SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

COMPUTER SCIENCE AND ENGINEERING

Python Programming

15IT322E

BOOKSTORE MANAGEMENT SYSTEM

STUDENT ID :
RA1611003010555 - AKSHAY VENKATESAN
RA1611003010875 - DEEPANSHU SEHGAL

Contents

1 INTRODUCTION

This program contains both front-end and back-end code for a bookstore app with a simple graphical user interface (GUI) built with Tkinter library. Tkinter serves well in terms of getting familiar with how to connect to databases(SQLite in this case), manipulate data, and reflect the changes on the front end with Python.

2 CODE

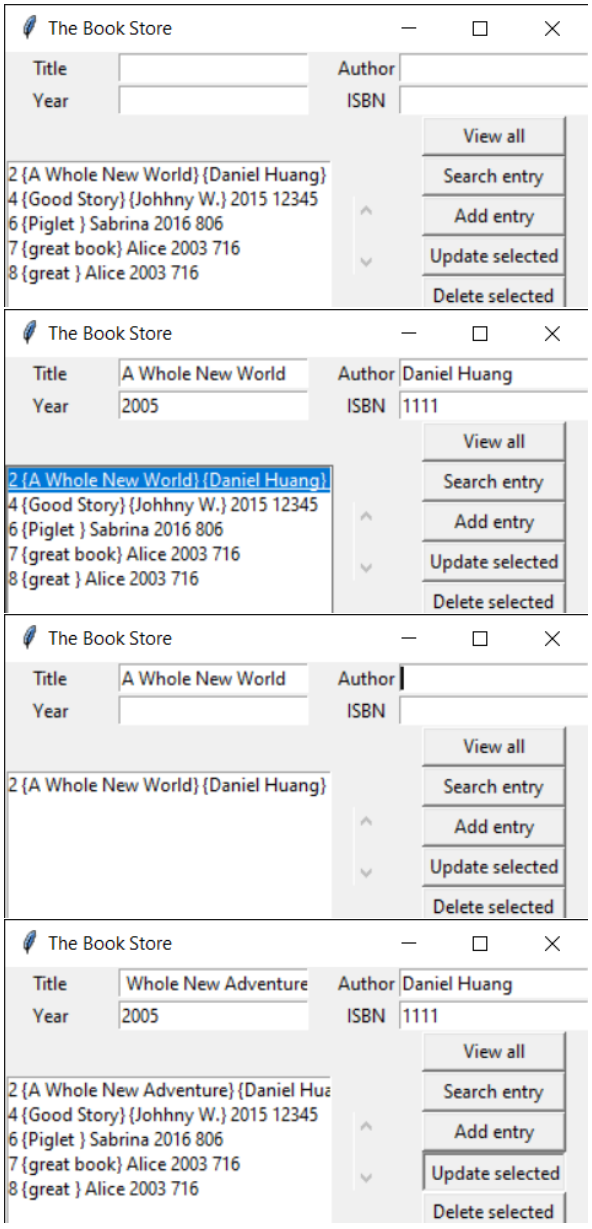
```
from tkinter import *
from backend import  Database

database = Database("books.db")

class Window(object):
    def __init__(self,window):
        self.window = window
        self.window.wm_title("The Book Store")

        l1 = Label(window, text="Title")
        l1.grid(row=0, column=0)

        l2 = Label(window, text="Author")
        l2.grid(row=0, column=2)

        l3 = Label(window, text="Year")
        l3.grid(row=1, column=0)

        l4 = Label(window, text="ISBN")
        l4.grid(row=1, column=2)

        self.title_text = StringVar()
        self.e1 = Entry(window, textvariable=self.title_text)
        self.e1.grid(row=0, column=1)

        self.author_text = StringVar()
        self.e2 = Entry(window, textvariable=self.author_text)
        self.e2.grid(row=0, column=3)
```

