Design Report
E-Library

By: Khadeeja Din, Xue Wei Fan, Kai Ying Chen

1. Introduction:

This report presents the design of an E-book sharing system, called E-library. E-library is an interface where people can share books with each other, and read books shared by others. A catalog of all the books are maintained, and new books can be added by users. A GUI for the application is designed, where users can easily carry out different actions. The GUI screens are described in section 5 of the report. The use cases for each scenario are provided in section 2. Section 3 provides E-R diagrams for the entire system, describing the attributes and key for each entity. Section 5 provides further detailed design of the main functionalities for every object. The pseudo-code for users, books, and library classes is given. The last 2 sections discuss the group meeting minutes and the updates made to the specification report.

2. Use Cases:

There are three users in this system: visitor, registered user and super user.

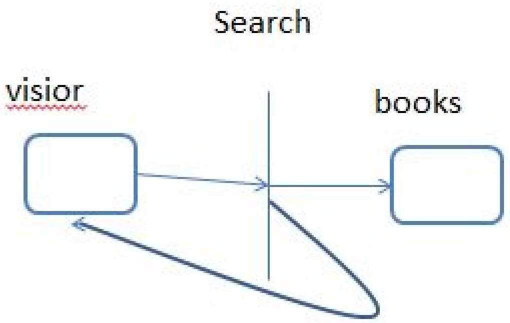
Visitor: browse available E-book catalog and the reviews/ratings, apply to be a new RU.

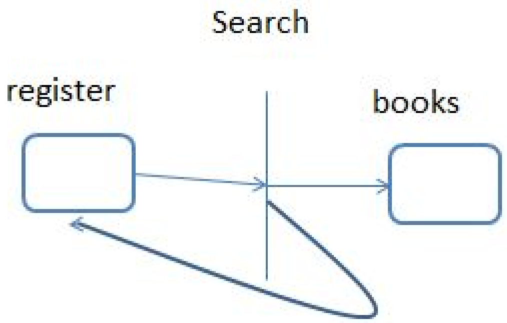
Registered User: all visitors' allowed operations, contribute books, read, rate and review a book, send complaints to Super Users.

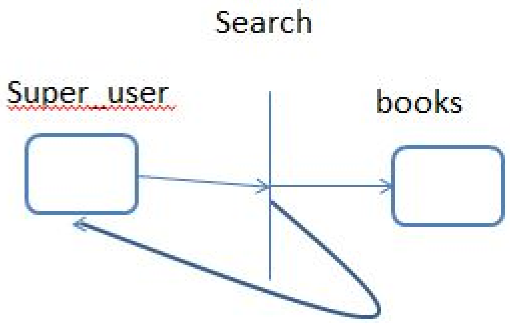
Super User: all RU's allowed operations, new user approval decisions, book approvals/updates and complaints processing, set up the points for different reading durations.

Use Case Scenarios & Petri-nets

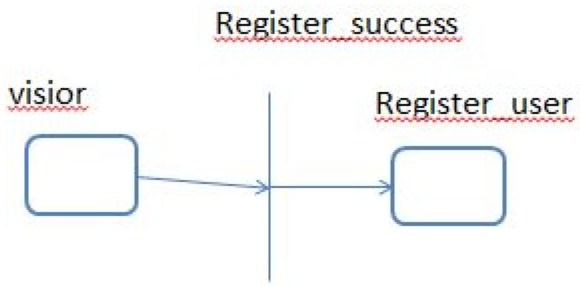
use case 1: visitors, registered users and super users can search books so that they know what books are available for reading in this system.

