

**Philadelphia University**

**Faculty of Administrative & Financial Sciences**

**Business Networking and Systems Management Department**

**Graduation project**

Student Tracking System

Project supervisor by: Dr. Ali Hadi al-shemery

Student name: Khalid Issam Ahmad hussien

201110771

***بسم الله الرحمن الرحيم***

***قال تعالى:***

***" وعلمك ما لم تكن تعلم وكان فضل الله عليك كبيرا ً "***

***صدق الله العظيم***

الأهداء

**إلى العطاء .. إلى نجاحي .. إلى من علموني تحدي الصعوبات .. إلى من وضّحوا لي خطوات الصبر مع النجاح .. إلى من يعجز لساني عن وصفهم**

**أبي الغالي أمي الغالية دعوني أقول لكم بصوت النجاح شكراً وهي كلمة لاتعبر عن جهودكم معي ومن اجلي .**

**هديتي لكم هذا النجاح المتواضع الذي يعد بمثابة افتخار لي في حياتي العلمية والعملية لا أجد سوى هذا النجاح فهل تقبلوه مني .**

شكر وتقدير

**نحمد الله سبحانه وتعالى ونشكره على أن وفقنا لإتمام هذا العمل المتواضع ونصلي ونسلم على النبي الأمين الذي أرسل رحمةً للعالمين**

**واتقدم بالشكر والتقدير إلى رواد التعليم الأوائل وإلى كل من شجع قلماً أو منح كتاباً يبتغي به وجه الله وقصد من ورائه إنارة الفكر والى كل من علمني حرفاً وهذب فينا خُلقا وأنار لي درب العلم والى كل من سعى إلى تربية الأجيال المسلمة لتكون خير امة أخرجت للناس وإلى كل عضو هيئة تدريس قدم لي مساعدة وساندني واتقدم بالشكر والعرفان لمشرفي الكريم على انجاز هذا المشروع الدكتور علي حسين الشمري**

**والى جميع من ساندني من اخوتي واخواتي حفظهم الله**

**وأعز الناس الي:**

**روزان محمد أبوشهيل**

**محمود شكري**

**علاء عيسى**

**اياد مقابله**

**Introduction**

* 1. Introduction ……………………………………………………………………………… 7
  2. Background ……………………………………………………………………..……….. 8
  3. Objectives ……………………………………………………………………..…………. 9
  4. The problem……………………………………………………………………..………. 9
  5. The proposed solution……………………………………………………………… 10

**Planning and Analysis**

2-1 Project constraints…………………………………………………………………… 12

2-2 Project management……………………………………………………………….. 12

2-3 Fact finding ……………………………………………………………………………… 13

2-4 Functional and non functional requirement …………………………….. 13

2-5 System use case ………………………………………………………………………. 16

2-6 System DFD ………………………………………………………………………………17

**Project design**

3-1 Data item dictionary………………………………………………………………….21

3-2 Table design ……………………………………………………………………………. 23

3-3 Data structure dictionary ………………………………………………………… 24

3-4 ERD ………………………………………………………………………………………….25

**Project implementation**

4-1 Project implementation………………………………….………………………..29

**Conclusion and Future Work**

5-1 Conclusion and Future Work…………………………………………………….72

Chapter (1)

Introduction

**1-1-Introduction**

**Why instructors need "Student Tracking system"?**

Based on research, most of instructors said that when they want to use the university's system they must be on their computers in the university and there is no way to use the system somewhere else unlike the "Student tracking system".

Besides that, the university's system does not have the features that the "Student Tracking System" has, such as:

- It could act as a pocket student's mark calculator for instructors which could be carrying with them anywhere and anytime.

* Has the capability of calculating your Quizzes and Assignments automatically. All the instructor needs to do is insert the student marks gained (quizzes and assignments) and it will automatically add and take the average.
* A built-in fully automated attendance system that can be used by the instructors to calculate how much marks to deduct from the final participation mark. For sure this depends on the student's number of days he/she was absent. The attendance system not only calculates marks to be deducted from the participation mark, but even alerts the instructor on the exact time/date that a student must be warned by a notice (first, second, final) then finally he/she might get dismissed from the class.
* Finally, if an instructor feels he/she needs other features to be added to the system; then he/she could easily adjust the system to fit those requirements. At the end of the day we are using the Open Source methodology and the source code will be available on the system's official website.

**1-2-Background**

WHAT DOES "STUDENT TRACKING SYSTEM" DO?

1-It is a pocket student's mark calculator for instructors which could be carrying with them anywhere and anytime, so he\she doesn’t have to be on his\her computer in the university.

2- It can calculate the student participation based on his\her quizzes marks, assignments and his\her absent (which would be deducted from his\her final participation).

3-We are using open source method so it can effort changing needs for every single instructor (the source code will be uploaded on the system's official website).

**1-3-Objective**

The objective of this project is to develop Student Grading System, application implemented using Python 2.7 and MySQLdb.

This system will provides Command Line inter face for instructors to set up coursework including assignments, projects,

and exams, and grade the coursework and give

**1-4-The Problems**

1. Most of instructors said that when they want to use the university's system they must be on their computers in the university and there is no way to use the system somewhere else
2. Quizzes , assignments and participation mark calculation is one of the the university's system problem, the instructor should manually calculate the quizzes , assignments and participation marks
3. The university's system don’t offer the changing needs of the instructors

**1-5-The Propose solutions**

1. The Instructors can carry the "Student Tracking System" with them anywhere and anytime.

2. The "Student Tracking System" Has the capability of calculating your Quizzes and Assignments automatically. All the instructor needs to do is insert the student marks gained (quizzes and assignments) and it will automatically add and take the average.

3. if an instructor feels he/she needs other features to be added to the system,then he/she could easily adjust the system to fit those requirements. At the end of the day we are using the Open Source methodology and the source code will be available on the system's official website.

Chapter (2)

Planning and analysis

**2-1 Project constrains**

The limitation of such a program High security is needed to be protected especially in the case of universities the contact details of students. Personal information protection is a serious responsibility and heavily punishable by the law.

The other limitation is the storage capacity needed for the amount of students. The more students, the more data, and the more data mean the more server processing power and memory that is required.

cannot take place of human because the system is always dependent of human.

