**Comp 3005 Fall 2022**

**Project Report**

Completed individually by:

Catherine Li

student ID:101209708

Table of Content

[Conceptual Design 3](#_Toc120991493)

[Reduction to Relation Schemas 4](#_Toc120991494)

[Normalization of Relation Schemas 5](#_Toc120991495)

[Database Schema Diagram 6](#_Toc120991496)

[Implementation 7](#_Toc120991497)

[Bonus Features 13](#_Toc120991498)

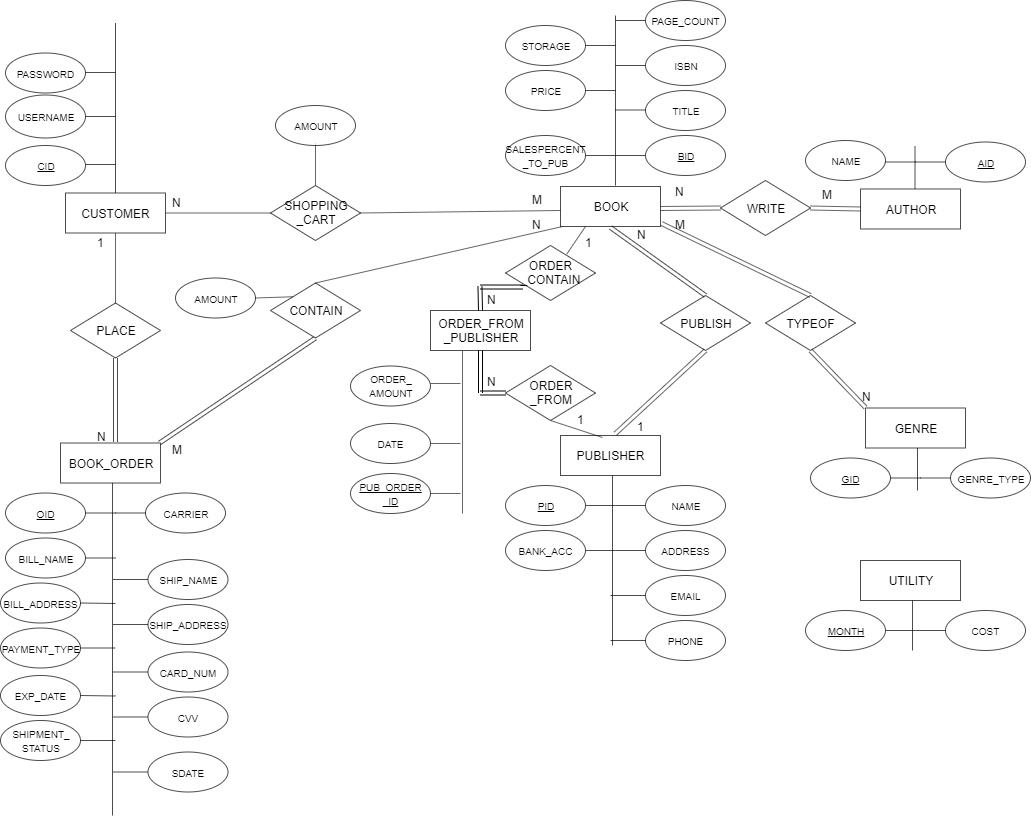
[GitHub Repository 13](#_Toc120991499)

[Appendix I 13](#_Toc120991500)

[Appendix II 13](#_Toc120991501)

# Conceptual Design

ER-diagram:



Assumptions regarding the ER diagram:

1. assume each book is published by only one publisher, one publisher can publish many books.
2. assume books may have same title
3. assume the ISBN of book is unique and the ISBN is 13 digits, stored as varchar type
4. assume the address in the database is stored within one attribute as varchar type
5. assume all publisher stored in database must publish a book
6. assume all author stored in database must write a book
7. assume authors could not have the same name
8. assume one author can write many books, one book can have more than one author.
9. assume publisher have unique name, but could have same bank account, same address, same phone and email (owed by same boss/person)
10. assume customer username must be unique

Other assumptions:

1. assume utility entity is updated every year, that is, database only save the utility for the current year
2. assume the bookstore opened the date of 2022/01/01
3. assume only the owner can only review the reports in the current year

# Reduction to Relation Schemas

PUBLISHER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| PID | BANK\_ACC | NAME | ADDRESS | EMAIL | PHONE |

BOOK

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| BID | TITLE | ISBN | PAGE\_COUNT | STORAGE | PRICE | SALESPERCENT\_TO\_PUB | PID |

GENRE

|  |  |
| --- | --- |
| GID | GENRE\_TYPE |

TYPEOF

|  |  |
| --- | --- |
| GID | BID |

AUTHOR

|  |  |
| --- | --- |
| AID | NAME |

WRITE

|  |  |
| --- | --- |
| AID | BID |

CUSTOMER

|  |  |  |
| --- | --- | --- |
| CID | USERNAME | PASSWORD |

SHOPPING\_CART

|  |  |  |
| --- | --- | --- |
| CID | BID | AMOUNT |

BOOK\_ORDER

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| OID | CARRIER | SHIP\_NAME | SHIP\_ADDRESS | BILL\_NAME | BILL\_ADDRESS | CARD\_NUM |
| CVV | SDATE | EXP\_DATE | PAYMENT\_TYPE | SHIPMENT\_STATUS | CID |  |

CONTAIN

|  |  |  |
| --- | --- | --- |
| OID | BID | AMOUNT |

ORDER\_FROM\_PUBLISHER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PUB\_ORDER\_ID | PID | BID | DATE | ORDER\_AMOUNT |

UTILITY

|  |  |
| --- | --- |
| MONTH | COST |

# Normalization of Relation Schemas

Here I denote set of functional dependencies as F for each schema

PUBLISHER TABLE:

F={PID -> BANK\_ACC, NAME, ADDRESS, EMAIL, PHONE

NAME - > BANK\_ACC, PID, ADDRESS, EMAIL, PHONE}

pid is superkey for publisher table, name is superkey for publisher table. Therefore, BCNF holds.

BOOK TABLE:

F={BID ->TITLE, ISBN, PAGE\_COUNT, STORAGE, PRICE, SALESPERCENT\_TO\_PUB, PID

ISBN -> TITLE, BID, PAGE\_COUNT, STORAGE, PRICE, SALESPERCENT\_TO\_PUB, PID}

BID is superkey for BOOK table, ISBN is superkey for BOOK table. Therefore, BCNF holds.

GENRE TABLE

F={GID -> GENRE\_TYPE

GENRE\_TYPE->GID}

GID is superkey for GENRE table, GENRE\_TYPE is superkey for GENRE table. Therefore, BCNF holds.

TYPEOF TABLE

F=, functional dependencies is empty for TYPEOF table, BCNF holds

AUTHOR TABLE

F={AID -> NAME}

F={ NAME ->AID}

AID is superkey for AUTHOR table. NAME is superkey for AUTHOR table. Therefore, BCNF holds

WRITE TABLE

F=, functional dependencies is empty for WRITE table, BCNF holds

CUSTOMER

F={CID -> USERNAME, PASSWORD

USERNAME -> CID, PASSWORD}

CID is superkey for CUSTOMER table, USERNAME is superkey for CUDTOMER table. Therefore, BCNF holds.

SHOPPING\_CART TABLE

F={CID, BID -> AMOUNT}

(CID,BID) is superkey for SHOPPING\_CART table. Therefore, BCNF holds.

