UNIVERSITY OF NORTH CAROLINA
CHARLOTTE

Project 2

Electronic Medical Record System Database based on Dermatology Clinic

Team members

Anushka Santosh Padyal – 801379909
Nishant Acharekar – 801363902
Shivangi Saxena – 801372350
Bulbul Roy – 801365911

Contents

Introduction

In healthcare, the emergence of digital technologies has brought about a revolutionary change in the delivery and management of patient care. To encompass essential aspects of EMR systems, we designed the 'Electronic Medical Record System Database based on the Dermatology Clinic project. This database system has been designed to streamline the processes involved in a dermatology clinic and promises to improve the efficiency of managing patient records, scheduling appointments, and tracking treatments. The system ensures a seamless integration of clinical operations by meticulously structuring tables to capture patient information, staff credentials, insurance details, and transaction records. Every effort is made to make this project error-free and efficient, ensuring the robustness of the system.

Functional Requirements

1. Patient Management
   a. The system shall allow the registration of new patients, including capturing personal and insurance information.
   b. The system shall provide the ability to update and maintain patient records.

2. Appointment Scheduling
   a. The system shall enable staff to schedule, reschedule, and cancel appointments for patients.

3. Staff Management
   a. The system shall maintain staff records, including personal details, professional credentials, and employment information.
   b. The system shall support the assignment of roles and responsibilities to staff members.

4. Insurance Handling
   a. The system shall record and track insurance details and verify insurance coverage for treatments.
   b. The system shall process insurance claims and handle billing procedures.

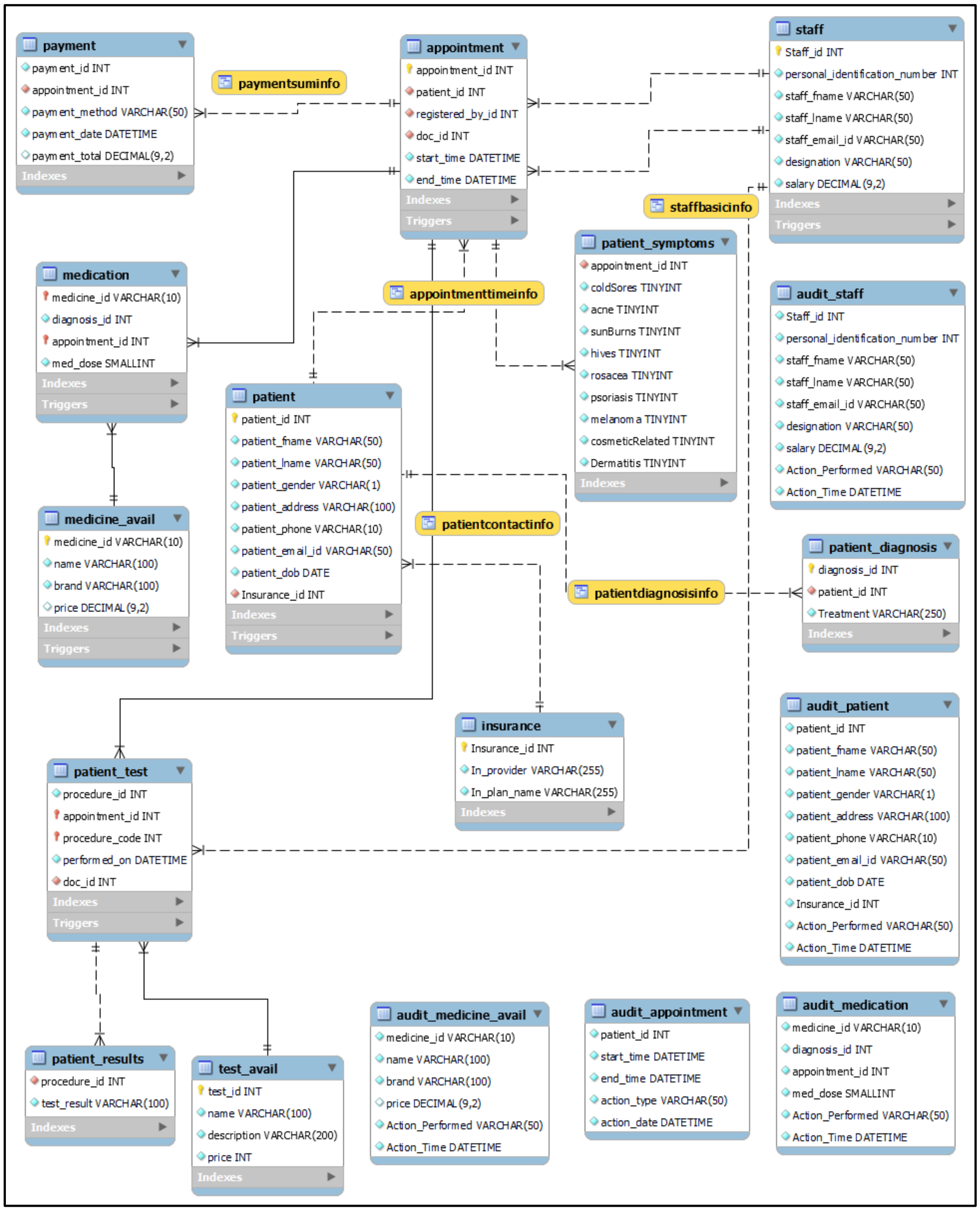
5. Medical Record Keeping
   a. The system shall allow doctors to enter and access patient medical records, including dermatological treatment history.
   b. The system shall ensure the confidentiality and security of patient medical records.

