

LIBRARY MANAGEMENT SYSTEM DATABASE PROJECT

PROJECT REPORT

Dongwei Qi
230002618

1. Project Description

This project is aimed at building a library management system.

Basic idea:

A student and faculty can issue books. The number of days will be distinct in the case of students and teachers to issue any book. Each book will have a different ID. The entry of all the books will be done, who issue that book and when and also duration. Detail of Fine(when the book is not returned at a time) is also stored.

Details of the project:

Two kinds of users can use this library management system. Users can log in as a student or a faculty. The database already logged 1000+ students and 200+ faculties from different departments with basic information like gender and email.

Each user has unique identification thus a student cannot get access to the system as a faculty. Once the password does not correspond to the user id, the user cannot get into the system either.

Different limits of rental are set for different users. A student can borrow a book for up to 4 months, while a faculty can borrow a book for up to 1 year. All users will be fined once passed the due time to return a borrowed book. The fine is defined as \$0.1 per day to all users.

All books have been added to the database already, with information on the book title, authors and current status. Currently, we have 1000+ books in storage. The default status for all books has been set 'available'. Once the book has been borrowed, the status will become 'unavailable'. Users are only allowed to borrow books available. Once the user tries to borrow an unavailable book, a message box will show up with a warning message.

Once the return date of the book passed the due date, a fine ticket with a certain amount will be added to the user's account.

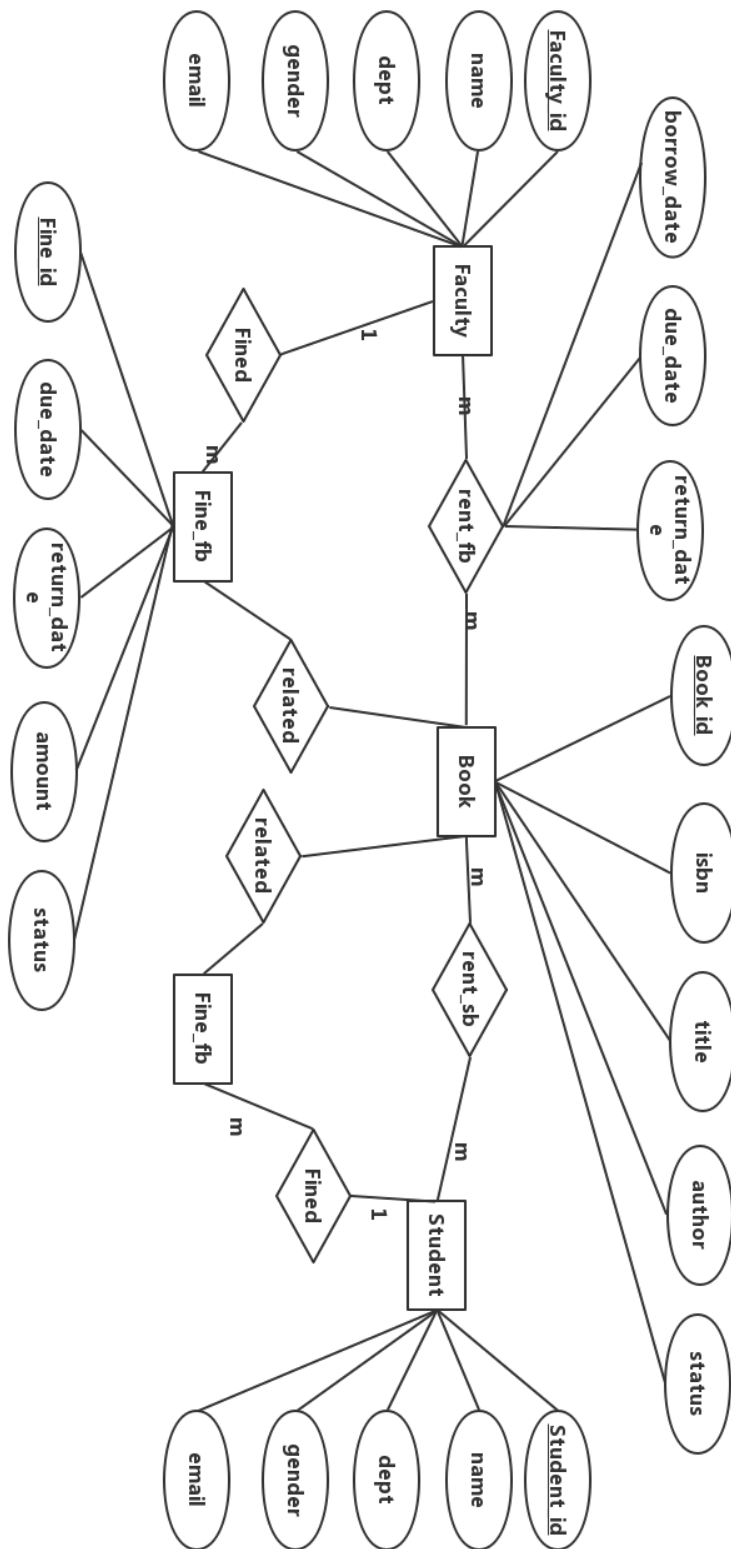
Users can go to the fine ticket interface to check how much they should pay and the payment button is provided.

