**A Project Report**

**On**

**BOOK STORE DESKTOP APPLICATION**

Submitted in partial fulfillment of the requirement for the award of the degree of

**BACHELOR’S OF COMPUTER SCIENCE AND ENGINEERING**

In

**COMPUTER SCIENCE DEPARTMENT**

**SESSION: 2019**



### UNIVERSITY COLLEGE OF ENGINEERING (UCOE)

**PUNJABI UNIVERSITY, PATIALA**

**SUBMITTED BY:**

**Name: GURPREET SINGH**

**Roll No: 11701137**

**Group: 3 CE – 5**

**ACKNOWLEDGMENT**

The satisfaction that accompanies the successful completion of this project would be in complete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain. A special gratitude I give to NIIT center head [Mr. Dinesh Asija] who guided me through the completion of this project.

Furthermore I would like to acknowledge with much appreciation the crucial role of staff who gave the permission to use all required tools and necessary materials to complete the this project on topic **BOOK STORE DESKTOP APPLICATION**. A special thanks goes to my team mate who help me to assemble the parts and gave suggestion about the task. Last but not least, my thanks go to my faculties’ mentors and higher management of UCOE Punjabi University who have invested their full effort in guiding me in achieving the goal. I have to appreciate the guidance given by other supervisors as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comments and advices.

**Gurpreet Singh**

**11701137**

**DECLARATON**

I hereby declare that project work entitled “**BOOK STORE DESKTOP APPLICATION**” is an authentic record of my own as requirements of 45 days Industrial training during the period from June 2019 to July 2019 under the guidance of **NIIT** for the award of Bachelors of Computer Engineering, **PUNJABI UNIVERSITY (PATIALA).**

**Gurpreet Singh**

Certified that above statement made by the student is correct to the best of our knowledge and belief.

**ABSTRACT**

**Book Store Application** is a database system which is made to run on single or multiple PC. The primary purpose of this computer program is entering and retrieving information of books from a computerized database and show all the accessed information on a Graphical User Interface (GUI). A characteristic of modern desktop database applications is that they facilitate simultaneous updates and queries from multiple users. Here Front-end is considered to be the GUI and back end means the server application and database that work behind the scenes to deliver information to the user. The user enters a request through the interface. A book store database application is to be made in which a user can login to access books or a new user can register for a new account to access books.

The main objective of making this desktop application is to manage the books inside the book store. User can perform SQL queries on the database of books. Python tkinter library is used for designing interfaces and sqlite3 library is used for managing the database for books.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| **1.** | **INTRODUCTION** | **1** |
| 1.1 | What is Desktop Database Application? | 1 |
| 1.2 | Objectives of System | 1 |
| 1.3 | Features | 3 |
| 1.4 | tkinter Library | 4 |
| 1.5 | sqlite3 Library | 8 |
| 1.6 | os Library | 12 |
|  |  |  |
| **2.** | **CREATING THE USER INTERFACE DESIGN** | **13** |
| 2.1 | Designing the first interface | 13 |
| 2.2 | Designing the second interface | 14 |
| 2.3 | Designing the third Interface | 20 |
|  |  |  |
| **3.** | **BUILDING BACKEND** | **30** |
| 3.1 | Connecting with database | 31 |
| 3.2 | Inserting into the database table | 32 |
| 3.3 | View the data | 32 |
| 3.4 | Searching into the database table | 33 |
| 3.5 | Deleting data from table | 34 |
| 3.6 | Update the database table | 35 |
| **4.** | **CONNECTING FRONT-END WITH BACK-END** | **35** |
| 4.1 | Command no.1 (view) | 36 |
| 4.2 | Command no.2 (search) | 37 |
| 4.3 | Command no.3 (add) | 37 |
| 4.4 | Command no.4 (delete) | 38 |
| 4.5 | Command no.5 (update) | 40 |
|  |  |  |
| **5.** | **STAND ALONE EXECUTABLE VERSION** | **41** |
|  |  |  |
| **6.** | **FUNCTIONALITY AND SCREENSHOTS** | **43** |
|  |  |  |
| **7.** | **BIBLIOGRAPHY** | **56** |

**1. INTRODUCTION**

**1.1 WHAT IS DESKTOP DATABASE APPLICATION**

Databases are basically containers for data. Because a public library stores books, we could also say that a library is a database of books. But strictly defined, databases are computer structures that save, organize, protect, and deliver data. A system that contains databases is called a database management system, or DBM. The typical diagram representation for a database is a cylinder.

The simplest form of databases is a text database. When data is organized in a text file in rows and columns, it can be used to store, organize, protect, and retrieve data. Saving a list of names in a file, starting with first name and followed by last name, would be a simple database. Each row of the file represents a record. You can update records by changing specific names, you can remove rows by deleting lines, and you can add new rows by adding new lines.

Desktop database programs are another type of database that's more complex than a text database but intended for a single user. A Microsoft Excel spreadsheet or Microsoft Access is good examples of desktop database programs. These programs allow users to enter data, store it, protect it, and retrieve it when needed. The benefit of desktop database programs over text databases is the speed of changing data, and the ability to store large amounts of data while keeping performance of the system manageable.

**1.2 OBJECTIVES OF SYSTEM**

### ****Removes Duplicity****

### If you have lots of data then data duplicity will occur for sure at any instance. DBMS guarantee it that there will be no data duplicity among all the records. While storing new records, DBMS makes sure that same data was not inserted before.

* **Multiple Users Access**

No one handles the whole database alone. There are lots of users who are able to access database. So this situation may happen that two or more users are accessing database. They can change whatever they want, at that time DBMS makes it sure that they can work concurrently.

### ****Data Protection****

Information such as bank details, employee’s salary details and sale purchase details should always be kept secured. Also all the companies need their data secured from unauthorized use. DBMS gives a master level security to their data. No one can alter or modify the information without the privilege of using that data.

### ****Data Backup and recovery****

Sometimes database failure occurs so there is no option like one can say that all the data has been lost. There should be a backup of database so that on database failure it can be recovered. DBMS has the ability to backup and recover all the data in database.

### ****Everyone can work on DBMS****

There is no need to be a master of programming language if you want to work on DBMS. Any accountant who is having less technical knowledge can work on DBMS. All the definitions and descriptions are given in it so that even a non-technical background w=person can work on it.

### ****Integrity****

Integrity means your data is authentic and consistent. DBMS has various validity checks that make your data completely accurate and consistence.

### ****Platform Independent****

One can run dbms at any platform. No particular platform is required to work on database management system.

* **Mass Storage**

### DBMS can store a lot of data in it. So for all the big firms, DBMS is really ideal technology to use. It can store thousands of records in it and one can fetch all that data whenever it is needed.

**The main objectives of the system are:**

* User can login to access the data of books.
* A new user can register or sign up and make account.
* User can view all the records.
* User can delete selected record.
* User can update selected record.
* User can insert new record.
* User can Search for any record.

**1.3 FEATURES**

* **Minimum Duplication and Redundancy**

Because there are many users who use the database so chances of data duplicity are very high. As in database management system, data files are shared that in turns minimizes data duplication and redundancy. All the information in database management system occurs only once so chances of duplicity are very less.

### ****Saves storage space and cost****

### All the Database management systems have a lot of data to save. But DBMS proper integration of data saves much more space. Companies are paying so much amount of money to store data. If they have managed data to save then it will saves their cost of saving data and data entry.

### ****Anyone can work on it****

### Users who are not having any technical skills can work on database management system. The query language provided by DBMS is so easy to understand. If you want to update, insert, delete and search any record then it is very easy with the help of queries provided by DBMS. Any non programming user can do this with the help of any programmer.

### ****Large database maintenance****

### Large databases of big companies can be maintained only by database management system. These databases require lots of security and other feature like backup and recovery. All these features are contained in DBMS. It can maintain a database with lots of data and information.

### ****Provides high level of security****

### Security is a very big concern for all the organizations who are handling a large amount of data. DBMS doesn’t give the full access of database except DBA or head of the department. They are able to alter the database and all the users are created by them so security level of DBMS becomes so high. No other person or user can access the full database; all of them have restrictions according to their work.

### ****Permanent Storage of Data****

### DBMS stores all the data files permanently and there is no chance of any loss of data. If somehow the data get lost then there is a backup and recovery method too that can save organization’s data files. So no need to worry about data loss in DBMS.

**1.4 INTRODUCTION TO TKINTER……………………**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter library is an easy task.

**To create a tkinter:**

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the python code. The name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x is ‘tkinter’.

There are two main methods used you the user need to remember while creating the Python application with GUI.