6. Payment Processing
   a. The system shall support various payment methods and process payments for services rendered.
   b. The system shall generate and provide detailed invoices to patients.

7. Reporting
   a. The system shall generate reports on patient visits, financial transactions, and staff performance.
   b. The system shall provide analytics for clinic performance and patient demographics.
8. Security and Compliance
   a. The system shall enforce user authentication and authorization for system access.
   b. The system shall comply with relevant healthcare regulations and data protection laws.

Entity Relationship

1. E-R diagram



The database is structured to facilitate the storage and management of various dermatological health-related data efficiently. It comprises multiple tables, each designed to handle specific aspects of patient care and clinic operations. The tables are interconnected using foreign key relationships to ensure data integrity and consistency.

Patient Table
Stores patient details including first name, last name, gender, address, phone number, email, date of birth, and insurance ID.

Staff Table
Manages information about clinic staff such as identification number, first name, last name, email, designation, and salary.

Insurance Table
Records insurance details including provider and plan name.

Appointment Table
Handles appointment scheduling with details like appointment ID, patient ID, registering staff ID, doctor ID, start time and end time.

Payment Table
Manages payment details for appointments, including payment ID, appointment ID, payment method, payment date, and total amount.

Patient symptoms Table
Captures patient symptoms related to various skin conditions identified by appointment ID.

Patient diagnosis Table
Records diagnosis details for patients, including diagnosis ID, patient ID, and treatment prescribed.

Medication Table
Tracks prescribed medications for patients with medicine ID, diagnosis ID, appointment ID, and medication dose.

Medicine avail Table
Manages available medications with details like medicine ID, name, brand, and price.

