Database Systems - Assignment-04

April 5, 2021

Team: INFOSEC
Professor: Reza Sadeghi
Working Group: Anuj Rane - arane1@unh.newhaven.edu
Kunal Dhanaitkar - kdhan1@unh.newhaven.edu
Ritesh Kumar Parmar - rparm1@unh.newhaven.edu
Udaypratap Prahlad Rathore - urath1@unh.newhaven.edu

Table of Contents

1. Executive Summary
2. Software Used
3. Description
4. Testing
5. Problems Faced
6. Conclusion

Executive Summary

The aim of this assignment is to create a graphical user interface for the Library management system database. A python library tkinter will be used for creating the graphical user interface for the database.

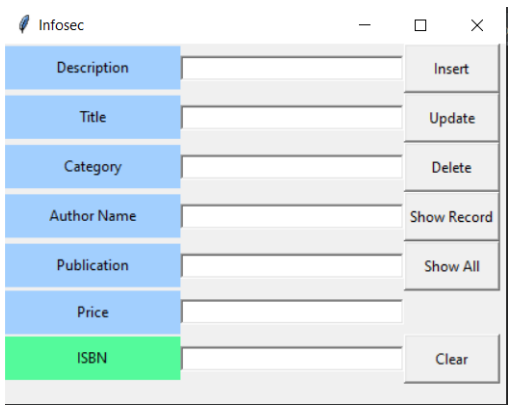
Software Used

1. My-SQL workbench
2. Pycharm IDE
3. Overleaf

Description

A library management system is being designed to keep track of different books, services and users in a library. In the previous assignments, first an ER diagram was created and then it was imported into MySQL using the MySQL-workbench. In this assignment adding to the previous assignments, a graphical user interface will be created for the database and some important operations like insert, update, delete and search are provided for the users. In the following sections, the whole process of creating a database, and then configuring all the curd operations will be explained in detail.

STEP 1: CREATING A GRAPHICAL USER INTERFACE



GUI for Libpos

In this section we will be looking at how the graphical user interface was designed for library management using the python Tkinter library. The attached image above shows the final output of our graphical user interface for this assignment.Let's break this GUI and discuss the code snippet in detail that was written for designing it.

```
bookDescription = Label(root, text="Description", width=20, height=2, bg="pink").grid(row=0, column=0, sticky=E)
bookTitle = Label(root, text="Title", width=20, height=2, bg="pink").grid(row=1, column=0)
bookCategory = Label(root, text="Category", width=20, height=2, bg="pink").grid(row=2, column=0)
bookAuthorName = Label(root, text="Author Name", width=20, height=2, bg="pink").grid(row=3, column=0)
bookPublication = Label(root, text="Publication", width=20, height=2, bg="pink").grid(row=4, column=0)
bookPrice = Label(root, text="Price", width=20, height=2, bg="pink").grid(row=5, column=0)
bookIsbn = Label(root, text="ISBN No.", width=20, height=2, bg="pink").grid(row=6, column=0)

e1 = Entry(root, width=30, borderwidth=2)
e1.grid(row=0, column=1)
e2 = Entry(root, width=30, borderwidth=2)
e2.grid(row=1, column=1)
e3 = Entry(root, width=30, borderwidth=2)
e3.grid(row=2, column=1)
e4 = Entry(root, width=30, borderwidth=2)
e4.grid(row=3, column=1)
e5 = Entry(root, width=30, borderwidth=2)
e5.grid(row=4, column=1)
e6 = Entry(root, width=30, borderwidth=2)
e6.grid(row=5, column=1)
e7 = Entry(root, width=30, borderwidth=2)
e7.grid(row=6, column=1)
```

Code snippet for label and input box

Creating label

The above code snippet was used to design the input boxes and the label for the said input boxes.The books table have seven attributes, so seven labels were created which lists all the attributes, namely Description, Title, Category, Author Name, Publication, Price and ISBN.

```
"bookDescription = Label(root, text="Description", width=20,
height=2, bg="pink").grid(row=0, column=0, sticky=E)"
```

For instance the above code snippet was used to design a label for the description input box. In this the 'root' defines that it will be using the main window, then the 'text' attribute is used to specify the text that will be displayed on the screen. The attributes 'width' and 'height' is used for the size of the label and the 'bg' attribute is used to specify the background color for the label. Further the '.grid() is used to position the label on the screen, by specifying the row and the column. Using the same method other labels were also created for GUI.