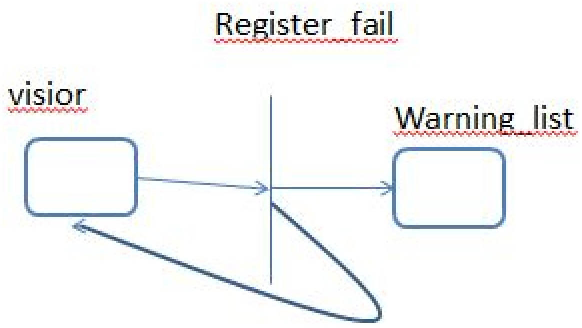




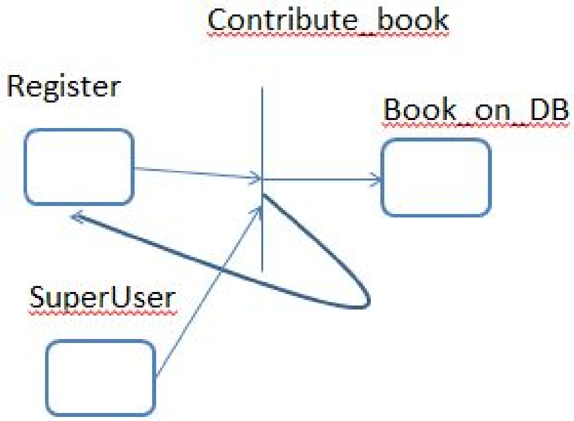


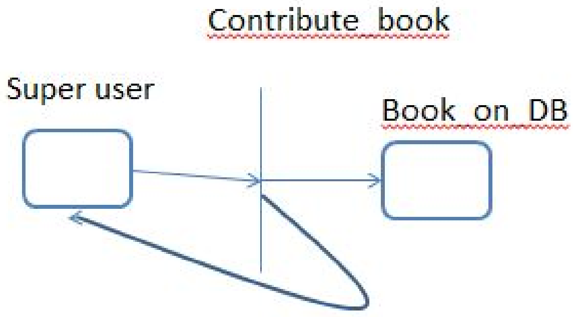
use case 2: visitors can apply as a new user of this system so that he/she can read books or do whatever he can as a registered user.



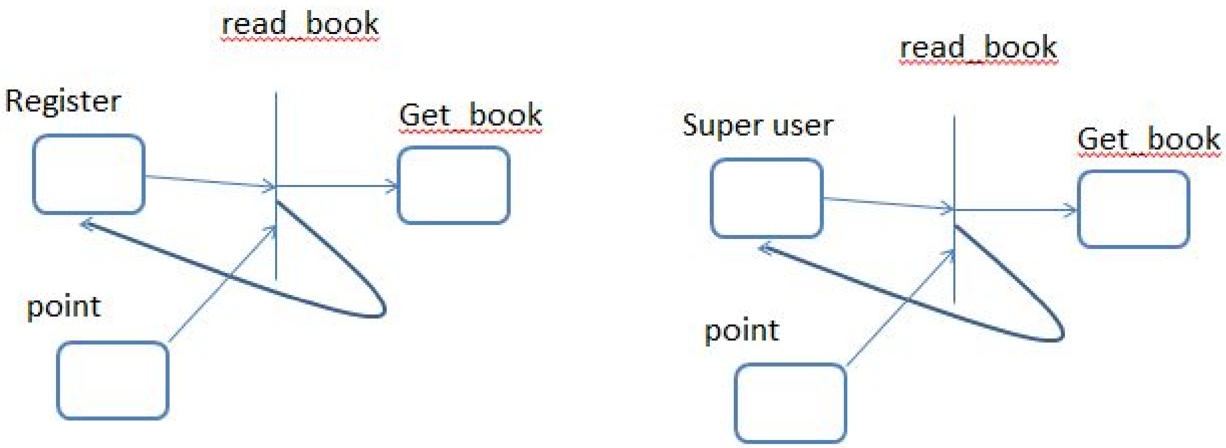


use case 3: registered users and super users can contribute books to the system so that they can get points that are used to read other books in this system.