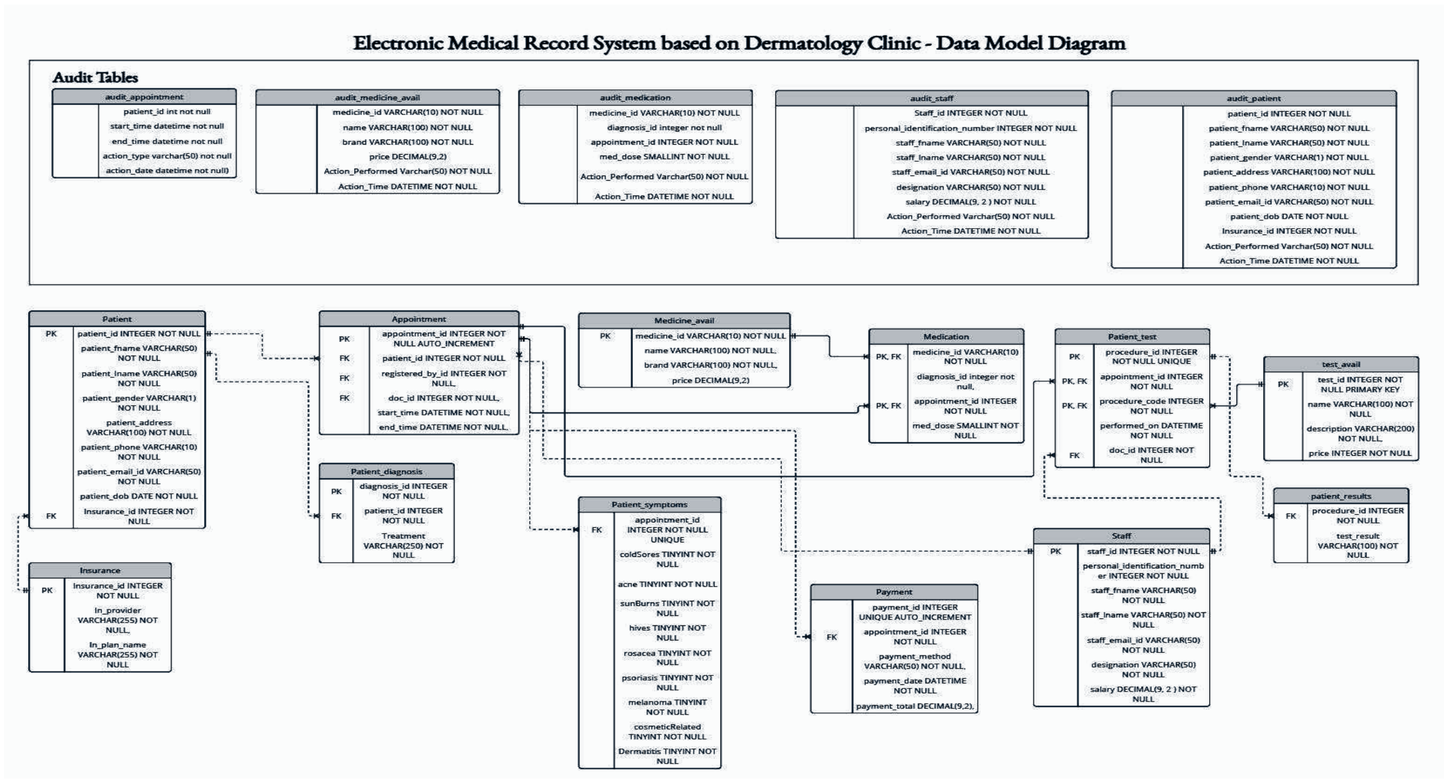
Test avail Table
Lists available tests or procedures with a unique test ID, name, description, and price.

Patient test Table
Records tests or procedures performed for patients with procedure ID, appointment ID, procedure code, performed date, and doctor ID.

Patient results Table: Stores test results for patients with procedure ID and test result details.

Audit Tables
Maintains audit logs for appointments, patients, staff, medication availability, and medication changes with action performed and timestamp.