Details of implementation:

The database is built based on MySQL. This library management system is built on python with pymysql module to connect with MySQL server and tkinter for GUI. This application provides several interfaces to interact with users, including login interface, user interface, borrow interface.

All interactions will be prompted with messages showing if the operation has been successfully conducted or warning of misconduct.

2. E-R Diagram



There are four entity sets: Student, Faculty, Book, Student_Fine_History, Faculty_Fine_History. And two multiple to multiple relationships Student_Rent_Books, Faculty_Rent_Books.

We made two entity Student and Faculty to discern the different privileges of faculty role and user role. Each has unique identification and separate attributes. A student can borrow a book for up to 4 months, while a faculty can borrow a book for up to 1 year.

The Book entity has multiple attributes to display its information. Users can search for book information by its id or title, also they can check for a books' availability. The status filed is set to discern if a book has been borrowed or not. Once a book has been borrowed but not returned, the status will become unavailable and user can only search for book information but cannot borrow it. After returned, the status of a borrowed book will be updated as available then users can borrow it.

The Student_Rent_Books is a multiple to multiple relationship, thus it takes Student_id as foreign key from Student and Book_id as foreign key from Book. Different from Student_Fine_History, Student_Rent_Books has to use Student_id and Book_id as part of its primary key because it is a relationship draw from two entity sets with multiple attributes. After a user borrows a book, the user's id and the book's id will be recorded, then insert a record into the database along with the conduction time of borrow. Once we have the borrow date, the due date is generated according to the user's role.

The Student_Fine_History has one to multiple relationship with Student and one to one relationship with Book, so it takes Student_id as foreign key from Student and Book_id as foreign key from Book. So does Faculty_Fine_History. All users will be fined once passed the due time to return a borrowed book and a corresponding record will be inserted into the database. The fine is defined as \$0.1 per day to all users. If the return date did not pass the due date, the database will only update the rent record and set the book status to available.

3. Table Normalization

Table Schema:

```
create table Student(Student_id int, name varchar(255) not null, Dept varchar(255) not null, gender varchar(1) not null, email varchar(255) not null, check(gender='F' or gender='M'), primary key(Student_id))
```

Student
Student id: int(11) primary key
name: varchar(255)
Dept: varchar(255)
gender: varchar(1)
email: varchar(255)

create table Faculty(Faculty_id int, name varchar(255) not null, Dept varchar(255) not null, gender varchar(1) not null,email varchar(255) not null,check(gender='F' or gender='M'),primary key(Faculty_id))

Faculty
Faculty_id: int(11) primary key
name: varchar(255)
Dept: varchar(255)
gender: varchar(1)
email: varchar(255)

create table Book(Book_id int,isbn long not null,title varchar(255) not null,author varchar(255) not null, status varchar(255) not null,primary key(Book_id))

Book
Book_id: int(11) primary key
isbn: mediumtext
title: varchar(255)
author: varchar(255)
status: varchar(255)

create table Student_Rent_Books(Student_id int,Book_id int,borrow_date timestamp not null,due_date timestamp,return_date timestamp,primary key(Student_id, Book_id, borrow_date),foreign key(Student_id) references Student(Student_id) ON DELETE CASCADE ON UPDATE CASCADE,foreign key(Book_id) references Book(Book_id) on delete cascade on update cascade)

Student_Rent_Books
Student_id: int(11) foreign key
Book_id: int(11) foreign key
borrow_date: timestamp primary key
due_date: timestamp
return_date: timestamp

create table Faculty_Rent_Books(Faculty_id int, Book_id int, borrow_date timestamp not null, due_date timestamp , return_date timestamp, primary key(Faculty_id, Book_id, borrow_date), foreign key(Faculty_id) references Faculty(Faculty_id) ON DELETE CASCADE ON UPDATE CASCADE, foreign key(Book_id) references Book(Book_id) on delete cascade on update cascade)