1. **Tk (screen Name = None, base Name = None, class Name = ’Tk’, use Tk=1):** To create a main window, tkinter offers a method ‘Tk (screen Name=None, base Name=None, class Name=’Tk’, use Tk=1)’. To change the name of the window, you can change the class Name to the desired one. The basic code used to create the main window of the application is:

x **=** tkinter. Tk ( ), where x is the name of the main window object.

1. **Mainloop ( ):** There is a method known by the name Mainloop ( ) is used when you are ready for the application to run. Mainloop ( ) is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.s

x.mainloop ( )

tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes:

1. **pack ( ) method:** It organizes the widgets in blocks before placing in the parent widget.
2. **grid ( ) method:** It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. **place ( ) method:** It organizes the widgets by placing them on specific positions directed by the programmer.

## 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. We present some of these widgets as well as a brief description in the following table –

|  |  |
| --- | --- |
| **S.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** | **Check button:** The Check button 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** | **List box:** The List box widget is used to provide a list of options to a user. |
| **8** | **Menu button:** The Menu button 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** | **Radio button:** The Radio button 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** | **PanedWindow:** A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically. |
| **17** | **LabelFrame:** A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. |
| **18** | **tkMessageBox:** This module is used to display message boxes in your applications. |

**1.5 INTRODUCTION TO SQLITE3**

SQLite3 can be integrated with Python using sqlite3 module, which was written by Gerhard Haring. It provides an SQL interface compliant with the DB-API 2.0 specification described by PEP 249. You do not need to install this module separately because it is shipped by default along with Python version 2.5.x onwards.

To use sqlite3 module, you must first create a connection object that represents the database and then optionally you can create a cursor object, which will help you in executing all the SQL statements.

## Python sqlite3 module APIs

Following are important sqlite3 module routines, which can suffice your requirement to work with SQLite database from your Python program. If you are looking for a more sophisticated application, then you can look into Python sqlite3 module's official documentation.

|  |  |
| --- | --- |
| **S.No.** | **API & Description** |
| 1 | **sqlite3.connect(database [,timeout ,other optional arguments])**  This API opens a connection to the SQLite database file. You can use ": memory:" to open a database connection to a database that resides in RAM instead of on disk. If database is opened successfully, it returns a connection object.  When a database is accessed by multiple connections, and one of the processes modifies the database, the SQLite database is locked until that transaction is committed. The timeout parameter specifies how long the connection should wait for the lock to go away until raising an exception. The default for the timeout parameter is 5.0 (five seconds).  If the given database name does not exist then this call will create the database. You can specify filename with the required path as well if you want to create a database anywhere else except in the current directory. |
| 2 | **connection.cursor([cursorClass])**  This routine creates a **cursor** which will be used throughout of your database programming with Python. This method accepts a single optional parameter cursorClass. If supplied, this must be a custom cursor class that extends sqlite3.Cursor. |
| 3 | **cursor.execute(sql [, optional parameters])**  This routine executes an SQL statement. The SQL statement may be parameterized (i. e. placeholders instead of SQL literals). The sqlite3 module supports two kinds of placeholders: question marks and named placeholders (named style).  **For example** − cursor.execute("insert into people values (?, ?)", (who, age)) |
| 4 | **connection.execute(sql [, optional parameters])**  This routine is a shortcut of the above execute method provided by the cursor object and it creates an intermediate cursor object by calling the cursor method, then calls the cursor's execute method with the parameters given. |
| 5 | **cursor.executemany(sql, seq\_of\_parameters)**  This routine executes an SQL command against all parameter sequences or mappings found in the sequence sql. |
| 6 | **connection.executemany(sql[, parameters])**  This routine is a shortcut that creates an intermediate cursor object by calling the cursor method, and then calls the cursor’s executemany method with the parameters given. |
| 7 | **cursor.executescript(sql\_script)**  This routine executes multiple SQL statements at once provided in the form of script. It issues a COMMIT statement first, and then executes the SQL script it gets as a parameter. All the SQL statements should be separated by a semi colon (;). |
| 9 | **connection.commit()**  This method commits the current transaction. If you don't call this method, anything you did since the last call to commit() is not visible from other database connections. |
| 10 | **connection.rollback()**  This method rolls back any changes to the database since the last call to commit (). |
| 11 | **connection.close()**  This method closes the database connection. Note that this does not automatically call commit(). If you just close your database connection without calling commit() first, your changes will be lost! |
| 12 | **cursor.fetchone()**  This method fetches the next row of a query result set, returning a single sequence, or None when no more data is available. |
| 13 | **cursor.fetchmany([size = cursor.arraysize])**  This routine fetches the next set of rows of a query result, returning a list. An empty list is returned when no more rows are available. The method tries to fetch as many rows as indicated by the size parameter. |
| 14 | **cursor.fetchall()**  This routine fetches all (remaining) rows of a query result, returning a list. An empty list is returned when no rows are available. |

**Connect To Database**

* Following Python code shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.
* Here, we can also supply database name as the special name: memory**:** to create a database in RAM. Now we run the below code to create our database **test.db** in the current directory. Database will be created respectively.
* Once you have a connection, you can create a cursor object and call it’s execute method to perform SQL commands like CREATE, SELECT, INSERT, DELETE, UPDATE, etc.

**SQLite** is widely-used and is favorite among the developers for many reasons –

* Extremely light-weighted (not more than 500 KBs)
* It is server less which means you do not need any separate server for availing its services
* No complex setup
* Fully transactional and concurrency-compliant

**Some of the most important SQL commands:**

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)

**1.6 INTRODUCTION TO OS LIBRARY**

The OS module in python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. The \*os\* and \*os.path\* modules include many functions to interact with the file system.

Following are some functions in OS module:

1. **os.name:**

This function gives the name of the operating system dependent module imported.

1. **os.getcwd( ):**

 Function os.getcwd ( ), returns the Current Working Directory (CWD) of the file used to execute the code, can vary from system to system.

1. **os.error:**

All functions in this module raise OSError in the case of invalid or inaccessible file names and paths, or other arguments that have the correct type, but are not accepted by the operating system. os.error is an alias for built-in OSError exception.

1. **os.popen( ):**

This method opens a pipe to or from command. The return value can be read or written depending on whether mode is ‘r’ or ‘w’.

1. **os.close( ):**

Close file descriptor fd. A file opened using open (), can be closed by close () only. But file opened through os.popen (), can be closed with close () or os.close ( ). If we try closing a file opened with open (), using os.close (), Python would throw Type Error.

1. **os.rename( ):**

A file old.txt can be renamed to new.txt, using the function os.rename (). The name of the file changes only if, the file exists and user has sufficient privilege permission to change the file.

**2. CREATING THE USER INTERFACE**

**DESIGN (FRONT-END)**

* We are going to make a book store project using GUI.
* This has two parts backend and frontend. The backend is database code that interacts with database and frontend is code of tkinter library.
* The requirements of this project are to show a list of current records, search a current entry, add entry, select, update, delete data and close the window.
* First of all, user will have to login to the store to access information about the books. If a user doesn’t have any account then he/she can register or sign up as a new user.
* Here three forms are made, the first form will ask the user if he want to login or he want to register as new user. The second form will be either login page or register as new user page. This consists of the username and password entries to be filled. Only registered users are allowed to use the book store. Third from will be the book store design which consists of various text boxes, entries, list box, scrollbar and labels.

**2.1 Designing the first form**

* This form will contain two buttons and a label
* Main window is created with screen variable declared as global.
* Screen geometry and title are set using predefined methods.
* Then label is designed with pack ( ) method with shown height, width, font, text, foreground and background color.
* For leaving a gap between label and buttons, a label is designed with empty text.
* After that buttons are also designed with pack ( ) method with all the parameters as shown in figure.
* At the end mainloop ( ) method infinitely runs the loop and wait for an event to occur and process the event till the window is not closed.
* All the other forms are made as children to the first form.
* We will only call the main\_screen ( ) method and all the other methods, all the other methods will be invoked internally using this method.
* In the two buttons, commands of login and register are given which will open a new window screen of login page and register page respectively.

