1. Introduction

Problem Statement

Managing the logistics and services for many pilgrims can be overwhelming when handled manually. Tasks such as registration, medical record-keeping, accommodation management, and permit generation require a centralized, secure, and efficient system to ensure a smooth pilgrimage experience.

Objectives

This project aims to design and implement a comprehensive system that:

1. Collects and validates accurate pilgrim registration data.
2. Manages medical profiles including allergies, vaccination status, and emergency contacts.
3. Facilitate accommodation and transport bookings.
4. Generates and manages permits for pilgrimage activities.
5. Provides feedback options for service quality.
6. Ensures administrative control with secure data handling and validation mechanisms.

2. Design

1. Database Design

a. Entities and Relationships

The core entities of the system are:

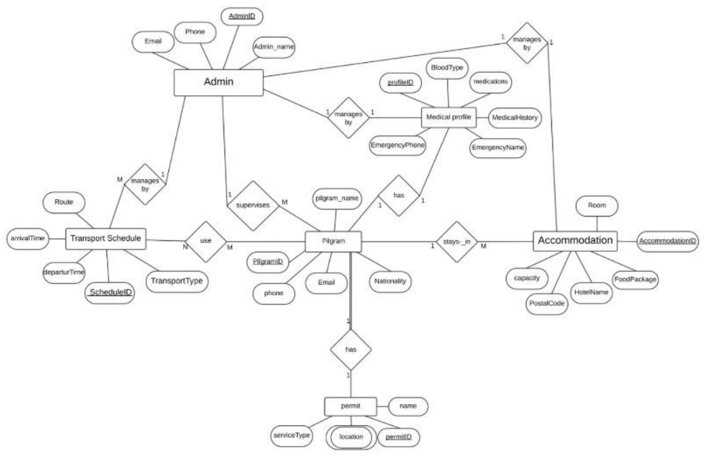
- Pilgrim
- Admin
- Medical Profile
- Accommodation
- Transport Schedule
- Permit
- Feedback

Entity Attributes:

- Pilgrim: pilgrimID (PK), name, age, gender
- Admin: adminID (PK), name, role

- Medical Profile: profileID (PK), bloodType, vaccination, medications, pilgrimID (FK)
- Accommodation: accommodationID (PK), roomType, capacity, address
- Transport Schedule: scheduleID (PK), arrivalTime, departureTime
- Permit: permitID (PK), name, serviceType, location, pilgrimID (FK)
- Feedback: feedbackID (PK), rating, comments, pilgrimID (FK)

b. Entity-Relationship Diagram (ERD)



c. Tables, Primary Keys, Foreign Keys, and Relationships

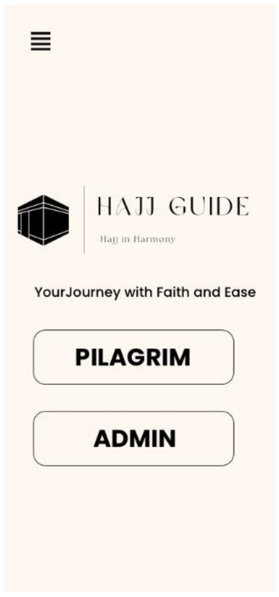
Table	Primary Key	Foreign Key	Relationship
Pilgrim	pilgrimID	-	-
Admin	adminID	-	-
Medical Profile	profileID	pilgrimID → Pilgrim	One-to-One
Accommodation	accommodationID	-	One-to-Many with Pilgrim
Transport Schedule	scheduleID	-	Many-to-Many with Pilgrim (via junction table)
Permit	permitID	pilgrimID → Pilgrim	One-to-Many
Feedback	feedbackID	pilgrimID → Pilgrim	One-to-Many

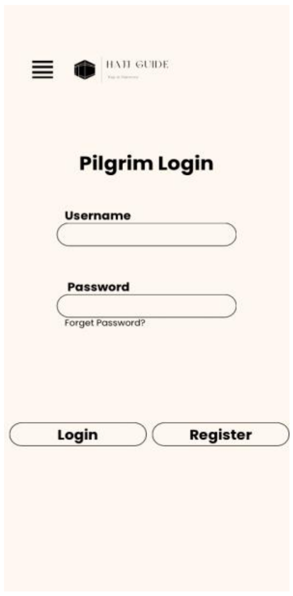
2. GUI Design

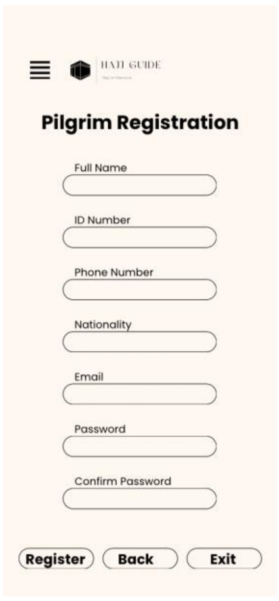
The system features:

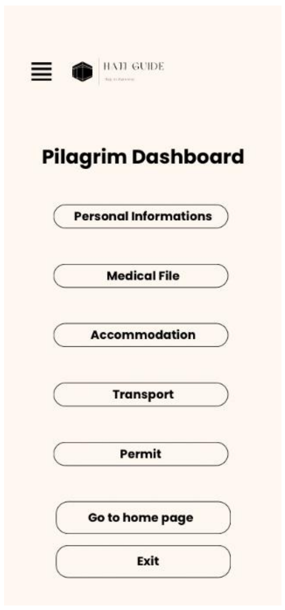
- **Pilgrim Interface:** Registration form, dashboard, transport & accommodation booking, permit view, and feedback form.

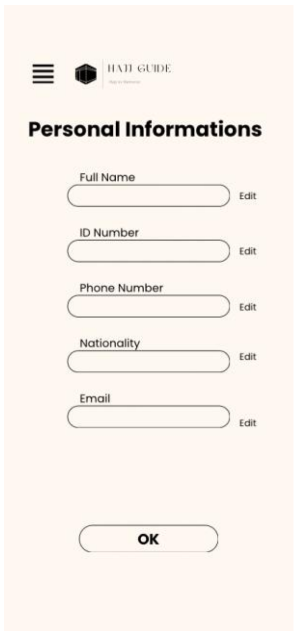
- **Admin Interface:** Pilgrim list, medical profile access, accommodation/transportation control, and permit generation panel.