Faculty_Rent_Books
Faculty_id: int(11) foreign key
Book_id: int(11) foreign key
borrow_date: timestamp primary key
due_date: timestamp
return_date: timestamp

create table Student_Fine_History(Fine_id int NOT NULL auto_increment,Student_id int,Book_id int,due_date timestamp not null,return_date timestamp not null,amount int not null,status varchar(255) not null,primary key(Fine_id),foreign key(Student_id) references Student(Student_id) on delete cascade on update cascade,foreign key(Book_id) references Book(Book_id) on delete cascade on update cascade)

Student_Fine_History
Fine_id: int(11) primary key
Student_id: int(11) foreign key
Book_id: int(11) foreign key
due_date: timestamp
return_date: timestamp
amount: int(11)
status: varchar(255)

create table Faculty_Fine_History(Fine_id int NOT NULL auto_increment,Faculty_id int,Book_id int,due_date timestamp not null,return_date timestamp not null,amount int not null,status varchar(255) not null,primary key(Fine_id),foreign key(Faculty_id) references Faculty(Faculty_id) on delete cascade on update cascade,foreign key(Book_id) references Book(Book_id) on delete cascade on update cascade)

Faculty_Fine_History
Fine_id: int(11) primary key
Faculty_id: int(11) foreign key
Book_id: int(11) foreign key
due_date: timestamp
return_date: timestamp
amount: int(11)
status: varchar(255)

Table normalization:

Student(Student_id, name, Dept, gender, email)
 (Student_id)->(name, Dept, gender, email)
 The keys for this relation is: Student_id
 (Student_id) +=(Student_id, name, Dept, gender, email)

Faculty(Faculty_id, name, Dept, gender, email)
 (Faculty_id)->(name, Dept, gender, email)
 The keys for this relation is: Faculty_id
 (Faculty_id) +=(Faculty_id ,name, Dept, gender, email)

Book(Book_id, isbn, title, author, status)
 (Book_id)->(isbn, title, author, status)
 The keys for this relation is: Book_id

(Book_id)+=(Book_id, isbn, title, author, status)

Student_Rent_Books(Student_id, Book_id, borrow_date, due_date, return_date)

(Student_id, Book_id, borrow_date)->(return_date)

(borrow_date)->(due_date)

The keys for this relation is: (Student_id, Book_id, borrow_date)

(Student_id, Book_id, borrow_date)+= (Student_id, Book_id, borrow_date, due_date, return_date)

In this case, there's no BCNF violations because: for (borrow_date)->(due_date), its left hand side does not contain any candidate key, also its right hand side is not contained in any candidate key.

Faculty_Rent_Books(Faculty_id, Book_id, borrow_date, due_date, return_date)

(Faculty_id, Book_id, borrow_date)->(return_date)

(borrow_date)->(due_date)

The keys for this relation is: (Faculty_id, Book_id, borrow_date)

(Faculty_id, Book_id, borrow_date)+= (Faculty_id, Book_id, borrow_date, due_date, return_date)

In this case, there's no BCNF violations because: for (borrow_date)->(due_date), its left hand side does not contain any candidate key, also its right hand side is not contained in any candidate key.

Student_Fine_History(Fine_id, Student_id, Book_id, borrow_date, return_date, amount, status)

(Fine_id)->(Student_id, Book_id, borrow_date, return_date, amount, status)

(borrow_date, return_date)->(amount)

The keys for this relation is: Fine_id

(Fine_id)+= (Fine_id, Student_id, Book_id, borrow_date, return_date, amount, status)

In this case, there's no BCNF violations because: for (borrow_date, return_date)->(amount), its left hand side does not contain any candidate key, also its right hand side is not contained in any candidate key.

Faculty_Fine_History(Fine_id, Faculty_id, Book_id, borrow_date, return_date, amount, status)

(Fine_id)->(Faculty_id, Book_id, borrow_date, return_date, amount, status)

(borrow_date, return_date)->(amount)

The keys for this relation is: Fine_id

(Fine_id)+= (Fine_id, Faculty_id, Book_id, borrow_date, return_date, amount, status)

In this case, there's no BCNF violations because: for (borrow_date, return_date)->(amount), its left hand side does not contain any candidate key, also its right hand side is not contained in any candidate key.

All the schemas are in BCNF.

Since all the schemas are designed with the unique primary key without redundant field, at the same time, the usage of foreign key refers to fields in other schemas make full use of relationships between relations, all the schemas are in good structures.

4. Data Collection

Basically I collect data from open dataset on the Internet. I extracted 1000+ book information from most popular books and extracted 1000+ user information from name dataset. I set up identification for each entity to make sure that there are no duplication. There are some interesting joins like: I used borrow_date in Student_Rent_Books and Faculty_Rent_Books as foreign key to make sure the uniqueness of the rent records.