**Fig 1**

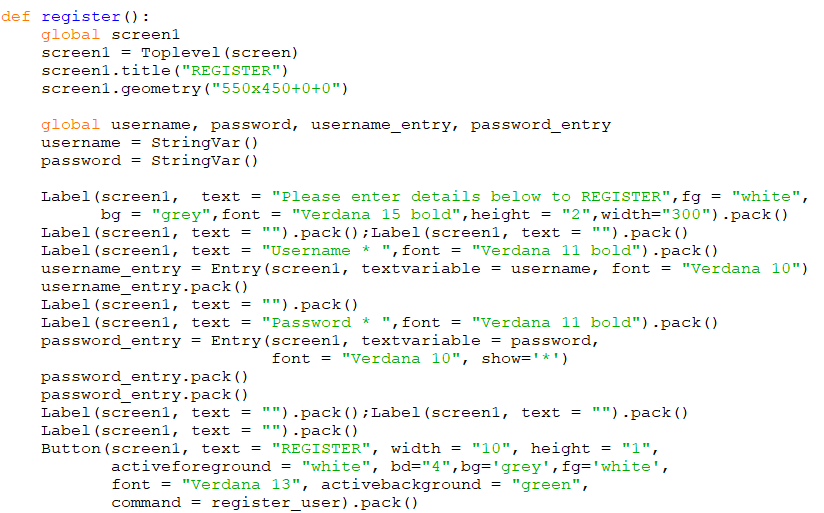
This is the code for the first form which is defined in main\_screen method

**2.2 Designing the second form**

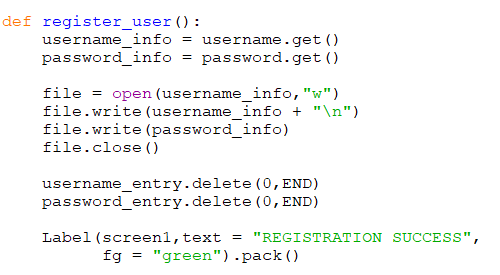
* This form is designed in login method and register method.
* Screen1 and screen2 variables are used as children to the screen variable (used in first form). This is done by using Toplevel ( ) method which is used when a python application needs to represent some extra information, pop-up, or the group of widgets on the new window. The top-level windows have the title bars, borders, and other window decorations.
* Here also title, geometry is given to both screen1 and screen2.

**2.2.1 Register ( ) method**

* As we need username and password. So username and password are created as String variable.
* Labels are created for username and password and empty label between them. Entries for username and password are created with Entry ( ) method with first argument as screen1 and text variable as username and password respectively (entries are assigned to store value into text variable). Pack ( ) method is used to put these widgets inside screen1.
* At the bottom we will add the register button. The function of this button is defined in register user method. Register user method will get the username and password details.
* Once we have all the values, we will enter into the text file.

**Fig 2**

This is the code for the second screen. User can register for new account using register method.



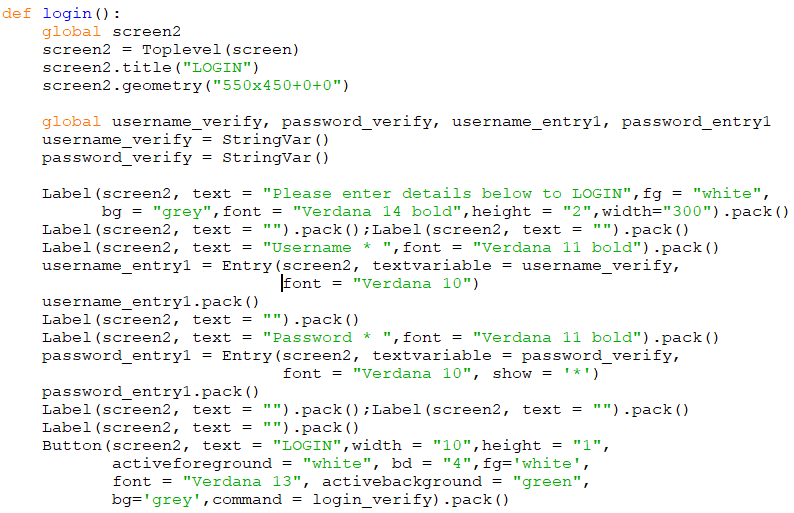
**Fig 3**

This is the code for command attached with button in register method.

* File variable is the File name which will be just a username of the user. We will open the file in the write mode. In the file we write the username and password information entered by the user. And then close the file.
* Username and password variables are made global in register method to be used in register\_user method. Entries are cleared using delete method from starting to end.
* At the end, we tell the user if registration is successful in a label.

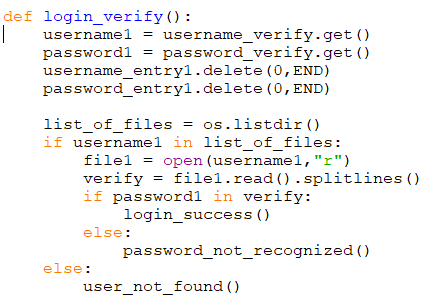
**2.2.2 Login ( ) method**

* Here also global screen2 variable is created which is used as first parameter in labels, entries and buttons. username\_verify and password\_verify are created as global string variable.
* Entries for username and password are created as global variable with respective text variables as shown in figure.

**Fig 4**

This is also code for second screen but for login method. It is used to login into book store.

* The login button is created with command login\_verify ( ). This command will validate username and password to check if the user account actually exists or not. Or if password is correctly recognized or not.
* We will get the entered username and password and store in variables username1 and password1 and clear both the entries.
* Now we need to gather list of names of files that are stored in directory where code is saving the accounts. This is done by listdir ( ) method of os library which lists all the Files with in the Current directory as a List. If usrname1 is in list of Files then we want to open the username1 file in read mode.
* Then we want to read the file in verify using read method and split lines method which breaks the string at line boundaries and returns a list of splitted lines in the form of list. It read all the lines in the text file and it will ignore blank space or blank line.
* Now we check if the entered password is in the username file which is read. If it is, then we invoke login\_success ( ) method. Else we invoke password\_not\_recognized ( ) method. If we don’t have username in the directory then we call user\_not\_found ( ) method.

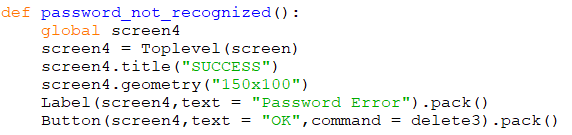


**Fig 5**

This is the code for command attached with button of login method. It is used to check user login details.

**2.2.3 Password\_not\_recognized ( ) method**

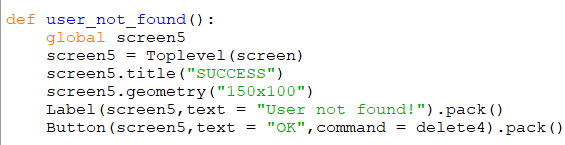
* This method is used to show the window pop up to just say that entered password is incorrect or Password
* Global variable screen4 is created with Toplevel screen which is present as first argument in label and button.
* In button, delete3 method is used to destroy this window.

**Fig 6**

Code for password not recognized method. A new screen will be displayed if password is not recognized.

**2.2.4 user\_not\_found ( ) method**

* This method is used when username is incorrect or no user is found in the directory.
* Screen4 is used; delete4 method used to destroy screen4 window.

**Fig 7**

If user is not found then a new screen popup will show up.

**2.2.4 login\_success ( ) method**

* Every time a user logs in, a session will be given to the user.
* This session consists of the code for the third window screen which is the actual book store.

C:\Users\daman\Pictures\Screenshots\Screenshot (453).png

**Fig 8**

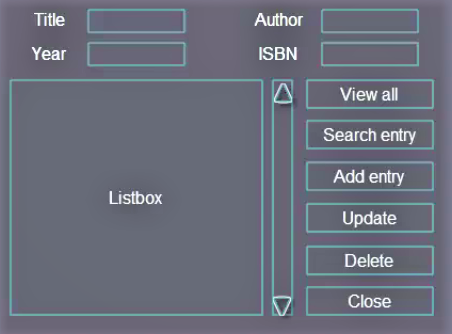
This method is defined in login verify function. If password is verified then this method is called.

**2.3 Designing the third form**

* Front end (Graphical User Interface) is made which just draws the concept show in figure and then backend is made and at the end we connect frontend with backend i.e. to connect all the commands of buttons with backend functions.
* So we will make a code that stores information about Title, Author, Year and ISBN of the book. User can view all records, Search an entry, add, update, delete records and close the code.
* The interface is to be made with the grid ( ) method of tkinter library.