2. UML diagram


Electronic Medical Record System based on Dermatology Clinic - Data Model Diagram
Audit Tables
audit_appointment
patient_id int not null
start_time datetime not null
end_time datetime not null
action_type varchar(50) not null
action_date datetime not null)
audit_medicine_avail
medicine_id VARCHAR(10) NOT NULL
name VARCHAR(100) NOT NULL
brand VARCHAR(100) NOT NULL
price DECIMAL(9,2)
Action_Performed Varchar(50) NOT NULL
Action_Time DATETIME NOT NULL
audit_medication
medicine_id VARCHAR(10) NOT NULL
diagnosis_id integer not null
appointment_id INTEGER NOT NULL
med_dose SMALLINT NOT NULL
Action_Performed Varchar(50) NOT NULL
Action_Time DATETIME NOT NULL
audit_staff
Staff_id INTEGER NOT NULL
personal_identification_number INTEGER NOT NULL
staff_fname VARCHAR(50) NOT NULL
staff_lname VARCHAR(50) NOT NULL
staff_email_id VARCHAR(50) NOT NULL
designation VARCHAR(50) NOT NULL
salary DECIMAL(9, 2) NOT NULL
Action_Performed Varchar(50) NOT NULL
Action_Time DATETIME NOT NULL
audit_patient
patient_id INTEGER NOT NULL
patient_fname VARCHAR(50) NOT NULL
patient_lname VARCHAR(50) NOT NULL
patient_gender VARCHAR(1) NOT NULL
patient_address VARCHAR(100) NOT NULL
patient_phone VARCHAR(10) NOT NULL
patient_email_id VARCHAR(50) NOT NULL
patient_dob DATE NOT NULL
Insurance_id INTEGER NOT NULL
Action_Performed Varchar(50) NOT NULL
Action_Time DATETIME NOT NULL
Patient
PK
patient_id INTEGER NOT NULL
patient_fname VARCHAR(50) NOT NULL
patient_lname VARCHAR(50) NOT NULL
patient_gender VARCHAR(1) NOT NULL
patient_address VARCHAR(100) NOT NULL
patient_phone VARCHAR(10) NOT NULL
patient_email_id VARCHAR(50) NOT NULL
patient_dob DATE NOT NULL
FK
Insurance_id INTEGER NOT NULL
Appointment
PK
appointment_id INTEGER NOT NULL AUTO_INCREMENT
FK
patient_id INTEGER NOT NULL
FK
registered_by_id INTEGER NOT NULL,
FK
doc_id INTEGER NOT NULL,
start_time DATETIME NOT NULL,
end_time DATETIME NOT NULL,
Medicine_avail
PK
medicine_id VARCHAR(10) NOT NULL
name VARCHAR(100) NOT NULL,
brand VARCHAR(100) NOT NULL,
price DECIMAL(9,2)
Medication
PK, FK
medicine_id VARCHAR(10) NOT NULL
diagnosis_id integer not null,
PK, FK
appointment_id INTEGER NOT NULL
med_dose SMALLINT NOT NULL
Patient_test
PK
procedure_id INTEGER NOT NULL UNIQUE
PK, FK
appointment_id INTEGER NOT NULL
PK, FK
procedure_code INTEGER NOT NULL
performed_on DATETIME NOT NULL
FK
doc_id INTEGER NOT NULL
test_avail
PK
test_id INTEGER NOT NULL PRIMARY KEY
name VARCHAR(100) NOT NULL
description VARCHAR(200) NOT NULL,
price INTEGER NOT NULL
Patient_diagnosis
PK
diagnosis_id INTEGER NOT NULL
FK
patient_id INTEGER NOT NULL
Treatment VARCHAR(250) NOT NULL
Patient_symptoms
FK
appointment_id INTEGER NOT NULL UNIQUE
coldSores TINYINT NOT NULL
acne TINYINT NOT NULL
sunBurns TINYINT NOT NULL
hives TINYINT NOT NULL
rosacea TINYINT NOT NULL
psoriasis TINYINT NOT NULL
melanoma TINYINT NOT NULL
cosmeticRelated TINYINT NOT NULL
Dermatitis TINYINT NOT NULL
patient_results
FK
procedure_id INTEGER NOT NULL
test_result VARCHAR(100) NOT NULL
Staff
PK
staff_id INTEGER NOT NULL
personal_identification_number INTEGER NOT NULL
staff_fname VARCHAR(50) NOT NULL
staff_lname VARCHAR(50) NOT NULL
staff_email_id VARCHAR(50) NOT NULL
designation VARCHAR(50) NOT NULL
salary DECIMAL(9, 2) NOT NULL
Payment
payment_id INTEGER UNIQUE AUTO_INCREMENT
FK
appointment_id INTEGER NOT NULL
payment_method VARCHAR(50) NOT NULL,
payment_date DATETIME NOT NULL
payment_total DECIMAL(9,2),
Insurance
PK
Insurance_id INTEGER NOT NULL
In_provider VARCHAR(255) NOT NULL,
In_plan_name VARCHAR(255) NOT NULL