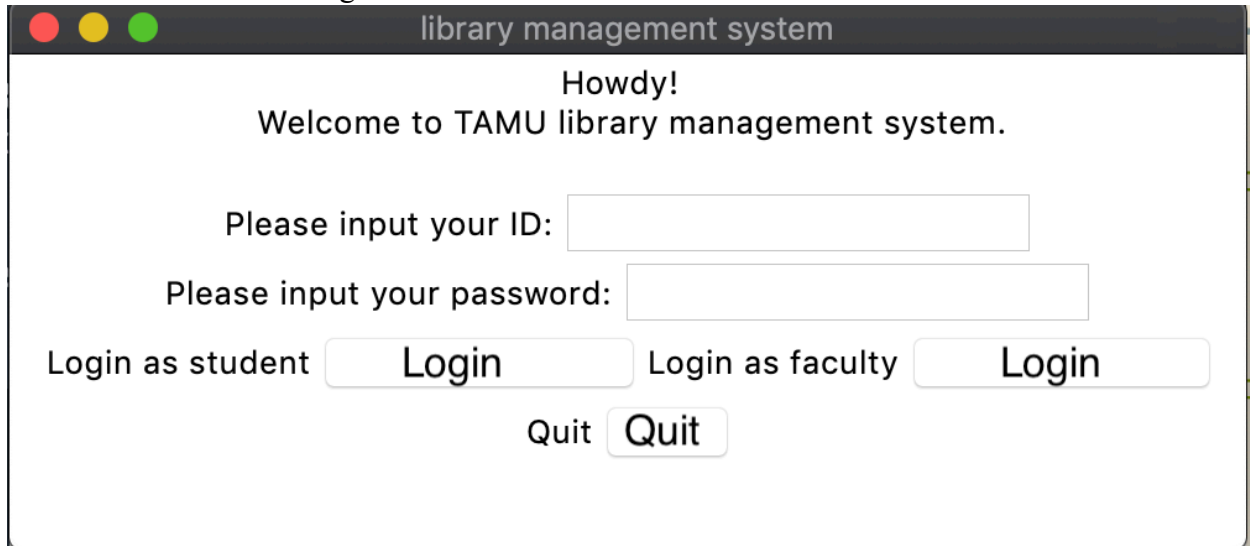
5. User Interface

Function list:

user_login
search_for_book_by_id
search_for_book_by_title
borrow_book
return_borrowed_book
check_for_fine
pay_for_the_fine

Each function takes parameters that can be inquired about in SQL query sentences. To conduct a query, other necessary fields like 'borrow_date', we can inquire for it at first and pass the result of a query as parameters of certain functions for further updates or modifications into SQL.

Instructions of usage:

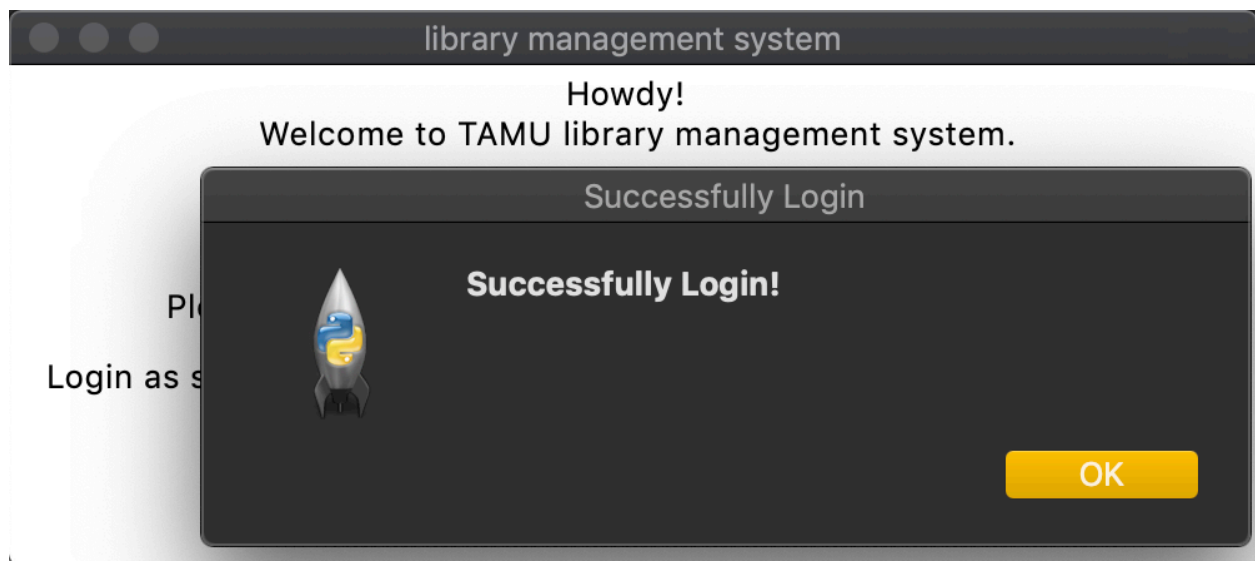


The screenshot shows a window titled "library management system". Inside the window, the text "Howdy!" is followed by "Welcome to TAMU library management system." Below this, there are two input fields: "Please input your ID:" and "Please input your password:". At the bottom, there are four buttons: "Login as student", "Login", "Login as faculty", and "Login". There are also two "Quit" buttons, one on the left and one on the right.

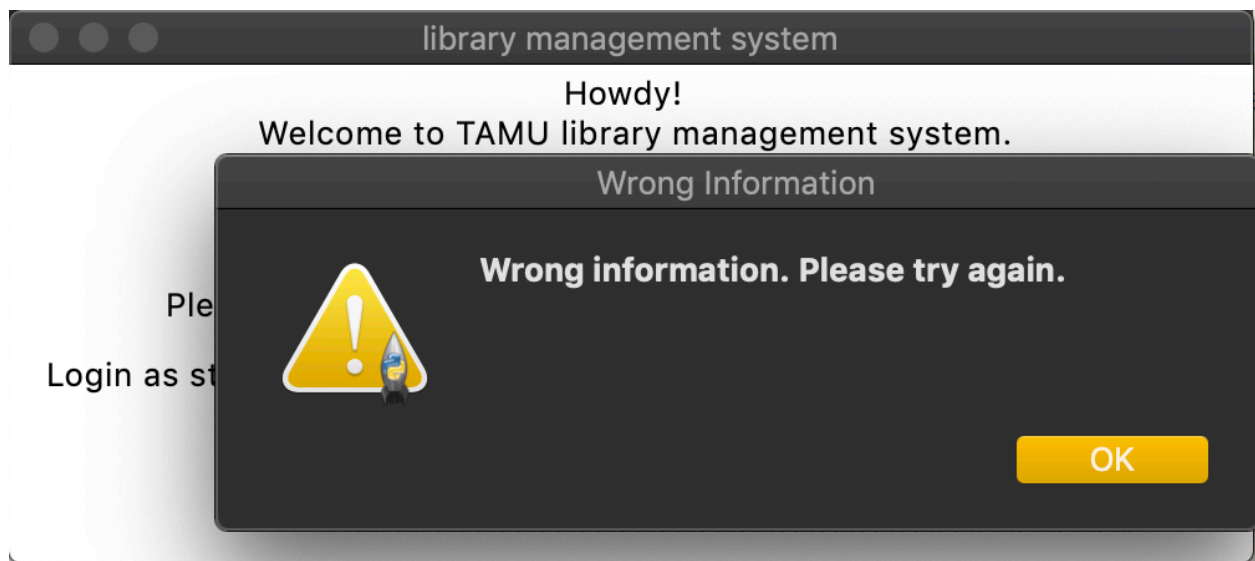
Users can input their user_id and password(which is the same as their user_id by default) and select to login in as a student or a faculty.

Or click quit to close the window.

We can input 230004501 to login in as a student.



Once the user_id and password is correct, there will be a system message box to send an information demonstrate successfully login. Otherwise an waring message will be sent to ask user to try again.



Once successfully logged in, a system welcome interface will show up and prompt user to search for a book by book id or by book title. User can also choose to return a borrowed book or check for fine tickets.

library management system

Welcome! Student Mary

You can search for book by input book id here:

You can search for book by input book title here:

Return borrowed book here:

Check for fine tickets here:

Quit

A message box will show up to prompt the information of the book and its current status. If the book is available, a borrow button will show up then user can click to borrow it.

library management system

Welcome! Student Mary

You can search for book by input book id here: 1


You can search for book by input book title here:

Return borrowed book here:

Check for fine tickets here:

Quit

Book Info



title: The Hunger Games
author: Suzanne Collins
unavailable now

Search a book by id. This book is unavailable now.

library management system

Welcome! Student Mary

You can search for book by input book id here:


You can search for book by input book title here: The Great Gatsby

Return borrowed book here:

Check for fine tickets here:

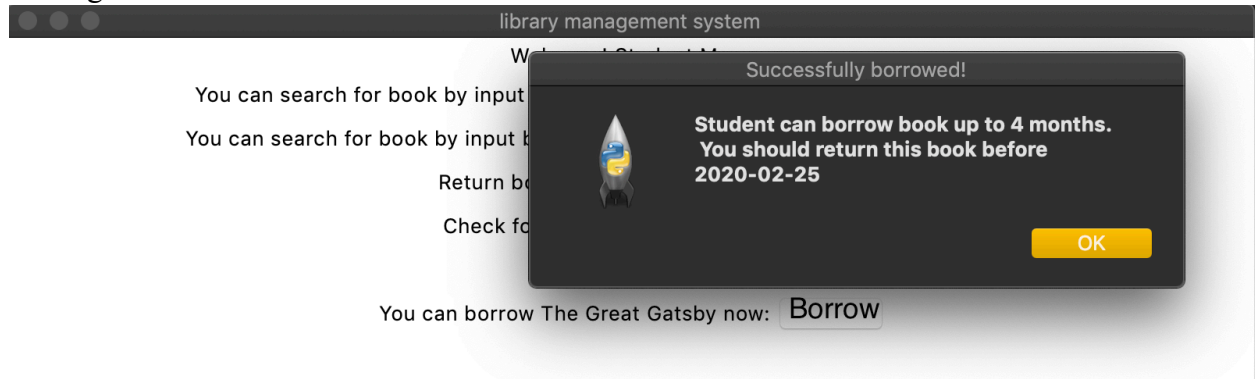
Quit

Book Info

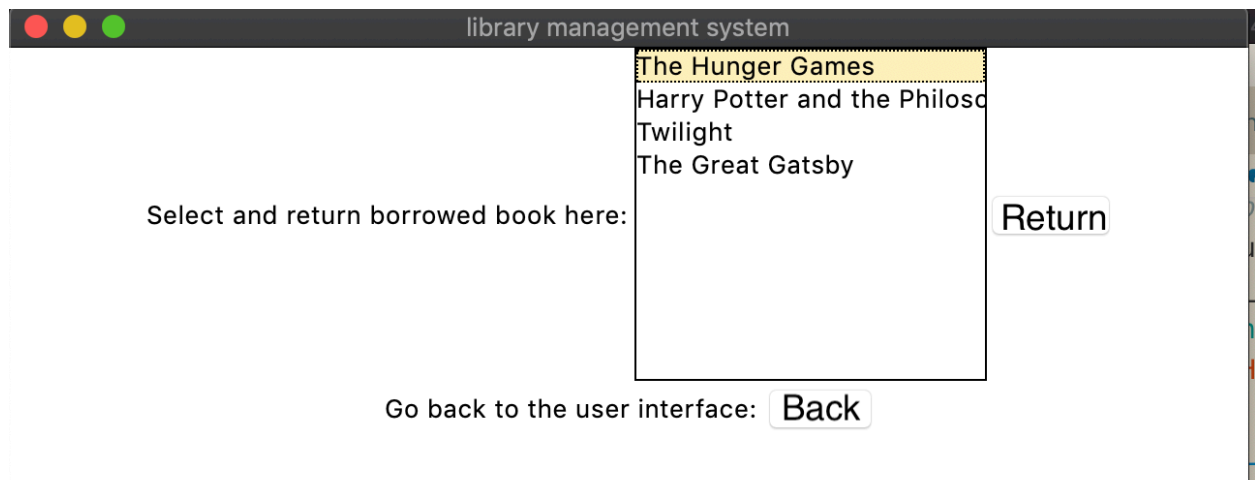


title: The Great Gatsby
author: F. Scott Fitzgerald
available now

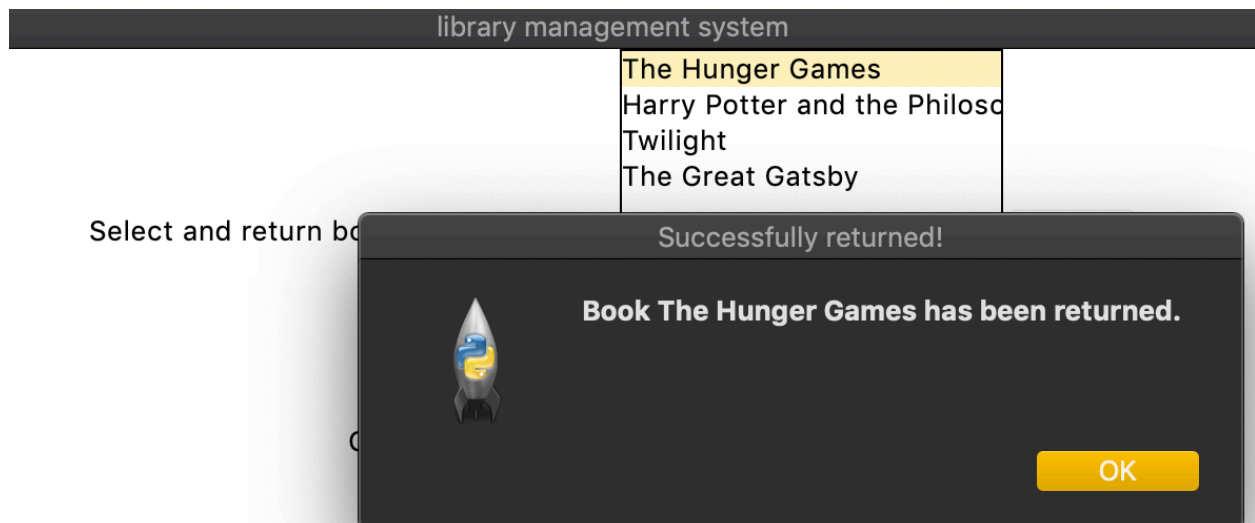
Search a book by title. This book is available now and we can click the borrow button. A message box will come out and tell user when to return the book.



