SIX WEEKS SUMMER TRAINING REPORT

On

“LPU - Learn Advanced Programming using Python ”

Submitted by:

BATHULA MANISAKETH

Registeration No: 12010155

Programme Name : BTech(CSE).

Under the Guidance of E-box

School of Computer Science & Engineering
Lovely Professional University , Phagwara

(May –Aug 2022)

DECLARATION

I here by declare that I have completed my twelve weeks summer training at E- box from May 2022 to July,2022. I have declare that I have worked with full dedication during these six weeks of training and my learning outcomes full fill the requirements of training for the award of degree of B.Tech(CSE) Lovely Professional University, Phagwara.

`Signature of Student
Name: BATHULA MANISAKETH

Reg.No: 12010155

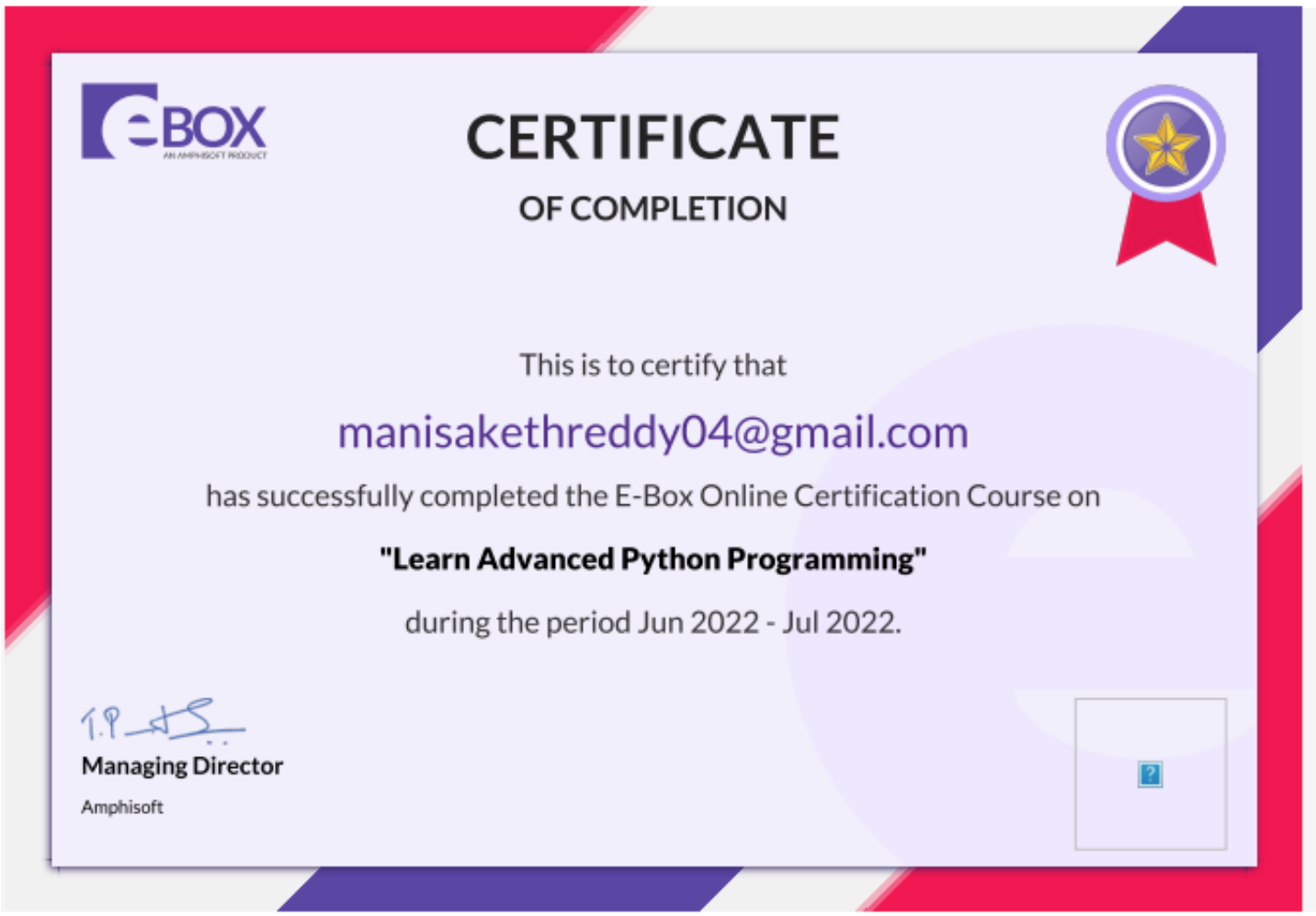
Start Date : May 2022

End Date : Aug 2022

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to my mentor whose contribution in stimulating suggestions and encouragement helped me to coordinate my project especially in writing this report. I express my thanks to my institution Lovely Professional University for giving me an opportunity to learn this interesting topic. I also convey my regards to my faculty assistance all through this training named “Object Oriented Programming”. Once again I would like to thank all my supporters from the core of my heart.