Proof of BCNF

The tables are in BCNF if they are in 1NF, 2NF, and 3NF form and if they adhere to the below rules –

1. Each table has a primary key that uniquely identifies each record.
2. There are no attributes in a table that are dependent on only a part of a composite primary key.
3. There are no attributes in a table that are dependent on non-key attributes.
4. All columns that determine other columns are candidate keys themselves. This means that every non-trivial functional dependency in the table is of the form **key -> attribute**.

Below is a brief explanation of the tables with their relational schema and their functional dependencies –

1. Patient (patient_id, patient_fname, patient_lname, patient_gender, patient_address, patient_phone, patient_email_id, patient_dob, Insurance_id)

   Primary key dependency:
   patient_id -> patient_fname, patient_lname, patient_gender, patient_address, patient_phone, patient_email_id, patient_dob, Insurance_id

   Foreign key dependency:
   Insurance_id -> Insurance details
   The details of the insurance would be defined in the Insurance table referenced by Insurance_id.

   There are no non-trivial functional dependencies other than the primary key.
   The primary key is patient_id, and all other attributes are dependent on it.
   Hence, the Patient table satisfies BCNF.

2. Staff (Staff_id, personal_identification_number, staff_fname, staff_lname, staff_email_id, designation, salary)

   Primary key dependency:

Staff_id -> (personal_identification_number, staff_fname, staff_lname, staff_email_id, designation, salary)

Similar to the Patient table, all attributes are dependent on the primary key Staff_id. Hence, the Staff table satisfies BCNF.

3. Appointment(appointment_id, patient_id, registered_by_id, doc_id, start_time, end_time)

   Primary key dependency:
   appointment_id -> (patient_id, registered_by_id, doc_id, start_time, end_time)

   The attributes patient_id, registered_by_id, and doc_id are foreign keys and are thus not considered in the FDs. The attributes start_time and end_time are dependent on the primary key appointment_id. Hence, the Appointment table satisfies BCNF.

4. Patient_symptoms(appointment_id, coldSores, acne, sunBurns, hives, rosacea, psoriasis, melanoma, cosmeticRelated, Dermatitis)

   Primary key dependency:
   appointment_id -> (coldSores, acne, sunBurns, hives, rosacea, psoriasis, melanoma, cosmeticRelated, Dermatitis)

   All attributes are dependent on the primary key appointment_id. Hence, the Patient_symptoms table satisfies BCNF.

5. Patient_test(procedure_id, appointment_id, procedure_code, performed_on, doc_id)

   Composite Primary key dependency:
   (appointment_id, procedure_code) -> (procedure_id, performed_on, doc_id)

   All attributes are dependent on the composite primary key (appointment_id, procedure_code). Hence, the Patient_test table satisfies BCNF.

6. Medication(medicine_id, diagnosis_id, appointment_id, med_dose)

   Composite Primary key dependency:
   (appointment_id, medicine_id) -> (diagnosis_id, med_dose)

   All attributes are dependent on the composite primary key (appointment_id, medicine_id). Hence, the Medication table satisfies BCNF.