Creating input box

```
"e1 = Entry(root, width=30, borderwidth=2)
e1.grid(row=0, column=1)"
```

To create an input box 'Entry()' will be used.In this 'root' attribute is used to specify that it will be on the main window. Further, the 'width' and 'height' attribute is used to define the size of the input box, and the 'grid()' with row and column is used to position the input box on the screen. In the same way, following the above approach other input boxes were created.

Creating button

```
"button1 = Button(root, text="Create", width=10,
height=2).grid(row=7, column=0)"
```

The above code snippet is used for creating the buttons for the CURD operations in the graphical user interface. To create a button 'Button()' function is used, and the attributes that we have to specify are 'root' to place it on the main window.Then 'text' to write some text on the button, and then 'width' and 'height' for defining the size of the button on the screen, and 'grid()' for placing the button on the right place in the screen. Following the same approach other buttons were created.

STEP 2: CONNECTING DATABASE AND GUI

```
"import mysql.connector
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="root",
    database="libpos"
)"
```

To connect the database and the GUI, a python package 'mysql.connector' will be imported which allows to connect our database and the GUI. Further to connect the database and the GUI, 'msql.connector.connect()' function was used and the 'host' name was provided. And for authentication the 'user' and the 'password' with the database to be connected is given. And after this mysql authenticates allows us to successfully connect our database and the GUI.

STEP 2: CREATING CRUD OPERATIONS

In this section the CRUD operations will be configured for the button of our GUI. For this assignment insert, update, delete and search operations are provided, and all these operations will covered with their code snippet in this section.

Insert operation

```
def insertData():
    Description = e1.get()
    BookTitle = e2.get()
    Category = e3.get()
    AuthorName = e4.get()
    Publication = e5.get()
    BookPrice = e6.get()
    ISBN = e7.get()
    if Description != "" and ISBN != "" and BookTitle != "" and Category != "" and AuthorName != "" and Publication != "" and BookPrice != "":
        insert_Data = "Insert into Book (Description,BookTitle,Category,AuthorName,Publication,BookPrice,ISBN) value " \
                      "(%s,%s,%s,%s,%s,%s,%s) "
        value = (Description, BookTitle, Category, AuthorName, Publication, BookPrice, ISBN)
        mycursor.execute(insert_Data, value)
        mydb.commit()
        messagebox.showinfo("Info", "Record Inserted")
    else:
        messagebox.showinfo("Info", "Enter Valid Records")
```

Insert function

The above code snippet belongs to the insert operation. An 'insertData()'function is created which triggers on the insert button click in the GUI. In this function first the data is fetched using the input box name followed by the '.get()' function. This pulls the data from the front end and stores them in variables. Then the if condition verifies if the input values are not null, else it will through an error 'Enter Valid Records'. Then the insert query pushes all the data into the database and displays a success message on the screen 'Record Inserted'.

Update Operation

```
def updateData():
    ISBN = e7.get()
    dbIsbn = ""
    Select = "Select * from Book where ISBN='%s'" % (ISBN)
    temp_label.config(text=ISBN)
    mycursor.execute(Select)
    result = mycursor.fetchall()
    temp_label1.config(text=Select)
    for i in result:
        ISBN = i[0]
        if ISBN != dbIsbn:
            messagebox.askyesno("Information", "Record Already exists")
            Description = e1.get()
            BookTitle = e2.get()
            Category = e3.get()
            AuthorName = e4.get()
            Publication = e5.get()
            BookPrice = e6.get()
            Isbn = e7.get()
            update_Data = "Update Book set Description,BookTitle,Category,AuthorName,Publication,BookPrice where ISBN "
                          "='%s'" % (Description, BookTitle, Category, AuthorName, Publication, BookPrice, Isbn)
            mycursor.execute(update_Data)
            mydb.commit()
            messagebox.showinfo("Info", "Record Updated")
        else:
            messagebox.askyesno("Information", "Record Doesn't exists")
            insertData()
            clearData()
```

Update function

An 'updateData()' function is created, which triggers on the update button click from the GUI. It first fetches the 'ISBN' of the book entered by the user to update, and then runs a query to pull the entry from the database based on the ISBN provided. Then grabs all the updated input entered by the user, and runs an update query and commits all the updated values to the database and shows a pop-up 'Record updated'. If the provided ISBN doesn't matches it shows an error 'Record Doesn't exists'.