BOOK\_ORDER TABLE

F={OID -> CARRIER,SHIP\_NAME, SHIP\_ADDRESS, BILL\_NAME, BILL\_ADDRESS, CARD\_NUM,

CVV, SDATE, EXP\_DATE, PAYMENT\_TYPE, SHIPMENT\_STATUS, CID

CARD\_NUM -> CVV, EXP\_DATE, PAYMENT\_TYPE, BILL\_NAME, BILL\_ADDRESS}

OID is superkey for BOOK\_ORDER table.

However, CARD\_NUM+ = CVV, EXP\_DATE, PAYMENT\_TYPE, BILL\_NAME, BILL\_ADDRESS

We know that CARD\_NUM is not superkey for BOOK\_ORDER table. This table needs to be decomposed.

Let R be the old table book\_order

Let result = R, α = CARD\_NUM, *β =* CVV, EXP\_DATE, PAYMENT\_TYPE, BILL\_NAME, BILL\_ADDRESS

*FIRST LOOP:*

*result* = (*R – β*) ∪ (α, *β ) =*

*(*OID, CARRIER, SHIP\_NAME, SHIP\_ADDRESS, SDATE, SHIPMENT\_STATUS, CID, CARD\_NUM) ∪ (CARD\_NUM, CVV, EXP\_DATE, PAYMENT\_TYPE, BILL\_NAME, BILL\_ADDRESS)

Check if (*R – β*) in BCNF:

Since OID -> CARRIER, SHIP\_NAME, SHIP\_ADDRESS, SDATE, SHIPMENT\_STATUS, CID, CARD\_NUM,

OID is the superkey of (*R – β*). Therefore, (*R – β*) is in BCNF

So, BOOK\_ORDER TABLE is decomposed to

BOOK\_ORDER:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| OID | CARRIER | SHIP\_NAME | SHIP\_ADDRESS | CARD\_NUM | SDATE | SHIPMENT\_STATUS | CID |

PAYMENT\_CARD:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CARD\_NUM | CVV | EXP\_DATE | PAYMENT\_TYPE | BILL\_NAME | BILL\_ADDRESS |

CONTAIN TABLE

F={OID,BID->AMOUNT}

(OID,BID) is superkey for CONTAIN table. Therefore, BCNF holds.

ORDER\_FROM\_PUBLISHER TABLE

F={ PUB\_ORDER\_ID->PID, BID, DATE, ORDER\_AMOUNT}

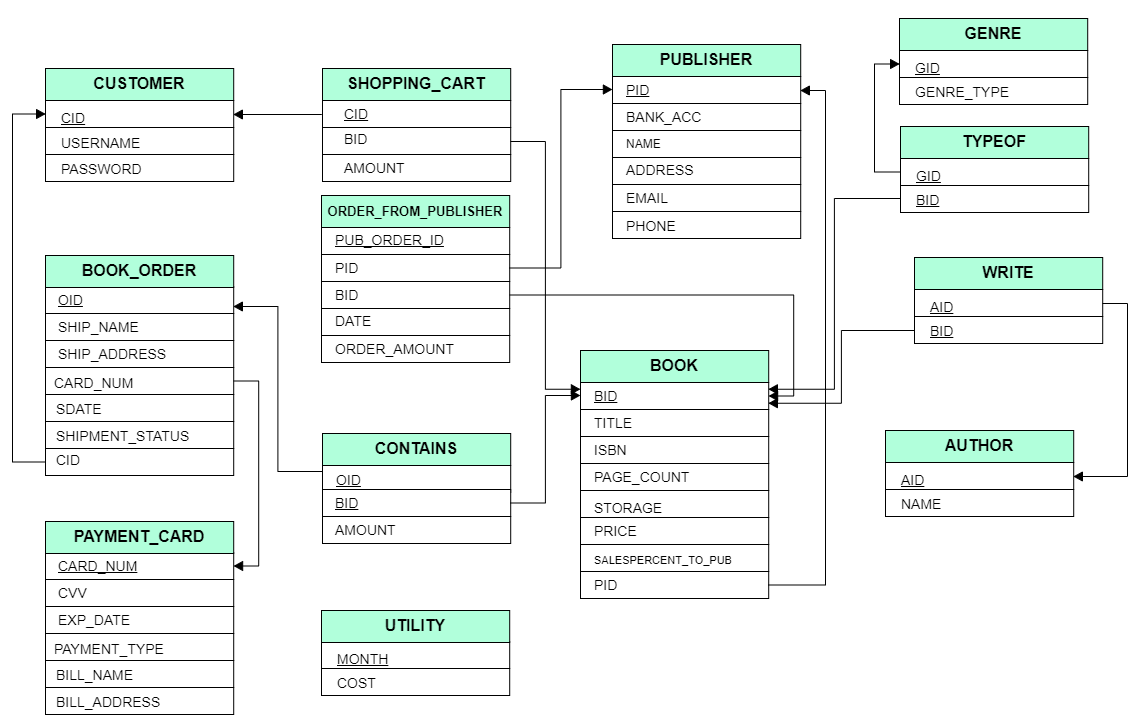
PUB\_ORDER\_ID is superkey for CONTAIN table. Therefore, BCNF holds.

UTILITY TABLE

F={MONTH -> COST}

MONTH is superkey for UTILITY table. Therefore, BCNF holds.

# Database Schema Diagram



# Implementation

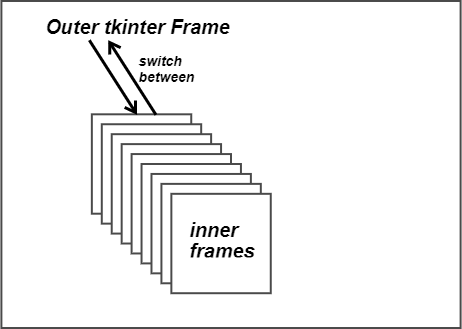
This program is written in Python, tkinter is used to build up GUIs, postgresql is the choice of databse.

This program mainly contains two modules: dbinitializer.py and app.py

The database is connected to an online PostgreSQL database: <https://www.elephantsql.com/>

**dbinitializer.py** builds up the database using DDL and input the initial data into database including 606 books information download from online database, 255 fake publishers’ information, 500 fake customer and 3000 fake book orders.