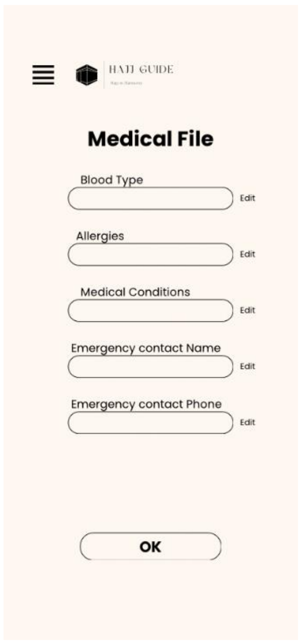


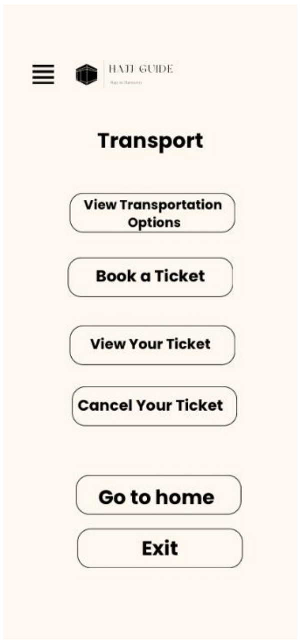


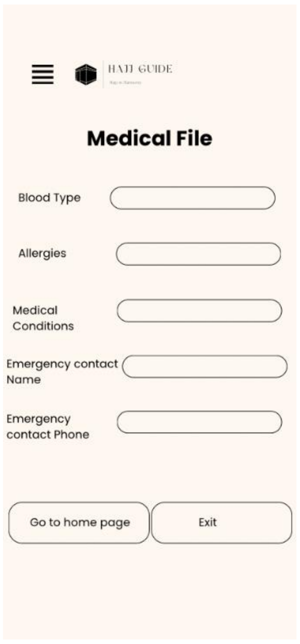


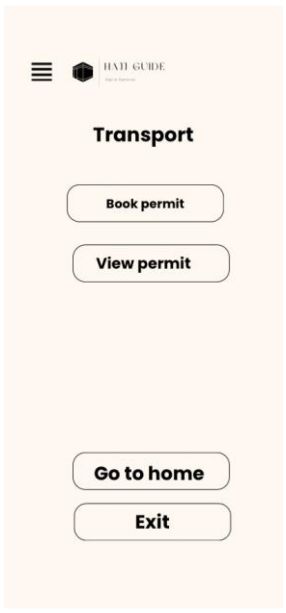


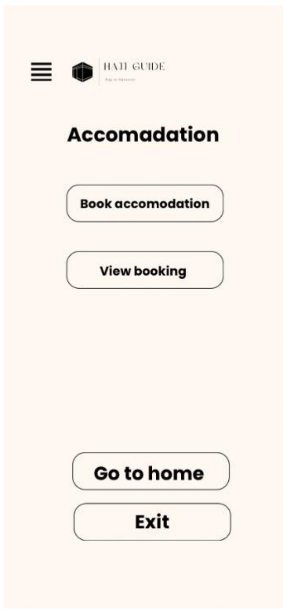


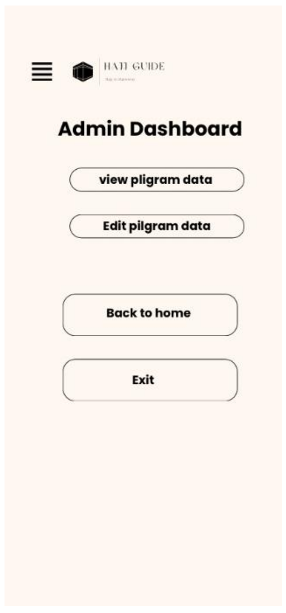


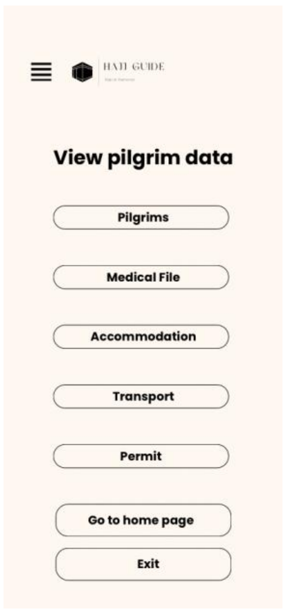


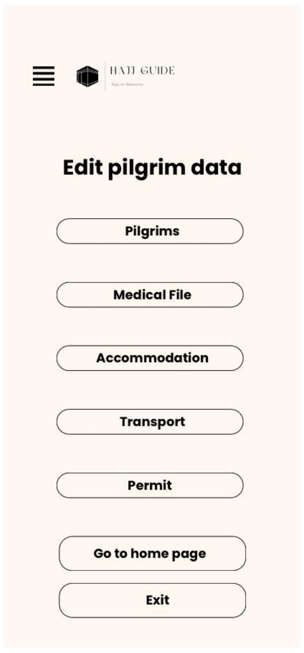


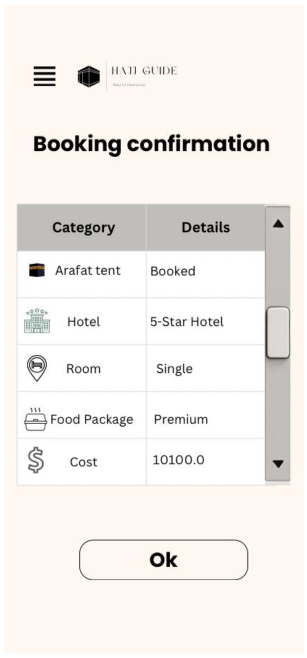






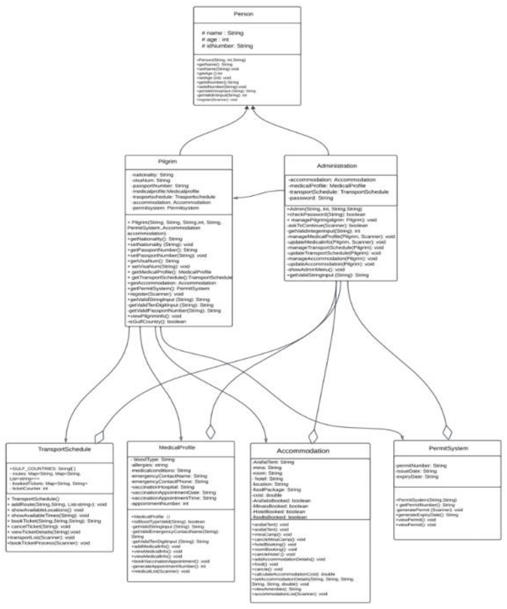






Category	Details
Arafat tent	Booked
Hotel	5-Star Hotel
Room	Single
Food Package	Premium
Cost	10100.0

