**Patient Appointment System**

Project by: Casey Castaner, Thomas Payne, Mike Lucero and Gene Tosmuk

**Scope:**

Encompasses functionalities related to managing patient appointments, interacting with a MySQL database, providing a user interface for interaction and basic testing capabilities.

* Database Interaction:
  + The program connects to a MySQL database server to perform operations such as viewing appointments, patients, and doctors, as well as adding, updating, and canceling appointments.
* Appointment Management:
  + Users can view appointments stored in the database, including details such as appointment ID, patient ID, doctor ID, and appointment date.
  + Functionality is provided to add, edit, and cancel appointments.
* Patient Management:
  + Users can view patient records stored in the database, including details such as patient ID, first name, last name, date of birth, gender, contact information, and address.
  + Functionality is provided to add, edit, and remove patient records.
* Doctor Management:
  + Users can view doctor records stored in the database, including details such as doctor ID and name.
  + Functionality is provided to add, edit, and remove doctor records.
* Error Handling:
  + The program includes error handling mechanisms to handle database connection errors, query execution errors, and result set retrieval errors.
* User Interaction:
  + Users interact with the program through a command-line interface, where they are prompted to choose options such as viewing appointments, patients, or doctors, and performing corresponding actions.
* Testing:
  + The program includes basic testing functionality to verify the correctness of views (appointments, patients, doctors) by querying the database and displaying the results.

**Prerequisites and Dependencies:**

We’ll need the following installed

* Source-code Editor (Visual Studio Code)v
* GCC Compiler
* MySQL Server & C Connector
* Bash Shell

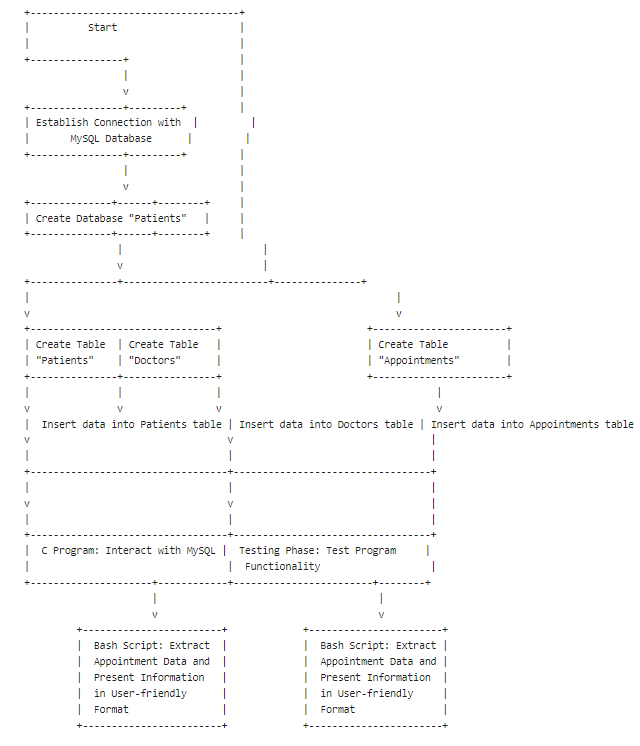
**Database Setup:**

We'll use MySQL to create tables for Appointments, Patients, and Doctors to organize our data efficiently.

Will allow the User to perform a variety of tasks like:

* List all patients for each Doctor sorted by Patient ID.
* List the number of Doctors sorted by Doctor ID.
* List the dates of appointments in order of date.
* Display a table of all Doctors, Patients and Appointments.

**Program Flow Chart:**

****

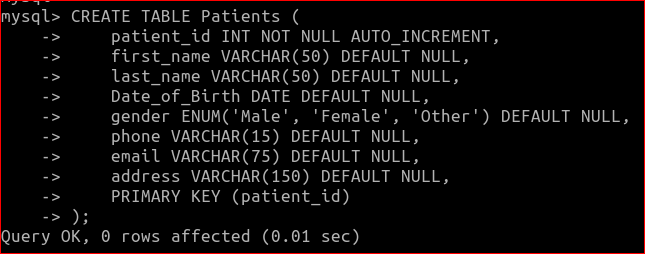
**MySQL Database:**

We'll develop functions in C to interact seamlessly with our MySQL database. This includes adding, modifying, deleting, and viewing appointments based on user input.

