COMPUTER SCIENCE PROJECT

-PRESENTED BY ALSALA AHMED, SANJEEV N., SIVARAMAN G.

BONAFIDE CERTIFICATE

Certified to be the Bonafide project work done by

_____

of Class 12 in_____ during the academic year: 2019-2020
at Saraswathi Vidyalaya Senior Secondary School,
Vadapalani,
Chennai-26

Dated Subject Instructor

Submitted for All India Senior Secondary Practical Examination held in _____ at Saraswathi Vidyalaya Senior Secondary School,
Chennai.

Dated External Examiner

Principal

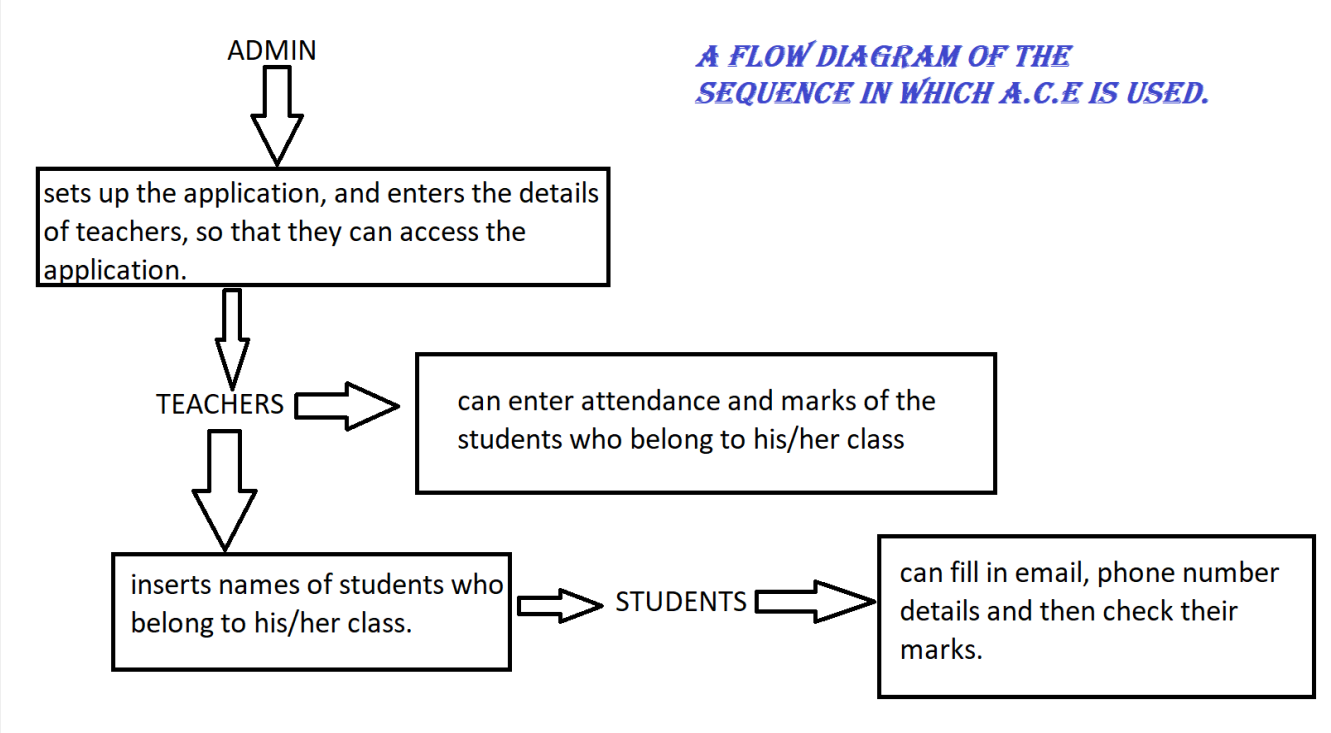
ACKNOWLEDGEMENT

We,_____ of Class XII would like to take this opportunity to Thank our Principal Mr.K.S.Natarajan for providing us with an incredible opportunity to explore the field of programming through this project. I would like to Thank the Academic coordinators for providing us with the resources and for the successful completion of the project. I would like to express my Sincere Gratitude to_____, _____Teacher without whom the project would not have been possible.

ATTENDANCE-***C***ONTROLLER-***E***XTRAORDINAIRE

Attendance Controller Extraordinaire (A. C. E) is a desktop application which lets the user maintain the attendance records of students for one academic year. It can also receive and store the exam marks of students.

The application will first be setup by an Admin. Next, once the admin gives access to teachers, the teachers can use the application. Once the teacher has entered the names of all students in his/her respective class, he/she can start marking their attendance and entering their test marks. Marked data can be viewed whenever required. Students can also view their marks.



-MODULES USED-

Programmed in Python, A.C.E. makes use of the MySQL-connector for Python to interact with the MySQL database. Pygame and Tkinter are the modules used for Graphical User Interface. The time and the calendar module, and the datetime library have also been used for dealing with date and time requirements. The sys module has also been used for small purposes. The rest of the modules are user-defined.

PROGRAMS:

Module: setuper.py - this is the program which helps the user setup the application.

```
sig = 'rep'
import mysql.connector as sql
import getpass
#accepting setup-input from admin
while sig != 'go':
    try:

        username = input('hello, user; please enter your name:\n>>>')
        Apwd = getpass.getpass(prompt = 'please set a password(atleast 7
characters) which you will use to access the application:\npasswords
entered wont be visible on the screen ;)\n>>>')
        sqluser = input('please enter your mysql username:\n>>>')
        sqlpass = getpass.getpass(prompt= 'enter your mysqlpassword,
please:\n>>>')
        Tpwd = getpass.getpass(prompt = 'configure a password for
teacher-access:\n>>>')

        conn = sql.connect(user = sqluser, password = sqlpass, host =
'127.0.0.1', port = 3306)

        if conn.is_connected() and username != '' and len(Tpwd) >= 7 and
len(Apwd) >= 7 :
            sig = 'go'
        elif not conn.is_connected():
            print('sql username - sql password mismatch\n***dear user, pls
check your sql password***\n')
            while sig != 'go':
                sqluser = input('please enter your mysql username:\n>>>')
                sqlpass = getpass.getpass(prompt= 'please enter correct
mysql password:\n>>>')
                conn = sql.connect(user = sqluser, password = sqlpass,
host = '127.0.0.1', port = 3306)
                if conn.is_connected():
                    sig = 'go'
                else:
```

```
                    print('wrong sql password for the username')

        elif username == '':
            print('***dear user, pls enter your name!***')
            while sig != 'go':
                username = input('please enter your name:\n>>>')
                if username != '':
                    sig = 'go'
                else:
                    pass


        elif len(Apwd) < 7:
            print('your password is too short :(')
            while sig != 'go':
                Apwd = getpass.getpass(prompt='please enter a password
with atleast 7 characters')
                if len(Apwd) < 7:
                    print('too short. pls enter a longer password')
                else:
                    sig = 'go'


        elif len(Tpwd) < 7:
            print('the password for Teachers is too short :(')
            while sig != 'go':
                Tpwd = getpass.getpass('please enter a teacher-password
with atleast 7 characters')
                if len(Tpwd) < 7:
                    print('too short. pls enter a longer
teacher-password')
                else:
                    sig = 'go'

    except Exception as e:
        print(e)
        print('sorry :( \nsomething went wrong. please try again')
        sig = 'rep'
        continue
```

```
    if conn.is_connected() and username != '' and len(Tpwd) >= 7 and
len(Apwd) >= 7 :
        sig = 'go'
    else:
        sig = 'rep'
        print('sorry you will have to do it all over again.')

curs = conn.cursor()
print('setting up.....')

import pickle
f = open('D:\Ace\Ace.dat', 'wb')
a = [sqluser, sqlpass, Tpwd]
pickle.dump(a, f)
f.close()

for i in ['create database attadmin', 'use attadmin']:
    curs.execute(i)
curs.execute('create table admin(Aname varchar(20), status varchar(15),
pass varchar(70))')
curs.execute('insert into admin values(%s, %s, %s)', (username, 'admin',
Apwd))
curs.execute("create table student(Sname varchar(20), Sclass varchar(10),
Semail varchar(25), Srfid varchar(20));")
curs.execute("create table teacher(Tname varchar(20), Tclass varchar(15),
TID varchar(12), password varchar(15), visit int);")
curs.execute("create table smarks(Sname varchar(20), Sclass varchar(10),
examname varchar(20), physics int, chemistry int, maths int,
computer_science int, english int, total int);")
conn.commit()
print('password inserted.')

sig = 'rep'
while sig != 'go':
    while sig != 'go':

        NoC = int(input("please enter the number of classes for which you
would like to use this application:\n>>>"))

        if NoC > 0:
```

```
            print("okay. enter the ", NoC, " class(es):\n>>>")
            sig = 'go'
        elif NoC == 0:
            print('please enter a finite no of classes...')
            sig = 'rep'
            continue
    Clist = []
    sig = 'rep'
    for r in range(1, NoC + 1):
        while sig != 'go':
            x = input(str(r) + ". ")
            if x != '':
                Clist += [x]
            else:
                print('sorry, cant accept blank class names. try
again.\n')
                continue
            sig = 'go'
        sig = 'rep'
    sig = 'go'

print('preparing to create tables for the classes...')
for c in Clist:
    curs.execute('create table ' + c + '(Sname varchar(25), toddate date,
attendance varchar(12), Srfid varchar(20));')
print('done. you are free to go.')
```