**app.py** contains the structure of the desktop app:



It contains one outer frame class and 15 inner frame classes:

Outer frame class: BookStoreAppGUI

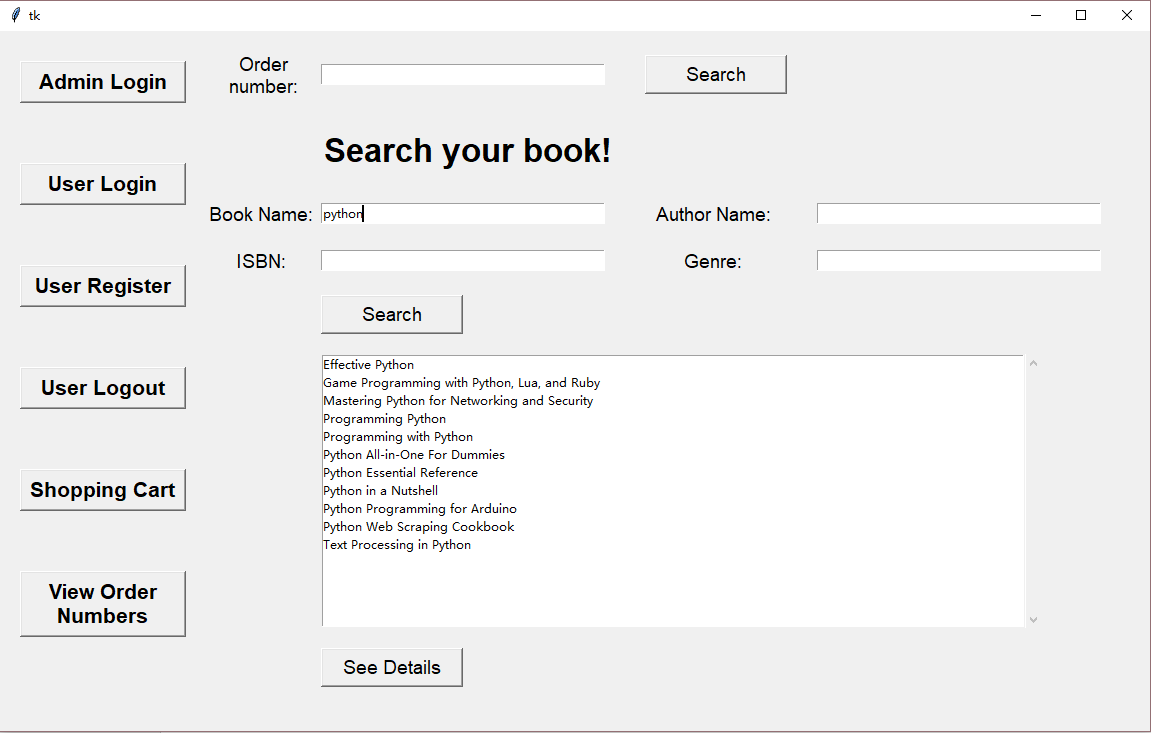
This class builds up the outer frame of the app. It is passed as parent frame to other class. By controlling the swicth\_frame() function in this class, we can go to any pages(these pages are built as inner frames).

Page class (inner frame class):

1. adminLoginPage: log in page for administration / the bookstore owner. After logging in through this page, owner will jump to the admin interface.
2. UserLoginPage: log in page for customer, after logging in, customer can add books to shopping cart and check out
3. UserRegPage: register page for customer
4. FirstPage: First Pages for both customer and owner. Owner can go to admin login page from this page.

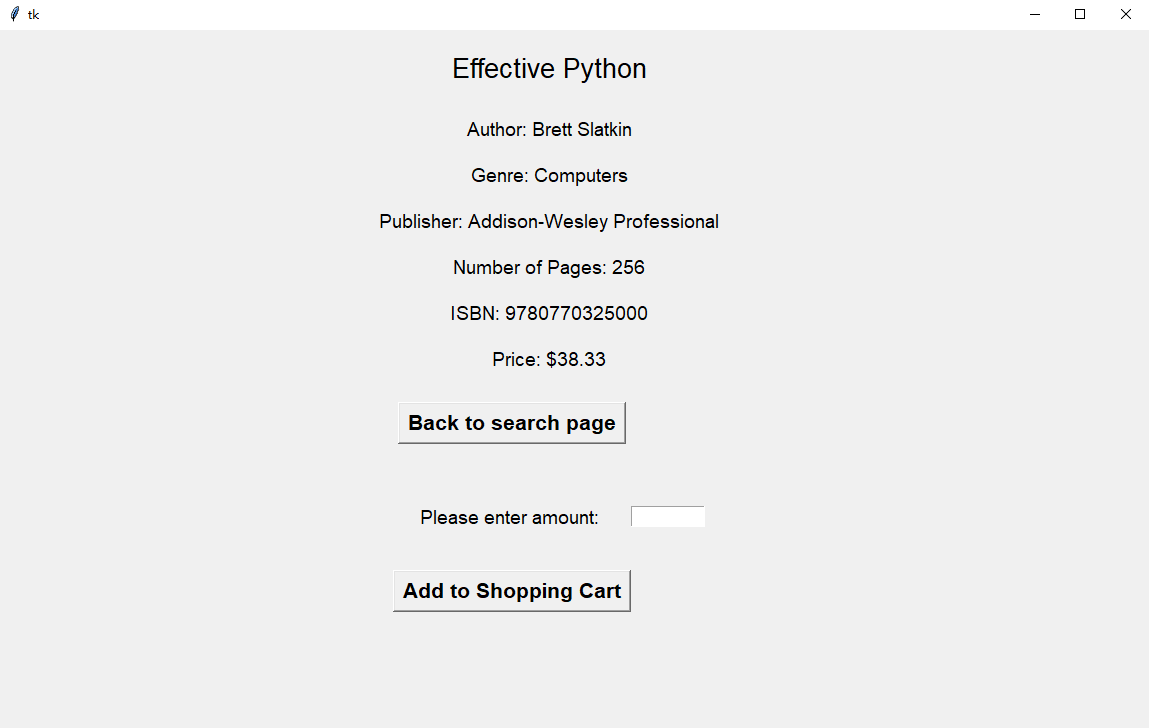
Customer can login/register from this page. Also, customer can search books in this page. The author name and genre should be enter only one name or genre type each time.

Customer can also search orders by order numbers after logging in.