1. C code to find mysql in system

2. C code to create database "Patients" in mysql

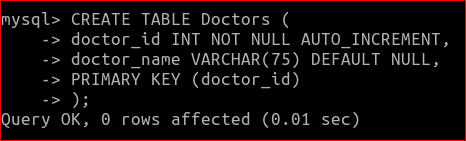
3. C Code to create table "Patients" in mysql database use existing Database (Patients) and below is the format.

****

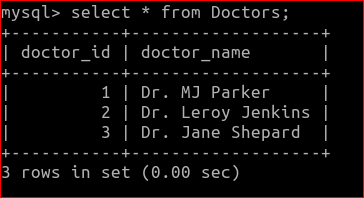
4. Insert data into the Patients Table using C code.

****

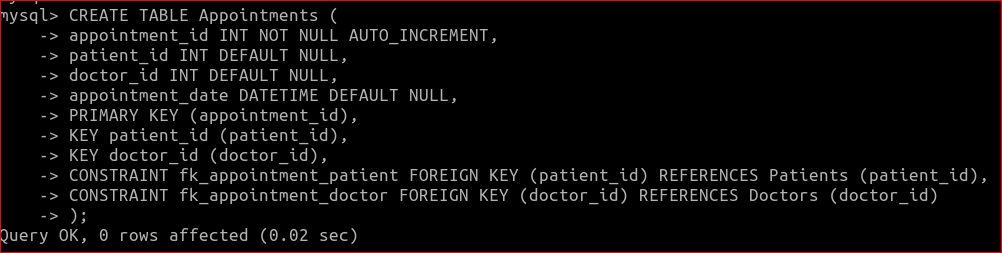
5. Write C code to create table "Doctors" in mysql database using existing Database (Doctors) and use below format.

****

6. Insert below data on the Doctors table using C code.

****

7. Write C code to create table "Appointments" in mysql database using existing Database Appointments and use below format.

****

8. Insert data on the Appointments table using the C code.

**Bash Scripting:**

Upon deploying this Bash Script, it extracts upcoming appointment data from the main C Program. Each segment of the script serves a unique function in interacting with the program. The initial segment grants root access to the program, followed by accessing MYSQL data in the subsequent part, and the remaining section formats and presents the requested information in a user-friendly table.

The script can then be run by command line by entering: chmod +x script\_name.sh and then ./script\_name.sh.

[Bash Script](https://drive.google.com/file/d/1_lpQXCLFZenu2rQHYBEdlynMpuMInkjw/view?usp=sharing)

#!/bin/bash

# Database connection details

DB\_HOST="localhost"

DB\_USER="root"

DB\_PASSWORD="student"

DB\_NAME="Patients"

# Function to execute a MySQL query

function execute\_query() {

local query="$1"

# Execute the query using MySQL command-line client

# -h: MySQL host

# -u: MySQL user

# -p: Password prompt (no space between -p and password)

# -e: Execute the specified SQL statement

# "$DB\_NAME": Specifies the database to use

if ! mysql -h "$DB\_HOST" -u "$DB\_USER" -p"$DB\_PASSWORD" -e "$query" "$DB\_NAME"; then

echo "Query execution error"

exit 1

fi

}

# Function to test appointments view

function AppointmentView() {

echo "Testing Appointments View..."

# Execute a query to select all records from the Appointments table

execute\_query "SELECT \* FROM AppointmentView"

}

# Function to test patients view

function PatientView() {

echo "Testing Patients View..."

# Execute a query to select all records from the Patients table

execute\_query "SELECT \* FROM PatientView"

}

# Function to test doctors view

function DoctorView() {

echo "Testing Doctors View..."

# Execute a query to select all records from the Doctors table

execute\_query "SELECT \* FROM DoctorView"

}

while true; do

# Display Main Menu

echo "What would you like to do? Enter a number"

echo "Enter 1 to enter appointments view"

echo "Enter 2 to enter patient view"

echo "Enter 3 to enter doctor view"

echo "Enter 4 to quit"