**Widgets used:**

* 4 labels
* 6 buttons
* 1 list box
* 4 entries
* 1 scrollbar

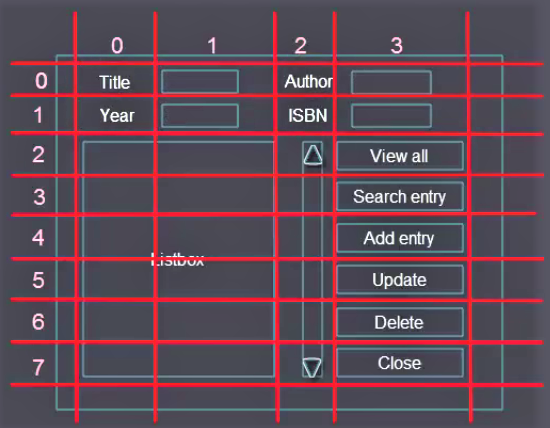


**Fig 9**

This is the design of the third from.

**2.3.1 Interface for every tool used**

* As we are using grid method, we make temporary grids on our sketch. So that when we create any widget, we pass the number of the row and the number of the column which makes things easier.
* For example “Author” label is in 0th  row and second column and update button is at 5th row and 3rd column.

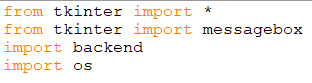


**Fig 10**

This is the structure of rows and columns to be used in third screen.

**2.3.2 Building Frontend for third form**

* The first line of the code imports all the functions from tkinter library.
* Then we initialize window with Tk ( ) method which creates a window object.
* At the end we will apply mainloop ( ) method which wraps up all the widgets that we will enter in this space.
* We give title to the window using wm\_title ( ) method. Here our window title is BookStore.
* All this code will reside in the session ( ) method.



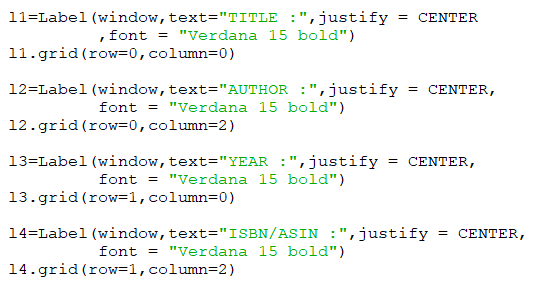
**Fig 11**

This code consists of all the modules used.

**2.3.3 Designing Labels**

Here we want to create 4 labels:

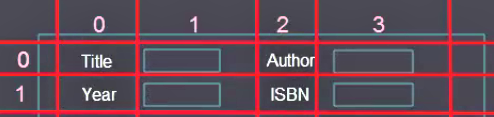
* Label l1 – “ Title ”
* Label l2 – “ Author ”
* Label l3 – “ Year ”
* Label l4 – “ ISBN ”
* l1, l2, l3, l4 are 4 label objects.
* Window is the parent window.
* grid ( ) method is applied according to the position number of row and column.

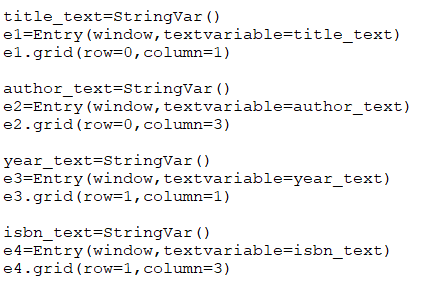
**Fig 12**

This is the code for designing labels.

**2.3.4 Designing Entries**

* Entry gets window as the first parameter like any other widget and a text variable parameter which expects as an argument the value that the user will enter in the entry widget, it is a special data type.
* We need to create an object of the data type. In case of Title entry we create title\_text equal to StringVar ( ), this function creates spatial object.



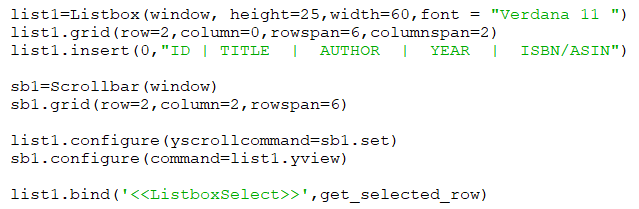
****

**Fig 13**

This is the code for designing Entries.

**2.3.5 Adding List box and scrollbar**

* In list box, we want to specify height and width of list box; we can test different height and width and select which fits better.
* We want to span the list box on 2 columns 0 and 1 and across 6 rows (from 2nd to 7th row).
* Configure method is applied both to list box and scrollbar. This method takes arguments, vertical scrollbar along the y axis will be set to this scrollbar
* When we scroll the bar, the vertical list of the view will change.
* Listbox method is used to make listbox as list on which window list box is to be made; height and width are specified in the arguments.
* Scrollbar is made with Scrollbar method on the window with 6 rows spanned from 2nd to 7th row and in 2nd column.
* Scroll bar is attached with List box using yview function (scrolled in Y – axis direction). get\_selected\_row function is used to select a row (discussed later)



**Fig 14**

Code for adding list box and scrollbar.

**2.3.6 Designing Buttons**

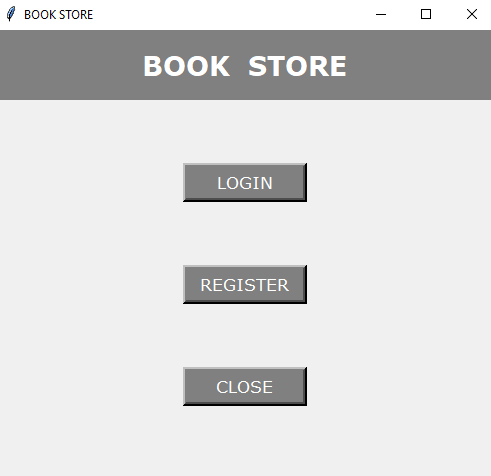
* 6 buttons are to be made and aligned in the 3rd column and from 2nd to 7th row.
* We have button method and arguments as window, text, width and command which decide what the particular button will do.
* Commands (defined later…) written as arguments in Button ( ) function performs a particular task. So, user can perform different task by just pressing particular button. In button b6 (close button) window.destroy command destroys the window.



**Fig 15**

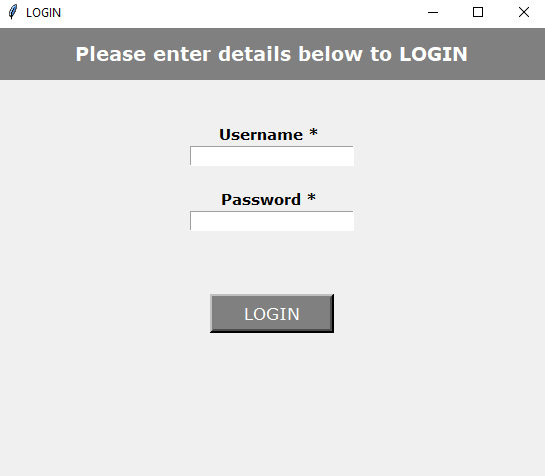
Code for designing and adding buttons.

**2.3.6 Final look of Interfaces**

****

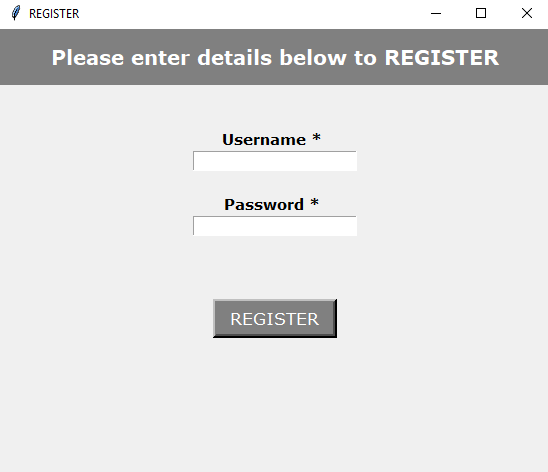
**Fig 16**

Main screen form. This form consist of



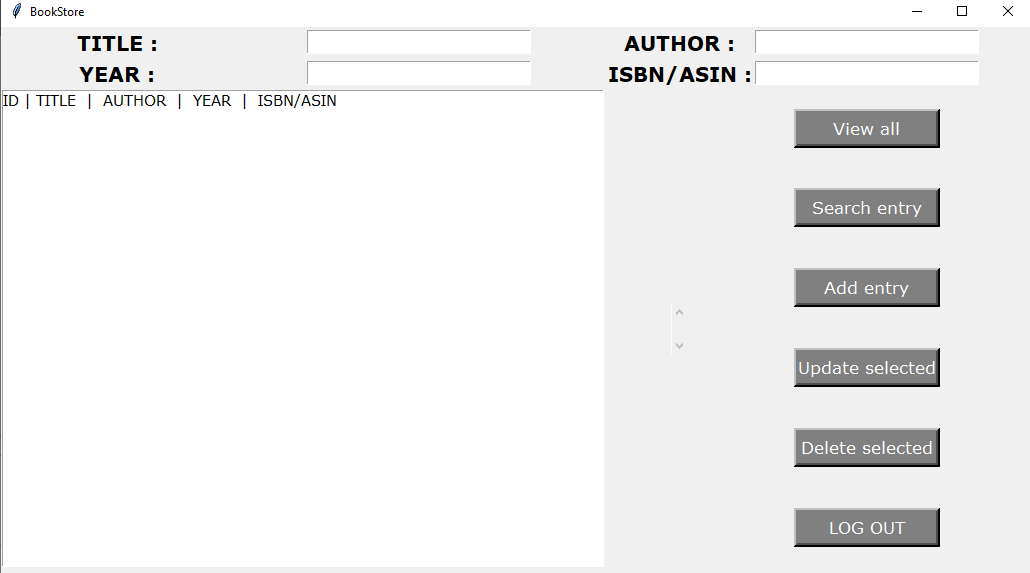
**Fig 17**

The Login Page.