**2-2 Project Management**

**2-2-1- Software Tools**

1. Python 2.7

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages. The language provides constructs intended to enable clear programs on both a small and large scale.

2. MySQL Database

One of data-base providers we are going to use it to build the web-site's data-base it is easy to store data in a good architecture as good as we need.

3.Operating System: Windows 7, Mac, Unix\Linux.

**2-3 – Fact Finding**

I have collected information through websites and through searching the internet, also through previous projects that dealt with this project, besides I did some interviews with doctors (Dr.Ali hadi and Dr.Ahmad al-ghoul).

And I have collected from them how the functionality of grades and attendance system work

(20 marks for the first exam\20 for second exam\20 for the participation mark divides to attendance, assignments and quizzes\40 marks for the final exam).

**2-4 – Functional and Non Functional Requirement**

**2-4-1 – Functional Requirement**

Those are the functional requirements for the system described the services .

1-Instructors operations:

* Adding instructors
* Update instructors
* Display all instructors

2-Students operations:

* Adding Students
* Update students
* Display all students

3- Subjects operations:

* Adding Subjects
* Update subjects
* Delete subjects
* Display all subjects

4-Sections operations:

* Adding sections
* Delete sections
* Display all sections

5-Attendance operations:

* Adding attendance files
* Update attendance files
* Delete attendance files
* Display all attendance files

6-Marks operations:

* Adding Exams marks(First, Second, Final)
* Adding participation mark(depending on absents\exclude absents)
* Update marks
* Display marks(one student\all students)

7-Make your own MySQL operation:

* This function can make you write your own MySQL statement

8-Backup your database

9-Import your backup

**2-4-2 – Nonfunctional Requirements**

1. Ease of Use:

The project should be easy and it should have GUI

1. Understanding:

The project should be easy to understand and the user does not need to have hard training courses or be an expert to use the system

1. Reliability

For the reliability to the project you can take a look from these:

Availability: the project should be available at any time. The project should not be repeated.

1. Performance

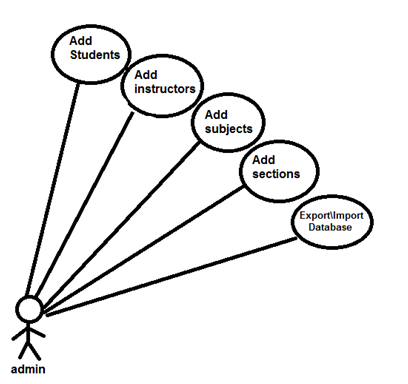
The project should be in a high speed in executing any kind of required procedures from the user

**2-5 – System Use Cases:**

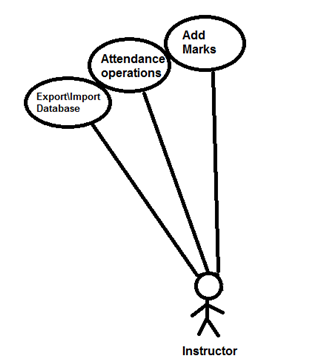
**2-5-1 – System Actors**

System Manger:

This side of the actor with the system as a system administrator, his goal is to manage the system and take responsibility for its operations.

