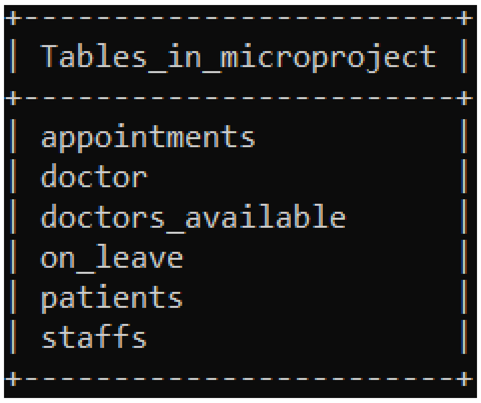
Conclusion:

The testing phase played a pivotal role in ensuring the quality, reliability, and functionality of the Hospital Management System. Through comprehensive testing methodologies, defects were addressed, and necessary improvements were implemented, leading to a robust and reliable HMS. The testing process instilled confidence in the system's stability, performance, and usability, ultimately resulting in an effective hospital management solution.

RESULTS

The HMS Platform project successfully passed all testing phases, demonstrating its functionality, usability, performance, security, and user acceptance. The system fulfils its purpose of providing a user-friendly interface for both the admins and patients to manage, and interact with platform effectively. Based on the testing results, the HMS Platform is ready for deployment, allowing individuals and organizations to express their thoughts, share knowledge, and engage with a global audience through the platform's intuitive interface and robust functionalities.

OUTPUT



```
+------------------------+
| Tables_in_microproject |
+------------------------+
| appointments           |
| doctor                 |
| doctors_available      |
| on_leave               |
| patients               |
| staffs                 |
+------------------------+
```

Fig 1: Tables used for the microproject in microproject database.

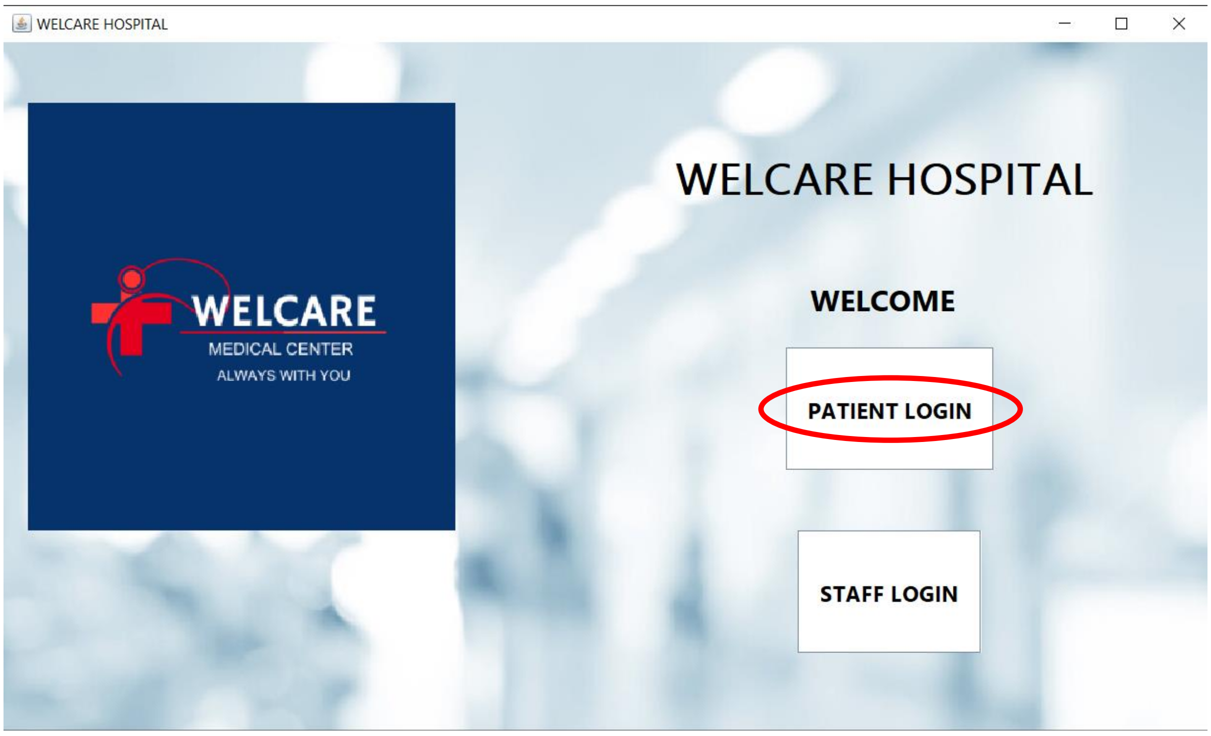
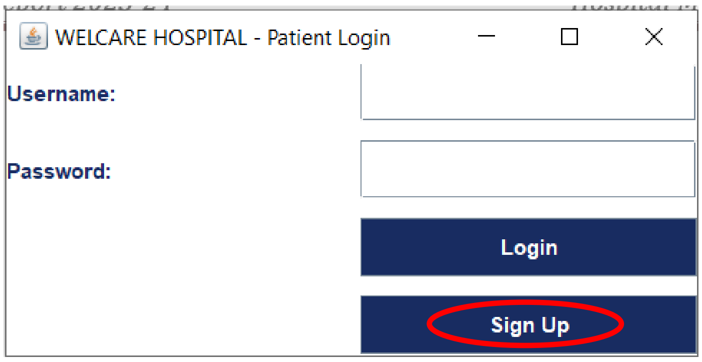
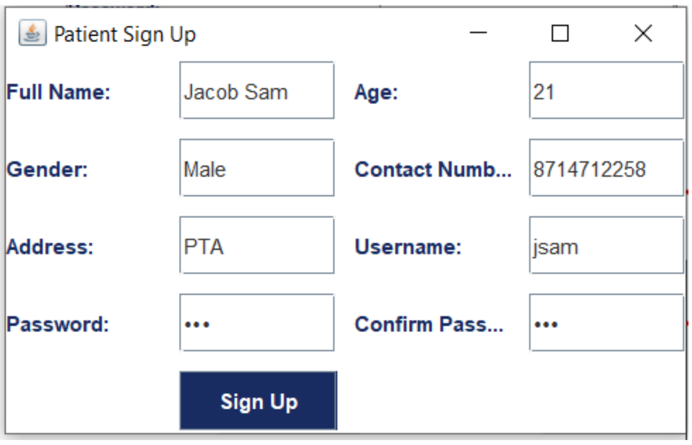


Fig 2: User Interface (Main Page)

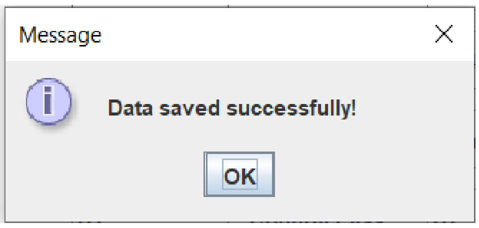
PATIENT LOGIN PLATFORM


WELCARE HOSPITAL - Patient Login

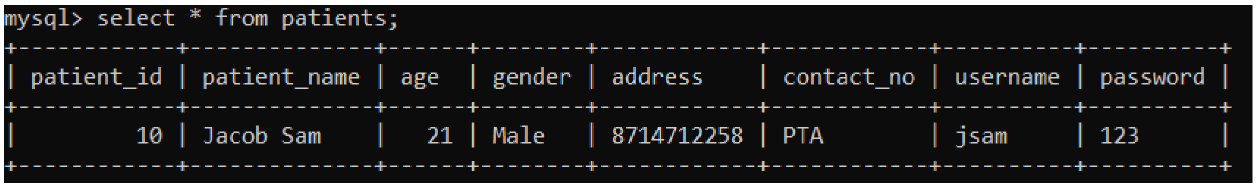
Username:	
Password:	


Patient Sign Up

Full Name:	Jacob Sam	Age:	21
Gender:	Male	Contact Numb...	8714712258
Address:	PTA	Username:	jsam
Password:	•••	Confirm Pass...	•••

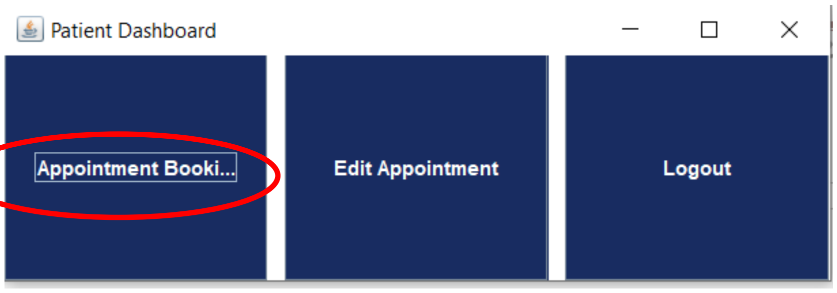

Message

Data saved successfully!

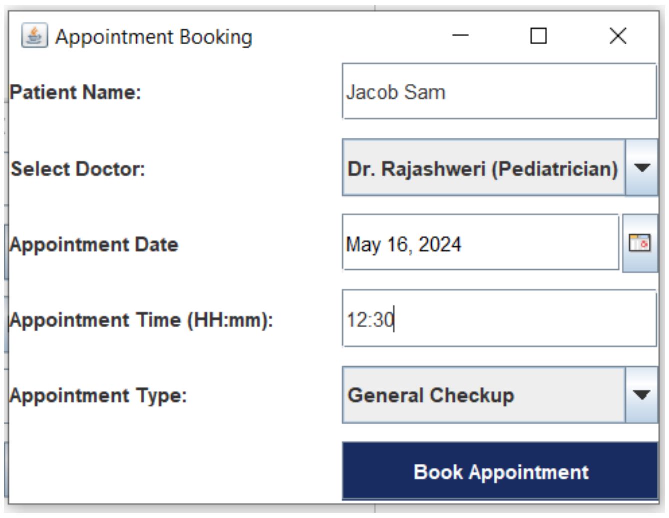

```
mysql> select * from patients;
+------------+--------------+------+--------+------------+------------+----------+----------+
| patient_id | patient_name | age  | gender | address    | contact_no | username | password |
+------------+--------------+------+--------+------------+------------+----------+----------+
|         10 | Jacob Sam    |   21 | Male   | 8714712258 | PTA        | jsam     | 123      |
+------------+--------------+------+--------+------------+------------+----------+----------+
```

All the above figures show a new patient signing up for a new account, the patient details get stored into the 'patients' table.

3.1)



3.2)



Patient Name: Jacob Sam

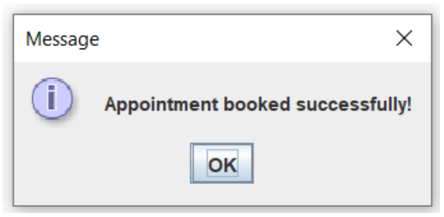
Select Doctor: Dr. Rajashweri (Pediatrician)

Appointment Date May 16, 2024

Appointment Time (HH:mm): 12:30

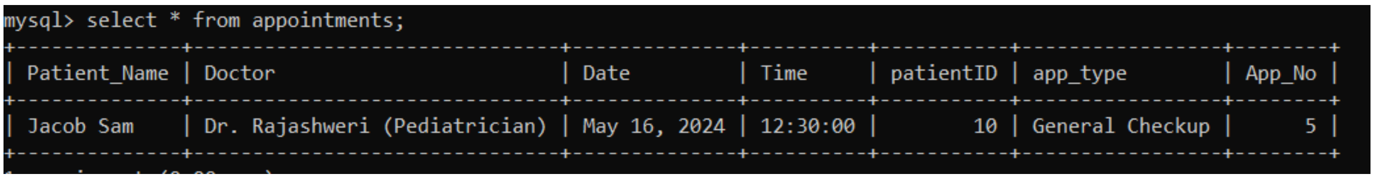
Appointment Type: General Checkup

3.3)



Appointment booked successfully!

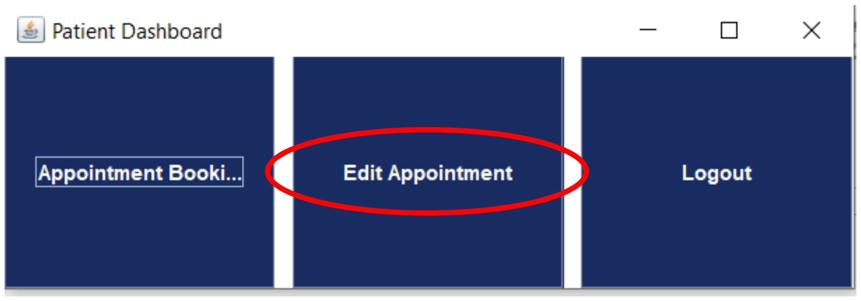
3.4



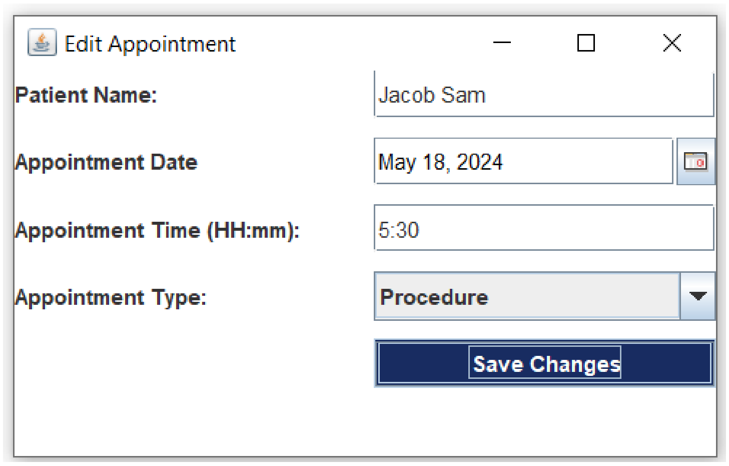
```
mysql> select * from appointments;
+--------------+-------------------------------+-------------+----------+-----------+----------------+--------+
| Patient_Name | Doctor                        | Date        | Time     | patientID | app_type       | App_No |
+--------------+-------------------------------+-------------+----------+-----------+----------------+--------+
| Jacob Sam    | Dr. Rajashweri (Pediatrician) | May 16, 2024| 12:30:00 |        10 | General Checkup|      5 |
+--------------+-------------------------------+-------------+----------+-----------+----------------+--------+
```

Fig 3.1: Patient Dashboard after signing up, proceeds to book an appointment.
Fig 3.2: Filling up Appointment Details
Fig 3.4: Appointment Details stored in 'appointments table.

4.1)



4.2)



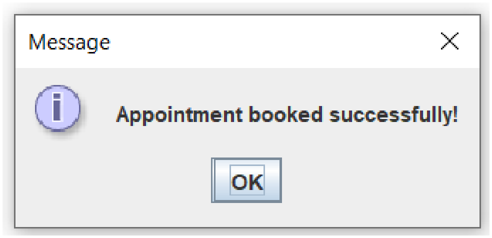
Patient Name: Jacob Sam

Appointment Date: May 18, 2024

Appointment Time (HH:mm): 5:30

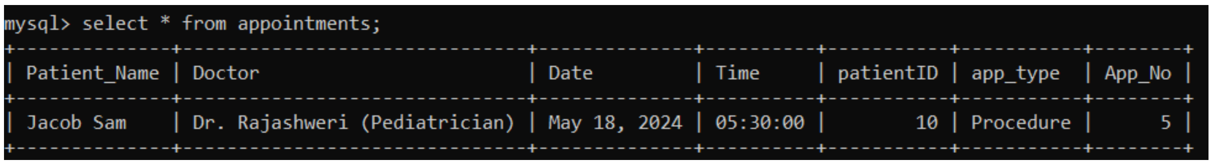
Appointment Type: Procedure

4.3)



Appointment booked successfully!

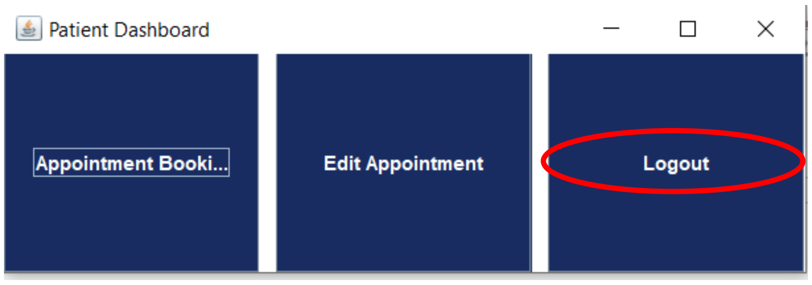
4.4)



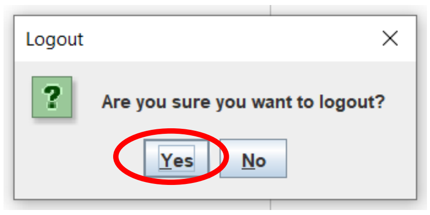
```
mysql> select * from appointments;
+--------------+-------------------------------+--------------+----------+-----------+-----------+--------+
| Patient_Name | Doctor                        | Date         | Time     | patientID | app_type  | App_No |
+--------------+-------------------------------+--------------+----------+-----------+-----------+--------+
| Jacob Sam    | Dr. Rajashweri (Pediatrician) | May 18, 2024 | 05:30:00 |        10 | Procedure |      5 |
+--------------+-------------------------------+--------------+----------+-----------+-----------+--------+
```

Fig 4.1: Edit Appointment
Fig 4.2: Edit Appointment Details
Fig 4.4: Patient Details updated in 'appointments table.

5.1)

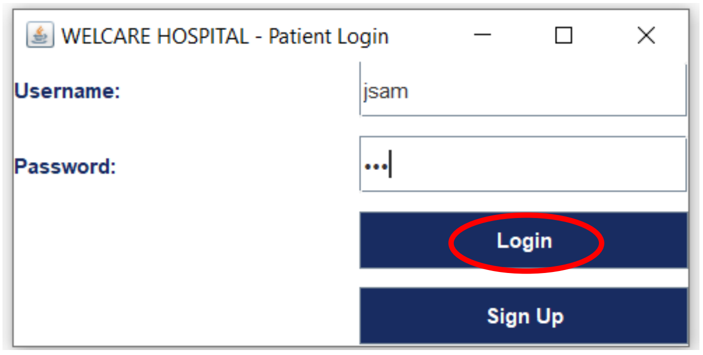


5.2)



Are you sure you want to logout?

5.3)



Username: jsam

Password: •••

5.4)

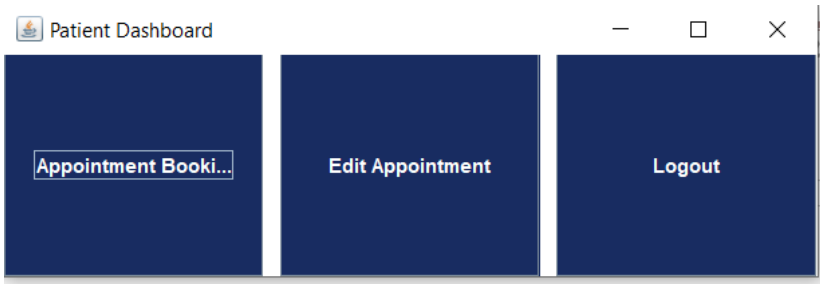
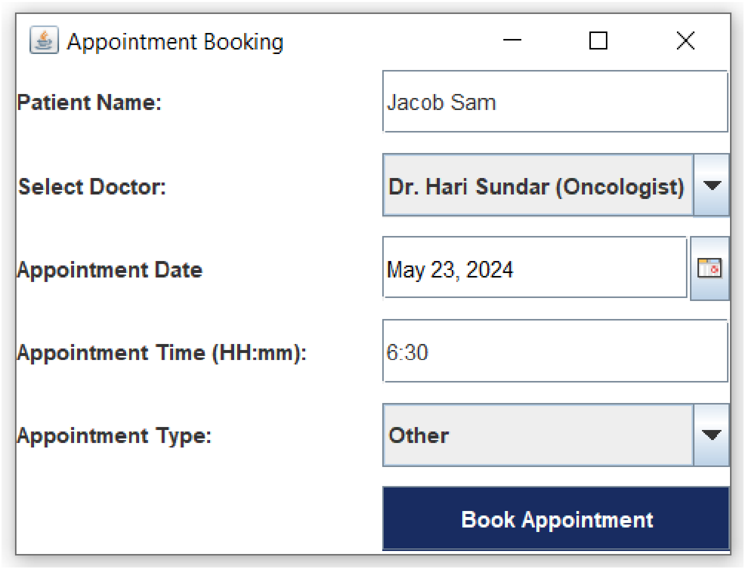


Fig 5.1: Log out
Fig 5.2: Log out confirmation
Fig 5.3: Log in for Existing Patients.
Fig 5.4: Redirects to the same window after sign up window.

The same patient with a unique ID can book one or more appointments that gets stored in the database.



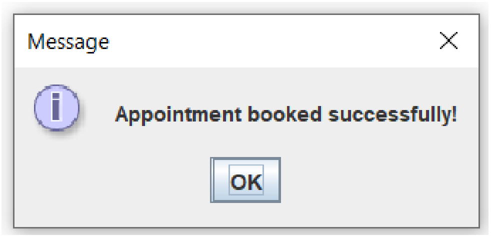
Patient Name: Jacob Sam

Select Doctor: Dr. Hari Sundar (Oncologist)

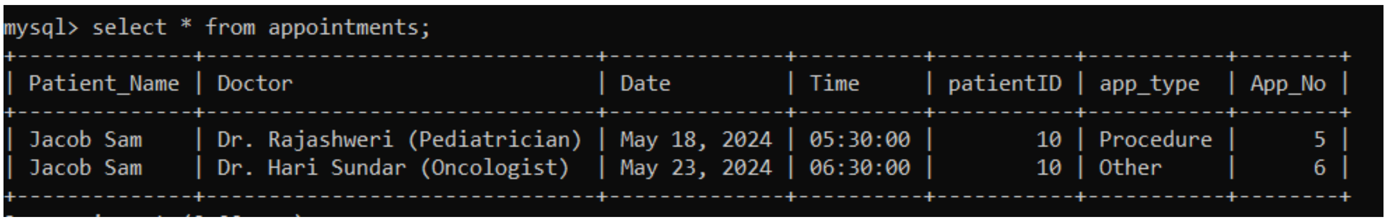
Appointment Date May 23, 2024

Appointment Time (HH:mm): 6:30

Appointment Type: Other



Appointment booked successfully!



```
mysql> select * from appointments;
+--------------+-------------------------------+--------------+----------+-----------+-----------+--------+
| Patient_Name | Doctor                        | Date         | Time     | patientID | app_type  | App_No |
+--------------+-------------------------------+--------------+----------+-----------+-----------+--------+
| Jacob Sam    | Dr. Rajashweri (Pediatrician) | May 18, 2024 | 05:30:00 |        10 | Procedure |      5 |
| Jacob Sam    | Dr. Hari Sundar (Oncologist)  | May 23, 2024 | 06:30:00 |        10 | Other     |      6 |
+--------------+-------------------------------+--------------+----------+-----------+-----------+--------+
```

STAFF LOGIN PLATFORM

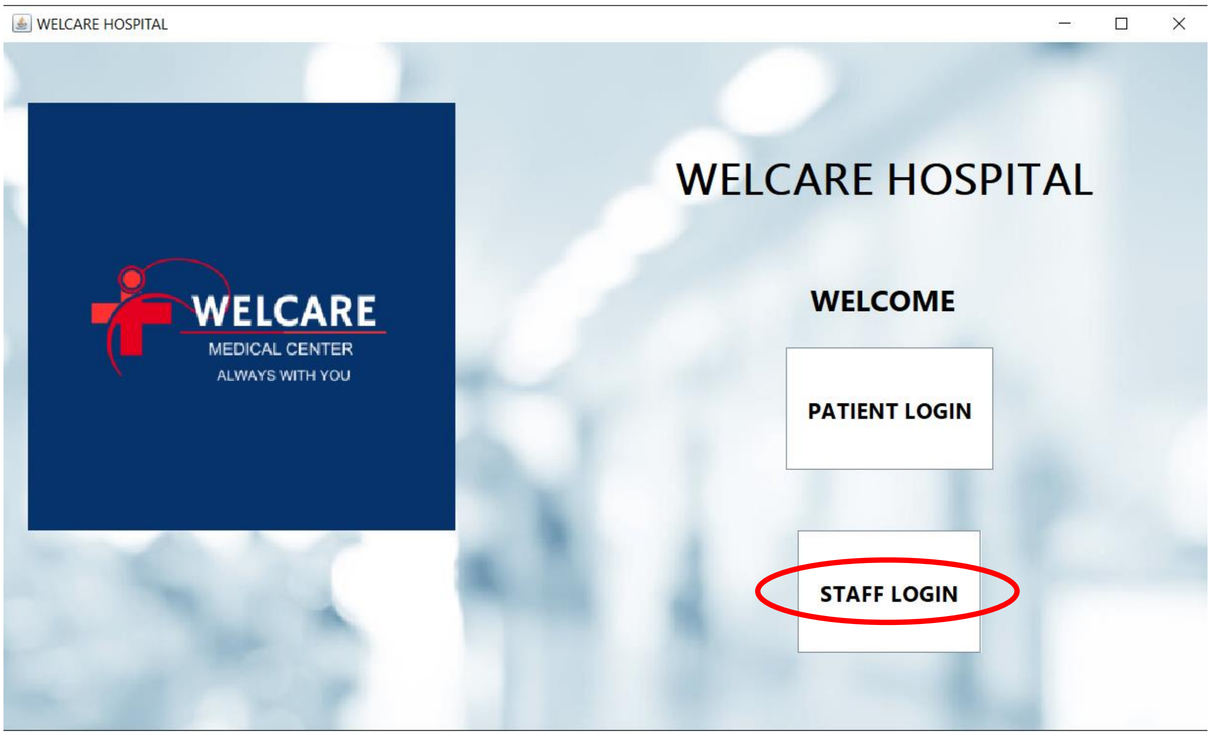
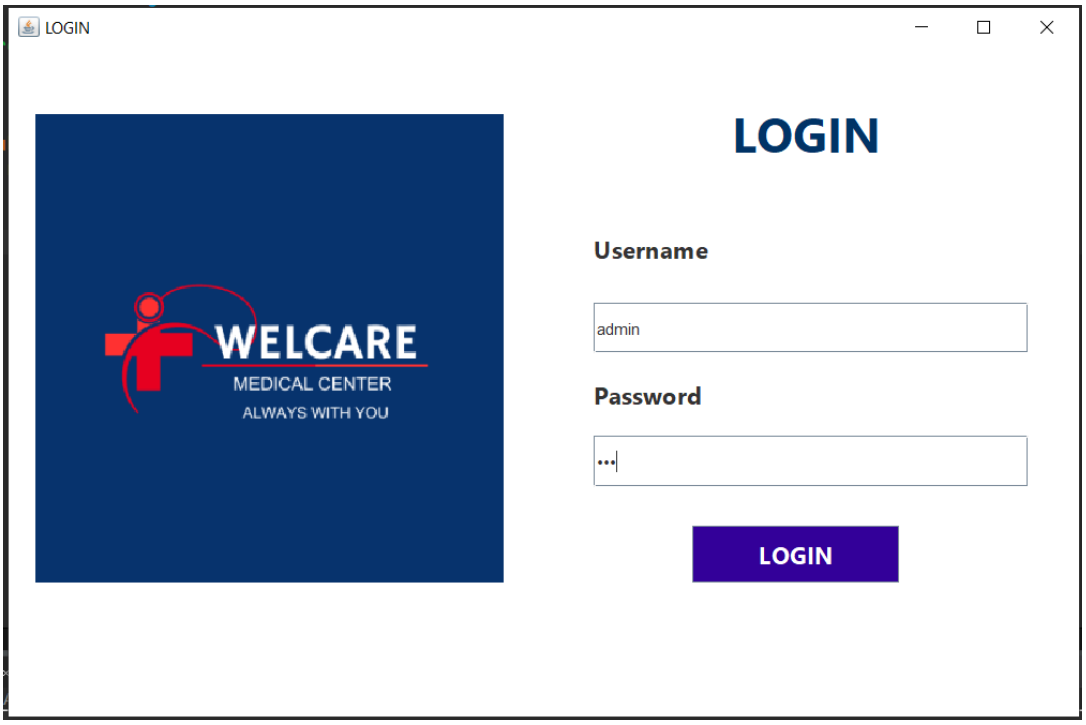


Fig 6.1

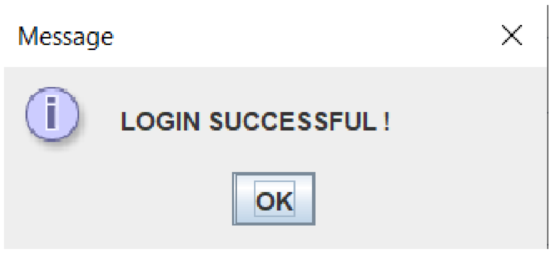


admin

•••

Fig 6.2

Fig 6.1: Staff/Admin Login Button
Fig 6.2: Admin Login Window


Message

LOGIN SUCCESSFUL !