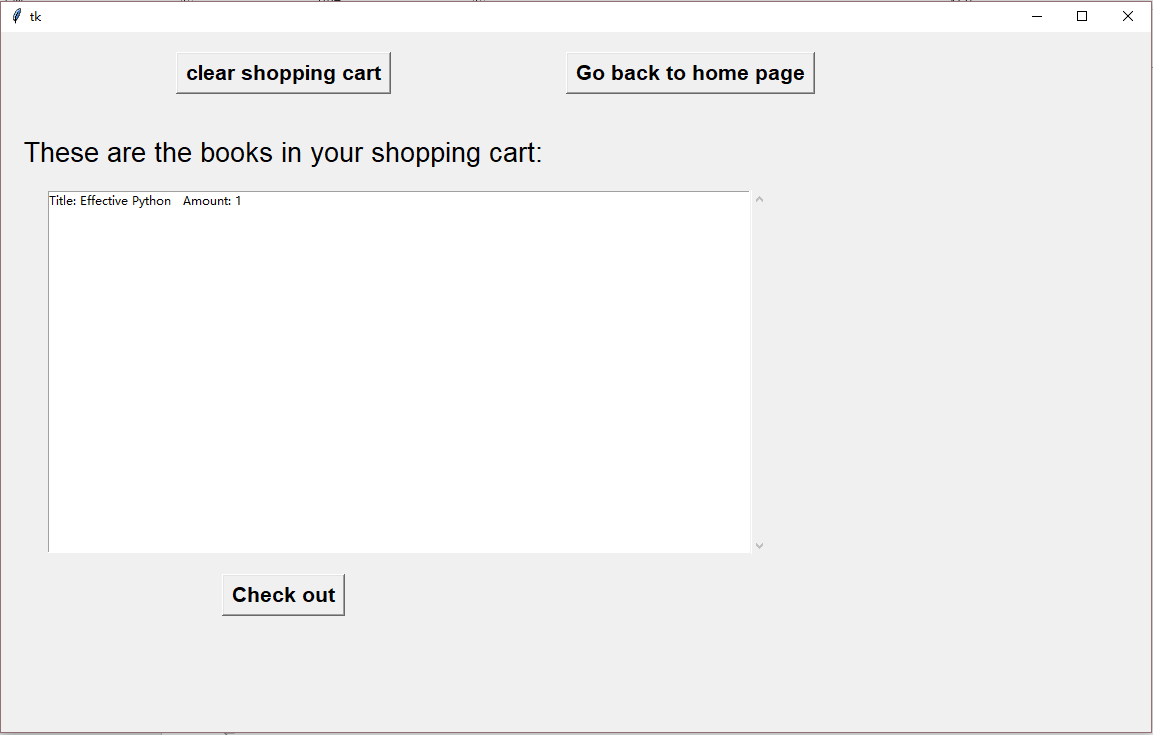


**Customer interface:**

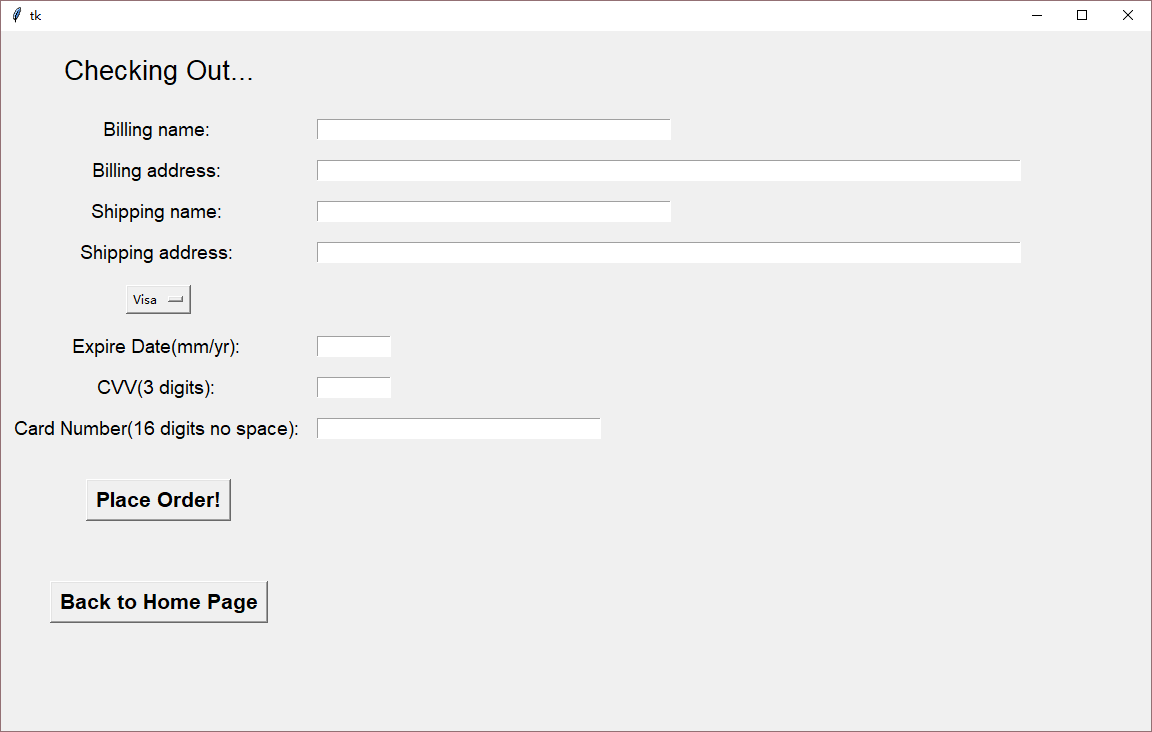
1. BookPage: shows the details of a book, customer can add this book to shopping cart after logging in



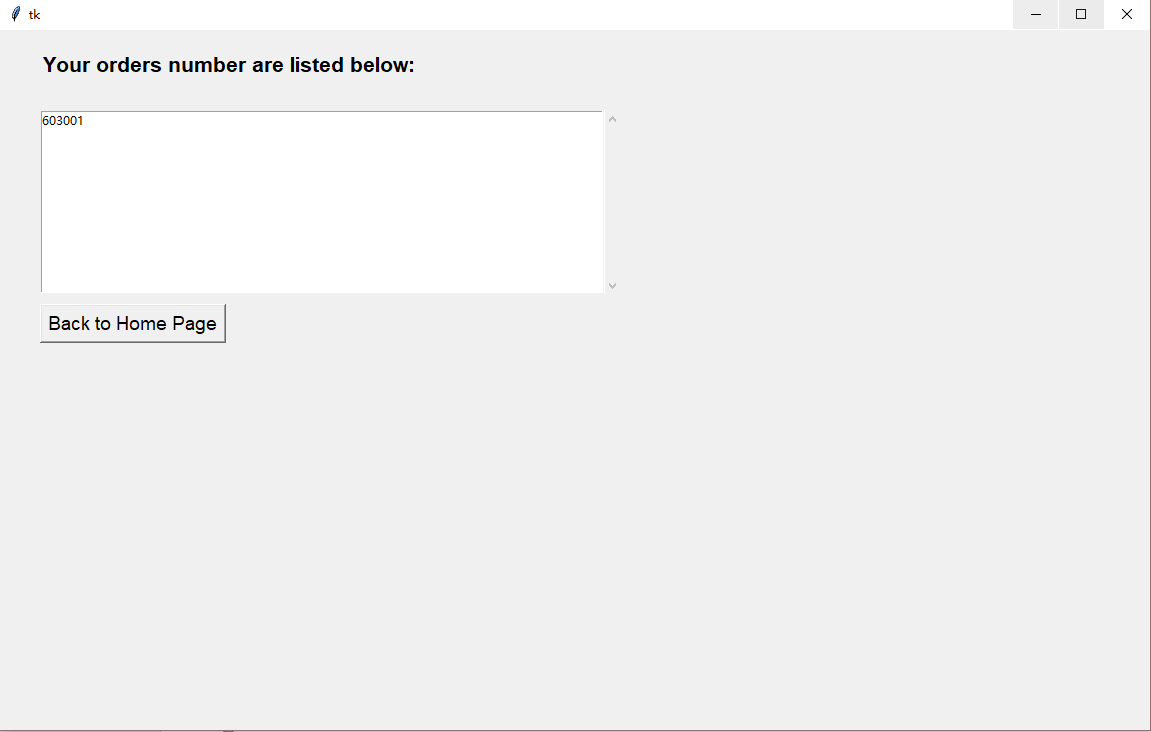
1. CartPage: shows all the books in customer’s shopping cart, check out button redirect to check out page



