using namespace System::Collections::Generic;

bool Healthifyme::healthaccountdetailsDB(int devtyp, Dictionary<int, String^>^ healthapp\_info, String^ wAppPath, String^ FileDetails)

{

bool transaction = false;

sqlite3\* DB;

sqlite3\* DB2;

sqlite3\* DB1;

sqlite3\* dataBase;

sqlite3\_stmt\* sk;

sqlite3\_stmt\* sr;

sqlite3\_stmt\* media\_stmt;

sqlite3\_stmt\* media\_count\_stmt;

sqlite3\_stmt\* media\_rec\_stmt;

sqlite3\_stmt\* download\_insert;

sqlite3\_stmt\* download\_ins;

char\* strQry;

char\* strQr;

int exit = 1;

int exitt = 1;

int entry = 1;

List<pickdata^>^ DataToBe = gcnew List<pickdata^>();

int error;

logger->Log("Started opening healthifyme db", Category::Info, Priority::Low);

std::string dbtmp;

sqlite3\_stmt\* st;

dbtmp = msclr::interop::marshal\_as<std::string>(wAppPath);

const char\* dbname = dbtmp.c\_str();

exit = sqlite3\_open(dbname, &DB);

int sd = sqlite3\_prepare(DB, "SELECT expert\_username from booking;", -1, &st, 0);

int ret = sqlite3\_step(st);

String^ userName = gcnew String(reinterpret\_cast<const char\*>(sqlite3\_column\_text(st, 0)));

ret = sqlite3\_finalize(st);

exit = sqlite3\_close(DB);

healthapp\_info->Add(85, userName);

int rec\_count = 0, media\_Id = 0, media\_rec = 0;

String^ dbPath = DBCreator::newMC\_DB\_writePath;

dbtmp = msclr::interop::marshal\_as<std::string>(dbPath);

dbname = dbtmp.c\_str();

error = sqlite3\_open(dbname, &DB1);

if (error != 0)

logger->Log("Mobilecheck Db cannot be opened in healthifymeDB", Category::Info, Priority::Low);

int rs\_id = sqlite3\_prepare\_v2(DB1, "select max(RecNo) from AccountDetails;", -1, &media\_count\_stmt, 0);

if (rs\_id == 0)

{

sqlite3\_step(media\_count\_stmt);

media\_Id = sqlite3\_column\_int(media\_count\_stmt, 0);

media\_Id++;

/\*if (media\_Id)

media\_Id = media\_Id + 1;

else

media\_Id = 1;\*/

sqlite3\_finalize(media\_count\_stmt);

}

else

{

return false;

}

sqlite3\_exec(DB1, "BEGIN TRANSACTION;", NULL, NULL, NULL);

transaction = true;

try

{

//Dictionary<int, String^>^ IndvidualDat = gcnew Dictionary<int, String^>();

//for each (pickdata ^ items in DataToBe) {

// IndvidualDat->Clear();

// IndvidualDat->Add(1, items->userName);

// IndvidualDat->Add(67, items->handleName);//healthifymename

// IndvidualDat->Add(63, items->dateOfBirth);

// IndvidualDat->Add(10, items->Gender);

// //IndvidualDat->Add(, items->igHandle);//instagramname

// IndvidualDat->Add(4, items->phone);

//}

//int mac = 16;

for each (KeyValuePair<int, String^> ^ dict\_data in healthapp\_info)

{

if (sqlite3\_prepare(DB1, "insert into AccountDetails(RecNo,Data,DataInfo\_Id,Dev\_Id,Ftype) values (?1,?2,?3,?4,?5);", -1, &download\_ins, 0) == SQLITE\_OK)

{

if (!String::IsNullOrEmpty(dict\_data->Value))

{

sqlite3\_bind\_int(download\_ins, 1, media\_Id);

String^ string = dict\_data->Value;

std::string cString = msclr::interop::marshal\_as<std::string>(string);

const char\* data = cString.c\_str();

sqlite3\_bind\_text(download\_ins, 2, data, strlen(data), 0);

sqlite3\_bind\_int(download\_ins, 3, dict\_data->Key);

sqlite3\_bind\_int(download\_ins, 4, DBCreator::deviceID);

sqlite3\_bind\_int(download\_ins, 5, 114);

int re = sqlite3\_step(download\_ins);

//sqlite3\_clear\_bindings(download\_ins);

// sqlite3\_reset(download\_ins);

//mac++;

}

}

int err = sqlite3\_finalize(download\_ins);

if (int errCode = sqlite3\_errcode(DB1) > 0)

{

String^ logMessage = "Downloads::PopulateDownloadsDbFile : " + sqlite3\_errcode(DB1) + " " + msclr::interop::marshal\_as<System::String^>(sqlite3\_errmsg(DB1));

errCode = 0;

}

rec\_count++;

}

}

catch (Exception^ e)

{

logger->Log("Exception occured in healthifyme Account db::populate healthifyme AccountDetails" + e->Message, Category::Exception, Priority::High);