# Read user input

read -r choice

# Process user choice

case "$choice" in

1) AppointmentView;; # Test appointments view

2) PatientView;; # Test patients view

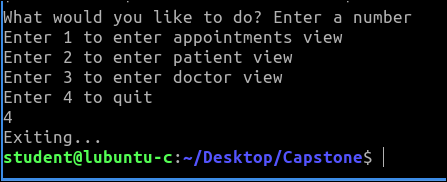
3) DoctorView;; # Test doctors view

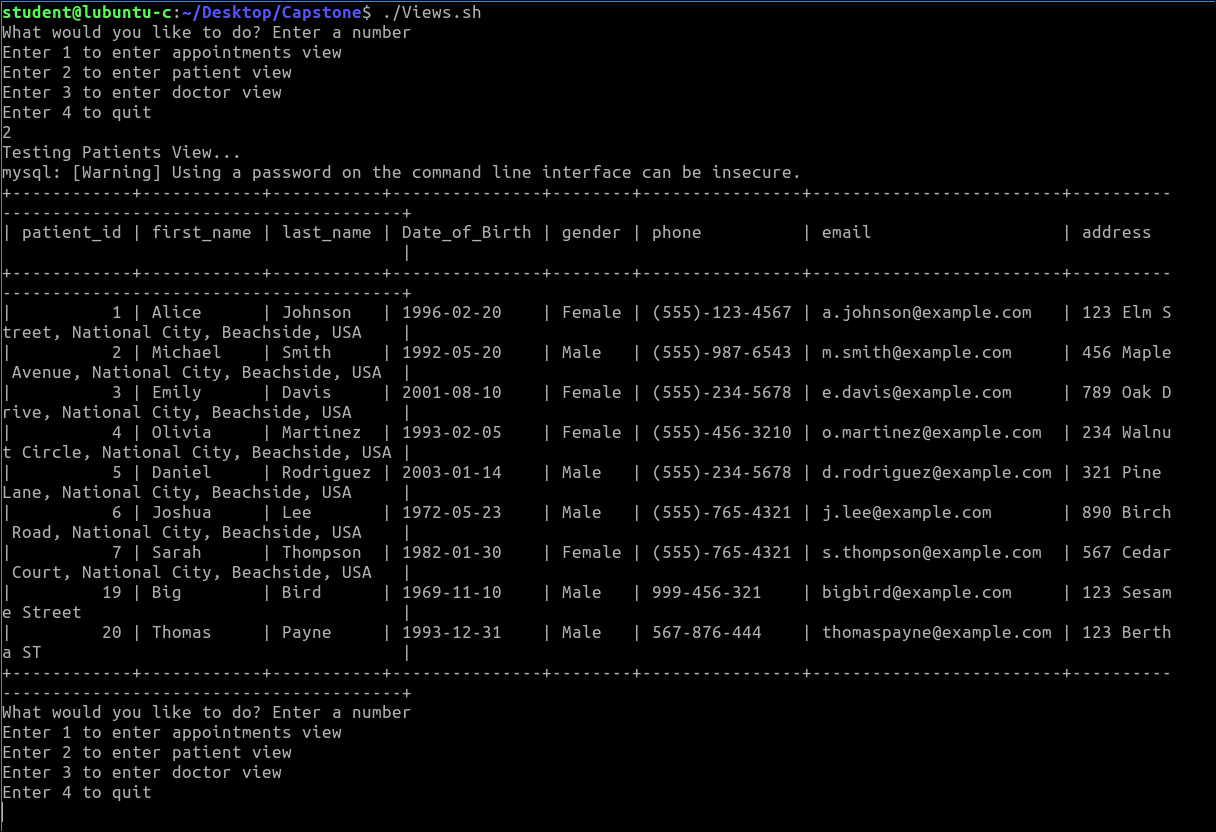
4) echo "Exiting..."; break;; # Exit

\*) echo "Invalid choice. Please try again.";;

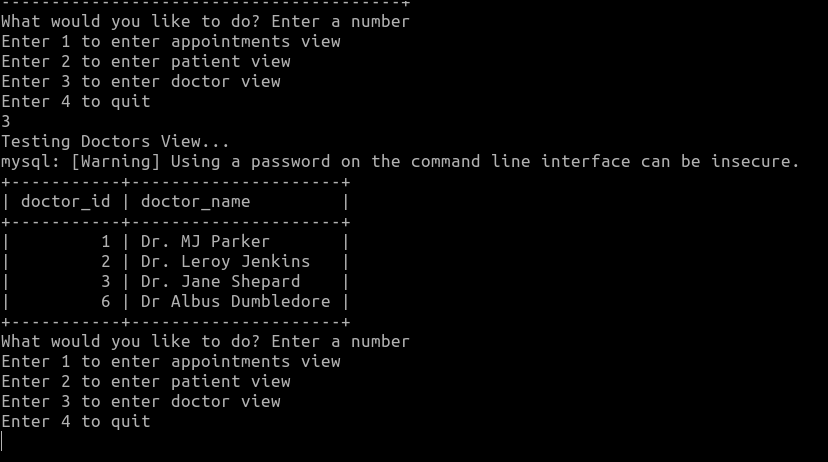
esac

done

**Database Views:** Executed by Bash Script

Patient View:

Doctor View:



**Integration:**

We'll integrate our C program with MySQL using MySQL Connector/C to establish connections and execute SQL queries. Additionally, we'll integrate Bash scripts with the C program to automate tasks such as sending reminders and updating appointment statuses.

The Makefile will help us integrate by automating the process of compiling our source codes and defining the dependencies.

[Makefile](https://drive.google.com/file/d/1DPMy0t1n41mWT9NTh8Rt3o9AO0s7dI4I/view?usp=sharing)

CC = gcc

CFLAGS = -Wall -Wextra -std=c99

LDFLAGS = -lmysqlclient

DB\_HOST = localhost

DB\_USER = root

DB\_PASSWORD = student

DB\_NAME = Patients

.PHONY: all clean test bash

all: debugCapstone Test Views

capstone: capstone.o

$(CC) $(CFLAGS) -o $@ $< $(LDFLAGS)

capstone.o: debugCapstone.c

$(CC) $(CFLAGS) -c -o $@ $<

test: Test

test\_program: Test.o

$(CC) $(CFLAGS) -o $@ $< $(LDFLAGS)

test.o: Test.c

$(CC) $(CFLAGS) -c -o $@ $<

bash:

Views.sh

clean:

rm -f capstone capstone.o test test.o

**Testing and Deployment:**

We'll rigorously test the system to ensure it functions correctly in various scenarios. Once tested, we'll deploy the system in the clinic or private practice environment and provide necessary training to users.

The program is testing the functionality of viewing data from our MySQL database tables related to Appointments, Patients and Doctors. Each test function ensures that the corresponding view in the database is functioning correctly by fetching the data and displaying it to the user. If any errors occur during the execution of SQL queries or fetching results, appropriate error messages are displayed, and the program exits with a failure status.

Test functions:

testAppointmentsView: This function executes a SQL query to retrieve all records from the Appointments table in the database. It then fetches each row of the result set and prints the Appointment ID, Patient ID, Doctor ID and Appointment Date.

testPatientsView: This function executes a SQL query to retrieve all records from the Patients table in the database. It fetches each row of the result set and prints the Patient ID, First Name, Last Name, Date of Birth, Gender, Phone Number, Email and Address.

testDoctorsView: This function executes a SQL query to retrieve all records from the Doctors table in the database. It fetches each row of the result set and prints the Doctor ID and Doctor Name.

**System Test Case:** MySQL Connection establishment with C Programming Code

Objective: To verify that the program successfully establishes a connection to the database.

Test Scenario: Attempt to establish a connection to the database using the program.

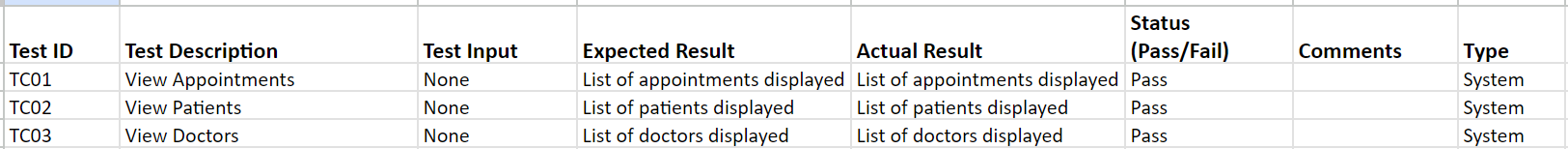
Expected Result: The program should establish a connection to the database without errors.

Actual Result: The program successfully establishes a connection to the database.

Status: Pass

Comments: The program was able to connect to the database as expected, indicating that the connection functionality is implemented correctly.