Module: ace.py - This is the heart of the application, wherein all modules are used together to run A.C.E.

```
#module for user interface
#

import mysql.connector as sql
from time import sleep

###***read the afile.txt ; based on its contents, decide whether to launch
setup, or to run ACE. ***###
decide = open('D:\Ace\Afile.txt', 'r')
choice = decide.read()
decide.close()

if choice == 'one':
   import pygame, sys, Teacher, datetime, Admin, Awindow, attdate, filer,
Student, markwindow

   www = filer.p

   connection = sql.connect(user = www[0], password = www[1], host =
'127.0.0.1', port = 3306, database = 'attadmin')
   firsttime = True

   char, page = '', 'loginpage'
   bcol = (0, 255, 0)
   from pygame.locals import *
   pygame.init()
   #print()
   font = pygame.font.Font('freesansbold.ttf', 32)
   f1 = pygame.font.Font('freesansbold.ttf', 22)
   f2 = pygame.font.Font('freesansbold.ttf', 18)
   algerian = pygame.font.Font('ALGER.ttf', 18)
   algerian32 = pygame.font.Font('ALGER.ttf', 32)
   tango = pygame.font.Font('tango.ttf', 22)
   monterey = pygame.font.Font('MontereyFLF.ttf', 32)
   assassin = pygame.font.Font('Assassin$.ttf', 32, italic = True)
   assassin18 = pygame.font.Font('Assassin$.ttf', 18, italic = True)
```

```
   vertigo = pygame.font.Font('VertigoFLF.ttf', 32, italic = True, bold =
True)
   bradhitc = pygame.font.Font('BRADHITC.ttf', 22, bold = True)
   mistral = pygame.font.Font('MISTRAL.ttf', 32)
   al = {8 : '', 97 : 'a', 98 : 'b', 99 : 'c', 100 : 'd', 101 : 'e', 102 :
'f', 103 : 'g', 104 : 'h', 105 : 'i', 106 : 'j', 107 : 'k', 108 : 'l', 109
: 'm', 110 : 'n', 111 : 'o', 112 : 'p', 113 : 'q', 114 : 'r', 115 : 's',
116 : 't', 117 : 'u', 118 : 'v', 119 : 'w', 120 : 'x', 121 : 'y', 122 :
'z', 48 : '0', 49 : '1', 50 : '2', 51 : '3', 53 : '4', 54 : '5', 55 : '6',
56 : '7', 57 : '8', 59 : '9'}
   wind = pygame.display.set_mode((900, 900))
   icon = pygame.image.load('D:\Ace\ice0.jpg')
   pygame.display.set_caption('A.C.E.')
   pygame.display.set_icon(icon)

   def txtdisp(dtxt, txtloc, f = font, tcol = (0, 0, 255), bcol = (0, 255,
0)):#for
      txt = f.render(dtxt, True, tcol, bcol)
      txtcoord = txt.get_rect()

      txtcoord.center = txtloc
      wind.blit(txt, txtcoord)
      return (txtloc[0] - (txtcoord[2]//2), txtloc[0] + (txtcoord[2]//2),
txtloc[1] - (txtcoord[3]//2), txtloc[1] + (txtcoord[3]//2))


   def accreq(ekey, char):
      txtdisp('please enter your name;', (450, 150), f = monterey)
      txtdisp('ATTENDANCE CONTROLLER EXTRAORDINAIRE', (450, 75), f =
assassin)
      pic = pygame.image.load('ce0.png')
      wind.blit(pic, (350, 180))

      if char != '' :
         if ekey == 8:

            tem = list(char)

            tem.pop()
            char = ''
```

```
            for i in tem:
               char += i
            txtdisp(char,(450, 450), f = monterey)

            return char
         else:
            char += al[ekey]
            txtdisp(char, (450, 450), f = monterey)
            return char

      else:
         char += al[ekey]
         txtdisp(char, (450, 450), f = monterey)

         return char

   def pwdreq(ekey, char):
      txtdisp('please enter your access password;', (450, 200), f =
vertigo)
      txtdisp('ATTENDANCE CONTROLLER EXTRAORDINAIRE', (450, 75), f =
assassin)

      if char != '' :
         if ekey == 8:

            tem = list(char)

            tem.pop()
            char = ''
            for i in tem:
               char += i
            txtdisp('*' * len(char),(450, 450), f = vertigo)

            return char
         else:
            char += al[ekey]
            txtdisp('*' * len(char), (450, 450), f = vertigo)
            return char

      else:
```

```
        char += al[ekey]
        txtdisp('*', (450, 450), f = vertigo)

        return char

   def accden():
      txtdisp('sorry...no.\ntry once more', (450, 200), f = assassin)
      sleep(3)
      wind.fill((0, 0, 0))
      txtdisp('please enter your name;', (450, 200), f = monterey)
      return 'loginpage'

   def button(page, event, dtxt, txtloc, bcol = (0, 255, 0),  f =
monterey, tcol = (0, 0, 255)):
      pointlist = txtdisp(dtxt, txtloc, f, tcol, bcol)
      if event.type == pygame.MOUSEBUTTONDOWN:
         if pygame.mouse.get_pos()[0] in range(pointlist[0], pointlist[1])
and pygame.mouse.get_pos()[1] in range(pointlist[2], pointlist[3]):
            if pygame.BUTTON_LEFT:

               txtdisp(dtxt, txtloc, f, tcol, bcol = (0, 0, 0))
               page = dtxt

               return page
            else:
               return page
         else:
            return page
      return page



   def updbutton(toddate, connection, event, dtxt, Sname, txtloc, bcol =
(0, 255, 0),  f = monterey, tcol = (0, 0, 255)):

      pointlist = txtdisp(dtxt, txtloc, f, tcol, bcol)
      if event.type == pygame.MOUSEBUTTONDOWN:
         if pygame.mouse.get_pos()[0] in range(pointlist[0], pointlist[1])
and pygame.mouse.get_pos()[1] in range(pointlist[2], pointlist[3]):
            if pygame.BUTTON_LEFT:
```

```
                wind.fill((220, 220, 20))
                txtdisp(dtxt, txtloc, f, tcol, bcol = (0, 0, 0))
                Teacher.upd(connection,Tname = nm, dtxt = dtxt, Sname =
Sname, toddate = toddate)


   def widgetbutton(event, dtxt, txtloc, widget, bcol = (0, 255, 0),  f =
monterey, tcol = (0, 0, 255)):
      pointlist = txtdisp(dtxt, txtloc, f, tcol, bcol)

      if event.type == pygame.MOUSEBUTTONDOWN:
         if pygame.mouse.get_pos()[0] in range(pointlist[0], pointlist[1])
and pygame.mouse.get_pos()[1] in range(pointlist[2], pointlist[3]):
            if pygame.BUTTON_LEFT:
               txtdisp(dtxt, txtloc, f, tcol, bcol = (0, 0, 0))
               widget = int(dtxt)
               return widget
            return widget
         return widget
      return widget


   def viewstudentdata(page, event):
      wind.fill((220, 220, 20))
      att = 'check/update attendance'
      page = button(page, event, att, (450, 200), bcol, f = assassin)


      return page

   def accgtd(event, user):
      sleep(1)
      txtdisp('access granted to ' + nm + "'s files ", (450, 90), f =
mistral)
      txtdisp(user + " access", (450, 50), f = mistral)
      return 'tmenu'

   def Tmenu(page, event):
      studata = 'view student data'
```

```
      yrdata = 'manage student marks'
      lo = 'loginpage'
      page = button(page, event, studata, (450, 150), bcol, f = bradhitc)
      pic = pygame.image.load('ce0.png')
      wind.blit(pic, (350, 180))
      page = button(page, event, yrdata, (450, 450), bcol, f = bradhitc)
      page = button(page, event, lo, (450, 550), bcol = (0, 0, 0), f =
monterey)
      if page == 'loginpage':
         wind.fill((0, 0, 0))
         txtdisp('please enter your name;', (450, 200), f = monterey)
      return page

   def Amenu(page, event):
      page = button(page, event, 'Manage Staff', (450, 150), bcol, f =
tango)
      page = button(page, event, 'loginpage', (450, 350), bcol = (0, 0,
0), f = monterey)
      page = button(page, event, 'reset app.', (450, 500), bcol, f =
assassin)
      if page == 'loginpage':
         wind.fill((0, 0, 0))
         txtdisp('please enter your name;', (450, 200), f = monterey)
      if page == 'reset app.':
         wind.fill((255, 0, 0))
      return page

   def Smenu(page, event, nm):
      txtdisp("Student Access", (450, 50), f = vertigo)
      txtdisp("WELCOME! " + nm, (450, 200), f = mistral)
      pic = pygame.image.load('ce0.png')
      wind.blit(pic, (350, 250))
      page = button(page, event, 'Manage your account', (450, 490), bcol,
f = tango)
      page = button(page, event, 'loginpage', (450, 550), bcol = (0, 0,
0), f = monterey)
      if page == 'loginpage':
         wind.fill((0, 0, 0))
         txtdisp('please enter your name;', (450, 200), monterey)
      return page
```

```
   def namedisplay(toddate, widget, user, page, namelist, event, bcol =
(0, 255, 0),  f = font, tcol = (0, 0, 255)):
      connection.commit()

      q = 0

      txtdisp('STUDENT', (72, 25), f = assassin18, tcol = (0, 0, 0))
      txtdisp('ATTENDANCE', (197, 25), f = assassin18)
      txtdisp(str(toddate), (447, 25), f = f2)
      for i in namelist:
         txtdisp(i[0], (72, 50 + q*50), f = tango, tcol = (0, 0, 0))
         txtdisp(i[1], (197, 50 + q*50), f = tango)

         updbutton(toddate, connection, event, 'present', i[0], (322, 50 +
q*50), f = algerian, bcol = (250, 150, 100))
         updbutton(toddate, connection, event, 'absent', i[0], (447, 50 +
q*50), f = algerian, bcol = (250, 150, 100))

         q += 1
      page = button(page, event, 'loginpage', (450, 550), bcol, f =
assassin)
      page = button(page, event, 'select date', (450, 650), bcol, f =
mistral)
      page = button(page, event, 'tmenu', (650, 550), bcol, f =
assassin18)
      pic = pygame.image.load('ce0.png')
      wind.blit(pic, (650, 50))

      widget = widgetbutton(event, str(widget + 1) , (465, 485), widget,
bcol = (0, 255, 0),  f = algerian32, tcol = (0, 0, 0))
      if widget > 0:
         widget = widgetbutton(event, str(widget - 1), (20, 485), widget,
bcol = (0, 255, 0),  f = algerian32, tcol = (0, 0, 0))
      if page == 'loginpage':
         wind.fill((0, 0, 0))
         txtdisp('please enter your name;', (450, 200), f = monterey)

      elif page == 'tmenu':
```

```
        wind.fill((255, 0, 0))
        page = accgtd(event, user)

    return (page, widget)
  def marksfn(event, widget, page, toddate, display_list = []):
    display_list = []
    pic = pygame.image.load('ce0.png')
    wind.blit(pic, (650, 50))
    try:
        for i in Teacher.nl(connection, Tname = nm, toddate =
toddate)[widget]:

            display_list += [i[0]]
    except IndexError:
        #widget -= 1
        pass

    widget = widgetbutton(event, str(widget + 1) , (465, 485), widget,
bcol = (0, 255, 0),  f = algerian, tcol = (0, 0, 0))
    if widget > 0:
        widget = widgetbutton(event, str(widget - 1), (20, 485), widget,
bcol = (0, 255, 0),  f = algerian, tcol = (0, 0, 0))

    markpage, q = '', 0
    ###markpage is the variable which will store the name of a student;
    txtdisp('STUDENTS', (250, 25), f = assassin, tcol = (0, 0, 0))

    for k in display_list:
        markpage = button(markpage, event, k, (250, 50 + q), f =
algerian)
        q += 50
    if bool(markpage):
        markwindow.main(markpage, Teacher.main(connection, Tname =
nm)[0], usertype = 'TEACHER')
    page = 'manage student marks'
    page = button(page, event, 'loginpage', (450, 550), bcol = (0, 0,
0), f = monterey)
    page = button(page, event, 'tmenu', (650, 550), bcol, f = bradhitc)

    return [page, widget]
```

```
   txtdisp('please enter your name;', (450, 200), f = monterey)
   while True:#main loop which executes everything

      for event in pygame.event.get():

         if event.type == QUIT:
            pygame.quit()
            sys.exit()

         elif page == 'loginpage' and event.type == pygame.KEYDOWN:


            if event.key != 13:

               if event.key in al.keys():
                  wind.fill((0, 0, 0))
                  char = accreq(event.key, char)
               else:
                  continue

               continue

            elif event.key == 13:


               if char in Teacher.main(connection, Tname = char)[4]:
                  wind.fill((0, 0, 0))
                  txtdisp('please enter your access password;', (450,
200), f = f1)
                  nm = char
                  char = ''
                  page = 'Tpwdpage'
```

```
                elif char in Admin.getAlist():
                   wind.fill((255, 255, 255))
                   txtdisp('please enter your access password;', (450,
200), f = f1)
                   nm = char
                   char = ''
                   page = 'Apwdpage'
                elif char in Student.getSlist():
                   wind.fill((100, 100, 100))
                   txtdisp('WELCOME! ' + char, (450, 200), f = f1)
                   nm = char
                   char = ''
                   page = 'Smenu'


                else:

                   #wind.fill((0, 0, 0))

                   char = ''
                   page = 'accden'

         elif page == 'Apwdpage' and event.type == pygame.KEYDOWN:


            if event.key != 13:
               if event.key in al.keys():
                  wind.fill((255, 255, 255))
                  char = pwdreq(event.key, char)
               else:
                  continue

            elif event.key == 13:
               if char == Admin.getApass(nm):

                  wind.fill((255, 0, 0))
                  char = ''
                  user = 'ADMIN'
                  sleep(1)
```

```
                  txtdisp('access granted to ' + nm + "'s files ", (450,
90))
                  txtdisp(user + " access", (450, 50), f = f2)
                  page = 'Amenu'
               else:
                  wind.fill((0, 0, 0))
                  char = ''
                  page = 'accden'

         elif page == 'Smenu':
            page = Smenu(page, event, nm)

         elif page == 'reset app.':
            check = Admin.getApass(nm)
            import getpass
            con = getpass.getpass(prompt = 'requesting confirmation to
reset A.C.E.;\ndear admin, please enter your ACE password: ')
            print('password recieved...')
            if con == check:
               print('authentication approved.')
               import resetter
               wind.fill((200, 200, 200))
               txtdisp("closing A.C.E.", (450, 450), f = vertigo, bcol=
(255, 0, 0))
               sleep(5)
               pygame.quit()
               sys.exit()
            else:
               print('authentication denied!')
               sleep(5)
               sys.exit()
               pygame.quit()

         elif page == 'Tpwdpage' and event.type == pygame.KEYDOWN:

            if event.key != 13:
               if event.key in al.keys():
                  wind.fill((0, 0, 0))
                  char = pwdreq(event.key, char)
               else:
```

```
                continue

            elif event.key == 13:
                if char == Teacher.main(connection, Tname = nm)[3]:
                    cur = connection.cursor()
                    cur.execute("select visit from Teacher where Tname =
%s", (nm, ))
                    if cur.fetchone()[0] == 0:
                        wind.fill((0, 0, 0))
                        sleep(3)
                        Teacher.classrender(connection, Tname = nm)
                        connection.close()
                        pygame.quit()
                        sys.exit()
                    else:
                        wind.fill((255, 0, 0))
                        char = ''
                        user = 'TEACHER'
                        page = accgtd(event, user)
                else:
                    #wind.fill((0, 0, 0))
                    char = ''
                    page = 'accden'


        elif page == 'accden':
            page = accden()

        elif page == 'Manage your account':
            wind.fill((100, 100, 100))
            import Sinput
            Sinput.fn(nm, Teacher.main(connection, Tname = nm)[0])
            page = 'Smenu'

        elif page == 'Amenu':
            page = Amenu(page, event)

        elif page == 'tmenu':

            page = Tmenu(page, event)
```

```
            if page == 'loginpage':
               wind.fill((0, 0, 0))
               txtdisp('please enter your name;', (450, 200))

         elif page == 'Manage Staff':
            Awindow.Afun()
            page = 'Amenu'
         elif page == 'select date':
            toddate = attdate.main()
            wind.fill((220, 220, 20))
            page = 'check/update attendance'

         elif page == 'view student data':

            page = viewstudentdata(page, event)
            if page == 'check/update attendance':
               wind.fill((220, 220, 20))

         elif page == 'manage student marks': ###in progress
            wind.fill((220, 220, 20))
            markpage, q = '', 0
            if firsttime:
               widget = 0
               toddate = datetime.date.today()

            firsttime = False
            #widget = 0#work here;
            ###markpage is the variable which will store the name of a
student;
            variable = marksfn(event, widget, page, toddate)
            page, widget = variable[0], variable[1]
            if page == 'loginpage':
               wind.fill((0, 0, 0))
               txtdisp('please enter your name;', (450, 200))
               firsttime = True

            elif page == 'tmenu':
               wind.fill((255, 0, 0))
               page = accgtd(event, user)
               firsttime = True
```

```
        elif page == 'check/update attendance':

            if firsttime:
                widget = 0

                toddate = attdate.main()

            namelist = Teacher.nl(connection, Tname = nm, toddate =
toddate)


            if widget < len(namelist):
                wind.fill((220, 220, 20))
                pw = namedisplay(toddate, widget, user, page,
namelist[widget], event)
            else:
                wind.fill((220, 220, 20))
                pw = namedisplay(toddate, widget - 1, user, page,
namelist[widget - 1], event)

            page = pw[0]
            widget = pw[1]
            firsttime = False
            pass


      pygame.display.update()

else:
   import setuper
   decide = open('D:\Ace\Afile.txt', 'w')
   decide.write('one')
   decide.close()
   print('all ready. pls login again.')
   sleep(5)
```