**Fig 18**

Registration Page.



**Fig 19**

This is the third form of book store.

**3. BUILDING THE BACKEND**

* For making back-end, we create other script and then import this script in the front-end. Let’s call new script backend.py
* First thing we want to do in backend script is to import sqlite3 library.

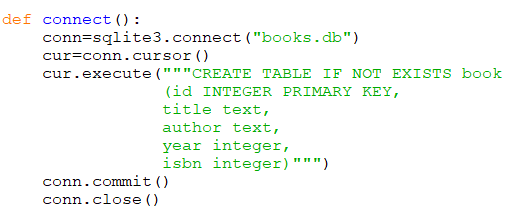
5 functions are created in the date base:

* connect( )
* insert( )
* view( )
* search( )
* delete( )
* update( )

These functions are then connected to the buttons in the front-end.

**3.1 Connecting with Database**

* First of all, we make a connect function which consists of code for establishing connection with database
* To use backend, we have to make a database file through which we can save our data. ‘books.db’ is created which contains all information regarding books.
* Conn is the connection object which establishes a connection with database using connect method. Then cursor object cur is defined using cursor method and executes the SQL statement inside double quotes.
* Every time a user executes a program the script has to check if there is an existing table in books.db database. If there is a table then we don’t create the table and if there is no table, we want to create the table. We name the table book.
* Here we need to specify arguments in the table. Starting from id which is just a number that will start from zero and it will auto increment by one, so that we have control of how many records we have in database. So id is set as primary key. All the other columns with data type are specified..
* Commit method is used to permanently save the changes made to the database
* At the end, connection is close using close method.
* We have to call connect function which will run any time when front end code is executed.

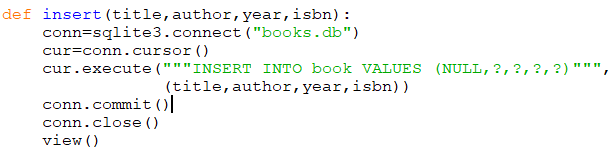


**Fig 20**

This code is used to connect to database and create table with given attributes.

**3.2 Inserting into database table**

* As there are 4 entries so we pass 4 parameters are passed inside insert function (used to insert data into database table) as shown.
* Here also we make connection to the database and make a cursor object. So when user presses add entry button, a new connection to database will be established.
* While inserting into the table book, first we have an id which is auto increment value. So we don’t have to pass that manually and we can pass NULL value. Python will create id automatically. 4 question marks are for each 4 values.
* Second parameter in the execute function is passed as a tuple of title, author, year and ISBN.

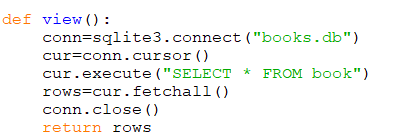


**Fig 21**

Insert method is used to insert data into database.

**3.3 Display the data**

* View function will fetch all the rows of the database table.
* Here cursor is executed by simple “SELECT” SQL query. As this is select statement so will not perform any changes to the database, so commit method is not used.
* rows variable is used which fetches all the rows of a query result. It returns all the rows as a list of tuples. Selected rows will be put into the list box. An empty list is returned if there is no record to fetch.
* Our data is stored in the rows variable.
* rows are returned after closing the connection.



**Fig 22**

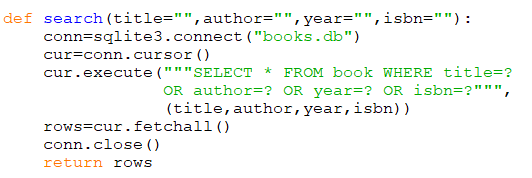
View method is used to display the data from database table.

**3.4 Searching into the database table**

* In this function, an OR search is implemented which means that user will enter a title or an author name, or the year, or the ISBN number or he can enter all of them at the same time. If user enters the year only then he’ll get all the rows corresponding to that year.
* So 4 parameters are passed. If user want to pass let’s say the author name only to the entries so the function will get only the arguments for the author parameter and other parameters will remain without a value, we may get error as the function is designed to have 4 parameters. In that case we pass some empty strings as default values, so now if user passes any particular author name, the SQL statement will search for an empty title or for author name or for empty year, all the rows are returned with that particular author name.
* So arguments are written as:

search (title = “ ”, author = “ ”, year = “ ”, isbn = “ ”)

* we connect to the database, cursor object is passed and then execute cursor which select row where title is equal something or author is equal something or author and so on….. And second parameter will be a tuple of arguments.
* Then we need to fetch all the rows that will be returned from this query using fetchall method and return the rows at the end after closing the connection.

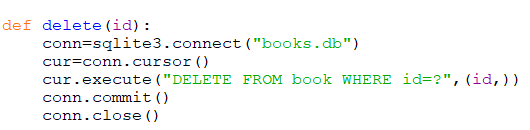


**Fig 23**

This is code for searching data. And is used to search from databases table.

**3.5 Deleting data from database table**

* User wants to select one of the records from the list box and they want to select one of the records and press delete button.
* So we want to grab the list selection as a tuple which has unique identification number like the ID. We want to refer to that ID and go to database table and delete the row with that ID.
* Here delete ( ) method is used which takes single argument, id. User can delete a record just by selecting record from the list box.
* We want to delete from book where id is equal to something.
* User can see if the record has been deleted or not, user can simply check it by displaying the record by clicking on view button.

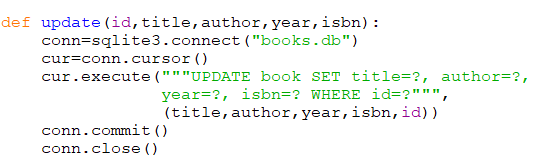


**Fig 24**

This is code for deleting data from database.

**3.6 Update the data**

* To update records, user may want to select a row from list box and value of that particular selected row would display in the entry widgets. And from there, user may want to change one of the entries and press the update button.
* Here also we will get the selection from the list box and then we would refer to the ID as the first parameter. We want to get the new values that the user wants from those entries.
* All the four parameters along with the ID is passed. And then we want to update the table where ID is equal to question mark.



**Fig 25**

This is code for updating data into database table.

**4. CONNECTING FRONT-END WITH**

**BACKEND**

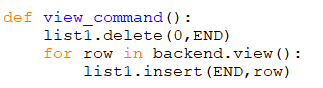
* Now we want frontend script is to fetch data from the backend script, or more specifically is able to get the function outputs which are basically tuples of data, and then insert those tuples of data in the list box. So basically the user presses the button and then these functions are triggered and the data are inserted to the list box.
* Connecting frontend with backend involvers import backend script.

**Following commands are implemented in the front end script which is connected with buttons:**

* View
* Search
* Add
* Delete
* Update
* Get\_selected\_row

**4.1 view command**

* In this function we want to get the records in the list box when the user presses the view all button.
* Here deleting the list box(list1) takes 2 arguments, 0 and END which ensures that everything is deleted from the row with index of 0 to the END(last row).
* Then we iterate through the view function in the backend and insert all the rows into the list box. Insert method takes two arguments. The first arguments is the index where we want to insert our values and second is the row in which is iterated through all the rows in backend view command.
* This function is attached with button using : - b1=Button(window,text="View all", width=12,command=view\_command)



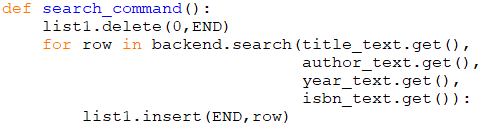
**Fig 26**

view\_command method is used to view all the records in database. This method internally used view method from the backend.