Fig 6.3

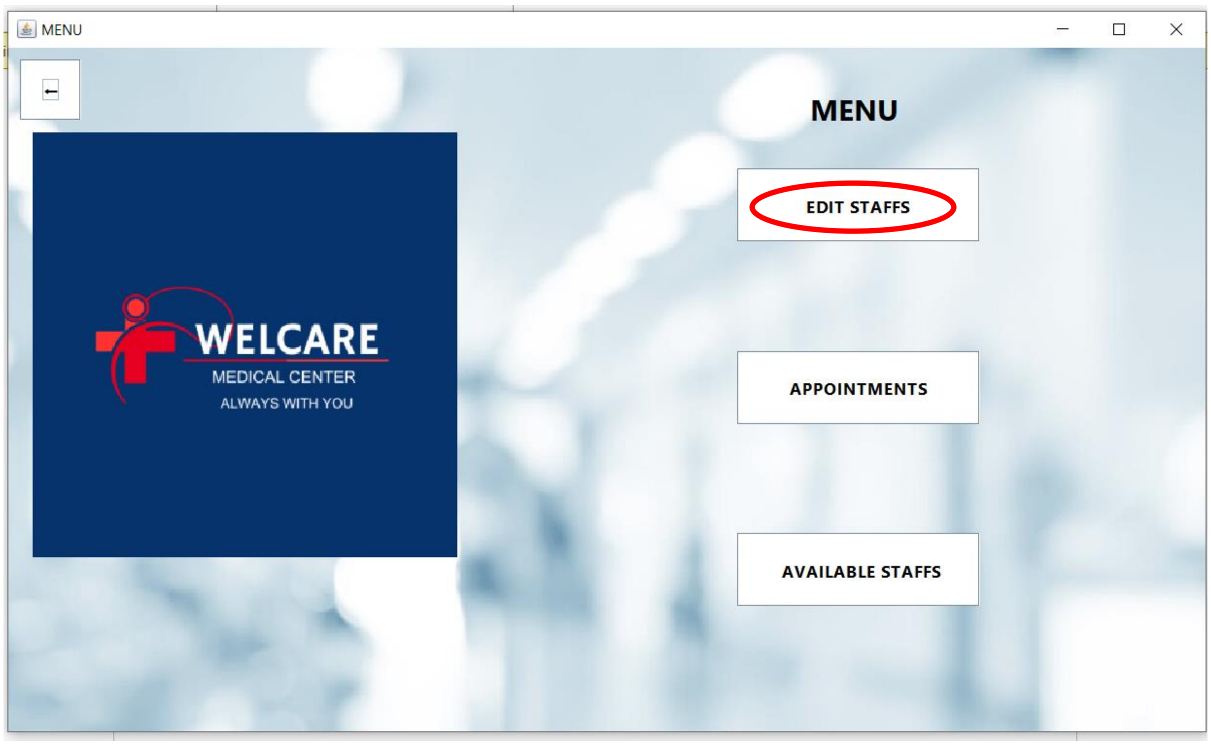


Fig7 .1

Fig 7.1: Menu with options such as
1) **EDIT STAFFS**
2) **APPOINTMENTS**
3) **AVAILABLE STAFFS**

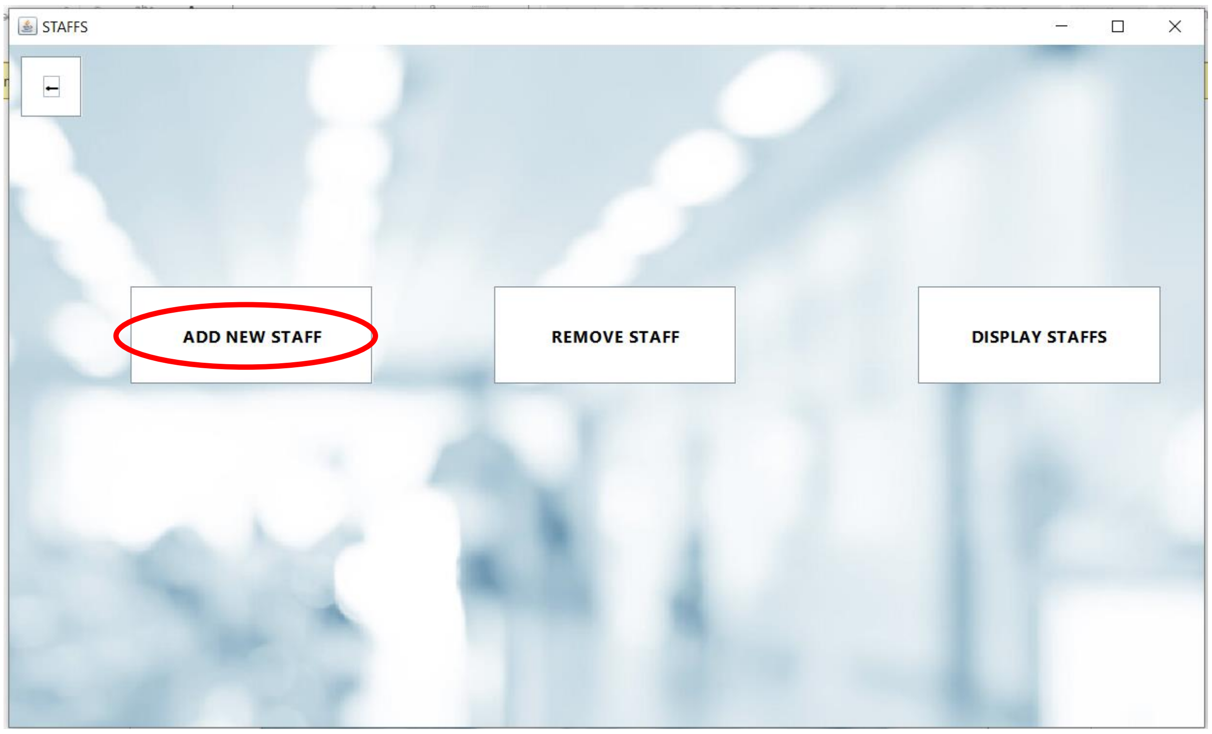
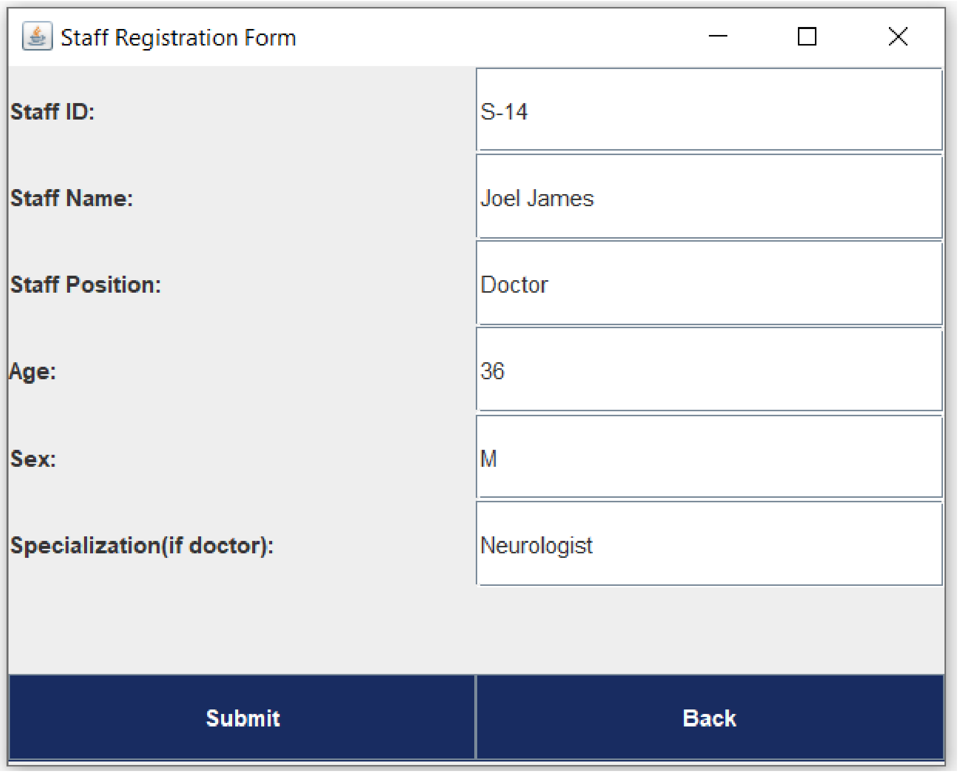


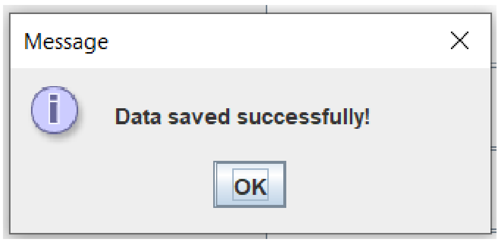
Fig 7.2



Staff ID:	S-14
Staff Name:	Joel James
Staff Position:	Doctor
Age:	36
Sex:	M
Specialization(if doctor):	Neurologist

Fig 7.3

Fig 7.2: Add New Staff option in Edit Staffs Window
Fig 7.3: New Staff Registration


Message

Data saved successfully!

Fig 7.4

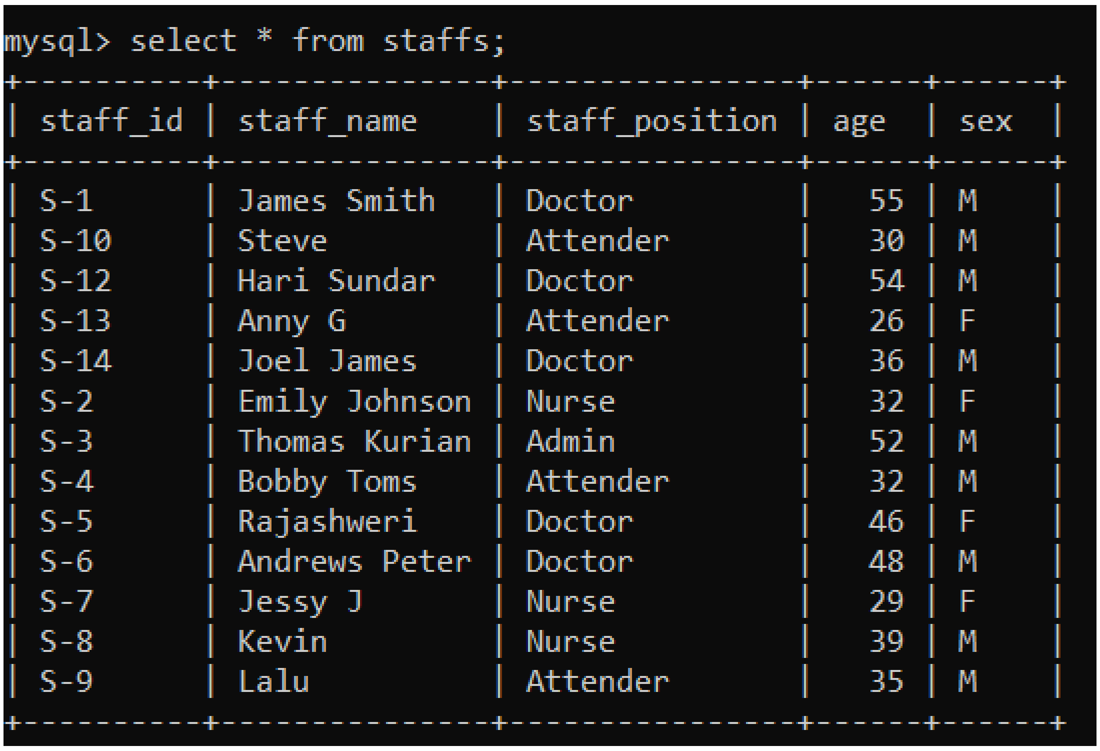

```
mysql> select * from staffs;
+----------+---------------+----------------+------+------+
| staff_id | staff_name    | staff_position | age  | sex  |
+----------+---------------+----------------+------+------+
| S-1      | James Smith   | Doctor         |   55 | M    |
| S-10     | Steve         | Attender       |   30 | M    |
| S-12     | Hari Sundar   | Doctor         |   54 | M    |
| S-13     | Anny G        | Attender       |   26 | F    |
| S-14     | Joel James    | Doctor         |   36 | M    |
| S-2      | Emily Johnson | Nurse          |   32 | F    |
| S-3      | Thomas Kurian | Admin          |   52 | M    |
| S-4      | Bobby Toms    | Attender       |   32 | M    |
| S-5      | Rajashweri    | Doctor         |   46 | F    |
| S-6      | Andrews Peter | Doctor         |   48 | M    |
| S-7      | Jessy J       | Nurse          |   29 | F    |
| S-8      | Kevin         | Nurse          |   39 | M    |
| S-9      | Lalu          | Attender       |   35 | M    |
+----------+---------------+----------------+------+------+
```

Fig 7.5

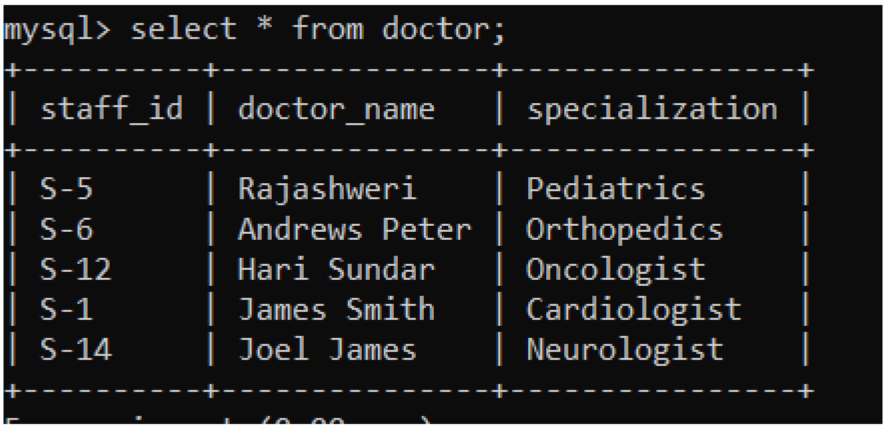

```
mysql> select * from doctor;
+----------+---------------+----------------+
| staff_id | doctor_name   | specialization |
+----------+---------------+----------------+
| S-5      | Rajashweri    | Pediatrics     |
| S-6      | Andrews Peter | Orthopedics    |
| S-12     | Hari Sundar   | Oncologist     |
| S-1      | James Smith   | Cardiologist   |
| S-14     | Joel James    | Neurologist    |
+----------+---------------+----------------+
[illegible]
```

Fig 7.6

Fig 7.5: New Staff gets added into 'staffs' table.
Fig 7.6: Since the new staff was a doctor, it also gets added into 'doctor' table.

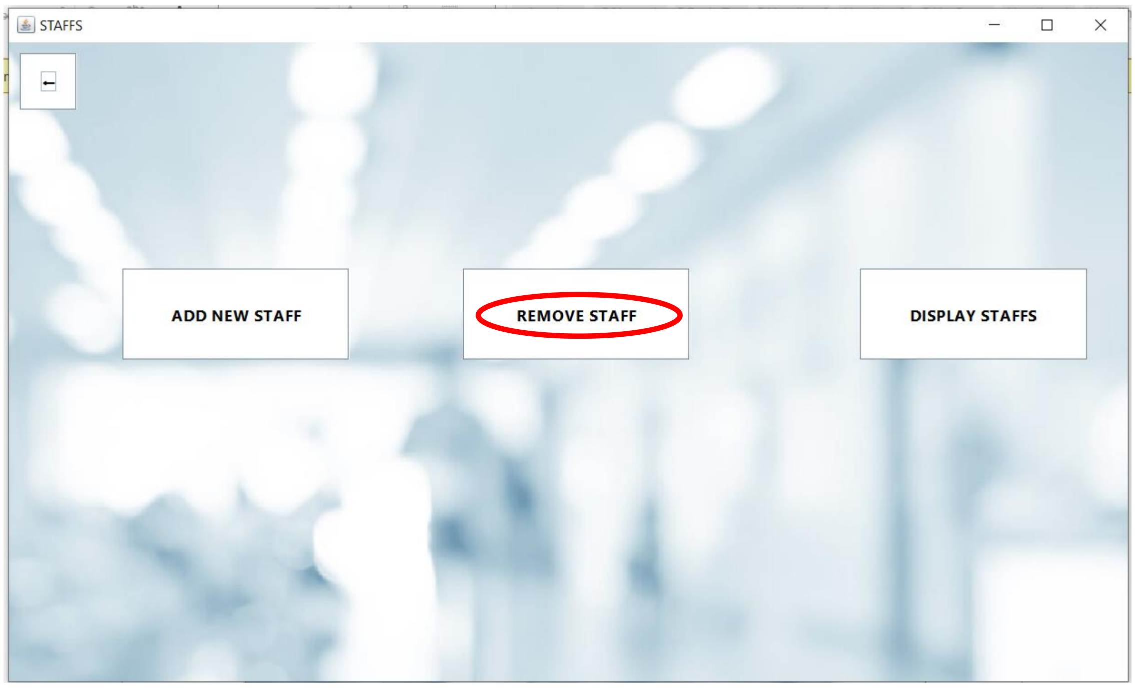
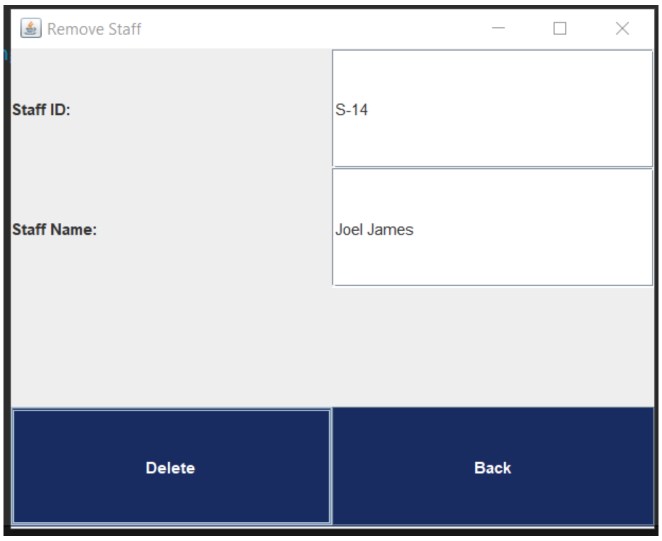


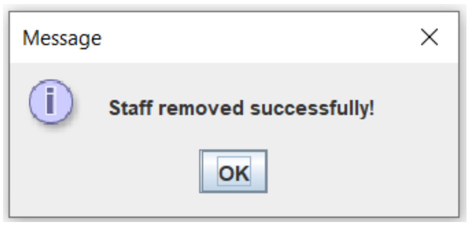
Fig 8.1



Staff ID: S-14

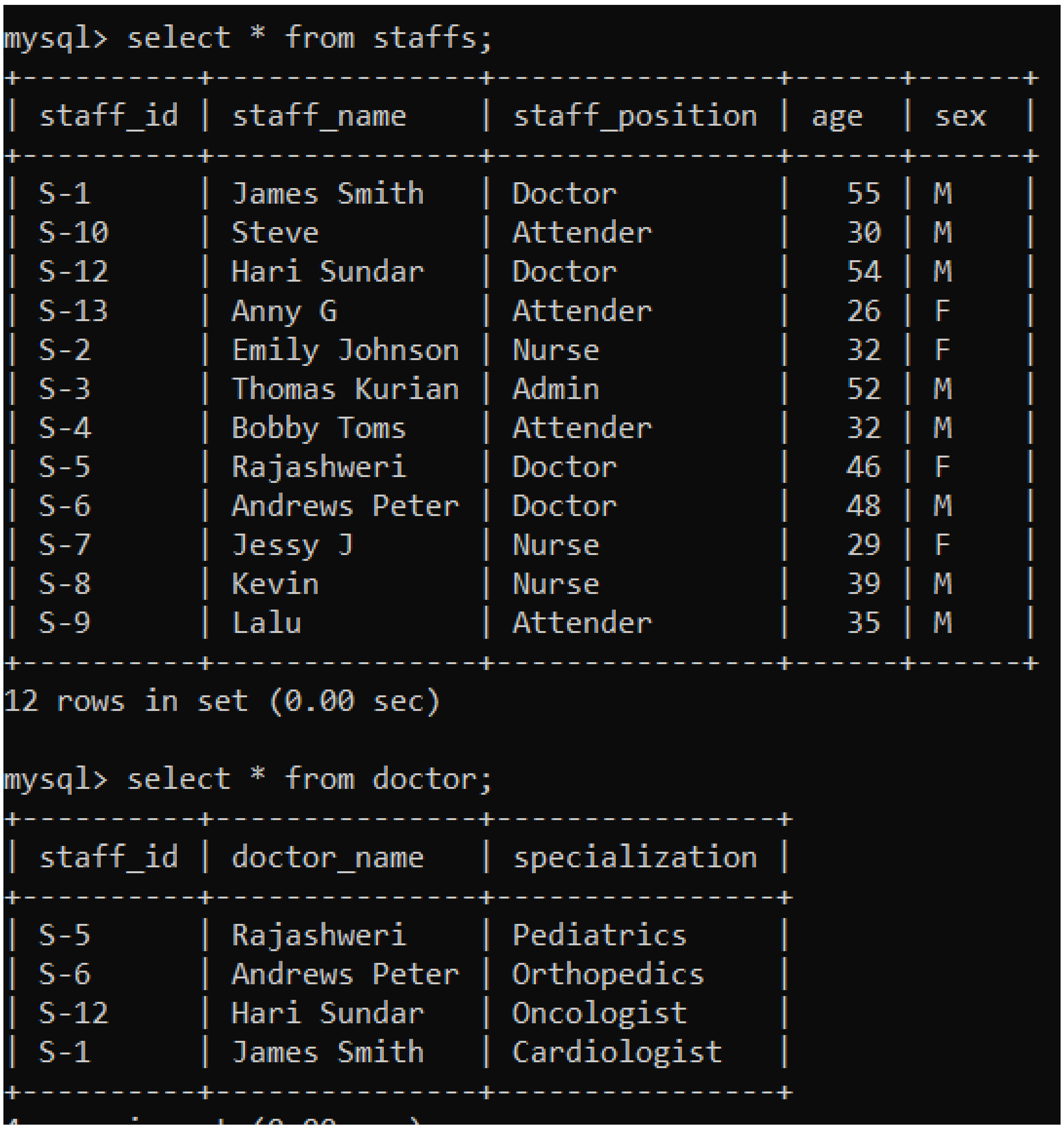
Staff Name: Joel James

Fig 8.2



Staff removed successfully!

Fig 8.3



```
mysql> select * from staffs;
+----------+--------------+----------------+------+------+
| staff_id | staff_name   | staff_position | age  | sex  |
+----------+--------------+----------------+------+------+
| S-1      | James Smith  | Doctor         |   55 | M    |
| S-10     | Steve        | Attender       |   30 | M    |
| S-12     | Hari Sundar  | Doctor         |   54 | M    |
| S-13     | Anny G       | Attender       |   26 | F    |
| S-2      | Emily Johnson| Nurse          |   32 | F    |
| S-3      | Thomas Kurian| Admin          |   52 | M    |
| S-4      | Bobby Toms   | Attender       |   32 | M    |
| S-5      | Rajashweri   | Doctor         |   46 | F    |
| S-6      | Andrews Peter| Doctor         |   48 | M    |
| S-7      | Jessy J      | Nurse          |   29 | F    |
| S-8      | Kevin        | Nurse          |   39 | M    |
| S-9      | Lalu         | Attender       |   35 | M    |
+----------+--------------+----------------+------+------+
12 rows in set (0.00 sec)

mysql> select * from doctor;
+----------+--------------+----------------+
| staff_id | doctor_name  | specialization |
+----------+--------------+----------------+
| S-5      | Rajashweri   | Pediatrics     |
| S-6      | Andrews Peter| Orthopedics    |
| S-12     | Hari Sundar  | Oncologist     |
| S-1      | James Smith  | Cardiologist   |
+----------+--------------+----------------+
```

Fig 8.4

Fig 8.1: Remove Staff option in Edit Staffs Window
Fig 8.2: Details of Staff to be removed.
Fig 8.4: Staff (Doctor) removed from ‘staffs’ and ‘doctor’ tables.

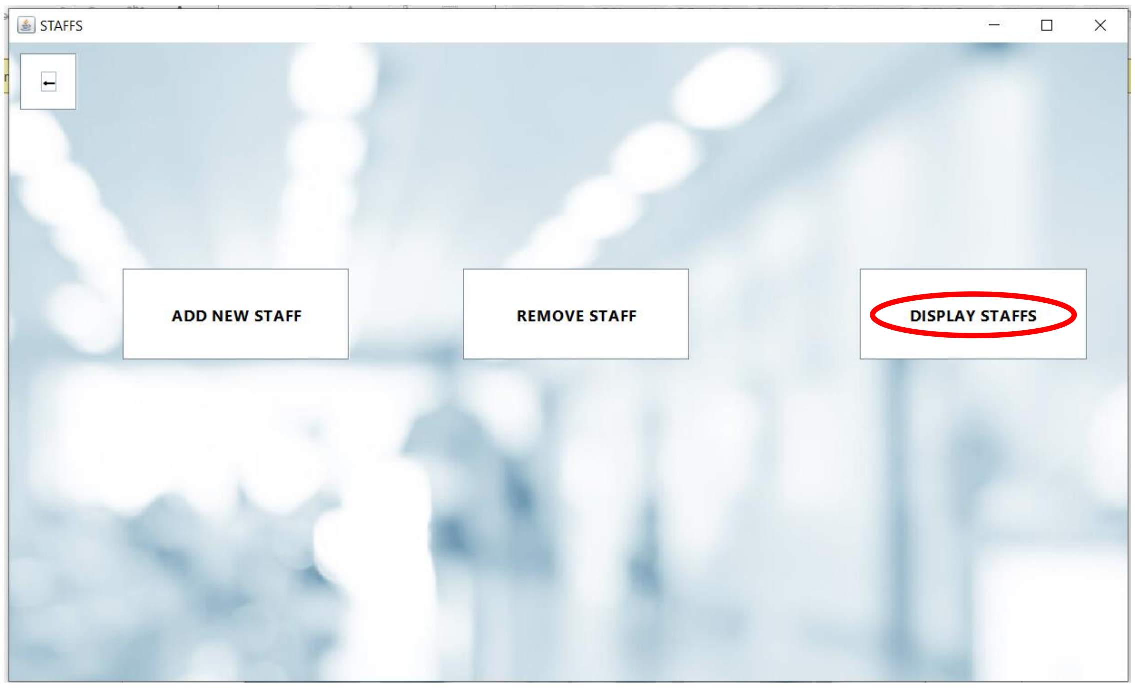
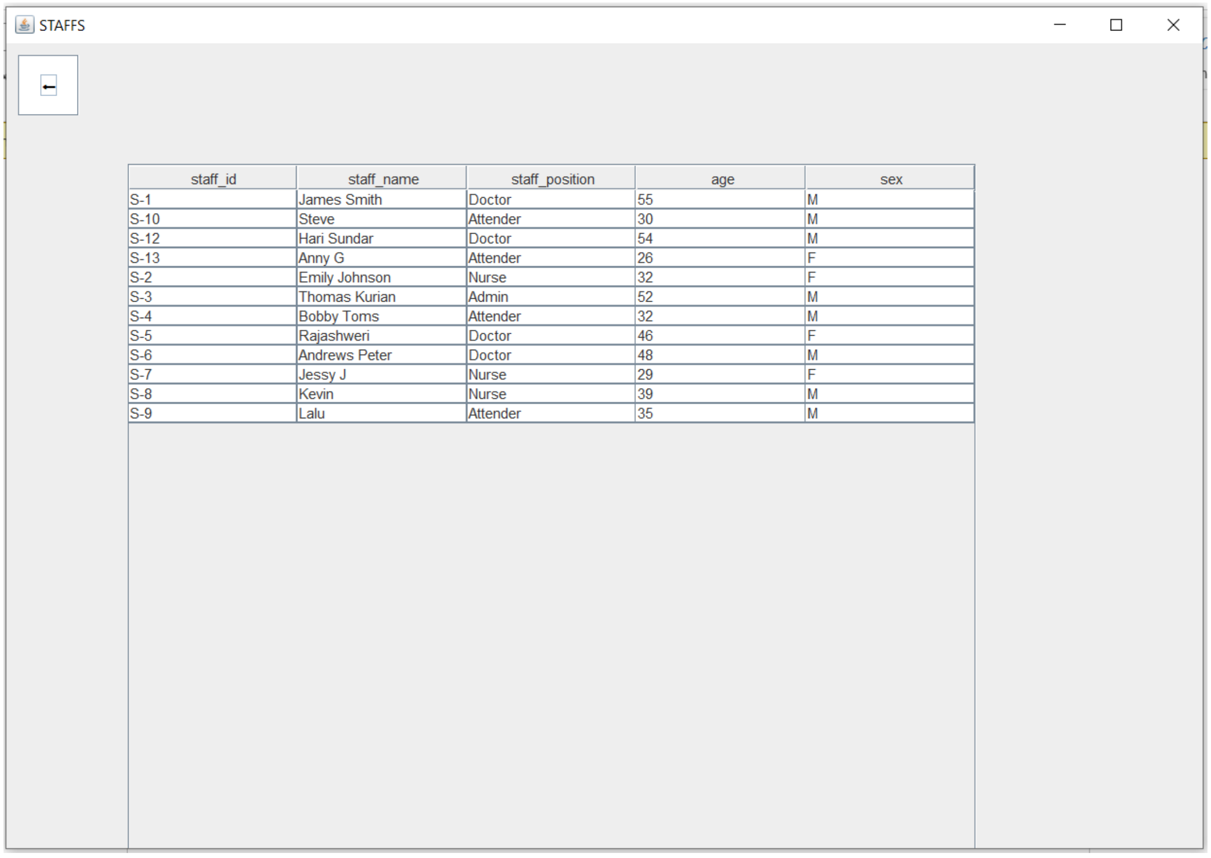


Fig 9.1: Display Staffs option in Edit Staff window.



staff_id	staff_name	staff_position	age	sex
S-1	James Smith	Doctor	55	M
S-10	Steve	Attender	30	M
S-12	Hari Sundar	Doctor	54	M
S-13	Anny G	Attender	26	F
S-2	Emily Johnson	Nurse	32	F
S-3	Thomas Kurian	Admin	52	M
S-4	Bobby Toms	Attender	32	M
S-5	Rajashweri	Doctor	46	F
S-6	Andrews Peter	Doctor	48	M
S-7	Jessy J	Nurse	29	F
S-8	Kevin	Nurse	39	M
S-9	Lalu	Attender	35	M

Fig 9.2: List of Staffs Info.

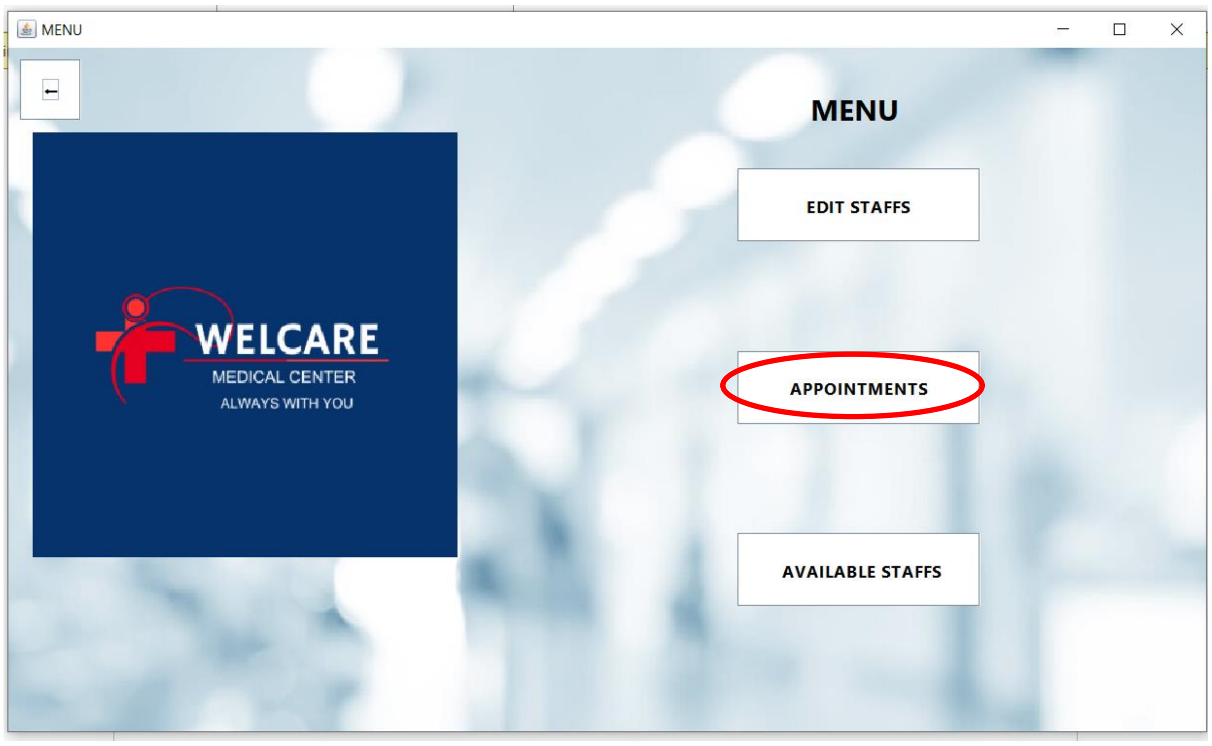


Fig 10.1: Appointments

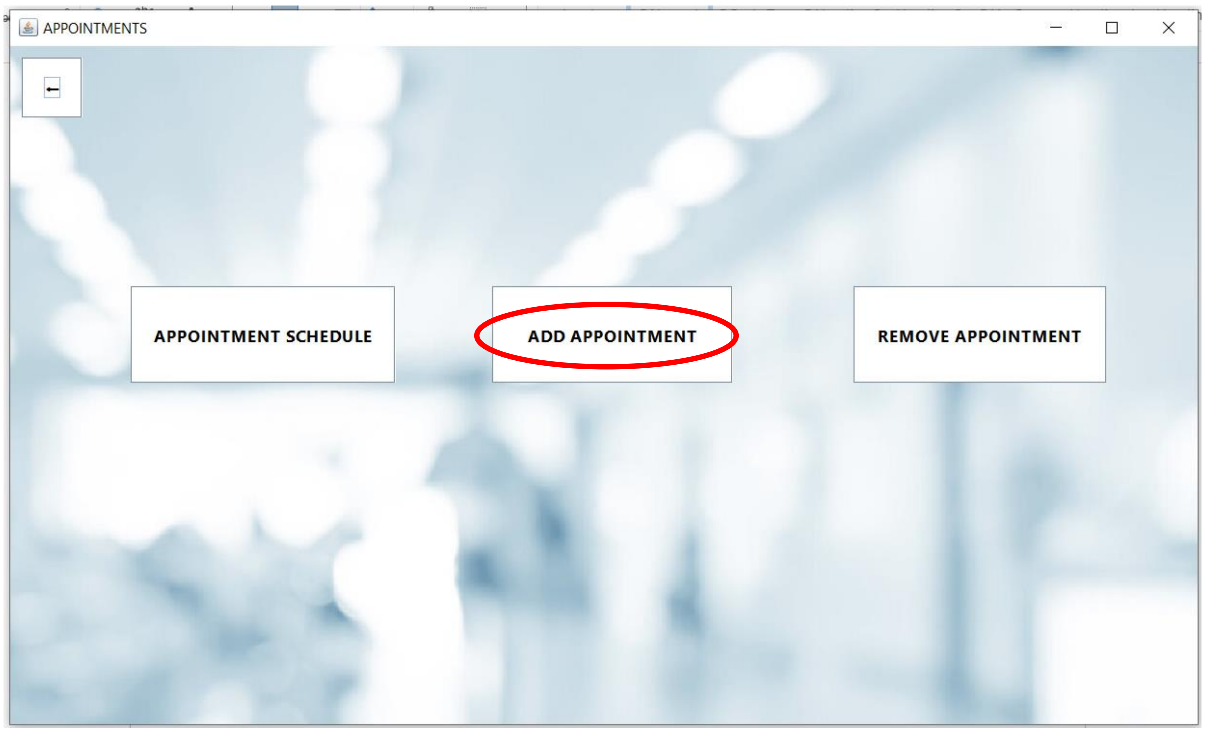
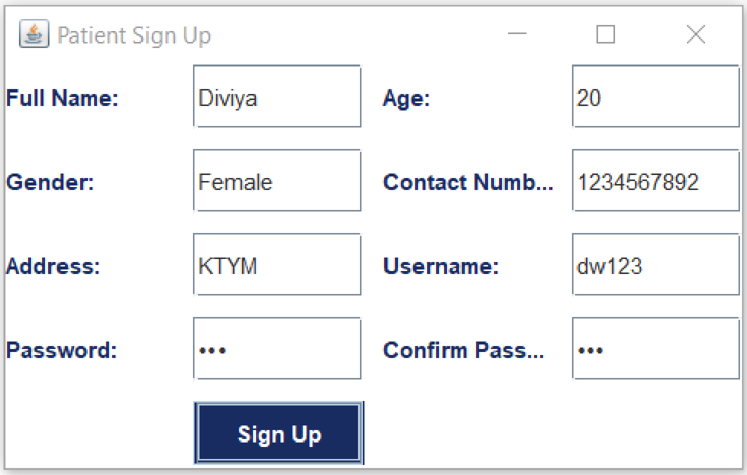


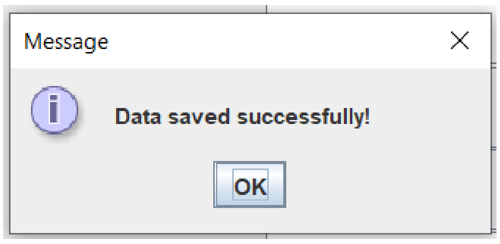
Fig 10.2: Add Appointment option in Appointments Window



Patient Sign Up

Full Name:	Diviya	**Age:**	20
Gender:	Female	**Contact Numb...**	1234567892
Address:	KTYM	**Username:**	dw123
Password:	•••	**Confirm Pass...**	•••

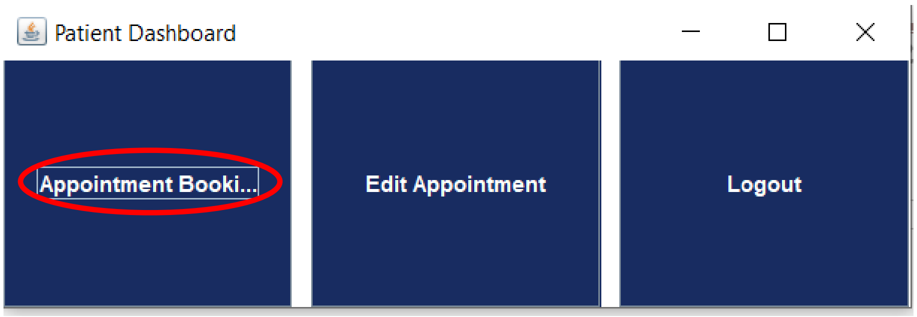
Fig 10.3



Message

Data saved successfully!