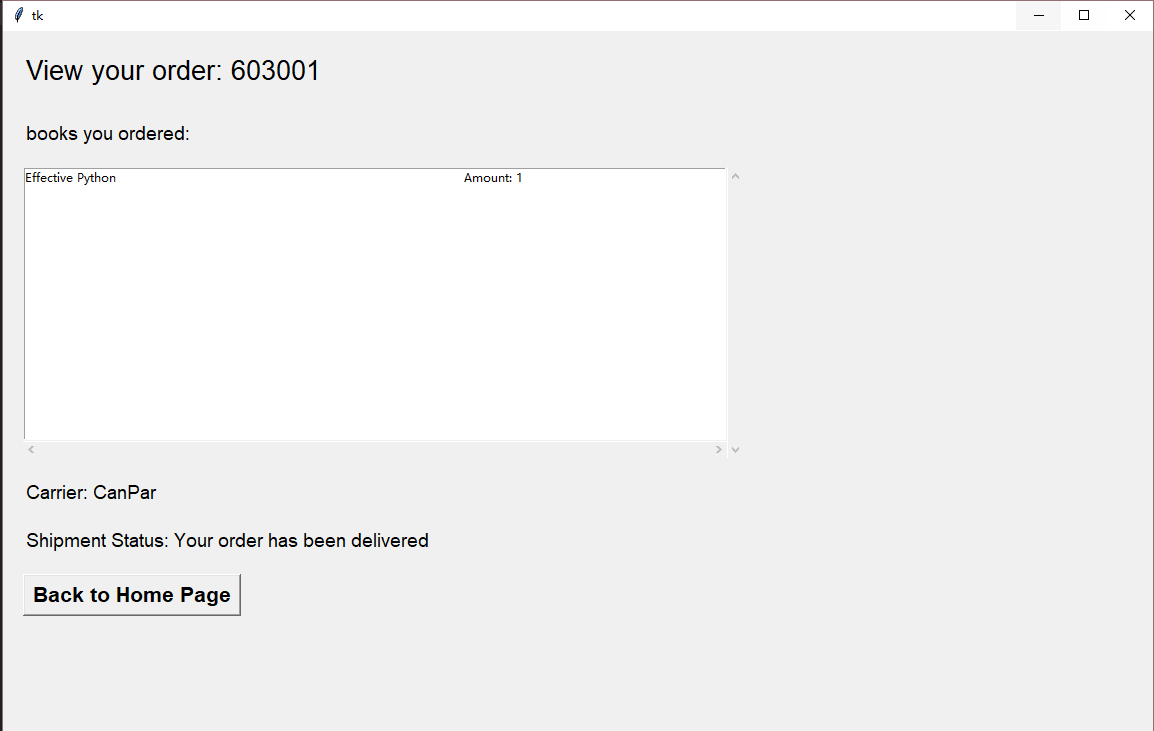
7. CheckoutPage: customer enter bill/ship information and place order in this page, after clicking place order button, an order number will be given.

****

8. GetOrderNumberPage: If the customer forgot their order number, they can click the view order numbers button in the first page and view all the history order numbers in this page

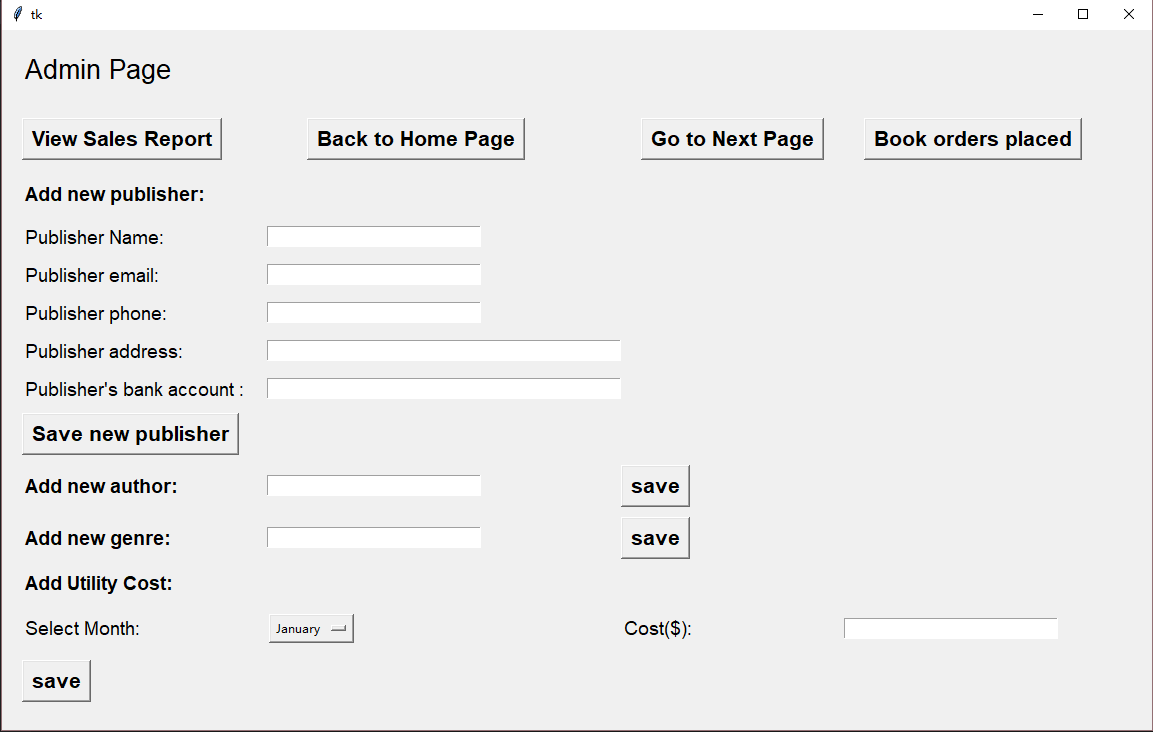
****

9. OrderPage: shows the details of the order number searched.

****

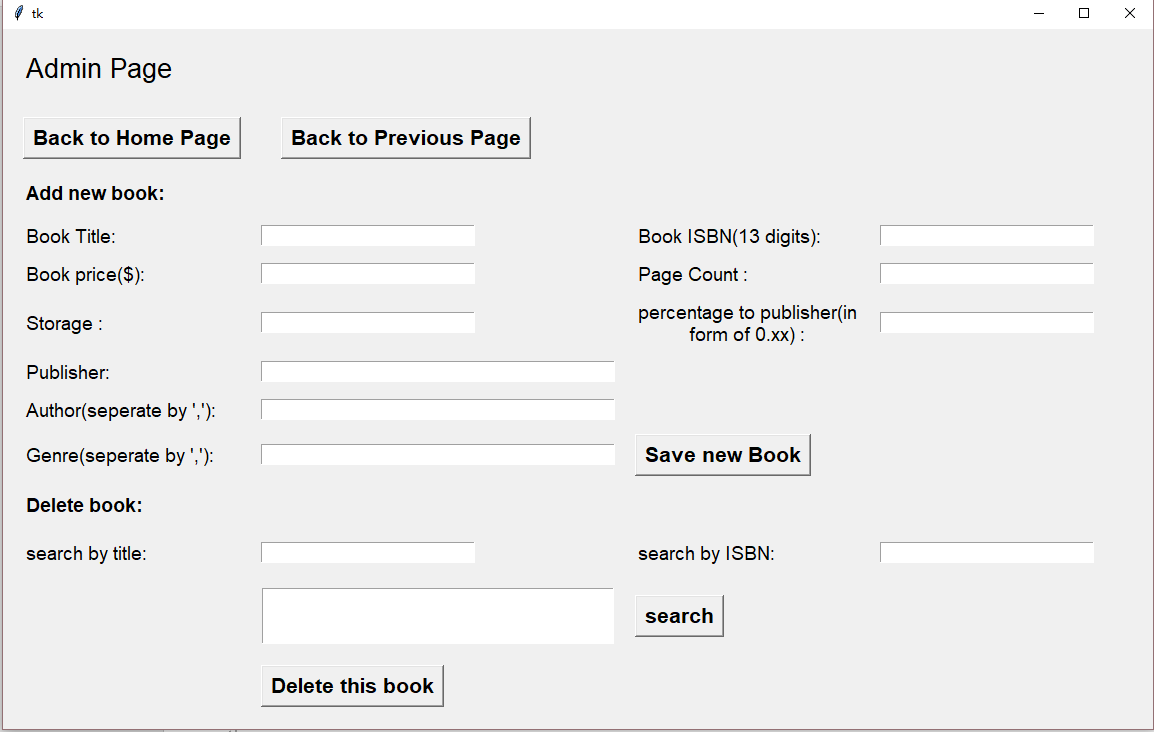
**Admin interface：**

10. adminFirstPage: Accessed after completing the admin login page.