Delete Operation

```
def deleteData():
    ISBN = e7.get()
    delete = "Delete from Book where ISBN = '%s'" % (ISBN)
    mycursor.execute(delete)
    mydb.commit()
    messagebox.showinfo("Information", "Record Deleted")
    clearData()
```

Delete function

To delete entries from the database, a 'deleteData()' function is created. This function, fetches the ISBN from the user, and query for the entry based on the ISBN provided and then runs a delete query to drop the entry from the database and shows a message 'Record deleted'.

Search Operation

```python
def searchRecord():
    ISBN = e7.get()
    dbISBN = ""
    Select = "Select ISBN from Book where ISBN ='%s'" % (ISBN)
    mycursor.execute(Select)
    result1 = mycursor.fetchall()
    for i in result1:
        dbISBN = i[0]
        Select1 = "Select Description,BookTitle,Category,AuthorName,Publication,BookPrice from Book where ISBN ='%s'" % (
            ISBN)
        mycursor.execute(Select1)
        result2 = mycursor.fetchall()
        Description = ""
        Title = ""
        Category = ""
        AuthorName = ""
        Publication = ""
        Price = ""
        ISBN = ""
        for i in result2:
            Description = i[0]
            Title = i[1]
            Category = i[2]
            AuthorName = i[3]
            Publication = i[4]
            Price = i[5]
        e1.insert(0, Description)
        e2.insert(0, Title)
        e3.insert(0, Category)
        e4.insert(0, AuthorName)
        e5.insert(0, Publication)
        e6.insert(0, Price)
```

Search function

To search an entry in the database, a 'searchRecord()' function is created. This takes ISBN number of the book from the user as an input, and the queries for the data based on the ISBN provided and then displays the result in the input box if the data is found.

Show all Operation

```python
def showAllRecord():
    class A(Frame):
        def __init__(self, parent):
            Frame.__init__(self, parent)
            self.CreateUI()
            self.LoadTable()
            self.grid(sticky=(N, S, W, E))
            parent.grid_rowconfigure(0, weight=1)
            parent.grid_columnconfigure(0, weight=1)

        def CreateUI(self):
            tv = Treeview(self)
            tv['columns'] = ('Description', 'Title', 'Category', 'AuthorName', 'Publication', 'Price', 'Isbn')
            tv.heading('#0', text='Description', anchor='center')
            tv.column('#0', anchor='center')
            tv.heading('#1', text='Title', anchor='center')
            tv.column('#1', anchor='center')
            tv.heading('#2', text='Category', anchor='center')
            tv.column('#2', anchor='center')
            tv.heading('#3', text='AuthorName', anchor='center')
            tv.column('#3', anchor='center')
            tv.heading('#4', text='Publication', anchor='center')
            tv.column('#4', anchor='center')
            tv.heading('#5', text='Price', anchor='center')
            tv.column('#5', anchor='center')
            tv.heading('#6', text='Isbn', anchor='center')
            tv.column('#6', anchor='center')
            tv.grid(sticky=(N, S, W, E))
            self.treeview = tv
            self.grid_rowconfigure(0, weight=1)
            self.grid_columnconfigure(0, weight=1)
```

Show all function

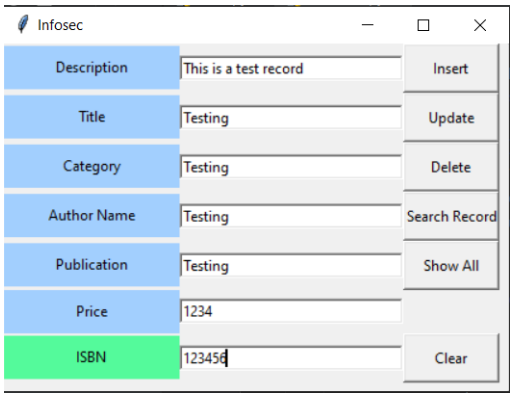
This is an additional feature that was included in this assignment, this displays all the entry in the database with the Show All record button is clicked. For this 'showAllRecord()' function is created which queries for all the entry in the database and then displays all them in a tabular form in a new pop-up window. This feature is just an additional feature to see all the entries in the database.