**2-5-2-Use case mode**

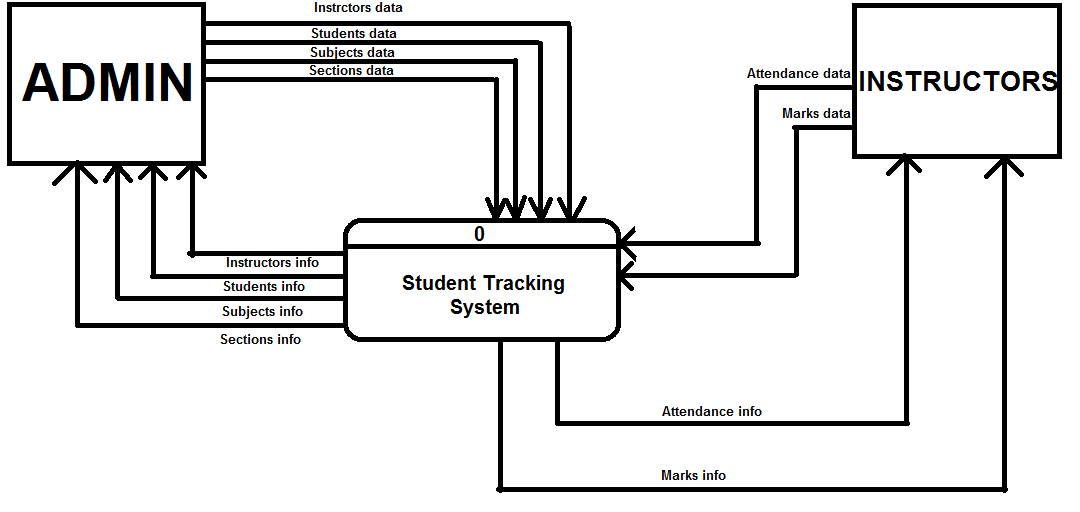
**ADMIN (2.1)**

****

**INSTRUCTORS (2.2)**

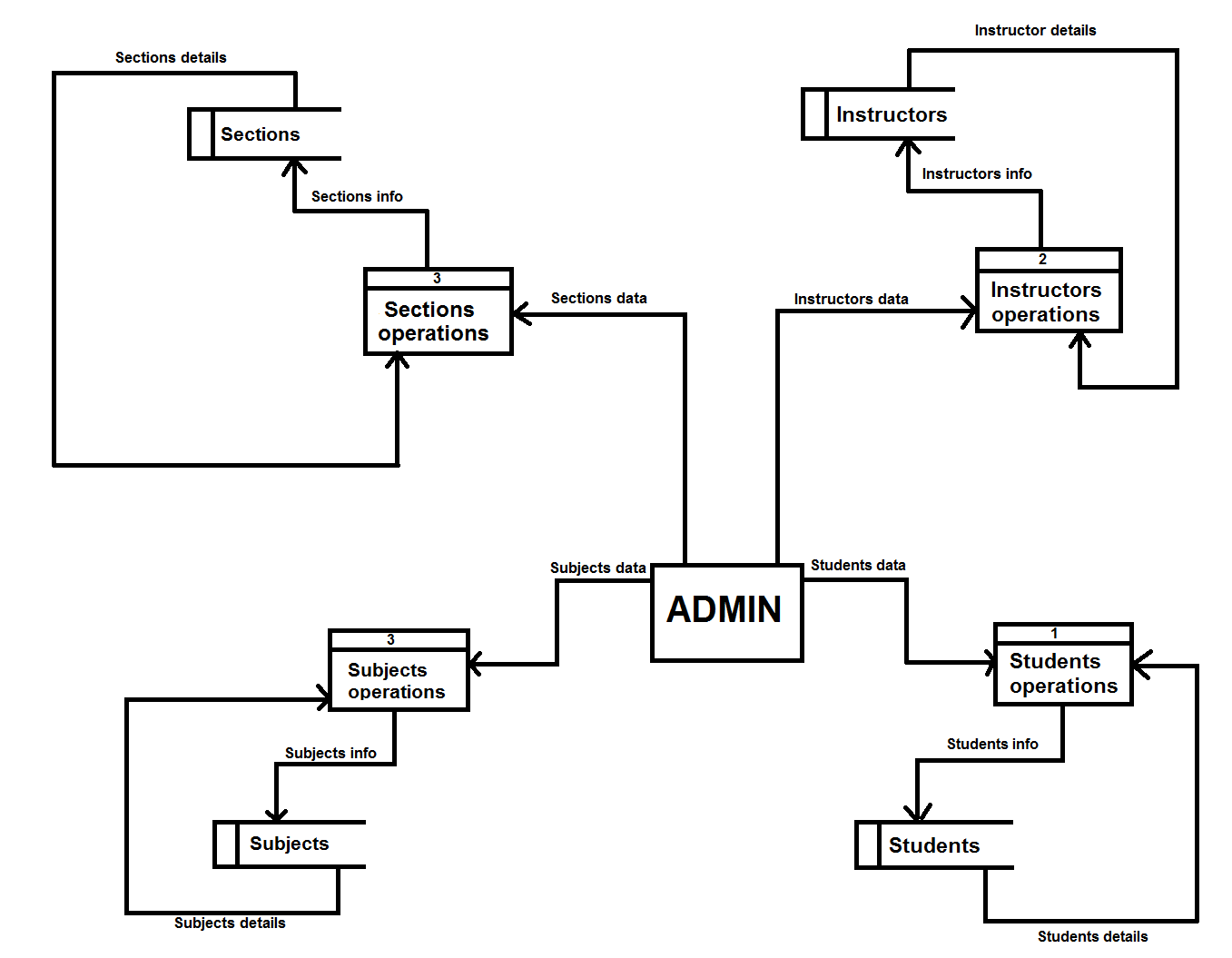
**2-6-System DFD:**

**2-6-1- Context diagram**

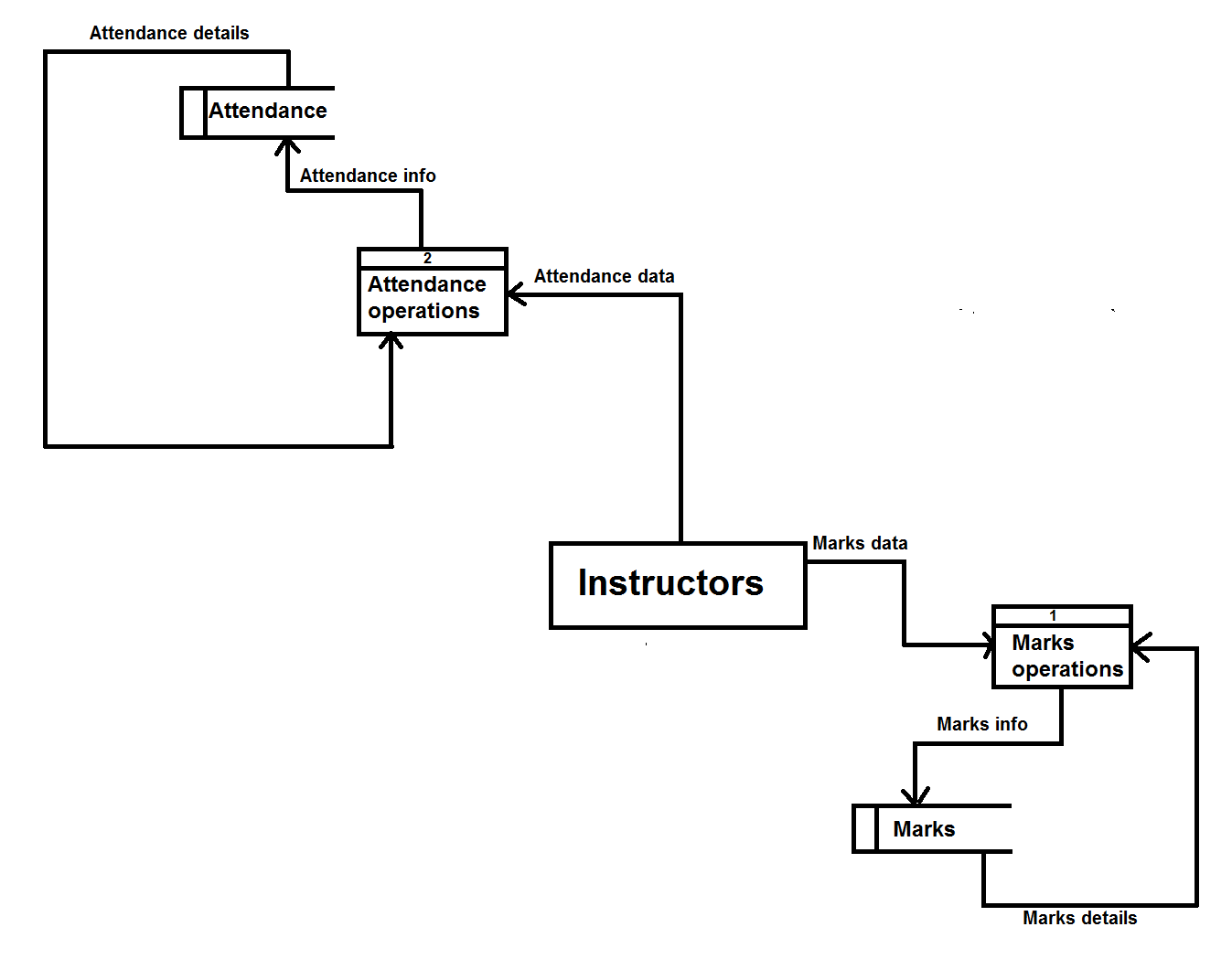


**Context diagram (2.3)**

**2-6-2- Level zero**



**Admin level zero(2.4)**

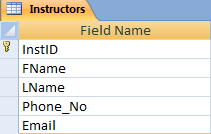


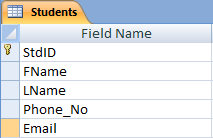
**Instructors level zero(2.5)**

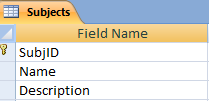
Chapter Three

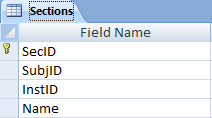
Project design