3. Class Hierarchy (UML Class Diagram)



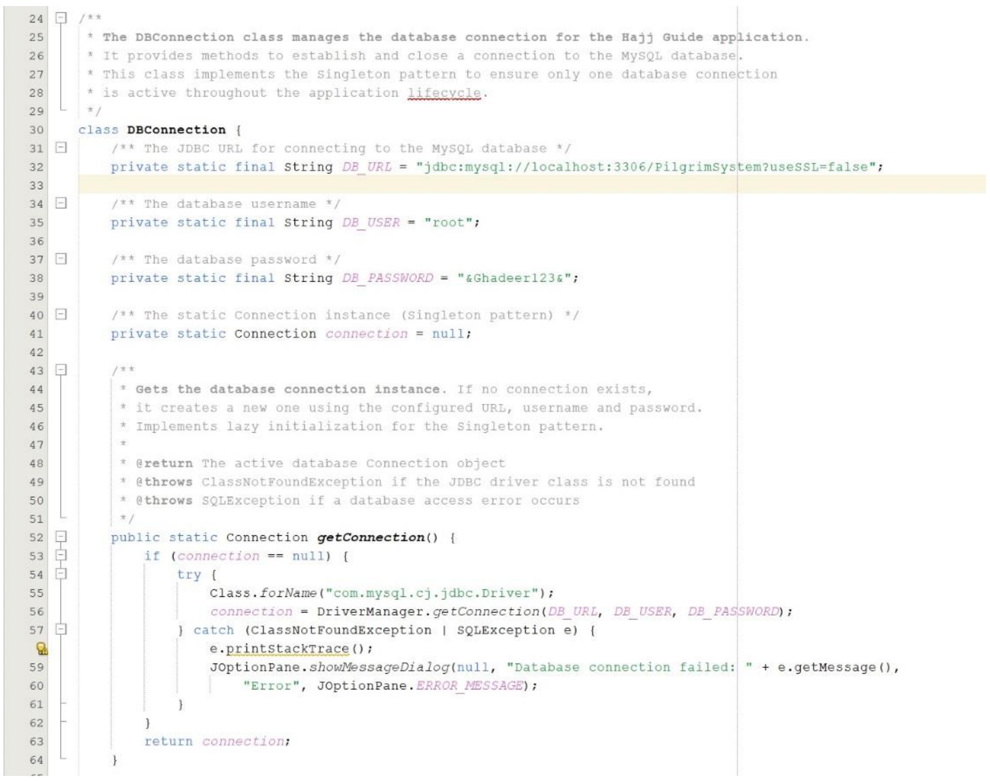
3. Implementation

Technologies Used

- **Java Programming Language**: Core programming language used for system development due to its object-oriented capabilities, platform independence, and robust standard libraries.
- **MySQL Database**: Relational database management system selected for storing pilgrim data, accommodation information, medical profiles, and permits.
- **JDBC (Java Database Connectivity)**: API used to establish connections between the Java application and the MySQL database, enabling data retrieval and manipulation.

Sample Code Snippet

- The first image shows the DBConnection class, which handles connecting to the MySQL database using the Singleton pattern.
- The second image shows the PilgrimDashboard class, which creates the main interface for pilgrims after login using Java Swing components.



