**Design documentaion**

|  |  |
| --- | --- |
| **Name** | **IDs** |
| **Youssef Rabeay** | **231001982** |
| **Hashem Yasser** | **231000770** |
| **Hussein Mohamed** | **231001746** |
| **Perihan Ashraf** | **231001189** |

**Project 4: database for medical laboratory system.**

1. **Er diagram:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

1. **Laboration :**

* **Laboration id(pk)**
* **First name**
* **Middle name**
* **Last name**
* **Phone**
* **Address**

1. **Patient:**

* **Patient id(pk)**
* **First name**
* **Middle name**
* **Last name**
* **Phone**
* **City**
* **Strret**
* **Country**
* **Birth date**
* **Job**

1. **MedicalTest:**

* **testID (PK)**
* **name**
* **price**

1. **Component:**

* **componentID (PK)**
* **componentname**
* **availabeQuantity**
* **MinQuantity**

**5.TestResult:**

* **resultID(PK)**
* **testID(FK→ MedicalTest)**
* **patientID (FK → Patient)**
* **laboratorianID (FK → Laboratorian)**
* **date**
* **result**

1. **MedicalTest\_component (Relationship Entity i.e between MedicalTest and Component using many-to-many):**

* **testID (PK, FK)**
* **componentID (PK, FK)**
* **quantityUsed**

**7.Patient\_component (Associative Entity for many-to-many between Patient and Component):**

* **patientID (PK, FK)**
* **componentID(PK, FK)**
* **takeDate (PK)**
* **quantityTaken.**

**Relationships:**

* **One TestResult relates to only one MedicalTest, one Patient, and one Laboratorian.**
* **one MedicalTest owns many Components each Component is used in many MedicalTest many to many with respect to MedicalTest\_component.**
* **One Patient takes many Components and Component can be taken by many Patients one-to-many relationship in Patient\_component**

1. **Explanation of Database Schema:**

**Laboratorian Table**

* **About lab technicians’ stores info**
* **Primary Key: laboratorianID**
* **Related to test result by laboration ID.**

**Patient Table**

* **Stores patient information**
* **Primary Key: patientID**
* **Related-to the test result and patient component table**

**TestResult Table**

* **Stores outputs of medical tests.**
* **PrimaryKey: resultID**
* **Foreign Keys: testID, patientID, laboratorianId**
* **Related-to medical test,patient,and laboration tables**

**Component Table**

* **Holds information about components used in tests.**
* **Primary Key: componentID**
* **Related to MedicalTest\_component and Patient\_component Tables.**

**MedicalTest\_component Table**

* **Many-to-many junction table of medical test and component**
* **Primary Key: (test ID component ID)**
* **Foreign Keys: testID and componentIDs**

**Patient\_component Table**

* **Junction table for many-to-many relationship between patient and component.**
* **Primary Key: (patientID, componentID, takeDate)**
* **Foreign Keys: patientID, componentID**

**3.Business Rules Implemented**

1. **A medical test has particular properties( many-to-many relationship)**
2. **Commands can be directly supplied by patients (Patient\_component completeness tracked)**
3. **Test results need to reference a test, patient and laboratorian that exist.**
4. **Component charges must be above the floor.**

**4.Design Decisions**

1. **Data Types: Instead of using INTEGER for flexibility with existing systems, VARCHAR used for IDs.**
2. **Junction Tables: Created a junction table for the many-to-many relationship to normalize the database.**
3. **Address Fields: New address fields broken out of city / street / country for improved reporting.**
4. **Demographic Analysis (Pateint table: included Job field in demographic analysis) .**
5. **Data Integrity: Foreign key constraints were enforced for data integrity employ.**

**5. Sample Queries:**

**Query 1 CBC tests of patients last year**

**SELECT p.firstName, p.lastName**

**FROM patient p**

**JOIN testResult tr ON p.patientID = tr.patientID**

**JOIN medicalTest mt ON tr.testID = mt.testID**

**WHERE mt.name = 'CBC'**

**AND tr.testDate BETWEEN DATE('now', '-1 year') AND DATE('now');**

* **Purpose: Identify patients who had complete blood count tests for follow-up care.**

**Query 2: Components with quantities below minimum**

**SELECT mt.name**

**FROM medicalTest mt**

**JOIN testResult tr ON mt.testID = tr.testID**

**WHERE tr.laboratorianID = ?;**

* **Purpose: List all tests performed by a specific laboratorian for performance evaluation.**