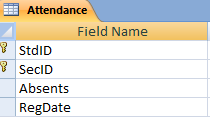
**3-1-Data item dictionary**

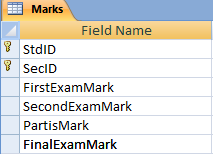








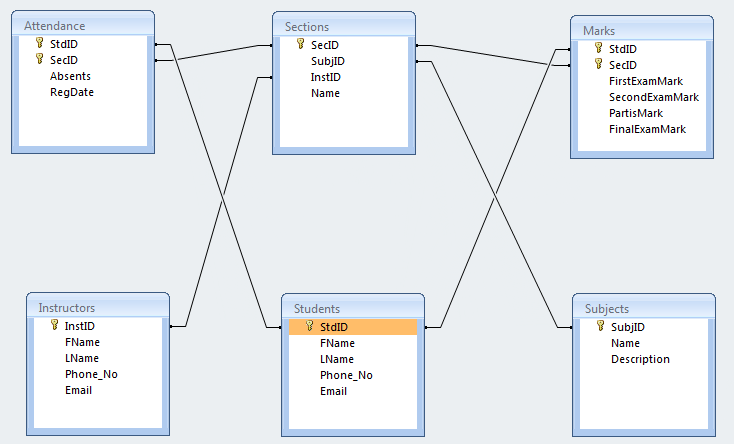




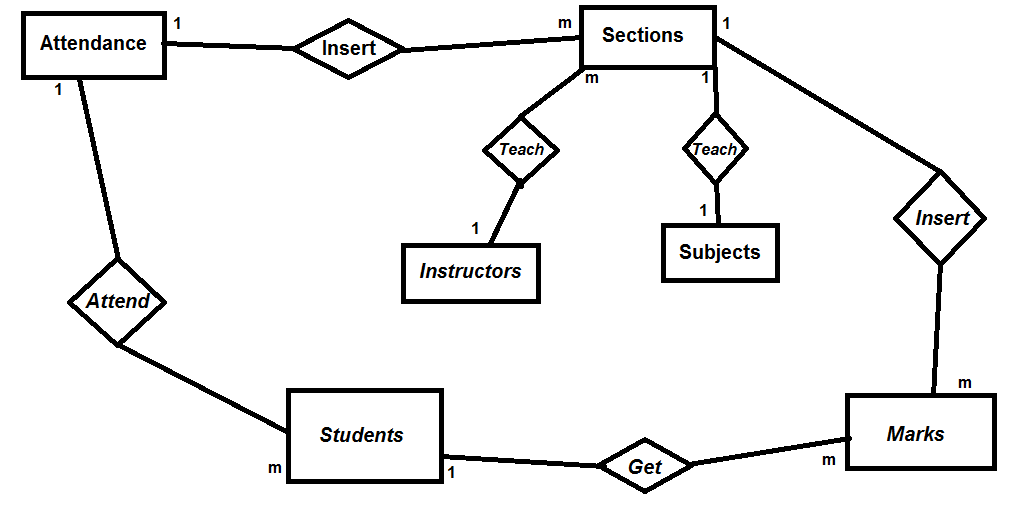
**3-2-Table design**

1. Instructors(**InstdID,**FName,LName,phone\_no,Email)
2. Students(**StdID**,FName,LName,phone\_no,Email)
3. Subjects(**SubjID** ,Name ,Description )
4. Sections(**SecID**,SubjID,InstID,Name)
5. Attendance(**StdID**,**SecID**,absents,RegDate)
6. Marks(**StdID**,**SecID**,FirstExamMark,SecondExamMark, Participation Mark,FinalExamMark)