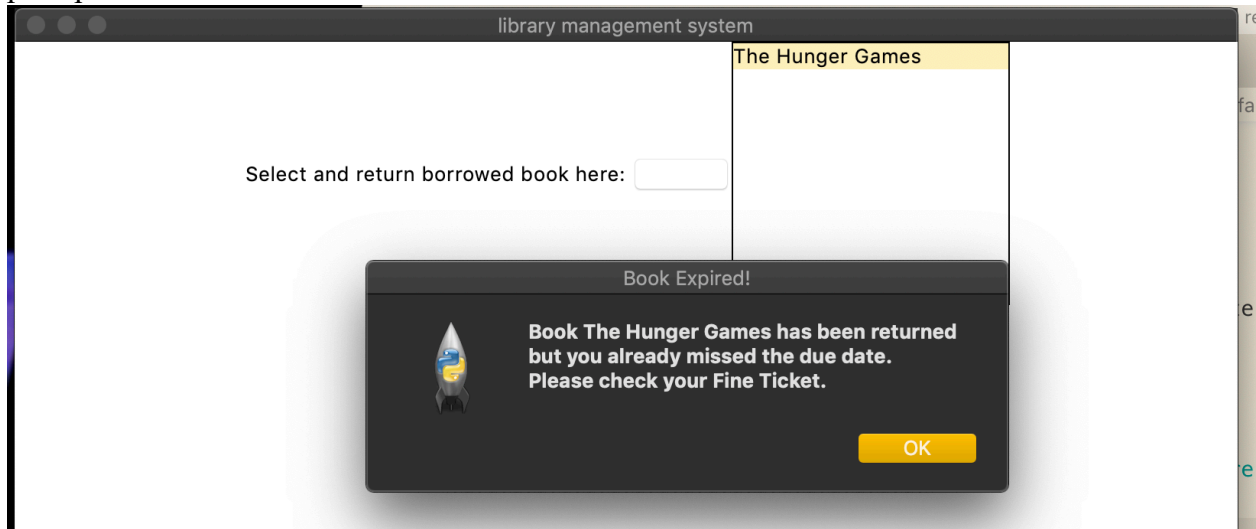
After click the return book button, user will be lead to a return book interface, with lists of borrowed book. User can choose a book and click return to return the book. Or go back to the last interface.



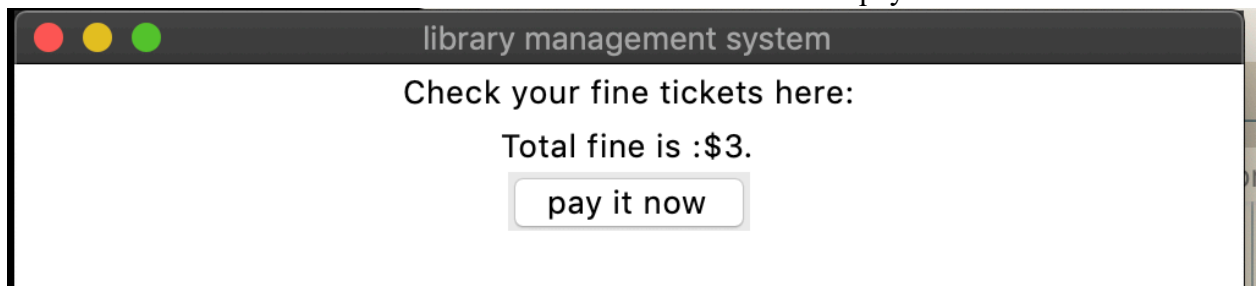
Once a book is returned, a message box will show up to prompt a message showing successfully returned.