Module: Admin.py - this is the module used for retrieving Admin related data.

```
import mysql.connector as sql
import filer
www = filer.p
connection = sql.connect(user = www[0], password = www[1], host =
'127.0.0.1', port = 3306, database = 'attadmin')
curs = connection.cursor()
def getAlist():
    curs.execute('select Aname from Admin;')

    return curs.fetchone()

def getApass(Aname):
    curs.execute('select pass from Admin where Aname = %s', (Aname, ))
    return curs.fetchone()[0]
```

Module: Awindow.py - the Module which presides over the graphical user interface for the Admin.

```
from tkinter.ttk import *
from tkinter import *
import mysql.connector, filer
from tkinter import messagebox
global www
www = filer.p
def Afun():

    mydb=mysql.connector.connect(host="localhost", user=www[0],
password=www[1], port = 3306, database = "attadmin")
    mycursor=mydb.cursor()


    def Register():

        TID=e3.get()
        dbst_id=""
        Select="select TID from Teacher;"
        mycursor.execute(Select)
        result=mycursor.fetchall()
        signal = 'go'
        tablist = []
        mycursor.execute('show tables;')
        for i in mycursor.fetchall():
            tablist.append(i[0])

        if(TID in result):
            messagebox.askokcancel("Information","That ID Already exists")
            signal = ''
        if signal == 'go':

            Insert="Insert into Teacher(Tname, Tclass, TID, password,
visit) values(%s, %s, %s, %s, 0)"
            Tname=e1.get()
            Tclass = e2.get()
```

```
            if(Tname != '' and Tclass in tablist):

                Values=(Tname, Tclass, TID, www[2])
                mycursor.execute(Insert,Values)
                mydb.commit()
                messagebox.askokcancel("Information","Record inserted")
                e1.delete(0, END)
                e2.delete(0, END)
                e3.delete(0, END)
            elif (Tname == "" or TID == ''):
                messagebox.askokcancel("Information", "Some fields are
left blank. pls try again :)")
            elif Tclass not in tablist:
                messagebox.askokcancel("Information","unregistered class
name!!!")
                e1.delete(0, END)
                e2.delete(0, END)
                e3.delete(0, END)


    def ShowRecord():
        TID=e3.get()
        dbst_id=""
        Select="select TID from Teacher"

        mycursor.execute(Select)
        result1=mycursor.fetchall()

        Select1="select * from Teacher where TID = %s;"
        mycursor.execute(Select1, (TID, ))
        result2=mycursor.fetchall()
        Tname = ''
        Tclass = ''
        e1.delete(0, END)
        e2.delete(0, END)
        if((TID, ) in result1):

            for i in result2:
                Tname = i[0]
```

```
                Tclass = i[1]
                e1.insert(0,Tname)
                e2.insert(0, Tclass)
        elif TID == '':
            messagebox.showinfo("Information","Teacher ID unentered...")
        else:
            messagebox.askokcancel("Information","No Record exists")
    def Delete():
        TID=e3.get()
        if TID != '':
            try:
                mycursor.execute("delete from Teacher where TID = %s",
(TID, ))
                messagebox.showinfo("Information","Record Deleted")
            except:
                messagebox.showinfo("Information","Record not Deleted")
            mydb.commit()
        elif TID == '':
            messagebox.showinfo("Information","Teacher ID unentered...")


        e3.delete(0, END)

    def Update():
        Tname=e1.get()
        Tclass=e2.get()
        TID=e3.get()
        if TID != '':
            if Tname != '':
                mycursor.execute("update Teacher set Tname = %s where TID
= %s", (Tname, TID))
                mydb.commit()
                messagebox.showinfo("Info","Record Updated: " + Tname)
            if Tclass != '':
                mycursor.execute('select Tclass from Teacher where TID =
%s', (TID, ))
                var = mycursor.fetchone()
                print(var[0])
                mycursor.execute('update Teacher set Tclass = %s where TID
= %s', (Tclass, TID))
```

```
                mycursor.execute('alter table ' + var[0] + ' rename to ' +
Tclass)
                mydb.commit()
                messagebox.showinfo("Info","Record Updated: " + Tclass)
            if Tname == '' and Tclass == '':
                messagebox.showinfo("Info","Please enter atleast one
record to be updated;")
        elif TID == '':
                messagebox.showinfo("Information","Teacher ID
Unentered...")


    def Showall():
        class A(Frame):
            def __init__(self, parent):
                Frame.__init__(self, parent)
                self.CreateUI()
                self.LoadTable()
                self.grid(sticky=(N, S, W, E))
                parent.grid_rowconfigure(0, weight=1)
                parent.grid_columnconfigure(0, weight=1)
            def CreateUI(self):
                tv= Treeview(self)
                tv['columns']=("Teacher's Name", "Teacher's class",
"Teacher's ID")

                tv.heading('#0',text='Teacher Name',anchor='center')
                tv.column('#0',anchor='center')
                tv.heading('#1', text='Teacher Class', anchor='center')
                tv.column('#1', anchor='center')
                tv.heading('#2', text='Teacher ID', anchor='center')
                tv.column('#2', anchor='center')

                tv.grid(sticky=(N,S,W,E))
                self.treeview = tv
                self.grid_rowconfigure(0,weight=1)
                self.grid_columnconfigure(0,weight=1)
            def LoadTable(self):
                Select="Select * from Teacher"
```

```
                mycursor.execute(Select)
                result=mycursor.fetchall()
                Tname = ''
                Tclass = ''
                TID = ''
                for i in result:
                    Tname = i[0]
                    Tclass = i[1]
                    TID=i[2]

                    self.treeview.insert("",'end',text= Tname
,values=(Tclass, TID))
        root=Tk()
        root.title("Overview Psubject")
        A(root)
    def Clear():
        e1.delete(0, END)
        e2.delete(0, END)
        e3.delete(0, END)
    def help():
        messagebox.showinfo('HELP', "to Register staff members, enter info
into the columns, and press register\nDelete: enter the ID of the teacher
whose record needs to be deleted, and press delete.\nShow Record: enter
the ID which you want to read, and press Show record.\nShow All: just
click Show All to view all records\nClear: click clear to clear all the
fields.\nTo Update: enter the Teacher ID of the entry which needs to be
updated, and enter the new name/class.")

    root=Tk()
    root.title("Staff Data")
    root.geometry("1200x700")
    root.configure(background = 'black')

    photo=PhotoImage(file='bpic.png')

    Label(root,image=photo).place(relwidth = 1, relheight = 1)
    label1=Label(root,text="Teacher
Name",width=20,height=2,bg="pink").grid(row=0,column=0)
    label2=Label(root,text="Teacher's
class",width=20,height=2,bg="pink").grid(row=1,column=0)
```

```
    label3=Label(root,text="Teacher's
ID",width=20,height=2,bg="pink").grid(row=2,column=0)

    e1=Entry(root,width=30,borderwidth=8)
    e1.grid(row=0,column=1)
    e2=Entry(root,width=30,borderwidth=8)
    e2.grid(row=1,column=1)
    e3=Entry(root,width=30,borderwidth=8)
    e3.grid(row=2,column=1)


button1=Button(root,text="Register",width=10,height=2,command=Register).gr
id(row=7,column=0)

button2=Button(root,text="Delete",width=10,height=2,command=Delete).grid(r
ow=7,column=1)

button3=Button(root,text="Update",width=10,height=2,command=Update).grid(r
ow=7,column=3)
    button4=Button(root,text="Show
record",width=10,height=2,command=ShowRecord).grid(row=7,column=5)
    button5=Button(root,text="Show
All",width=10,height=2,command=Showall).grid(row=7,column=7)

button6=Button(root,text="Clear",width=10,height=2,command=Clear).grid(row
=7,column=9)
    button7 =
Button(root,text="HELP",width=10,height=2,command=help).grid(row=6,column=
9)

    root.mainloop()
```