Training certificate from organization



INTRODUCTION TO PYTHON

Python Language Introduction

Python is a widely used general-purpose, high level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.
Python is a programming language that lets you work quickly and integrate systems more efficiently. Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

PYTHON FEATURES

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below − □ It supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

TRAINING CONTENTS

1. **Introduction to Python**

Learn how to install Python, distinguish between important data types and use basic features of the Python interpreter, IDLE.

2. **Using Variables in Python**

   Learn about numeric, string, sequence and dictionary data types and relevant operations while practicing Python syntax.

3. **Basics of Programming in Python**

   Learn how to write programs using conditionals, loops, iterators and generators, functions and modules and packages.

4. **Principles of Object-oriented Programming (OOP)**

   Learn about the important features of Object-oriented Programming while using Classes and Objects, two main aspects of the OOP paradigm.

5. **Connecting to SQLite Database**

   Learn about relational databases while learning how to store and retrieve data from an SQLite database through Python.

6. **Developing a GUI with PyQT**

   Learn how to install PyQt5 toolkit, Qt Designer and create a graphical user interface using common widgets and menu systems.

7. **Application of Python in Various Disciplines**

   Learn about various resources to extend your learning for the Python programming language.

PROBLEM IDENTIFICATION AND CAUSE OF THE PROBLEM

management system is the framework that enables companies to achieve their operational and business objectives through a process of continuous improvement. In its simplest form, a management system implements the Plan, Do, Check, Act/Adjust cycle. Several choices are available for management systems (ISO is commonly applied), whether they are certified by third-party registrars and auditors, self-certified, or used as internal guidance and for potential certification readiness.

Business Benefits of a Well-Documented Management System
The connection between management systems and compliance is vital in avoiding recurring compliance issues and in reducing variation in compliance performance. In fact, reliable and effective regulatory compliance is commonly an outcome of consistent and reliable implementation of a management system.

Beyond that, there are a number of business reasons for implementing a well-documented management system (environmental, safety, quality, food safety, other) and associated support methods and tools:

Establishes a common documented framework to achieve more consistent implementation of compliance policies and processes—addressing the eight core functions of compliance:
Inventories
Permits and authorizations
Plans
Training
Practices in place
Monitoring and inspection
Records
Reporting

OBJECTIVE TO BE ACHIEVED

Create a Student Management System in Python. The should should have all the features displayed in the mock-up screens in the scenario.

Features

-> Add Student

-> Delete Student

-> View all Student

-> Restet

LIBRARIES AND MODULES

Tkinter

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps –

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter Widgets

- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.
- There are currently 15 types of widgets in Tkinter.

these widgets as well as a brief description in the following table –

Sr.No.	Operator & Description
1	Button The Button widget is used to display buttons in your application.
2	Canvas The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
3	Checkbutton

	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
4	Entry The Entry widget is used to display a single-line text field for accepting values from a user.
5	Frame The Frame widget is used as a container widget to organize other widgets.
6	Label The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
7	Listbox The Listbox widget is used to provide a list of options to a user.
8	Menubutton The Menubutton widget is used to display menus in your application.
9	Menu The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
10	Message The Message widget is used to display multiline text fields for accepting values from a user.
11	Radiobutton The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
12	Scale The Scale widget is used to provide a slider widget.
13	Scrollbar The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
14	Text The Text widget is used to display text in multiple lines.
15	Toplevel

	The Toplevel widget is used to provide a separate window container.
16	Spinbox The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
17	PanedWindow A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
18	LabelFrame A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
19	tkMessageBox This module is used to display message boxes in your applications.

Standard attributes

Let us take a look at how some of their common attributes. such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

- **The *pack()* Method** − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- **The *grid()* Method** − This geometry manager organizes widgets in a table-like structure in the parent widget.
- **The *place()* Method** − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Random import

Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc. The random module provides access to functions that support these types of operations. The **random** module is another library of functions that can extend the basic features of python. Other modules we have seen so far are **string**, **math**, **time** and **graphics**. With the exception of the graphics module, all of these modules are built into python. To get access to the **random** module, we add **from random import *** to the top of our program (or type it into the python shell).

Date time

In Python, date and time are not a data type of their own, but a module named datetime can be imported to work with the date as well as time. Python Datetime module comes built into Python, so there is no need to install it externally. Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

Tk message box

The tkMessageBox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Some of these functions are showinfo, showwarning, showerror, askquestion, askokcancel, askyesno, and askretryignore.

Here is the simple syntax to create this widget –

```
tkMessageBox.FunctionName(title, message [, options])
```

Parameters

- **FunctionName** − This is the name of the appropriate message box function.
- **title** − This is the text to be displayed in the title bar of a message box.
- **message** − This is the text to be displayed as a message.
- **options** − options are alternative choices that you may use to tailor a standard message box. Some of the options that you can use are default and parent. The default option is used to specify the default button, such as ABORT, RETRY, or IGNORE in the message box. The parent option is used to specify the window on top of which the message box is to be displayed.