****

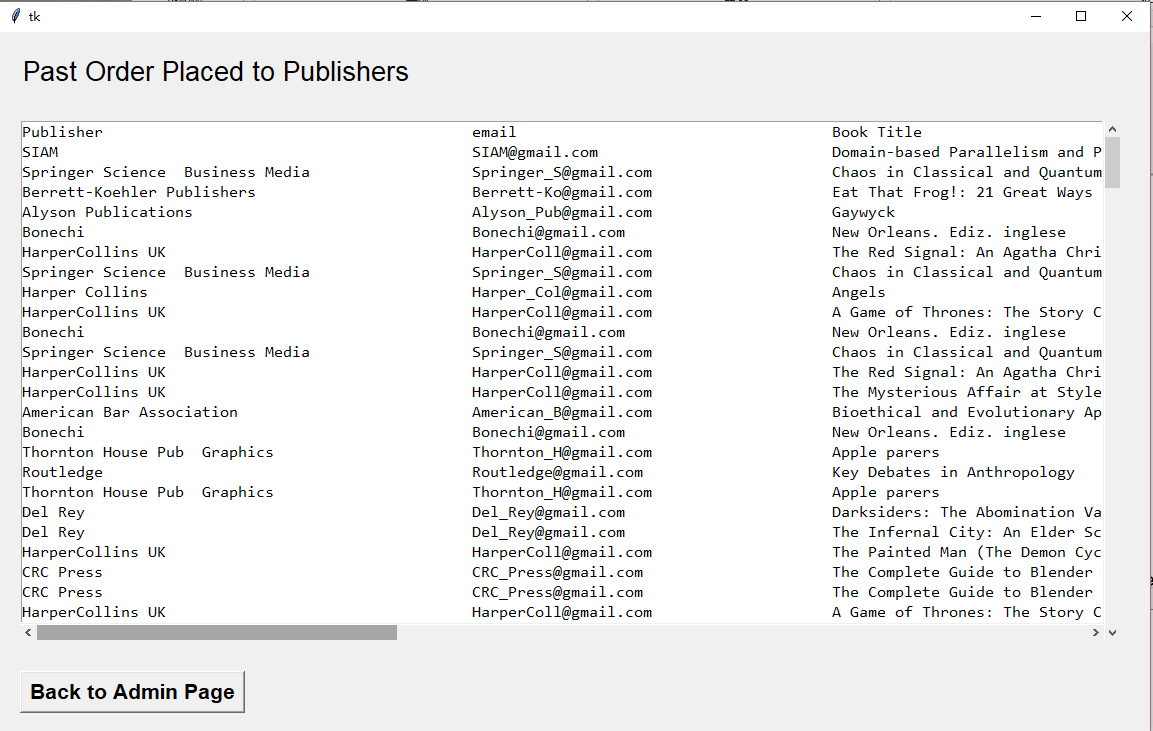
Owners can add new publisher, new author, new genre, and record utility cost in this page. Owner must add single author name and single genre type per time.

1. adminSecondPage：owners can add/delete books in this page. Owner can enter multiple authors, genres (separate them with comma), the percentage sent to publisher must entered as decimal number, if the publisher is not added, owner must go to adminFirstPage and add the full information of the publisher, if the author or genre are not added, the system will add the author and genre automatically in this page



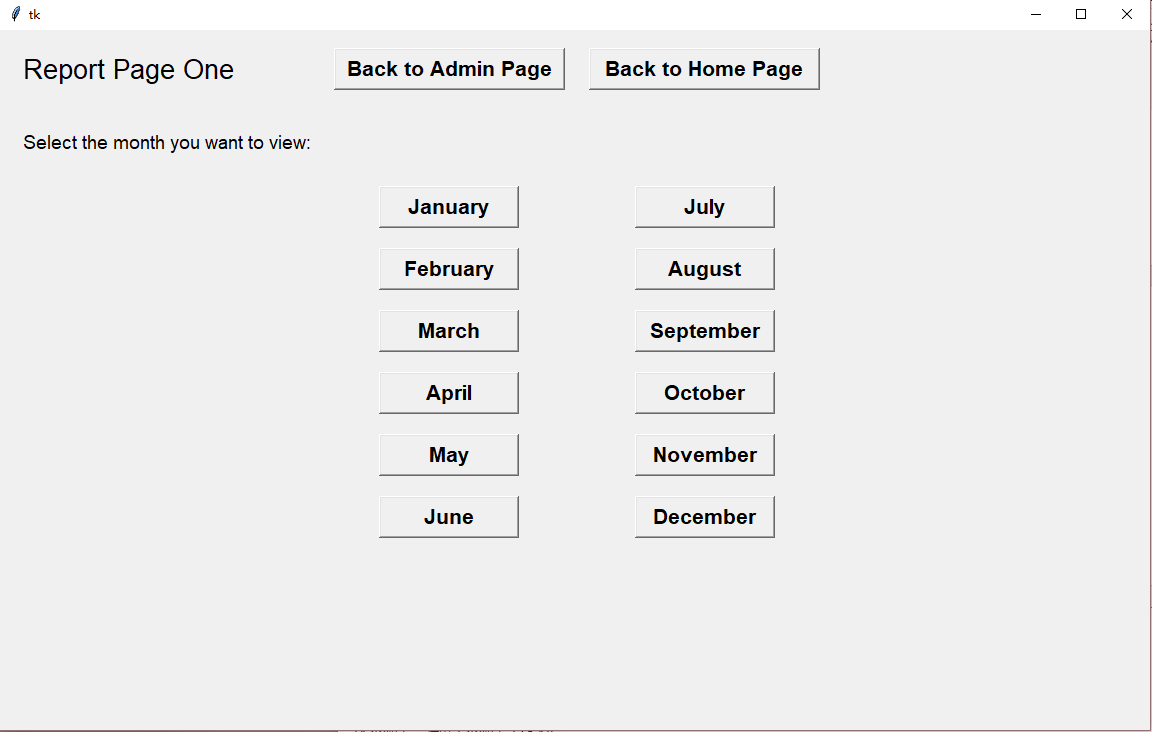
1. OrderfromPubPage: The page is accessed by the ‘books order placed’ button in the admin first page.

It shows the previous order automatically placed by the bookstore when the storage of a book is less than 20. The email sending component is not implemented in this project, but the information used to write to email is stored in the database and shown as record in this table.