Module: attdate.py - the module used for selecting the desired date for which attendance needs to be marked or viewed.

```
from tkcalendar import *
from tkinter import *
import datetime
def main():
    root = Tk()
    calwind = Calendar(root, selectmode = 'day', date_pattern = 'y-mm-dd')
    calwind.pack(pady = 20)
    labe = Label(root, text = '')
    labe.pack(pady = 10)
    def getdate():
        global D
        D = calwind.get_date()

        labe.config(text = 'selected date : ' + D)


    b = Button(root, text = 'select', command = getdate)
    b.pack(pady = 20)
    root.mainloop()
    try:
        print('D is: ', D)
        return D

    except:
        return str(datetime.datetime.today()).split(' ')[0]
```

Module: Teacher.py - This is the module used for accessing a lot of data from the database, which would be required for the Teacher to use.

```
###module for managing user access to database tables, (especially the
teacher table)
dt1 = {6 : (30, 30), 7 : (31, 31), 8 : (31, 31), 9 : (30, 30), 10 : (31,
31), 11 : (30, 30), 12 : (31, 31), 1 : (31, 31), 2 : (28, 29), 3 : (31,
31), 4 : (30, 30), 5 : (31, 31)}
import mysql.connector as sql
import datetime, filer
Tlist = ['mrx']
Tpass = 'password'
tab = 'employee'
www = filer.p
class Tmain:

   def __init__(self, connection, Tname, Sname, dtxt, toddate):
      self.toddate = toddate
      self.conn = connection
      self.Tname = Tname
      self.Sname = Sname
      self.dtxt = dtxt

   def getTpass(self):
      T = self.Tname
      curs = self.conn.cursor()
      curs.execute('select password from Teacher where Tname =
"{}"'.format(T))
      mno = curs.fetchone()
      if mno != None:
         return mno[0]
      else:
         return ''
      self.conn.commit()

   def getTlist(self):
      Tlist = []
      curs = self.conn.cursor()
```

```
      curs.execute('select Tname from Teacher')
      mno = curs.fetchall()
      if mno != None:
         for i in mno:
            Tlist += [i[0]]
         print('list of teachers: ', Tlist)
         return Tlist
      else:
         return ''
      self.conn.commit()



   def getclass(self): #function to get teacher's class
      T = self.Tname
      curs = self.conn.cursor()
      curs.execute('select Tclass from Teacher where Tname =
"{}"'.format(T))
      mno = curs.fetchone()
      if mno != None:
         return mno[0]
      else:
         return ''

      self.conn.commit()

   def getstulist(self):  # to get the list of students belonging to a
specific class

      stulist = []
      curs = self.conn.cursor()
      T = self.Tname
      curs.execute('select Tclass from Teacher where Tname =
"{}"'.format(T))
      pqr = curs.fetchone()
      if pqr != None:
         C = pqr[0]
      elif pqr == None:
         C = ''
```

```
      curs.execute('select Sname from Student where Sclass =
"{}"'.format(C))
      mno = curs.fetchall()
      if mno != None:
         for i in mno:
            stulist += [i[0]]
         return stulist
      else:
         return ''

   def Updater(self): #updating attendance into database

      curs = self.conn.cursor()
      T = self.Tname
      print("OK", T, "computer")
      curs.execute('select Tclass from Teacher where Tname =
"{}"'.format(T))
      pqr = curs.fetchone()
      if pqr != None:
         C = pqr[0]
      elif pqr == None:
         C = ''
      selcomm = 'update ' + C + ' set attendance = %s where Sname = %s and
toddate = %s;'

      curs.execute(selcomm, (self.dtxt, self.Sname, self.toddate))
      self.conn.commit()
      pass

   def getnamelist(self):
      count, lis1, lis2 = 0, [], []
      curs = self.conn.cursor()
      T = self.Tname
      curs.execute('select Tclass from Teacher where Tname =
"{}"'.format(T))
      pqr = curs.fetchone()
      if pqr != None:
         C = pqr[0]
      elif pqr == None:
         C = ''
```

```
      selcomm = 'select Sname, attendance from ' + C + ' where toddate =
"{}"'
      print(self.toddate)
      curs.execute(selcomm.format(self.toddate))
      r = curs.fetchall()
      cou = 0

      for k in range(0, len(r), 5):

         while cou < 5:

            if count < len(r):
               lis1 += [r[count]]
            else:
               break
            count += 1
            cou += 1

         cou = 0

         lis2 += [lis1]
         lis1 = []

      return lis2

   def stuentry(self):
      applist = []
      curs = self.conn.cursor()
      Tname = self.Tname
      print("welcome, " + Tname + '!!')
      cS = int(input("please enter the strength of your class."))
      print("okay. enter the ", cS, " name(s)")
      for r in range(1, cS + 1):
         x = input(str(r) + ". ")
         applist += [x]


      print("okay. student names collected.\nplease wait for
sometime....")
```

```
      curs.execute('select Tclass from Teacher where Tname =
"{}"'.format(Tname))
      C = curs.fetchone()[0]
      for s in applist:
         curs.execute('insert into Student values(%s, %s, %s, %s);', (s,
C, ' ', ' '))
      selcomm = "insert into " + C + " values(%s, %s, %s, %s)"
      ct = 0
      dlim = 0

      yr = str(self.toddate).split(sep = '-')[0]
      for m in dt1:

         if int(yr) % 4 == 0:
            dlim = 1
         if m == 1:
            yr = str(int(str(self.toddate).split(sep = '-')[0]) + 1)

         D = dt1[m]

         for d in range(1, D[dlim] + 1):
            for i in range(len(applist)):
               curs.execute(selcomm, (applist[i], yr + "-" + str(m) + "-"
+ str(d) , "unentered", ' '))
               self.conn.commit()
               ct += 1
         dlim = 0
      print(ct, " rows have been rendered ready for updation.")
      curs.execute("update Teacher set visit = 1 where Tname = %s",
(Tname, ))
      self.conn.commit()
      print('all okay. you may login again and use the application!')
      from time import sleep
      sleep(10)

connection = sql.connect(user = www[0], password = www[1], host =
'127.0.0.1', port = 3306, database = 'attadmin')
def main(connection, Tname = '', Sname = '', dtxt = 'unentered', toddate =
datetime.date.today()):
   mainobj = Tmain(connection, Tname, Sname, dtxt, toddate)
```

```
    one = mainobj.getclass()
    two = mainobj.getstulist()

    four = mainobj.getTpass()
    five = mainobj.getTlist()
    return [one, two, 0, four, five]

def upd(connection, Tname = '', Sname = '', dtxt = 'unentered', toddate =
datetime.date.today()):
    mainobj = Tmain(connection, Tname, Sname, dtxt, toddate)
    mainobj.Updater()

def nl(connection, toddate, Tname = '', Sname = '', dtxt = 'unentered'):

    mainobj = Tmain(connection, Tname, Sname, dtxt, toddate)
    three = mainobj.getnamelist()
    return three

def classrender(connection, Tname = '', Sname = '', dtxt = 'unentered',
toddate = datetime.date.today()):
    mainobj = Tmain(connection, Tname, Sname, dtxt, toddate)
    mainobj.stuentry()
connection.close()
```

Module: Sinput.py - This module fetches information from students, and allows them to view their marks.

```
import mysql.connector as a
from datetime import datetime
import markwindow, filer
www = filer.p
def fn(name, sclass):
    dbConn=a.connect(host = "127.0.0.1",user = www[0],password =
www[1],database = "attadmin")or die("couldn't connect to DB")
    print('hi there, ', name, '!\n')
    lis = ['yes', 'y', 'yeah', 'yup', 'okay', 'ok', 'alright']
    cursor=dbConn.cursor()
    cursor.execute("Select * from student;")
    message = ''

    try:
        result=cursor.fetchall()

        for i in result:
            print(i[0] + str(bool(i[3])))
            print(i)
            if i[0] == name and bool(i[3]):
                nn = i[0]
                message = 'proceeding'
                break
            else:
                message = "User already exists lol\n"
                continue
    except Exception as e:
        print("User already exists\n", e)
    if message == 'proceeding':
        print('////////student registration//////////')
        stu_email=input("Enter your email: ")
        stu_con=input("Enter Contact Number: ")
        stu_rfid= input("Enter given rfid number: ")
        stu_rfid=" "+stu_rfid
        print(message)
```

```
        cursor.execute("update student set Semail = %s, Srfid = %s where
Sname = %s;",(stu_email,stu_rfid, nn))
        print('inserted')
    ans = input('proceed to view marks? :  ')
    if ans.lower() in lis:
        markwindow.main(name, sclass, 'STUDENT')
        proceed = ''

    dbConn.commit()
    cursor.close()
```

Module: filer.py - the module used for accessing the stored database data.

```
import pickle
#module used to get pre-stored sql password, etc
f = open('D:\Ace\Ace.dat', 'rb')
p = pickle.load(f)

U = p[0]
P = p[1]
f.close()
```