**4.2 search command**

* In this function also, first we want to empty the list using delete method.
* Then we need to loop to the backend list search( ) method which takes 4 parameters. Text that the user is inputting in the entry widget is taken. Since title\_text is a StringVar object so its not a plain string. We need append a get method to output a string object. If user enter any text in the widget, get method will produce a simple text string.
* Search function will output for us when it searches the database, then we simply insert new values at the end of the list using insert method.
* This function is attached with button using – (wrapper function)

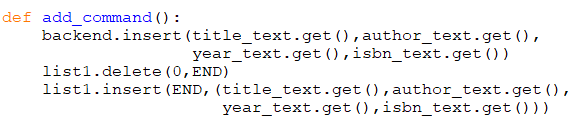
b2 = Button (window,text="Search entry", width=12, command = search\_command)

**Fig 27**

search\_command is used to perform searching in the database table. This method internally calls search method from backend.

**4.3 add command**

* This function is used to store the data that user is inputting in the entries.
* Here we need to call insert method from backend and get all the 4 arguments of this method.
* Entry is shown in list box when user presses add entry.
* First we need to ensure if the list box is empty using delete method.
* Then we insert new values at the end of the list which we have taken from get method (i.e. input from user). Here we put all the user input values as a single value, as a tuple.
* This function is attached with button using : -
* b3=Button(window,text="Add entry", width=12,command=add\_command)

**Fig 28**

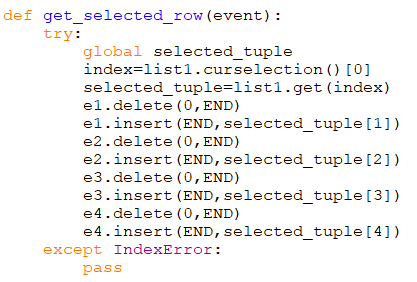
add\_command method is used to insert the records entered by user into the database table.

**4.4 delete command**

* Here when the user selects one of the rows in the list box, we need to grab the ID of that row and send back ID to the delete function over the backend script.
* A user wants to delete a particular row, so we implement a method to get a row selected by the user.
* bind( ) method is used to bind a function to a widget event. It takes 2 arguments the event type and function that we want to bind to the event type.

This is written as: list1.bind('<<ListboxSelect>>', get\_selected\_row)

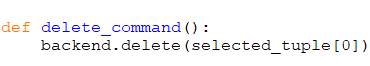
* This function gets special parameter called event which holds the information about the type of the event. So python knows that when we pass these to the bind method, it expects the function to have this event parameter.
* We can implement get\_selected\_row function in try except block. When the get\_selected\_row function is called, Python will try to execute the indented block under try. If there is an Index Error none of the lines under try will be executed. Instead the line under except will be executed which is pass. The pass statement means do nothing. So the function will do nothing when there's an empty list box.
* line 6, in get\_selected\_row , index=list1.curselection()[0] Index Error: tuple index out of range
* To get the index of the selected row of the list box, curselection method is used. We grab the item with index 0 of the tuple.
* From the list box, get the tuple with index x.
* If we print the selected\_tuple and select a row from a list box then we get row printed out as tuple.



**Fig 29**

get\_selected\_row method returns the selected tuple from the list box and fills the values of that selected record in the respective entries.

* Now delete command is implemented
* As the delete method get the argument as ID from backend, so we have tuple called get selected row, we need to return the selected , if we call the function we get the selected row and we need the item with index 0 from the get selected row method.
* We want the item with index 0 from the selected tuple.
* When user select one of the rows we want TO fill the entries with selected row, insert method of entry widget is used. First we make sure that entries are empty using delete method from 0 to end. We want to insert at the END in author entry (first entry) the selected\_tuple[1] as id has 0 index and so on for the other remaining entries.

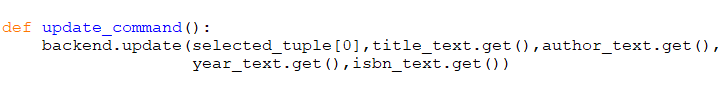


**Fig 30**

Delete command is used to delete selected tuple. This method internally calls delete function from backend.

**4.5 update command**

* The update method from backend takes five arguments.
* A user can update any of the particular record by selecting that record from the list box and then making changes to the particular entries and press the update button. And further he can check the updated record by pressing view all button.
* We want to keep the ID of the selected row as the user is changing the other values not the ID. When selected row has all the data in the entries we can update or change them. All updated values are sent back to the selected row.
* This function is attached with button using : -
* b5 = Button(window,text = "Add entry", width = 12,command = update\_command)

**Fig 31**

Update\_command is used to update records. This method internally calls update function from the backend.

**5. STANDALONE EXECUTABLE**

**VERSION**

* We can execute the script using python but if you want to give your project to someone else, they can run it if they have python installed in their PC. And execute the program by going to the terminal and invoking python and the script. That may be difficult for users who are not experienced with programming.
* We want to make standalone executable version that we can send to anyone and they can click and go ahead, insert data, retrieve data and so on.
* File will work for executable version in any operating system.
* There exists a library to make standalone executables in python i.e. “pyinstaller” (can be installed using pip install pyinstaller).
* Pyinstaller script makes a .exe file on windows, .f file on MAC and also bunch of other files are generated which are associated without program. We can go to the files and find the error if any.

Pyinstaller --onefile --windowed frontend.py

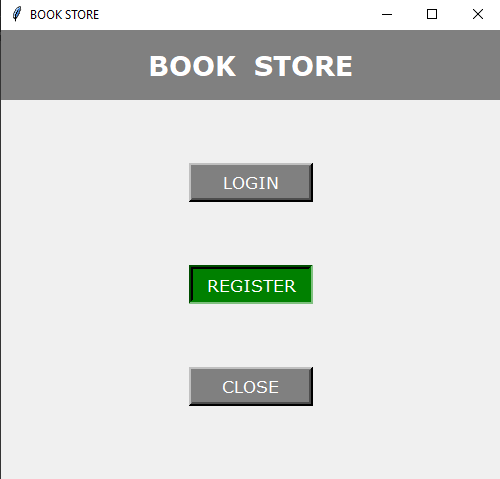
* This will create a single executable file.
* frontend.py is holding all the code and it also imports the backend script.
* After running this command where frontend.py is present, frontend.exe will be created (in windows).
* By opening the frontend.exe file we get the whole window interface. books.db file is generated by default. When frontend.exe is executed for the first time, connect method in the backend script which creates the database.
* If we click view all button, no data will be shown in the list box as we don’t have any data because books.db is created from scratch.
* So we want to give the user both the executable file and the existing database.

C:\Users\daman\Pictures\Screenshots\Screenshot (420).pngC:\Users\daman\Pictures\Screenshots\Screenshot (419).png

**Fig 32**

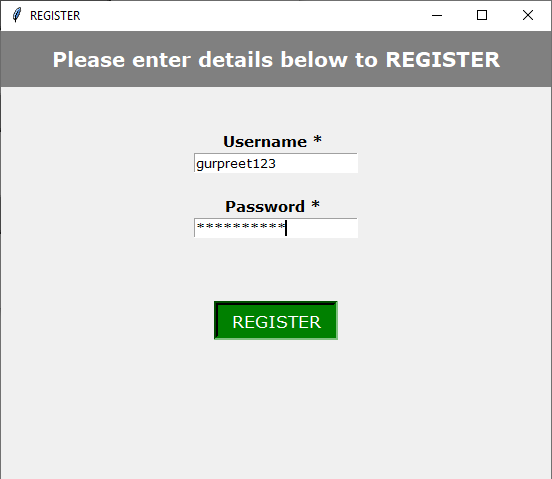
Commands for making single executable file

**6. FUNCTIONALITY / SCREENSHOTS**



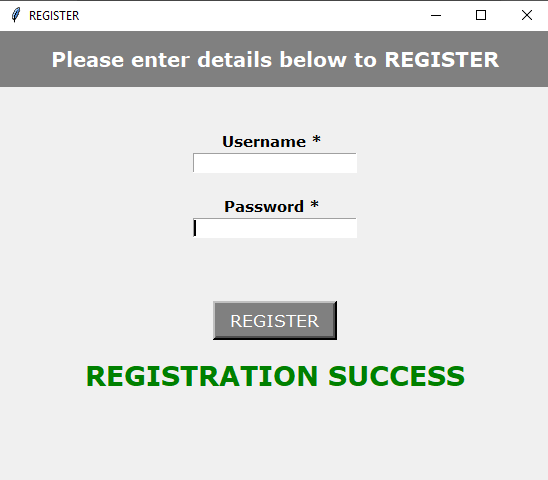
**Fig 33**

A new user can register by pressing REGISTER button.