Testing

In this section we will be testing the graphical user interface for the library management system, and we will also test all the CRUD operations provided.

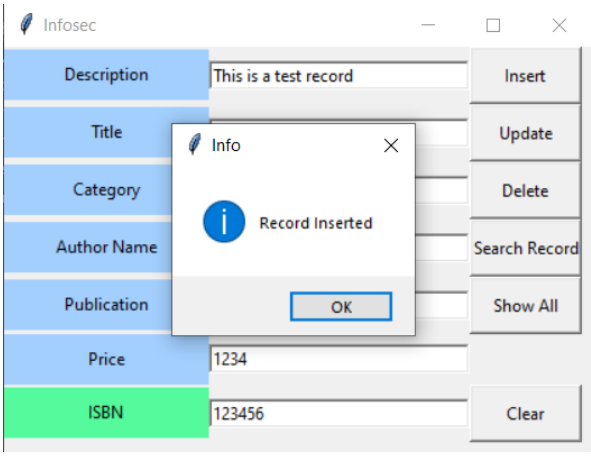
a. Insert

Now we will test to enter some record in the database. Input some record in the input box, and then click the insert button.



Inserting record in database

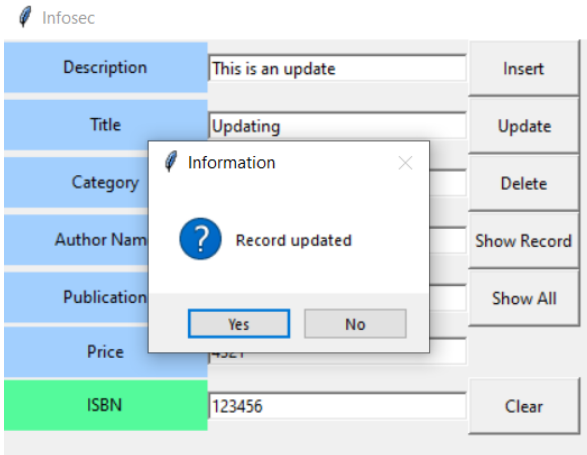
When the insert button is clicked, then a pop-up appears that says 'Record Inserted', and hence our record is successfully inserted into the database.



Successfully inserted

b. Update

Now an existing record will be updated in the database. Enter the updated values with ISBN number of the book that needs to be updated and then click the update button.

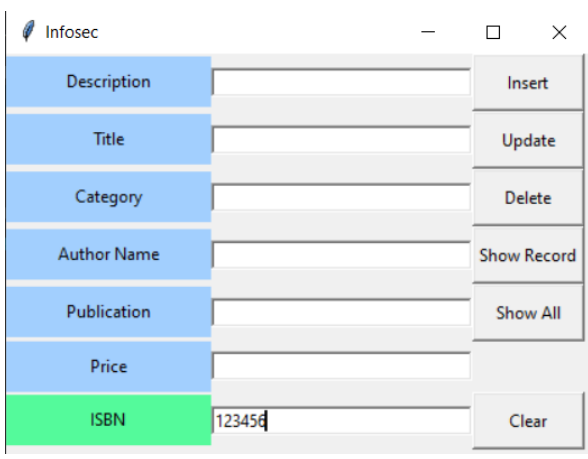


Successfully updated

In the above screen shot it can be seen the data was successfully updated.