You could use one of the following functions with dialogue box −

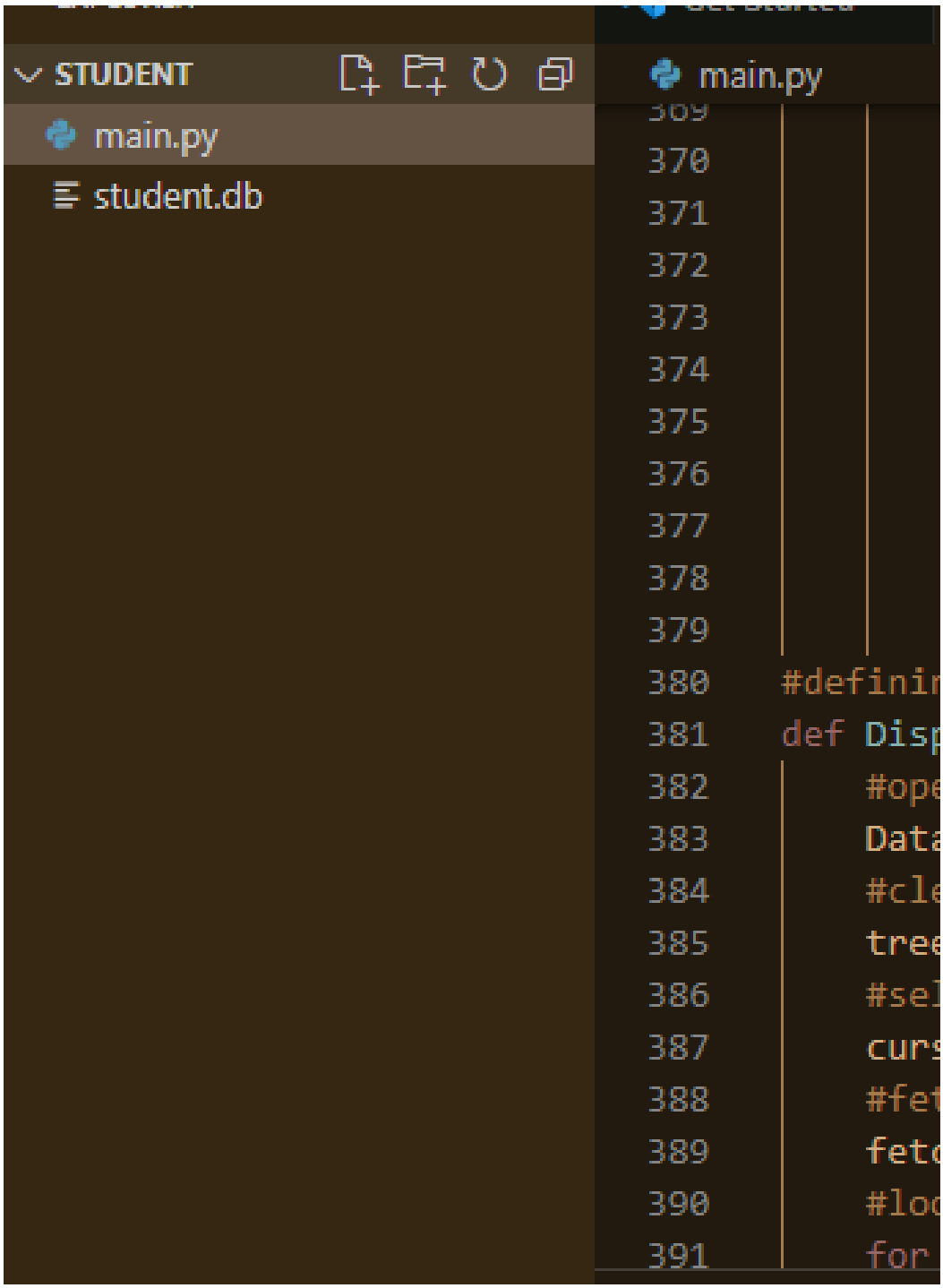
- showinfo()
- showwarning()
- showerror ()
- askquestion()
- askokcancel()
- askyesno ()
- askretrycancel ()

Database sqlite3

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

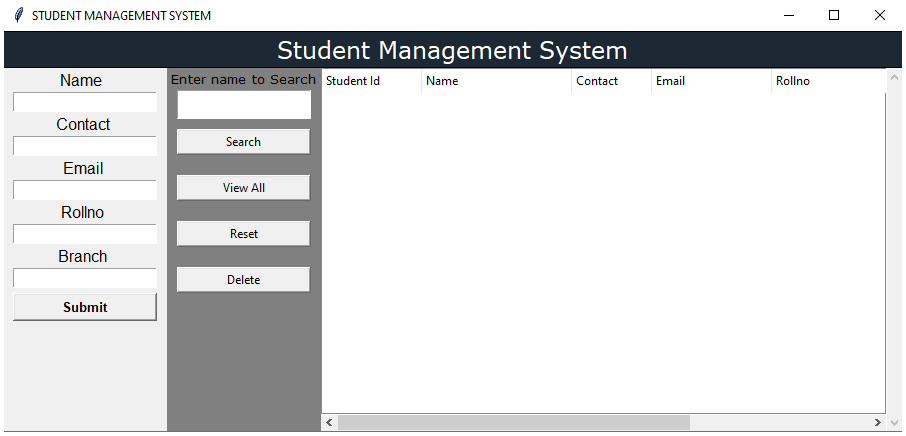
SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format. SQLite database files are a recommended storage format by the US Library of Congress. Think of SQLite not as a replacement for Oracle but as a replacement for fopen()