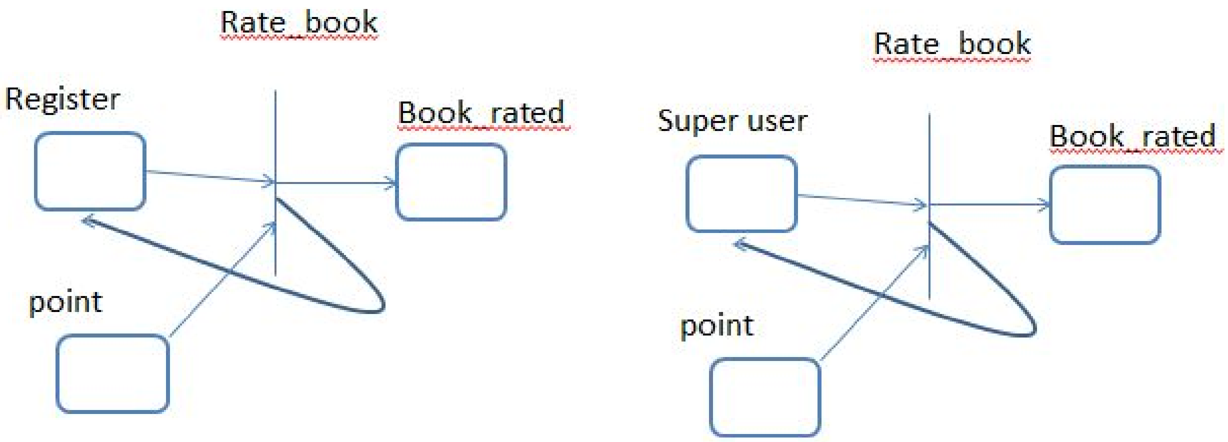




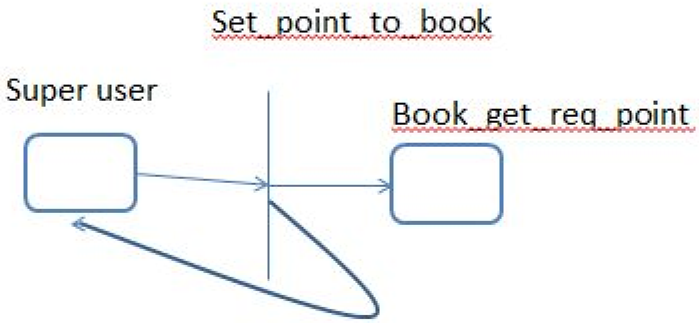
use case 4: registered users and super users can read books available in the system.



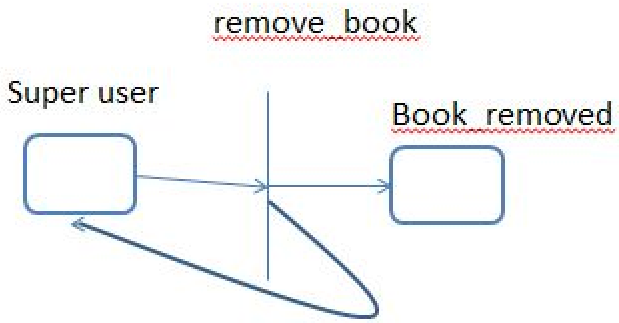
use case 5: registered users and super users can review and rate a book once they have read it.



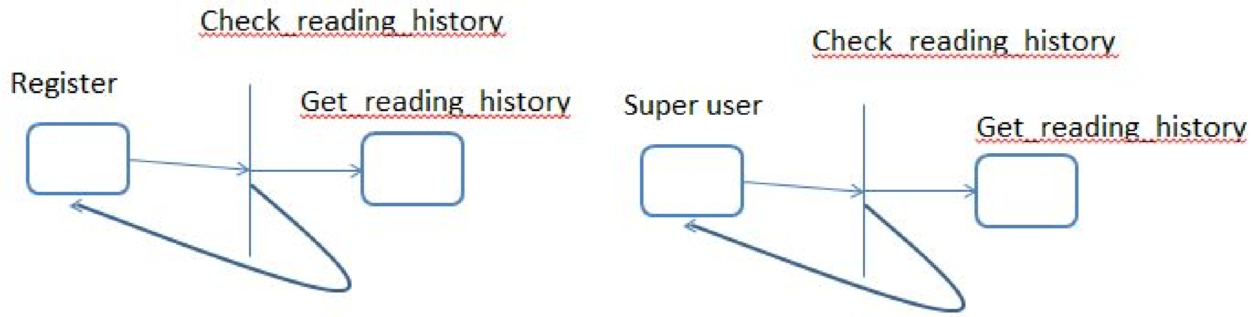
use case 6: super users can set points needed to read a specific book.