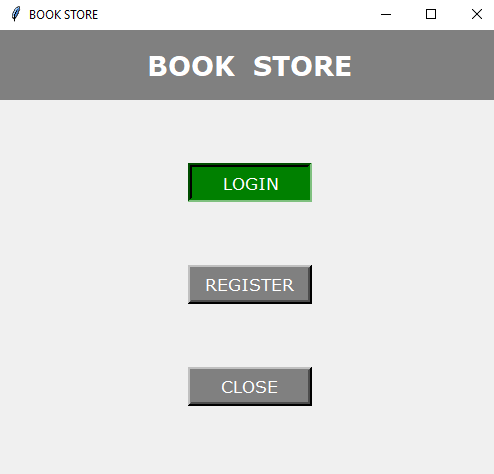
**Fig 34**

This is the second form which will be displayed after pressing REGISTER button. New user can enter username and password to register for new account.



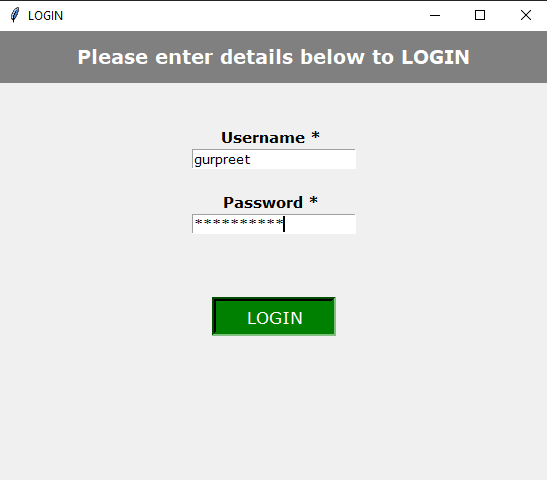
**Fig 35**

After entering username and password, a message will be displayed if user is successfully registered.



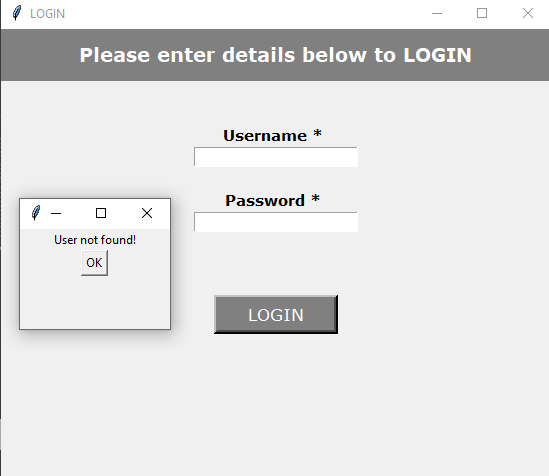
**Fig 36**

After registration, user can login by pressing LOGIN button.

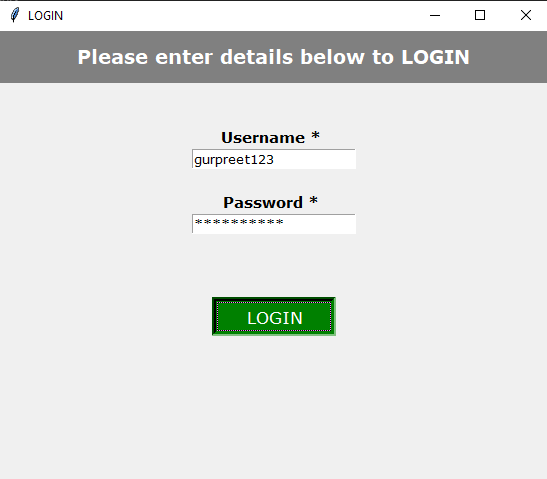


**Fig 37**

In login screen, user can enter registered username and password to access database. Here we try to login with wrong username.

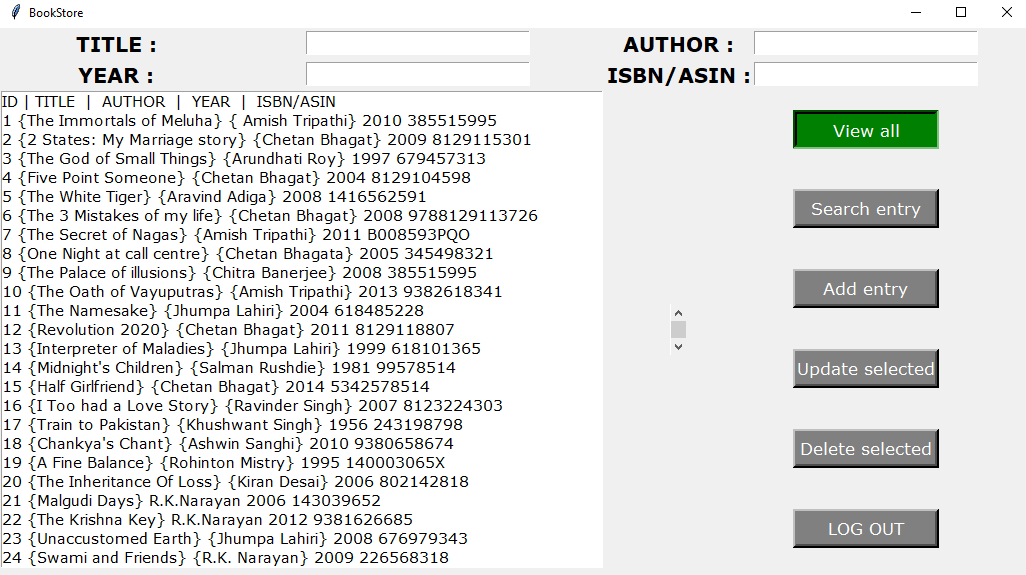
**Figure 38**

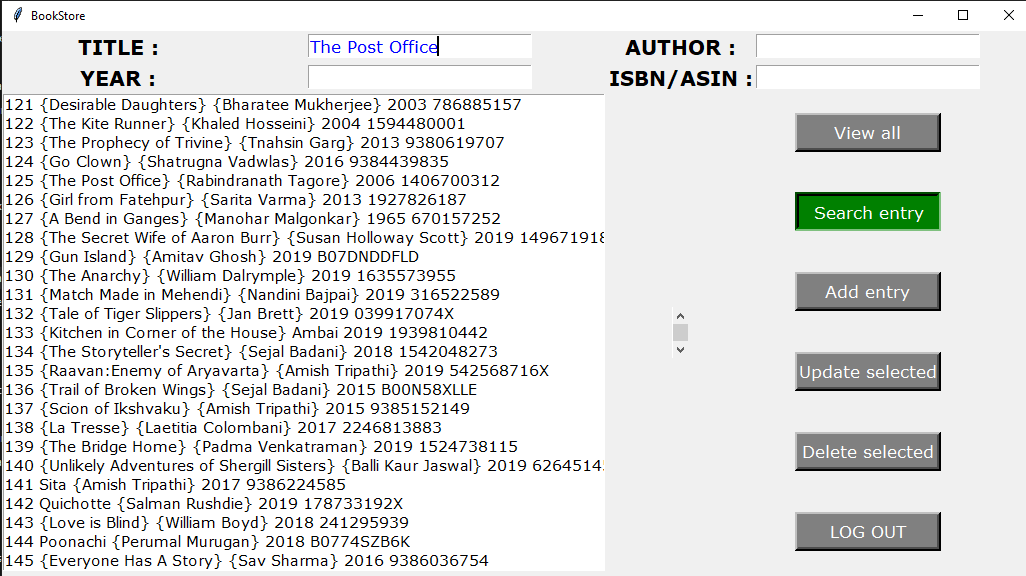
If user is not found then a new window popup will show up.