**3-3-Data structure dictionary**

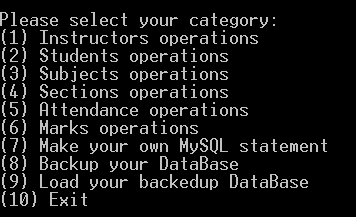
****

**3-4-ERD**

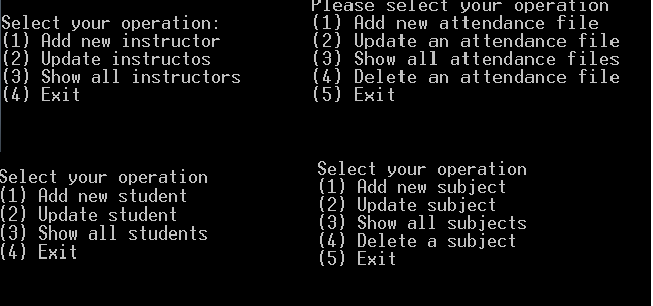
****

**ERD (3.1)**

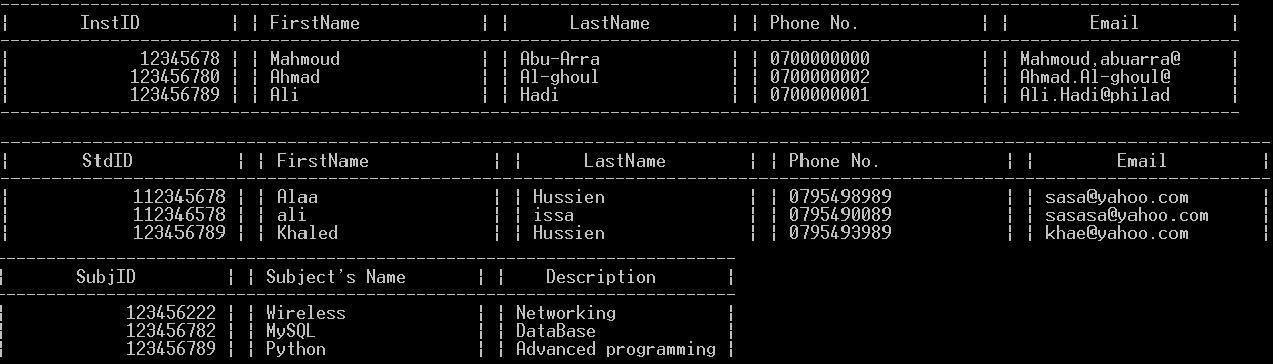
**3-5-Interface & Screens Design**

****

**Home Screen (3.2)**

****

**Inserting Screens (3.3)**

****

**Some Output Screens (3.4)**

Chapter Four

Project

Implementation

**Database operations:**

#Creating tables

#DataBase Operations

###########################

import MySQLdb

import os

import shutil, errno

###########################

def connect():

x=MySQLdb.connect('127.0.0.1', 'root', 'khaled', 'sts');

return x

def disconnect():

y=connect().close()

return y

def initializing():

try:

connect()

cur = connect().cursor()

CTAT="""CREATE TABLE attendance(StdID INT(9),SecID INT(9),absents INT(2),RegDate DATE)"""

cur.execute(CTAT)

#CTAT is a variable includes creating table for student's attendance.

CTSM="""CREATE TABLE marks (StdID INT(9),SecID INT(9),FirstExamMark INT(2),SecondExamMark INT(2),PartisMark INT(2),FinalExamMark INT(2))"""

cur.execute(CTSM)

#CTSD is a variable includes creating table for student's marks.

CTIN="""CREATE TABLE instructors (InstID INT(9) NOT NULL PRIMARY KEY,FName char(10),LName char(10),phone\_no char(10),Email char(30))"""

cur.execute(CTIN)

#CTIN is a variable includes creating table for instructors.

CTST="""CREATE TABLE students (StdID INT(9) NOT NULL PRIMARY KEY,FName char(10),LName char(10),phone\_no char(10),Email char(30))"""

cur.execute(CTST)

#CTST is a variable includes creating table for students.

CTSU="""CREATE TABLE subjects (SubjID INT(9) NOT NULL PRIMARY KEY,Name char(20),Description varchar(20))"""

cur.execute(CTSU)

#CTSU is a variable includes creating table for subjects.

CTSE="""CREATE TABLE sections (SecID INT(9) NOT NULL,SubjID int(9),InstID int(9),Name char(20))"""

cur.execute(CTSE)

#CTSE is a variable includes creating table for sections.

#Preparing for linking between tables.

cur.execute("ALTER TABLE attendance ADD INDEX (StdID)")

cur.execute("ALTER TABLE attendance ADD INDEX (SecID)")

cur.execute("ALTER TABLE marks ADD INDEX (StdID)")

cur.execute("ALTER TABLE marks ADD INDEX (SecID)")

cur.execute("ALTER TABLE sections ADD INDEX (SecID)")

cur.execute("ALTER TABLE sections ADD INDEX (SubjID)")

cur.execute("ALTER TABLE sections ADD INDEX (InstID)")

#Linking tables.

cur.execute("ALTER TABLE attendance ADD FOREIGN KEY (StdID) REFERENCES sts.students (StdID)")

cur.execute("ALTER TABLE attendance ADD FOREIGN KEY (SecID) REFERENCES sts.sections (SecID)")

cur.execute("ALTER TABLE marks ADD FOREIGN KEY (StdID) REFERENCES sts.students (StdID)")

cur.execute("ALTER TABLE marks ADD FOREIGN KEY (SecID) REFERENCES sts.sections (SecID)")

cur.execute("ALTER TABLE sections ADD FOREIGN KEY (SubjID) REFERENCES sts.subjects (SubjID)")

cur.execute("ALTER TABLE sections ADD FOREIGN KEY (InstID) REFERENCES sts.instructors (InstID)")

except Exception:

print ("You have created your DataBase Before")

finally:

disconnect()

def CLS(Command):

try:

con=connect()

cur=con.cursor()

cur.execute(Command)

if "select" in Command :

print cur.fetchall()

else :

con.commit()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

disconnect()

def Export():

if os.path.exists("C:\\Users\\user\\Desktop\\BackUP\\sts"):

shutil.rmtree("C:\\Users\\user\\Desktop\\BackUP\\sts")

shutil.copytree("C:\\wamp\\bin\\mysql\\mysql5.5.20\\data\\sts","C:\\Users\\user\\Desktop\\BackUP\\sts")

def Import():

if os.path.exists("C:\\wamp\\bin\\mysql\\mysql5.5.20\\data\\sts"):

shutil.rmtree("C:\\wamp\\bin\\mysql\\mysql5.5.20\\data\\sts")

shutil.copytree("C:\\Users\\user\\Desktop\\BackUP\\sts","C:\\wamp\\bin\\mysql\\mysql5.5.20\\data\\sts")

**instructors operations:**

#Instructors operations

###########################

import MySQLdb

import DB\_OP

###########################

def inst(InstID,FName="NULL",LName="NULL",phone\_no="NULL",Email="NULL"):

try:

con=DB\_OP.connect()

cur=con.cursor()

cur.execute("""INSERT INTO instructors (InstID,FName,LName,phone\_no,Email) VALUES(%s,%s,%s,%s,%s)"""%(InstID,FName,LName,phone\_no,Email))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

con.commit()

DB\_OP.disconnect()

def update(InstID,NInstID,FName="NULL",LName="NULL",phone\_no="NULL",Email="NULL"):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.instructors SET InstID=%s,FName=%s,LName=%s,Phone\_no =%s,Email=%s WHERE instructors.InstID=%s"%(NInstID,FName,LName,phone\_no,Email,InstID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def query():

try:

con=DB\_OP.connect()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

Q=con.cursor()