7. Payment(payment_id, appointment_id, payment_method, payment_date, payment_total)

   Primary key dependency:
   payment_id -> (appointment_id, payment_method, payment_date, payment_total)

   All attributes are dependent on the primary key payment_id. Hence, the Payment table satisfies BCNF.

8. Insurance(Insurance_id, In_provider, In_plan_name)

   Primary key dependency:
   Insurance_id -> (In_provider, In_plan_name)

   The primary key is Insurance_id, and all other attributes are dependent on it. Hence, the Insurance table satisfies

BCNF.

9. Patient_diagnosis(diagnosis_id, patient_id, Treatment)

   Primary key dependency:
   diagnosis_id -> (patient_id, Treatment)

   All attributes are dependent on the primary key diagnosis_id. Hence, the Patient_diagnosis table satisfies BCNF.

10. Medicine_avail(medicine_id, name, brand, price)

    Primary key dependency:
    medicine_id -> (name, brand, price)

    All attributes are dependent on the primary key medicine_id. Hence, the Medicine_avail table satisfies BCNF.

11. test_avail(test_id, name, description, price)

    Primary key dependency:
    {test_id -> (name, description, price)

    All attributes are dependent on the primary key test_id. Hence, the test_avail table satisfies BCNF.

12. patient_results(procedure_id, test_result)

    Foreign key dependency:
    procedure_id ->(test_result)

    All attributes are dependent on the primary key procedure_id. Hence, the patient_results table satisfies BCNF.

13. For the Audit Tables:
    All the audit tables have a non-trivial functional dependency on the primary key of the respective tables they are auditing. The action performed and action time are dependent on the primary keys of the respective tables. Therefore, they also satisfy BCNF.

Table Information

Table Name	**Column Names**	**Attributes**
Patient	patient_id	INTEGER
	patient_fname	VARCHAR
	patient_lname	VARCHAR
	patient_gender	VARCHAR
	patient_address	VARCHAR
	patient_phone	VARCHAR
	patient_email_id	VARCHAR
	patient_dob	DATE
	Insurance_id	INTEGER
Staff	Staff_id	INTEGER
	personal_identification_number	INTEGER
	staff_fname	VARCHAR
	staff_lname	VARCHAR
	staff_email_id	VARCHAR
	designation	VARCHAR
	salary	DECIMAL
Insurance	Insurance_id	INTEGER
	In_provider	VARCHAR
	In_plan_name	VARCHAR
Appointment	appointment_id	INTEGER
	patient_id	INTEGER
	registered_by_id	INTEGER
	doc_id	INTEGER
	start_time	DATETIME
	end_time	DATETIME
Payment	payment_id	INTEGER
	appointment_id	INTEGER
	payment_method	VARCHAR
	payment_date	DATETIME
	payment_total	DECIMAL
Patient_symptoms	appointment_id	INTEGER
	coldSores	TINYINT
	acne	TINYINT
	sunBurns	TINYINT
	hives	TINYINT
	rosacea	TINYINT
	psoriasis	TINYINT
	cosmeticRelated	TINYINT
	melanoma	TINYINT
	Dermatitis	TINYINT

Patient_diagnosis	diagnosis_id	INTEGER
	patient_id	INTEGER
	Treatment	VARCHAR
Medication	medicine_id	INTEGER
	diagnosis_id	INTEGER
	appointment_id	INTEGER
	med_dose	SMALLINT
Medicine_avail	medicine_id	VARCHAR
	name	VARCHAR
	brand	VARCHAR
	price	DECIMAL
test_avail	test_id	INTEGER
	name	VARCHAR
	description	VARCHAR
	price	INTEGER
Patient_test	procedure_id	INTEGER
	appointment_id	INTEGER
	procedure_code	INTEGER
	performed_on	DATETIME
	doc_id	INTEGER
patient_results	procedure_id	INTEGER
	test_result	VARCHAR
audit_appointment	patient_id	INTEGER
	start_time	DATETIME
	end_time	DATETIME
	action_type	VARCHAR
	action_date	DATETIME
audit_patient	patient_id	INTEGER
	patient_fname	VARCHAR
	patient_lname	VARCHAR
	patient_gender	VARCHAR
	patient_address	VARCHAR
	patient_phone	VARCHAR
	patient_email_id	VARCHAR
	patient_dob	DATE
	Insurance_id	INTEGER
	Action_Performed	VARCHAR
	Action_Time	DATETIME
audit_staff	Staff_id	INTEGER