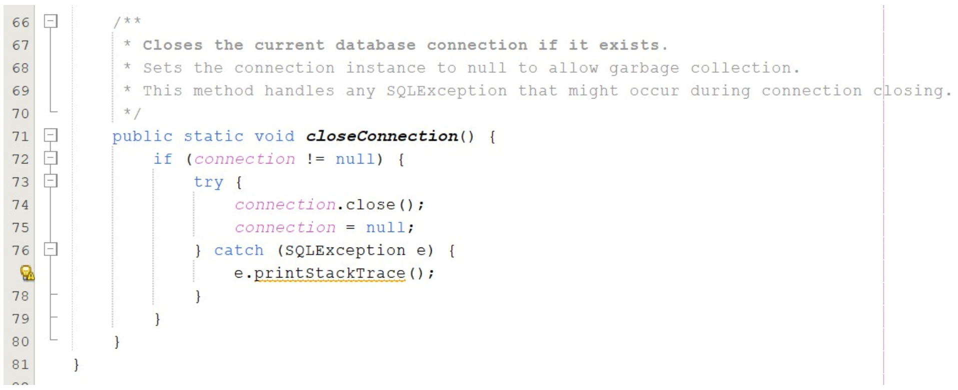
```java
/**
 * The DBConnection class manages the database connection for the Hajj Guide application.
 * It provides methods to establish and close a connection to the MySQL database.
 * This class implements the Singleton pattern to ensure only one database connection
 * is active throughout the application lifecycle.
 */
class DBConnection {
    /** The JDBC URL for connecting to the MySQL database */
    private static final String DB_URL = "jdbc:mysql://localhost:3306/PilgrimSystem?useSSL=false";

    /** The database username */
    private static final String DB_USER = "root";

    /** The database password */
    private static final String DB_PASSWORD = "&Ghadeer123&";

    /** The static Connection instance (Singleton pattern) */
    private static Connection connection = null;

    /**
     * Gets the database connection instance. If no connection exists,
     * it creates a new one using the configured URL, username and password.
     * Implements lazy initialization for the Singleton pattern.
     *
     * @return The active database Connection object
     * @throws ClassNotFoundException if the JDBC driver class is not found
     * @throws SQLException if a database access error occurs
     */
    public static Connection getConnection() {
        if (connection == null) {
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
                connection = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);
            } catch (ClassNotFoundException | SQLException e) {
                e.printStackTrace();
                JOptionPane.showMessageDialog(null, "Database connection failed: " + e.getMessage(),
                    "Error", JOptionPane.ERROR_MESSAGE);
            }
        }
        return connection;
    }
```