Module: markwindow.py

```
import mysql.connector, filer
from tkinter.ttk import *

global www
www = filer.p
mydb=mysql.connector.connect(host="localhost", user=www[0],
password=www[1], port = 3306, database = "attadmin")
mycursor=mydb.cursor()
def main(sname, sclass, usertype, cursobj = mycursor):

    # Import tkinter as tk
    import tkinter as tk
    from tkinter import messagebox


    # creating a new tkinter window
    master = tk.Tk()

    # assigning a title
    master.title("ACE_MARKSHEET_" + sname)

    # specifying geomtery for window size
    master.geometry("700x250")

    # declaring objects for entering data
    examname = tk.Entry(master)

    #register = tk.Entry(master)
    #rollno = tk.Entry(master)
    #photo=PhotoImage(file='skyandsea.png')
    #Label(root, image=photo).place(relwidth = 1, relheight = 1)
    chem = tk.Entry(master)
    phy = tk.Entry(master)
    math = tk.Entry(master)
    csci = tk.Entry(master)
    eng = tk.Entry(master)
    tk.Label(master, text="Total Marks").grid(row=8, column=3)
```

```
    def show():
        exam = examname.get()
        mycursor.execute('select chemistry, physics, maths,
computer_science, english, total from smarks where examname = %s and sname
= %s', (exam, sname))#the select command for marks-display
        sublis = mycursor.fetchone()
        try:
            chem.delete(0, END)
            phy.delete(0, END)
            math.delete(0, END)
            csci.delete(0, END)
            eng.delete(0, END)

            chem.insert(0, sublis[0])
            phy.insert(0, sublis[1])
            math.insert(0, sublis[2])
            csci.insert(0, sublis[3])
            eng.insert(0, sublis[4])

            tk.Label(master, text="Total Marks: " +
str(sublis[5])).grid(row=8, column=3)
        except Exception as e:
            print(e)
            messagebox.showinfo('error;', 'marks for ' + exam + ' have not
been entered')


    def markupdate():

        e = examname.get()
        ch = chem.get()
        p = phy.get()
        m = math.get()
        cs = csci.get()
        en = eng.get()
        print(ch, p, m, cs, en)
        try:
```

```
            mycursor.execute('insert into smarks values(%s, %s, %s, %s,
%s, %s, %s, %s, %s);', (sname, sclass, e, int(p), int(ch), int(m),
int(cs), int(en), sum([int(p), int(ch), int(m), int(cs), int(en)])))
            mydb.commit()
            messagebox.showinfo('information', 'marks inserted.')
            e = examname.set('')
            ch = chem.delete(0, END)
            p = phy.delete(0, END)
            m = math.delete(0, END)
            cs = csci.delete(0, END)
            en = eng.delete(0, END)
        except Exception as e:
            print('woops!\n', e)


    # label to enter name
    tk.Label(master, text="exam name:").grid(row=0, column=0)

    # label for registration number
    #tk.Label(master, text="Reg.No").grid(row=0, column=3)

    # label for roll Number
    #tk.Label(master, text="Roll.No").grid(row=1, column=0)

    # labels for serial numbers
    tk.Label(master, text="Srl.No").grid(row=2, column=0)
    tk.Label(master, text="1").grid(row=3, column=0)
    tk.Label(master, text="2").grid(row=4, column=0)
    tk.Label(master, text="3").grid(row=5, column=0)
    tk.Label(master, text="4").grid(row=6, column=0)
    tk.Label(master, text="5").grid(row=7, column=0)


    # labels for subject codes
    tk.Label(master, text="Subject").grid(row=2, column=1)
    tk.Label(master, text="CHM").grid(row=3, column=1)
    tk.Label(master, text="PHY").grid(row=4, column=1)
    tk.Label(master, text="MAT").grid(row=5, column=1)
    tk.Label(master, text="CSC").grid(row=6, column=1)
    tk.Label(master, text="ENG").grid(row=7, column=1)
```

```
    # label for grades
    tk.Label(master, text="MARKS").grid(row=2, column=2)
    chem.grid(row=3, column=2)
    phy.grid(row=4, column=2)
    math.grid(row=5, column=2)
    csci.grid(row=6, column=2)
    eng.grid(row=7, column=2)


    # taking entries of name, register, roll number respectively
    examname=tk.Entry(master)
    #register=tk.Entry(master)
    #rollno=tk.Entry(master)

    # organizing them in the grid
    examname.grid(row=0, column=1)
    #register.grid(row=0, column=4)
    #rollno.grid(row=1, column=1)
    if usertype == 'TEACHER':
        button1=tk.Button(master, text="submit", bg="green",
command=markupdate)
        button1.grid(row=8, column=1)
    button2=tk.Button(master, text="show", bg="blue", command=show,font =
'Assassin$.ttf')
    button2.grid(row=0, column=2)


    master.mainloop()
```

Module: Student.py - the module used for retrieving student related information.

```
import mysql.connector as sql
import filer
www = filer.p
connection = sql.connect(user = www[0], password = www[1], host =
'127.0.0.1', port = 3306, database = 'attadmin')
curs = connection.cursor()

def getSlist():
    curs.execute('select Sname from Student;')
    x = curs.fetchall()
    ret = []
    for i in x:
        ret += i
    return ret

def getSclass(Sname):
    curs.execute('select Sclass from Student where Sname = %s;', (Sname,
))
    return curs.fetchone()[0]

def getrfid(Sname):
    curs.execute('select Srfid from Student where Sname = %s;', (Sname, ))
    return curs.fetchone()

def rfid_update(Srfid, Sname, Sclass):
    curs.execute('update Student set Srfid = %s where Sname = %s;',
(Srfid, Sname))
    connection.commit()
    selcomm = 'update ' + Sclass + ' set Srfid = %s where Sname = %s;'

    curs.execute(selcomm, (Srfid, Sname))
    connection.commit()
```

Module: resetter.py - the module used for resetting A.C.E, at the end of the academic year.

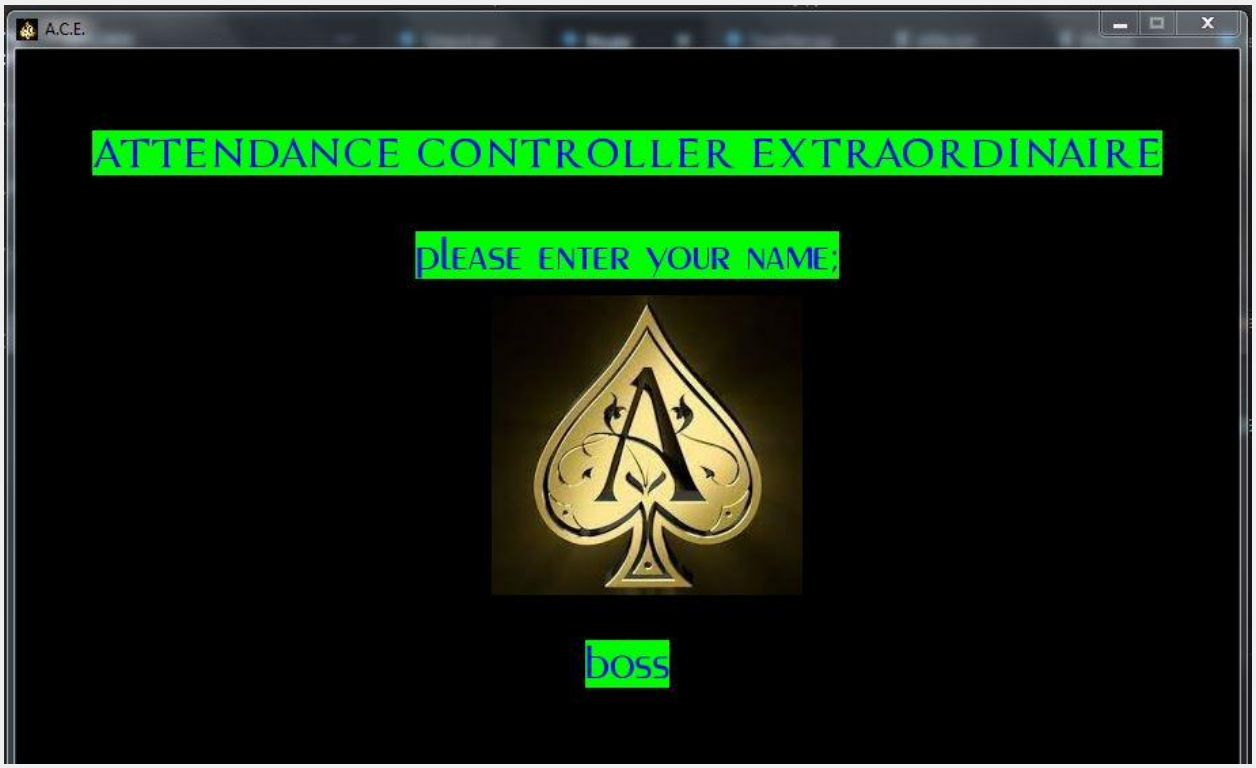
```
import mysql.connector, pickle, filer
from time import sleep
global www
www = filer.p
mydb=mysql.connector.connect(host="localhost", user=www[0],
password=www[1], port = 3306, database = "attadmin")
mycursor=mydb.cursor()

tf = open('tfile.txt', 'w')
tf.write('zero')
print('teacher file overwritten.')
tf.close()
af = open('Afile.txt', 'w')
af.write('zero')
print('admin file overwritten.')
af.close()
print('you may exit after a few seconds;\nThank You')
mycursor.execute('drop database attadmin;')
print('goobye')
mydb.commit()
mycursor.close()
mydb.close()
print('deleted all tables...')
sleep(5)


with open('Ace.dat', 'wb') as ace:
    pickle.dump({}, ace)
    print('ace file overwritten.')
print('resetting complete')
```

OUTPUTS:

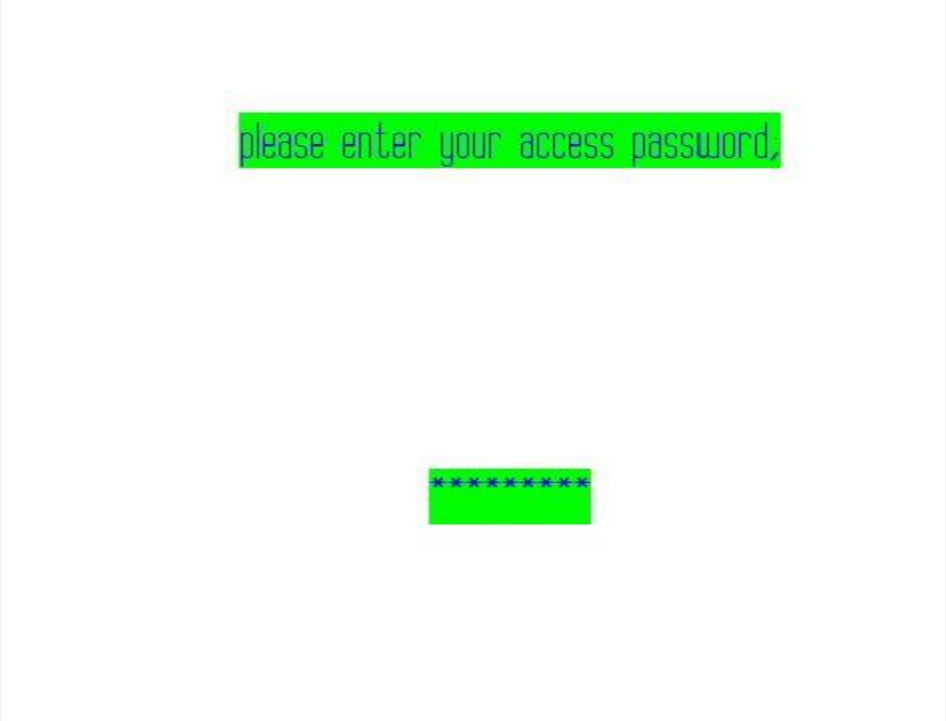
```
hello, user; please enter your name:
>>>boss
please set a password(atleast 7 characters) which you will use to access the application:
passwords entered wont be visible on the screen ;)
>>>
please enter your mysql username:
>>>root
enter your mysqlpassword, please:
>>>
configure a password for teacher-access:
>>>
setting up.....
password inserted.
please enter the number of classes for which you would like to use this application:
>>>3
okay. enter the  3  class(es):
>>>
1. 12a
2. 12b
3. 12c
preparing to create tables for the classes...
done. you are free to go.
all ready. pls login again.
```