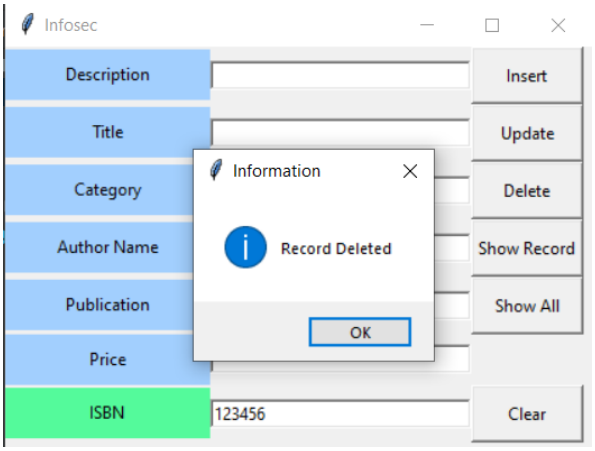
c. Delete

Enter the ISBN number of the record that you want delete from your database, and hit the delete button.



ISBN to Delete

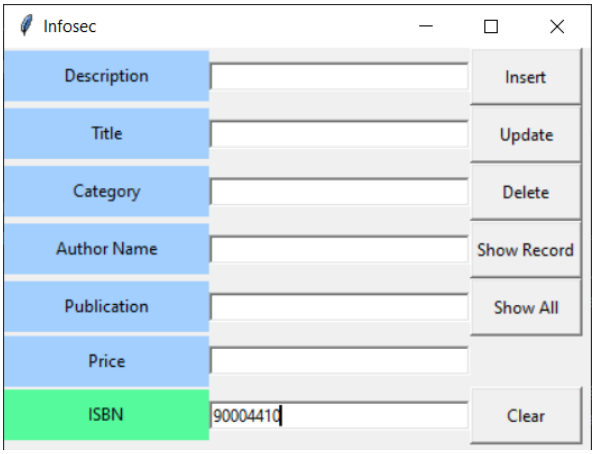
As the record was found for the entered ISBN in our database, it was successfully deleted.



Successfully deleted

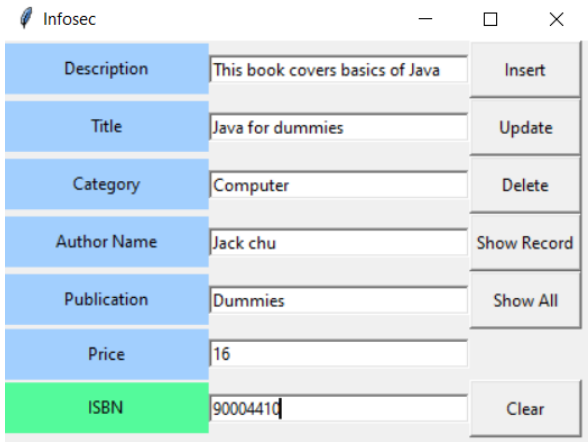
d. Search

To search a record in the database, provide an ISBN number for the record that you want to search.



ISBN for Search

Based on the ISBN provided the data was pulled from the database and displayed in the input boxes.



Record Found

e. Show all

To display all the record available in the database, just click show all button and all the data in the database, will be displayed in a pop-up.

Description	Title	Category	AuthorName	Publication	Price	Isbn
This book covers	Calculus for dum	Math	Mark Ryan	Dummies	12	90002202
This book covers	Cryptocurrency	Technology	Kiana Danial	Dummies	23	90003301
This book covers	SQL for dummies	Computer	Peter Weverka	Dummies	19	90004402
This book covers	PHP for dummies	Computer	Doug Lowe	Dummies	18	90004406
This book covers	Javascript for dun	Computer	Doug Lowe	Dummies	17	90004408
This book covers	Java for dummies	Computer	Jack chu	Dummies	16	90004410
This book covers	Linux for dummie	Computer	Greg Harvey	Dummies	19	90004412
This book covers	Android for dumr	Computer	Sundar Pi	Dummies	18	90004414
This book covers	Web-DEV for dun	Computer	Bhavik Nahar	Dummies	23	90004416

Displaying Record

f. Clear

The clear button clears all the data from the input if there is any.

Problems Faced

The team faced some problems while connecting the database with GUI, and then some problem in the update operation. But we figured everything pretty quickly and faced no problems after that in completing the assignment.

Conclusion

A graphical user interface for Library management system was successfully created and CRUD operations like insert, update, delete and search was provided.