PROJECT STRUCTURE

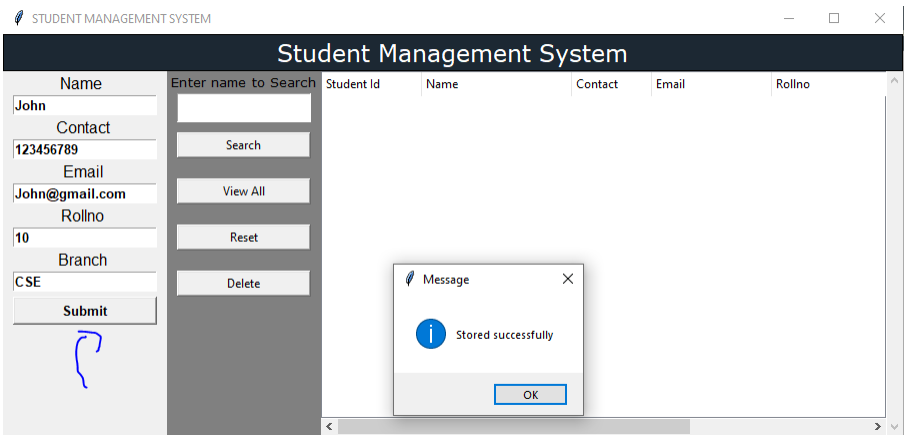


DEMO OF PROJECT

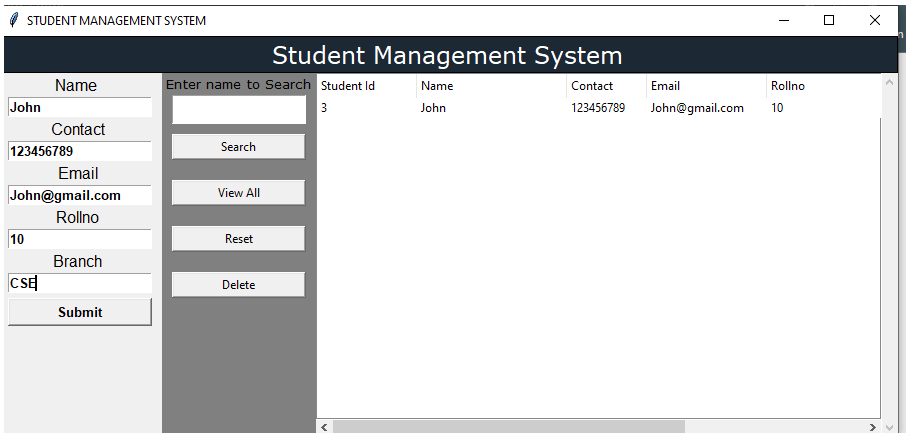
LANDING PAGE



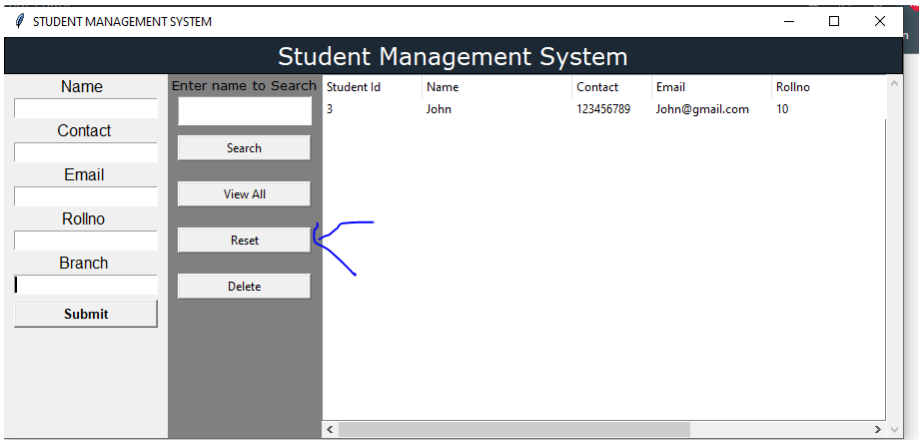
ADD STUDENTS



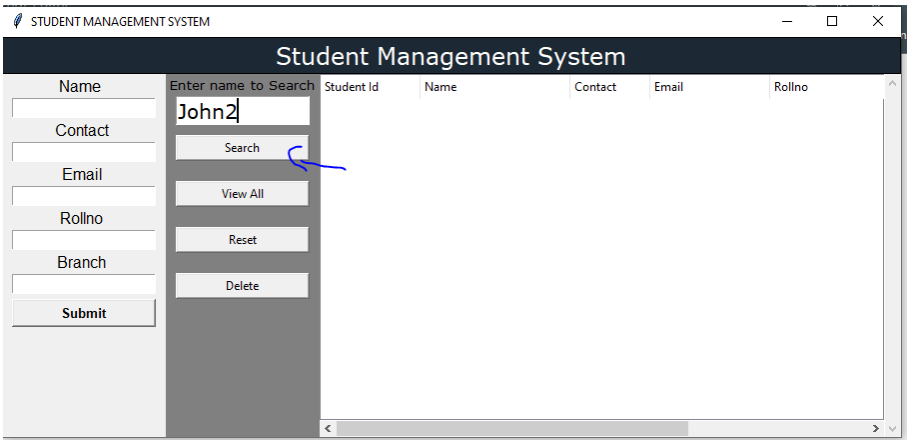
ADDED SUCCESSFULLY



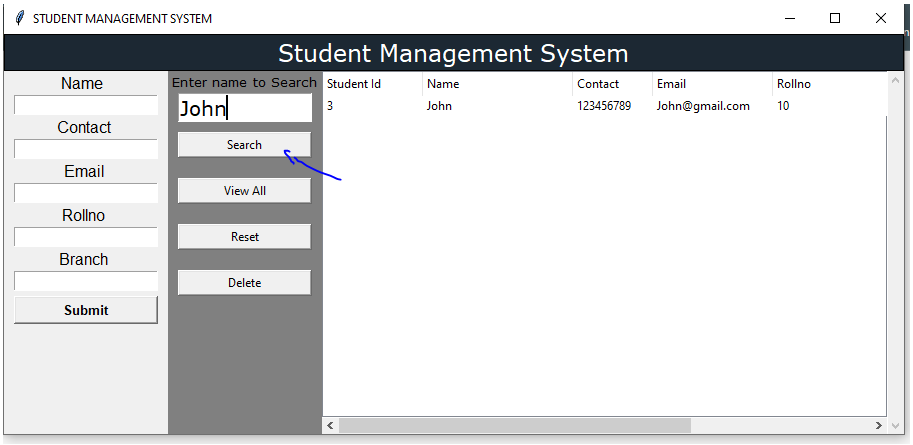
RESET INPUT FIELDS



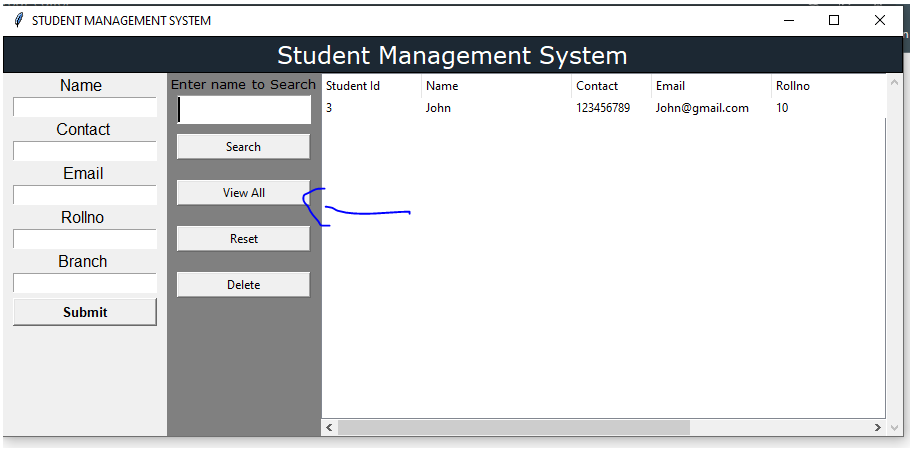
SEARCH STUDENTS



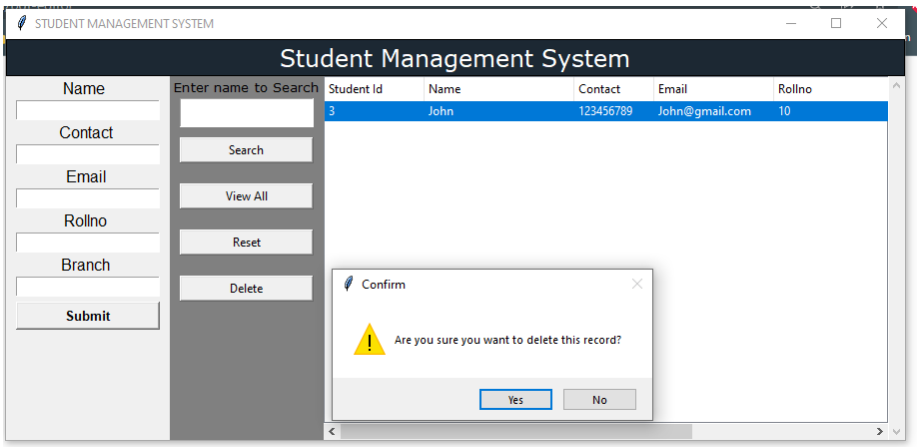