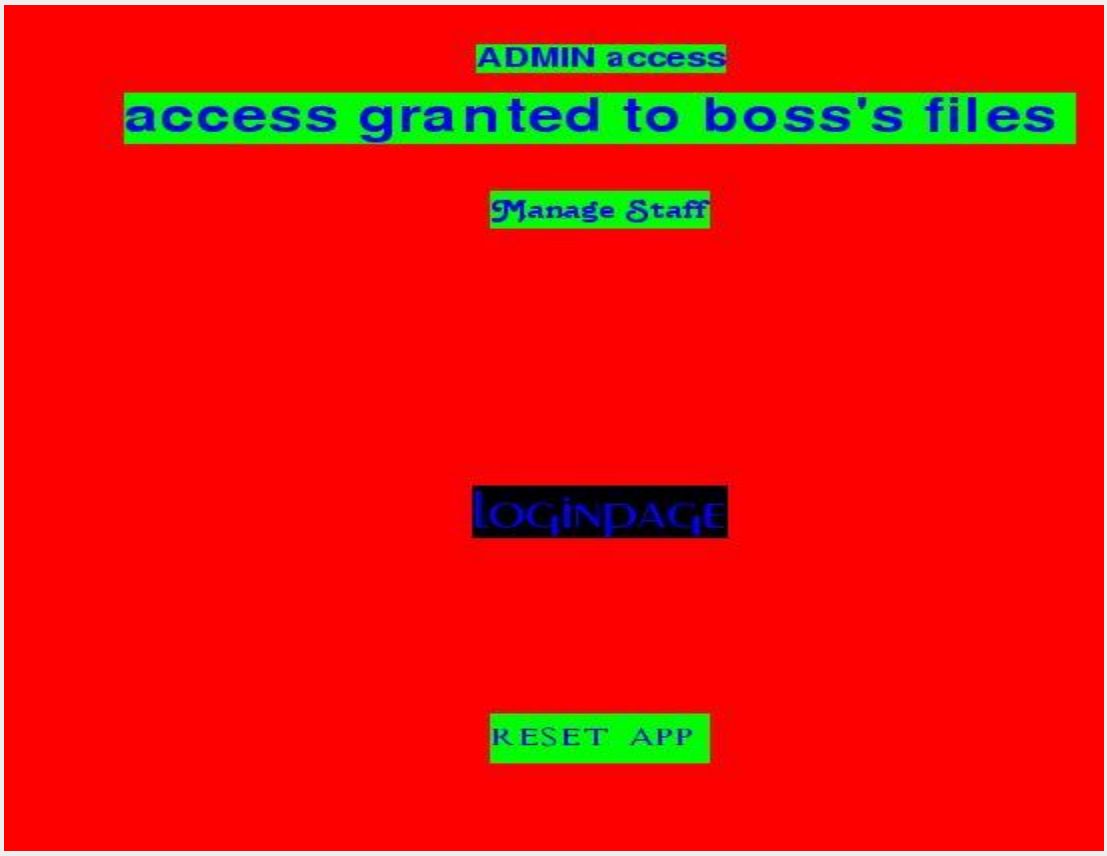
Pic.1.-admin sets up the application.

Pic2.-user logs in.



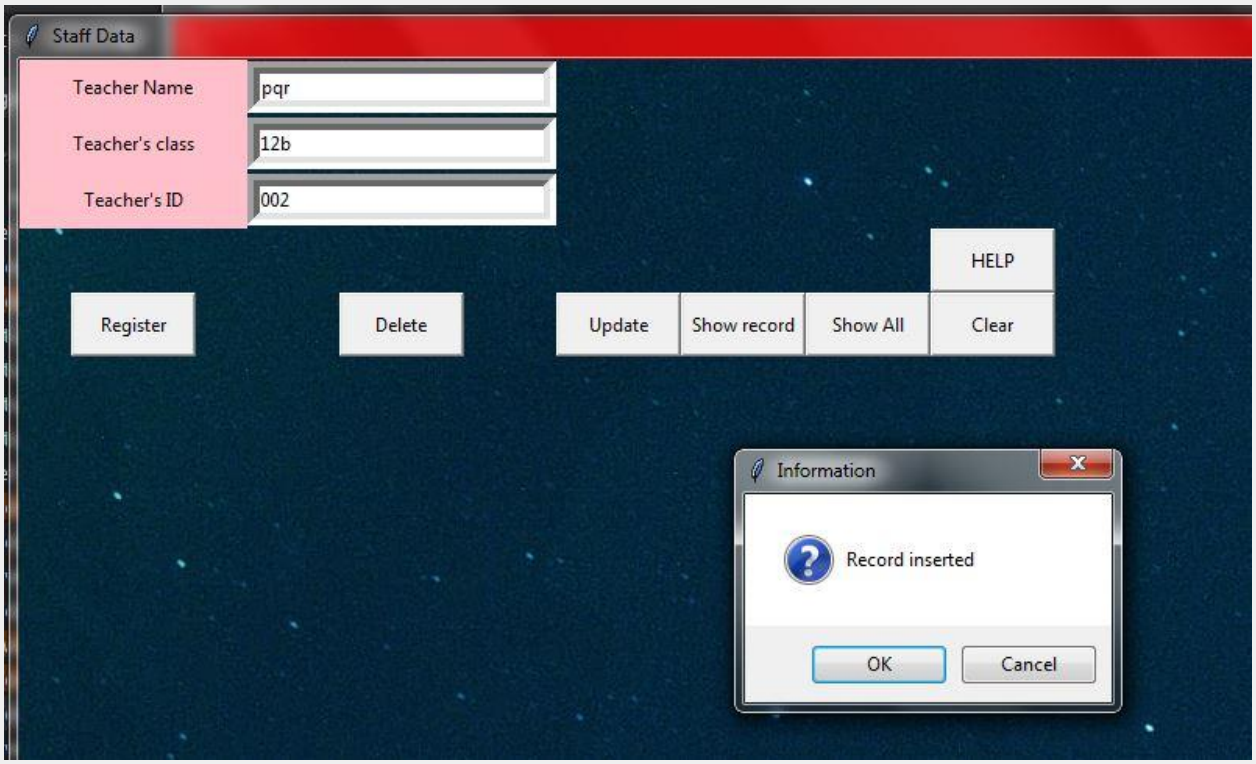
Pic3.- admin enters password.



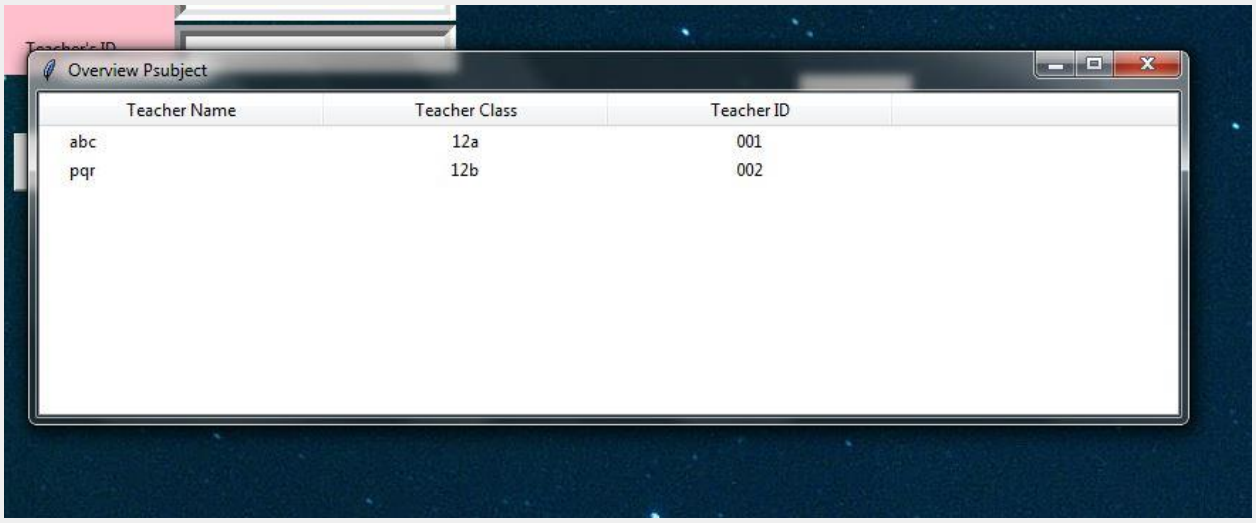


Pic.4.the Admin page

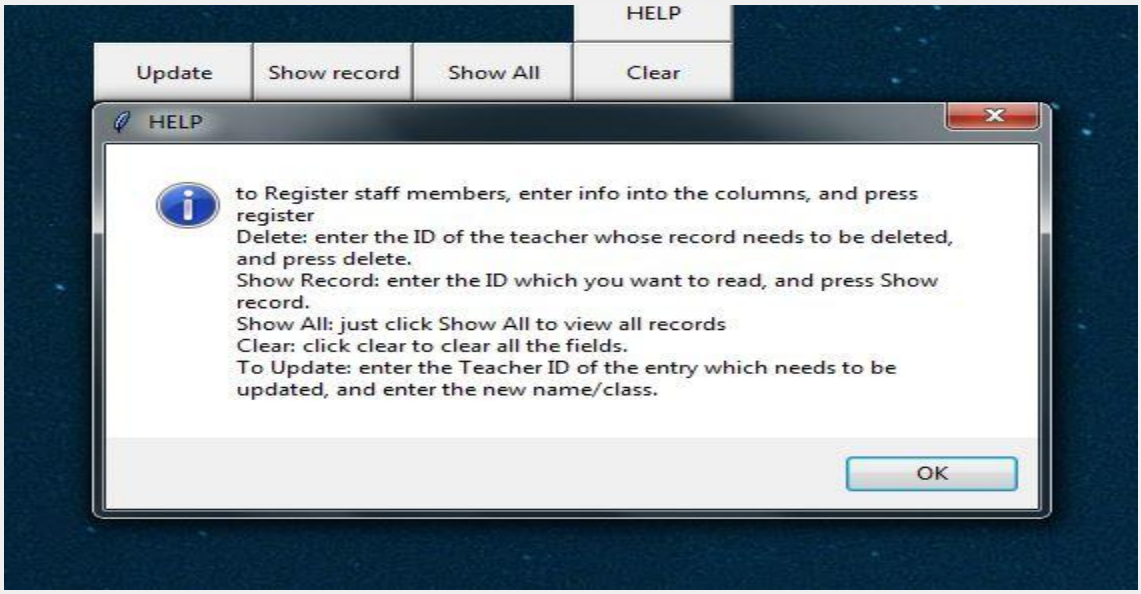
Pic.5-Admin window- inserting teacher records.



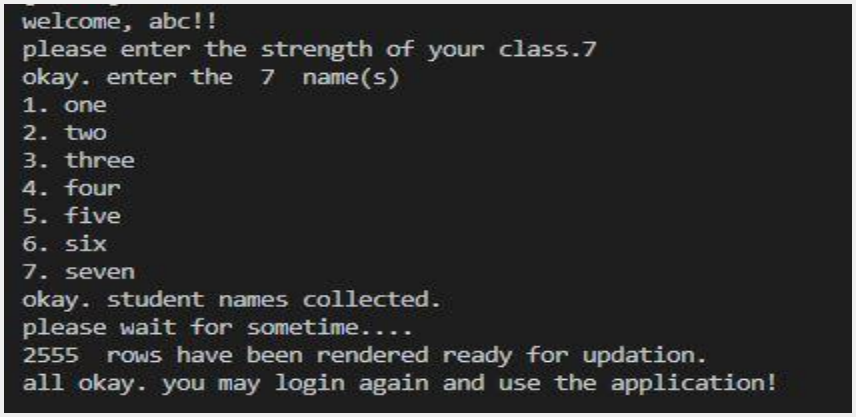
Pic.6- showing list of teachers;



Pic.7- help box.