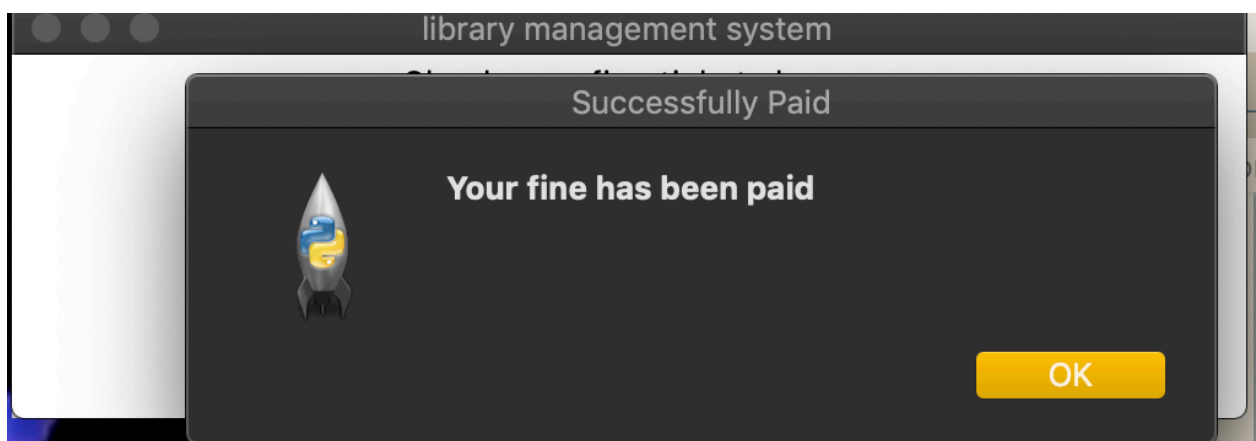
Once the user passed the due date of a book to be returned, a warning message will show up to prompt user to check the fine tickets.



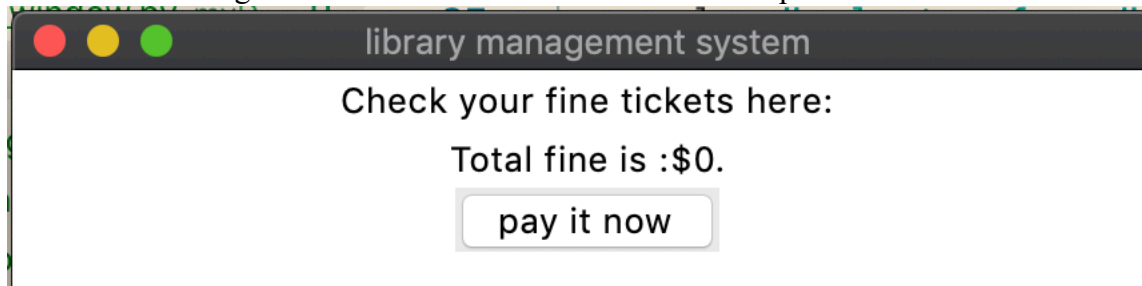
After user clicked the check for fine tickets button, user will be lead to the fine interface. The total fine has been showed on the interface and user can choose to pay the fine.



After user clicked the 'pay it now' button, a message will show up prompting that the fine has been paid.



User then will be guide to the refreshed fine interface with updated fine amount.



6. Project Source Code

Source code is pushed on https://github.com/tomatoJr/myDB_library_management_system

7. Discussion

I used to work on a database project when I was a sophomore. We built a bank simulation system in 6-people team-work. But this time, I have to do it all by myself.

In the first week, this project was assigned, I began to work on finding the proper project. Indeed I am a huge fan of the library, I even used to write a web crawler to extract records from the public library and my alma mater Shanghai University just to analyze how many books I read this year and what is the most popular genera in my book lists. That is why I chose to simulate a library management system. However this time, it is a pity for me that I didn't fulfill the generating yearly report feature in my simulated library management system.

I used to work on java and android to implement applications, but this time I choose to use Python with tkinter to fulfill the GUI. I think the course project is a chance for me to fulfill a goal via different methods every time. It's cool for me to build the database but it's I found it very hard to use a brand new module to implement the GUI. Thkinter is a stranger to me but I found it similar to android components to some extents. The past developing experience helped me a lot so I can get used to it quickly.

From the project, I learned a lot from self-programming. I myself had to work as both the customer, the product manager, the project manager, and the developer. I had to push myself to fulfill the functions, check for bugs and make improvements to the features. I discovered a lot of pleasure in the project. One can learn a lot from himself.