SEARCH SUCCESSFULLY



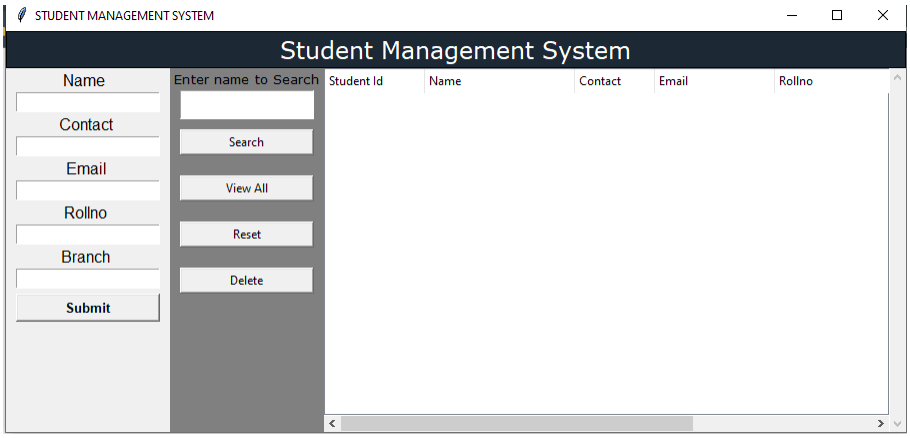
VIEW ALL STUDENTS



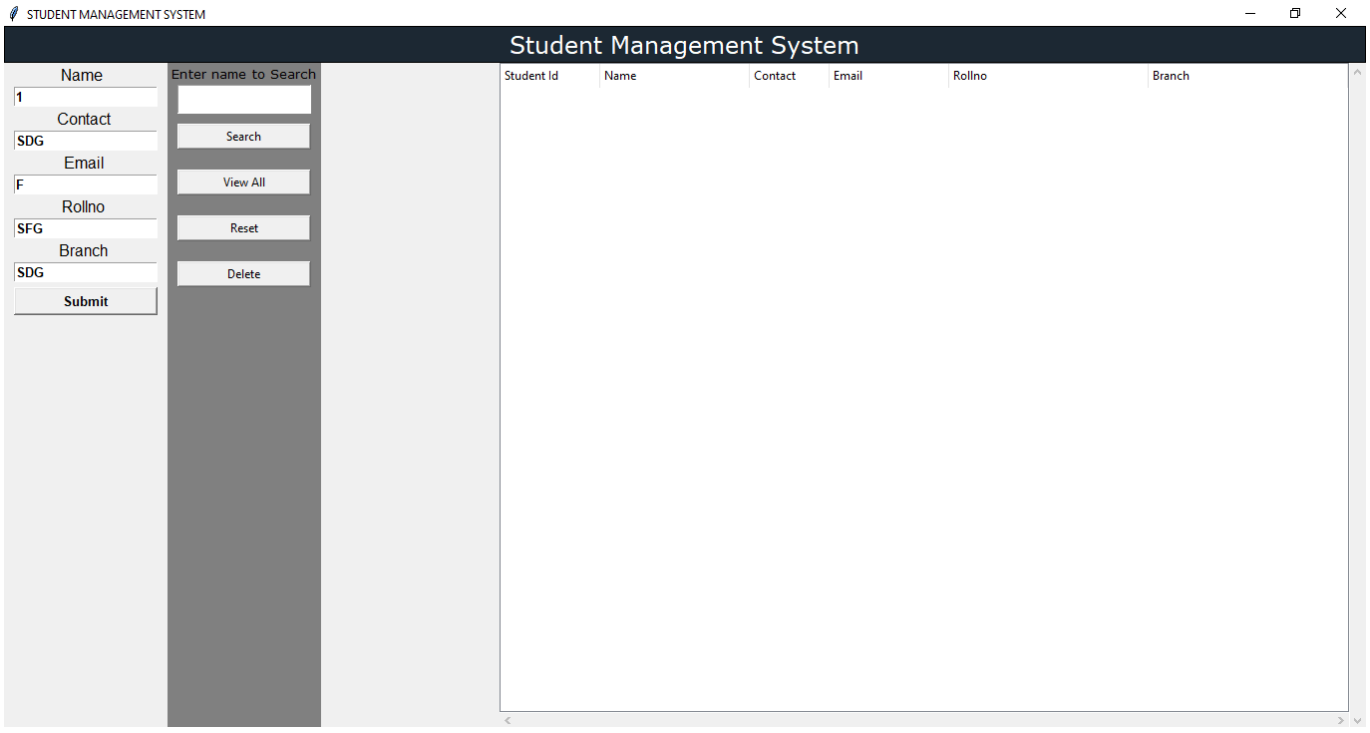
DELETE STUDENTS



DELETE SUCCESSFULLY



MAXIMIZED WINDOW



CODE

```
#import libraries
from tkinter import *
import tkinter.ttk as ttk
import tkinter.messagebox as tkMessageBox
import sqlite3
```

SETUP DATABASE

```
#function to define database
def Database():
    global conn, cursor
    #creating student database
    conn = sqlite3.connect("student.db")
    cursor = conn.cursor()
    #creating STUD_REGISTRATION table
    cursor.execute(
        "CREATE TABLE IF NOT EXISTS STUD_REGISTRATION (STU_ID INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
        STU_NAME TEXT, STU_CONTACT TEXT, STU_EMAIL TEXT, STU_ROLLNO TEXT, STU_BRANCH TEXT)")
```