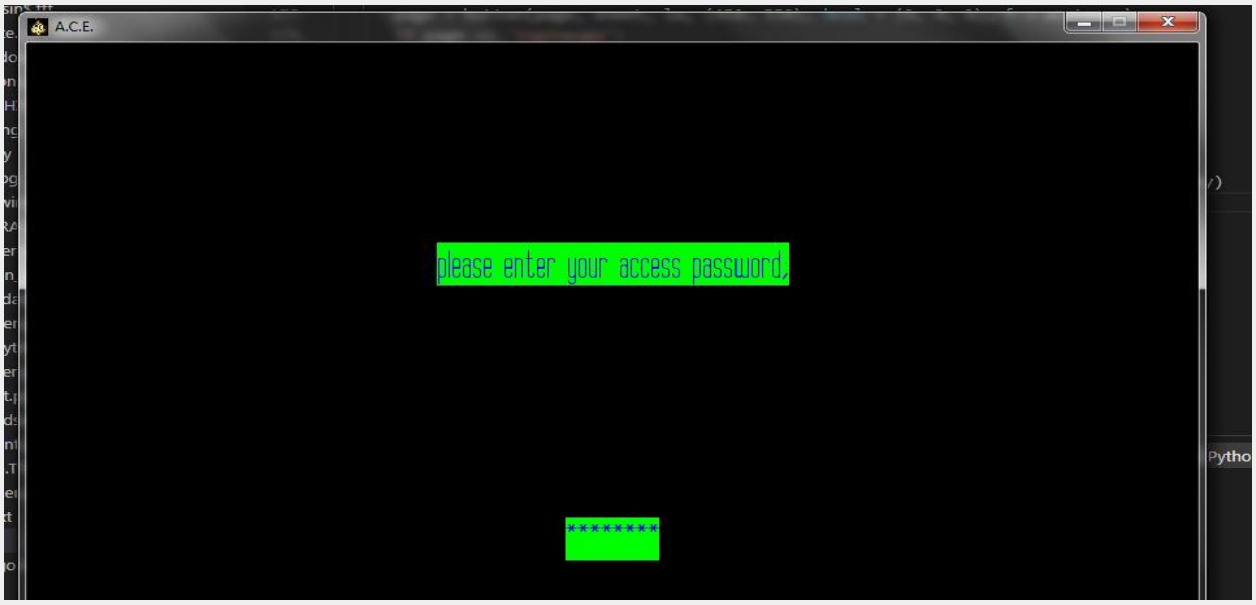


Pic.8- Teacher enters details of class

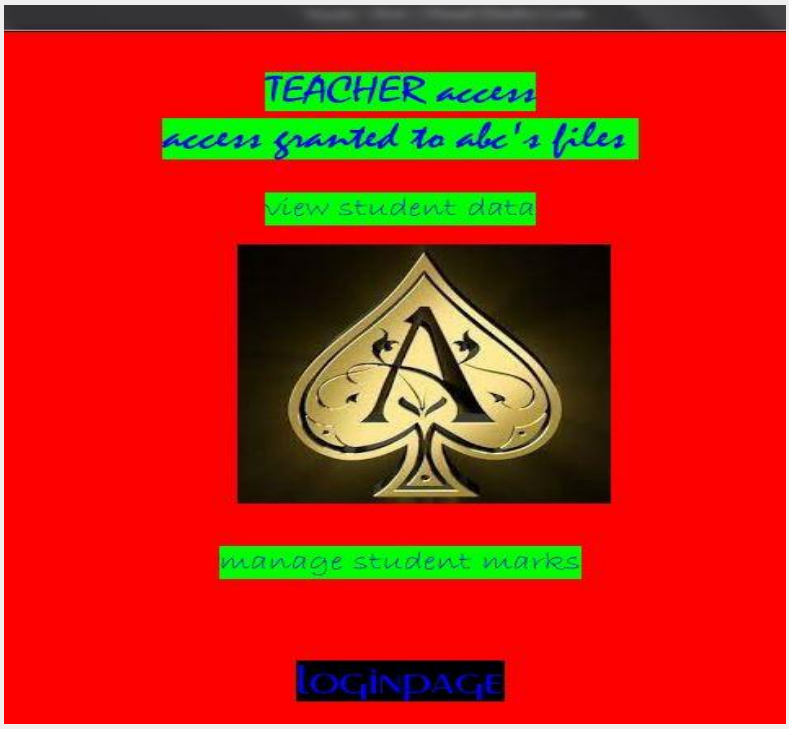


```
welcome, abc!!
please enter the strength of your class.7
okay. enter the  7  name(s)
1. one
2. two
3. three
4. four
5. five
6. six
7. seven
okay. student names collected.
please wait for sometime....
2555  rows have been rendered ready for updation.
all okay. you may login again and use the application!
```

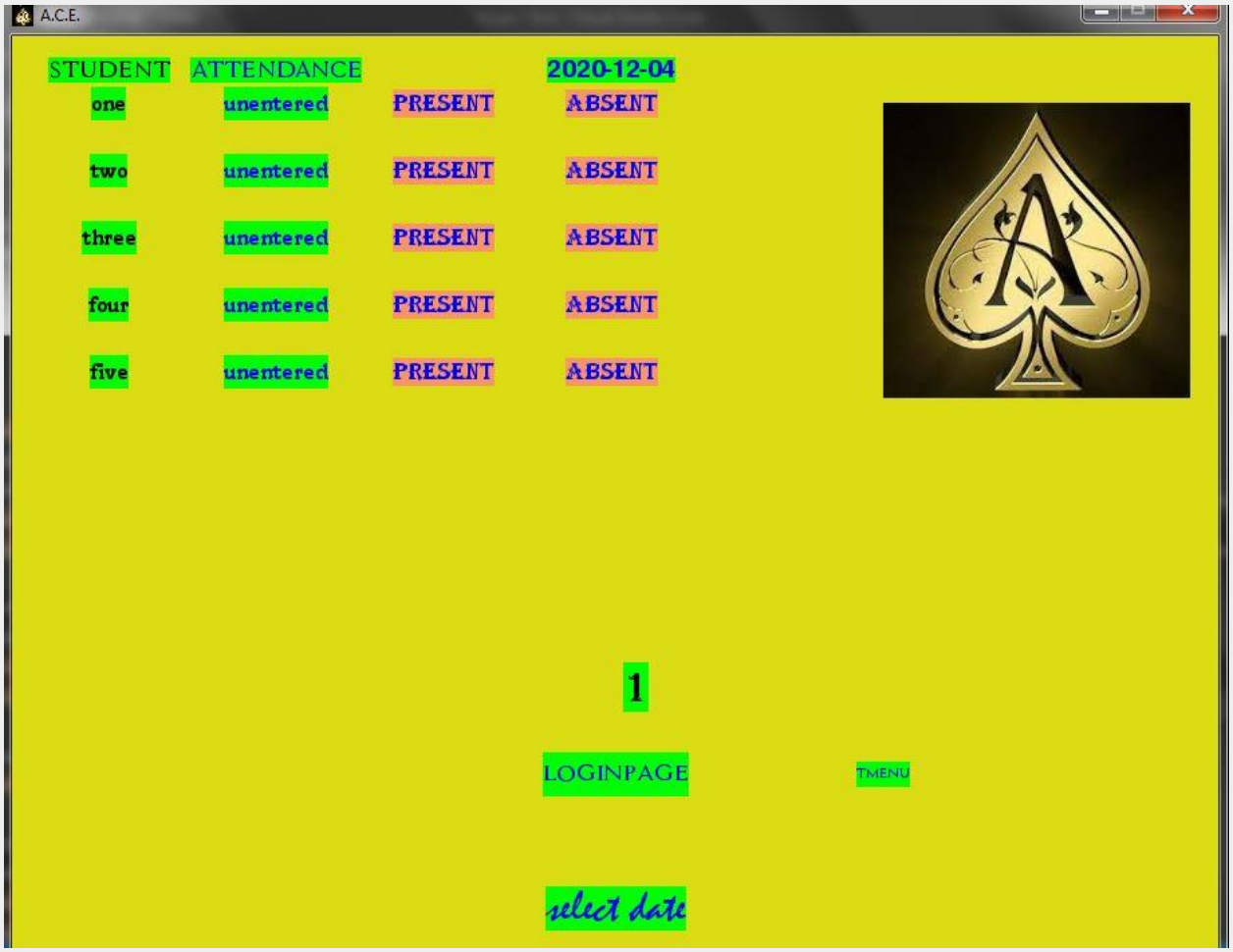
Pic.9- Teacher enters password.



Pic.10- Teacher's window.

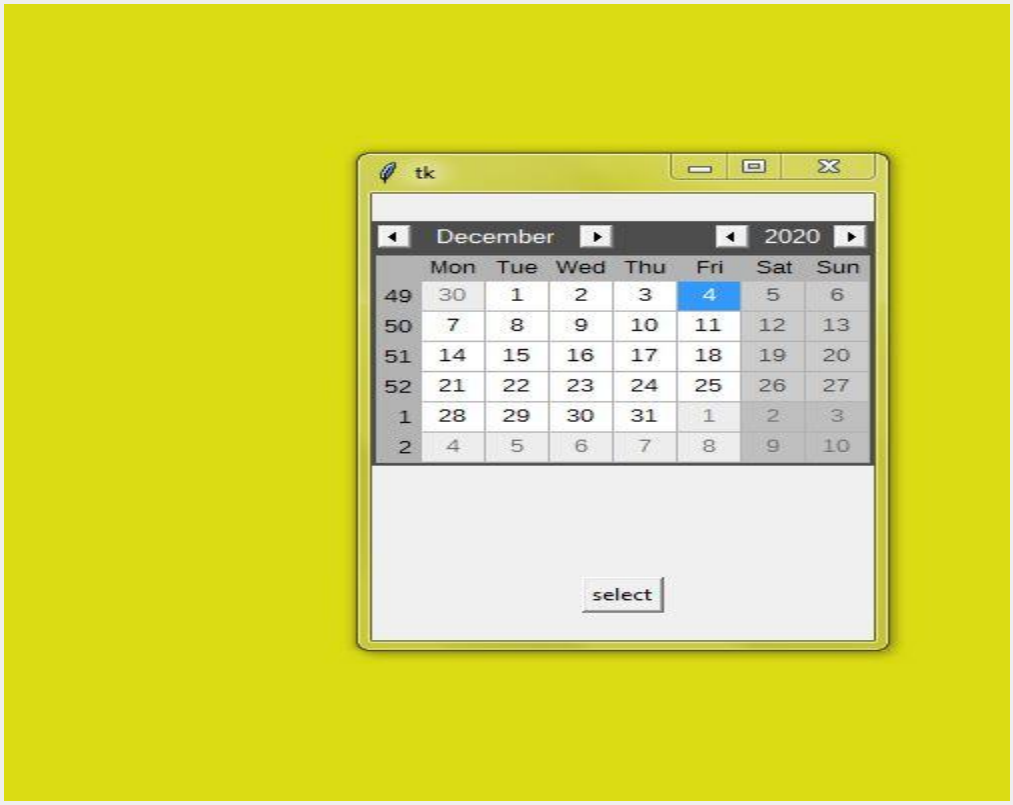


Pic.11- Attendance marking window.(a)