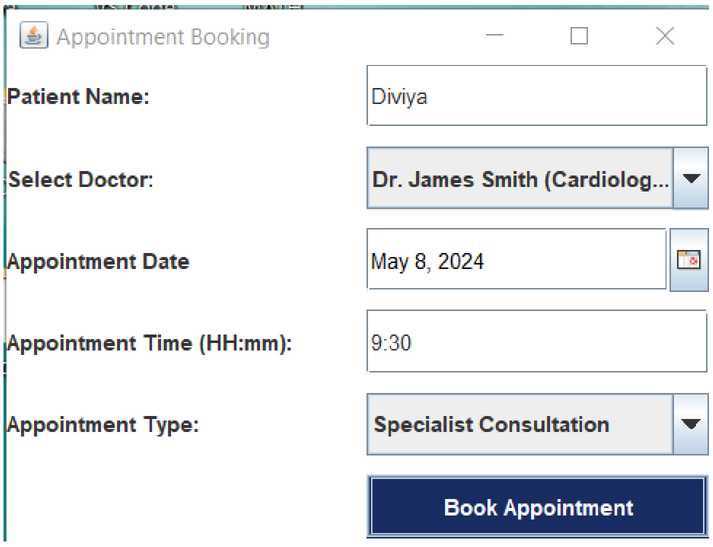
Fig 10.4



Patient Dashboard

Fig 10.5

Fig(s) 10.3 ,10.4,10.5: New Patient registered before Appointment Booking.



Patient Name: Diviya

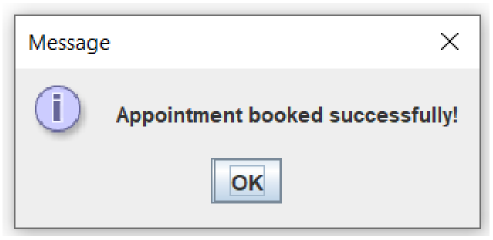
Select Doctor: Dr. James Smith (Cardiolog...

Appointment Date May 8, 2024

Appointment Time (HH:mm): 9:30

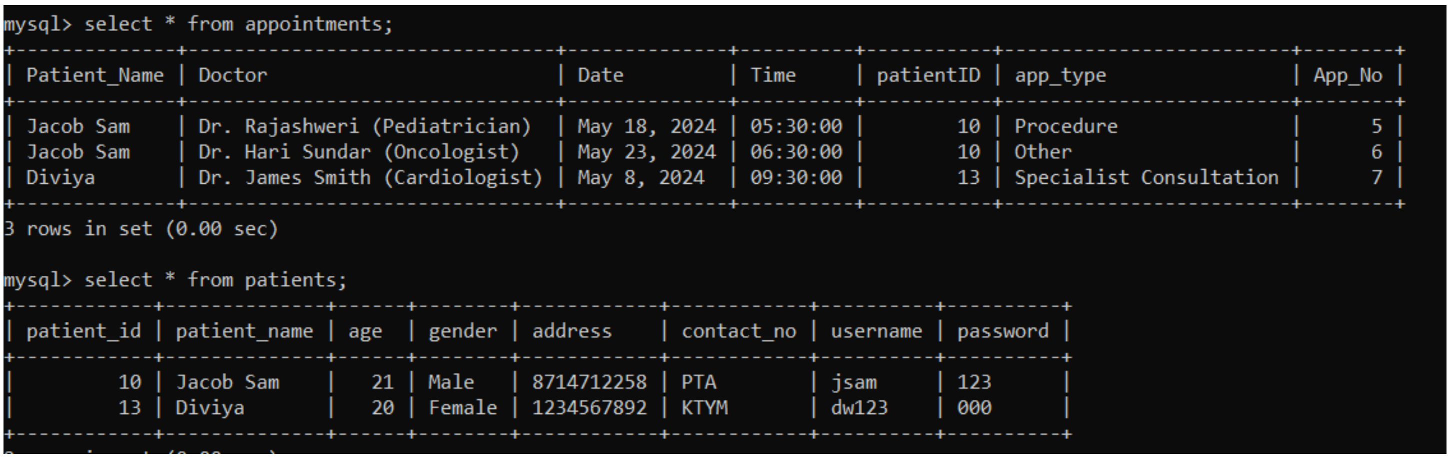
Appointment Type: Specialist Consultation

Fig 10.6



Appointment booked successfully!

Fig 10.7



```
mysql> select * from appointments;
+-------------+-------------------------------+--------------+----------+-----------+-------------------------+--------+
| Patient_Name | Doctor                        | Date         | Time     | patientID | app_type                | App_No |
+-------------+-------------------------------+--------------+----------+-----------+-------------------------+--------+
| Jacob Sam   | Dr. Rajashweri (Pediatrician) | May 18, 2024 | 05:30:00 |        10 | Procedure               |      5 |
| Jacob Sam   | Dr. Hari Sundar (Oncologist)  | May 23, 2024 | 06:30:00 |        10 | Other                   |      6 |
| Diviya      | Dr. James Smith (Cardiologist)| May 8, 2024  | 09:30:00 |        13 | Specialist Consultation |      7 |
+-------------+-------------------------------+--------------+----------+-----------+-------------------------+--------+
3 rows in set (0.00 sec)

mysql> select * from patients;
+------------+--------------+-----+--------+------------+------------+----------+----------+
| patient_id | patient_name | age | gender | address    | contact_no | username | password |
+------------+--------------+-----+--------+------------+------------+----------+----------+
|         10 | Jacob Sam    |  21 | Male   | 8714712258 | PTA        | jsam     | 123      |
|         13 | Diviya       |  20 | Female | 1234567892 | KTYM       | dw123    | 000      |
+------------+--------------+-----+--------+------------+------------+----------+----------+
```

Fig 10.8

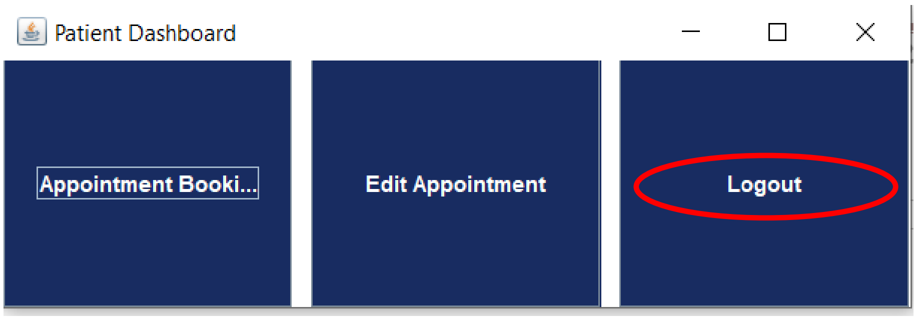


Fig 10.9: Log out option

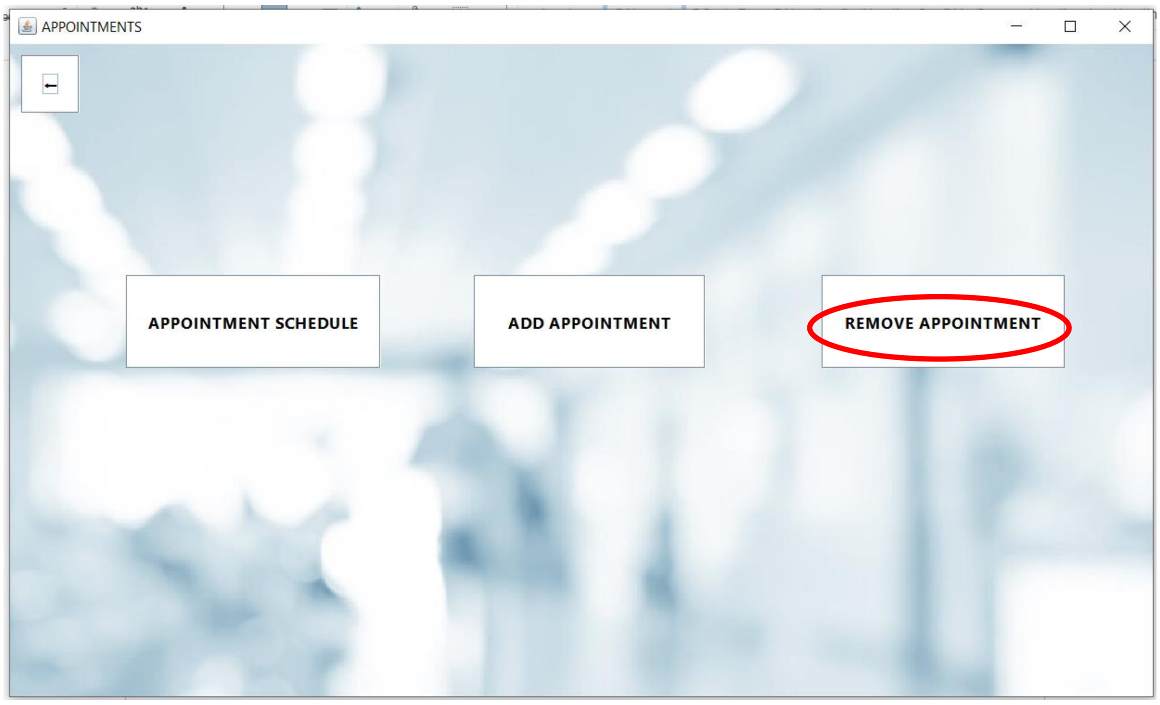
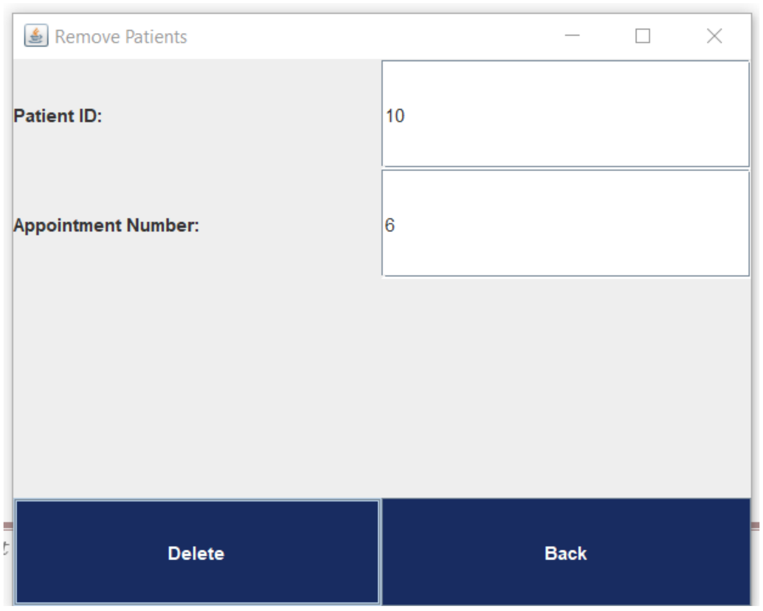


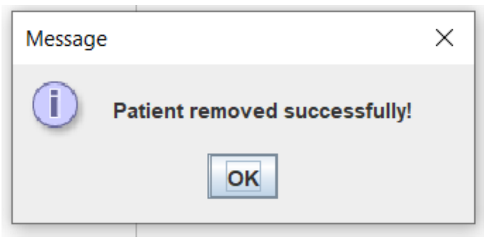
Fig 11: Remove Appointment option in Appointments window



Patient ID: 10

Appointment Number: 6

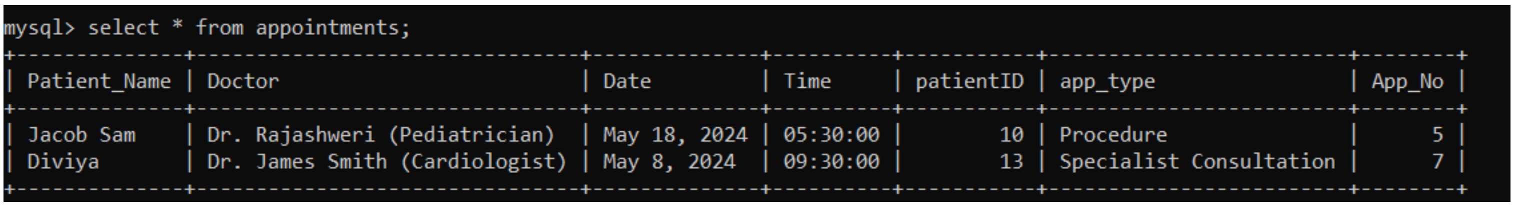
Fig 11.1



Message

Patient removed successfully!

Fig 11.2



```
mysql> select * from appointments;
+--------------+---------------------------------+--------------+----------+-----------+--------------------------+--------+
| Patient_Name | Doctor                          | Date         | Time     | patientID | app_type                 | App_No |
+--------------+---------------------------------+--------------+----------+-----------+--------------------------+--------+
| Jacob Sam    | Dr. Rajashweri (Pediatrician)   | May 18, 2024 | 05:30:00 |        10 | Procedure                |      5 |
| Diviya       | Dr. James Smith (Cardiologist)  | May 8, 2024  | 09:30:00 |        13 | Specialist Consultation  |      7 |
+--------------+---------------------------------+--------------+----------+-----------+--------------------------+--------+
```

Fig 11.3

Fig 11.1: Appointment Removal using Appointment Number and Patient ID
Fig 11.3: Removed from appointments.

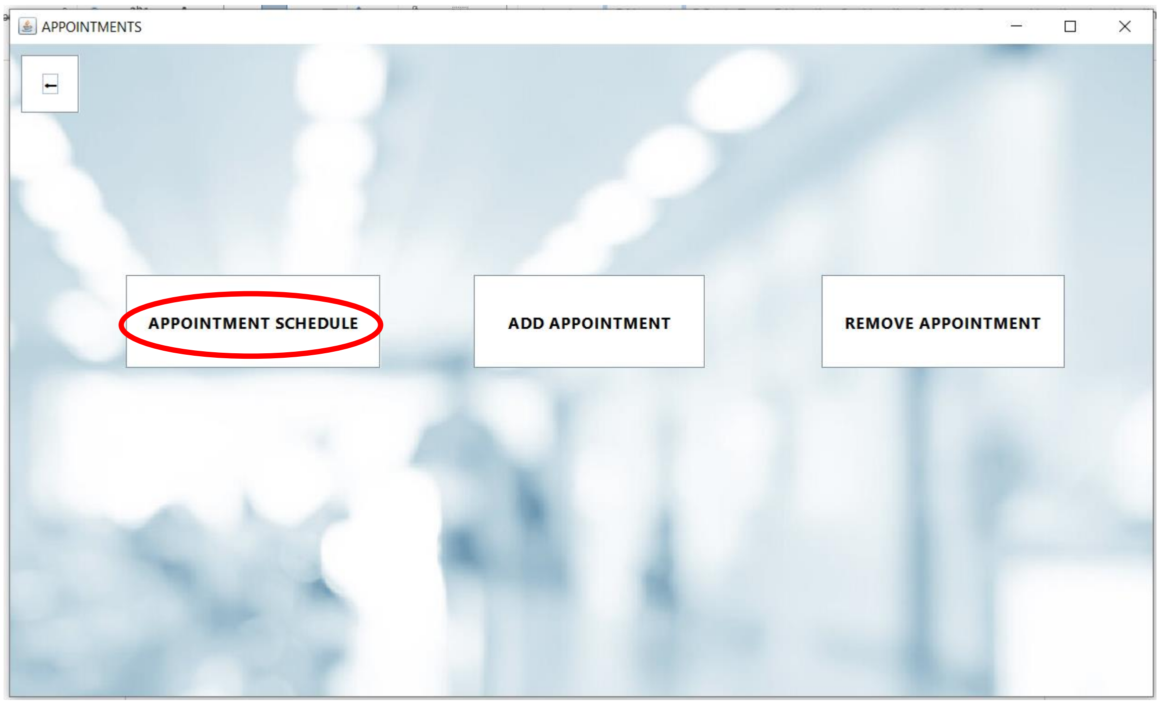
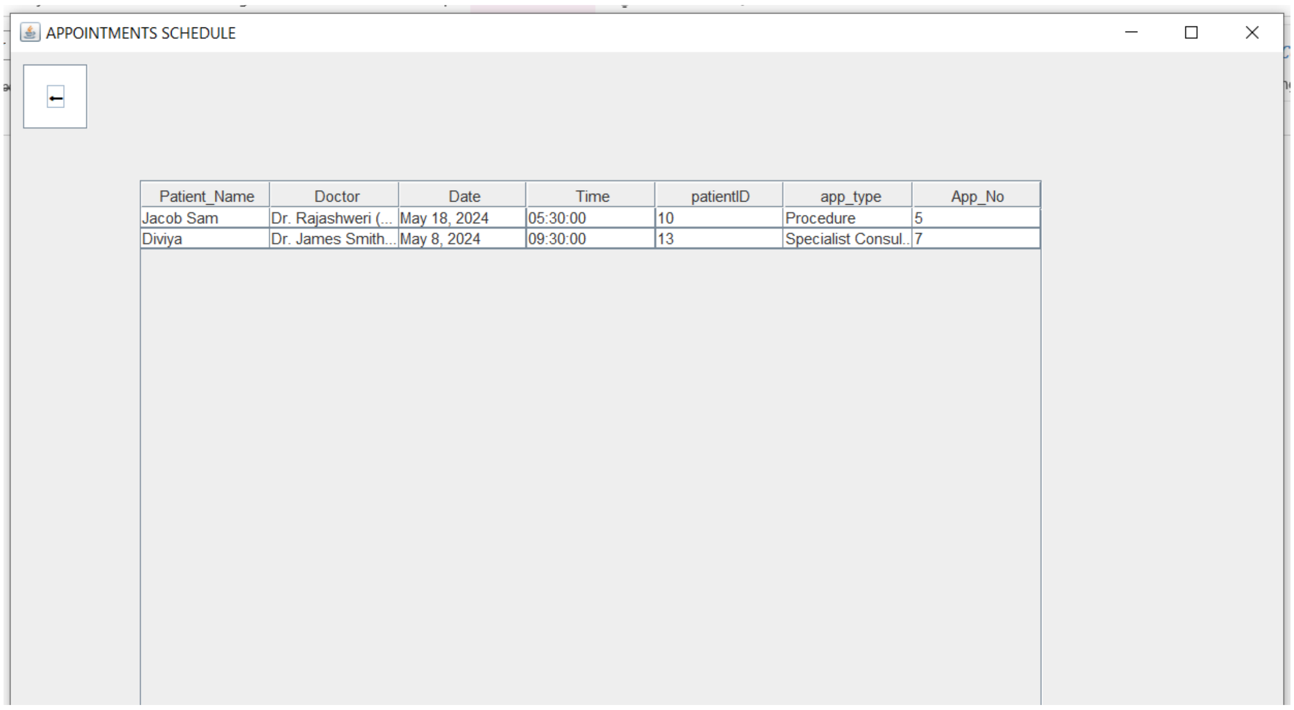


Fig 12.1



Patient_Name	Doctor	Date	Time	patientID	app_type	App_No
Jacob Sam	Dr. Rajashweri (...	May 18, 2024	05:30:00	10	Procedure	5
Diviya	Dr. James Smith...	May 8, 2024	09:30:00	13	Specialist Consul...	7

Fig 12.2

Fig 12.1: Appointment Schedule option in Appointments window.
Fig 12.2: View Appointments Schedule.

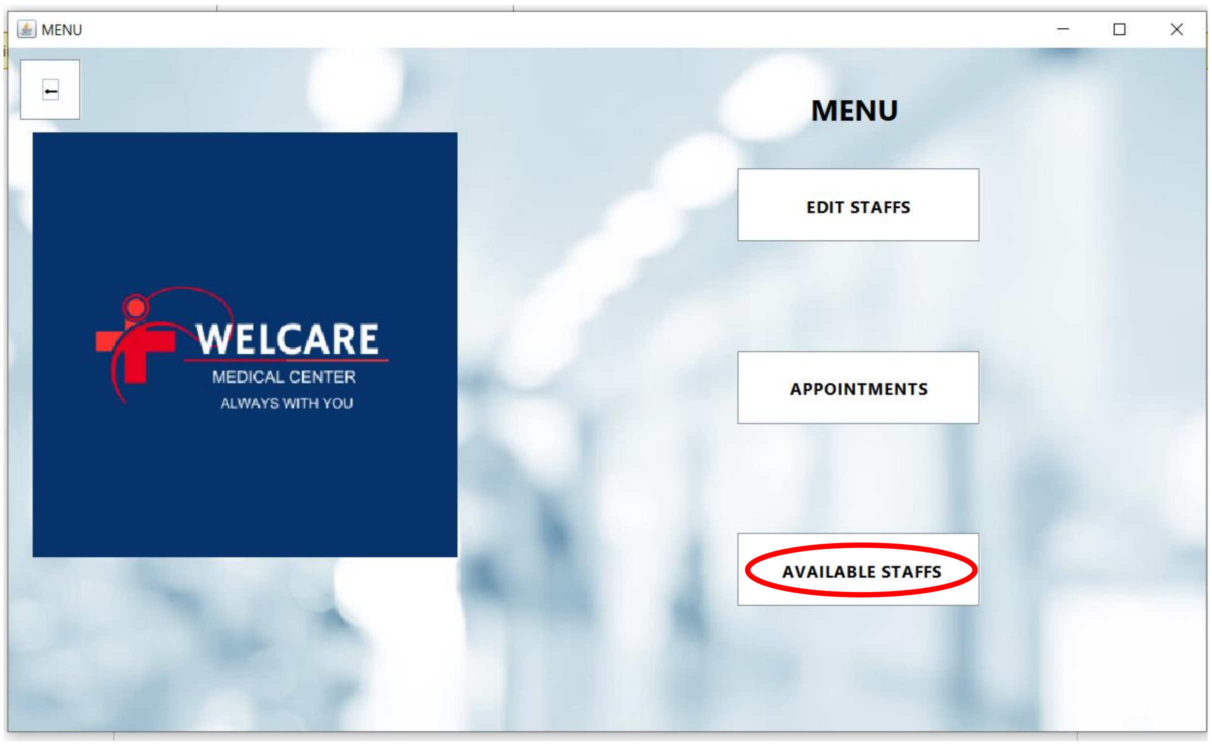


Fig 13.1: Available Staffs Button

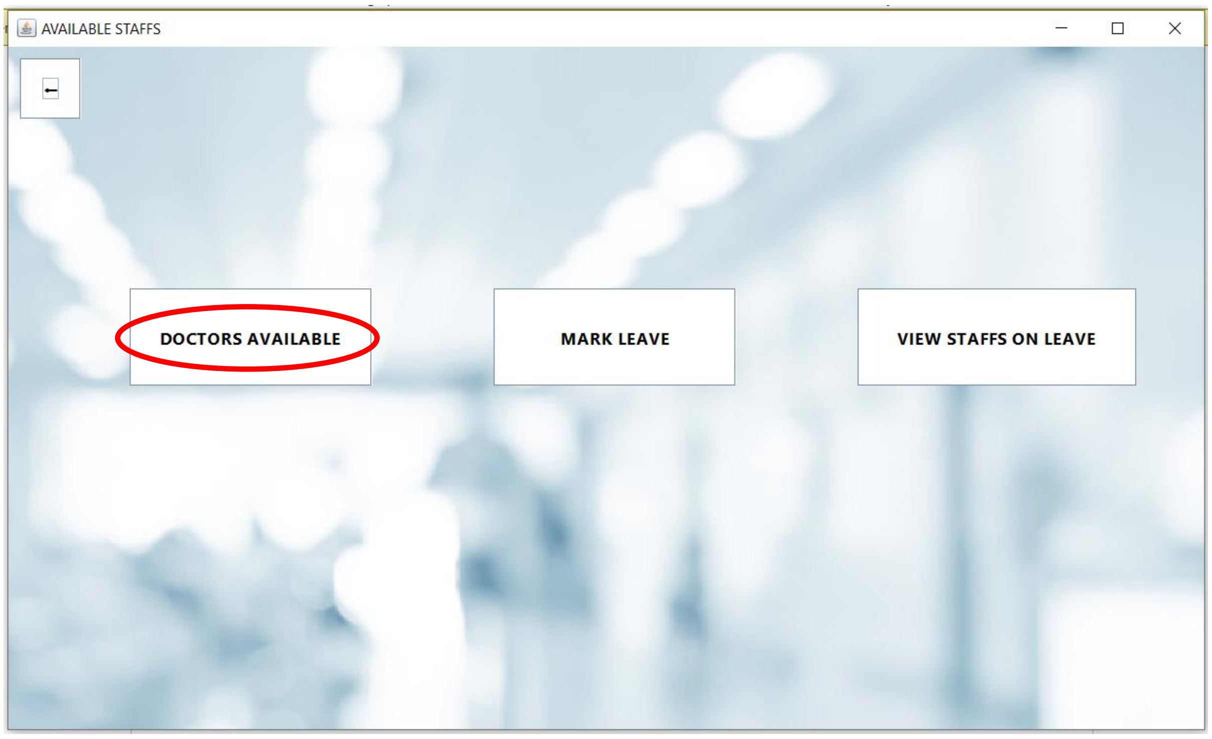
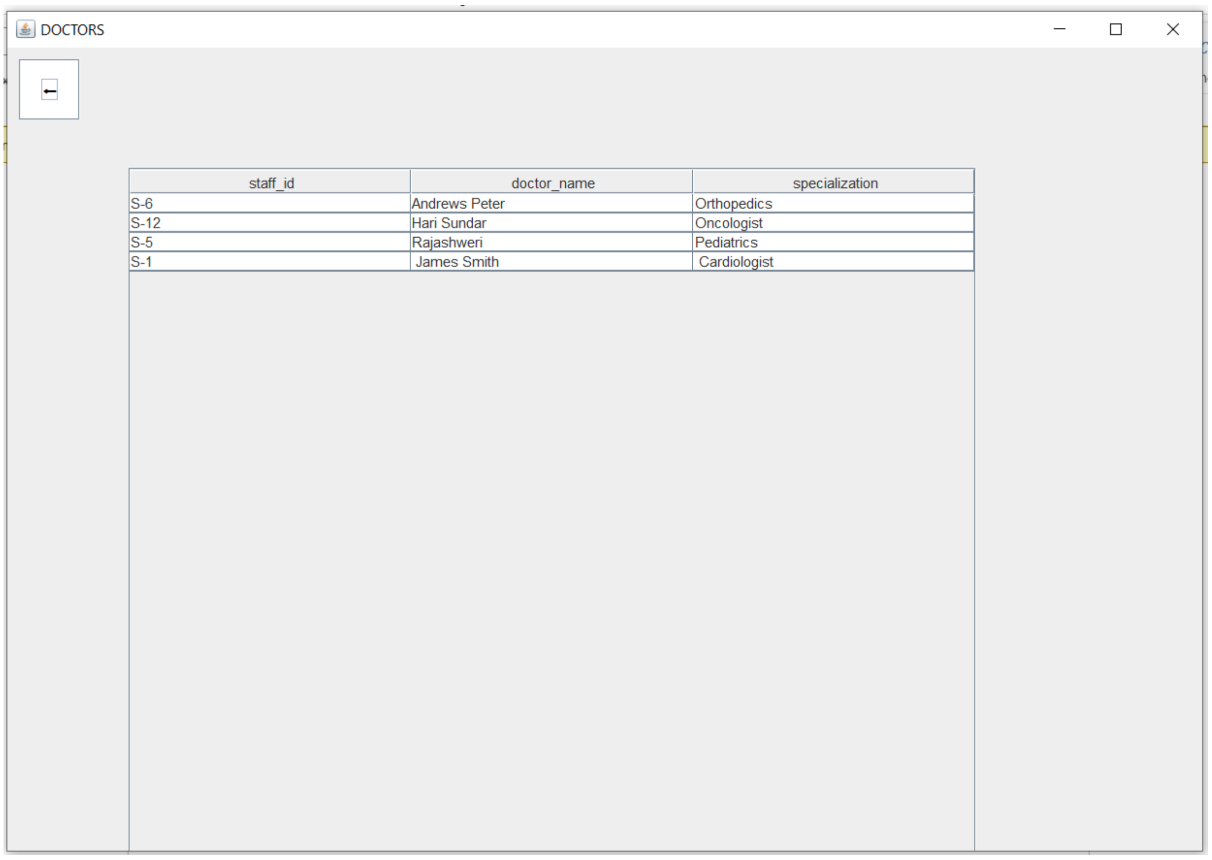


Fig 13.2: Doctors Available option in Available Staffs window


DOCTORS

staff_id	doctor_name	specialization
S-6	Andrews Peter	Orthopedics
S-12	Hari Sundar	Oncologist
S-5	Rajashweri	Pediatrics
S-1	James Smith	Cardiologist

Fig 13.3: View Doctors available

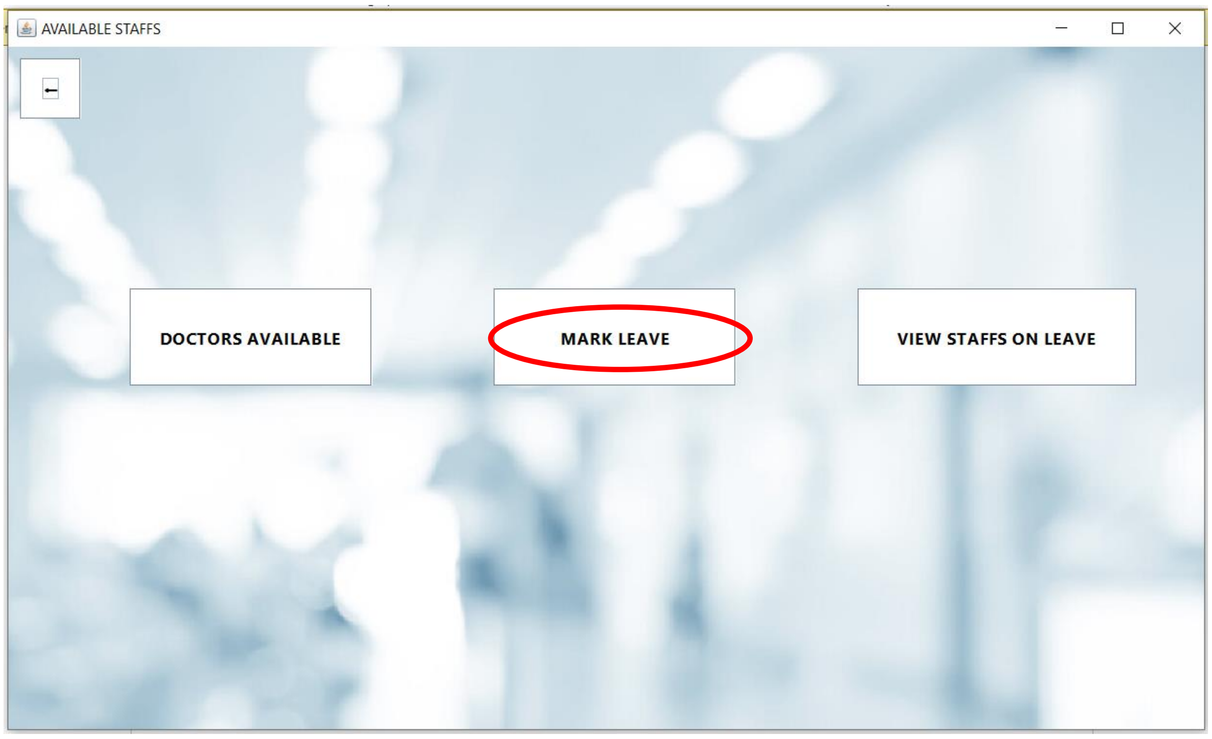
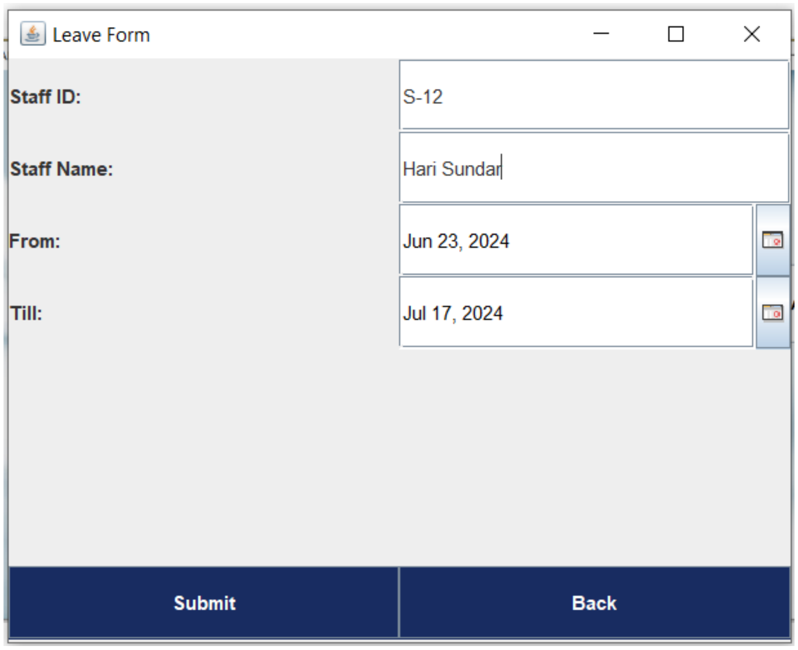