Q.execute("select \* from instructors")

return Q.fetchall()

**Students operations:**

#students Operations

###########################

import MySQLdb

import DB\_OP

###########################

def std(StdID,FName="NULL",LName="NULL",phone\_no="NULL",Email="NULL"):

try:

con=DB\_OP.connect()

cur=con.cursor()

cur.execute("""INSERT INTO students (StdID,FName,LName,phone\_no,Email) VALUES(%s,%s,%s,%s,%s)"""%(StdID,FName,LName,phone\_no,Email))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

con.commit()

DB\_OP.disconnect()

def update(StdID,NStdID,FName,LName,phone\_no,Email):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.students SET StdID=%s,FName=%s,LName=%s,Phone\_no =%s,Email=%s WHERE students.StdID=%s"%(NStdID,FName,LName,phone\_no,Email,StdID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def query():

try:

con=DB\_OP.connect()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

Q=con.cursor()

Q.execute("select \* from students")

return Q.fetchall()

**Subjects operations:**

#Subjects Opeations

###########################

import MySQLdb

import DB\_OP

###########################

def subj(SubjID,Name="NULL",description="NULL"):

try:

con = DB\_OP.connect()

cur=con.cursor()

cur.execute("""INSERT INTO subjects (SubjID,Name,description) VALUES(%s,%s,%s)"""%(SubjID,Name,description))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

con.commit()

DB\_OP.disconnect()

def update(SubjID,NSubjID,Name,description):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.subjects SET SubjID=%s,Name=%s,description=%s WHERE subjects.SubjID=%s"%(NSubjID,Name,description,SubjID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def query():

try:

con=DB\_OP.connect()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

Q=con.cursor()

Q.execute("select \* from subjects")

return Q.fetchall()

def delete(SubjID):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("DELETE FROM subjects WHERE SubjID =%s"%(SubjID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

**Sections operations:**

#sections Operations

###########################

import MySQLdb

import DB\_OP

###########################

def sec(SecID,SubjID,InstID,Name="NULL"):

try:

con = DB\_OP.connect()

cur=con.cursor()

cur.execute("""INSERT INTO sections (SecID,SubjID,InstID,Name) VALUES(%s,%s,%s,%s)"""%(SecID,Name,SubjID,InstID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

con.commit()

DB\_OP.disconnect()

def query():

try:

con=DB\_OP.connect()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

Q=con.cursor()

Q.execute("select \* from sections")

return Q.fetchall()

def delete(SecID,SubjID,InstID):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("DELETE FROM sections WHERE SecID=%s AND SubjID =%s AND InstID=%s"%(SecID,SubjID,InstID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

**Attendance operations:**

#Attendance Opeations

###########################

import MySQLdb

import DB\_OP

###########################

def att(StdID,SecID,absents="NULL",RegDate="NULL"):

try:

con = DB\_OP.connect()

cur=con.cursor()

cur.execute("""INSERT INTO attendance (StdID,SecID,absents,RegDate) VALUES(%s,%s,%s,%s)"""%(StdID,SecID,absents,RegDate))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

con.commit()

DB\_OP.disconnect()

def update(StdID,SecID,NStdID,NSecID,absents,RegDate):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.attendance SET attendance.StdID=%s,attendance.SecID=%s,attendance.absents=%s,attendance.RegDate=%s WHERE attendance.StdID=%s AND attendance.SecID=%s"%(NStdID,NSecID,absents,RegDate,StdID,SecID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def query():

try:

con=DB\_OP.connect()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

Q=con.cursor()

Q.execute("select \* from attendance")

return Q.fetchall()

def delete(StdID,SecID):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("DELETE FROM attendance WHERE StdID=%s AND SecID =%s"%(StdID,SecID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def absents(StdID,SecID):

try:

con = DB\_OP.connect()

cur=con.cursor()

cur.execute("select absents from attendance where StdID=%s AND SecID=%s"%(StdID,SecID))

return cur.fetchall()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

DB\_OP.disconnect()

**Marks operations:**

#marks operations

###########################

import MySQLdb

import DB\_OP

###########################

def Fmark(StdID,SecID,FirstExamMark="NULL"):

try:

con=DB\_OP.connect()

cur=con.cursor()

cur.execute("""INSERT INTO marks (StdID,SecID,FirstExamMark) VALUES(%d,%d,%d)"""%(StdID,SecID,FirstExamMark))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

con.commit()

DB\_OP.disconnect()

def Smark(StdID,SecID,SecondExamMark="NULL"):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.marks SET SecondExamMark=%d WHERE marks.StdID=%d AND marks.SecID=%d"%(SecondExamMark,StdID,SecID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def Pmark(StdID,SecID,PartisMark="NULL"):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.marks SET PartisMark=%d WHERE marks.StdID=%d AND marks.SecID=%d"%(PartisMark,StdID,SecID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def Fimark(StdID,SecID,FinalExamMark="NULL"):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.marks SET FinalExamMark=%d WHERE marks.StdID=%d AND marks.SecID=%d"%(FinalExamMark,StdID,SecID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def update(StdID,SecID,NStdID,NSecID,FirstExamMark,SecondExamMark,PartisMark,FinalExamMark):

try:

edit = DB\_OP.connect()

cur=edit.cursor()

cur.execute("UPDATE sts.marks SET StdID=%d,SecID=%d,FirstExamMark=%d,SecondExamMark=%d,PartisMark=%d,FinalExamMark=%d WHERE marks.StdID=%d AND marks.SecID=%d"%(NStdID,NSecID,FirstExamMark,SecondExamMark,PartisMark,FinalExamMark,StdID,SecID))

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

finally:

edit.commit()

DB\_OP.disconnect()

def querymarks(StdID,SecID):

try:

con=DB\_OP.connect()

Q=con.cursor()

Q.execute("select \* from marks WHERE StdID =%d AND SecID=%d"%(StdID,SecID))

x= Q.fetchall()

return x

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

def query():

try:

con=DB\_OP.connect()

except MySQLdb.Error as err:

print("Something went wrong: {}".format(err[1]))

Q=con.cursor()

Q.execute("select \* from marks")

return Q.fetchall()

