UNIVERSITI TEKNOLOGI MARA

DIPLOMA IN LIBRARY INFORMATICS

UNIVERSITI TEKNOLOGI MARA (UITM) KEDAH, CAMPUS SUNGAI PETANI

FACULTY OF INFORMATION SCIENCE STUDIES COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

IML208 PROGRAMMING FOR LIBRARIES

GROUP ASSIGNMENT

PREPARED BY:

NAME	STUDENT NO
NUR FARISHA ALEEYA BINTI MOHAMED ZAKHARI	2022644296
NURIN ANISA BINTI ANUAR	2022834762
PUTRI NUR YASMEEN BINTI ADIE PUTRA	2022663052
WASILATUL ‘ATHIYYAH BINTI MOHAMMAD JALI	2022893212

CLASS:

CDIM1443B

PREPARED FOR:

SIR AIRUL SHAZWAN BIN NORSHAHIMI

SUBMISSION DATE:

17th JANUARY 2024

IML208 PROGRAMMING FOR LIBRARIES

GROUP ASSIGNMENT

NUR FARISHA ALEEYA BINTI MOHAMED ZAKHARI (2022644296)

NURIN ANISA BINTI ANUAR (2022834762)

PUTRI NUR YASMEEN BINTI ADIE PUTRA (2022663052)

WASILATUL ‘ATHIYYAH BINTI MOHAMMAD JALI (2022893212)

CDIM1443B

DIPLOMA IN LIBRARY INFORMATICS

UNIVERSITI TEKNOLOGI MARA (UITM) KEDAH, CAMPUS SUNGAI PETANI

FACULTY OF INFORMATION SCIENCE STUDIES COLLEGE OF COMPUTING, INFORMATICS AND MATHEMATICS

SUBMISSION DATE:

17th JANUARY 2024

ACKNOWLEDGEMENT

First, we are grateful to The Almighty God for establishing us to complete this group assignment. In performing our assignment, it is a successful one we had to take the help and guideline.

We would like to express our deepest appreciation and gratitude to Sir Airul Shazwan bin Norshahimi, our esteemed lecturer, for his exceptional guidance, knowledge, and unwavering support throughout this course (IML 208). His expertise and dedication to teaching have truly made a profound impact on our assignment journey. We are incredibly grateful of his ability to create an inclusive and nurturing learning environment. His approachable nature, patience, and willingness to listen have fostered a sense of belonging and encouraged open dialogue among students.

Sincere thanks to all friends for their kindness and moral support during this group assignment.

Finally, our deepest gratitude goes to beloved family for their endless love, prayers, and encouragement. To those who indirectly contributed to this research, your kindness means a lot to us.

Thank you very much.

TABLE OF CONTENT

1.0 Introduction **1**

2.0 Problem Statement 2

3.0 Objectives **3**

4.0 Flowchart **4**

4.1 User registration 4

4.2 Book entry 5

4.3 Calculation of book 6

5.0 Snapshot of Code **7**

5.1 Code user registration 7

5.2 Code book entry 8-10

5.3 Code calculation of book 11-12

6.0 Snapshot of project (GUI) **13**

6.1 GUI user registration 13

6.2 GUI book entry 14

6.3 GUI calculation of book 15

7.0 Snapshot of database (XAMPP) **16**

7.1 Database (XAMPP) user registration 16

7.2 Database (XAMPP) book entry 17

7.3 Database (XAMPP) calculation of book 18

8.0 Conclusion **19**

1.0 INTRODUCTION

In today's digital age, libraries are embracing technology to streamline their operations and enhance the borrowing experience for their patrons. One essential aspect of library management is the rental book system, which allows users to borrow books and keep track of due dates, fines, and book availability.

The form is built using the Tkinter library in Python and provides fields for entering book ID, book title, book type, book price, and date of borrowing. The main purpose of this code is to capture book details from users and perform actions such as updating, deleting, or submitting the entered book information. However, it is important to note that the functionality for these actions is not implemented in the code. The form includes various user interface elements such as labels, entry fields, dropdown menus, and buttons. Users can enter book information into the corresponding fields, select the book type from a dropdown menu, and interact with the buttons for update, delete, or submit.

Design a GUI-based book price calculator using tkinter that allows users to select a book type from a dropdown menu, enter the number of packs, and the number of days late. The calculator should calculate the total price based on the book type and number of packs, and also calculate the late return fine based on the number of days late. The calculated results should be displayed in the GUI. Additionally, the program should store the user's input (book type, packs, days late, total price, and late return fine) in a MySQL database.

2.0 PROBLEM STATEMENT

The library currently relies on manual processes for book rentals, which are time-consuming and error prone. Library staff must manually record book check-outs, due dates, and fine calculations, leading to inefficiencies and potential inaccuracies. Additionally, library patrons face challenges in finding available books, tracking due dates, and managing fines.

Users typically purchase books from actual bookstores and libraries, or factual shops. It requires users to physically visit the libraries or stores and browse the various shelves, and it also requires store managers to stock, display, and move the merchandise that users need. To complete these procedures, personnel, time, and space are required. Additionally, the high price of new books proved to be an issue for today's students because they had to purchase books that they would only need for a short while, like for semester exams.

The Book Rental System provides a solution for reducing and optimising these costs. Authorized users do not need to go to the factual shops and libraries to choose and bring the books it can saves time and save human efforts. The customer can save money of the book price and also reusing books proves helpful to save nature.

The goal is to develop a software solution that automates and streamlines the rental book process for both library staff and patrons. The system should provide an intuitive user interface for patrons to search for books, reserve them, and receive notifications about due dates and fines. For library staff, the system should enable easy management of book records, user accounts, and fine calculations.

3.0 OBJECTIVES

1. Book Management: The system should allow library staff to add, update, and delete book records, including information such as title, author, genre, and availability status.

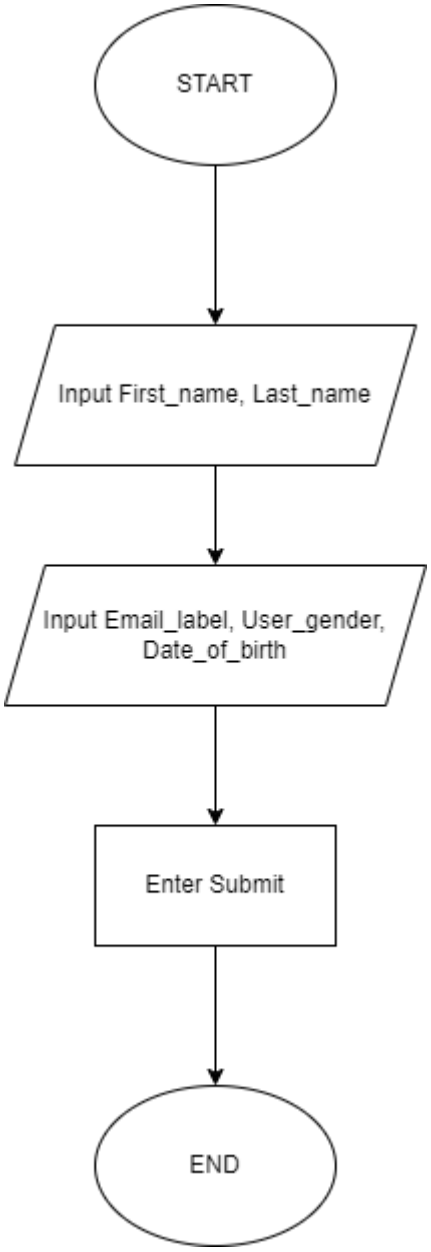
2. User Management: The system should enable library staff to create and manage user accounts, including personal information, borrowing history, and fine records.

3. Book Search and Reservation: Patrons should be able to search for books based on various criteria, such as title, author, or genre. They should also be able to reserve books if they are currently unavailable.