**DDL**

**create database LAB;**

**use LAB ;**

**create table if not exists laboratorian(**

**laboratorianID varchar (8) primary key,**

**firstName text,**

**middlename text,**

**lastname text,**

**phone Varchar (13),**

**address text**

**);**

**insert into laboratorian (laboratorianID) VALUES**

**('L001'),**

**('L002'),**

**('L003'),**

**('L004'),**

**('L005'),**

**('L006');**

**create table patient(**

**patientID varchar (9) primary key,**

**firstname text,**

**middlename text,**

**lastname text,**

**phone varchar (13),**

**city text ,**

**street text,**

**country text,**

**birthdate date,**

**job text**

**);**

**INSERT INTO patient (patientID) VALUES**

**('P001'),**

**('P002'),**

**('P003'),**

**('P004'),**

**('P005'),**

**('P006'),**

**('P007'),**

**('P008'),**

**('P009'),**

**('P010'),**

**('P011'),**

**('P012');**

**create table medicalTest(**

**testID varchar (9) primary key,**

**name text,**

**price int**

**);**

**insert into medicalTest (testID, name, price) values**

**('T001','CBC', 500),**

**('T002', 'AB', 400),**

**('T003','DIL', 300);**

**create table testResult(**

**resultID varchar (9) primary key,**

**testID varchar (9),**

**patientID varchar (9),**

**laboratorianID varchar (8),**

**foreign key (testID) references medicalTest(testID) ,**

**foreign key (patientID) references patient(patientID),**

**foreign key (laboratorianID) references laboratorian(laboratorianID),**

**testDate date,**

**result text**

**);**

**INSERT INTO testResult (resultID) VALUES**

**('R001'),**

**('R002'),**

**('R003'),**

**('R004'),**

**('R005'),**

**('R006'),**

**('R007');**

**create table component(**

**componentID varchar (9) PRIMARY KEY ,**

**componentname text,**

**availabeQuantity int ,**

**MinQuantity int**

**);**

**INSERT INTO component (componentID,componentname) values**

**('C001','serum'),**

**('C002','A.B'),**

**('C003','Anistetic');**

**-- Table for many-to-many relationship between MedicalTest and Component**

**CREATE TABLE MedicalTest\_component (**

**testID VARCHAR(9),**

**componentID VARCHAR(9),**

**quantityUsed INT,**

**PRIMARY KEY (testID, componentID),**

**FOREIGN KEY (testID) REFERENCES medicalTest(testID),**

**FOREIGN KEY (componentID) REFERENCES component(componentID)**

**);**

**-- Sample data for MedicalTest\_Component**

**INSERT INTO MedicalTest\_component (testID, componentID, quantityUsed) VALUES**

**('T001', 'C001', 2),**

**('T001', 'C002', 1),**

**('T002', 'C001', 1),**

**('T003', 'C003', 3);**

**-- Table for relationship between Patient and Component (Takes)**

**CREATE TABLE Patient\_component (**

**patientID VARCHAR(9),**

**componentID VARCHAR(9),**

**quantityTaken INT,**

**takeDate DATE,**

**PRIMARY KEY (patientID, componentID, takeDate),**

**FOREIGN KEY (patientID) REFERENCES patient(patientID),**

**FOREIGN KEY (componentID) REFERENCES component(componentID)**

**);**

**-- Sample data for Patient\_Component**

**INSERT INTO Patient\_component (patientID, componentID, quantityTaken, takeDate) VALUES**

**('P001', 'C001', 1, '2025-05-01'),**

**('P001', 'C002', 2, '2025-05-02'),**

**('P002', 'C001', 1, '2025-05-03');**

* **Bonus A working application/UI :**

**1. Database Setup**

**Function: create\_database()**

**SQLite database file named LAB is created.**

**Establish the tables with corresponding columns, data type and constraints:**

* **laboratorian**
* **patient**
* **medicalTest**
* **testResult**
* **component**

**Many-To-Multiple Relationship (to the junction table MedicalTest\_component ).**

**Patient\_component (junction table for many-to-many relationship)**

**Enforces Foreign Key and unique Constraints.**

2.Create, Read, Update, Delete) operations:

Insert Functions:

* insert\_laboratorian:

Validator on required fields with validation for type and data

* insert\_patient: Add a new patient with required fields validation and unique for phone.
* insert\_medical\_test: Inserts new medical test with price should be an integer.
* insert\_component(insert\_quantity): Insert a new component check quantity fields allowed.
* insert\_test\_result(): Adds a test result by foreign key ref.

Update Functions:

* update\_patient() : Update the patient details with validating input

**Delete Functions:**

* **delete\_laboratorian delete\_patient(), delete\_medical\_test(), delete\_component(), delete\_test\_result(): Removes records by ID with validation**

**Search Functions:**

* **search\_laboratorian(), search\_patient(), search\_medical\_test(), search\_component(), search\_test\_result(): Retrieves and displays records by ID.**

**3. Business Logic Queries**

**List CBC Patients (Last Year):**

* **list\_cbc\_patients(): Displays patients who underwent CBC tests in the past year using SQL joins and date filtering.**

**Low Quantity Components List:**

* **list\_low\_quantity\_components() Find components with available\_quantity below min threshold**

**Total Payments by Patient (Last 3 Years):**

**list\_total\_paid\_by\_patient(): Calculates the sum of test prices for a specific patient over three years.**

**4.Data Export**

**Function: export\_to\_csv(table\_name)**

**Exports table data to a CSV file with headers.**

**Supports tables: laboratorian, patient, medicalTest, component, testResult.**

**5.User Interface**

**Input Validation:**

**Ensures required fields are filled.**

**Validates data types (e.g., phone numbers as integers, dates in YYYY-MM-DD format).**

**Feedback:**

**Uses messagebox to show success/error messages (e.g., "Laboratorian added successfully!").**

1. **Libraries:**

* **tkinter: For GUI.**
* **sqlite3: For database operations.**
* **csv: For data export.**