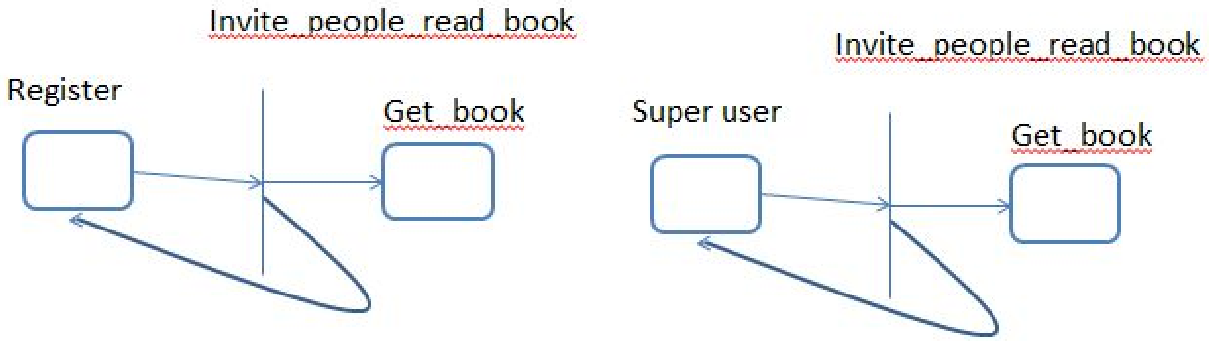
use case 7: super users can remove a book from this system if there are complaints for the book.



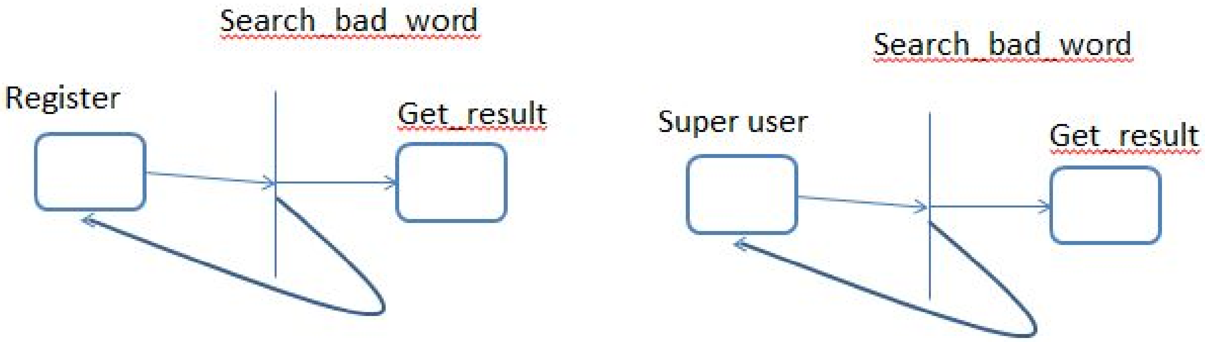
use case 8:registered users and super users can check their reading histories.