4. Fine Calculation: The system should automatically calculate fines for late returns based on predefined rules, such as a fixed fine rate per day or a grace period.

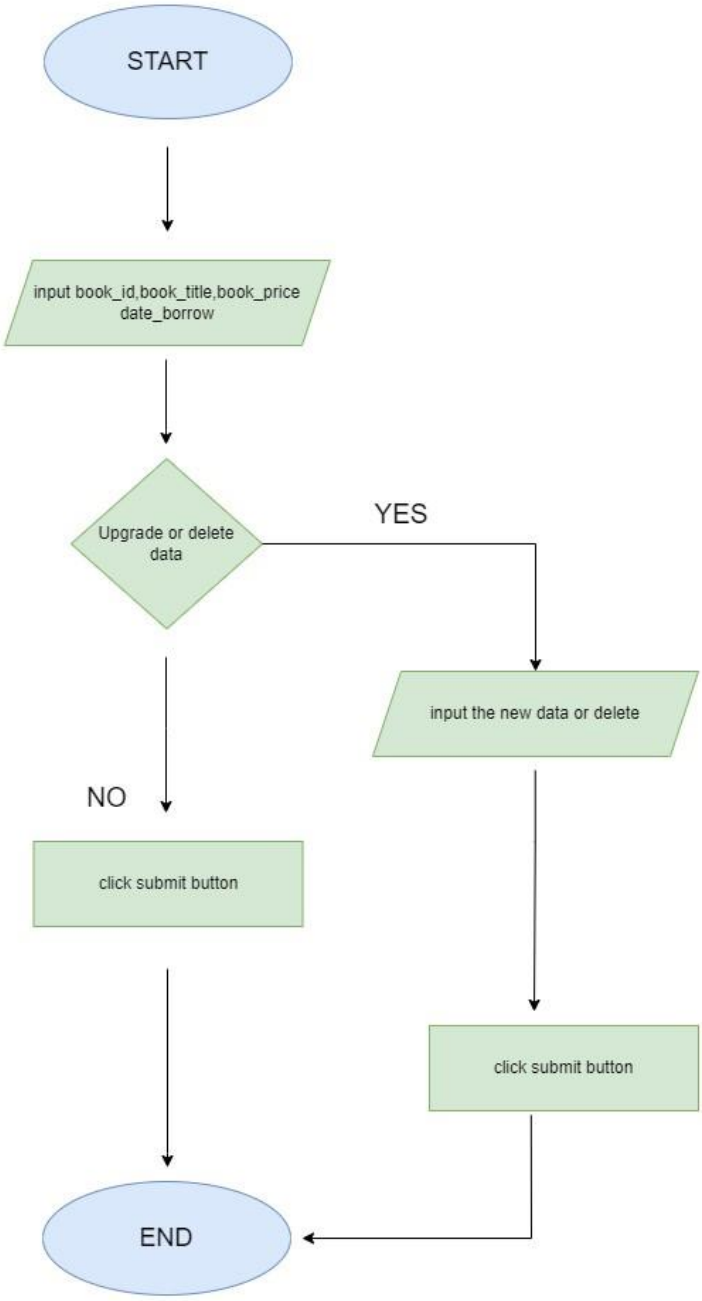
4.0 FLOWCHART

4.1 USER REGISTRATION



FLOWCHART 2

4.2 BOOK ENTRY



FLOWCHART 3

4.3 CALCULATION OF BOOK

5.0 SNAPSHORT OF CODE

CODE USER REGISTRATION

register.py > ...

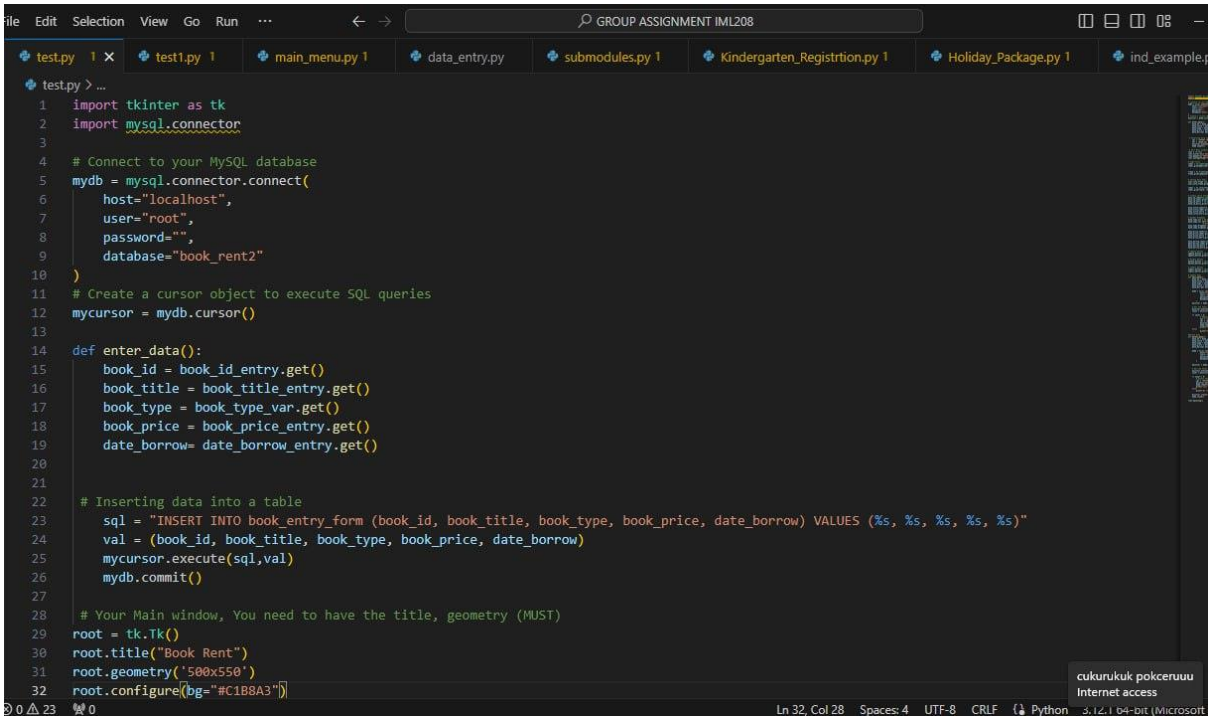
```python
import tkinter as tk
from tkcalendar import DateEntry
import mysql.connector

# Connect to your MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="register"
)

# Create a cursor object to execute SQL queries
mycursor = mydb.cursor()

# Function to handle the calculation and database saving
def submit_data():
    first_name = first_name_entry.get()
    last_name = last_name_entry.get()
    user_email = user_email_entry.get()
    user_password = user_password_entry.get()
    user_gender = user_gender_label.get()
    date_birth = date_birth_entry.get()

    # To insert your Data to your database.
    sql = "INSERT INTO user_registration (First_Name, Last_Name, User_Email, User_Password, User_Gender, Date_of_Birth) VALUES (%s, %s, %s, %s, %s, %s)"
    val = (first_name, last_name, user_email, user_password, user_gender, date_birth)
    mycursor.execute(sql, val)
    mydb.commit()


root = tk.Tk()
root.title("User Register")
root.geometry('500x400')
root.configure(bg='#a36370')
```

register.py > ...