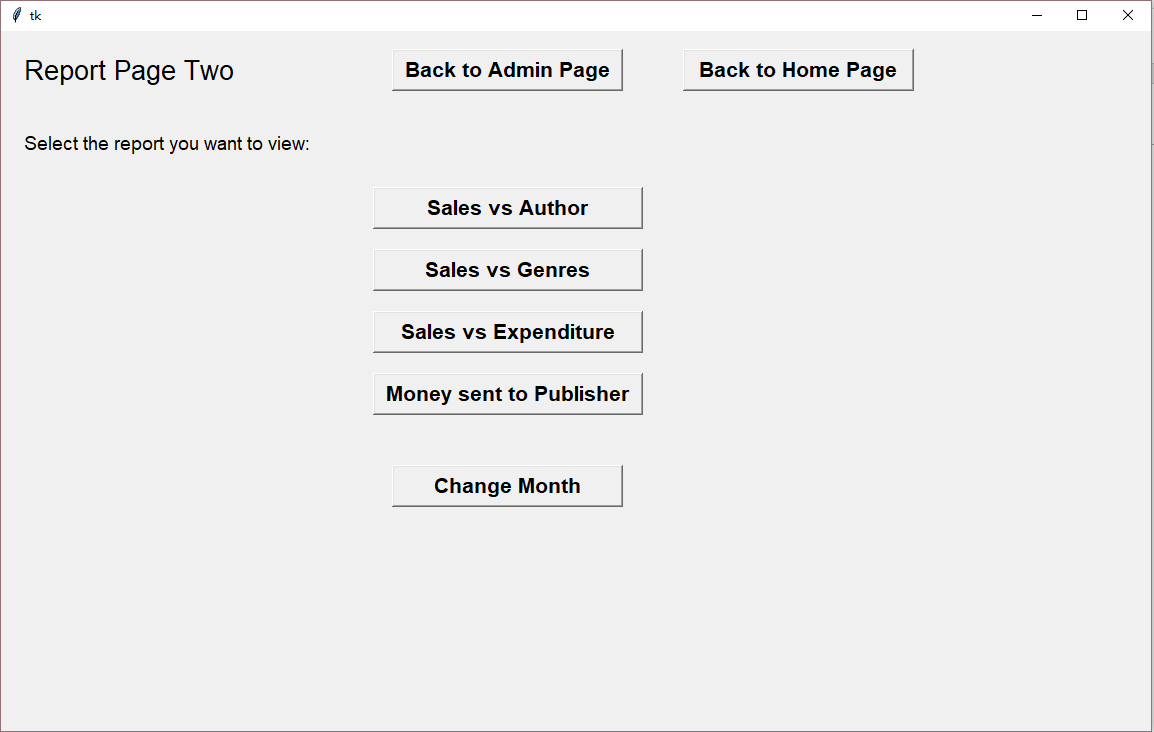


1. reportFirstPage: This page is accessed by clicking the view sales report button in the first page.

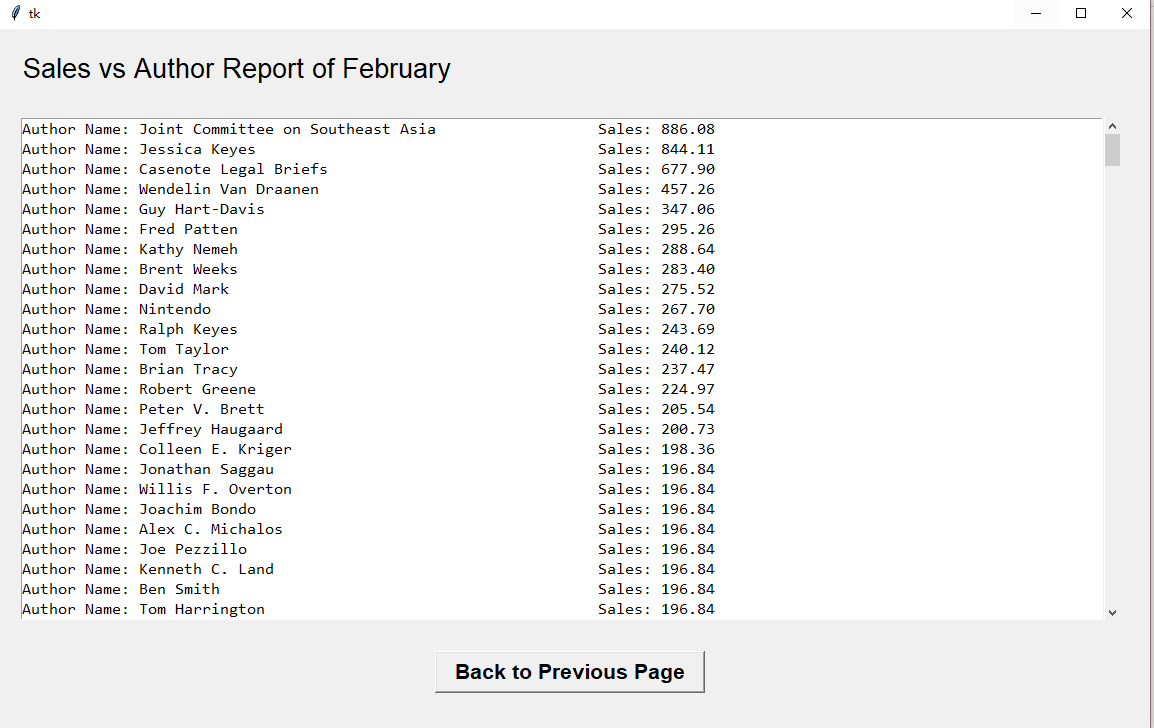
Owner can select the month of report they want to view.

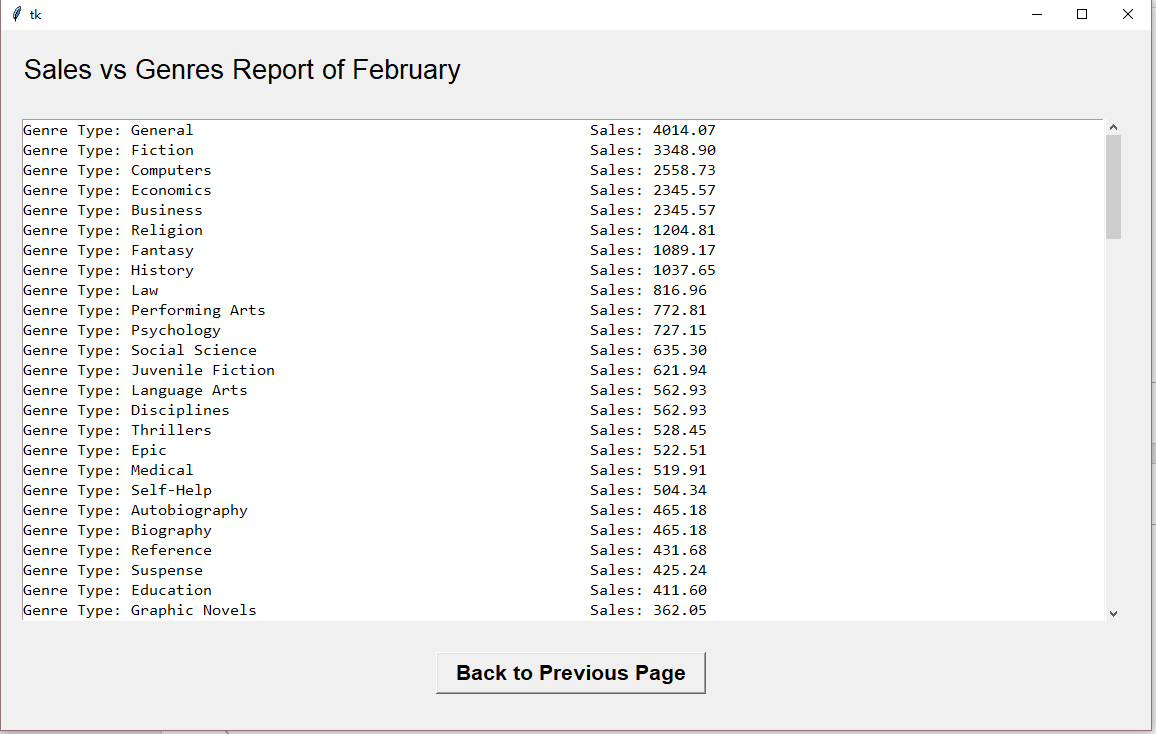


1. reportSecondPage: Accessed after selecting the month, owner choose which report they want to view