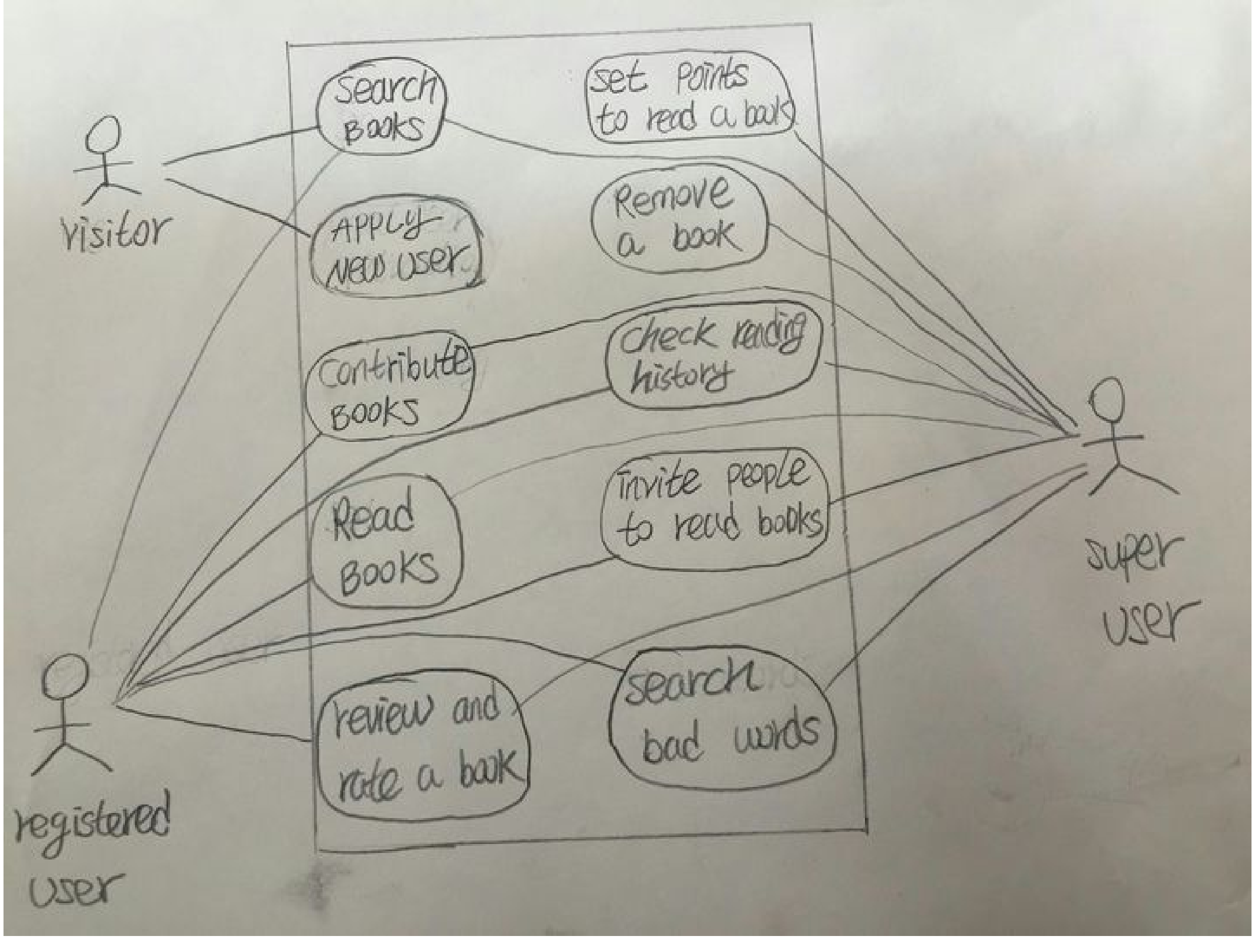
use case 9: registered users and super users can invite people to read a book.



use case 10:registered users and super users can aslo search bad words in the book.

