**CONSOLE CODE - EMBEDDED SQL :**

#include <stdio.h>

#include<pgtypes\_date.h>

#include<string.h>

#include<stdlib.h>

#include<ctime>

int main() {

EXEC SQL BEGIN DECLARE SECTION;

char \*test= "201501187@10.100.71.21:5432";

char \*usr = "201501187";

char \*pwd = "201501187";

int option;

EXEC SQL END DECLARE SECTION;

EXEC SQL CONNECT TO :test USER :usr USING :pwd;

EXEC SQL set search\_path to etourism;

MAIN: printf(“\nSelect :”);

ABV1: printf("\n1. SignUp as new User.\n");

printf("2. Login.\n");

scanf("%d",&option);

if(option == 1){

addNewUser();

}

else if(option == 2){

flag = registeredUser();

if(flag == 1)

goto MAIN;

}

else{

printf(“Select valid option : ”);

goto ABV1;

}

return 0;

}

Void addNewUser(){

EXEC SQL BEGIN DECLARE SECTION;

char loginId[8], username[25], passwrd[30], vpasswrd[30], emailId[25];

string contactNo;

const char \*stmt = "INSERT INTO “User” VALUES(?,?,?,?,?)";

int c = 1;

EXEC SQL END DECLARE SECTION;

printf("\nEnter your Name : ");

scanf("%s", username);

printf("\nSelect some LoginId : ");

ABV2: scanf("%s", loginId);

EXEC SQL SELECT count(loginid) INTO :c FROM “User” where loginid = :loginId;

if(c == 1) //If similar login id already exists.

{

printf(“\nSelected loginId is unavailable.\nSelect some other LoginId :”)

goto ABV2;

}

printf("\nMake Password : ");

ABV3: scanf("%s", passwrd);

printf("\nVerify Password : ");

scanf("%s", vpasswrd);

if(strcmp(passwrd, vpasswrd)) == 0)

{

printf(“\nPassword not verified. Enter again : ”)

goto ABV3;

}

printf("\nEnter contact Number : ");

ABV4: scanf("%s", contactNo);

if(contactNo.length() != 10){

printf(“\nEnter valid contact number : ”);

goto ABV4;

}

printf("\nEnter valid email Id : ");

scanf("%s", emailId);

EXEC SQL PREPARE mystmt FROM :stmt;

EXEC SQL EXECUTE mystmt USING :loginId, :username, :emailId, :contactNo, :passwrd;

EXEC SQL COMMIT;

EXEC SQL DISCONNECT CURRENT;

showOptions(loginId);

}

int registeredUser(){

EXEC SQL BEGIN DECLARE SECTION;

char loginId[8], passwrd[30], pwd[30];

int opt, c = 0;

EXEC SQL END DECLARE SECTION;

printf(“\nLogin Id : ”);

scanf(“%d”, loginId);

EXEC SQL SELECT count(loginid) INTO :c FROM “User” where loginid = :loginId;

if(c == 0)

{

printf(“\nInvalid Login Id.”);

return(1);

}

printf(“\nPassword : ”);

scanf(“%d”, passwrd);

EXEC SQL SELECT “Password” INTO :pwd FROM “User” where loginid = :loginId;

if(passwrd != pwd)

{

printf(“\nIncorrect password.”);

return(1);

}

showOptions(loginId);

return 0;

}

void showOptions(char loginId[]){

EXEC SQL END DECLARE SECTION;

int opt, countViews;

char cityName[15], sState[20], sRegion[10], sCategory[10];

EXEC SQL SELECT count(loginid) INTO :c FROM “User” where loginid = :loginId;

SHOW: printf(“\nSelect : ”);

printf(“\n1. Search by city.”);

printf(“\n2. Search by region.”);

printf(“\n3. Search by category.”);

printf(“\n4. Search by state.”);

printf(“\n5. Logout”);

scanf(“%d”, opt);

switch(opt)

{

case 1 : printf(“\nEnter city name : ”);

scanf(“%s”, cityName);

EXEC SQL SELECT count INTO countViews from VIEWHISTORY WHERE loginid = :loginId AND cityid = :cityId;

countViews++;

EXEC SQL INSERT into VIEWHISTORY (loginid, cityid, count) VALUES (:loginId, :cityId, :countViews);

showInCity(loginId, cityName);

break;

case 2 : printf(“\nEnter region : ”);

scanf(“%s”, sRegion);

EXEC SQL SELECT cityname from CITY as c where c.region = :sRegion;

goto SHOW;

break;

case 3 : printf(“\nEnter state : ”);

scanf(“%s”, sState);

EXEC SQL SELECT cityname from CITY as c where c.state = :sState;

goto SHOW;

break;

case 4 : printf(“\nEnter category : ”);

scanf(“%s”, sCategory);