	personal_identification_number	INTEGER
	staff_fname	VARCHAR
	staff_lname	VARCHAR
	staff_email_id	VARCHAR
	designation	VARCHAR
	salary	DECIMAL
	Action_Performed	VARCHAR
	Action_Time	DATETIME
audit_medicine_avail	medicine_id	VARCHAR
	name	VARCHAR
	brand	VARCHAR
	price	DECIMAL
	Action_Performed	VARCHAR
	Action_Time	DATETIME
audit_medication	medicine_id	VARCHAR
	diagnosis_id	INTEGER
	appointment_id	INTEGER
	med_dose	SMALLINT
	Action_Performed	VARCHAR
	Action_Time	DATETIME

Stored Procedure Information

There are several stored procedure created and we are using four of them i.e getAllPatientNames, PatientInsert, patientUpdate, and delete_patientInfo in our project.

Procedure Names	Description
TestAvailInsert	Inserts data into the test_avail table, typically for adding test availability records.
StaffInsert	to insert staff data in staff table
insert_medicaiton	to insert data into medication table
AppointmentInsert	to insert data appointment table.
PatientInsert	to Insert data into patient table.
pid_match_testing	Stored Procedure for retrieving a patient from the system
AppointmentUpdate	using the Stored Procedure we are updating the data in Appointment table
TestAvailUpdate	using the Stored Procedure we are Updating data in test_avail table.
patientUpdate	updating the patient table for address phone no. and insurance id using procedure.
update_medication	Updating medication table
StaffUpdate	To find update the staff table using stored procedure
update_PatientSymptoms	to update patient symptoms

delete_patientInfo	for deleting patient info from patient table
delete_StaffInformation	for deleting staff information
delete_Appointment	for deleting appointment
delete_testAvail	for deleting test availability in the clinic
delete_medciation	for deleting medication
delete_medAvail	for deleting a particular medicine avaialbility
paymentInfo	for selecting payment info of patient
patientSymptoms	for selecting patient symptoms
patientName	for selecting patient name from patient table
getAllPatientNames	for selecting patient name from patient table

User authentication/ Role-based access control

There are several user authentication role and we are using two of them i.e admin_staff and test&technical_staff in our project.

User	Description
doctor_staff	The user can • perform any operation on the Patient_test table. • view patient-related data, audit records related to patients,view patient diagnosis-related data. • perform any operation on the medication and audit_medication tables. • view patient results-related data. • The subsequent GRANT EXECUTE ON PROCEDURE statements provide the user account ‘doctor_staff’ with execution rights for specific stored procedures within the proj_dermatC_db database. • These procedures include update_PatientSymptoms, patientSymptoms, insert_medicaiton, delete_medciation, and update_medication. The user can execute these procedures to perform specific actions within the database.
receptionist_staff	The user can: • perform any operation on the patient and Appointment table. • view data from the Patient_Symptoms and staff table. • modify and view data from the payment table. • view audit-related data from the Audit_Patient table. • execute these procedures to perform specific actions within the database. • The subsequent GRANT EXECUTE ON PROCEDURE statements provide the user account ‘receptionist_staff’ with execution