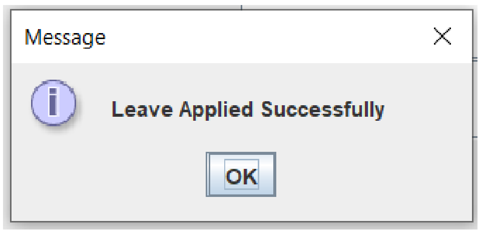
Fig 14.1: Mark Leave for any staff



Leave Form

Staff ID:	S-12
Staff Name:	Hari Sundar
From:	Jun 23, 2024
Till:	Jul 17, 2024

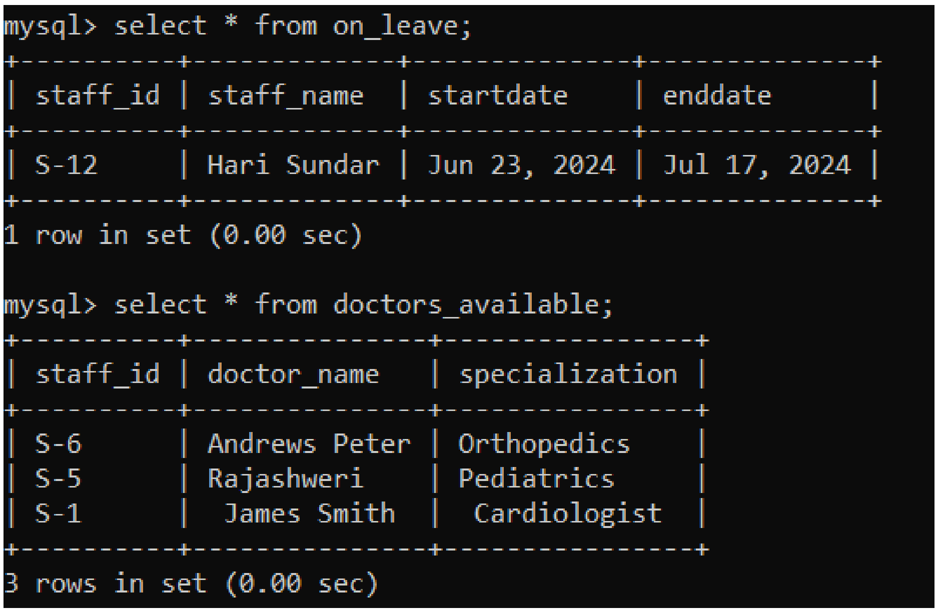
Fig 14.2: Fill Staff details



Message

Leave Applied Successfully

Fig 14.3



```
mysql> select * from on_leave;
+----------+-------------+--------------+--------------+
| staff_id | staff_name  | startdate    | enddate      |
+----------+-------------+--------------+--------------+
| S-12     | Hari Sundar | Jun 23, 2024 | Jul 17, 2024 |
+----------+-------------+--------------+--------------+
1 row in set (0.00 sec)

mysql> select * from doctors_available;
+----------+---------------+----------------+
| staff_id | doctor_name   | specialization |
+----------+---------------+----------------+
| S-6      | Andrews Peter | Orthopedics    |
| S-5      | Rajashweri    | Pediatrics     |
| S-1      |  James Smith  |  Cardiologist  |
+----------+---------------+----------------+
3 rows in set (0.00 sec)
```

Fig 14.4: Add Staffs to on leave table.
Also removed from 'doctors available' table.

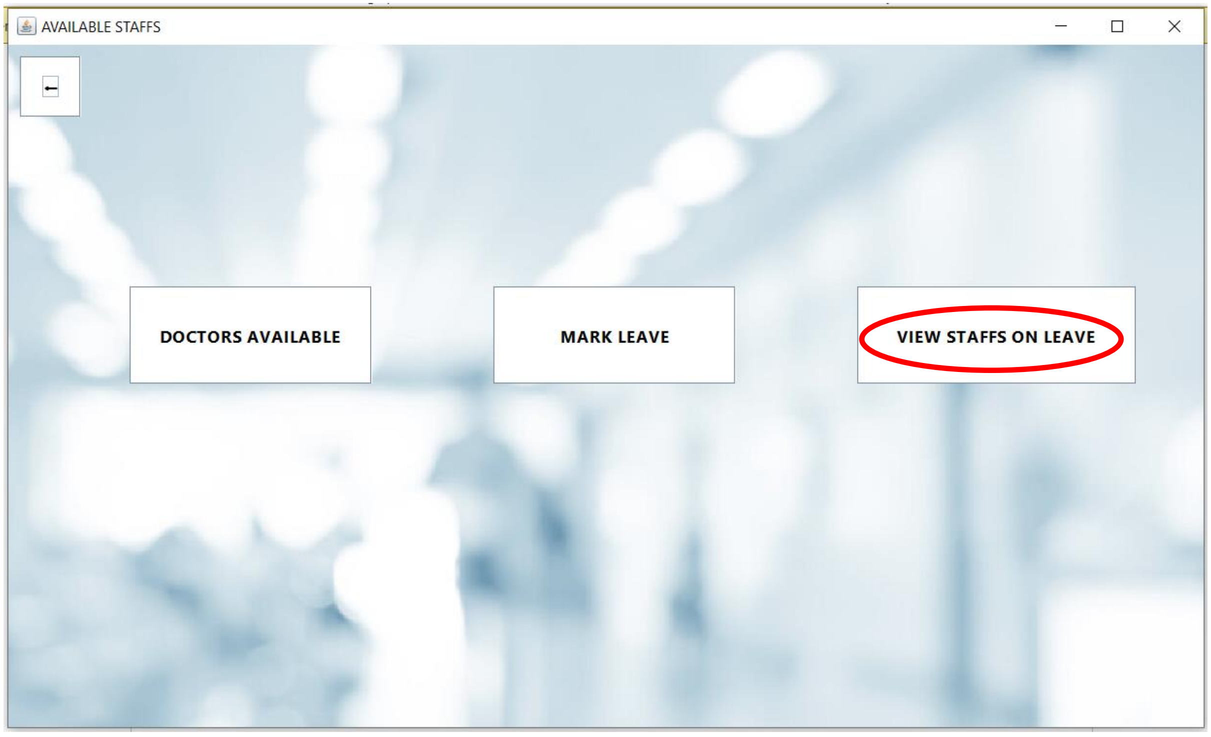
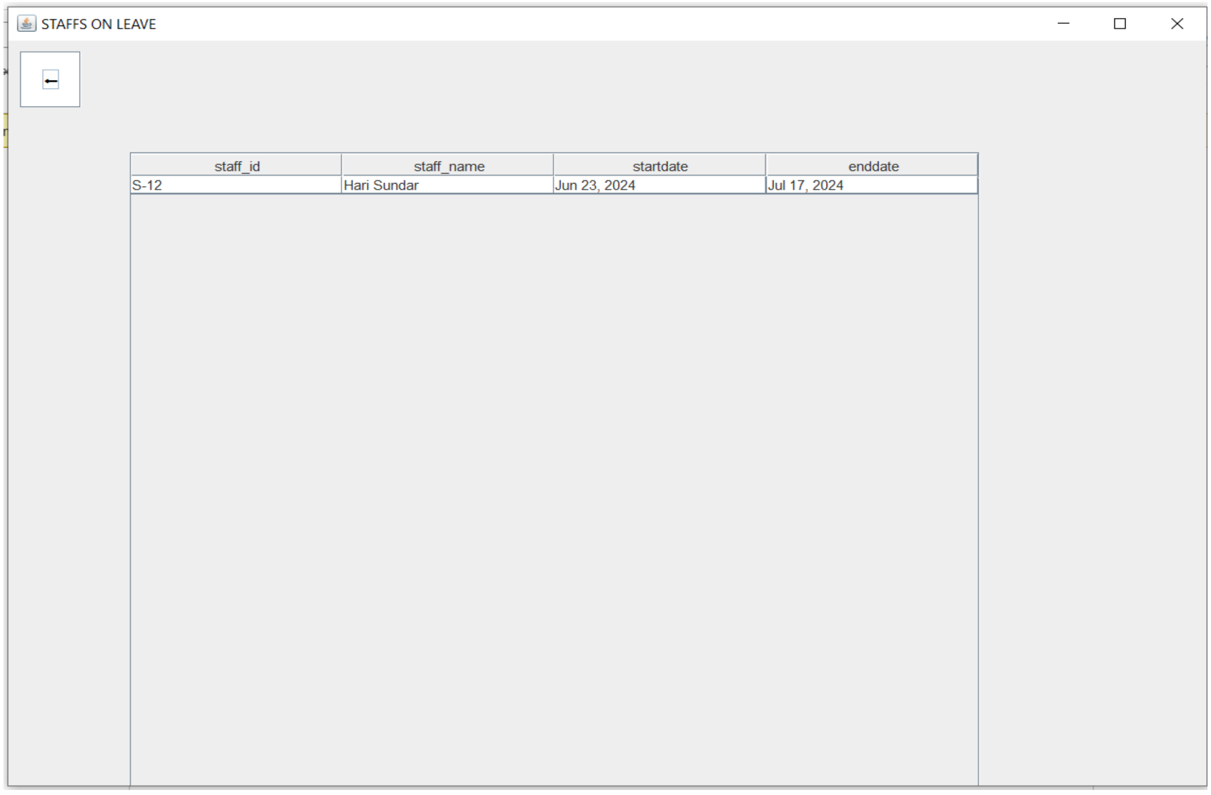


Fig 15.1: View staffs on leave



staff_id	staff_name	startdate	enddate
S-12	Hari Sundar	Jun 23, 2024	Jul 17, 2024

Fig 15.2

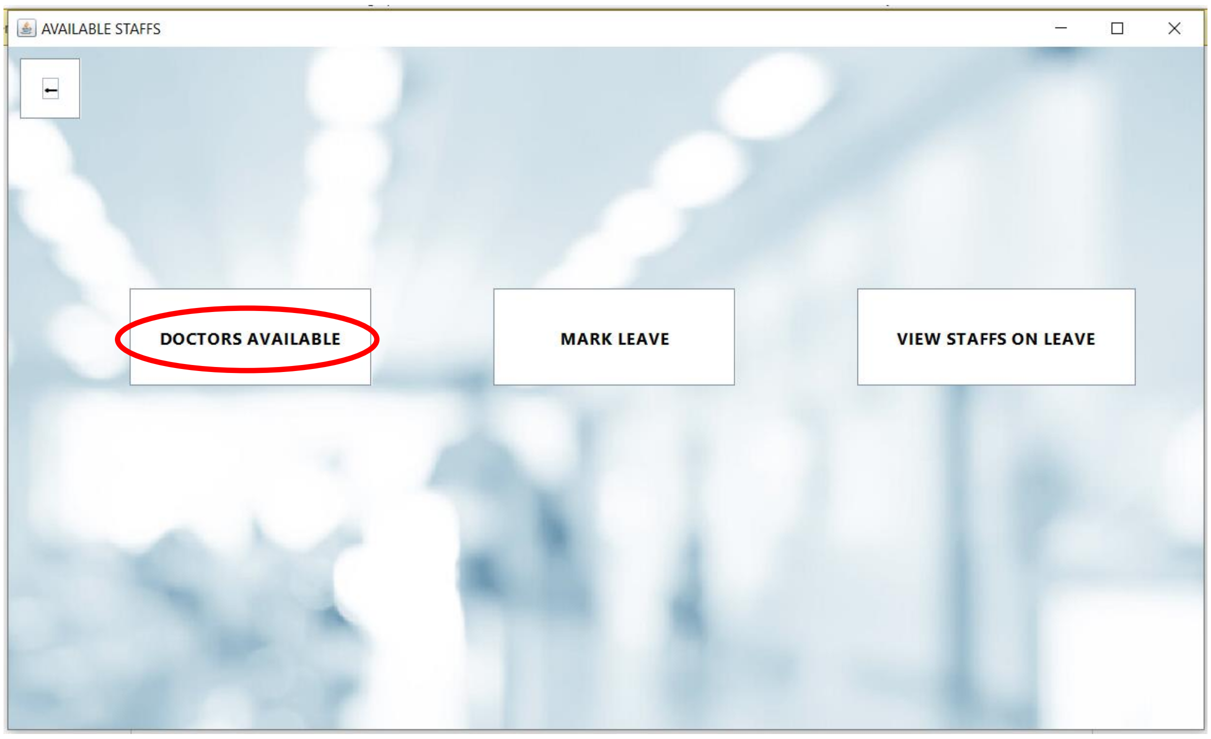
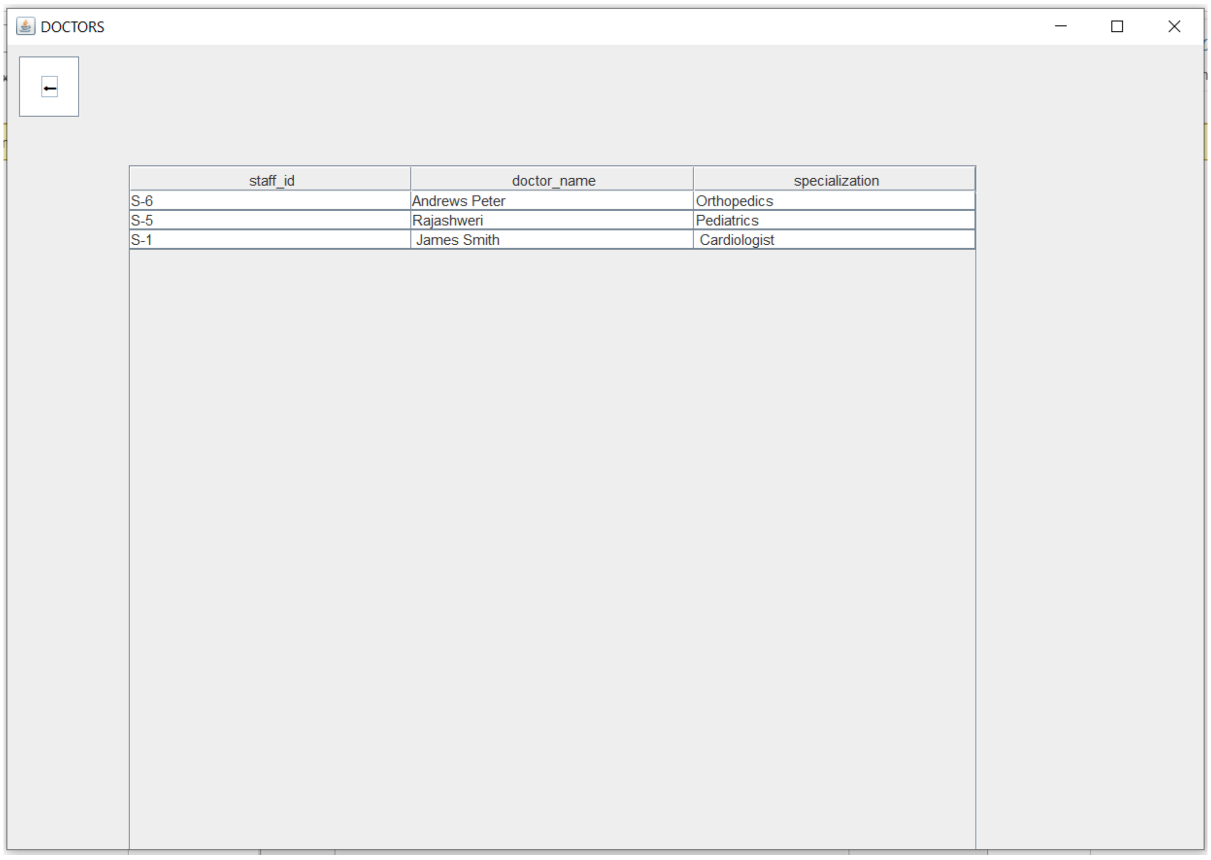


Fig 15.3

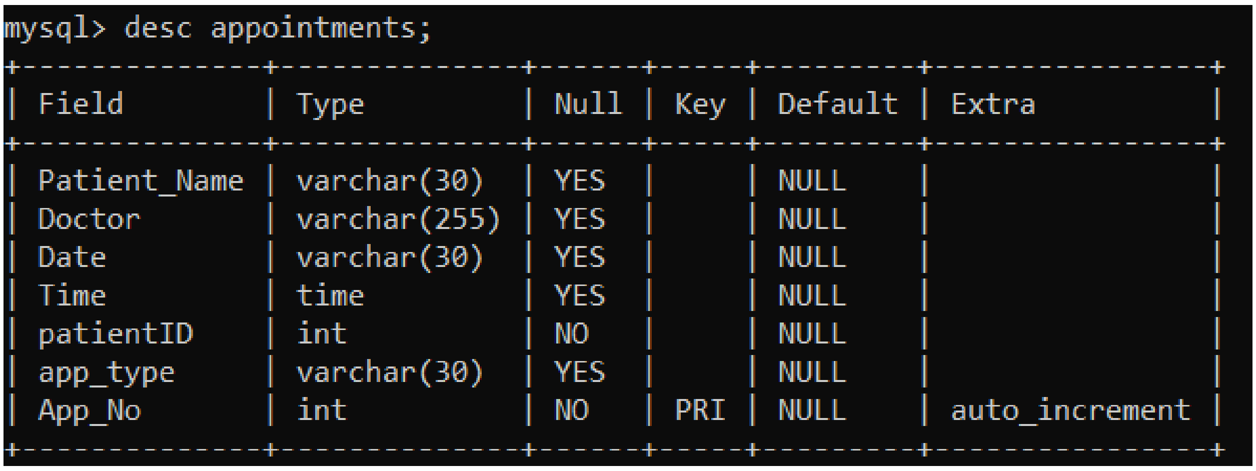


staff_id	doctor_name	specialization
S-6	Andrews Peter	Orthopedics
S-5	Rajashweri	Pediatrics
S-1	James Smith	Cardiologist

Fig 15.4

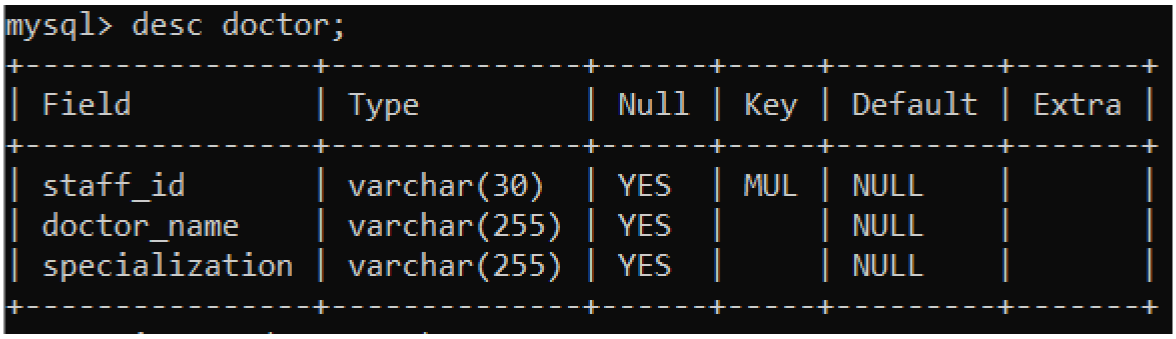
Structure of Tables

Table: Appointments



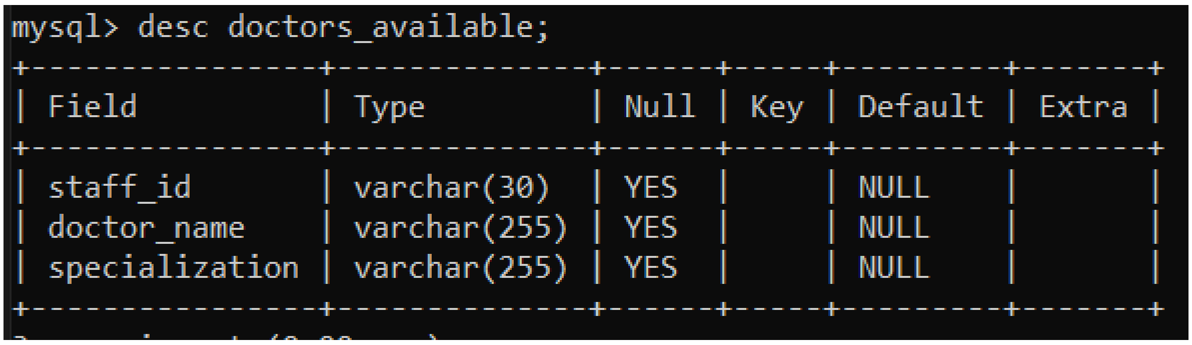
```
mysql> desc appointments;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| Patient_Name | varchar(30)  | YES  |     | NULL    |                |
| Doctor       | varchar(255) | YES  |     | NULL    |                |
| Date         | varchar(30)  | YES  |     | NULL    |                |
| Time         | time         | YES  |     | NULL    |                |
| patientID    | int          | NO   |     | NULL    |                |
| app_type     | varchar(30)  | YES  |     | NULL    |                |
| App_No       | int          | NO   | PRI | NULL    | auto_increment |
+--------------+--------------+------+-----+---------+----------------+
```

Table: doctor



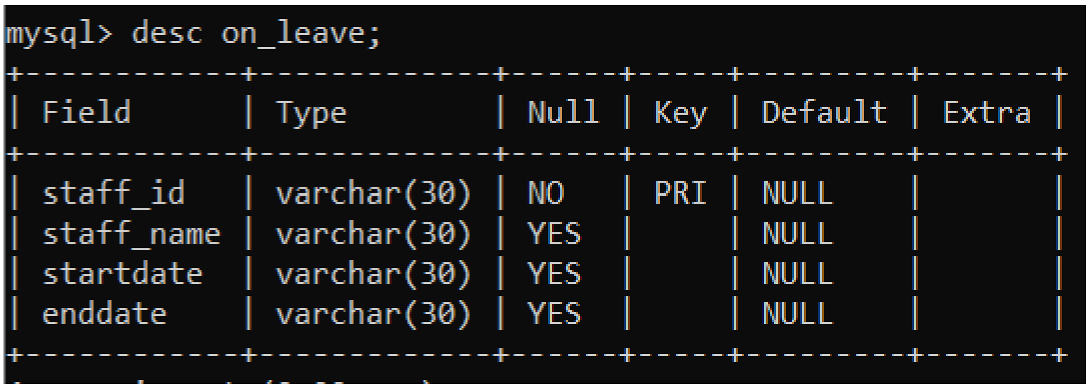
```
mysql> desc doctor;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| staff_id       | varchar(30)  | YES  | MUL | NULL    |       |
| doctor_name    | varchar(255) | YES  |     | NULL    |       |
| specialization | varchar(255) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

Table: doctors_available



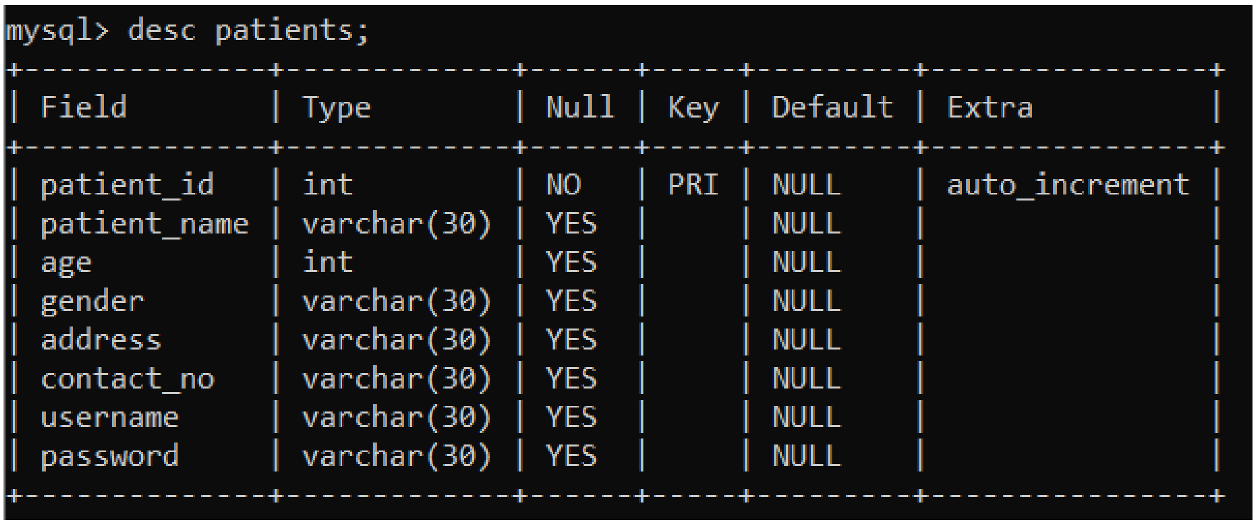
```
mysql> desc doctors_available;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| staff_id       | varchar(30)  | YES  |     | NULL    |       |
| doctor_name    | varchar(255) | YES  |     | NULL    |       |
| specialization | varchar(255) | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

Table: on_leave



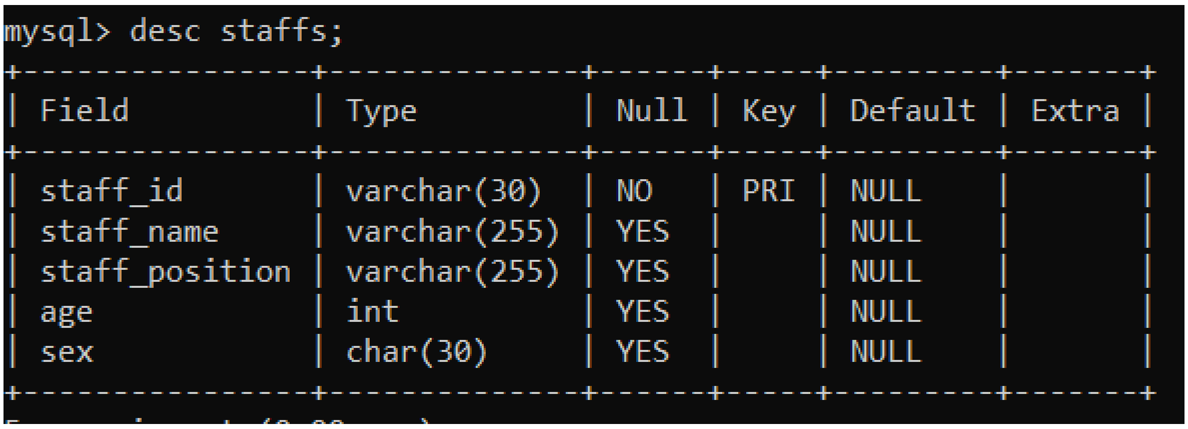
```
mysql> desc on_leave;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| staff_id   | varchar(30) | NO   | PRI | NULL    |       |
| staff_name | varchar(30) | YES  |     | NULL    |       |
| startdate  | varchar(30) | YES  |     | NULL    |       |
| enddate    | varchar(30) | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Table: patients



```
mysql> desc patients;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| patient_id   | int         | NO   | PRI | NULL    | auto_increment |
| patient_name | varchar(30) | YES  |     | NULL    |                |
| age          | int         | YES  |     | NULL    |                |
| gender       | varchar(30) | YES  |     | NULL    |                |
| address      | varchar(30) | YES  |     | NULL    |                |
| contact_no   | varchar(30) | YES  |     | NULL    |                |
| username     | varchar(30) | YES  |     | NULL    |                |
| password     | varchar(30) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
```

Table: staffs



```
mysql> desc staffs;
+----------------+--------------+------+-----+---------+-------+
| Field          | Type         | Null | Key | Default | Extra |
+----------------+--------------+------+-----+---------+-------+
| staff_id       | varchar(30)  | NO   | PRI | NULL    |       |
| staff_name     | varchar(255) | YES  |     | NULL    |       |
| staff_position | varchar(255) | YES  |     | NULL    |       |
| age            | int          | YES  |     | NULL    |       |
| sex            | char(30)     | YES  |     | NULL    |       |
+----------------+--------------+------+-----+---------+-------+
```

CONCLUSION

<u>Advantages & Applications of the Hospital Management System:</u>

Advantages:

a. Streamlined Workflow: The HMS offers a streamlined and efficient workflow for managing hospital operations, including patient records, appointments, and staff administration. This enhances operational efficiency and reduces administrative burdens.

b. User-Friendly Interface: The HMS features a user-friendly interface designed to facilitate easy navigation and seamless interaction for hospital staff. Its intuitive design enhances user experience and promotes productivity.

c. Customization Options: Hospital administrators have the flexibility to customize the system according to their specific requirements. This includes configuring appointment schedules, staff roles, and patient categories to suit the hospital's workflow.

d. Enhanced Data Management: The integration with a secure database system ensures efficient storage and retrieval of patient records, appointment details, and staff information. This promotes data integrity, security, and scalability within the hospital environment.

Real-time Updates and Notifications: The HMS provides real-time updates and notifications for appointments, patient admissions, and staff scheduling. This ensures timely communication and facilitates effective coordination among hospital personnel.

Applications:

a. Hospital Administration: The HMS serves as a comprehensive solution for hospital administration, enabling efficient management of patient records, appointments, and staff resources.

b. Patient Care: Healthcare professionals can utilize the HMS to access patient records, track medical histories, and monitor treatment plans. This promotes continuity of care and enhances patient outcomes.

c. Appointment Scheduling: The system facilitates appointment scheduling and management, allowing patients to book appointments online, view available slots, and receive reminders for upcoming visits.

d. Staff Management: Hospital administrators can use the HMS to manage staff schedules, assign duties, and track employee performance. This optimizes staff allocation and enhances workforce productivity.

e. Reporting and Analysis: The HMS generates reports and analytics on key performance indicators such as patient admissions, appointment trends, and resource utilization. This enables data-driven decision-making and supports strategic planning initiatives within the hospital.

Future Enhancement

Future Enhancements for the Hospital Management System:

a. Enhanced User Authentication: Implement a robust user authentication system to ensure the security and privacy of patient data. This system would include features such as user registration, login/logout functionality, and password management, restricting access to authorized hospital staff only.

b. Patient Profiles: Enhance the system to support comprehensive patient profiles, including medical history, allergies, and demographic information. This would provide healthcare providers with a holistic view of each patient's health status and facilitate personalized care delivery.

c. Medical Records Management: Implement advanced features for managing electronic health records (EHR), such as secure storage, access controls, and versioning. This would streamline medical record keeping, ensure data integrity, and support compliance with healthcare regulations.

d. Appointment Scheduling Optimization: Integrate intelligent scheduling algorithms to optimize appointment booking and resource allocation. This would minimize patient wait times, maximize healthcare provider efficiency, and enhance overall patient satisfaction.

e. Telemedicine Integration: Incorporate telemedicine functionality to enable remote consultations and virtual visits between patients and healthcare providers. This would expand access to healthcare services, particularly in remote or underserved areas, and improve patient convenience.

f. Medication Management: Develop features for managing medication prescriptions, dispensing, and adherence tracking within the system. This would improve medication safety, reduce medication errors, and enhance patient medication compliance.

g. Integration with IoT Devices: Integrate with Internet of Things (IoT) devices such as wearable health monitors and smart medical devices to collect real-time patient data. This would enable proactive health monitoring, early detection of health issues, and personalized treatment recommendations.

h. Advanced Analytics and Reporting: Enhance the system with advanced analytics and reporting capabilities to derive insights from patient data, clinical outcomes, and operational performance. This would support data-driven decision-making, quality improvement initiatives, and healthcare research.

i. Multi-Language Support: Implement multi-language support to cater to diverse patient populations and healthcare providers. This would improve accessibility and inclusivity, ensuring that language barriers do not hinder effective communication and care delivery.

j. Continuous Performance Optimization: Continuously optimize system performance through regular updates, performance monitoring, and feedback-driven improvements. This would ensure that the system remains efficient, reliable, and responsive to evolving healthcare needs and technological advancements.

Summary

The Hospital Management System project is a Java-based application designed to facilitate the management of hospital operations. It comprises three core modules: GUI (Graphical User Interface), Database Connectivity, and Operations.

The GUI module is developed using the Java Swing framework, offering a user-friendly interface for various hospital management tasks. Users can input patient details, appointments, and medical records. The GUI includes buttons for adding, editing, and deleting records, as well as for viewing patient information. Additionally, it features a display area for presenting relevant data.

The Database Connectivity module enables seamless interaction with a MySQL database. Utilizing the JDBC API, it establishes connections and executes database operations such as adding new patient records, updating existing information, and removing records as required.

The Operations module serves as the backbone of the system, encompassing the business logic. It includes methods for handling patient admission, scheduling appointments, managing medical records, and other essential hospital operations. This module interacts with the Database Connectivity module to execute SQL queries and update the database accordingly.

Integration of the system is achieved through coordinated interactions between the GUI, Database Connectivity, and Operations modules. The GUI captures user inputs and triggers corresponding operations in the Operations module. Subsequently, the Operations module communicates with the Database Connectivity module to execute database operations, with results reflected back in the GUI.

Testing the Hospital Management System involves running the application and performing various actions such as adding patients, scheduling appointments, and updating medical records. The GUI provides feedback through message dialogs, indicating the success or failure of operations performed.

The Hospital Management System project delivers a robust foundation for efficiently managing hospital operations, aiding in patient care and administrative tasks.

REFRENCES

➢ **Oracle Documentation: JDBC - JavaDatabaseConnectivity** (https://docs.oracle.com/javase/tutorial/jdbc/inde x.html)
➢ **Java Swing Documentation**(https://docs.oracle.com/javase/tutorial/uiswing/)
➢ **MySQL Documentation**(https://dev.mysql.com/doc/)
➢ **Stack overflow,** https://stackoverflow.com/questions/48690990/how-to-execute-a-sql-statement-in-java-with-variables
➢ **GeeksforGeeks**. https://www.geeksforgeeks.org/java-program-for-hospital-management-system/.
➢ **Javatpoint**. "https://www.javatpoint.com/java-swing

APPENDIX A: SOURCE CODE

IDE used: Eclipse IDE