EXEC SQL SELECT cityname from CITY as c NATURAL JOIN PLACES as p where p.category = :sCategory;

goto SHOW;

break;

case 5 : return;

default : printf(“\nInvalid input.”);

goto SHOW;

}

return;

}

void showInCity(char loginId[], char cName[]){

int opt;

JUMP: printf(“\nSelect : ”);

printf(“\n1. View places in %s.”, cName);

printf(“\n2. View hotels in %s.”, cName);

printf(“\n3. View restaurants in %s.”, cName);

printf(“\n4. Transportation facilities.”);

printf(“\n5. Back”);

scanf(“%d”, opt);

switch(opt) {

case 1: printf(“\nPlaces to visit in %s :\n”, cName);

EXEC SQL SELECT pname, category, address, rating, besttimetovisit from PLACES NATURAL JOIN CITY where cityname = :CName;

goto JUMP;

break;

case 2: showHotelDetails(loginId, cName);

goto JUMP;

break;

case 3: printf(“\nRestaurants in %s :\n”, cName);

EXEC SQL SELECT \* from RESTAURANT NATURAL JOIN CITY where cityname = :CName;

goto JUMP;

break;

case 4: showTransportationDetails(cName);

goto JUMP;

break;

case 5: showOptions();

return;

default : printf(“\nInvalid input.”);

goto JUMP;

}  
 return;

}

void showHotelDetails(char loginId[], char cName[]){

int option, opt;

EXEC SQL BEGIN DECLARE SECTION;

char hId[8];

EXEC SQL END DECLARE SECTION;

POOL: printf(“\nSelect :”);

printf(“\n1. View hotels in %s. ”, cName);

printf(“\n2. Book rooms”);

printf(“\n3. Back”);

scanf(“%d”, option);

switch(option) {

case 1: printf(“\nHotel List : \n”);

EXEC SQL SELECT \* from HOTELS NATURAL JOIN CITY where cityname = :cName;

LABEL: printf(“\nSelect : ”);

printf(“\n1. Further information about any hotel.”);

printf(“\n2. Back.”);

scanf(“%d”, opt);

if(opt == 1){

printf(“\nEnter hotel Id : ”);

scanf(“%s”, hID);

EXEC SQL SELECT \* from HOTELS NATURAL JOIN TYPESOFROOMS where hotelid = :hID;

goto LABEL;

}

else{

goto POOL;

}

break;

case 2: booksRoom(loginId);

goto POOL;

break;

default : printf(“\nInvalid input.”);

goto POOL;

}

return;

}

void booksRoom(char loginId[]){

EXEC SQL BEGIN DECLARE SECTION;

char typeName[10], promoId[8], hId[8];

integer numOfRooms, f = 0, option, transactionAmt, transactionNum;

string bookinDate, checkInDate, checkOutDate, bookingTime;

const char \*stmt = "INSERT INTO BOOKING VALUES(?,?,?,?,?,?,?,?,?)";

EXEC SQL END DECLARE SECTION;

printf(“\nEnter Hotel Id : ”);

JMP1: scanf(“%s”, hId);

EXEC SQL SELECT count(hotelid) INTO :f from HOTELS where hotelid = :hId;

if(f == 0){

printf(“\nEnter valid Hotel Id : ”);

goto JMP1;

}

printf(“\nEnter checkInDate as d/m/y : ”);

scanf(“%s”, checkInDate);

printf(“\nEnter checkOutDate as d/m/y : ”);

scanf(“%s”, checkOutDate);

printf(“\nEnter room type : ”);

JMP2: scanf(“%s”, typeName);

EXEC SQL SELECT count(hotelid) INTO :f from HOTELS NATURAL JOIN TYPEOFROOM where hotelid = :hId AND tname = :typeName;

if(f == 0){

printf(“\nEnter valid room type of Hotel Id %d : ”, hId);

goto JMP2;

}

printf(“\nEnter number of rooms : ”);

JMP3: scanf(“%d”, numOfRooms);

// Here this data is send to the hotel’s website, where it cross checks all avaibility and provides bool for booking of the room for that period.