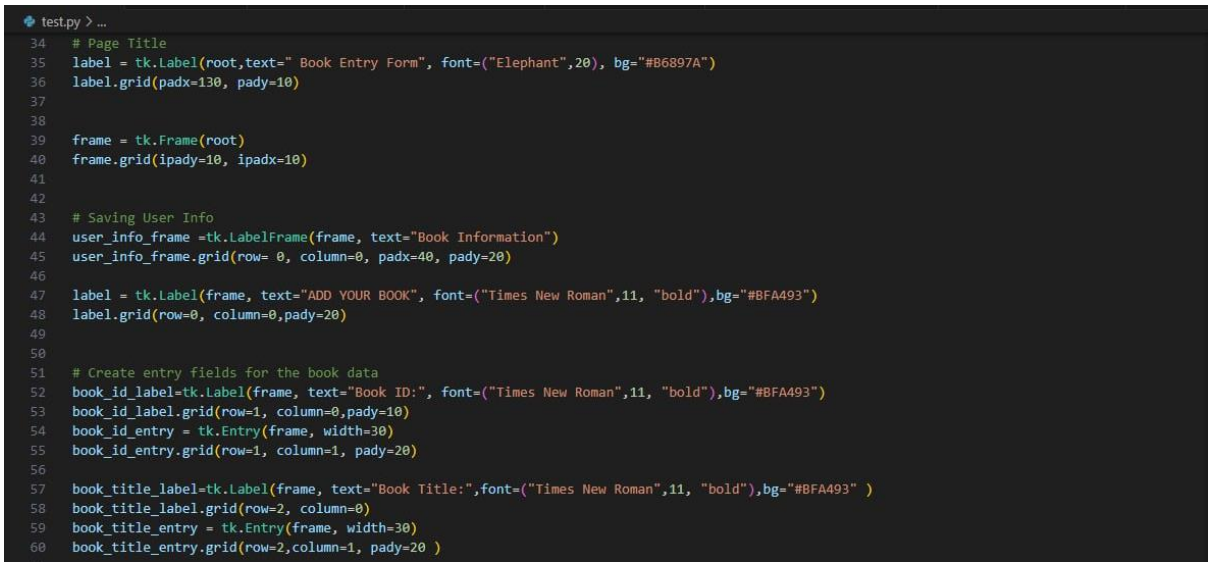
```python
label = tk.Label(root, text="USER REGISTRATION", font=("New York",20,"bold"), bg='#a36370', fg='#ffffff')
label.grid(row=0, column=1, padx=5, pady=5)

## User first name
first_name_label = tk.Label(root, text="First Name", font=("New York",15,"bold"), bg='#a36370', fg='#ffffff')
first_name_label.grid(row=1, column=0, padx=5, pady=5)
first_name_entry = tk.Entry(root, bd=3)
first_name_entry.grid(row=1, column=1, padx=5, pady=5)

# User last name
last_name_label = tk.Label(root, text="Last Name", font=("New York",15,"bold"), bg='#a36370', fg='#ffffff')
last_name_label.grid(row=2, column=0, padx=5, pady=5)
last_name_entry = tk.Entry(root, bd=3)
last_name_entry.grid(row=2, column=1, padx=5, pady=5)

# User email
user_email_label = tk.Label(root, text="Email", font=("New York",15,"bold"), bg='#a36370', fg='#ffffff')
user_email_label.grid(row=3, column=0, padx=5, pady=5)
user_email_entry = tk.Entry(root, bd=3)
user_email_entry.grid(row=3, column=1, padx=5, pady=5)

# User password
user_password_label = tk.Label(root, text="Password", font=("New York",15,"bold"), bg='#a36370', fg='#ffffff')
user_password_label.grid(row=4, column=0, padx=5, pady=5)
user_password_entry = tk.Entry(root, bd=3)
user_password_entry.grid(row=4, column=1, padx=5, pady=5)

# User gender
user_gender_label = tk.StringVar(root)
user_gender_label.set("Gender")
trip_dropdown = tk.OptionMenu(root, user_gender_label, "Male", "Female")
trip_dropdown.grid(row=5, columnspan=2, padx=5, pady=5)
```

```python
# User date of birth
date_birth_label = tk.Label(root, text= "Date of Birth", font=("New York",15,"bold"))
date_birth_label.grid(pady= 10, columnspan=2)
date_birth_entry = DateEntry(root, date_pattern="yyyy-mm-dd", bd=3)
date_birth_label['background'] = "#F8CBDC"
date_birth_entry.grid(pady=5, columnspan=2)

# Save Button
save_button = tk.Button(root, text="SUBMIT",font=("New York",15,"bold"), bg='#a36370', fg='#ffffff', command=submit_data)
save_button.grid(row=8, column=0, columnspan=2)

root.mainloop()
```

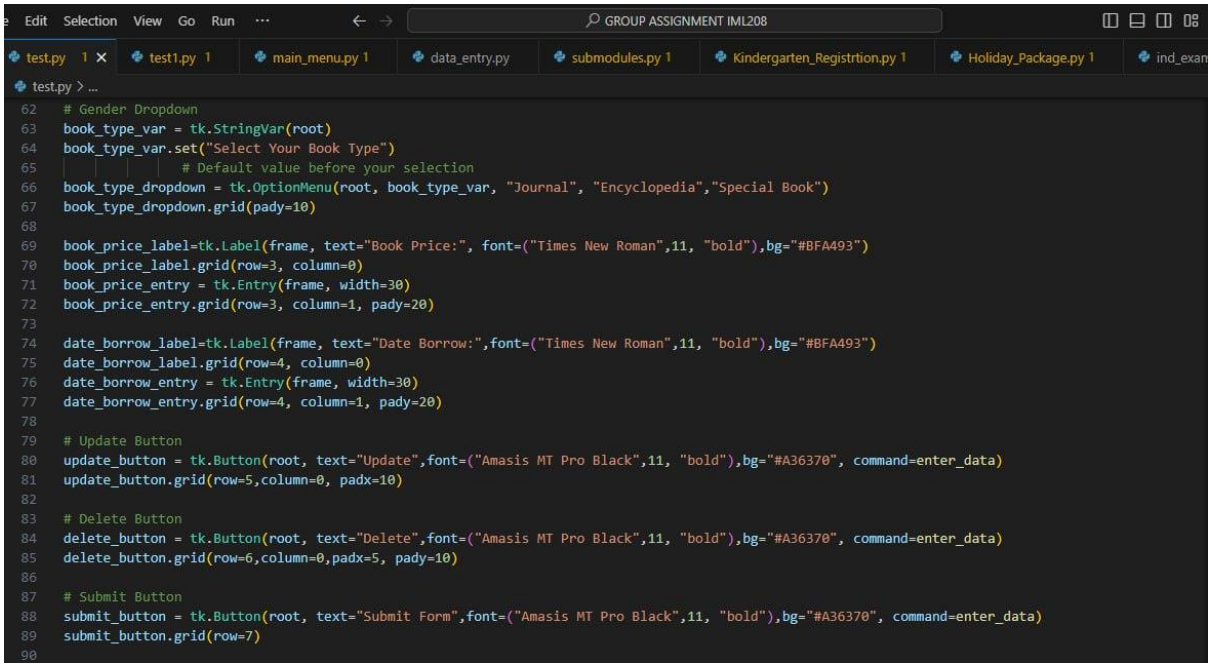
5.2 Code Book Entry



```python
import tkinter as tk
import mysql.connector

# Connect to your MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="book_rent2"
)
# Create a cursor object to execute SQL queries
mycursor = mydb.cursor()

def enter_data():
    book_id = book_id_entry.get()
    book_title = book_title_entry.get()
    book_type = book_type_var.get()
    book_price = book_price_entry.get()
    date_borrow= date_borrow_entry.get()


 # Inserting data into a table
    sql = "INSERT INTO book_entry_form (book_id, book_title, book_type, book_price, date_borrow) VALUES (%s, %s, %s, %s, %s)"
    val = (book_id, book_title, book_type, book_price, date_borrow)
    mycursor.execute(sql,val)
    mydb.commit()

 # Your Main window, You need to have the title, geometry (MUST)
root = tk.Tk()
root.title("Book Rent")
root.geometry('500x550')
root.configure(bg="#C1B8A3")
```



```python
# Page Title
label = tk.Label(root,text=" Book Entry Form", font=("Elephant",20), bg="#B6897A")
label.grid(padx=130, pady=10)


frame = tk.Frame(root)
frame.grid(ipady=10, ipadx=10)


# Saving User Info
user_info_frame =tk.LabelFrame(frame, text="Book Information")
user_info_frame.grid(row= 0, column=0, padx=40, pady=20)

label = tk.Label(frame, text="ADD YOUR BOOK", font=("Times New Roman",11, "bold"),bg="#BFA493")
label.grid(row=0, column=0,pady=20)


# Create entry fields for the book data
book_id_label=tk.Label(frame, text="Book ID:", font=("Times New Roman",11, "bold"),bg="#BFA493")
book_id_label.grid(row=1, column=0,pady=10)
book_id_entry = tk.Entry(frame, width=30)
book_id_entry.grid(row=1, column=1, pady=20)

book_title_label=tk.Label(frame, text="Book Title:",font=("Times New Roman",11, "bold"),bg="#BFA493" )
book_title_label.grid(row=2, column=0)
book_title_entry = tk.Entry(frame, width=30)
book_title_entry.grid(row=2,column=1, pady=20 )
```