[Test Matrix](https://docs.google.com/spreadsheets/d/1uF1DXX9fqm_1X4uWbjXiApGJ5Hc-zPvcpuXhOFkj-VU/edit?usp=sharing)



[Test Code](https://drive.google.com/file/d/1FfwpOrA16A-msqPnbd71K8chTPJNEc7G/view?usp=sharing)

#include <stdio.h>

#include <stdlib.h>

#include <mysql/mysql.h>

// Define database connection parameters

#define DB\_HOST "localhost"

#define DB\_USER "root"

#define DB\_PASSWORD "student"

#define DB\_NAME "Patients"

MYSQL \*conn; // MySQL connection object

// Function to execute SQL queries

void execute\_query(const char \*query) {

if (mysql\_query(conn, query)) { // Execute the query

// If query execution fails, print error message and exit

fprintf(stderr, "Query execution error: %s\n", mysql\_error(conn));

mysql\_close(conn); // Close the connection

exit(EXIT\_FAILURE); // Exit the program

}

}

// Function to test the Appointments view

void testAppointmentsView() {

printf("Testing Appointments View...\n");

if (mysql\_query(conn, "SELECT \* FROM Appointments")) { // Execute SQL query

// If query execution fails, print error message and exit

fprintf(stderr, "Failed to execute query: Error: %s\n", mysql\_error(conn));

exit(EXIT\_FAILURE); // Exit the program

}

MYSQL\_RES \*result = mysql\_store\_result(conn); // Store the result set

if (result) {

MYSQL\_ROW row;

printf("Appointment ID\t| Patient ID\t| Doctor ID\t| Appointment Date\n");

while ((row = mysql\_fetch\_row(result))) { // Fetch each row from the result set

printf("%s\t| %s\t| %s\t| %s\n", row[0], row[1], row[2], row[3]); // Print row data

}

mysql\_free\_result(result); // Free the result set

} else {

// If failed to store result set, print error message

fprintf(stderr, "Failed to store result: Error: %s\n", mysql\_error(conn));

}

}

// Function to test the Patients view

void testPatientsView() {

printf("Testing Patients View...\n");

if (mysql\_query(conn, "SELECT \* FROM Patients")) { // Execute SQL query

// If query execution fails, print error message and exit

fprintf(stderr, "Failed to execute query: Error: %s\n", mysql\_error(conn));

exit(EXIT\_FAILURE); // Exit the program

}

MYSQL\_RES \*result = mysql\_store\_result(conn); // Store the result set

if (result) {

MYSQL\_ROW row;

printf("Patient ID\t| First name\t| Last name\t| DOB\t| Gender\t| Phone\t| E-Mail\t| Address\n");

while ((row = mysql\_fetch\_row(result))) { // Fetch each row from the result set

printf("%s\t| %s\t| %s\t| %s\t| %s\t| %s\t| %s\t| %s\n", row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7]); // Print row data

}

mysql\_free\_result(result); // Free the result set

} else {

// If failed to store result set, print error message

fprintf(stderr, "Failed to store result: Error: %s\n", mysql\_error(conn));

}

}

// Function to test the Doctors view

void testDoctorsView() {

printf("Testing Doctors View...\n");

if (mysql\_query(conn, "SELECT \* FROM Doctors")) { // Execute SQL query

// If query execution fails, print error message and exit

fprintf(stderr, "Failed to execute query: Error: %s\n", mysql\_error(conn));

exit(EXIT\_FAILURE); // Exit the program

}

MYSQL\_RES \*result = mysql\_store\_result(conn); // Store the result set

if (result) {

MYSQL\_ROW row;

printf("Doctor ID\t| Doctor Name\n");

while ((row = mysql\_fetch\_row(result))) { // Fetch each row from the result set

printf("%s\t| %s\n", row[0], row[1]); // Print row data

}

mysql\_free\_result(result); // Free the result set

} else {

// If failed to store result set, print error message

fprintf(stderr, "Failed to store result: Error: %s\n", mysql\_error(conn));

}

}

int main() {

conn = mysql\_init(NULL); // Initialize MySQL connection object

if (conn == NULL) { // Check if initialization failed

// If initialization failed, print error message and exit

fprintf(stderr, "mysql\_init () failed\n");

return EXIT\_FAILURE; // Exit the program

}

if (mysql\_real\_connect(conn, DB\_HOST, DB\_USER, DB\_PASSWORD, NULL, 0, NULL, 0) == NULL) {