DISPLAY INPUT FORM

```
    #defining function for creating GUI Layout
def DisplayForm():
    #creating window
    display_screen = Tk()
    #setting width and height for window
    display_screen.geometry("900x400")
    #setting title for window
    display_screen.title("krazyprogrammer.com presents")
    #declaring variables
    global tree
    global SEARCH
    global name,contact,email,rollno,branch
    SEARCH = StringVar()
    name = StringVar()
    contact = StringVar()
    email = StringVar()
    rollno = StringVar()
    branch = StringVar()
    #creating frames for layout
    #topview frame for heading
    TopViewForm = Frame(display_screen, width=600, bd=1, relief=SOLID)
    TopViewForm.pack(side=TOP, fill=X)
    #first left frame for registration from
    LFrom = Frame(display_screen, width="350")
    LFrom.pack(side=LEFT, fill=Y)
    #seconf left frame for search form
    LeftViewForm = Frame(display_screen, width=500,bg="gray")
    LeftViewForm.pack(side=LEFT, fill=Y)
    #mid frame for displaying students record
```

SETUP GUI

```
#mid frame for displaying students record
MidViewForm = Frame(display_screen, width=600)
MidViewForm.pack(side=RIGHT)
#label for heading
lbl_text = Label(TopViewForm, text="Student Management System", font=('verdana', 18), width=600,bg="#1C2833",fg="white")
lbl_text.pack(fill=X)
#creating registration form in first left frame
Label(LFrom, text="Name  ", font=("Arial", 12)).pack(side=TOP)
Entry(LFrom,font=("Arial",10,"bold"),textvariable=name).pack(side=TOP, padx=10, fill=X)
Label(LFrom, text="Contact ", font=("Arial", 12)).pack(side=TOP)
Entry(LFrom, font=("Arial", 10, "bold"),textvariable=contact).pack(side=TOP, padx=10, fill=X)
Label(LFrom, text="Email ", font=("Arial", 12)).pack(side=TOP)
Entry(LFrom, font=("Arial", 10, "bold"),textvariable=email).pack(side=TOP, padx=10, fill=X)
Label(LFrom, text="Rollno ", font=("Arial", 12)).pack(side=TOP)
Entry(LFrom, font=("Arial", 10, "bold"),textvariable=rollno).pack(side=TOP, padx=10, fill=X)
Label(LFrom, text="Branch ", font=("Arial", 12)).pack(side=TOP)
Entry(LFrom, font=("Arial", 10, "bold"),textvariable=branch).pack(side=TOP, padx=10, fill=X)
Button(LFrom,text="Submit",font=("Arial", 10, "bold"),command=register).pack(side=TOP, padx=10,pady=5, fill=X)
```

LABELS FOR SEARCH

```
#creating search label and entry in second frame
lbl_txtsearch = Label(LeftViewForm, text="Enter name to Search", font=('verdana', 10),bg="gray")
lbl_txtsearch.pack()
#creating search entry
search = Entry(LeftViewForm, textvariable=SEARCH, font=('verdana', 15), width=10)
search.pack(side=TOP, padx=10, fill=X)
#creating search button
btn_search = Button(LeftViewForm, text="Search", command=SearchRecord)
btn_search.pack(side=TOP, padx=10, pady=10, fill=X)
#creating view button
btn_view = Button(LeftViewForm, text="View All", command=DisplayData)
btn_view.pack(side=TOP, padx=10, pady=10, fill=X)
#creating reset button
btn_reset = Button(LeftViewForm, text="Reset", command=Reset)
btn_reset.pack(side=TOP, padx=10, pady=10, fill=X)
#creating delete button
btn_delete = Button(LeftViewForm, text="Delete", command=Delete)
btn_delete.pack(side=TOP, padx=10, pady=10, fill=X)
#setting scrollbar
scrollbarx = Scrollbar(MidViewForm, orient=HORIZONTAL)
scrollbary = Scrollbar(MidViewForm, orient=VERTICAL)
tree = ttk.Treeview(MidViewForm,columns=("Student Id", "Name", "Contact", "Email","Rollno","Branch"),
                    selectmode="extended", height=100, yscrollcommand=scrollbary.set, xscrollcommand=scrollbarx.set)
```

SETUP GEOMETRY

```
scrollbary.config(command=tree.yview)
scrollbary.pack(side=RIGHT, fill=Y)
scrollbarx.config(command=tree.xview)
scrollbarx.pack(side=BOTTOM, fill=X)
#setting headings for the columns
tree.heading('Student Id', text="Student Id", anchor=W)
tree.heading('Name', text="Name", anchor=W)
tree.heading('Contact', text="Contact", anchor=W)
tree.heading('Email', text="Email", anchor=W)
tree.heading('Rollno', text="Rollno", anchor=W)
tree.heading('Branch', text="Branch", anchor=W)
#setting width of the columns
tree.column('#0', stretch=NO, minwidth=0, width=0)
tree.column('#1', stretch=NO, minwidth=0, width=100)
tree.column('#2', stretch=NO, minwidth=0, width=150)
tree.column('#3', stretch=NO, minwidth=0, width=80)
tree.column('#4', stretch=NO, minwidth=0, width=120)
tree.pack()
DisplayData()
```