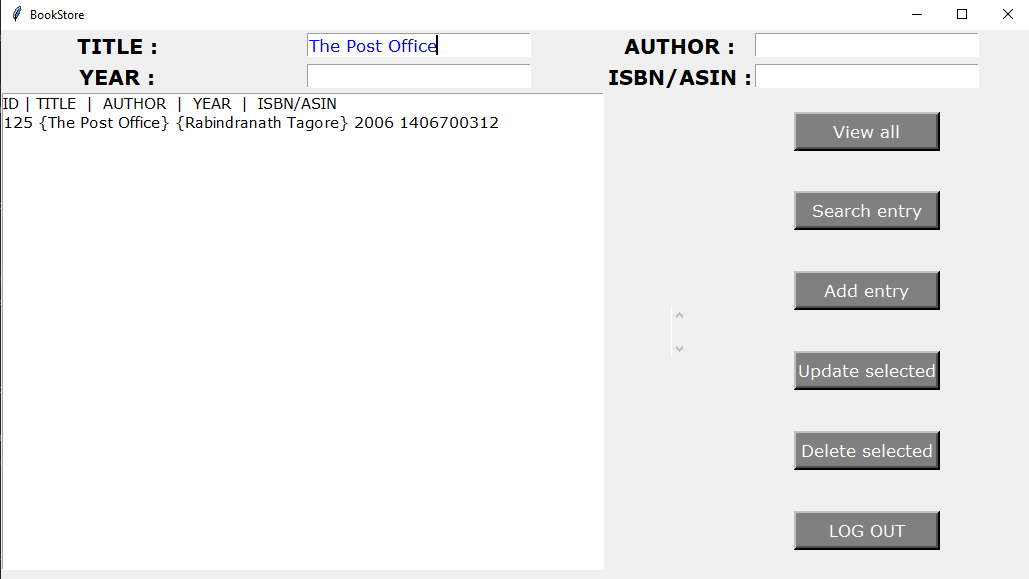
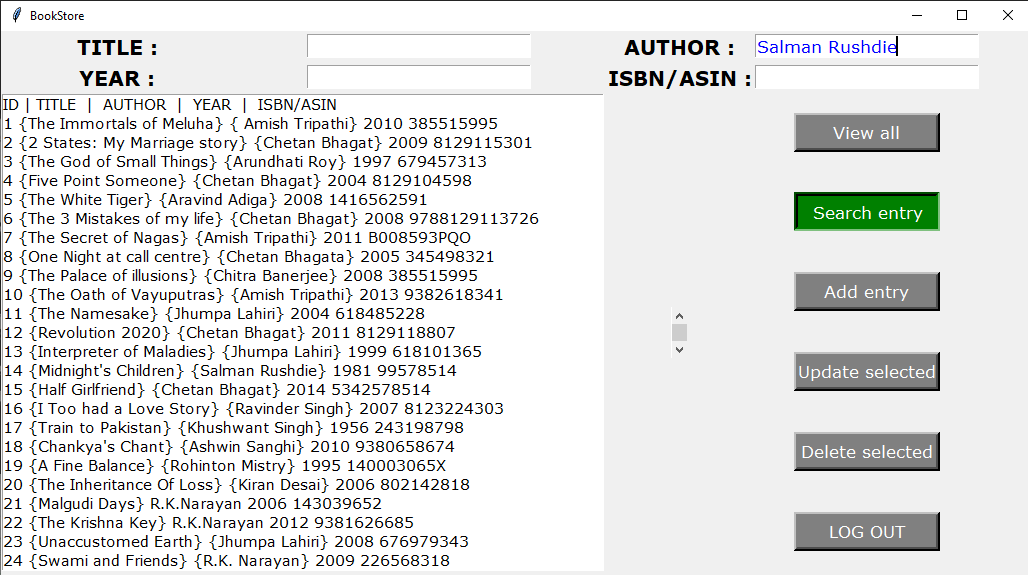
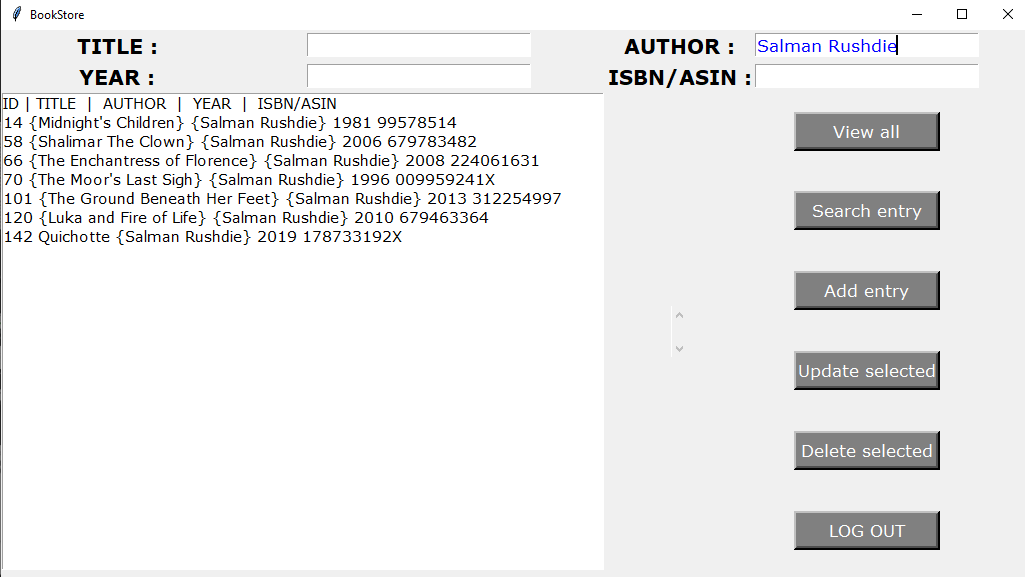
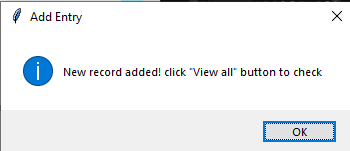
**Figure 39**

Entering the correct username and password

* Figure 40 represents the interface form of bookstore.
* Figure 41 represents that when “View all” button is pressed, data of all the books will be shown.

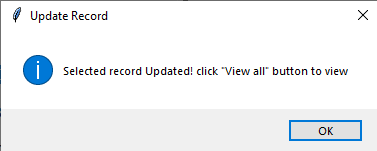
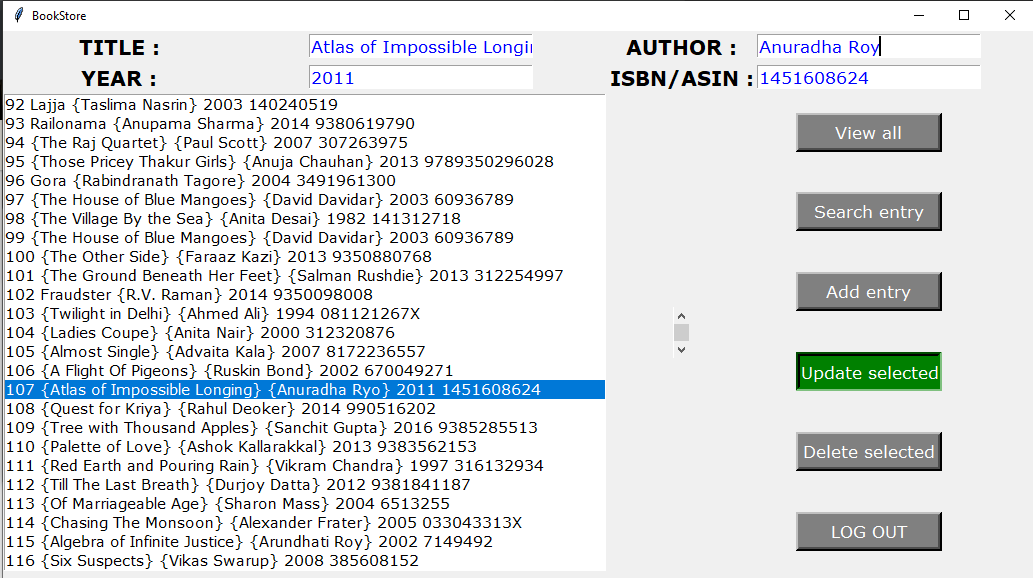
**Fig 40**



**Fig 41**: Searching a book by the Title.**Fig 42:** Book found! **Fig 43:** Searching a book by Author**Fig 44:** All the Author’s written books will be listed. 

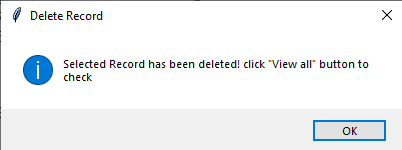
**Fig 45**

To add a book, we fill all the required entries and press the “ Add entry ” button. New book will be added at the end of all the books.



**Fig 46**

We can update a record just by selecting that particular record and then changing the entries of attributes of book, then press ” Update Selected “ button.



**Fig 47**

To delete any book’s record, just select that record from the list and press “Delete selected” button.

**7. BIBLIOGRAPHY**

**Websites:**

* [www.w3schools.com](http://www.w3schools.com)
* [www.wikipedia.com](http://www.wikipedia.com)
* [www.goodreads.com](http://www.goodreads.com)
* [www.quora.com](http://www.quora.com)
* www.tutorialspoint.com