	rights for specific stored procedures within the proj_dermatC_db database. • These procedures include StaffUpdate, StaffInsert, paymentInfo, pid_match_testing, patientName, patientInsert, patientUpdate, delete_StaffInformation, delete_patientInfo, delete_Appointment, AppointmentUpdate, AppointmentInsert, and getAllPatientNames.
nursing_staff	The user can: • perform any operation on the Patient_Symptoms table. • view patient data and audit records related to patients. • modify and view medication-related data. • view audit records related to medication,patient test-related data, and information about available medicines. • The subsequent GRANT EXECUTE ON PROCEDURE statements provide the user account ‘nursing_staff’ with execution rights for specific stored procedures within the proj_dermatC_db database.These procedures include update_PatientSymptoms, patientSymptoms, insert_medicaiton, delete_medciation, and update_medication.The user can execute these procedures to perform specific actions within the database.
admin_staff	All grants are supplied to administrative staff to ensure that the application is running properly.
test&technical_staff	The user can: • view audit-related data related to patients and appointments tables. • perform any operation on the patient_results, patient_test, and test_avail tables. • view medication-related data. • The subsequent GRANT EXECUTE ON PROCEDURE statements provide the user account ‘test&technical_staff’ with execution rights for specific stored procedures within the proj_dermatC_db database. These procedures include delete_testAvail, TestAvailInsert, and TestAvailUpdate. The user can execute these procedures to perform specific actions within the database.

Tables for audit trail (triggers)

Triggers Name	Description
after_UpdateAppointment	Creating a trigger for maintaining an audit record to keep track of a patient's appointment history
after_InsertAppointment	Creating a trigger for preserving an audit record to keep track of a patient's appointment change history.
after_DeleteAppointment	Creating a trigger for maintaining an audit record for keeping the Record/track of appointment changes history of a patient
after_InsertPatientInfo after_UpdatePatientInfo after_DeletePatientInfo	Creating a trigger for keeping an audit record of changes in patient information.
after_InsertStaffInfo after_UpdateStaffInfo after_DeleteStaffInfo	Creating a trigger for keeping an audit record of changes in patient information.
after_insertMedicineAvail after_updateMedicineAvail after_deleteMedicineAvail	Creating a trigger for establishing an audit record for the recording/tracking of changes in medicine availability information
after_insertMedication after_updateMedication after_deleteMedication	Creating a trigger for preserving an audit record for the record/track of changes in medication availability information.
InsertPaymentTable_medication	Insert into payment table using triggers from medicine and patient_test tables.
UpdatePaymentTable_medication	Use the medication table trigger to update data into payment.
InsertPaymentTable_test	Use the patient_test table to enter data into the payment.
UpdatePaymentTable_test	Update the data in the payment table using the patient_test.

Indexes

Indexes	Description
patient_information	Index to identify patients based on their first names.
patient_insurance_info	Index for identifying patients on patient insurance IDs
Doctor_info	Index for identifying physicians by name.
Doctor_designation_info	Index for identifying physicians based on designation.
Doctor_PIN_info	Index for identifying physicians based on unique identification numbers.
medicine_avail_info	Index for medicine availability by name.
test_avail_info	Index to get test availability information by name
patient_payment_info	Index to obtain payment information based on payment ID

Views

Views	Description
appointmentTimeInfo	View for appointment time information.
staffBasicInfo	View basic staff information.
patientContactInfo	View patient contact information.
paymentSumInfo	View for payment sum-to-dateinformation.
patientDiagnosisInfo	View for data on patients who have received specific diagnoses and medications for a certain diagnosis.

Tools Used

The database was designed and implemented using MySQL Workbench. The project files, including SQL scripts for table creation, triggers, stored procedures, sample data insertion, user profiles, and main test scripts, were utilized to develop and test the database.

We are using Jupyter notebook as our programming interface and developing our code in Python Programming language. You can refer to the instruction document provided in the project folder and additionally we have provided a readme file to follow the steps to execute the project.

References

1. https://www.w3schools.com/MySQL/default.asp
2. https://www.aad.org/public/fad/what-is-a-derm
3. https://ieeexplore.ieee.org/abstract/document/6498195
4. https://www.python.org/downloads/
5. https://jupyter.org/install
6. https://dev.mysql.com/downloads/workbench/