```
import javax.swing.*;

public class Project
{
  Project()
  {
       new IntroPage();
  }
  public static void main(String[] args) {
    new Project();
  }
}
```

```
import java.awt.*;
import javax.swing.*;
public class IntroPage extends JFrame
{
  JLabel welcare_hospital,logofield,background;
  JLabel Welcome;
  JButton PLogin;
  JButton Slogin;


  public IntroPage()
  {
    Welcome = new JLabel();
    welcare_hospital = new JLabel();
    PLogin = new JButton();
    Slogin = new JButton();
    ImageIcon backgroundImage = new
ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\bg.jpeg");
    background = new JLabel(backgroundImage);
    ImageIcon logo = new ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\Wellness (1).png");
    logofield = new JLabel(new
ImageIcon(logo.getImage().getScaledInstance(350, 350,
Image.SCALE_SMOOTH)));
```

```
    setVisible(true);
    setLayout(null);
    setSize(1000,600);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("WELCARE HOSPITAL");
    setLocationRelativeTo(null);
    logofield.setBounds(20, 50, 350, 350);
    add(logofield);


    Welcome.setFont(new java.awt.Font("Segoe UI", 1, 24));
    Welcome.setForeground(Color.BLACK);
    Welcome.setText("WELCOME");
    Welcome.setBounds(660,110,200,200);
    add(Welcome);

    welcare_hospital.setFont(new java.awt.Font("Segoe UI Historic", 1,
36));
    welcare_hospital.setForeground(Color.BLACK);
    welcare_hospital.setText("WELCARE HOSPITAL");
    welcare_hospital.setBounds(550,10,700,200);
    add(welcare_hospital);

    PLogin.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
    PLogin.setText("PATIENT LOGIN");
    PLogin.setForeground(Color.BLACK);
    PLogin.setBackground(new java.awt.Color(255, 255, 255));
    PLogin.setBounds(640, 250, 170, 100);
    add(PLogin);
    PLogin.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          PLoginActionPerformed(evt);
      }
    });
    Slogin.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
    Slogin.setText("STAFF LOGIN");
    Slogin.setForeground(Color.BLACK);
    Slogin.setBackground(new java.awt.Color(255, 255, 255));
    Slogin.setBounds(650, 400, 150, 100);
    add(Slogin);
    Slogin.addActionListener(new java.awt.event.ActionListener()
    {
        public void actionPerformed(java.awt.event.ActionEvent evt)
        {
            SloginActionPerformed(evt);
        }
    });

    background.setBounds(0, 0, 1000, 600);
    add(background);
  }
  private void PLoginActionPerformed(java.awt.event.ActionEvent evt)
  {
     new PatientLoginPage();
    this.dispose();
  }
  private void SloginActionPerformed(java.awt.event.ActionEvent evt)
  {
    LoginAdmin LoginAdminFrame=new LoginAdmin();
    LoginAdminFrame.setVisible(true);
    LoginAdminFrame.pack();
    LoginAdminFrame.setLocationRelativeTo(null);
    this.dispose();
  }
  public static void main(String args[])
  {
      new IntroPage();
  }
}
```

```
import java.awt.Image;
import javax.swing.*;
public class LoginAdmin extends JFrame
{
  JLabel logofield;
  JLabel LOGIN;
  JPanel Left;
  JLabel Password;
  JPanel Right;
  JLabel Username;
  JPanel jPanel1;
  JButton login;
  JPasswordField password;
  JTextField username;
  public LoginAdmin()
  {
    jPanel1 = new javax.swing.JPanel();
    Left = new javax.swing.JPanel();
    Right = new javax.swing.JPanel();
    LOGIN = new javax.swing.JLabel();
    Username = new javax.swing.JLabel();
    Password = new javax.swing.JLabel();
    username = new javax.swing.JTextField();
    password = new javax.swing.JPasswordField();
    login = new javax.swing.JButton();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("LOGIN");

    ImageIcon logo = new ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\Wellness (1).png");
    logofield = new JLabel(new
ImageIcon(logo.getImage().getScaledInstance(350, 350,
Image.SCALE_SMOOTH)));
    logofield.setBounds(20, 50, 350, 350);
    add(logofield);

    jPanel1.setBackground(new java.awt.Color(255, 255, 255));
    jPanel1.setPreferredSize(new java.awt.Dimension(800, 500));
    jPanel1.setLayout(null);

    Left.setBackground(new java.awt.Color(255, 255, 255));
    Left.setPreferredSize(new java.awt.Dimension(400, 500));

    javax.swing.GroupLayout LeftLayout = new
javax.swing.GroupLayout(Left);
    Left.setLayout(LeftLayout);
    LeftLayout.setHorizontalGroup(

LeftLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
        .addGap(0, 400, Short.MAX_VALUE)
    );
    LeftLayout.setVerticalGroup(

LeftLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADI
NG)
        .addGap(0, 500, Short.MAX_VALUE)
    );

    jPanel1.add(Left);
    Left.setBounds(0, 0, 400, 500);

    Right.setBackground(new java.awt.Color(255, 255, 255));
    Right.setMinimumSize(new java.awt.Dimension(400, 500));

    LOGIN.setBackground(new java.awt.Color(51, 0, 153));
    LOGIN.setFont(new java.awt.Font("Segoe UI", 1, 36)); // NOI18N
    LOGIN.setForeground(new java.awt.Color(0, 51, 102));
    LOGIN.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    LOGIN.setText("LOGIN");

    Username.setBackground(new java.awt.Color(0, 0, 0));
    Username.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
    Username.setText("Username");

    Password.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
    Password.setText("Password");

    username.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            usernameActionPerformed(evt);
        }
    });

    login.setBackground(new java.awt.Color(51, 0, 153));
    login.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
    login.setForeground(new java.awt.Color(255, 255, 255));
    login.setText("LOGIN");
    login.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            loginActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout RightLayout = new
javax.swing.GroupLayout(Right);
    Right.setLayout(RightLayout);
    RightLayout.setHorizontalGroup(

RightLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
        .addGroup(RightLayout.createSequentialGroup()

.addGroup(RightLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING)
                .addGroup(RightLayout.createSequentialGroup()
                    .addGap(134, 134, 134)
                    .addComponent(LOGIN,
javax.swing.GroupLayout.PREFERRED_SIZE, 123,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addGroup(RightLayout.createSequentialGroup()
                    .addGap(37, 37, 37)

.addGroup(RightLayout.createParallelGroup(javax.swing.GroupLayout.Alig
nment.LEADING, false)
                        .addComponent(Username,
javax.swing.GroupLayout.DEFAULT_SIZE, 325, Short.MAX_VALUE)
                        .addComponent(Password,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(username)
                        .addComponent(password)
                        .addGroup(RightLayout.createSequentialGroup()
                            .addGap(74, 74, 74)
                            .addComponent(login,
javax.swing.GroupLayout.PREFERRED_SIZE, 154,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addGap(22, 22, 22))))
                .addGroup(RightLayout.createSequentialGroup()
                    .addGap(103, 103, 103)
                    ))
            .addContainerGap(38, Short.MAX_VALUE))
    );
    RightLayout.setVerticalGroup(

RightLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEAD
ING)
        .addGroup(RightLayout.createSequentialGroup()
            .addGap(40, 40, 40)
            .addComponent(LOGIN)
            .addGap(50, 50, 50)
            .addComponent(Username)
            .addGap(28, 28, 28)
            .addComponent(username,
javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(Password)
            .addGap(18, 18, 18)
            .addComponent(password,
javax.swing.GroupLayout.PREFERRED_SIZE, 39,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(28, 28, 28)
            .addComponent(login,
javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
55, Short.MAX_VALUE)
            .addGap(22, 22, 22))
    );

    jPanel1.add(Right);
    Right.setBounds(400, 0, 400, 500);

    javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
    );
    layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(0, 0, Short.MAX_VALUE))
    );

    pack();
  }
  private void usernameActionPerformed(java.awt.event.ActionEvent evt)
{
}

    private void loginActionPerformed(java.awt.event.ActionEvent evt)
    {
        String Uname = username.getText();
        String Pword = password.~~getText~~();
        if(Uname.equals("") && Pword.equals(""))
        {
            JOptionPane.showMessageDialog(this,"Please Enter Username
and Password !");

        }
        else if(Pword.equals(""))
        {
            JOptionPane.showMessageDialog(this,"Please Enter Password
!");
        }
        else if(Uname.equals(""))
        {
            JOptionPane.showMessageDialog(this,"Please Enter Username
!");
        }
        else if(Uname.equals("admin") && Pword.equals("123"))
        {
            JOptionPane.showMessageDialog(this,"LOGIN SUCCESSFUL !");
            new FunctionPage();
            this.dispose();
        }
        else
        {
            JOptionPane.showMessageDialog(this,"Invalid Credentials
!");
        }

    }
}
```

```
import javax.swing.*;
import java.awt.*;
public class FunctionPage extends JFrame
{
  JLabel MENU,background,logofield;;
  JButton app,staffs,avail_staffs,back_button;
  public FunctionPage()
  {
    MENU = new JLabel();
    app =  new JButton();
    staffs = new JButton();
    avail_staffs =  new JButton();
    back_button = new JButton("⟵");
    ImageIcon backgroundImage = new
ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\bg.jpeg");
    background = new JLabel(backgroundImage);
    ImageIcon logo = new ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\Wellness (1).png");
    logofield = new JLabel(new
ImageIcon(logo.getImage().getScaledInstance(350, 350,
Image.SCALE_SMOOTH)));
```

```
    setLayout(null);
    setSize(1000,600);
    setVisible(true);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("MENU");
    setLocationRelativeTo(null);


    logofield.setBounds(20, 70, 350, 350);
    add(logofield);

    MENU.setFont(new java.awt.Font("Segoe UI", 1, 24));
    MENU.setForeground(Color.BLACK);
    MENU.setText("MENU");
    MENU.setBounds(660,0,200,100);
    add(MENU);

    back_button.setForeground(Color.BLACK);
    back_button.setBounds(10, 10, 50, 50);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);


    app.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
    app.setText("APPOINTMENTS");
    app.setForeground(Color.BLACK);
    app.setBackground(new java.awt.Color(255, 255, 255));
    app.setBounds(600, 250, 200, 60);
    app.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
          appActionPerformed(evt);
      }
  });
    add(app);

    staffs.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
    staffs.setText("EDIT STAFFS");
    staffs.setForeground(Color.BLACK);
    staffs.setBackground(new java.awt.Color(255, 255, 255));
    staffs.setBounds(600, 100, 200, 60);
    staffs.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        staffsActionPerformed(evt);
      }
  });
    add(staffs);

    avail_staffs.setFont(new java.awt.Font("Segoe UI", 1, 14));
    avail_staffs.setText(" AVAILABLE STAFFS");
    avail_staffs.setForeground(Color.BLACK);
    avail_staffs.setBackground(new java.awt.Color(255, 255, 255));
    avail_staffs.setBounds(600, 400, 200, 60);
    avail_staffs.addActionListener(new java.awt.event.ActionListener()
{
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        avail_staffsActionPerformed(evt);
      }
  });
    add(avail_staffs);

    background.setBounds(0, 0, 1000, 600);
    add(background);
  }
  private void appActionPerformed(java.awt.event.ActionEvent evt)
  {
     new Appointments();
    this.dispose();
  }

  private void staffsActionPerformed(java.awt.event.ActionEvent evt)
  {
    new View_Staffs();
    this.dispose();
  }
  private void  avail_staffsActionPerformed(java.awt.event.ActionEvent
evt)
  {
    new Avail_Staffs();
    this.dispose();
  }
  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new IntroPage();
  }
  public static void main(String args[])
  {
        new FunctionPage();
  }
}
```

```
import javax.swing.*;
import java.awt.*;
public class View_Staffs extends JFrame
{
  JButton add_staff,remove_staff,back_button,display_staffs;
  JLabel background;
  View_Staffs()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,600);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("STAFFS");
    setLocationRelativeTo(null);

    ImageIcon backgroundImage = new
ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\bg.jpeg");
    background = new JLabel(backgroundImage);

    add_staff = new JButton();
    remove_staff = new JButton();
    display_staffs = new JButton();
    back_button = new JButton("⟵");

    add_staff.setFont(new java.awt.Font("Segoe UI", 1, 14));
    add_staff.setText("ADD NEW STAFF");
    add_staff.setForeground(Color.BLACK);
    add_staff.setBackground(new java.awt.Color(255, 255, 255));
    add_staff.setBounds(100, 200, 200, 80);
    add_staff.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          addstaffActionPerformed(evt);
      }
    });
    add(add_staff);

    remove_staff.setFont(new java.awt.Font("Segoe UI", 1, 14));
    remove_staff.setText("REMOVE STAFF");
    remove_staff.setForeground(Color.BLACK);
    remove_staff.setBackground(new java.awt.Color(255, 255, 255));
    remove_staff.setBounds(400, 200, 200, 80);
    remove_staff.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          removestaffActionPerformed(evt);
      }
    });
    add(remove_staff);

    display_staffs.setFont(new java.awt.Font("Segoe UI", 1, 14));
    display_staffs.setText("DISPLAY STAFFS");
    display_staffs.setForeground(Color.BLACK);
    display_staffs.setBackground(new java.awt.Color(255, 255, 255));
    display_staffs.setBounds(750, 200, 200, 80);
    display_staffs.addActionListener(new
java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          displayActionPerformed(evt);
      }
    });
    add(display_staffs);

    back_button.setForeground(Color.BLACK);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);

    background.setBounds(0, 0, 1000, 600);
    add(background);
  }

  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    setVisible(false);
     new FunctionPage();
  }
  private void displayActionPerformed(java.awt.event.ActionEvent evt)
  {
         setVisible(false);
         new Display_Staff();
  }
  private void  addstaffActionPerformed(java.awt.event.ActionEvent
evt)
  {
         setVisible(false);
         new Add_Staff();
  }
  private void  removestaffActionPerformed(java.awt.event.ActionEvent
evt)
  {
         setVisible(false);
         new Remove_Staff();
  }

}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class Add_Staff extends JFrame
{
      private JLabel idLabel, nameLabel, positionLabel, ageLabel,
sexLabel,spezLabel;
      private JTextField idField, nameField, positionField, ageField,
sexField,spezField;
      private JButton submitButton,back_button;
      private Color darkBlue = new Color(24, 44, 97);
      public Add_Staff()
      {
             setTitle("Staff Registration Form");
             setSize(500, 400);
             setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
             setLocationRelativeTo(null);

             idLabel = new JLabel("Staff ID:");
             nameLabel = new JLabel("Staff Name:");
```

```
            positionLabel = new JLabel("Staff Position:");
            ageLabel = new JLabel("Age:");
            sexLabel = new JLabel("Sex:");
            spezLabel = new JLabel("Specialization(if doctor):");

            idField = new JTextField(20);
            nameField = new JTextField(20);
            positionField = new JTextField(20);
            ageField = new JTextField(50);
            sexField = new JTextField(50);
            spezField = new JTextField(50);

            submitButton = new JButton("Submit");
            submitButton.setBackground(darkBlue);
            submitButton.setForeground(Color.WHITE);
            back_button = new JButton("Back");
            back_button.setBackground(darkBlue);
            back_button.setForeground(Color.WHITE);

            submitButton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    String id = idField.getText();
                    String name = nameField.getText();
                    String position = positionField.getText();
                    String age = ageField.getText();
                    String sex = sexField.getText();
                    String spez = spezField.getText();

                    saveToDatabase(id, name, position, age,
sex,spez);

                    idField.setText("");
                    nameField.setText("");
                    positionField.setText("");
                    ageField.setText("");
                    sexField.setText("");
                    spezField.setText("");
                }
            });
            back_button.addActionListener(new ActionListener()
            {
              public void actionPerformed(java.awt.event.ActionEvent
evt)
              {
                  backActionPerformed(evt);
              }
            });
```

```
              JPanel panel = new JPanel(new GridLayout(8, 2));
              panel.add(idLabel);
              panel.add(idField);
              panel.add(nameLabel);
              panel.add(nameField);
              panel.add(positionLabel);
              panel.add(positionField);
              panel.add(ageLabel);
              panel.add(ageField);
              panel.add(sexLabel);
              panel.add(sexField);
              panel.add(spezLabel);
              panel.add(spezField);
              panel.add(new JLabel());
              panel.add(new JLabel());
              panel.add(submitButton);
              panel.add(back_button);

              add(panel);
            setVisible(true);
       }
          private void saveToDatabase(String id, String name, String 
position, String age, String sex,String spez) {
              String url = "jdbc:mysql://localhost:3306/microproject";
              String username = "root";
              String password = "jacob";

              try (Connection conn = DriverManager.getConnection(url, 
username, password)) {
                 if(position.equals("Doctor")) {
                      String query1 = "INSERT INTO staffs (staff_id, 
staff_name, staff_position, age, sex) VALUES (?, ?, ?, ?, ?)";
                      String query2 = "INSERT INTO doctor 
(staff_id,doctor_name,specialization) VALUES (?,?,?)" ;
                      String query3 = "INSERT INTO doctors_available 
(staff_id,doctor_name,specialization) VALUES (?,?,?)" ;
                       PreparedStatement statement1 = 
conn.prepareStatement(query1);
                       statement1.setString(1, id);
                       statement1.setString(2, name);
                       statement1.setString(3, position);
                       statement1.setString(4, age);
                       statement1.setString(5, sex);

                       PreparedStatement statement2 = 
conn.prepareStatement(query2);
```

```
                              statement2.setString(1, id);
                              statement2.setString(2, name);
                              statement2.setString(3, spez);

                              PreparedStatement statement3 =
conn.prepareStatement(query3);
                              statement3.setString(1, id);
                              statement3.setString(2, name);
                              statement3.setString(3, spez);


                              int rowsInserted1 = statement1.executeUpdate();
                              int rowsInserted2 = statement2.executeUpdate();
                              statement3.executeUpdate();
                              if (rowsInserted1 > 0 && rowsInserted2 > 0) {
                                  JOptionPane.showMessageDialog(null, "Data
saved successfully!");
                              }
                     }
                     else
                     {
                      String query = "INSERT INTO staffs (staff_id,
staff_name, staff_position, age, sex) VALUES (?, ?, ?, ?, ?)";
                      PreparedStatement statement =
conn.prepareStatement(query);
                      statement.setString(1, id);
                      statement.setString(2, name);
                      statement.setString(3, position);
                      statement.setString(4, age);
                      statement.setString(5, sex);

                      int rowsInserted = statement.executeUpdate();
                      if (rowsInserted > 0) {
                          JOptionPane.showMessageDialog(null, "Data saved
successfully!");
                      }
                     }
                 } catch (SQLException ex) {
                      ex.printStackTrace();
                      JOptionPane.showMessageDialog(null, "Error: Failed to
save data to database!");
                 }
            }
            private void backActionPerformed(java.awt.event.ActionEvent
evt)
            {
              this.dispose();
```

```
          new View_Staffs();
        }
        public static void main(String[] args) {
          new Add_Staff();
        }
}
```

```
import java.awt.*;
import javax.swing.*;
import java.sql.*;
import net.proteanit.sql.DbUtils;


public class Display_Staff extends JFrame
{
  JTable table;
  JButton back_button;
  Display_Staff()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,700);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("STAFFS");
    setLocationRelativeTo(null);

    back_button = new JButton("⬅");

    table = new JTable();
      String url = "jdbc:mysql://localhost:3306/microproject";
    String username = "root";
    String password = "jacob";

    try(Connection conn = DriverManager.getConnection(url, username,
password))
    {
      String query = "select * from staffs ";
      Statement statement = conn.createStatement();
      ResultSet rs = statement.executeQuery(query);
      table.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch (Exception e){
      e.printStackTrace();
    }

    JScrollPane jsp = new JScrollPane(table);
    jsp.setBounds(100,100,700,600);
    add(jsp);

    back_button.setForeground(Color.BLACK);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);

  }
  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new View_Staffs();
  }
  public static void main(String args[])
  {
       new Display_Staff();
  }

}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Remove_Staff extends JFrame
{
      private JLabel idLabel, nameLabel;
      private JTextField idField, nameField;
      private JButton submitButton,back_button;
      private Color darkBlue = new Color(24, 44, 97);

      public Remove_Staff()
      {

         setTitle("Remove Staff");
```

```
        setSize(500, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        idLabel = new JLabel("Staff ID:");
        nameLabel = new JLabel("Staff Name:");

        idField = new JTextField(20);
        nameField = new JTextField(20);

        submitButton = new JButton("Delete");
        submitButton.setBackground(darkBlue);
        submitButton.setForeground(Color.WHITE);
        back_button = new JButton("Back");
        back_button.setBackground(darkBlue);
        back_button.setForeground(Color.WHITE);

        submitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String id = idField.getText();
                String name = nameField.getText();

                saveToDatabase(id, name);

                idField.setText("");
                nameField.setText("");
            }
        });

        back_button.addActionListener(new ActionListener()
        {
          public void actionPerformed(java.awt.event.ActionEvent evt)
          {
              backActionPerformed(evt);
          }
        });

        JPanel panel = new JPanel(new GridLayout(4, 2));
        panel.add(idLabel);
        panel.add(idField);
        panel.add(nameLabel);
        panel.add(nameField);
        panel.add(new JLabel());
        panel.add(new JLabel());
        panel.add(submitButton);
        panel.add(back_button);
        add(panel);
```

```
           setVisible(true);

      }

       private void saveToDatabase(String id, String name)
       {
            String url = "jdbc:mysql://localhost:3306/microproject";
           String username = "root";
           String password = "jacob";

           try (Connection conn = DriverManager.getConnection(url,
username, password))
           {
            String query = "DELETE FROM staffs WHERE staff_id = ? AND
staff_name = ?";
            String query2 = "DELETE FROM doctor WHERE staff_id = ? AND
doctor_name = ?";
            String query3 = "DELETE FROM doctors_available WHERE
staff_id = ? AND  doctor_name = ?";
            PreparedStatement statement = conn.prepareStatement(query);
               statement.setString(1, id);
               statement.setString(2, name);

            PreparedStatement statement2 =
conn.prepareStatement(query2);
               statement2.setString(1, id);
               statement2.setString(2, name);

            PreparedStatement statement3 =
conn.prepareStatement(query3);
               statement3.setString(1, id);
               statement3.setString(2, name);
               int rowsDeleted2 = statement2.executeUpdate();
               int rowsDeleted = statement.executeUpdate();
               statement3.executeUpdate();
                  if (rowsDeleted> 0 && rowsDeleted2>0 )
                  {
                      JOptionPane.showMessageDialog(null, "Staff
removed successfully!");
                  }
           }
           catch (SQLException ex)
           {
               ex.printStackTrace();
               JOptionPane.showMessageDialog(null, "Error: Failed to
delete data from database!");
           }
       }
        private void backActionPerformed(java.awt.event.ActionEvent
evt)
        {
          this.dispose();
          new View_Staffs();
        }
         public static void main(String[] args)
         {
              new Remove_Staff();
         }
}
```

```
import javax.swing.*;
import java.awt.*;


public class Appointments extends JFrame
{
  JButton app_view,app_add,app_remove,back_button;
  JLabel background;
  Appointments()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,600);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("APPOINTMENTS");
    setLocationRelativeTo(null);

    ImageIcon backgroundImage = new
ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\bg.jpeg");
    background = new JLabel(backgroundImage);

    app_view =  new JButton();
    app_add = new JButton();
    app_remove = new JButton();
    back_button = new JButton("⟵");

    app_view.setFont(new java.awt.Font("Segoe UI", 1, 14));
    app_view.setText("APPOINTMENT SCHEDULE");
    app_view.setForeground(Color.BLACK);
    app_view.setBackground(new java.awt.Color(255, 255, 255));
    app_view.setBounds(100, 200, 220, 80);
    app_view.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          appviewActionPerformed(evt);
      }
    });
    add(app_view);

    app_add.setFont(new java.awt.Font("Segoe UI", 1, 14));
    app_add.setText(" ADD APPOINTMENT ");
    app_add.setForeground(Color.BLACK);
    app_add.setBackground(new java.awt.Color(255, 255, 255));
    app_add.setBounds(400, 200, 200, 80);
    app_add.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          appaddActionPerformed(evt);
      }
    });
    add(app_add);

    app_remove.setFont(new java.awt.Font("Segoe UI", 1, 14));
    app_remove.setText("REMOVE APPOINTMENT");
    app_remove.setForeground(Color.BLACK);
    app_remove.setBackground(new java.awt.Color(255, 255, 255));
    app_remove.setBounds(700, 200, 210, 80);
    app_remove.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          appremoveActionPerformed(evt);
      }
    });
    add(app_remove);

    back_button.setText("←");
    back_button.setForeground(Color.BLACK);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);

    background.setBounds(0, 0, 1000, 600);
    add(background);


  }
  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new FunctionPage();
  }
  private void appviewActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new App_View();
  }
  private void appaddActionPerformed(java.awt.event.ActionEvent evt)
  {
         new SignUpForm();
  }
  private void appremoveActionPerformed(java.awt.event.ActionEvent
evt)
  {
         this.dispose();
         new Remove_App();
  }

}
```

```
import java.awt.*;
import javax.swing.*;
import java.sql.*;
import net.proteanit.sql.DbUtils;


public class App_View extends JFrame
{
  JTable table;
  JButton back_button;
  App_View()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,700);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("APPOINTMENTS SCHEDULE");
    setLocationRelativeTo(null);

    back_button = new JButton("←");

    table = new JTable();
      String url = "jdbc:mysql://localhost:3306/microproject";
    String username = "root";
    String password = "jacob";

    try(Connection conn = DriverManager.getConnection(url, username,
password))
    {
      String query = "select * from appointments ";
      Statement statement = conn.createStatement();
      ResultSet rs = statement.executeQuery(query);
      table.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch (Exception e){
      e.printStackTrace();
    }

    JScrollPane jsp = new JScrollPane(table);
    jsp.setBounds(100,100,700,600);
    add(jsp);

    back_button.setForeground(Color.BLACK);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);
  }
  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new Appointments();
  }
  public static void main(String args[])
  {
        new App_View();
  }

}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

public class Remove_App extends JFrame
{
      private JLabel idLabel, appnoLabel;
      private JTextField idField, appnoField;
      private JButton submitButton,back_button;
      private Color darkBlue = new Color(24, 44, 97);

      public Remove_App()
      {

         setTitle("Remove Patients");
         setSize(500, 400);
         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         setLocationRelativeTo(null);

         idLabel = new JLabel("Patient ID:");
         appnoLabel = new JLabel("Appointment Number:");

         idField = new JTextField(20);
         appnoField = new JTextField(20);

         submitButton = new JButton("Delete");
         submitButton.setBackground(darkBlue);
         submitButton.setForeground(Color.WHITE);
         back_button = new JButton("Back");
         back_button.setBackground(darkBlue);
```

```
        back_button.setForeground(Color.WHITE);

        submitButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String id = idField.getText();
                String name = appnoField.getText();

                saveToDatabase(id, name);

                idField.setText("");
                appnoField.setText("");
            }
        });

        back_button.addActionListener(new ActionListener()
        {
          public void actionPerformed(java.awt.event.ActionEvent evt)
          {
              backActionPerformed(evt);
          }
        });

        JPanel panel = new JPanel(new GridLayout(5, 2));
        panel.add(idLabel);
        panel.add(idField);
        panel.add(appnoLabel);
        panel.add(appnoField);
        panel.add(new JLabel());
        panel.add(new JLabel());
        panel.add(new JLabel());
        panel.add(new JLabel());
        panel.add(submitButton);
        panel.add(back_button);
        add(panel);
           setVisible(true);

      }

       private void saveToDatabase(String id, String appno)
       {
             String url = "jdbc:mysql://localhost:3306/microproject";
            String username = "root";
            String password = "jacob";

            try (Connection conn = DriverManager.getConnection(url,
username, password))
            {
```

```
            String query = "DELETE FROM appointments WHERE patientID =
? AND  App_No = ?";
            PreparedStatement statement = conn.prepareStatement(query);
               statement.setString(1, id);
               statement.setString(2, appno);

               int rowsDeleted = statement.executeUpdate();
                  if (rowsDeleted> 0)
                  {
                      JOptionPane.showMessageDialog(null, "Patient
removed successfully!");
                  }
           }
           catch (SQLException ex)
           {
               ex.printStackTrace();
               JOptionPane.showMessageDialog(null, "Error: Failed to
delete data from database!");
           }
       }
        private void backActionPerformed(java.awt.event.ActionEvent
evt)
        {
          this.dispose();
          new Appointments();
        }
          public static void main(String[] args) {
              new Remove_App();
          }
}
```

```
import javax.swing.*;
import java.awt.*;
public class Avail_Staffs extends JFrame
{
  JButton doct_avail,mark_leave,back_button,on_leave;
  JLabel background;
  Avail_Staffs()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,600);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("AVAILABLE STAFFS");
    setLocationRelativeTo(null);

    ImageIcon backgroundImage = new
ImageIcon("C:\\Users\\JACOP\\eclipse-
workspace\\HospitalMS\\src\\icons\\bg.jpeg");
    background = new JLabel(backgroundImage);

    doct_avail = new JButton();
    mark_leave = new JButton();
    on_leave = new JButton();
    back_button = new JButton("⟵");

    doct_avail.setFont(new java.awt.Font("Segoe UI", 1, 14));
    doct_avail.setText("DOCTORS AVAILABLE");
    doct_avail.setForeground(Color.BLACK);
    doct_avail.setBackground(new java.awt.Color(255, 255, 255));
    doct_avail.setBounds(100, 200, 200, 80);
    doct_avail.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          doctavailActionPerformed(evt);
      }
    });
    add(doct_avail);

    mark_leave.setFont(new java.awt.Font("Segoe UI", 1, 14));
    mark_leave.setText("MARK LEAVE");
    mark_leave.setForeground(Color.BLACK);
    mark_leave.setBackground(new java.awt.Color(255, 255, 255));
    mark_leave.setBounds(400, 200, 200, 80);
    mark_leave.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          markleaveActionPerformed(evt);
      }
    });
    add(mark_leave);

    on_leave.setFont(new java.awt.Font("Segoe UI", 1, 14));
    on_leave.setText("VIEW STAFFS ON LEAVE");
    on_leave.setForeground(Color.BLACK);
    on_leave.setBackground(new java.awt.Color(255, 255, 255));
    on_leave.setBounds(700, 200, 230, 80);
    on_leave.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          onleaveActionPerformed(evt);
      }
    });
    add(on_leave);