// Connect to MySQL server

// If connection failed, print error message and exit

fprintf(stderr, "mysql\_real\_connect () failed\n");

mysql\_close(conn); // Close the connection

return EXIT\_FAILURE; // Exit the program

}

// USE Patient DB

execute\_query("USE Patients"); // Use the Patients database

int choice;

// Main loop

while (1) {

// Prompt user for choice

printf("\n");

printf("What would you like to do? Enter a number\n");

printf("Enter 1 to enter appointments view\n");

printf("Enter 2 to enter patient view\n");

printf("Enter 3 to enter doctor view\n");

printf("Enter 4 to quit\n");

scanf("%d", &choice); // Read user choice

switch (choice) {

case 1:

testAppointmentsView(); // Test appointments view

break;

case 2:

testPatientsView(); // Test patients view

break;

case 3:

testDoctorsView(); // Test doctors view

break;

case 4:

printf("Exiting...\n"); // Quit

break;

default:

printf("Invalid choice. Please try again.\n"); // Invalid choice

}

if (choice == 4) { // Check if user chose to quit

break; // Exit the loop

}

}

mysql\_close(conn); // Close MySQL connection

return 0; // Exit the program

}

**C Program:**

This C Program serves as a simple command-line interface for managing a patient appointment database stored in a MySQL database. It provides basic CRUD operations for appointments, patients and doctors.

Code: Accomplishes the following tasks

* Establishes a connection to a MySQL database using the provided host, username, password, and database name.
* Present a menu to the user with these options:
  + View Appointments, Patients or Doctors
  + Alter (Update/Delete) Appointments, Patients or Doctors
  + Add new Appointments, Patients or Doctors
  + Quit the program
* Executes SQL queries, depending on user’s choice, to perform selected operation on the corresponding database table
* Displays appropriate success or error messages based on the outcome of SQL queries

[C Code](https://drive.google.com/file/d/1SRW52SA5lPU_EmpTxzeDONCdZlgucbLb/view?usp=sharing)

#include <mysql/mysql.h>

#include <stdio.h>

#include <stdlib.h>

#define DB\_HOST "localhost" // Database host

#define DB\_USER "root" // Database username

#define DB\_PASSWORD "student" // Database password

#define DB\_NAME "Patients" // Database name

MYSQL \*conn; // MySQL connection object

MYSQL\_RES \*res; // MySQL result object

MYSQL\_ROW row; // MySQL row object

// Function to execute a MySQL query

void execute\_query(const char \*query)

{

// Execute the query and check for errors

if (mysql\_query(conn, query))

{

// If there's an error, print the error message and exit the program

fprintf(stderr, "Query execution error: %s\n", mysql\_error(conn));

mysql\_close(conn);

exit(EXIT\_FAILURE);

}

}

int main()

{

// Initialize MySQL connection

conn = mysql\_init(NULL);

// Check if MySQL connection initialization was successful

if (conn == NULL)

{

// If initialization failed, print an error message and return EXIT\_FAILURE

fprintf(stderr, "mysql\_init () failed\n");

return EXIT\_FAILURE;

}

// Connect to MySQL server

if (mysql\_real\_connect(conn, DB\_HOST, DB\_USER, DB\_PASSWORD, NULL, 0, NULL, 0) == NULL)

{

// If connection fails, print an error message, close the connection, and return EXIT\_FAILURE

fprintf(stderr, "mysql\_real\_connect () failed\n");

mysql\_close(conn);

return EXIT\_FAILURE;

}

// Select the database to work with

execute\_query("USE Patients");

int choice;

// Main loop to present options to the user

while (1)

{

// Prompt the user for their choice

printf("What would you like to do? Enter a number\n");

printf("Enter 1 to enter appointments view\n");

printf("Enter 2 to enter patient view\n");

printf("Enter 3 to enter doctor view\n");

printf("Enter 4 to quit\n");

scanf("%d", &choice);

// Switch statement based on the user's choice

switch (choice)

{

case 1:

// View appointments

printf("Viewing Appointments\n");

// Execute query to select all appointments

if(mysql\_query(conn, "SELECT \* FROM Appointments")) {

fprintf(stderr, "Failed to execute query: Error: %s\n", mysql\_error(conn));

exit(1);

}

// Store the result of the query

res = mysql\_store\_result(conn);

// Print the result in a tabular format

printf("appointment\_id\t| patient\_id\t| doctor\_id\t| appointment\_date\n");

while ((row = mysql\_fetch\_row(res)))