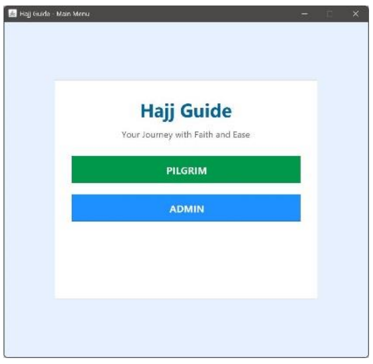
```java
    /**
     * Closes the current database connection if it exists.
     * Sets the connection instance to null to allow garbage collection.
     * This method handles any SQLException that might occur during connection closing.
     */
    public static void closeConnection() {
        if (connection != null) {
            try {
                connection.close();
                connection = null;
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

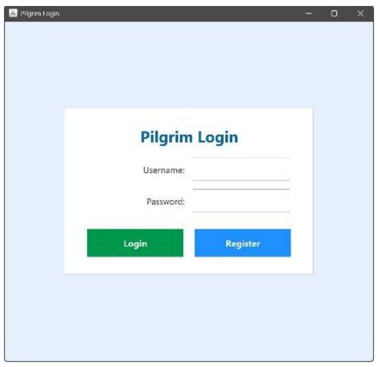
Challenges Faced

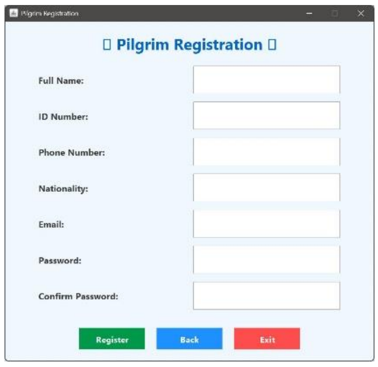
- **Database Integration Complexity:** Establishing effective relationships between multiple entities like Pilgrims, Medical Profiles, and Permits required careful consideration of foreign key constraints and relationship mapping.
- **Concurrent User Management:** Implementing mechanisms to handle multiple users accessing the system simultaneously presented challenges in ensuring data consistency and preventing conflicts.
- **UI/UX Design Balance**: Designing interfaces that were intuitive for diverse users (including those with limited technical experience) while including all necessary functionalities required multiple iterations

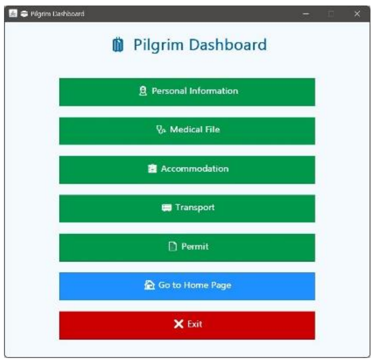
4. Testing

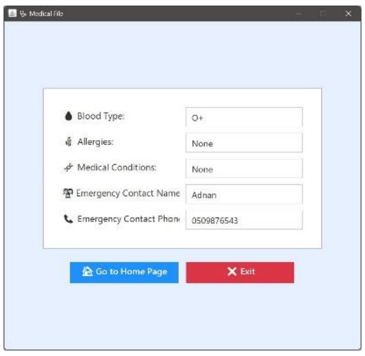
Test Cases and Results



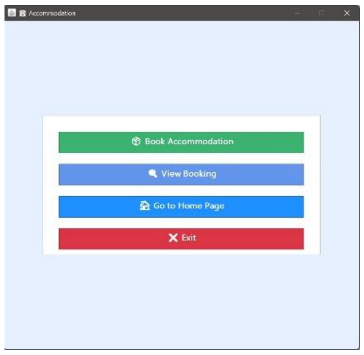


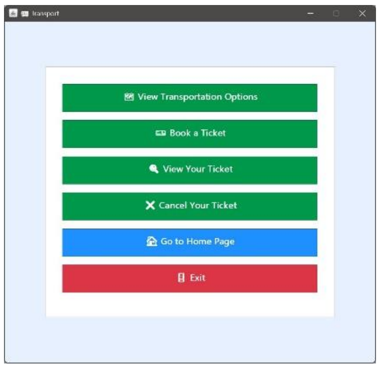


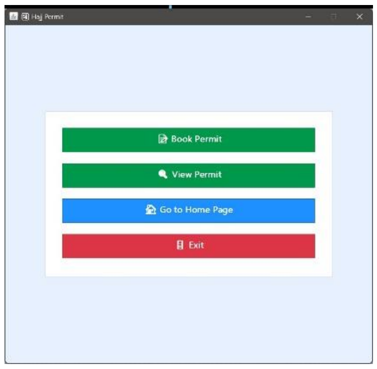


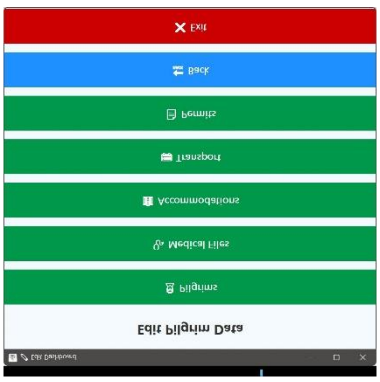


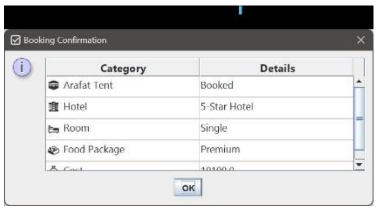
Blood Type:	O+
Allergies:	None
Medical Conditions:	None
Emergency Contact Name	Adnan
Emergency Contact Phon	0509876543



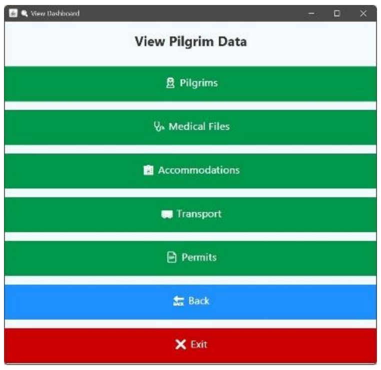


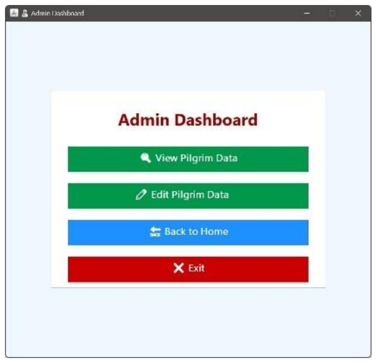






Category	Details
Arafat Tent	Booked
Hotel	5-Star Hotel
Room	Single
Food Package	Premium
Cost	10100.0





5. Conclusion

This project successfully delivers a fully functional Pilgrim Management System that centralizes registration, health records, accommodation, transportation, and permits. It enhances both administrative efficiency and user experience.

6. References

[1] Y. Daniel Liang, “Introduction to JAVA Programming, Comprehensive Version”. 10th ed. Pearson Education, 2014.

7. [2] Esam Ali Khan, Mohd Khaled Yousef Shambour, “An analytical study of mobile applications for Hajj and Umrah services”. 4th ed. Applied computing and informatics, 2018.
8. [3] Mohd Khaled Shambour, Adnan Gutub, “Progress of IoT research technologies and applicationsserving Hajj and Umrah”. 9th ed. Arabian Journal for Science and Engineering, 2022.
9. [4] Peter Haggar, “Practical Java: programming language guide”. 3rd ed. Addison-Wesley Professional, 2000.