Edit Selection View Go Run ···
GROUP ASSIGNMENT IML208
test.py 1
test1.py 1
main_menu.py 1
data_entry.py
submodules.py 1
Kindergarten_Registrtion.py 1
Holiday_Package.py 1
test.py > ...
# Gender Dropdown
book_type_var = tk.StringVar(root)
book_type_var.set("Select Your Book Type")
# Default value before your selection
book_type_dropdown = tk.OptionMenu(root, book_type_var, "Journal", "Encyclopedia","Special Book")
book_type_dropdown.grid(pady=10)
book_price_label=tk.Label(frame, text="Book Price:", font=("Times New Roman",11, "bold"),bg="#BFA493")
book_price_label.grid(row=3, column=0)
book_price_entry = tk.Entry(frame, width=30)
book_price_entry.grid(row=3, column=1, pady=20)
date_borrow_label=tk.Label(frame, text="Date Borrow:",font=("Times New Roman",11, "bold"),bg="#BFA493")
date_borrow_label.grid(row=4, column=0)
date_borrow_entry = tk.Entry(frame, width=30)
date_borrow_entry.grid(row=4, column=1, pady=20)
# Update Button
update_button = tk.Button(root, text="Update",font=("Amasis MT Pro Black",11, "bold"),bg="#A36370", command=enter_data)
update_button.grid(row=5,column=0, padx=10)
# Delete Button
delete_button = tk.Button(root, text="Delete",font=("Amasis MT Pro Black",11, "bold"),bg="#A36370", command=enter_data)
delete_button.grid(row=6,column=0,padx=5, pady=10)
# Submit Button
submit_button = tk.Button(root, text="Submit Form",font=("Amasis MT Pro Black",11, "bold"),bg="#A36370", command=enter_data)
submit_button.grid(row=7)

test.py > ...
# Update data
def edit_data():
book_id = book_id_entry.get()
book_title = book_title_entry.get()
book_type = book_type_var.get()
book_price = book_price_entry.get()
date_borrow= date_borrow_entry.get()
mydb = mysql.connector.connect(
host="localhost",
user="root",
password="",
database="book_rent"
)
mycursor = mydb.cursor()
# Get the total count of rows in the database
mycursor.execute("SELECT COUNT(*) FROM book_entry_form")
count = mycursor.fetchone()[0]
if count > 0:
# Update the last row in the database (book_id, book_title, book_type, book_price, date_borrow)
sql = "UPDATE book_entry_form SET book_id= %s, book_title = %s, book_type = %s, book_price = %s, date_borrow =%s ORDER BY name DESC
val = (book_id, book_title, book_type, book_price, date_borrow)
mycursor.execute(sql, val)
mydb.commit()
print("Record updated!!!")
else:
print("No records update!!!")

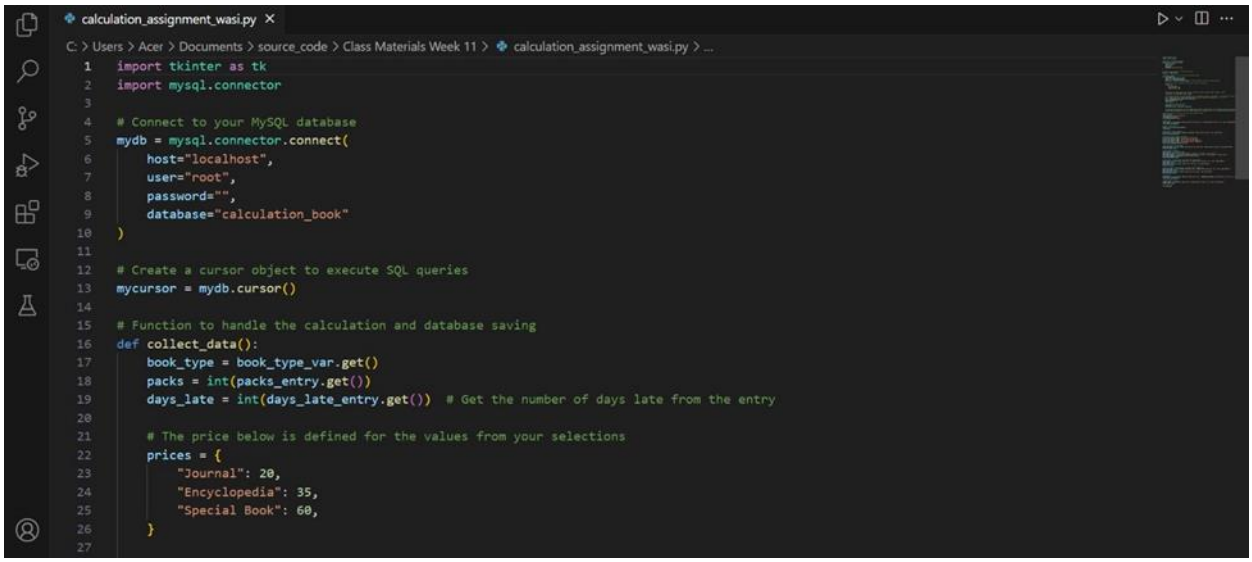
test.py > ...

```python
#Delete data
def delete_data():
    book_id = book_id_entry.get()
    book_title = book_title_entry.get()
    book_type = book_type_var.get()
    book_price = book_price_entry.get()
    date_borrow= date_borrow_entry.get()

    mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="book_rent2"
        )

    mycursor = mydb.cursor()

    # Get the total count of rows in the database
    mycursor.execute("SELECT COUNT(*) FROM book_entry_form")
    count = mycursor.fetchone()[0]

    if count > 0:
        # Delete the last row in the database
        sql = "DELETE FROM book_entry_form ORDER BY name DESC LIMIT 1"
        mycursor.execute(sql)
        mydb.commit()
        print("Last record deleted.")
    else:
        print("No records to delete.")

    mycursor.close()
    mydb.close()
```