{

printf("%s\t| %s\t| %s\t| %s\n", row[0], row[1], row[2], row[3]);

}

// Free the result object

mysql\_free\_result(res);

// Prompt the user for further actions

printf("What would you like to do? Enter a number\n");

printf("Enter 1 to alter appointments\n");

printf("Enter 2 to cancel appointments\n");

printf("Enter 3 to set an appointment\n");

printf("Enter 4 to quit\n");

scanf("%d", &choice);

switch(choice){

case 1:

// Alter appointments

printf("Enter Appointment ID of appointment you wish to update\n");

int aid;

scanf("%d", &aid);

printf("Which column would you like to update?\n");

char update[20];

scanf(" %[^\n]", update);

printf("What is the new information?\n");

char newvalue[50];

scanf(" %[^\n]", newvalue);

// Construct and execute the update query

char query1[1000];

snprintf(query1, sizeof(query1), "UPDATE Appointments SET `%s` = '%s' WHERE appointment\_id = %d", update, newvalue, aid);

if(mysql\_query(conn, query1)){

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Appointment edited successfully!\n");

break;

case 2:

// Cancel appointments

printf("Enter Appointment ID of appointment you wish to cancel\n");

int aid\_cancel;

scanf("%d", &aid\_cancel);

// Construct and execute the delete query

char query\_cancel[1000];

snprintf(query\_cancel, sizeof(query\_cancel), "DELETE FROM Appointments WHERE appointment\_id = %d", aid\_cancel);

if(mysql\_query(conn, query\_cancel)){

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Appointment cancelled successfully!\n");

break;

case 3:

// Set an appointment

printf("Enter appointment details\nPatient ID:\n");

int pid;

scanf("%d", &pid);

printf("Enter Doctor ID:\n");

int did;

scanf("%d", &did);

printf("Enter Appointment Date (YYYY-MM-DD HH:MM:SS):\n");

char app\_date[12];

scanf(" %[^\n]", app\_date);

// Construct and execute the insert query

char query\_set[1000];

snprintf(query\_set, sizeof(query\_set), "INSERT INTO Appointments(patient\_id, doctor\_id, appointment\_date) VALUES ('%d', '%d', '%s')", pid, did, app\_date);

if (mysql\_query(conn, query\_set)) {

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Appointment inserted successfully!\n");

break;

}

break;

case 2:

// View patients

printf("Viewing patients\n");

// Execute query to select all patients

if (mysql\_query(conn, "SELECT \* FROM Patients")){

fprintf(stderr, "Failed to execute query: Error: %s\n", mysql\_error(conn));

exit(1);

}

// Store the result of the query

res = mysql\_store\_result(conn);

// Print the result in a tabular format

printf("patient\_id\t| first\_name\t| last\_name\t| Date\_of\_Birth\t| gender\t| phone\t| email\t| address\n");

while ((row = mysql\_fetch\_row(res))) {

printf("%s\t| %s\t| %s\t| %s\t| %s\t| %s\t| %s\t| %s\n", row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7]);

}

// Free the result object

mysql\_free\_result(res);

// Prompt the user for further actions

printf("What would you like to do? Enter a number\n");

printf("Enter 1 to alter patients\n");

printf("Enter 2 to close patient account\n");

printf("Enter 3 to add new patient\n");

printf("Enter 4 to quit: \n");

scanf("%d", &choice);

switch(choice){

case 1:

// Alter patients

printf("Enter Patient ID of the patient you wish to update\n");

int pid\_alter;

scanf("%d", &pid\_alter);

printf("Which column would you like to update?\n");

char update\_patient[20];

scanf(" %[^\n]", update\_patient);

printf("What is the new information?\n");

char newvalue\_patient[50];

scanf(" %[^\n]", newvalue\_patient);

// Construct and execute the update query

char query\_alter[1000];

snprintf(query\_alter, sizeof(query\_alter), "UPDATE Patients SET `%s` = '%s' WHERE patient\_id = %d", update\_patient, newvalue\_patient, pid\_alter);

if(mysql\_query(conn, query\_alter)){

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Patient edited successfully!\n");

break;

case 2:

// Close patient account

printf("Enter Patient ID of patient you wish to remove\n");

int pid\_delete;

scanf("%d", &pid\_delete);