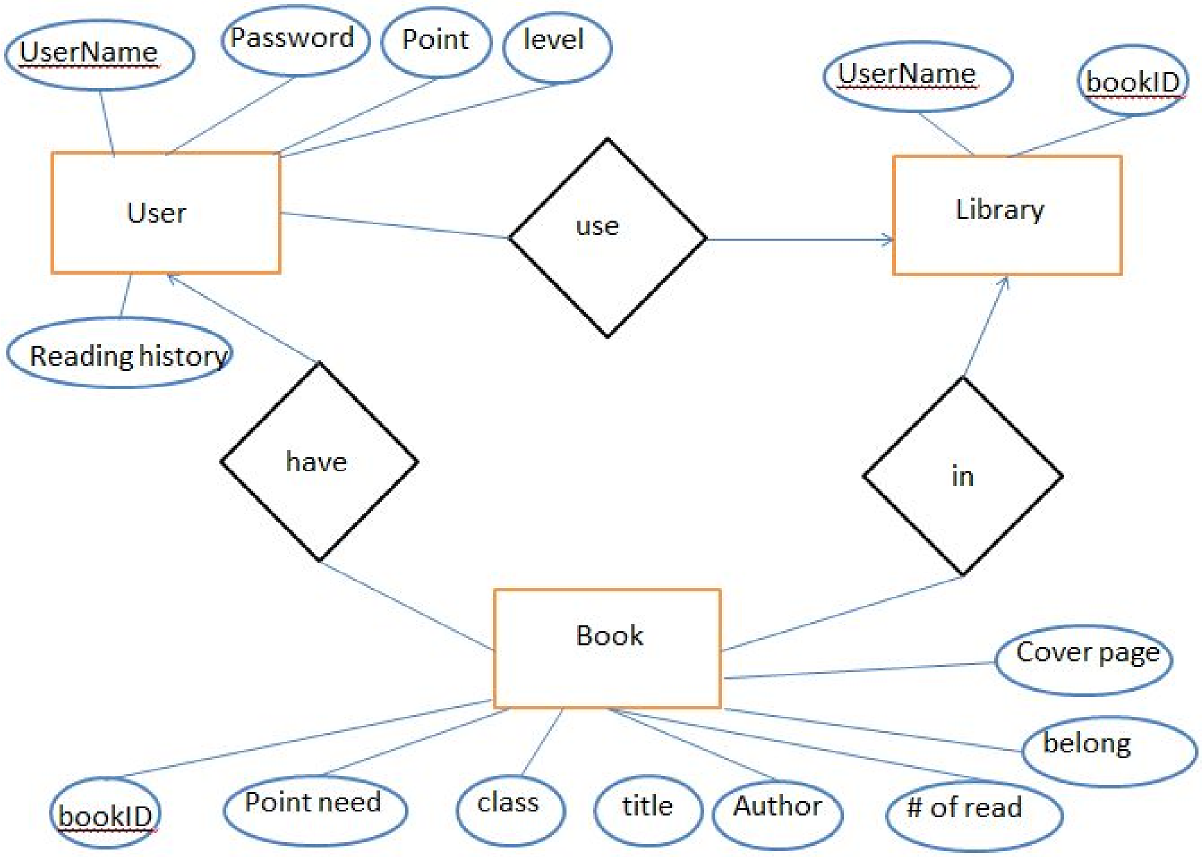


The combination of all these user cases are shown in the diagram below.



Use Case Diagram

3. E-R Diagram:



The following is the description of the above E-R diagram for the E-library system. We have three main entities in our system:

1- User: There are three **levels** of users: visitor, registered user, and super user. Visitors can sign up for an account with the E-library. However, the super users need to make approval decisions first. Once a user is approved registration they become a registered user. Each registered user gets a unique **username** and **password.** Users can earn **points** from reading books from the library.

2- Book: Each book have a unique **call number/bookId** and a unique owner to whom the book **belongs**. Every book has a **title**, **author**, **cover page**, **summary**, **points needed**, **class**(category type/genre), and **number of reads**.

3- Library: Library is a catalog containing all books. It have the record of **usernames** and **bookId**.

Relations: Users use the library catalog to search the books, and each book in the library have a unique user who contributed that book.

4. Detailed Design:

Pseudo-code for main User Classes

```
class Visitor:
  def __init__(self):
  def search_book(self):
  def register(self):

class RegisteredUser(Visitor): #registered user inherits from visitor class
  registered_user_count = 0 #total number of users in system
  def __init__(self, name, username, password, points):
    self.name = name
    self.username = username
    self.password = password
    self.points = points
    RegisteredUser.registered_user_count += 1

  def read_book(self):
    #first check if user have enough points to read book
    self.book_callnumber = ""
    self.book_read = False #sets bool to true once user reads the book

  def contribute_book(self):
    self.book_callnumber = "" #assign a unique callnumber to book
    self.ask_points = 0
    self.book = "" #read the book from input file

  def rate_book(self):
    self.book_callnumber = 1
    #first check if user has read the book
  def review_book(self):
    self.book_callnumber = 1
    #first check if user has read the book

  def read_reviews(self):
    self.book_callnumber = 1

  def send_complaints(self):
     #complain to superusers if see any bad words in any book in the catalog
  def invite_users(self):
      #invite users to read book with you
  def book_history(self):
    #record of books the user read. This will help suggest books to user in future

class SuperUser(RegisteredUser): #SuperUser inherits from RegisteredUser class
  super_user_count = 0 #total number of superuser in system
  def __init__(self):
    super().__init__()
    SuperUser.super_user_count += 1

  def approve_new_user(self):
   new_user requests = 0  #total number of new user requests
   #superusers either approve or disapprove their requests
  def book_approval(self):

   #total number of superuser in system


  def book_updates(self):

   #update books


  def process_complaints(self):

   #super user will read complaints from registered users, and process them accordingly.


  def setup_reading_points(self):

   #superuser decides the reading_points for each book
```