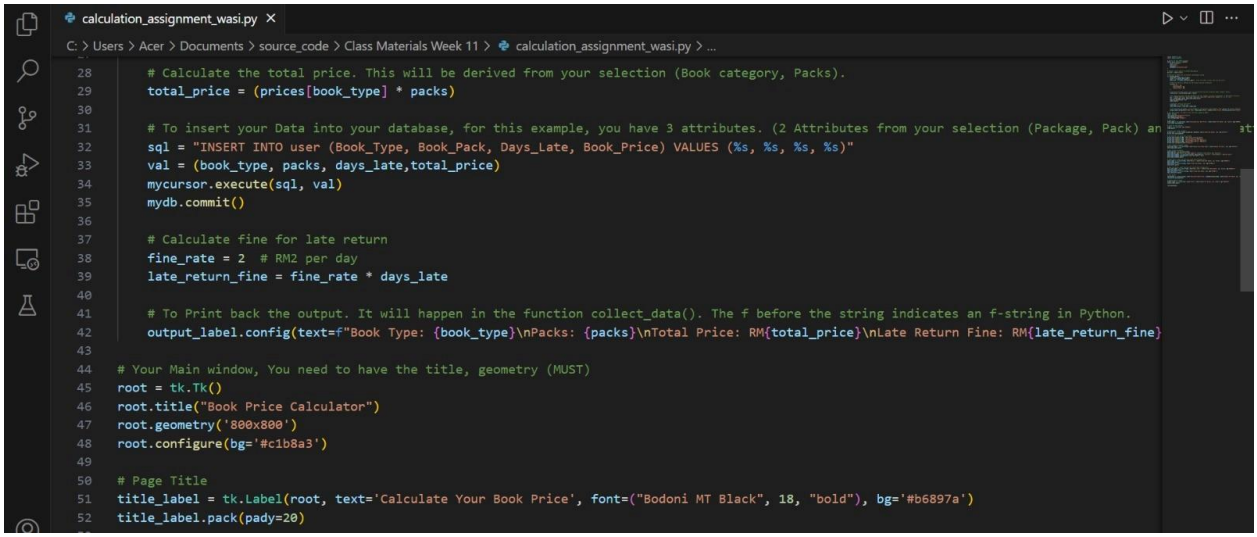
TikTok

```python

    mycursor.close()
    mydb.close()

root.mainloop()

```

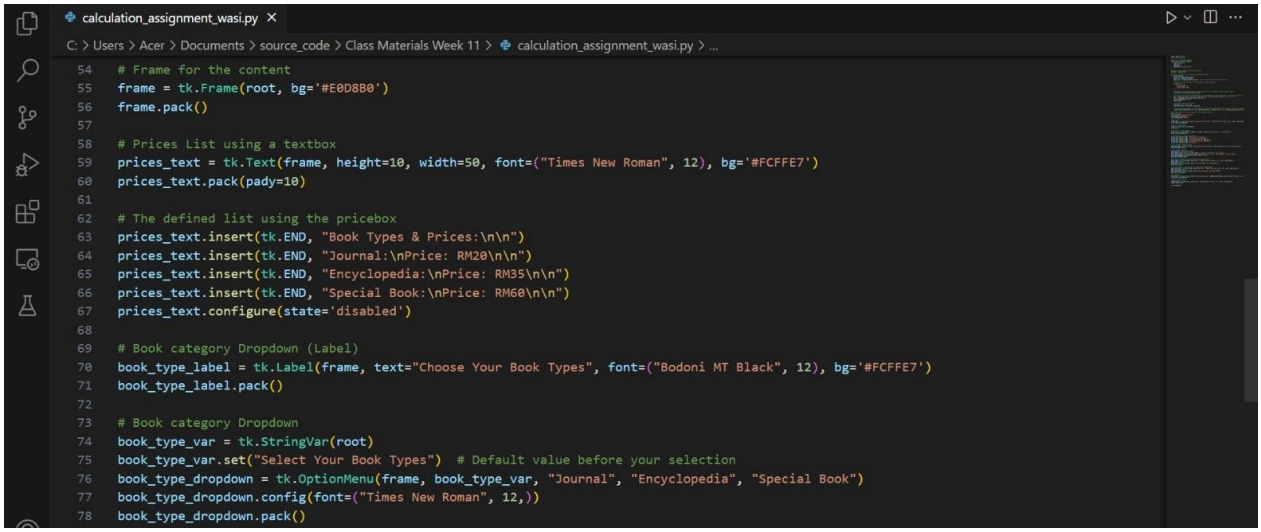
5.3 CODE Calculation of book



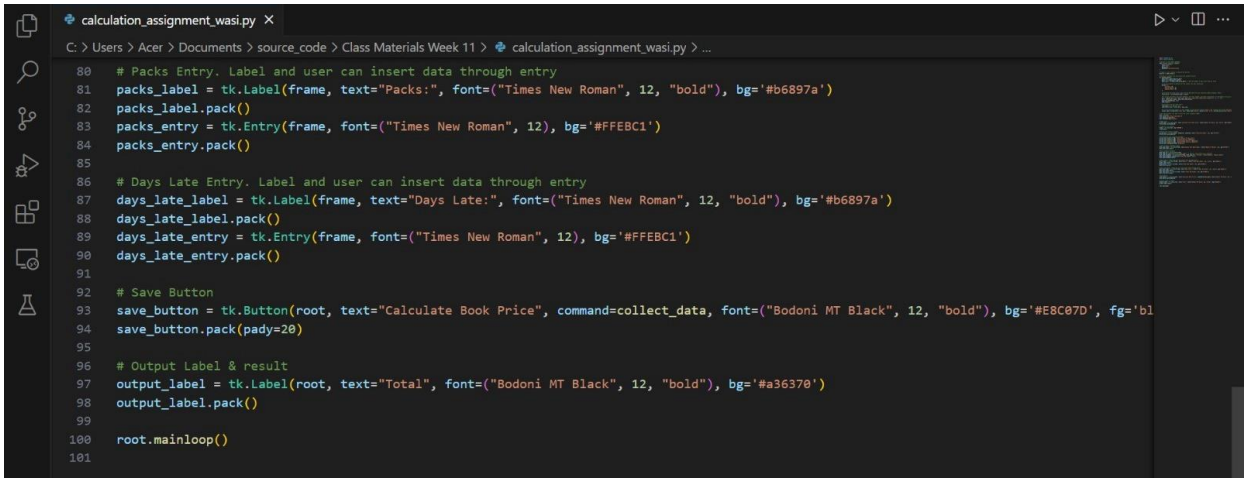
```python
import tkinter as tk
import mysql.connector

# Connect to your MySQL database
mydb = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="calculation_book"
)

# Create a cursor object to execute SQL queries
mycursor = mydb.cursor()

# Function to handle the calculation and database saving
def collect_data():
    book_type = book_type_var.get()
    packs = int(packs_entry.get())
    days_late = int(days_late_entry.get())  # Get the number of days late from the entry

    # The price below is defined for the values from your selections
    prices = {
        "Journal": 20,
        "Encyclopedia": 35,
        "Special Book": 60,
    }
```



```python
    # Calculate the total price. This will be derived from your selection (Book category, Packs).
    total_price = (prices[book_type] * packs)

    # To insert your Data into your database, for this example, you have 3 attributes. (2 Attributes from your selection (Package, Pack) an
    sql = "INSERT INTO user (Book_Type, Book_Pack, Days_Late, Book_Price) VALUES (%s, %s, %s, %s)"
    val = (book_type, packs, days_late,total_price)
    mycursor.execute(sql, val)
    mydb.commit()

    # Calculate fine for late return
    fine_rate = 2  # RM2 per day
    late_return_fine = fine_rate * days_late

    # To Print back the output. It will happen in the function collect_data(). The f before the string indicates an f-string in Python.
    output_label.config(text=f"Book Type: {book_type}\nPacks: {packs}\nTotal Price: RM{total_price}\nLate Return Fine: RM{late_return_fine}

# Your Main window, You need to have the title, geometry (MUST)
root = tk.Tk()
root.title("Book Price Calculator")
root.geometry('800x800')
root.configure(bg='#c1b8a3')

# Page Title
title_label = tk.Label(root, text='Calculate Your Book Price', font=("Bodoni MT Black", 18, "bold"), bg='#b6897a')
title_label.pack(pady=20)
```