    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setForeground(Color.BLACK);
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);

    background.setBounds(0, 0, 1000, 600);
    add(background);
  }

  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    setVisible(false);
     new FunctionPage();
  }
  private void onleaveActionPerformed(java.awt.event.ActionEvent evt)
  {
    setVisible(false);
    new On_Leave();

  }
  private void markleaveActionPerformed(java.awt.event.ActionEvent
evt)
  {
    setVisible(false);
    new Mark_Leave();
  }
  private void doctavailActionPerformed(java.awt.event.ActionEvent
evt)
  {
    setVisible(false);
    new Doctors_Available();

  }
}
```

```
import java.awt.*;
import javax.swing.*;
import java.sql.*;
import net.proteanit.sql.DbUtils;


public class On_Leave extends JFrame
{
  JTable table;
  JButton back_button;
  On_Leave()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,700);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("STAFFS ON LEAVE");
    setLocationRelativeTo(null);

    back_button = new JButton("⟵");

    table = new JTable();
      String url = "jdbc:mysql://localhost:3306/microproject";
    String username = "root";
    String password = "jacob";

    try(Connection conn = DriverManager.getConnection(url, username,
password))
    {
      String query = "select * from on_leave ";
      Statement statement = conn.createStatement();
      ResultSet rs = statement.executeQuery(query);
      table.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch (Exception e){
      e.printStackTrace();
    }

    JScrollPane jsp = new JScrollPane(table);
    jsp.setBounds(100,100,700,600);
    add(jsp);

    back_button.setForeground(Color.BLACK);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);

  }
  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new Avail_Staffs();
  }
  public static void main(String args[])
  {
       new On_Leave();
  }

}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.toedter.calendar.JDateChooser;
```

```
public class Mark_Leave extends JFrame{
	private JLabel idLabel, nameLabel, startLabel,endLabel;
	private JTextField idField, nameField,startField,endField;
	private JButton submitButton,back_button;
	private JDateChooser startdate,enddate;
	private Color darkBlue = new Color(24, 44, 97);
	public Mark_Leave()
	{
		setTitle("Leave Form");
		setSize(500, 400);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setLocationRelativeTo(null);

		idLabel = new JLabel("Staff ID:");
		nameLabel = new JLabel("Staff Name:");
		startLabel = new JLabel("From:");
		endLabel = new JLabel("Till:");
		startdate = new JDateChooser();
		enddate = new JDateChooser();

		idField = new JTextField(20);
		nameField = new JTextField(20);


		submitButton = new JButton("Submit");
		submitButton.setBackground(darkBlue);
		submitButton.setForeground(Color.WHITE);
		back_button = new JButton("Back");
		back_button.setBackground(darkBlue);
		back_button.setForeground(Color.WHITE);

		submitButton.addActionListener(new ActionListener() {
		    public void actionPerformed(ActionEvent e) {
			String id = idField.getText();
			String name = nameField.getText();
			String start = ((JTextField)
startdate.getDateEditor().getUiComponent()).getText();
			String end = ((JTextField)
enddate.getDateEditor().getUiComponent()).getText();

			saveToDatabase(id, name,start,end);

			idField.setText("");
			nameField.setText("");
			startdate.setDate(null);
			enddate.setDate(null);
```

```
                  }
              });
              back_button.addActionListener(new ActionListener()
              {
                public void actionPerformed(java.awt.event.ActionEvent
evt)
                {
                    backActionPerformed(evt);
                }
              });

              JPanel panel = new JPanel(new GridLayout(8, 2));
              panel.add(idLabel);
              panel.add(idField);
              panel.add(nameLabel);
              panel.add(nameField);
              panel.add(startLabel);
              panel.add(startdate);
              panel.add(endLabel);
              panel.add(enddate);
              panel.add(new JLabel());
              panel.add(new JLabel());
              panel.add(new JLabel());
              panel.add(new JLabel());
              panel.add(new JLabel());
              panel.add(new JLabel());
              panel.add(submitButton);
              panel.add(back_button);

              add(panel);
           setVisible(true);
       }
           private void saveToDatabase(String id, String name, String
start, String end) {
              String url = "jdbc:mysql://localhost:3306/microproject";
              String username = "root";
              String password = "jacob";

              try (Connection conn = DriverManager.getConnection(url,
username, password)) {
                  String query1 = "INSERT INTO on_leave ( staff_id,
staff_name, startdate, enddate) VALUES (?, ?, ?, ?)";
                  String query2 = "DELETE FROM  doctors_available WHERE
staff_id = ? OR doctor_name = ? ";
                  PreparedStatement statement1 =
conn.prepareStatement(query1);
                  statement1.setString(1, id);
```

```
                 statement1.setString(2, name);
                 statement1.setString(3, start);
                 statement1.setString(4, end);

                 PreparedStatement statement2 =
conn.prepareStatement(query2);
                 statement2.setString(1, id);
                 statement2.setString(2, name);

                 int rowsInserted = statement1.executeUpdate();
                 int rowsInserted2 = statement2.executeUpdate();
                 if (rowsInserted > 0) {
                     JOptionPane.showMessageDialog(null, "Leave
Applied Successfully");
                 }
                 statement2.executeUpdate();

             }
             catch (SQLException ex) {
                 ex.printStackTrace();
                 JOptionPane.showMessageDialog(null, "Error!");
             }
         }
         private void backActionPerformed(java.awt.event.ActionEvent
evt)
         {
           this.dispose();
           new Avail_Staffs();
         }
         public static void main(String[] args) {
           new Mark_Leave();
         }

}
```

```
import java.awt.*;
import javax.swing.*;
import java.sql.*;
import net.proteanit.sql.DbUtils;


public class Doctors_Available extends JFrame
{
  JTable table;
  JButton back_button;
  Doctors_Available()
  {
    setLayout(null);
    setVisible(true);
    setSize(1000,700);

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
    setTitle("DOCTORS");
    setLocationRelativeTo(null);

    back_button = new JButton("⟵");
    back_button.setForeground(Color.BLACK);
    back_button.setBackground(new java.awt.Color(255, 255, 255));
    back_button.setBounds(10, 10, 50, 50);
    back_button.addActionListener(new java.awt.event.ActionListener()
    {
      public void actionPerformed(java.awt.event.ActionEvent evt)
      {
          backActionPerformed(evt);
      }
    });
    add(back_button);



    table = new JTable();
     String url = "jdbc:mysql://localhost:3306/microproject";
    String username = "root";
    String password = "jacob";

    try(Connection conn = DriverManager.getConnection(url, username,
password))
    {
     String query = "select * from  doctors_available ";
     Statement statement = conn.createStatement();
     ResultSet rs = statement.executeQuery(query);
     table.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch (Exception e){
      e.printStackTrace();
    }

    JScrollPane jsp = new JScrollPane(table);
    jsp.setBounds(100,100,700,600);
    add(jsp);
  }
  private void backActionPerformed(java.awt.event.ActionEvent evt)
  {
    this.dispose();
    new Avail_Staffs();
  }
  public static void main(String args[])
  {
       new Doctors_Available();
  }

}
```

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import com.toedter.calendar.JDateChooser;

public class HospitalManagementSystem {

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            new PatientLoginPage();
        });
    }
}

class PatientLoginPage extends JFrame implements ActionListener {

    private JTextField usernameField;
    private JPasswordField passwordField;
    private Color darkBlue = new Color(24, 44, 97);

    public PatientLoginPage() {
        setTitle("WELCARE HOSPITAL - Patient Login");
        setSize(400, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        JPanel loginPanel = new JPanel(new GridLayout(4, 2, 10, 10));
        loginPanel.setBackground(Color.WHITE);

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setForeground(darkBlue);
        usernameField = new JTextField(20);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setForeground(darkBlue);
        passwordField = new JPasswordField(20);

        JButton loginButton = new JButton("Login");
        loginButton.addActionListener(this);
        loginButton.setBackground(darkBlue);
        loginButton.setForeground(Color.WHITE);

        JButton signUpButton = new JButton("Sign Up");
        signUpButton.addActionListener(this);
        signUpButton.setBackground(darkBlue);
        signUpButton.setForeground(Color.WHITE);

        loginPanel.add(usernameLabel);
        loginPanel.add(usernameField);
        loginPanel.add(passwordLabel);
        loginPanel.add(passwordField);
        loginPanel.add(new JLabel());
        loginPanel.add(loginButton);
        loginPanel.add(new JLabel());
        loginPanel.add(signUpButton);

        getContentPane().setBackground(darkBlue);
        add(loginPanel, BorderLayout.CENTER);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
         String usernamelg = usernameField.getText();
        String password = new String(passwordField.getPassword());

        if (e.getActionCommand().equals("Login")) {
            if (isValidLogin(usernamelg , password)) {
                dispose();
                new PatientDashboardAfterLogin(usernamelg );
            } else {
                JOptionPane.showMessageDialog(this, "Invalid Username
or Password", "Error", JOptionPane.ERROR_MESSAGE);
```

```
            }
        } else if (e.getActionCommand().equals("Sign Up")) {
            dispose(); // Close the login page
            new SignUpForm().setVisible(true);
        }
    }

    private boolean isValidLogin(String username, String password) {
        String dbUrl = "jdbc:mysql://localhost:3306/microproject"; //
Change to your database URL
        String dbUser = "root"; // Your database username
        String dbPassword = "jacob"; // Your database password

        Connection conn = null;
        PreparedStatement ps = null;
        ResultSet rs = null;

        try {
            // Establish a connection
            conn = DriverManager.getConnection(dbUrl, dbUser,
dbPassword);
            // Prepare a query to check if the username and password
exist
            String query = "SELECT * FROM patients WHERE username = ?
AND password = ?";
            ps = conn.prepareStatement(query);
            ps.setString(1, username);
            ps.setString(2, password);
            rs = ps.executeQuery();

            // Check if a record exists with the given username and
password
            if (rs.next()) {
                return true; // User found, login is valid
            } else {
                return false; // No matching user found
            }
        }
        catch (SQLException ex) {
            ex.printStackTrace();
            return false;
        }
    }
}

class SignUpForm extends JFrame implements ActionListener {
```

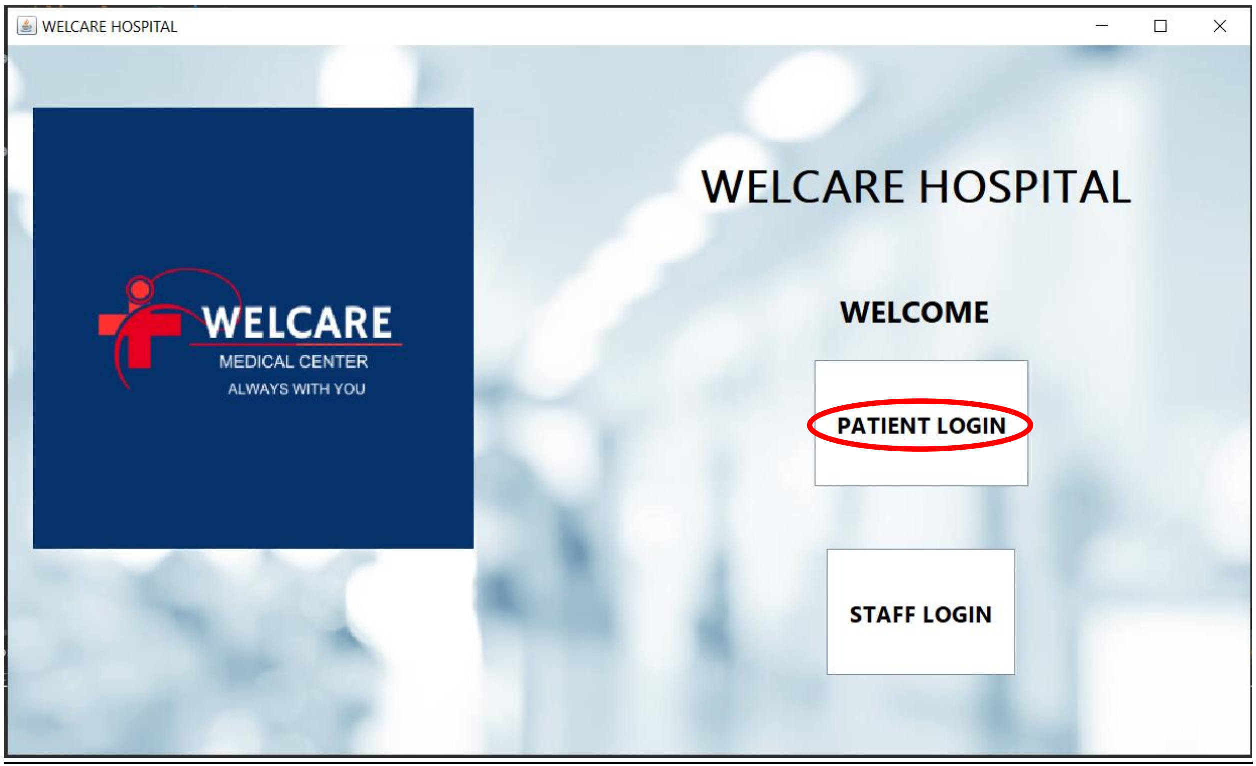
```java
    private JTextField fullNameField;
    private JTextField ageField;
    private JTextField genderField;
    private JTextField contactField;
    private JTextField addressField;
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JPasswordField confirmPasswordField;
    private Color darkBlue = new Color(24, 44, 97);

    public SignUpForm() {
      setVisible(true);
        setTitle("Patient Sign Up");
        setSize(400, 250); // Reduced height to fit fewer components
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);


        JLabel fullNameLabel = new JLabel("Full Name:");
        fullNameLabel.setForeground(darkBlue);
        fullNameField = new JTextField(20);

        JLabel ageLabel = new JLabel("Age:");
        ageLabel.setForeground(darkBlue);
        ageField = new JTextField(20);

        JLabel genderLabel = new JLabel("Gender:");
        genderLabel.setForeground(darkBlue);
        genderField = new JTextField(20);

        JLabel contactLabel = new JLabel("Contact Number:");
        contactLabel.setForeground(darkBlue);
        contactField = new JTextField(20);

        JLabel addressLabel = new JLabel("Address:");
        addressLabel.setForeground(darkBlue);
        addressField = new JTextField(20);

        JPanel signUpPanel = new JPanel(new GridLayout(5, 2, 10, 10));
        signUpPanel.setBackground(Color.WHITE);

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setForeground(darkBlue);
        usernameField = new JTextField(20);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setForeground(darkBlue);
        passwordField = new JPasswordField(20);

        JLabel confirmPasswordLabel = new JLabel("Confirm Password:");
        confirmPasswordLabel.setForeground(darkBlue);
        confirmPasswordField = new JPasswordField(20);

        JButton signUpButton = new JButton("Sign Up");
        signUpButton.addActionListener(this);
        signUpButton.setBackground(darkBlue);
        signUpButton.setForeground(Color.WHITE);

        signUpPanel.add(fullNameLabel);
        signUpPanel.add(fullNameField);
        signUpPanel.add(ageLabel);
        signUpPanel.add(ageField);
        signUpPanel.add(genderLabel);
        signUpPanel.add(genderField);
        signUpPanel.add(contactLabel);
        signUpPanel.add(contactField);
        signUpPanel.add(addressLabel);
        signUpPanel.add(addressField);
        signUpPanel.add(usernameLabel);
        signUpPanel.add(usernameField);
        signUpPanel.add(passwordLabel);
        signUpPanel.add(passwordField);
        signUpPanel.add(confirmPasswordLabel);
        signUpPanel.add(confirmPasswordField);
        signUpPanel.add(new JLabel());
        signUpPanel.add(signUpButton);

        getContentPane().setBackground(darkBlue);
        add(signUpPanel, BorderLayout.CENTER);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Sign Up")) {
           String fullname = fullNameField.getText();
           String age = ageField.getText();
           String gender = genderField.getText();
           String contact  = contactField.getText();
           String address = addressField.getText();
            String usernamesp = usernameField.getText();
            String password = new String(passwordField.getPassword());
            String confirmPassword = new
String(confirmPasswordField.getPassword());

            if (!password.equals(confirmPassword)) {
                JOptionPane.showMessageDialog(this, "Passwords do not
match!", "Error", JOptionPane.ERROR_MESSAGE);
            } else {

      saveToDatabase(fullname,age,gender,contact,address,usernamesp,
password);
            }
        }
    }
      private void saveToDatabase(String fullname,String age,String
gender,String contact,String address,String usernamesp,String
password)
      {
            String dburl = "jdbc:mysql://localhost:3306/microproject";
           String dbusername = "root";
           String dbpassword = "jacob";

           try (Connection conn = DriverManager.getConnection(dburl,
dbusername, dbpassword))
           {
            String query = "INSERT INTO patients (patient_name,
age,gender,address,contact_no,username,password) VALUES (?,
?,?,?,?,?,?)";;
            PreparedStatement statement = conn.prepareStatement(query);
            statement.setString(1,fullname);
               statement.setString(2, age);
               statement.setString(3, gender);
               statement.setString(4, contact);
               statement.setString(5, address);
               statement.setString(6, usernamesp);
               statement.setString(7, password);

                  int rowsInserted = statement.executeUpdate();
                  if (rowsInserted > 0) {
                      JOptionPane.showMessageDialog(null, "Data saved
successfully!");
                      dispose(); // Close the sign-up form
                      new PatientDashboardAfterLogin(usernamesp);
                  }
              } catch (SQLException ex) {
                  ex.printStackTrace();
                  JOptionPane.showMessageDialog(null, "Error: Failed to
save data to database!");
              }
           }

      }
```

```
class PatientDashboardAfterLogin extends JFrame implements
ActionListener {

    private String username;
    private Color darkBlue = new Color(24, 44, 97);

    public PatientDashboardAfterLogin(String username) {
        this.username = username;
        setTitle("Patient Dashboard ");
        setSize(500, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel dashboardPanel = new JPanel(new GridLayout(1, 3, 10,
10));
        dashboardPanel.setBackground(Color.WHITE);

        JButton appointmentBookingButton = createButton("Appointment
Booking");
        JButton editAppointmentButton = createButton("Edit
Appointment");
        JButton logoutButton = createButton("Logout");

        dashboardPanel.add(appointmentBookingButton);
        dashboardPanel.add(editAppointmentButton);
        dashboardPanel.add(logoutButton);

        getContentPane().setBackground(darkBlue);
        add(dashboardPanel);
        setVisible(true);
    }

    private JButton createButton(String text) {
        JButton button = new JButton(text);
        button.addActionListener(this);
        button.setBackground(darkBlue);
        button.setForeground(Color.WHITE);
        return button;
    }

    public void actionPerformed(ActionEvent e) {
        String actionCommand = e.getActionCommand();

        if (actionCommand.equals("Appointment Booking")) {
            AppointmentBookingForm appointmentForm = new
AppointmentBookingForm();
```

```
            appointmentForm.setVisible(true);
        } else if (actionCommand.equals("Edit Appointment")) {
            EditAppointmentForm editForm = new EditAppointmentForm();
            editForm.setVisible(true);
        } else if (actionCommand.equals("Logout")) {
            int option = JOptionPane.showConfirmDialog(this, "Are you 
sure you want to logout?", "Logout", JOptionPane.YES_NO_OPTION);
            if (option == JOptionPane.YES_OPTION) {
                dispose();
                new PatientLoginPage().setVisible(true);
            }
        }
    }
}

class AppointmentBookingForm extends JFrame implements ActionListener 
{

    private JTextField patientNameField;
    private JComboBox<String> doctorComboBox;
    private JTextField appointmentDateField;
    private JTextField appointmentTimeField;
    private JComboBox<String> typeComboBox;
    private JDateChooser appdate;
    private Color darkBlue = new Color(24, 44, 97);

    public AppointmentBookingForm() {
        setTitle("Appointment Booking");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel appointmentPanel = new JPanel(new GridLayout(6, 2, 10, 
10));
        appointmentPanel.setBackground(Color.WHITE);

        appointmentPanel.add(new JLabel("Patient Name:"));
        patientNameField = new JTextField(20);
        appointmentPanel.add(patientNameField);

        appointmentPanel.add(new JLabel("Select Doctor:"));
        String[] doctors = {
            "Dr. Rajashweri (Pediatrician)",
            "Dr. Andrews Peter (Orthopedic S)",
            "Dr. Hari Sundar (Oncologist)",
            "Dr. James Smith (Cardiologist)",
        };
        doctorComboBox = new JComboBox<>(doctors);
        appointmentPanel.add(doctorComboBox);

        appointmentPanel.add(new JLabel("Appointment Date"));
        appdate = new JDateChooser();
        appointmentPanel.add(appdate);

        appointmentPanel.add(new JLabel("Appointment Time (HH:mm):"));
        appointmentTimeField = new JTextField(20);
        appointmentPanel.add(appointmentTimeField);

        appointmentPanel.add(new JLabel("Appointment Type:"));
        String[] types = {"General Checkup", "Specialist 
Consultation", "Procedure", "Other"};
        typeComboBox = new JComboBox<>(types);
        appointmentPanel.add(typeComboBox);

        JButton bookButton = new JButton("Book Appointment");
        bookButton.addActionListener(this);
        bookButton.setBackground(darkBlue);
        bookButton.setForeground(Color.WHITE);

        appointmentPanel.add(new JLabel());
        appointmentPanel.add(bookButton);

        getContentPane().setBackground(darkBlue);
        add(appointmentPanel);
    }

    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Book Appointment")) {
            String patientName = patientNameField.getText();
            String selectedDoctor = (String) 
doctorComboBox.getSelectedItem();
            String date = ((JTextField) 
appdate.getDateEditor().getUiComponent()).getText();
            String appointmentTime = appointmentTimeField.getText();
            String appointmentType = (String) 
typeComboBox.getSelectedItem();


saveToDatabase(patientName,selectedDoctor,date,appointmentTime,appoint
mentType);

            patientNameField.setText("");
            appdate.setDate(null);
            appointmentTimeField.setText("");
        }
    }
    private void saveToDatabase(String patientName, String
selectedDoctor, String date, String appointmentTime,String
appointmentType) {
        String url = "jdbc:mysql://localhost:3306/microproject";
        String username = "root";
        String password = "jacob";
        PreparedStatement ps = null;
        ResultSet rs = null;
        try (Connection conn = DriverManager.getConnection(url,
username, password)) {
           String id = null;
            String query = "SELECT patient_id FROM patients WHERE
patient_name = ?";
            ps = conn.prepareStatement(query);
            ps.setString(1, patientName);
            rs = ps.executeQuery();

            if (rs.next()) {
                id = rs.getString("patient_id");
            }
            String query2 = "INSERT INTO appointments (patient_name,
doctor, date, time,patientID, app_type) VALUES (?, ?, ?, ?,?, ?)";
            PreparedStatement statement =
conn.prepareStatement(query2);
            statement.setString(1, patientName);
            statement.setString(2, selectedDoctor);
            statement.setString(3, date);
            statement.setString(4, appointmentTime);
            statement.setString(5,id);
            statement.setString(6, appointmentType);

            int rowsInserted = statement.executeUpdate();
            if (rowsInserted > 0) {
                JOptionPane.showMessageDialog(null, "Appointment
booked successfully!");
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error: Failed to save
data to database!");
        }
    }
}

class EditAppointmentForm extends JFrame implements ActionListener {
    private JTextField appointmentDateField;
    private JTextField appointmentTimeField;
    private JTextField patientNameField2;
    private JComboBox<String> typeComboBox;
    private JDateChooser appdate;
    private Color darkBlue = new Color(24, 44, 97);

    public EditAppointmentForm() {
        setTitle("Edit Appointment");
        setSize(400, 250);
        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel editPanel = new JPanel(new GridLayout(6, 2, 10, 10));
        editPanel.setBackground(Color.WHITE);

        editPanel.add(new JLabel("Patient Name:"));
        patientNameField2 = new JTextField(20);
        editPanel.add(patientNameField2);

        editPanel.add(new JLabel("Appointment Date"));
        appdate = new JDateChooser();
        editPanel.add(appdate);

        editPanel.add(new JLabel("Appointment Time (HH:mm):"));
        appointmentTimeField = new JTextField(20);
        editPanel.add(appointmentTimeField);

        editPanel.add(new JLabel("Appointment Type:"));
        String[] types = {"General Checkup", "Specialist
Consultation", "Procedure", "Other"};
        typeComboBox = new JComboBox<>(types);
        editPanel.add(typeComboBox);

        JButton saveButton = new JButton("Save Changes");
        saveButton.addActionListener(this);
        saveButton.setBackground(darkBlue);
        saveButton.setForeground(Color.WHITE);

        editPanel.add(new JLabel());
        editPanel.add(saveButton);

        getContentPane().setBackground(darkBlue);
        add(editPanel);
    }
    public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Save Changes")) {
           String patientName2 = patientNameField2.getText();;
           String date = ((JTextField)
appdate.getDateEditor().getUiComponent()).getText();
            String appointmentTime = appointmentTimeField.getText();
            String appointmentType = (String)
typeComboBox.getSelectedItem();


saveToDatabase(patientName2,date,appointmentTime,appointmentType);

            appointmentDateField.setText("");
            appointmentTimeField.setText("");



        }
    }
    private void saveToDatabase(String patientName2,String date,
String appointmentTime,String appointmentType) {
        String url = "jdbc:mysql://localhost:3306/microproject";
        String username = "root";
        String password = "jacob";

        try (Connection conn = DriverManager.getConnection(url,
username, password)) {
            String query = "UPDATE appointments SET Date = ?,Time =
?,app_type = ? WHERE Patient_Name = ?";
            PreparedStatement statement =
conn.prepareStatement(query);
            statement.setString(1, date);
            statement.setString(2, appointmentTime);
            statement.setString(3, appointmentType);
            statement.setString(4, patientName2);
            statement.executeUpdate();

            JOptionPane.showMessageDialog(null, "Appointment booked
successfully!");

        }
        catch (SQLException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error: Failed to save
data to database!");
        }
    }
}
```

APPENDIX B: USER MANUAL

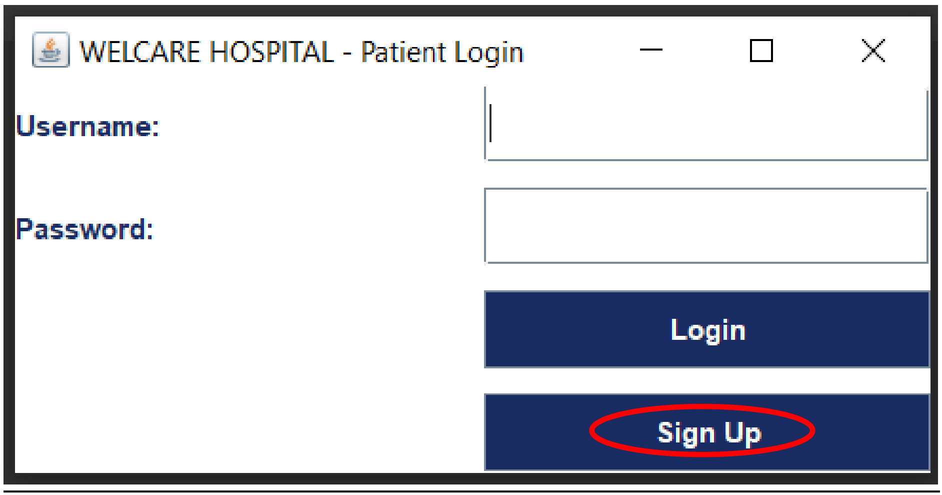
User Interface



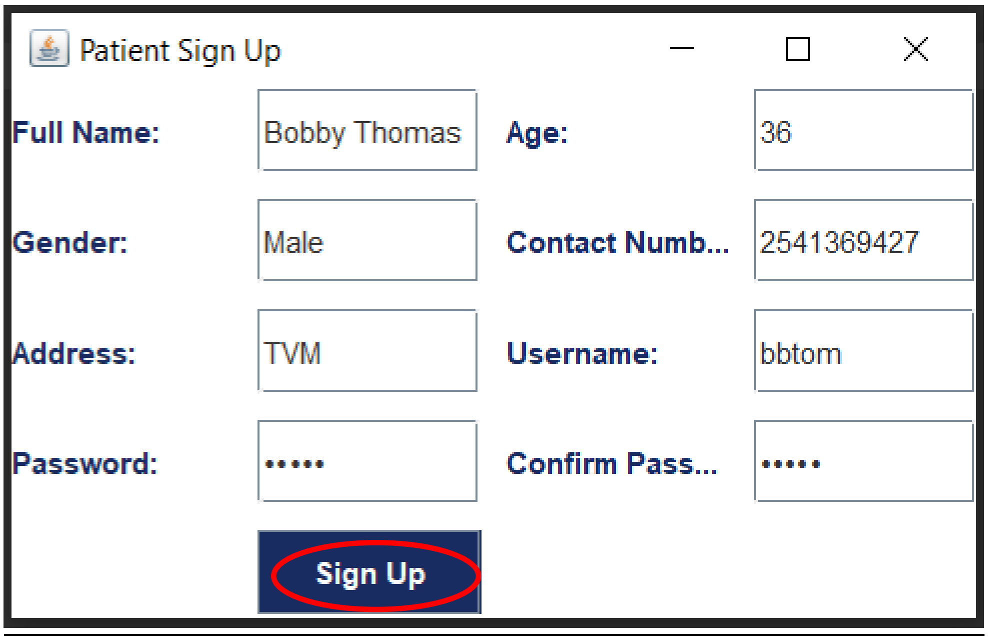
Patient Login

A new patient can register in the portal. To register as a new patient:

1) Click on sign up.



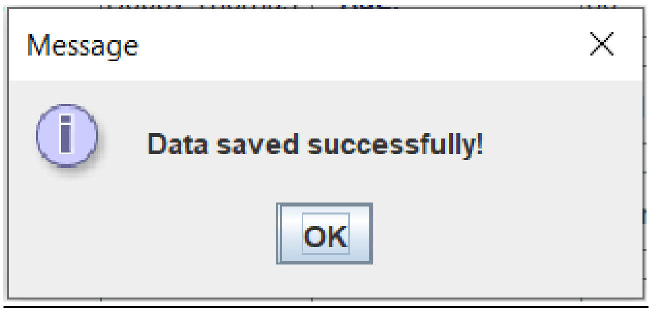
2) Fill the Patient Details along with a new username and password for future logins.



Patient Sign Up

Full Name:	Bobby Thomas	Age:	36
Gender:	Male	Contact Numb...	2541369427
Address:	TVM	Username:	bbtom
Password:	•••••	Confirm Pass...	•••••

3) The Patient is successfully registered and stored in the database.



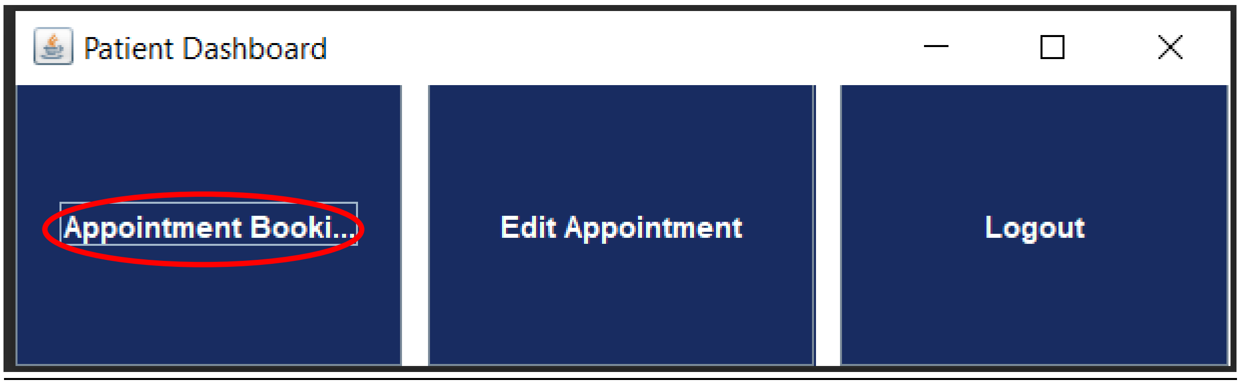
Message

Data saved successfully!