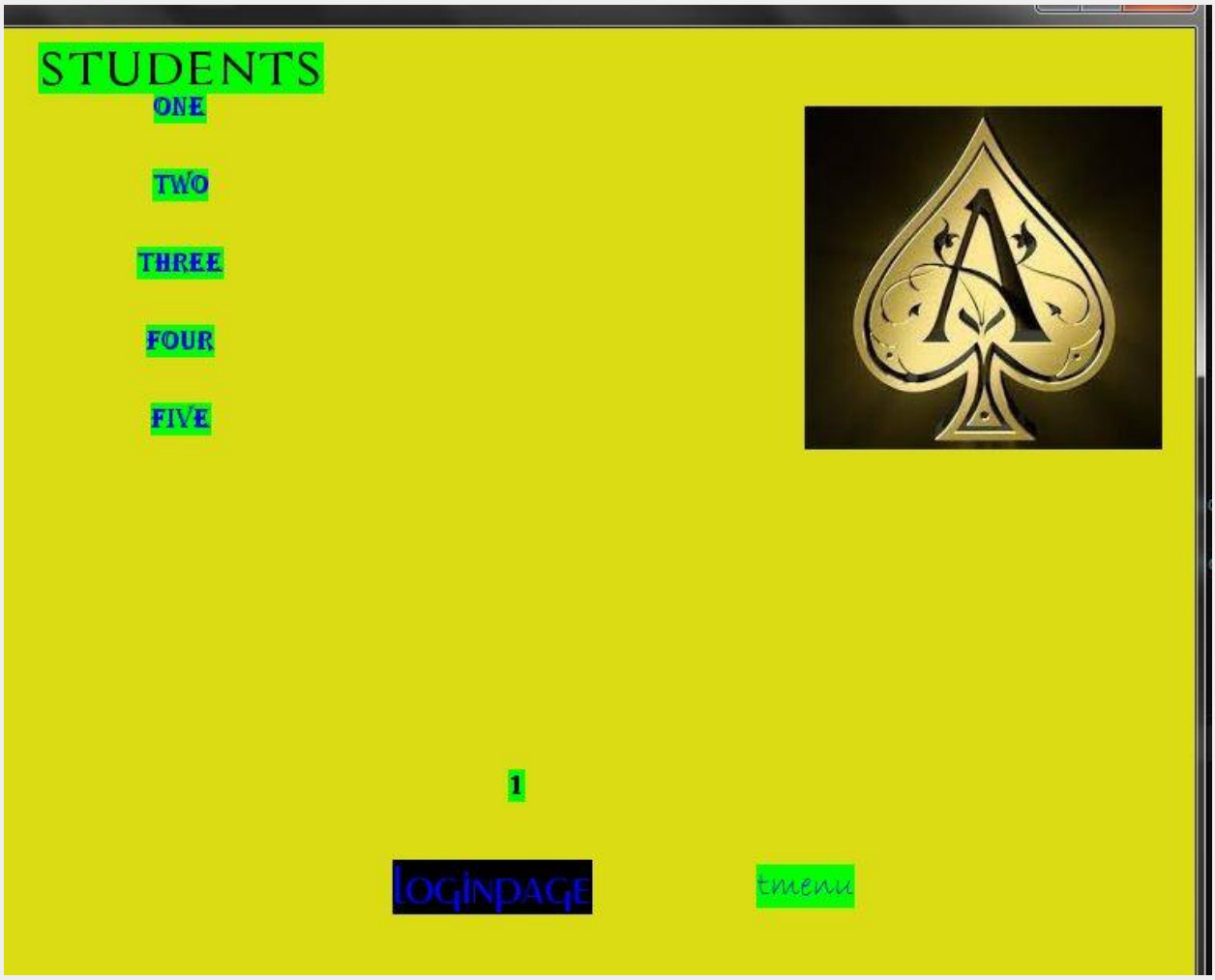


Pic.12-Attendance marking window.(b)

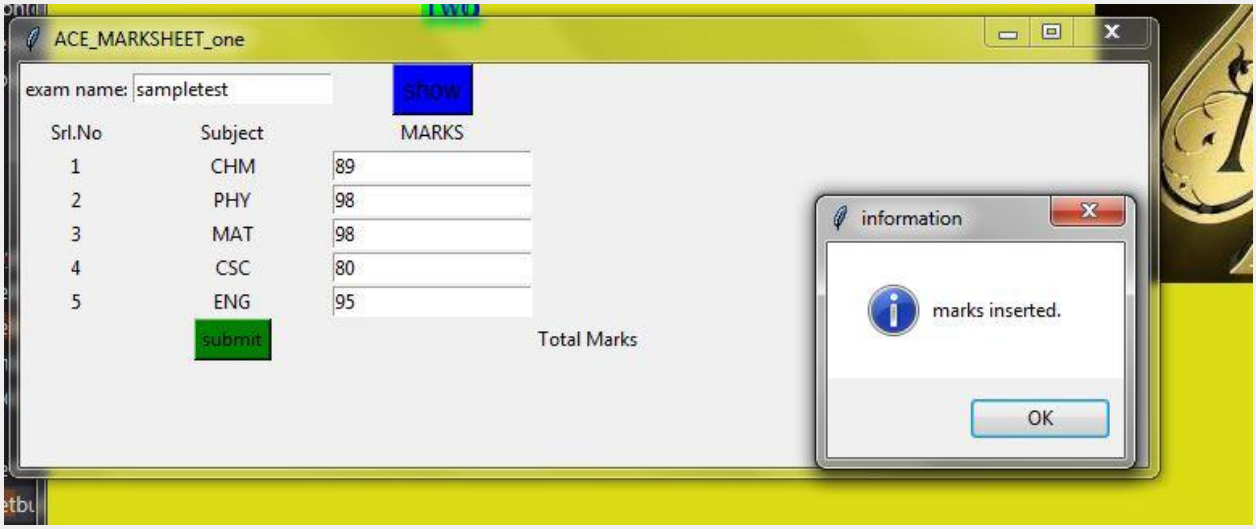
Pic.13-date selection for attendance marking.



Pic.14-menu for marking student marks(a)

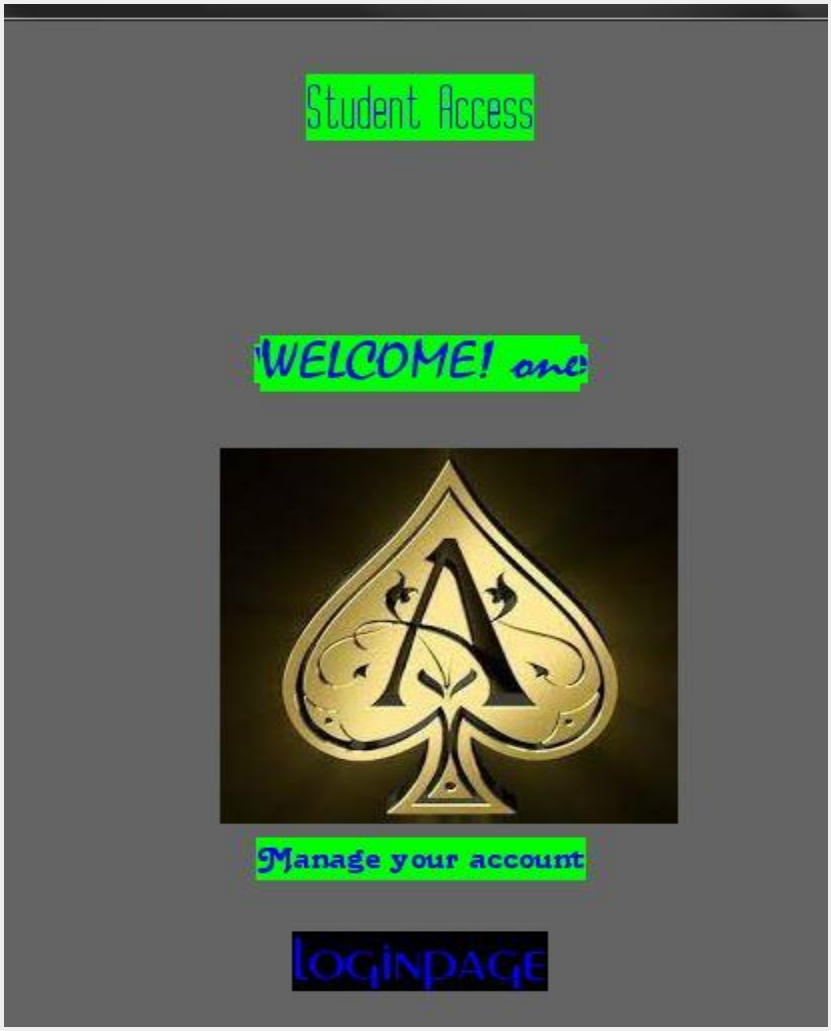


Pic.15-Teacher window for inserting student marks.



Pic.16-menu for marking student marks(b)

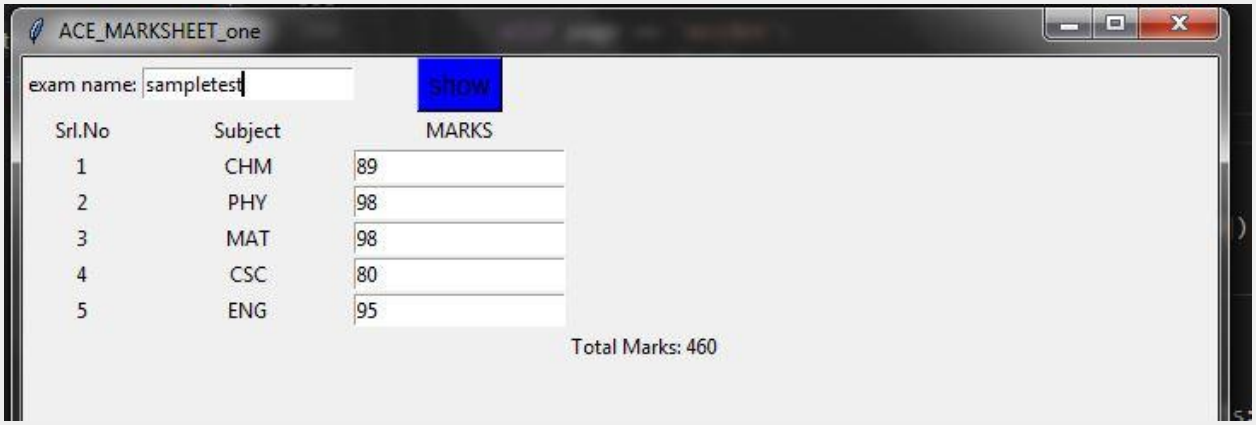
Pic.16-menu for student interface.



Pic.17-student enters his/her details after login.

```
hi there,  one !

Enter your email: email@gmail.com
Enter Contact Number: 1234123498
Enter given rfid number: 2367
proceeding
inserted
proceed to view marks? :  yeah
```

Pic.18-student checks marks.



Pic.19-Admin resets the application.

```
requesting confirmation to reset A.C.E.;
dear admin, please enter your ACE password:
password recieved...
authentication approved.
teacher file overwritten.
admin file overwritten.
you may exit after a few seconds;
Thank You
```

THE CONCLUSION:

Thereby, A.C.E. can be applied to a School and can be used for maintaining records of student attendance and marks.