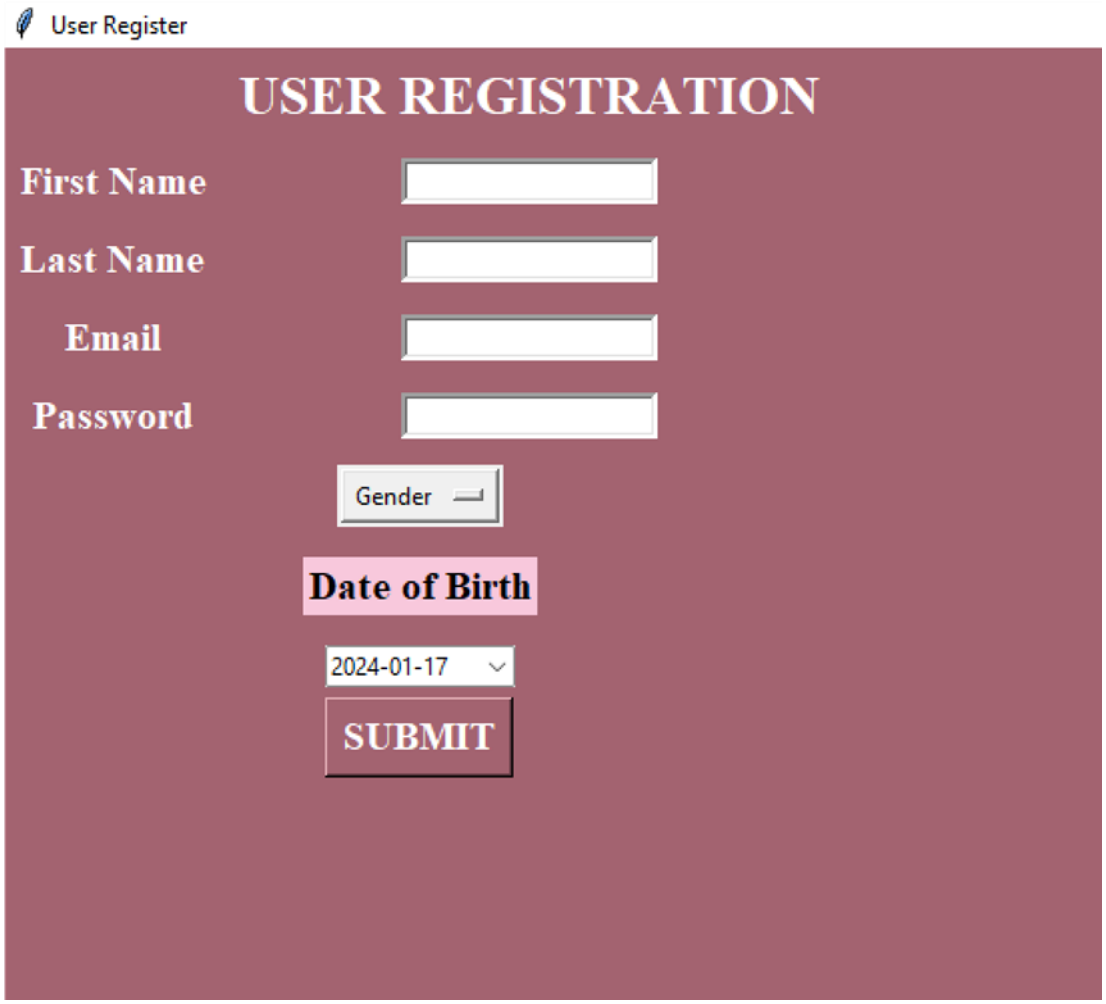


```python
# Frame for the content
frame = tk.Frame(root, bg='#E0D8B0')
frame.pack()

# Prices List using a textbox
prices_text = tk.Text(frame, height=10, width=50, font=("Times New Roman", 12), bg='#FCFFE7')
prices_text.pack(pady=10)

# The defined list using the pricebox
prices_text.insert(tk.END, "Book Types & Prices:\n\n")
prices_text.insert(tk.END, "Journal:\nPrice: RM20\n\n")
prices_text.insert(tk.END, "Encyclopedia:\nPrice: RM35\n\n")
prices_text.insert(tk.END, "Special Book:\nPrice: RM60\n\n")
prices_text.configure(state='disabled')

# Book category Dropdown (Label)
book_type_label = tk.Label(frame, text="Choose Your Book Types", font=("Bodoni MT Black", 12), bg='#FCFFE7')
book_type_label.pack()

# Book category Dropdown
book_type_var = tk.StringVar(root)
book_type_var.set("Select Your Book Types")  # Default value before your selection
book_type_dropdown = tk.OptionMenu(frame, book_type_var, "Journal", "Encyclopedia", "Special Book")
book_type_dropdown.config(font=("Times New Roman", 12,))
book_type_dropdown.pack()
```

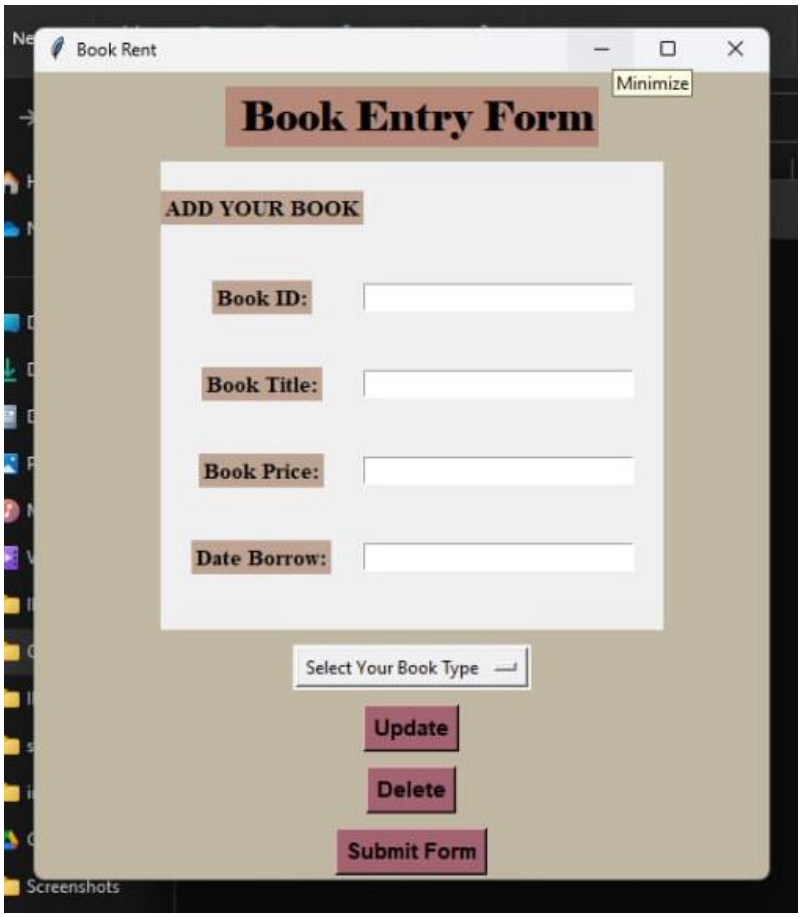

calculation_assignment_wasi.py
C: > Users > Acer > Documents > source_code > Class Materials Week 11 > calculation_assignment_wasi.py > ...
80 # Packs Entry. Label and user can insert data through entry
81 packs_label = tk.Label(frame, text="Packs:", font=("Times New Roman", 12, "bold"), bg='#b6897a')
82 packs_label.pack()
83 packs_entry = tk.Entry(frame, font=("Times New Roman", 12), bg='#FFEBC1')
84 packs_entry.pack()
85
86 # Days Late Entry. Label and user can insert data through entry
87 days_late_label = tk.Label(frame, text="Days Late:", font=("Times New Roman", 12, "bold"), bg='#b6897a')
88 days_late_label.pack()
89 days_late_entry = tk.Entry(frame, font=("Times New Roman", 12), bg='#FFEBC1')
90 days_late_entry.pack()
91
92 # Save Button
93 save_button = tk.Button(root, text="Calculate Book Price", command=collect_data, font=("Bodoni MT Black", 12, "bold"), bg='#E8C07D', fg='bl
94 save_button.pack(pady=20)
95
96 # Output Label & result
97 output_label = tk.Label(root, text="Total", font=("Bodoni MT Black", 12, "bold"), bg='#a36370')
98 output_label.pack()
99
100 root.mainloop()
101