// Construct and execute the delete query

char query\_delete[1000];

snprintf(query\_delete, sizeof(query\_delete), "DELETE FROM Patients WHERE patient\_id = %d", pid\_delete);

if(mysql\_query(conn, query\_delete)){

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Patient removed successfully!\n");

break;

case 3:

// Add new patient

printf("Enter patient details\nFirst name: \n");

char fn[20];

scanf("%s", fn);

printf("Enter Last Name:\n");

char ln[20];

scanf("%s", ln);

printf("\nEnter Date-of-Birth (YYYY-MM-DD):\n");

char dob[12];

scanf("%s", dob);

printf("\nEnter Gender:\n");

char gen[10];

scanf("%s", gen);

printf("\nEnter Phone Number:\n");

char pnum[15];

scanf("%s",pnum);

printf("\nEnter Email:\n");

char email[50];

scanf("%s", email);

printf("\nEnter Address:\n");

char address[50];

scanf(" %[^\n]", address);

// Construct and execute the insert query

char query\_add[1000];

snprintf(query\_add, sizeof(query\_add), "INSERT INTO Patients(first\_name, last\_name, Date\_of\_Birth, Gender, phone, email, address) VALUES ('%s', '%s', '%s', '%s', '%s', '%s', '%s')", fn, ln, dob, gen, pnum, email, address);

if (mysql\_query(conn, query\_add)) {

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Patient inserted successfully!\n");

break;

}

break;

case 3:

// View doctors

printf("Viewing Doctors\n");

// Execute query to select all doctors

if (mysql\_query(conn, "SELECT \* FROM Doctors")){

fprintf(stderr, "Failed to execute query: Error: %s\n", mysql\_error(conn));

exit(1);

}

// Store the result of the query

res = mysql\_store\_result(conn);

// Print the result in a tabular format

printf("\nDoctor ID\t| Doctor Name\n");

while ((row = mysql\_fetch\_row(res))){

printf("%s\t| %s\n", row[0], row[1]);

}

// Free the result object

mysql\_free\_result(res);

// Prompt the user for further actions

printf("\nWhat would you like to do? Enter a number\n");

printf("Enter 1 to alter doctors\n");

printf("Enter 2 to remove doctor\n");

printf("Enter 3 to add new doctor\n");

printf("Enter 4 to quit\n");

scanf("%d", &choice);

switch(choice){

case 1:

// Alter doctors

printf("Enter Doctor ID of the doctor you wish to update\n");

int did\_alter;

scanf("%d", &did\_alter);

printf("Which column would you like to update?\n");

char update\_doctor[20];

scanf(" %[^\n]", update\_doctor);

printf("What is the new information?\n");

char newvalue\_doctor[50];

scanf(" %[^\n]", newvalue\_doctor);

// Construct and execute the update query

char query\_alter\_doctor[1000];

snprintf(query\_alter\_doctor, sizeof(query\_alter\_doctor), "UPDATE Doctors SET `%s` = '%s' WHERE doctor\_id = %d",update\_doctor, newvalue\_doctor, did\_alter);

if(mysql\_query(conn, query\_alter\_doctor)){

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Doctor edited successfully!\n");

break;

case 2:

// Remove doctor

printf("Enter Doctor ID of the doctor you wish to remove\n");

int did\_delete;

scanf("%d", &did\_delete);

// Construct and execute the delete query

char query\_delete\_doctor[1000];

snprintf(query\_delete\_doctor, sizeof(query\_delete\_doctor), "DELETE FROM Doctors WHERE doctor\_id = %d", did\_delete);

if(mysql\_query(conn, query\_delete\_doctor)){

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Doctor removed successfully!\n");

break;

case 3:

// Add new doctor

printf("Enter doctor name (Dr First Last)\n");

char dn[20];

scanf(" %[^\n]", dn);

// Construct and execute the insert query

char query\_add\_doctor[1000];

snprintf(query\_add\_doctor, sizeof(query\_add\_doctor), "INSERT INTO Doctors(doctor\_name) VALUES ('%s')", dn);

if (mysql\_query(conn, query\_add\_doctor)) {

fprintf(stderr, "%s\n", mysql\_error(conn));

return 1;

}

printf("Doctor inserted successfully!\n");

break;

}

break;

case 4:

// Exit the program

printf("Exiting program...\n");

mysql\_close(conn);

return 0;

}

}

return 0;

}

**Examples of Code execution:**