}

finally

{

sqlite3\_exec(DB1, "END TRANSACTION;", NULL, NULL, NULL);

error = sqlite3\_close(DB1);

logger->Log("Completed and stoped healthifyme db", Category::Info, Priority::Low);

}

return 1;

}

bool Healthifyme::healthuserdetailsDB(int devtyp, Dictionary<int, String^>^ healthifymeappPath, String^ FileDetails)

{

bool transaction = false;

sqlite3\* DB;

sqlite3\* DB5;

sqlite3\* dataBase;

sqlite3\_stmt\* st;

sqlite3\_stmt\* sk;

sqlite3\_stmt\* sr;

sqlite3\_stmt\* media\_stmt;

sqlite3\_stmt\* media\_count\_stmt;

sqlite3\_stmt\* media\_rec\_stmt;

sqlite3\_stmt\* download\_insert;

sqlite3\_stmt\* download\_ins;

char\* strQry;

char\* strQr;

int exit = 1;

int exitt = 1;

int entry = 1;

std::string dbtmp;

std::string healthifymedbpath;

const char\* dbname = dbtmp.c\_str();

Dictionary<int, String^>^ IndvidualDat = gcnew Dictionary<int, String^>();

//List<pickdat^>^ DataToBe = gcnew List<pickdat^>();

int error;

for each (KeyValuePair<int, String^> ^ dict\_data in healthifymeappPath)

{

if (dict\_data->Key == 5 || dict\_data->Key == 71 || dict\_data->Key == 9)

{

if (dict\_data->Key == 71) {

IndvidualDat->Add(2, dict\_data->Value);

}

else

{

IndvidualDat->Add(dict\_data->Key, dict\_data->Value);

}

}

}

int rec\_count = 0, media\_Id = 0, media\_rec = 0;

String^ dbPath = DBCreator::newMC\_DB\_writePath;

dbtmp = msclr::interop::marshal\_as<std::string>(dbPath);

dbname = dbtmp.c\_str();

error = sqlite3\_open(dbname, &DB5);

if (error != 0)

logger->Log("Mobilecheck Db cannot be opened in healthifymeDB", Category::Info, Priority::Low);

int rs\_id = sqlite3\_prepare(DB5, "select max(RecNo) from UserAccounts;", -1, &media\_count\_stmt, 0);

if (rs\_id == 0)

{

sqlite3\_step(media\_count\_stmt);

media\_Id = sqlite3\_column\_int(media\_count\_stmt, 0);

media\_Id++;

sqlite3\_finalize(media\_count\_stmt);

}

else

{

return false;

}

sqlite3\_exec(DB5, "BEGIN TRANSACTION;", NULL, NULL, NULL);

transaction = true;

int i = 1;

try

{

//Dictionary<int, String^>^ IndvidualDat = gcnew Dictionary<int, String^>();

IndvidualDat->Add(13, FileDetails);

IndvidualDat->Add(4, "114");

for each (KeyValuePair<int, String^> ^ dict\_data in IndvidualDat)

{

if (sqlite3\_prepare\_v2(DB5, "insert into UserAccounts(RecNo,Data,DataInfo\_Id,Dev\_Id,UserAccounts\_Id) values (?1,?2,?3,?4,?5);", -1, &download\_ins, 0) == SQLITE\_OK) {

if (!String::IsNullOrEmpty(dict\_data->Value))

{

sqlite3\_bind\_int(download\_ins, 1, media\_Id);

//sqlite3\_bind\_int(download\_ins, 1, 999);

String^ string = dict\_data->Value;

//String^ string = "Moj";

std::string cString = msclr::interop::marshal\_as<std::string>(string);

const char\* data = cString.c\_str();

sqlite3\_bind\_text(download\_ins, 2, data, strlen(data), 0);

sqlite3\_bind\_int(download\_ins, 3, dict\_data->Key);

sqlite3\_bind\_int(download\_ins, 4, DBCreator::deviceID);

sqlite3\_bind\_int(download\_ins, 5, 40);

/\*sqlite3\_bind\_int(download\_ins, 4, 444);

sqlite3\_bind\_int(download\_ins, 5, 555);\*/

//sqlite3\_bind\_int(download\_insert, 6, 41);

int re = sqlite3\_step(download\_ins);

// sqlite3\_clear\_bindings(download\_ins);

//sqlite3\_reset(download\_ins);

}

}

int err = sqlite3\_finalize(download\_ins);

if (int errCode = sqlite3\_errcode(DB5) > 0)

{

String^ logMessage = "Downloads::PopulateDownloadsDbFile : " + sqlite3\_errcode(DB5) + " " + msclr::interop::marshal\_as<System::String^>(sqlite3\_errmsg(DB5));

errCode = 0;

}

}

}

catch (Exception^ e)

{

logger->Log("Exception occured in healthifyme Account db::populate healthifyme AccountDetails" + e->Message, Category::Exception, Priority::High);

}

finally

{

sqlite3\_exec(DB5, "END TRANSACTION;", NULL, NULL, NULL);

error = sqlite3\_close(DB5);

logger->Log("Completed and stoped healthifyme db", Category::Info, Priority::Low);

}

return 1;

}