6.0 SNAPSHOT OF PROJECT (GUI)

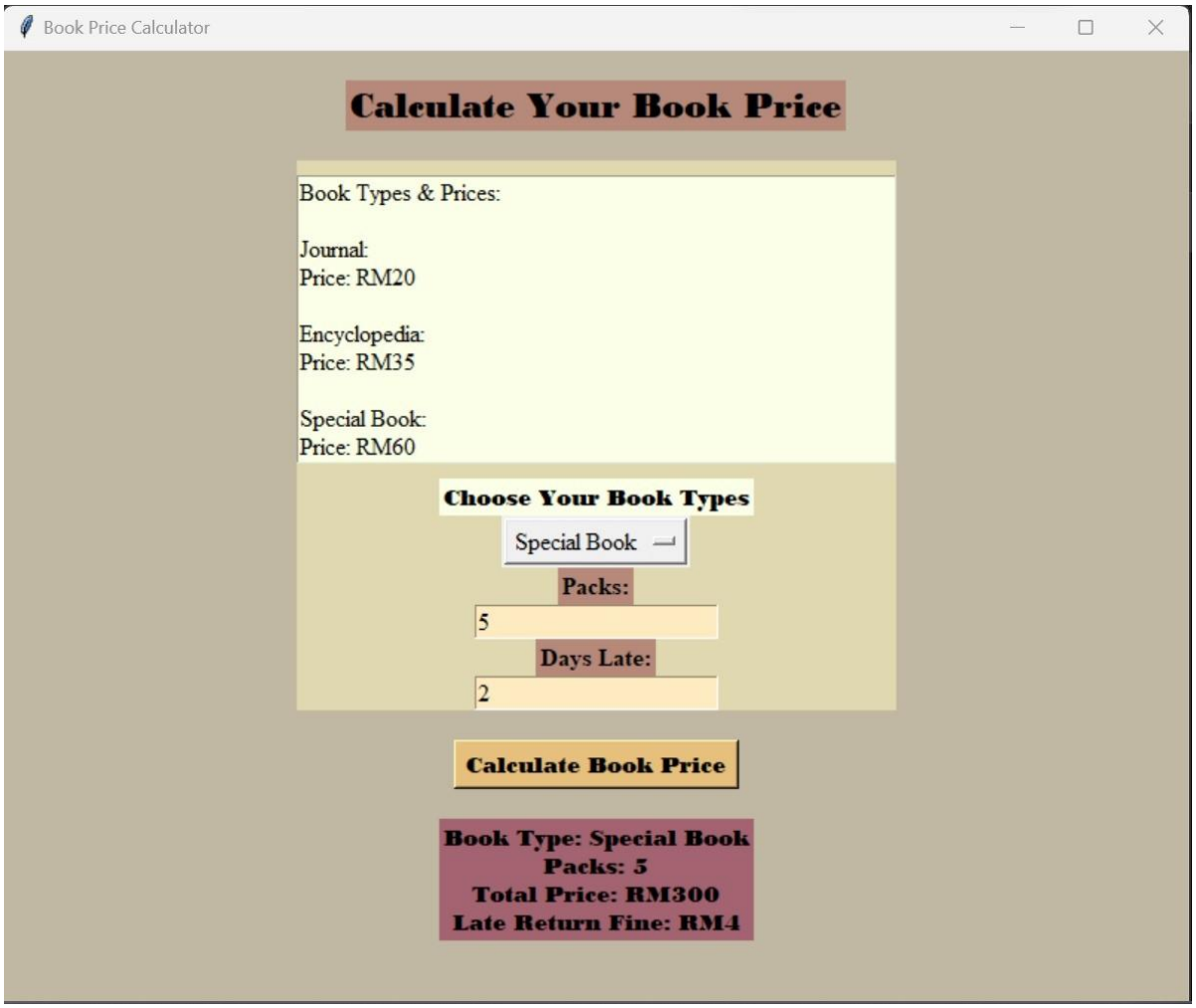
6.1 GUI User Registration



6.2 GUI Book Entry

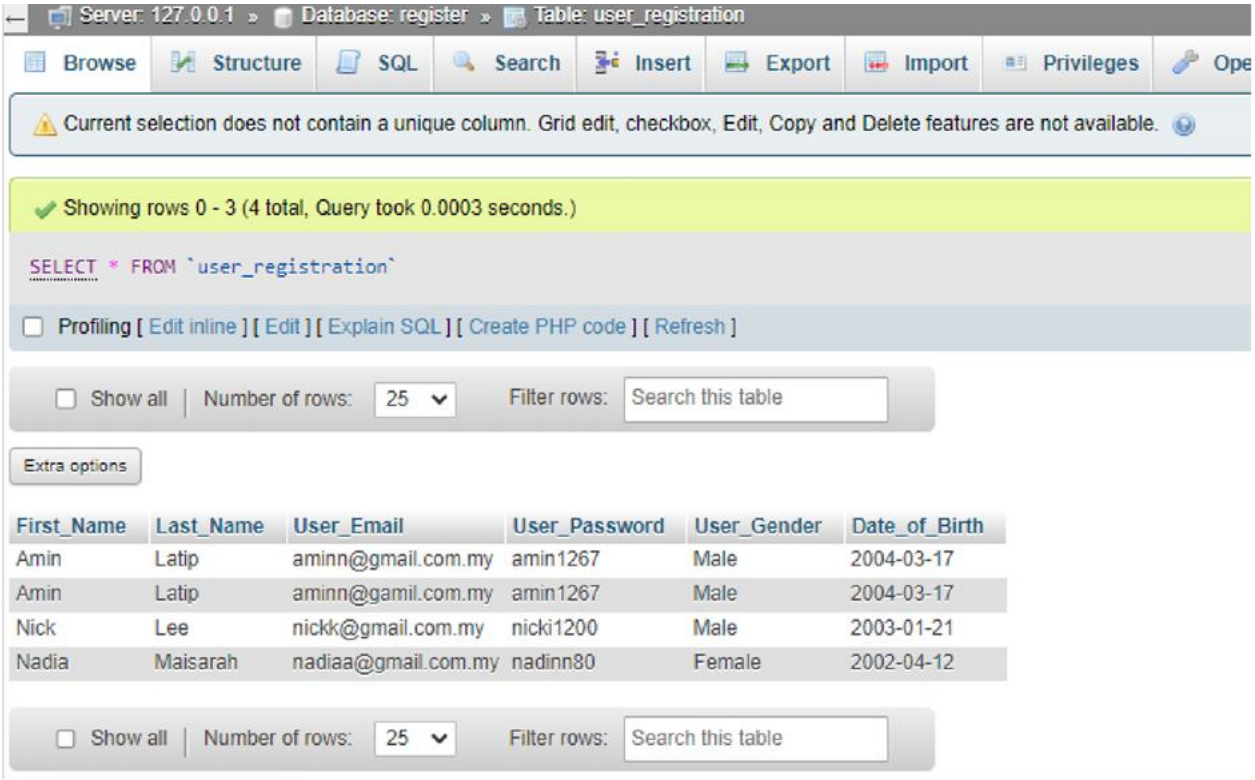


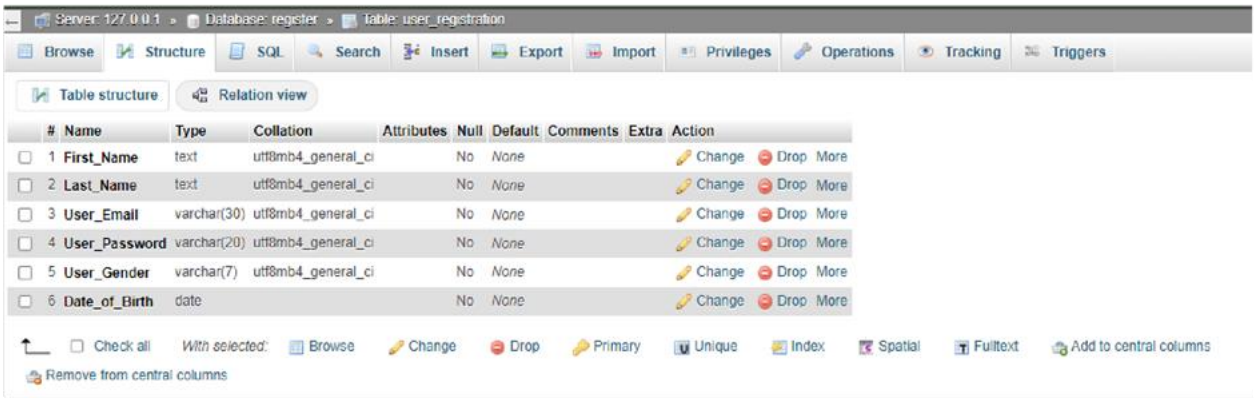
6.3 GUI Calculation of book

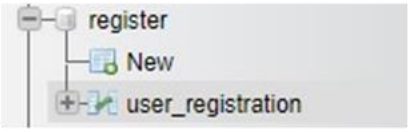


7.0 SNAPSHOT OF DATABASE (XAMPP)

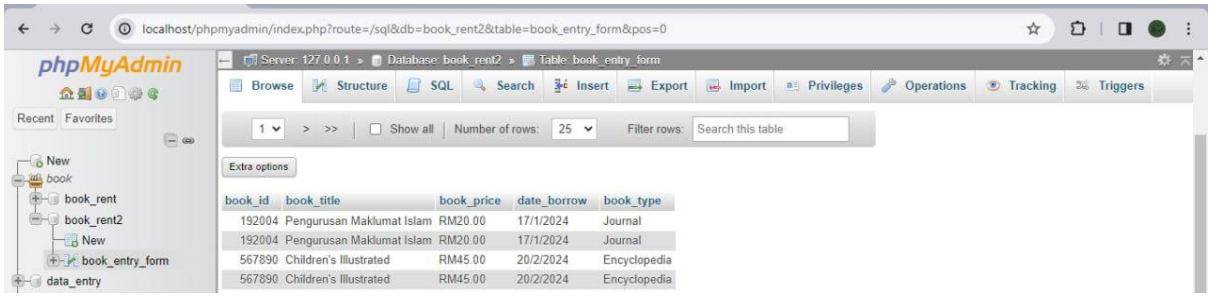
7.1 Database (XAMPP) User Registration



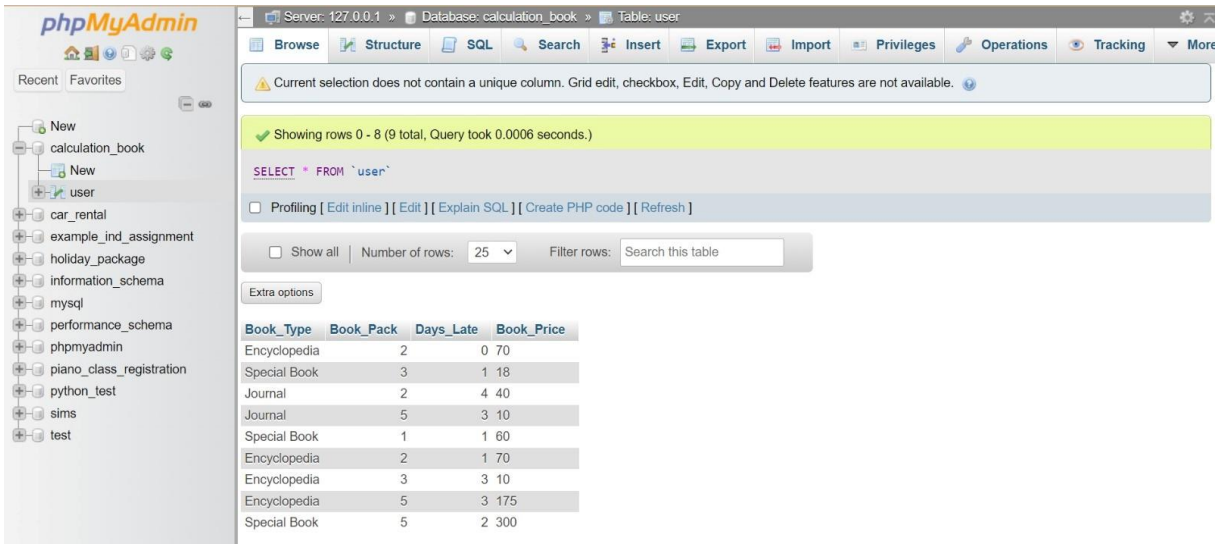


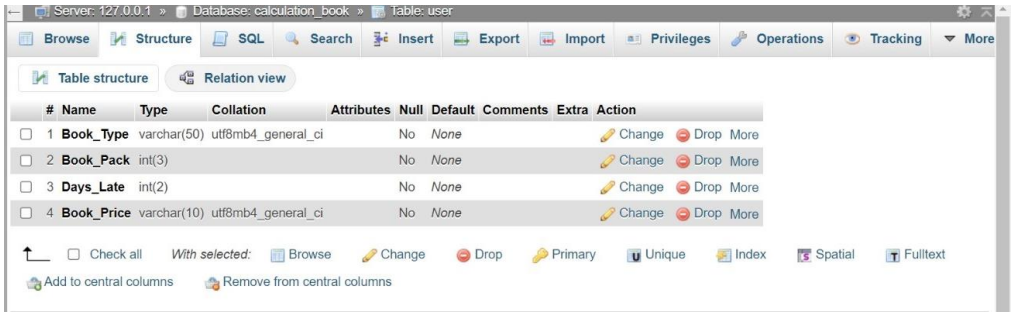


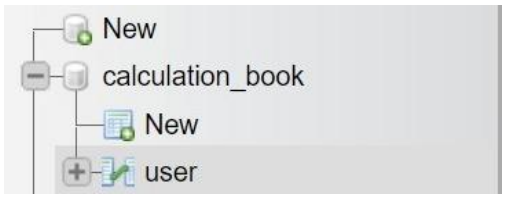
7.2 Database (XAMPP) Book Entry



7.3 Database (XAMPP) Calculation of book







8.0 CONCLUSION

The purpose of the rental book code system is to manage the book rentals in a bookstore or library. Users can go through the books that are offered, rent, return, and see their rental history.

The system is made up of various essential parts such as book class, user class, library class and main application. Book class can holds information on a book, including its title, author, genre, and availability. It also contains ways to show book details and adjust the availability status. Other than that, user class can holds information about a user, including name, ID, and renting history. It provides instructions on how to check rental history and return and rent books. Besides, library class oversees the entire system of rentals. It keeps track of a list of registered users, a list of books that are available, and a list of books that are rented. It provides tools for managing book rents and returns, adding books, registering users, and producing reports on past rentals. Lastly, main application offers a user interface through which users can interact with the system and makes use of the Library class. Options to peruse books, look up specific books, rent, return, and view rental history are possible.

Additional features like user identification, fine computation for overdue books, and book reservations can improve the system even more. All things considered, the rental book code system streamlines the management of book rentals, increasing its effectiveness and convenience for patrons as well as library employees.