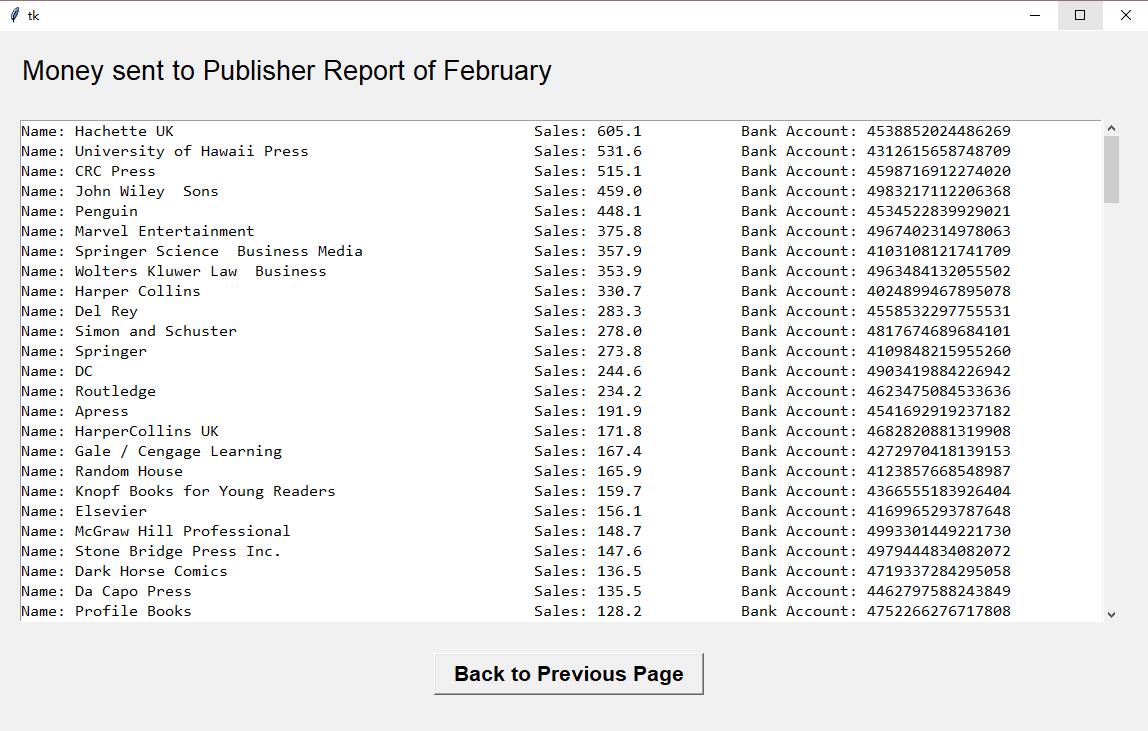


1. reportThirdPage: Accessed after chosen the type of reports. There are four reports shown in this page:

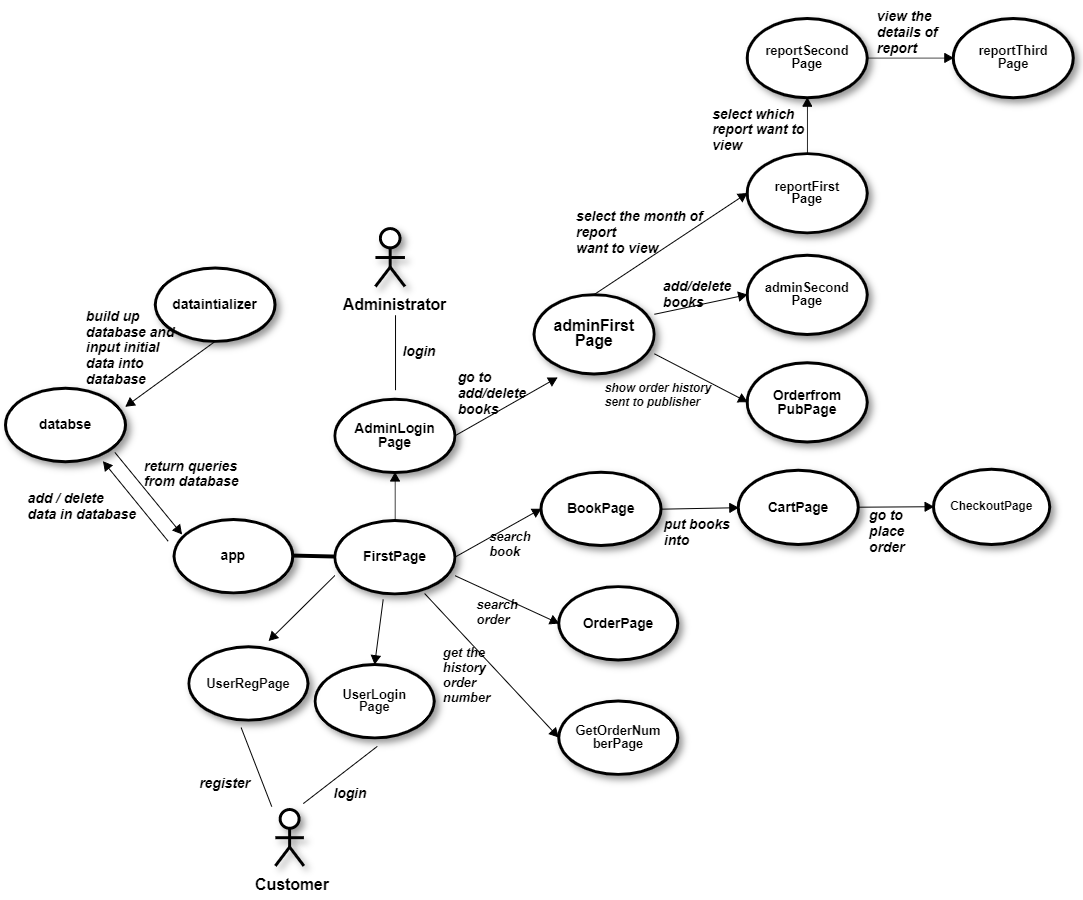


****

****

****

Below is the diagram of the application’s architecture:



# Bonus Features

approximate search for books: user can search book by title or author by entering any parts of the words.

For example: if user search “enny”, the result will include authors named “Jenny, Benny…”.

# GitHub Repository

<https://github.com/catherine28shiro/3005project>

# Appendix I

My availability is 2pm-5pm 12th 2022

# Appendix II

The password and username of the online database: <https://www.elephantsql.com/>

username: catherineli3@cmail.carleton.ca

psw: Database3005!

The source of the book data: <https://www.kaggle.com/datasets/bilalyussef/google-books-dataset>