ADD STUDENT TO DATABASE

```
    #function to insert data into database
def register():
    Database()
    #getting form data
    name1=name.get()
    con1=contact.get()
    email1=email.get()
    rol1=rollno.get()
    branch1=branch.get()
    #applying empty validation
    if name1=='' or con1==''or email1=='' or rol1==''or branch1=='':
        tkMessageBox.showinfo("Warning","fill the empty field!!!")
    else:
        #execute query
        conn.execute('INSERT INTO STUD_REGISTRATION (STU_NAME,STU_CONTACT,STU_EMAIL,STU_ROLLNO,STU_BRANCH) \
            VALUES (?,?,?,?,?)',(name1,con1,email1,rol1,branch1));
        conn.commit()
        tkMessageBox.showinfo("Message","Stored successfully")
        #refresh table data
        DisplayData()
        conn.close()
```

Logic for Rest Field Values

```
    def Reset():
        #clear current data from table
        tree.delete(*tree.get_children())
#refresh table data
DisplayData()
#clear search text
SEARCH.set("")
name.set("")
contact.set("")
email.set("")
rollno.set("")
branch.set("")
```

Logic for deleting data from datbase

```
def Delete():

#open database
    Database()
if not tree.selection():
    tkMessageBox.showwarning("Warning","Select data to delete")
else:
    result = tkMessageBox.askquestion('Confirm', 'Are you sure you want to delete this record?',
                                      icon="warning")
    if result == 'yes':
        curItem = tree.focus()
        contents = (tree.item(curItem))
        selecteditem = contents['values']
        tree.delete(curItem)
        cursor=conn.execute("DELETE FROM STUD_REGISTRATION WHERE STU_ID = %d" % selecteditem[0])
        conn.commit()
        cursor.close()
        conn.close()
```

Search student by their name

```
        #function to search data
def SearchRecord():
    #open database
    Database()
    #checking search text is empty or not
    if SEARCH.get() != "":
        #clearing current display data
        tree.delete(*tree.get_children())
        #select query with where clause
        cursor=conn.execute("SELECT * FROM STUD_REGISTRATION WHERE STU_NAME LIKE ?", ('%' + str(SEARCH.get()) + '%',))
        #fetch all matching records
        fetch = cursor.fetchall()
        #loop for displaying all records into GUI
        for data in fetch:
            tree.insert('', 'end', values=(data))
        cursor.close()
        conn.close()
```

Display students

```
    #defining function to access data from SQLite database
def DisplayData():
    #open database
    Database()
    #clear current data
    tree.delete(*tree.get_children())
    #select query
    cursor=conn.execute("SELECT * FROM STUD_REGISTRATION")
    #fetch all data from database
    fetch = cursor.fetchall()
    #loop for displaying all data in GUI
    for data in fetch:
        tree.insert('', 'end', values=(data))
    cursor.close()
    conn.close()
```

Main Function

```
#calling function
DisplayForm()
if __name__=='__main__':
#Running Application
 mainloop()
```

CONCLUSION

1. Improves The General Performance Of Students:

For students to come out with good grades, the focus is highly needed. With this school administration software, students are able to use their precious time for relevant things which is studying rather than keep track of their records to make sure things are intact. Moreover, fear of losing important records to manual management is completely off the line with the help of this software.

2. It helps Simply And Streamlines All Task:

As a teacher, keeping track of all the activities done by each student is never easy and inefficient.
But with this online school management software, teachers are able to keep track of each student work and what is yet to be done. In most software, there is a dashboard and a single screen display to make things even easier and more efficient.

3. Improved Communication:

It's impossible for every student in a class to get each lesson passed across while teaching. This might be a problem on the path of the students being timid or reluctant at that point in time or insufficient time to ask questions. With the help of this software, that has been sorted out. Most online school management software now has an i nbuilt discussion panel where students could easily communicate with their teachers and ask relevant questions.

4. Can Be Accessed By All Parties Involved:

Gone are the days when parents know little to nothing about what their child or ward does at school.
This open source school management system has made it possible for parents to have access to their children's school activities, assignments, attendance, and other relevant information just by using the software.

5. Efficient Management And Organization of Timetables:

The management of a school timetable is the work of the admin department of a school. Despite being dedicated to a department, it never correlates. With this software, the reverse is the case. This school administration software is able to organize school timetables in such a way that they won't be a clash of lectures. In addition to that, students, teachers, and parents have access to it.

6. Helps To Keep Track Of All Students:

School activity goes beyond the wall of the classrooms. There are other activities like sports, interaction, and other extracurricular activities and all these needs proper documentation. This management software has features that see to that and make sure the record of each student is intact.

7. Reduction Of Human Labour, Papers and Workload:

The cost incurred from employing staff to manage the activities of a school is an additional cost that shouldn't be if you decide to make use of this new technology. With this student management system, your number of staffs, excessive buying of writing materials will be minimal and functionality will improve.

REFRENCE

- https://docs.python.org/3/
- https://www.sqlite.org/index.html
- https://www.sqlitetutorial.net/sqlite-python/