# Medical Cabinet Project Report

## 1. Introduction

The "Medical Cabinet Project" is a Java-based application designed to facilitate the management of medical practices. It aims to streamline patient records, appointments, and staff information, offering a user-friendly interface for medical professionals to handle administrative tasks efficiently. By automating routine processes, this project addresses the growing need for digital solutions in healthcare administration.

## 2. Needs Analysis

### Functional Requirements:

- **Patient Management**: Ability to add, update, and delete patient records.
- **Appointment Scheduling**: Management of appointment bookings and cancellations.
- **Medical Records**: Secure storage and retrieval of patient medical histories.
- **Staff Management**: Tracking of doctors and administrative personnel.

### Non-Functional Requirements:

- **Usability**: Simple and intuitive interface for non-technical users.
- **Performance**: Ensure fast retrieval and updates of records.
- **Security**: Protect sensitive medical and personal data through encryption.
- **Scalability**: Handle increasing numbers of patients and appointments.

## 3. Design

## 3.1 Use Cases:

### *Use Case 1: Adding a Patient*

- **Actors**: Receptionist
- **Steps**: Receptionist inputs patient details (e.g., name, age, medical history) into the system.
- **Expected Outcome**: A new patient record is created and stored securely.

### *Use Case 2: Scheduling an Appointment*

- **Actors**: Receptionist, Patient
- **Steps**: Receptionist selects a doctor and an available time slot for the patient.
- **Expected Outcome**: Appointment is added to the schedule.

## 3.2 Class Diagram:

The project consists of the following primary classes:

- **Patient**: Handles patient information (name, age, medical history).
- **Appointment**: Manages appointment details (date, time, doctor).
- **Doctor**: Tracks doctor information (specialty, availability).
- **MedicalRecord**: Stores patient medical history.
- **Staff**: Handles administrative staff details.

## 3.3 System Architecture:

The system follows a modular design pattern, dividing functionality into distinct modules:

- **Data Layer**: Manages database connectivity and CRUD operations.
- **Logic Layer**: Implements business logic and validations.
- **Presentation Layer**: Provides a graphical or command-line interface for users.

**4. Development**

## 4.1 Tools and Technologies:

- **Programming Language**: Java
- **IDE**: IntelliJ IDEA

- **Database**: MySQL
- **Version Control**: GitHub (repository: Medical Cabinet Project)
- **Libraries Used**: Hibernate for ORM, JavaFX for GUI.

## 4.2 Code Overview:

### *Key Features:*

- **Patient Class**:
  - Fields: `id`, `name`, `age`, `medicalHistory`.
  - Methods: `addPatient()`, `updatePatient()`, `deletePatient()`.
- **Appointment Class**:
  - Fields: `appointmentId`, `date`, `time`, `doctorId`, `patientId`.
  - Methods: `scheduleAppointment()`, `cancelAppointment()`.
- **Doctor Class**:
  - Fields: `id`, `name`, `specialty`, `availability`.
  - Methods: `getAvailableDoctors()`.

## 4.3 Challenges and Solutions:

- **Data Security**: Implemented encryption for sensitive data using Java's cryptography library.
- **Scalability**: Used efficient data structures (e.g., hash maps) to ensure fast lookups.
- **Error Handling**: Incorporated robust exception handling to ensure smooth user experience.

**5. Features**

## Key Functionalities:

1. **Patient Management**:
   a. Add, update, and delete patient records.
   b. Search functionality to retrieve patient details quickly.
2. **Appointment Scheduling**:
   a. Supports recurring and one-time appointments.
   b. Conflict detection to prevent overlapping schedules.
3. **Doctor Management**:
   a. Displays available doctors based on specialty and time slots.

4. **Reports Generation**:
    a. Generates daily, weekly, and monthly appointment summaries.
5. **User Authentication**:
    a. Secure login system for staff and administrators.

## 6. Conclusion

The "Medical Cabinet Project" successfully addresses the challenges of managing patient records, appointments, and staff details in medical practices. By automating routine tasks, it reduces administrative overhead and enhances operational efficiency. The project is scalable, secure, and user-friendly, making it a valuable tool for healthcare professionals. Future enhancements could include:

- Integration with external healthcare APIs for broader interoperability.
- Mobile application development for on-the-go access.
- Advanced analytics for patient and appointment trends.

This report outlines the technical and functional aspects of the project, demonstrating its alignment with the specified requirements.