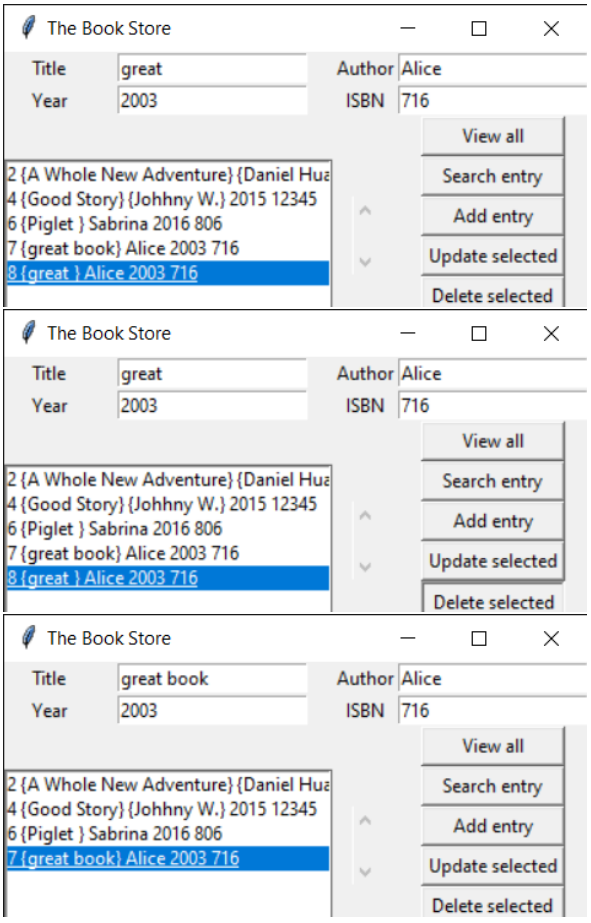
```
        self.year_text = StringVar()
        self.e3 = Entry(window, textvariable=self.year_text)
        self.e3.grid(row=1, column=1)

        self.ISBN_text = StringVar()
        self.e4= Entry(window, textvariable=self.ISBN_text)
        self.e4.grid(row=1, column=3)

        self.list1 = Listbox(window, height=6, width=35)
        self.list1.grid(row=2, column=0, rowspan=6, columnspan=2)

        self.list1.bind('<<ListboxSelect>>', self.get_selected_row)

        # now we need to attach a scrollbar to the listbox, and the other direction
        sb1 = Scrollbar(window)
        sb1.grid(row=2, column=2, rowspan=6)
        self.list1.config(yscrollcommand=sb1.set)
        sb1.config(command=self.list1.yview)

        b1 = Button(window, text="View all", width=12, command=self.view_command)
        b1.grid(row=2, column=3)

        b2 = Button(window, text="Search entry", width=12, command=self.search_comm
        b2.grid(row=3, column=3)

        b3 = Button(window, text="Add entry", width=12, command=self.add_command)
        b3.grid(row=4, column=3)

        b4 = Button(window, text="Update selected", width=12, command=self.update_c
        b4.grid(row=5, column=3)

        b5 = Button(window, text="Delete selected", width=12, command=self.delete_c
        b5.grid(row=6, column=3)

        b6 = Button(window, text="Close", width=12, command=window.destroy)
        b6.grid(row=7, column=3)


    def get_selected_row(self,event):   #the "event" parameter is needed b/c we've
        try:
```

```
            index = self.list1.curselection()[0]
            self.selected_tuple = self.list1.get(index)
            self.e1.delete(0,END)
            self.e1.insert(END,self.selected_tuple[1])
            self.e2.delete(0, END)
            self.e2.insert(END,self.selected_tuple[2])
            self.e3.delete(0, END)
            self.e3.insert(END,self.selected_tuple[3])
            self.e4.delete(0, END)
            self.e4.insert(END,self.selected_tuple[4])
        except IndexError:
            pass                #in the case where the listbox is empty, the code w

    def view_command(self):
        self.list1.delete(0, END)  # make sure we've cleared all entries in the lis
        for row in database.view():
            self.list1.insert(END, row)

    def search_command(self):
        self.list1.delete(0, END)
        for row in database.search(self.title_text.get(), self.author_text.get(), s
            self.list1.insert(END, row)

    def add_command(self):
        database.insert(self.title_text.get(), self.author_text.get(), self.year_te
        self.list1.delete(0, END)
        self.list1.insert(END, (self.title_text.get(), self.author_text.get(), self

    def delete_command(self):
        database.delete(self.selected_tuple[0])
        self.view_command()

    def update_command(self):
        #be careful for the next line ---> we are updating using the texts in the e
        database.update(self.selected_tuple[0],self.title_text.get(), self.author_t
        self.view_command()

#code for the GUI (front end)
window = Tk()
Window(window)
```

```
window.mainloop()
```

3 SCREENSHOTS




The Book Store
Title
great
Author
Alice
Year
2003
ISBN
716
View all
Search entry
Add entry
Update selected
Delete selected
2 {A Whole New Adventure} {Daniel Hua
4 {Good Story} {Johhny W.} 2015 12345
6 {Piglet } Sabrina 2016 806
7 {great book} Alice 2003 716
8 {great } Alice 2003 716
The Book Store
Title
great
Author
Alice
Year
2003
ISBN
716
View all
Search entry
Add entry
Update selected
Delete selected
2 {A Whole New Adventure} {Daniel Hua
4 {Good Story} {Johhny W.} 2015 12345
6 {Piglet } Sabrina 2016 806
7 {great book} Alice 2003 716
8 {great } Alice 2003 716
The Book Store
Title
great book
Author
Alice
Year
2003
ISBN
716
View all
Search entry
Add entry
Update selected
Delete selected
2 {A Whole New Adventure} {Daniel Hua
4 {Good Story} {Johhny W.} 2015 12345
6 {Piglet } Sabrina 2016 806
7 {great book} Alice 2003 716