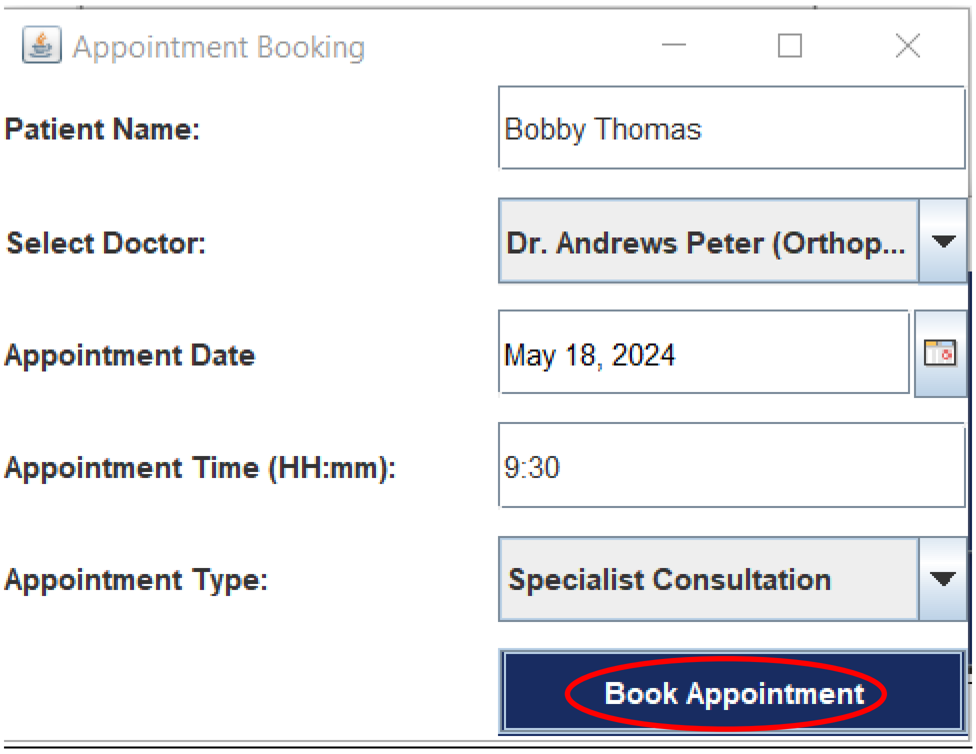
After sign up, the patient is redirected to the Patient Dashboard window.

If the patient needs to book a new appointment:

1) Click on 'Appointment Booking'

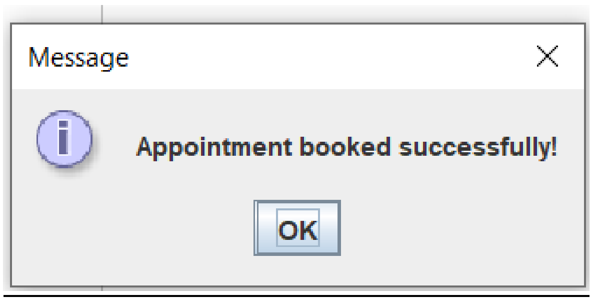


2) Fill out the new appointment details and click book appointments



Patient Name:	Bobby Thomas
Select Doctor:	Dr. Andrews Peter (Orthop...
Appointment Date	May 18, 2024
Appointment Time (HH:mm):	9:30
Appointment Type:	Specialist Consultation

3) Appointment is booked successfully and stored in the database.

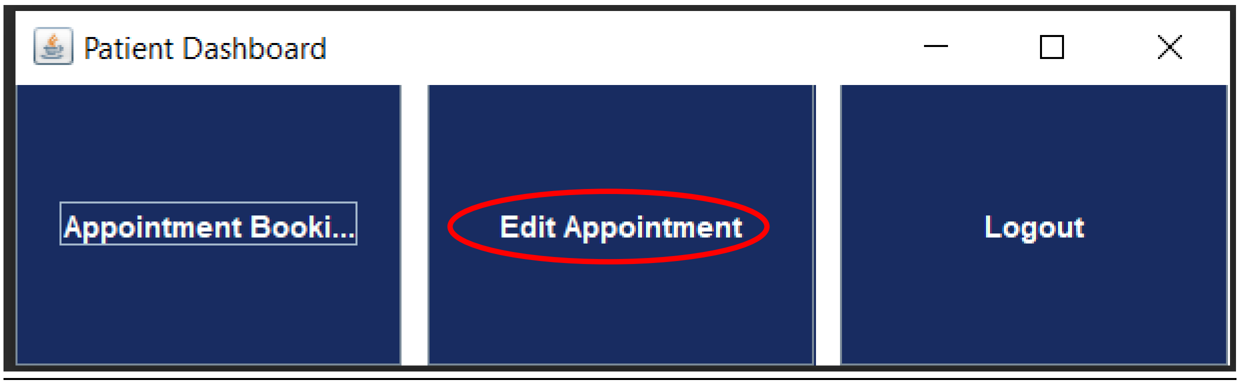

Message

Appointment booked successfully!

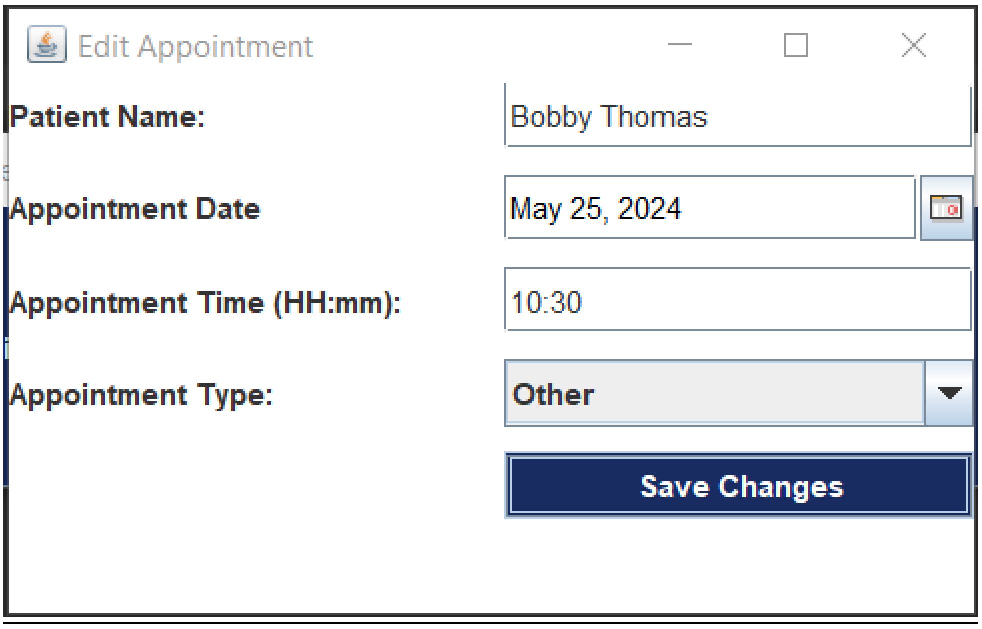
If the patient needs to edit a previously booked appointment

1) Click on ‘Edit Appointment.

:



2) Edit the required details and click save changes.


Edit Appointment

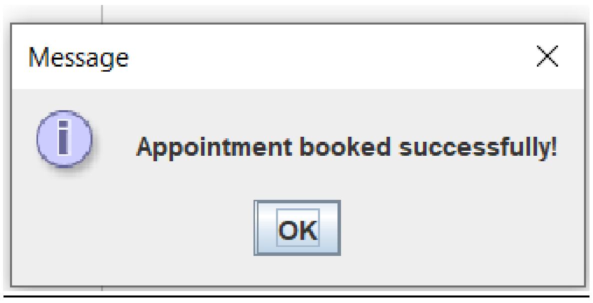
Patient Name: Bobby Thomas

Appointment Date May 25, 2024

Appointment Time (HH:mm): 10:30

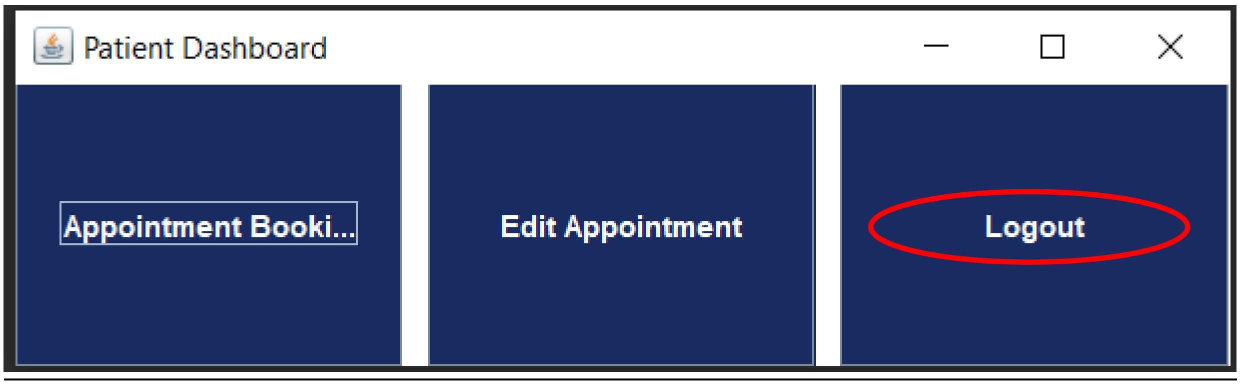
Appointment Type: Other

3) The Appointment is updated successfully

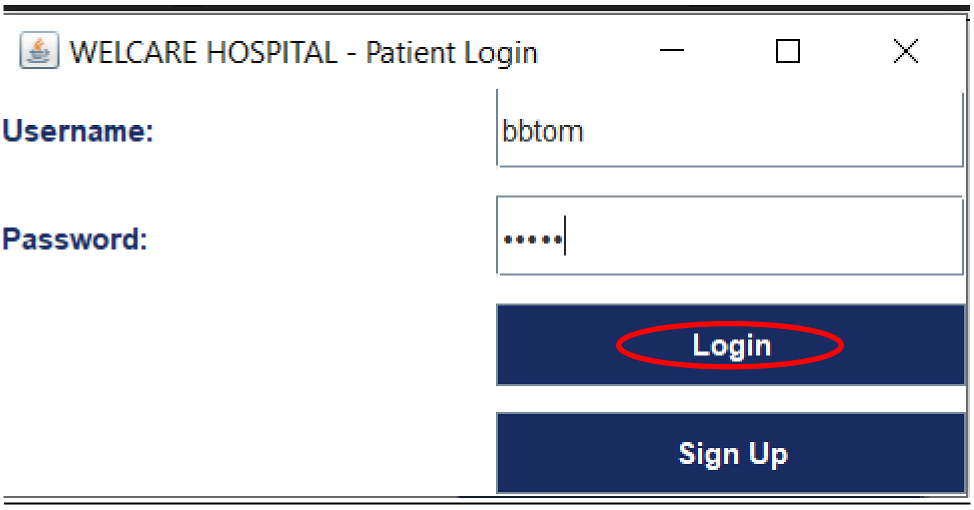


Appointment booked successfully!

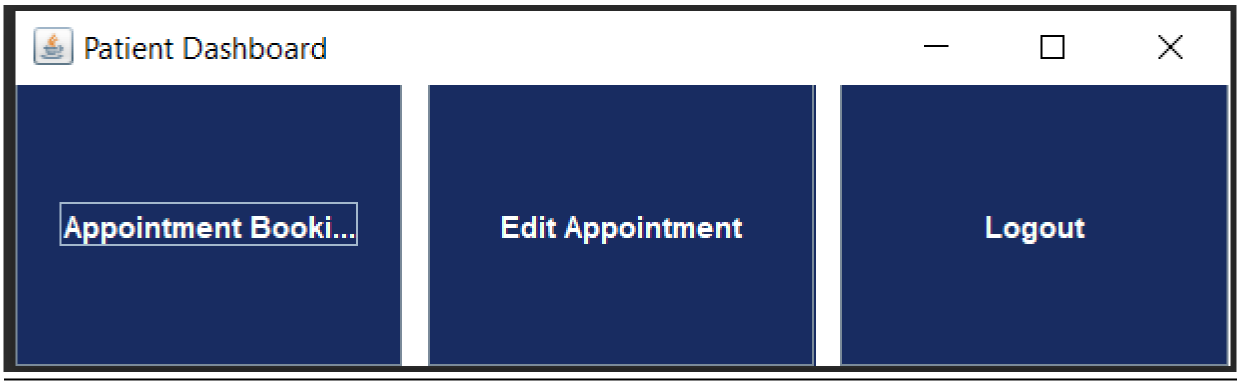
4) Log out from patient dashboard

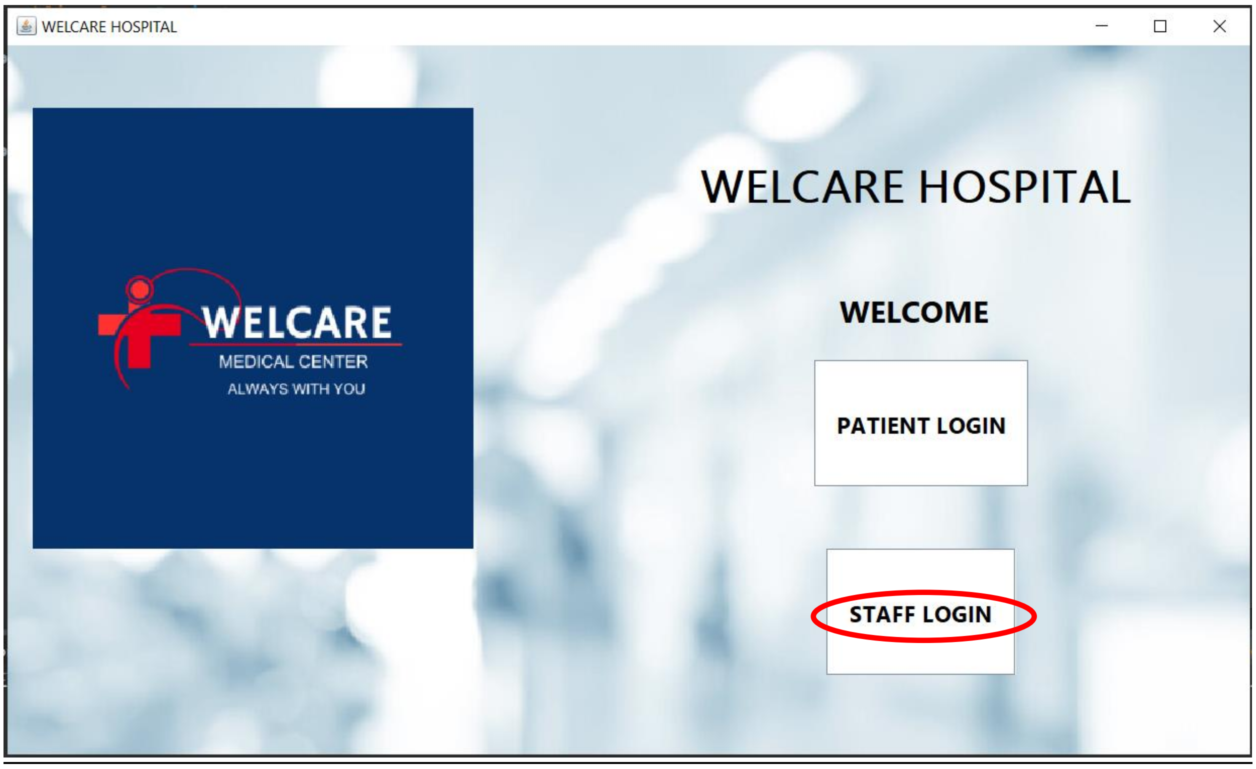


- An existing user can log using his credentials (username and password)



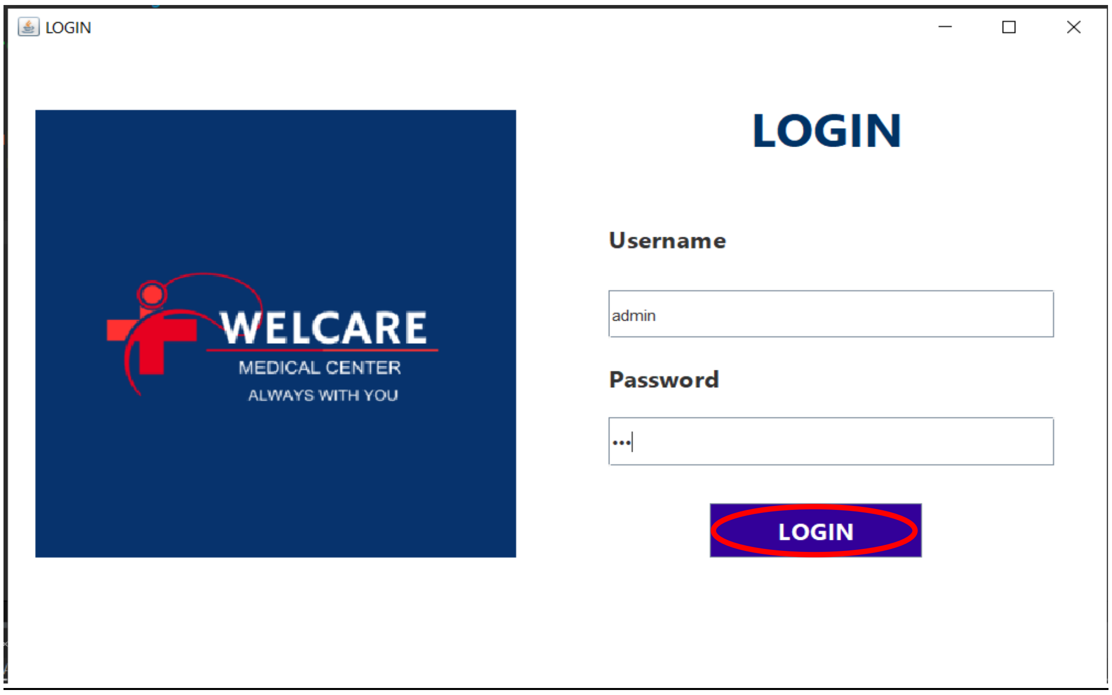
- After login, the patient is redirected to the Patient Dashboard window.

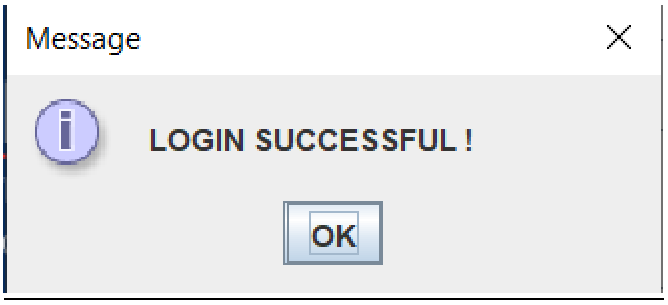




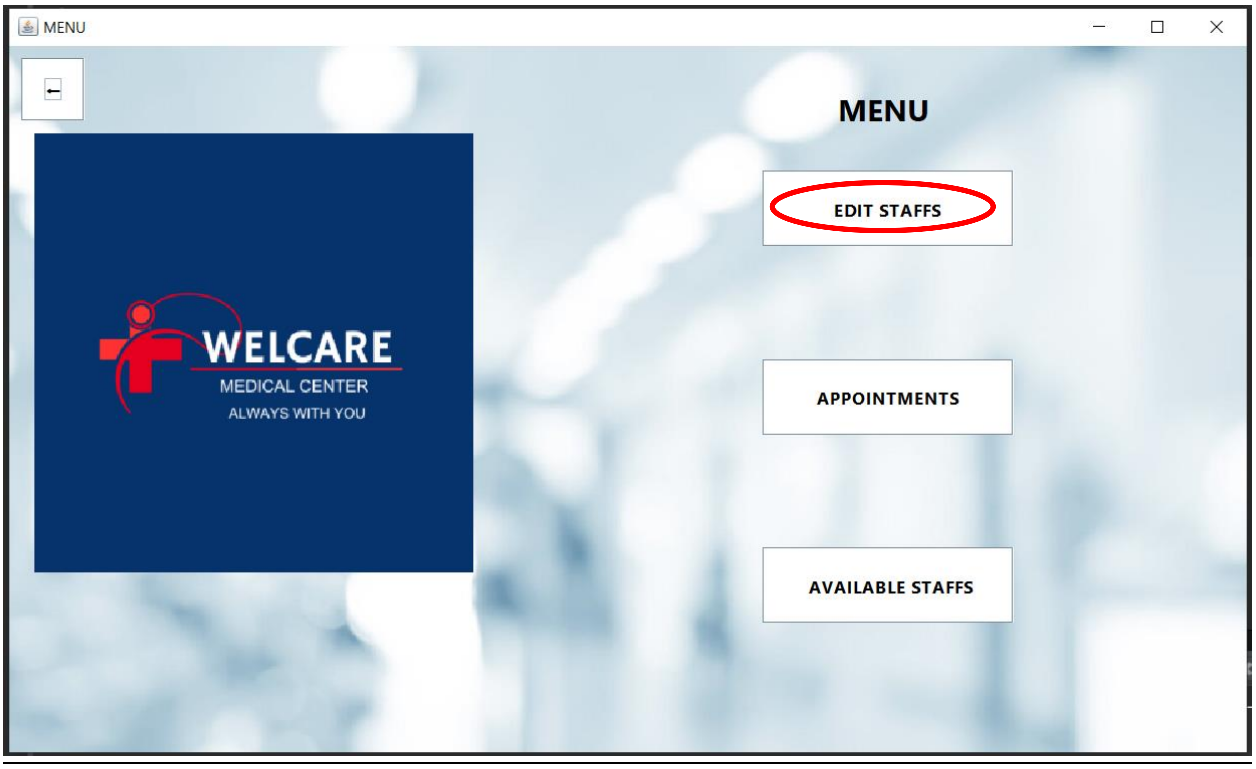
Staff /Admin Login

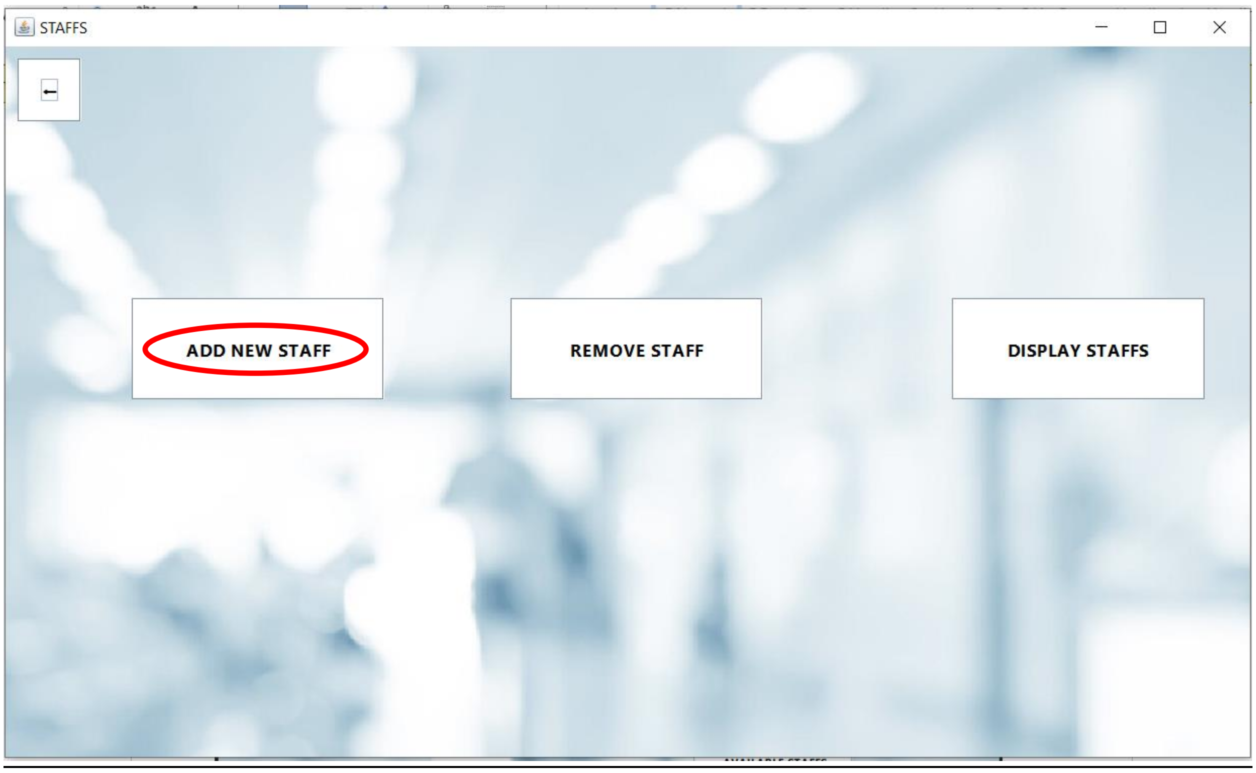
Any staff or particularly the admin has few control over the staffs and appointments in the portal. To login as an admin the username is admin' and password is '123'.

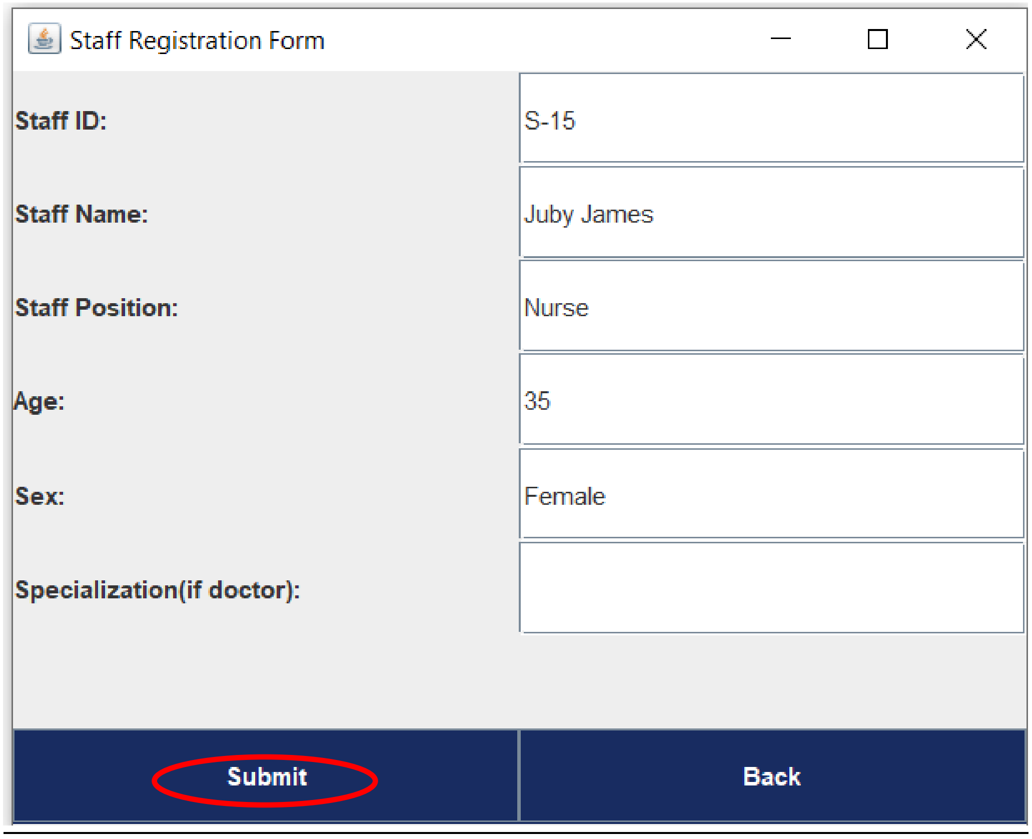



Message

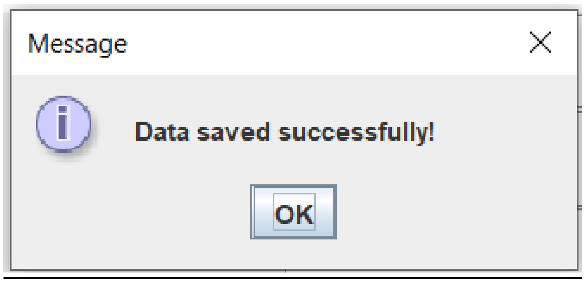
LOGIN SUCCESSFUL !



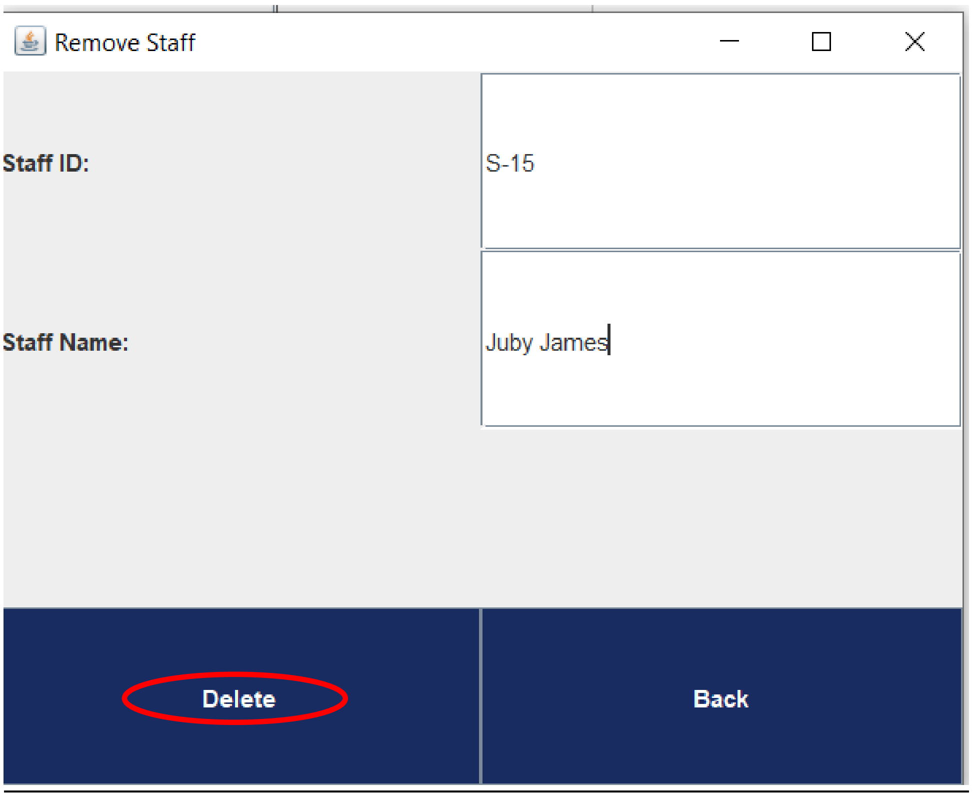



Staff Registration Form

Staff ID:	S-15
Staff Name:	Juby James
Staff Position:	Nurse
Age:	35
Sex:	Female
Specialization(if doctor):	

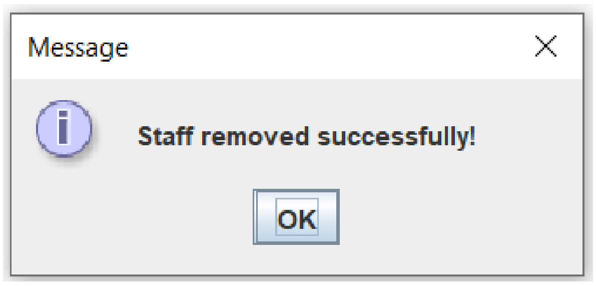

Message

Data saved successfully!