**Main File:**

import MySQLdb

import DB\_OP

import inst\_op

import marks\_op

import att\_op

import std\_op

import subj\_op

import sec\_op

print """

############################################

# #

# Welcome To Student Tracking System #

# #

############################################

"""

while True:

CNDB=raw\_input("To create your database please press 1 , If you already have a database press 2 \n")

CNDB="\""+CNDB+"\""

if "1" not in CNDB and "2" not in CNDB:

print "Sorry invalid input"

if "1" in CNDB:

DB\_OP.initializing()

break

elif "2" in CNDB :

break

while True:

print """

Please select your category:

(1) Instructors operations

(2) Students operations

(3) Subjects operations

(4) Sections operations

(5) Attendance operations

(6) Marks operations

(7) Make your own MySQL statement

(8) Backup your DataBase

(9) Load your backedup DataBase

(10) Exit

"""

select=input("\n")

if select==1:

while True:

print"""

Select your operation:

(1) Add new instructor

(2) Update instructos

(3) Show all instructors

(4) Exit

"""

Iselect=input("\n")

if Iselect == 1:

InstID=input("Please insert the Instructor ID: ")

FName=raw\_input("Please insert the instructor's First Name: ")

FName="\'"+FName+"\'"

LName=raw\_input("Please insert the instructor's Last Name: ")

LName="\'"+LName+"\'"

phone\_no=raw\_input("Please insert the instructor's phone number: ")

phone\_no="\'"+phone\_no+"\'"

Email=raw\_input("Please insert the instructor's Email: ")

Email="\'"+Email+"\'"

inst\_op.inst(InstID,FName,LName,phone\_no,Email)

elif Iselect == 2:

InstID=input("Please insert the old Instructor ID: ")

NInstID=input("Please insert the new Instructor ID: ")

FName=raw\_input("Please insert the new instructor's First Name: ")

FName="\'"+FName+"\'"

LName=raw\_input("Please insert the new instructor's Last Name: ")

LName="\'"+LName+"\'"

phone\_no=raw\_input("Please insert the new instructor's phone number: ")

phone\_no="\'"+phone\_no+"\'"

Email=raw\_input("Please insert the new instructor's Email: ")

Email="\'"+Email+"\'"

inst\_op.update(InstID,NInstID,FName,LName,phone\_no,Email)

elif Iselect == 3:

A=inst\_op.query()

print "----------------------------------------------------------------------------------------------------------------------------"

print ("|\tInstID\t | | FirstName\t | | LastName\t | | Phone No.\t | | Email |")

print "----------------------------------------------------------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","{:20}".format(val),"|",

print

print "----------------------------------------------------------------------------------------------------------------------------"

elif Iselect == 4:

break

elif select == 2:

while True:

print"""

Select your operation

(1) Add new student

(2) Update student

(3) Show all students

(4) Exit

"""

STselect=input("\n")

if STselect == 1:

StdID=input("Please insert the Student's ID: ")

FName=raw\_input("Please insert the Student's First Name: ")

FName="\'"+FName+"\'"

LName=raw\_input("Please insert the Student's Last Name: ")

LName="\'"+LName+"\'"

phone\_no=raw\_input("Please insert the Student's phone number: ")

phone\_no="\'"+phone\_no+"\'"

Email=raw\_input("Please insert the Student's Email: ")

Email="\'"+Email+"\'"

std\_op.std(StdID,FName,LName,phone\_no,Email)

elif STselect == 2:

StdID=input("Please insert the old student's ID: ")

NStdID=input("Please insert the new student's ID: ")

FName=raw\_input("Please insert the new Student's First Name: ")

FName="\'"+FName+"\'"

LName=raw\_input("Please insert the new Student's Last Name: ")

LName="\'"+LName+"\'"

phone\_no=raw\_input("Please insert the new Student's phone number: ")

phone\_no="\'"+phone\_no+"\'"

Email=raw\_input("Please insert the new Student's Email: ")

Email="\'"+Email+"\'"

std\_op.update(StdID,NStdID,FName,LName,phone\_no,Email)

elif STselect == 3:

A=std\_op.query()

print "----------------------------------------------------------------------------------------------------------------------------"

print ("|\tStdID \t | | FirstName\t | | LastName\t | | Phone No.\t | | Email |")

print "----------------------------------------------------------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","{:20}".format(val),"|",

print

print "----------------------------------------------------------------------------------------------------------------------------"

elif STselect == 4:

break

elif select == 3:

while True:

print """

Select your operation

(1) Add new subject

(2) Update subject

(3) Show all subjects

(4) Delete a subject

(5) Exit

"""

Suselect=input("\n")

if Suselect == 1:

SubjID=input("Please insert the subject's ID: ")

Name=raw\_input("Please Insert the subject's Name: ")

Name="\'"+Name+"\'"

description=raw\_input("Please insert the subject's description: ")

description="\'"+description+"\'"

subj\_op.subj(SubjID,Name,description)

elif Suselect == 2:

SubjID=input("Please insert the old subject's ID: ")

NSubjID=input("Please insert the new subject's ID: ")

Name=raw\_input("Please Insert the subject's Name: ")

Name="\'"+Name+"\'"

description=raw\_input("Please insert the subject's description: ")

description="\'"+description+"\'"

subj\_op.update(SubjID,NSubjID,Name,description)

elif Suselect == 3:

A=subj\_op.query()

print "--------------------------------------------------------------------------"

print ("|\tSubjID\t | | Subject's Name\t| | Description\t |")

print "--------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","{:20}".format(val),"|",

print

print "--------------------------------------------------------------------------"

elif Suselect == 4:

SubjID=input("Please insert the subject's ID you want to delete: ")

subj\_op.delete(SubjID)

elif Suselect == 5:

break

elif select == 4:

while True:

print"""

Select your operation

(1) Add new section

(2) Delete a section

(3) Show all sections

(4) Exit

"""

Seselect = input("\n")

if Seselect == 1:

SecID=input("Please insert the section's ID: ")

SubjID=input("Please insert the Subject that will be taught in this section: ")

InstID=input("Please insert the instructor's ID who will teach this Subject: ")

Name=raw\_input("Please insert the section's name: ")