If that bool is false - Hotel booking cannot be done -> return;

Else continue ahead process;

time\_t tt;

struct tm \* ti; // Declaring variable to store return value of localtime() and date.

time (&tt);

ti = localtime(&tt);

string date = “”, time = “”, mday, month, year, hr, min, sec;

itoa(ti->tm\_mday, mday, 10);

itoa(ti->tm\_mon, month, 10);

itoa(ti->tm\_year, year, 10);

itoa(ti->tm\_hour, hr, 10)

itoa(ti->tm\_min, min, 10);

itoa(ti->tm\_sec, sec, 10);

date += mday+”/”+month+”/”+year;

time += hour+”:”+min+”/”+sec;

bookingDate = date;

bookingTime = time;

JMP4: printf(“\nSelect : \n1. Check promocode available to you.\n2. Apply promocode and continue payment.”);

scanf(“%d”, option);

if(option == 1){

printf(“\nApplicable promocode with you :\n”);

EXEC SQL SELECT \* from PROMOCODEAVAILABLETOUSER NATURAL JOIN PROMOCODE where loginid = :loginId;

goto JMP4;

}

else if(option == 2)

{

printf(“\nEnter promocode : ”);

scanf(“%s”, promoId);

EXEC SQL SELECT count(promoid) INTO :f from PROMOCODEAVAILABLETOUSER where loginid = :loginId;

if(f == 0){

printf(“\nThis promocode is not applicable. ”);

Promoid = NULL;

goto JMP4;

}

EXEC SQL PREPARE mystmt FROM :stmt;

countViews++;

transactionNum++;

EXEC SQL EXECUTE mystmt USING :transactionNum, :loginId, :hId, :bookingDate, :bookingTime, :promoId, :typeName, :checkInDate, :checkOutDate, :numOfRooms;

EXEC SQL COMMIT;

EXEC SQL SELECT (r.CostPerDay \* b.NoOfRoomsBooked )- ((r.CostPerDay \* b.NoOfRoomsBooked )\*p.Discount)/100) AS :transactionAmt FROM promocode AS p NATURAL JOIN booking AS b NATURAL JOIN roomtype AS r;

printf(“\nTotal amount to be paid : %d”, transactionAmt);

}

return;

}

Void showTransportationDetails(char cityName[]){

int options;

EXEC SQL BEGIN DECLARE SECTION;

char cityToVisit[15];

EXEC SQL END DECLARE SECTION;

SUN: printf(“\nSelect : ”);

printf(“\n1. Intracity transportation facitilies.”);

printf(“\n2. Interacity transportation facitilies.”);

printf(“\n3. Back”);

scanf(“%d”, options);

switch(options):

{

case 1: printf(“\nIntracity transport facility in %s :\n”, cityName);

EXEC SQL SELECT \* from INTRACITYTRANSPORT NATURAL JOIN CITY where cityname = :cityName;

goto SUN;

break;

case 2: printf(“Enter city to visit from %s ”, cityName);

scanf(“%s”, cityToVisit);

EXEC SQL SELECT \* from INTERCITYTRANSPORTION NATURAL JOIN CITY where city1 = cityName AND city2 = cityToVisit;

goto SUN;

break;

case 3: return;

default : printf(“\nInvalid input.”);

Goto SUN;

}

}

**Console code in C for execution and updation through queries :**

#include <stdio.h>

#include <stdlib.h>

#include <postgresql/libpq-fe.h>

#include <string.h>

PGconn \*conn;

PGresult \*res;

int main() {

conn = PQconnectdb("hostaddr=10.100.71.21 port=5432

user=201501187

dbname=201501187 password=201501187");

if (PQstatus(conn) == CONNECTION\_BAD) {

puts("We were unable to connect to the database");

exit(0);

}

char \*string=(char \*)malloc(1000\*sizeof(char));

while(1){

res=PQexec(conn,"set search\_path to \"etourism\";");

printf("Select:\n1. To run query.\n2. To update.\n3. Exit\n");

int type;

scanf("%d\n",&type);

fflush(stdout);

if(type==1)

{ scanf("%[^\n]s",string);

executeQuery(string);

}

else if(type==2)

{

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

updateQuery(string);

}else{

PQclear(res);

PQfinish(conn);

return 0;

}

}

return 0;

}

void executeQuery(char \*string){

int rec\_count;

int row;

int col;

res = PQexec(conn,string);

if (PQresultStatus(res) != PGRES\_TUPLES\_OK) {

printf("%s",PQresultErrorMessage(res));

}

rec\_count = PQntuples(res);

printf("We received %d records.\n", rec\_count);

int col\_count=PQnfields(res);

for (col=0; col<col\_count; col++) {

printf("%s\t", PQfname(res,col));

}

puts("");

for (row=0; row<rec\_count; row++) {

for (col=0; col<col\_count; col++) {

printf("%s\t", PQgetvalue(res, row, col));

}

puts("");

}

return;

}

void updateQuery(char \*string){

res = PQexec(conn,string);

if (PQresultStatus(res) != PGRES\_COMMAND\_OK) {

printf("%s",PQresultErrorMessage(res));

}

else

{

printf("Update query executed succesfully\n");

}

return;

}