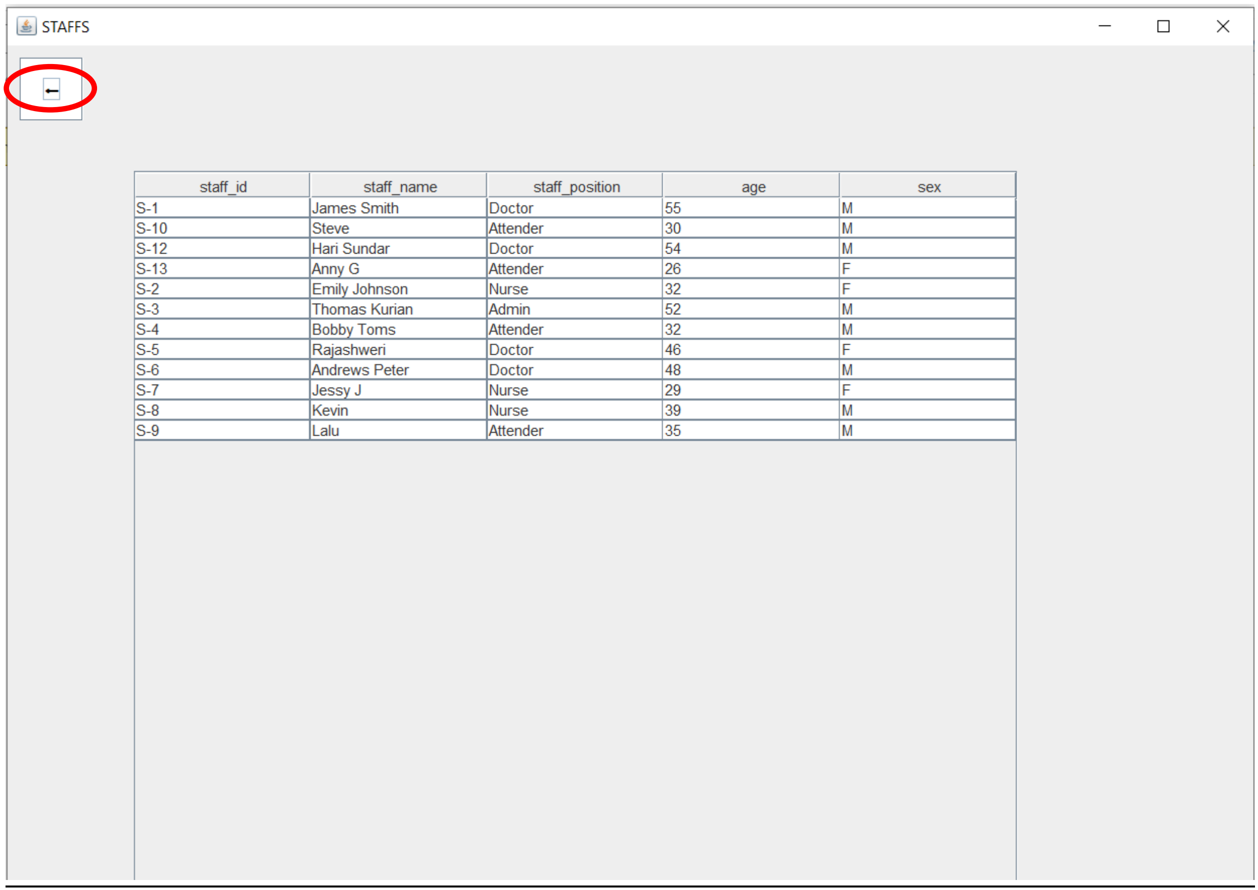


Staff ID: S-15

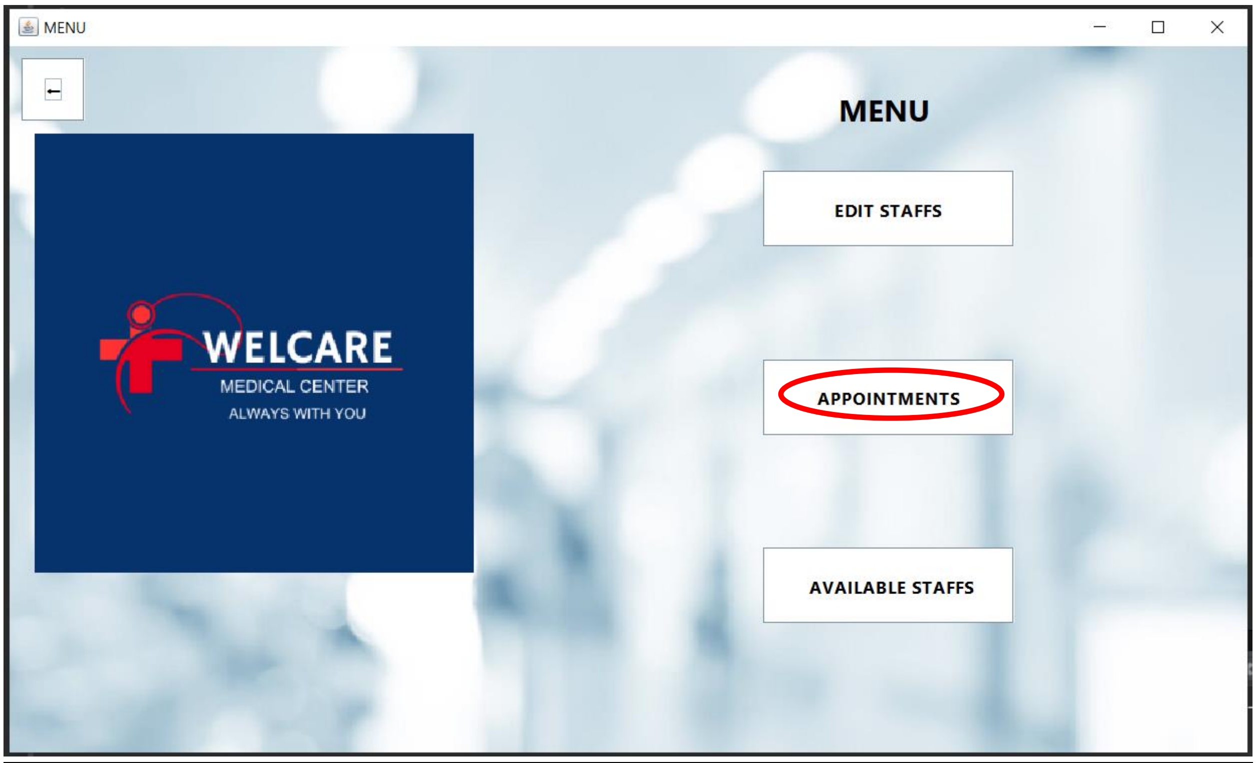
Staff Name: Juby James

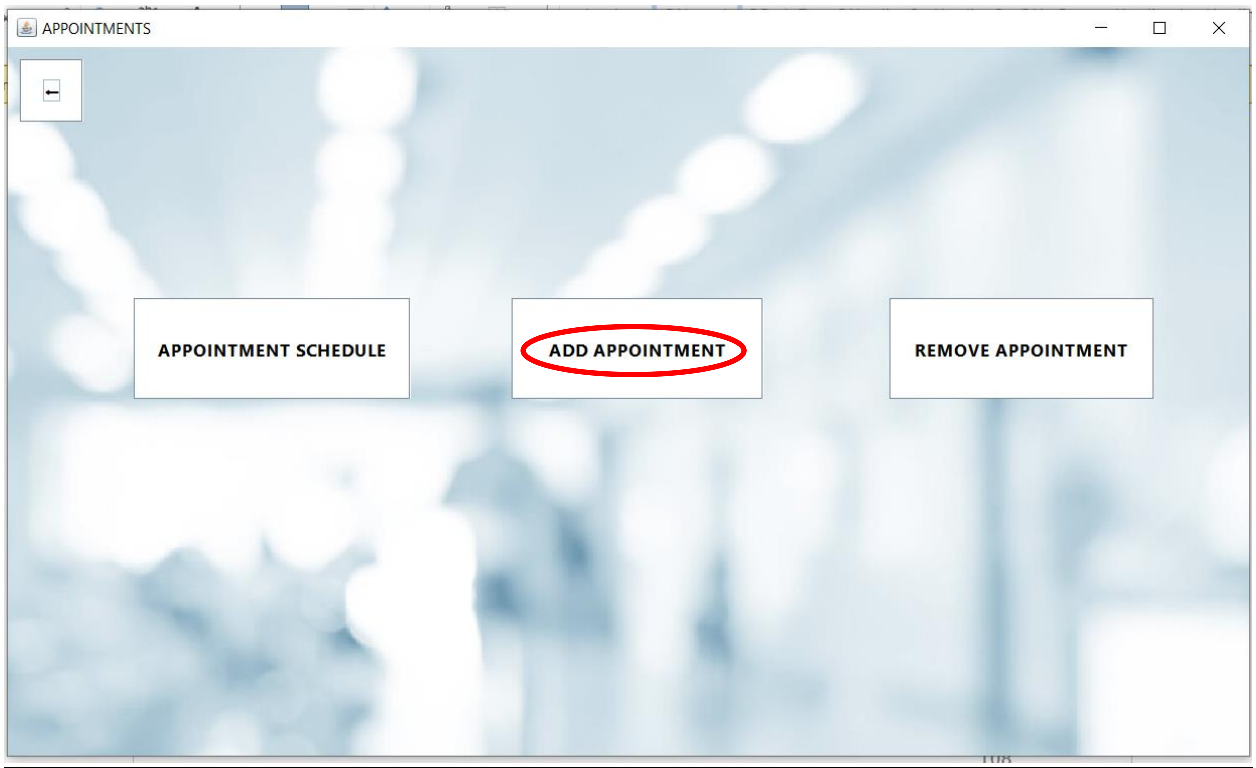

Message

Staff removed successfully!

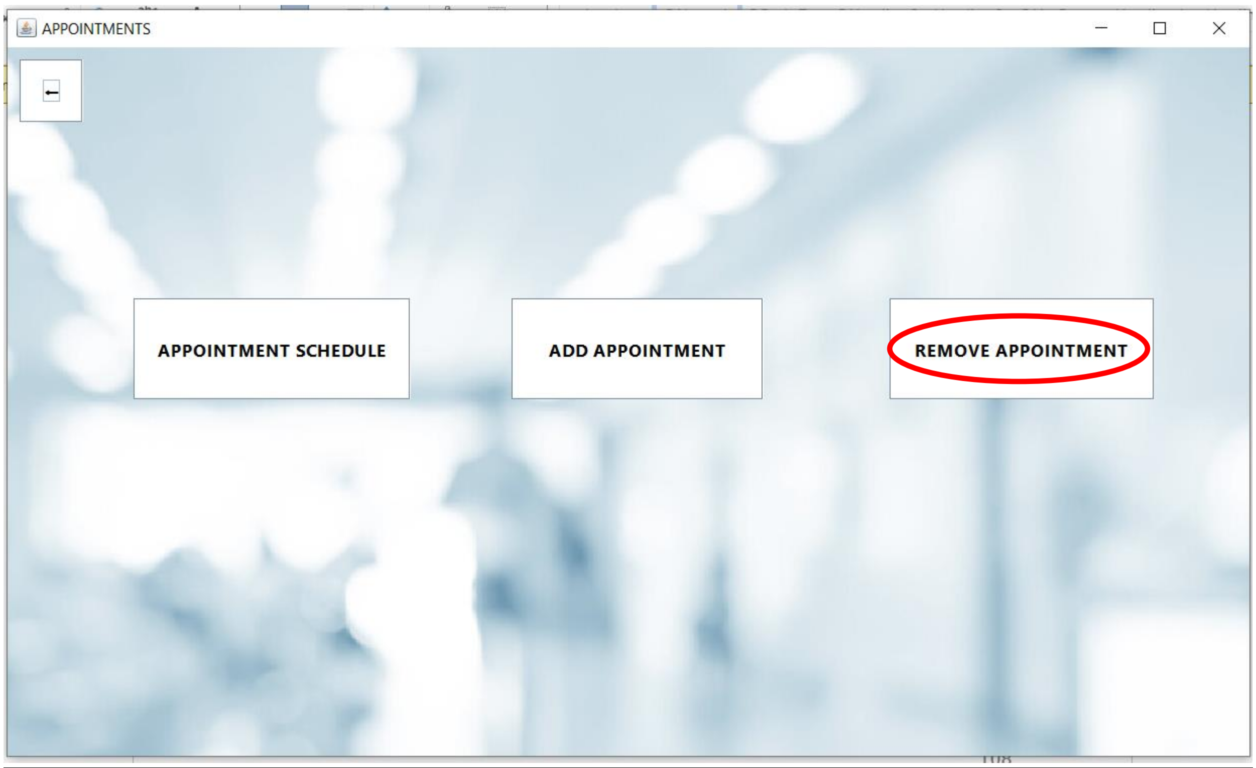


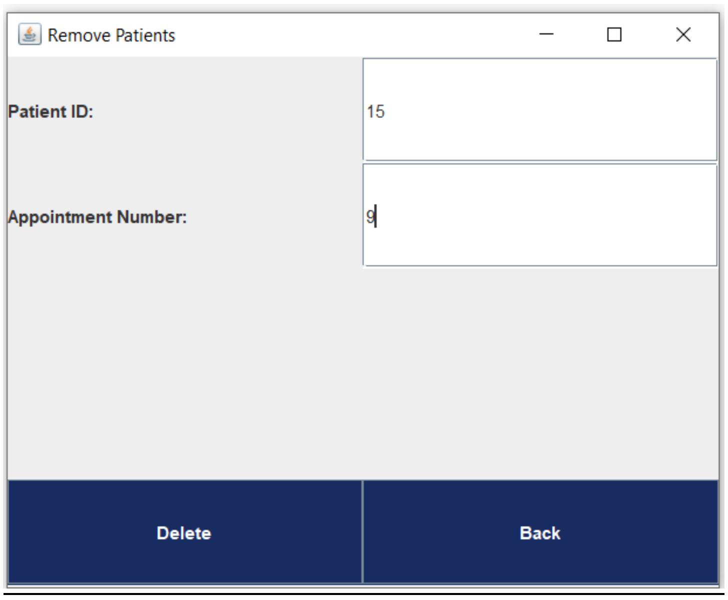
staff_id	staff_name	staff_position	age	sex
S-1	James Smith	Doctor	55	M
S-10	Steve	Attender	30	M
S-12	Hari Sundar	Doctor	54	M
S-13	Anny G	Attender	26	F
S-2	Emily Johnson	Nurse	32	F
S-3	Thomas Kurian	Admin	52	M
S-4	Bobby Toms	Attender	32	M
S-5	Rajashweri	Doctor	46	F
S-6	Andrews Peter	Doctor	48	M
S-7	Jessy J	Nurse	29	F
S-8	Kevin	Nurse	39	M
S-9	Lalu	Attender	35	M





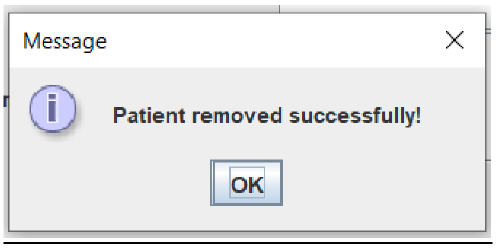
*Same as adding appointment in the Patient Portal



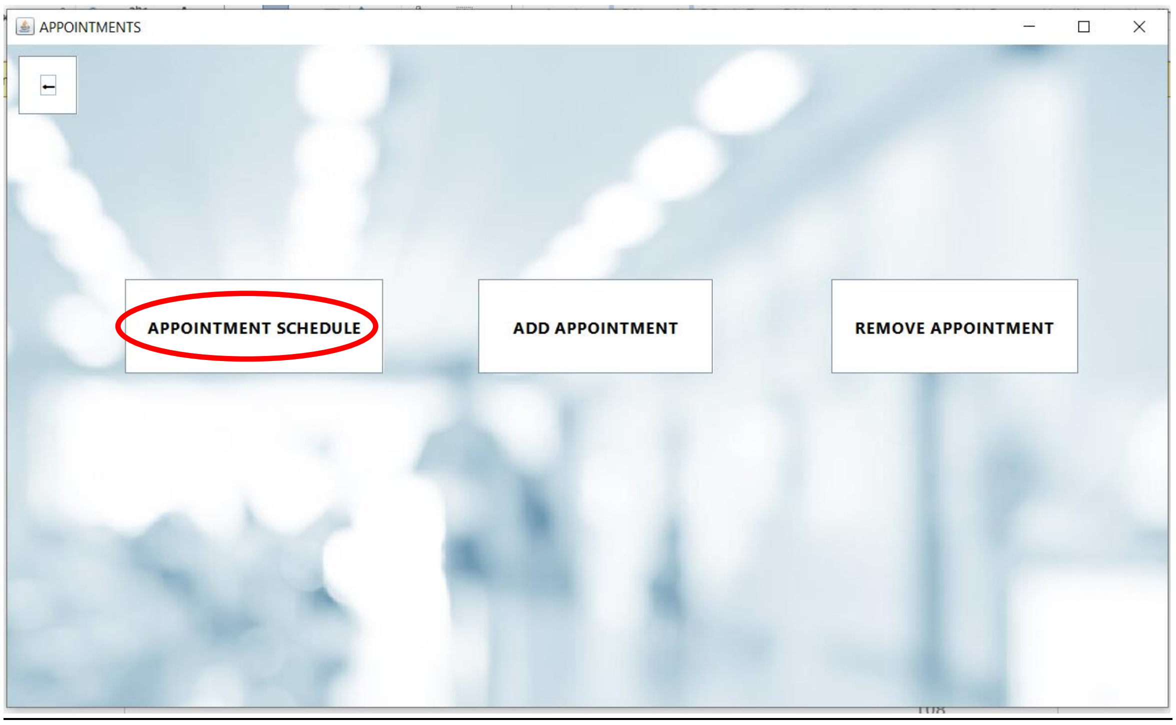


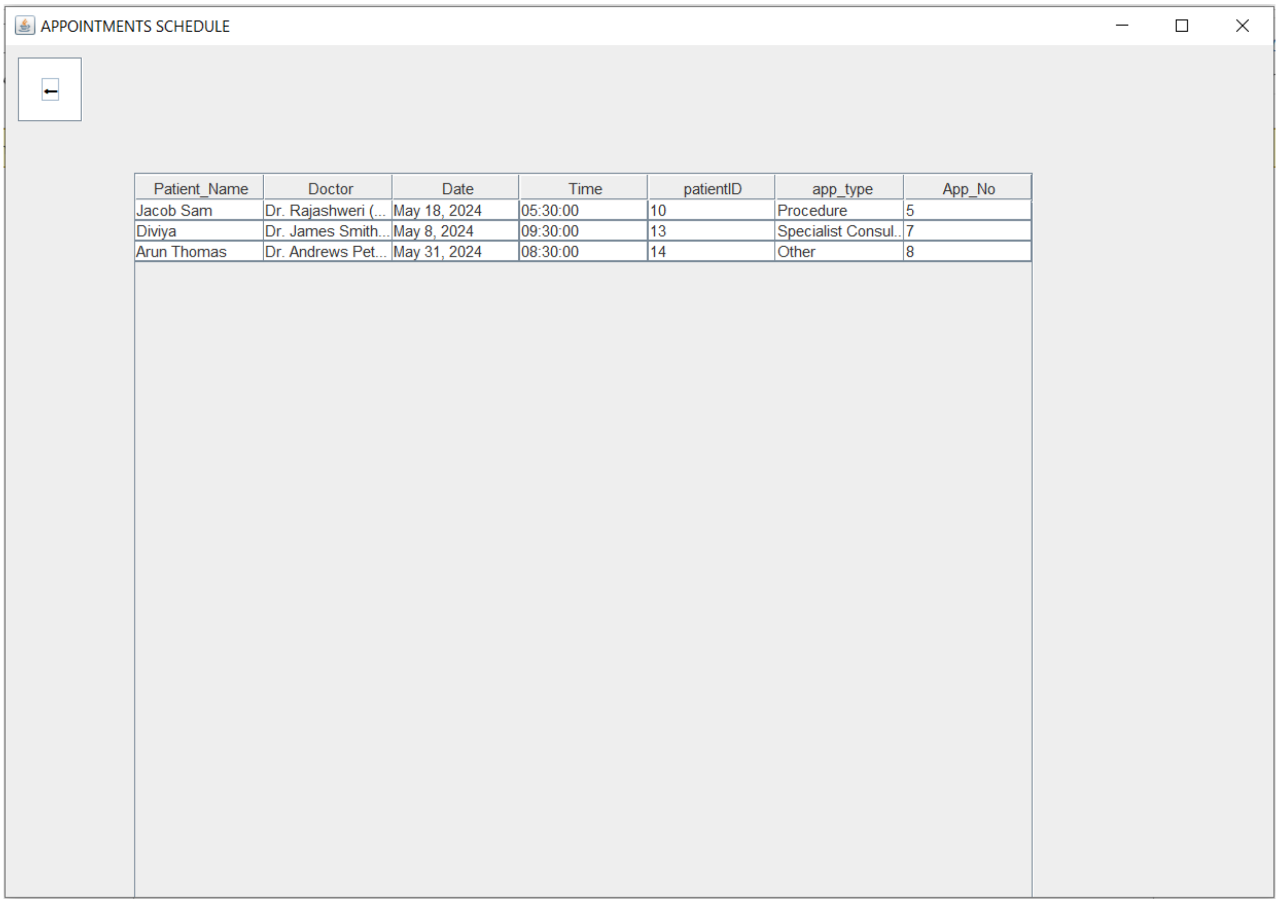
Patient ID: 15

Appointment Number: 9

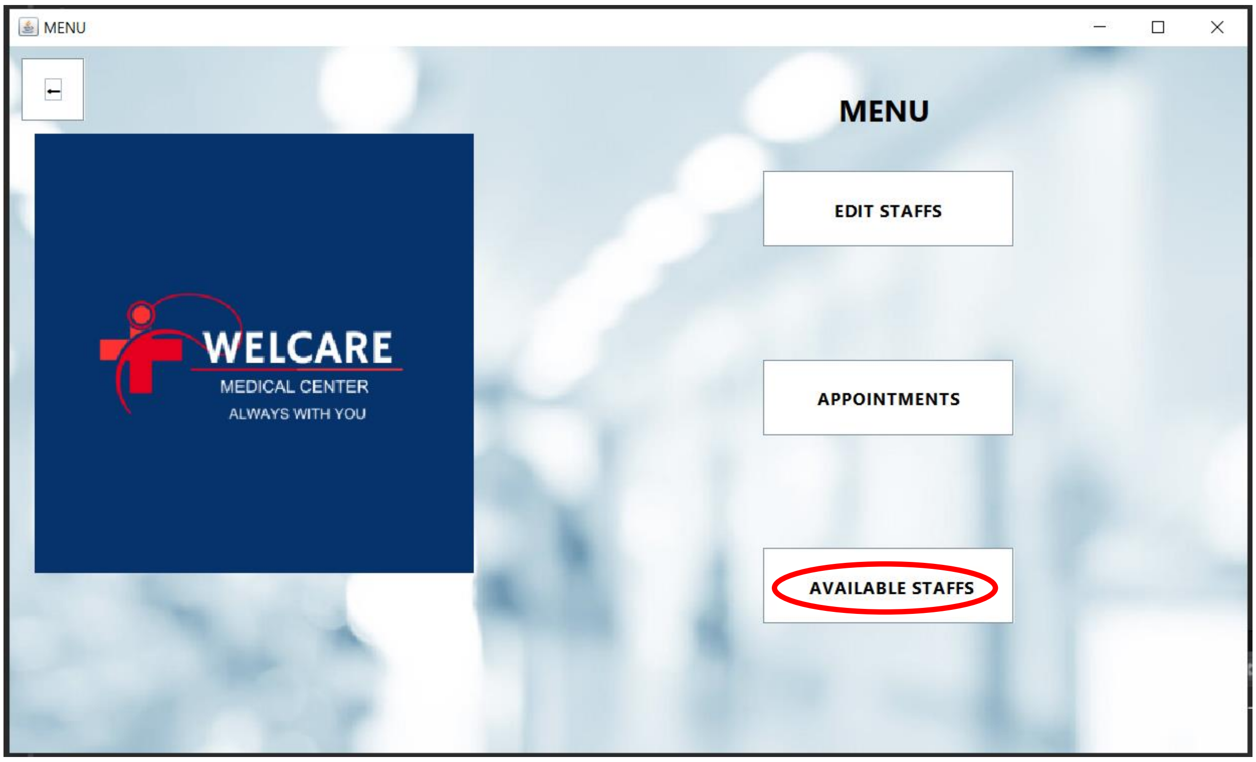


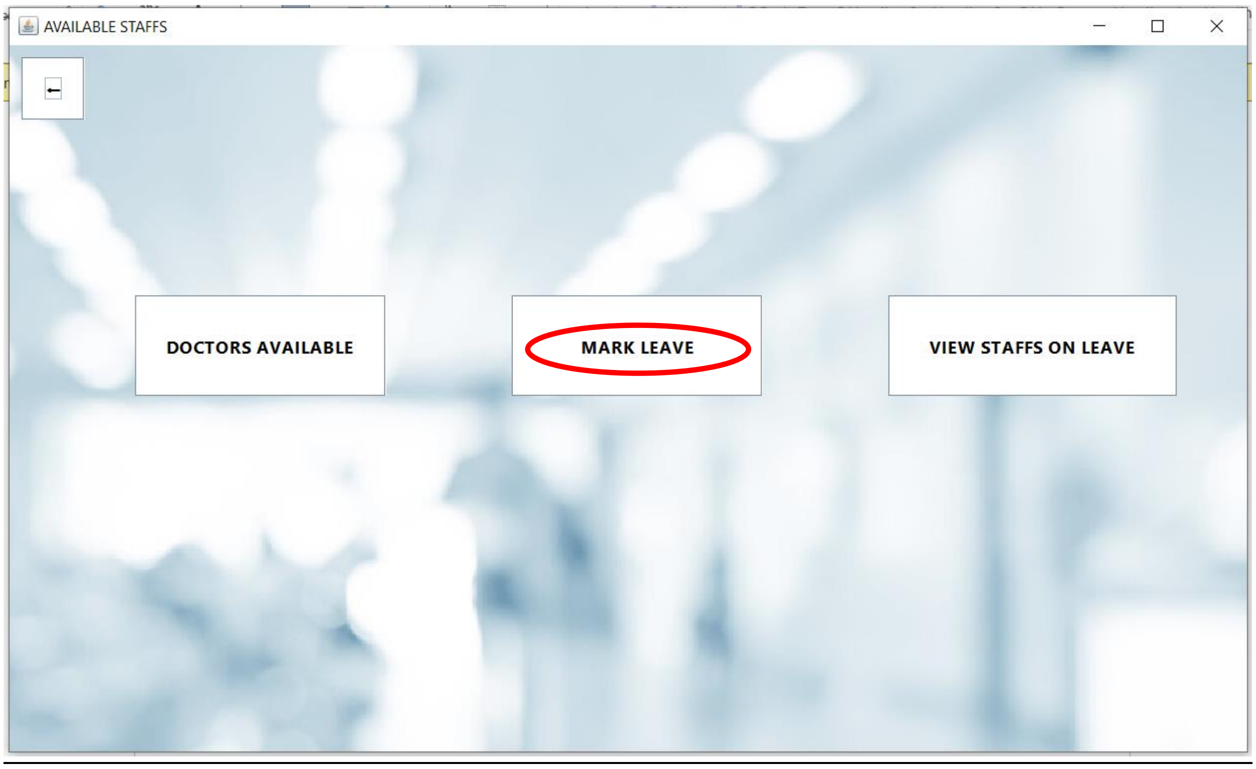
Patient removed successfully!

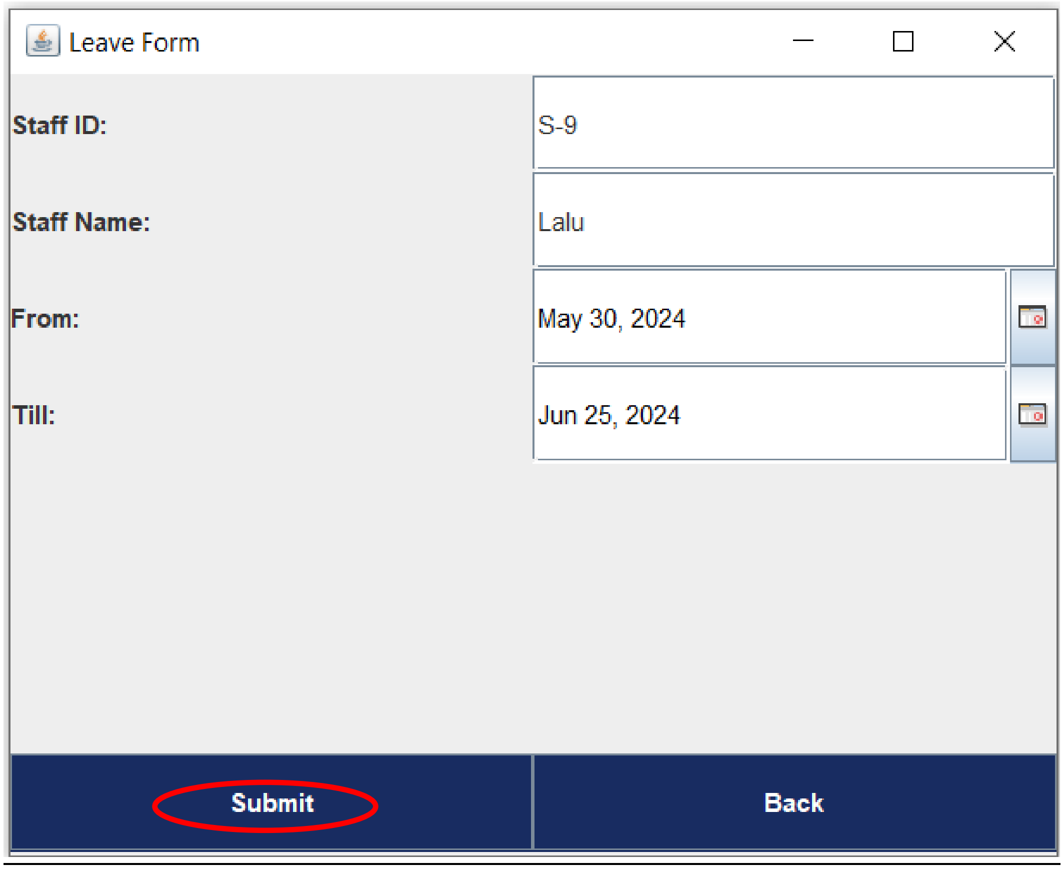



APPOINTMENTS SCHEDULE

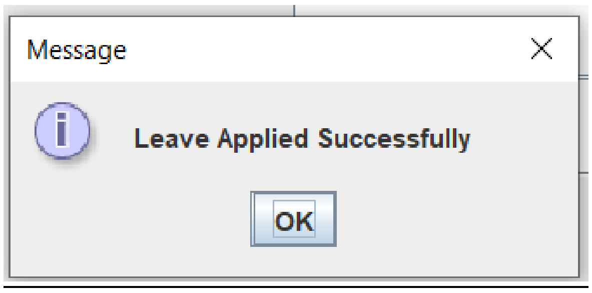
Patient_Name	Doctor	Date	Time	patientID	app_type	App_No
Jacob Sam	Dr. Rajashweri (...	May 18, 2024	05:30:00	10	Procedure	5
Diviya	Dr. James Smith...	May 8, 2024	09:30:00	13	Specialist Consul...	7
Arun Thomas	Dr. Andrews Pet...	May 31, 2024	08:30:00	14	Other	8





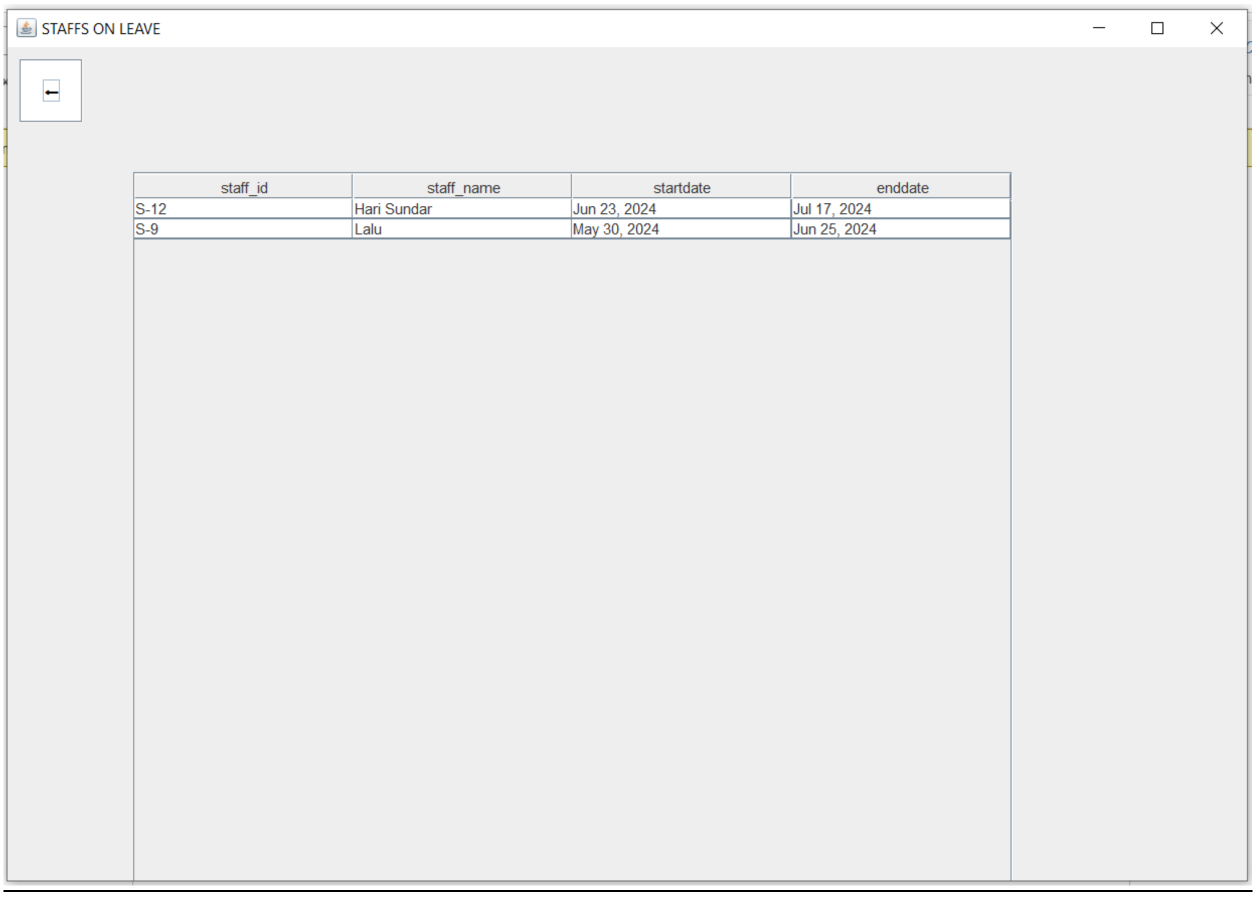

Leave Form

Staff ID:	S-9
Staff Name:	Lalu
From:	May 30, 2024
Till:	Jun 25, 2024



Message

Leave Applied Successfully



STAFFS ON LEAVE

staff_id	staff_name	startdate	enddate
S-12	Hari Sundar	Jun 23, 2024	Jul 17, 2024
S-9	Lalu	May 30, 2024	Jun 25, 2024