Name="\'"+Name+"\'"

sec\_op.sec(SecID,SubjID,InstID,Name)

elif Seselect == 2:

SecID=input("Please insert the section's ID: ")

SubjID=input("Please insert the Subject's ID: ")

InstID=input("Please insert the instructor's ID: ")

sec\_op.delete(SecID,SubjID,InstID)

elif Seselect == 3:

A=sec\_op.query()

print "---------------------------------------------------------------------------------------------------"

print ("|\tSecID\t | | SubjID \t| | InstID \t | | Name\t\t |")

print "---------------------------------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","{:20}".format(val),"|",

print

print "---------------------------------------------------------------------------------------------------"

elif Seselect == 4:

break

elif select == 5:

while True:

print"""

Please select your operation

(1) Add new attendance file

(2) Update an attendance file

(3) Show all attendance files

(4) Delete an attendance file

(5) Exit

"""

Atselect=input("\n")

if Atselect == 1:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

absents=input("Please insert the absents of the student: ")

RegDate=raw\_input("Please insert the regestery date (yyyy-mm-dd)")

RegDate="\'"+RegDate+"\'"

att\_op.att(StdID,SecID,absents,RegDate)

elif Atselect == 2:

StdID=input("Please insert the student's old ID: ")

SecID=input("Please insert the section's old ID: ")

NStdID=input("Please insert the student's new ID: ")

NSecID=input("Please insert the section's new ID: ")

absents=input("Please insert the absents of the student: ")

RegDate=raw\_input("Please insert the regestery date (yyyy-mm-dd)")

RegDate="\'"+RegDate+"\'"

att\_op.update(StdID,SecID,NStdID,NSecID,absents,RegDate)

elif Atselect == 3:

A=att\_op.query()

print "-------------------------------------------------------------------------"

print ("|\t StdID \t| |\t SecID \t| |\tAbsents\t| |\tRegDate\t\t|")

print "-------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","\t","%s"%(val),"\t","|",

print

print "-------------------------------------------------------------------------"

elif Atselect == 4:

SecID=input("Please insert the section's ID: ")

StdID=input("Please insert the student's ID: ")

att\_op.delete(StdID,SecID)

elif Atselect == 5:

break

elif select == 6:

while True:

print"""

Please select your operation

(1) Add the first exam mark

(2) Add the second exam mark

(3) Add the participation mark

(4) Add the participation mark(depending on absents)

(5) Add the final exam mark

(6) Update marks

(7) Show marks for one student

(8) Show all marks

(9) Exit

"""

Maselect=input("\n")

if Maselect==1:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

FirstExamMark=input("Please insert the exam's mark: ")

marks\_op.Fmark(StdID,SecID,FirstExamMark)

elif Maselect == 2:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

SecondExamMark=input("Please insert the exam's mark: ")

marks\_op.Smark(StdID,SecID,SecondExamMark)

elif Maselect == 3:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

PartisMark=input("Please insert the participation mark: ")

marks\_op.Pmark(StdID,SecID,PartisMark)

elif Maselect == 4:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

PartisMark=input("Please insert the participation mark: ")

absents=int(att\_op.absents(StdID,SecID)[0][0])

Fpm=PartisMark-absents

marks\_op.Pmark(StdID,SecID,Fpm)

elif Maselect == 5:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

FinalExamMark=input("Please insert the exam's mark: ")

marks\_op.Fimark(StdID,SecID,FinalExamMark)

elif Maselect == 6:

StdID=input("Please insert the old student's ID: ")

SecID=input("Please insert the old section's ID: ")

NStdID=input("Please insert the new student's ID: ")

NSecID=input("Please insert the new section's ID: ")

FirstExamMark=input("Please insert the new first exam's mark: ")

SecondExamMark=input("Please insert the new second exam's mark: ")

PartisMark=input("Please insert the new participation mark: ")

FinalExamMark=input("Please insert the new final exam's mark: ")

marks\_op.update(StdID,SecID,NStdID,NSecID,FirstExamMark,SecondExamMark,PartisMark,FinalExamMark)

elif Maselect == 7:

StdID=input("Please insert the student's ID: ")

SecID=input("Please insert the section's ID: ")

A=marks\_op.querymarks(StdID,SecID)

print "-----------------------------------------------------------------------------------------------------------"

print ("| StdID | | SecID\t | | First Exam | | Second Exam | | Participation | | Final Exam |")

print "-----------------------------------------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","{:13}".format(val),"|",

print

print "-----------------------------------------------------------------------------------------------------------"

elif Maselect == 8:

A=marks\_op.query()

print "-----------------------------------------------------------------------------------------------------------"

print ("| StdID | | SecID\t | | First Exam | | Second Exam | | Participation | | Final Exam |")

print "-----------------------------------------------------------------------------------------------------------"

x=[]

for i in A:

x.append(list(i))

for i in range (0,len(x)):

x[i][0]=int(x[i][0])

for row in x:

for val in row:

print "|","{:13}".format(val),"|",

print

print "-----------------------------------------------------------------------------------------------------------"

elif Maselect == 9:

break

elif select == 7:

while True:

print"""

Select your operation

(1) Write your own MySQL statment

(2) Exit

"""

LL=input("\n")

if LL == 1:

Command=raw\_input("Write your MySQL statment please:\n")

DB\_OP.CLS(Command)

elif LL == 2:

break

elif select == 8:

DB\_OP.Export()

elif select == 9:

DB\_OP.Import()

elif select == 10:

print "Bye Bye :)"

break

Chapter Five

Conclusion

and

Future Work

**Conclusion and Future Work:**

**Conclusion:**

In Conclusion,

Based on research, most of instructors said that when they want to use the university's system they must be on their computers in the university and there is no way to use the system somewhere else, The "Student Tracking system" SOLVED it.

Cannot act as a pocket student's mark calculator for instructors and it does not auto calculate the assignment unlike the "student tracking system".

**Future Work:**

Because of the Lack of time I did not get my all goals.

* The Graphical user interface was in my priority, but I did not reach it so my one of my future plans is the GUI.
* And I must separate the administrator's system and the instructors system or do some logging functions.
* Connect the "student tracking system" with the registration department.
* Make the "student tracking system" a web application.
* Make the student a part of the system by giving every student an access for his marks.