Pseudo Code for Library Catalog

```
class Library():
  def __init__(self):
     self.books = []

  def searchBook(self, title):
     '''
     :param title:
     :return: book
     '''

  def searchUser(self, name):
     '''
     :param name:
     :return: user
     '''
```

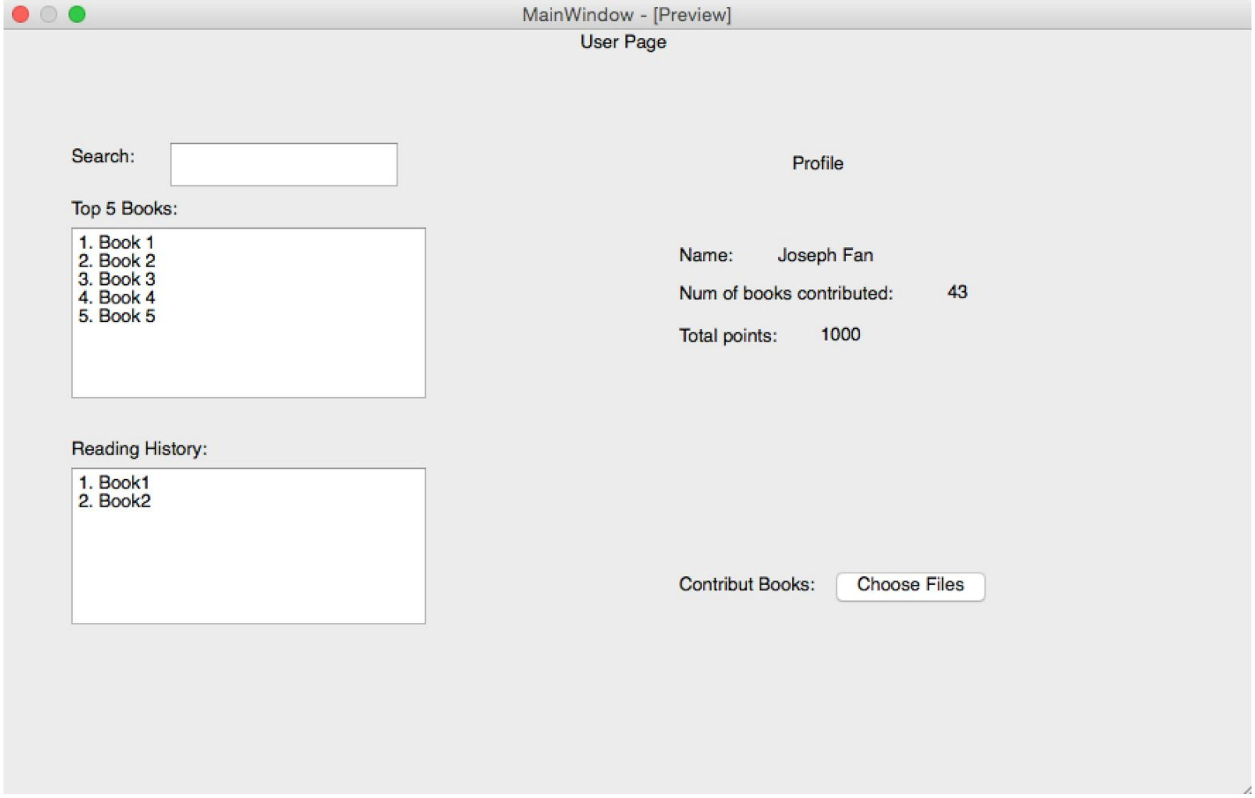
Pseudo Code for Books

```
class Book:

  book_count = 0 #shared variable by all objects of type Book. Total number of books in catalog

  def __init__(self, callnumber):

     self.callnumber = callnumber

     Book.book_count += 1
```

```
def cover_page(self):

   #read cover page for each book from input file, and add it to the catalog

  #show cover page when user selects the book to read


def summary(self):

  #read summary for each book from input file


def clock(self):

    #timer for book reading durations


def points_needed(self):

    #points needed to read the book for different time durations


def book_reviews(self, count):

    self.count = 0 #initially the number of reviews are 0


def book_rating(self, rate):

    self.rate = 0 #initialize the rating to 0


def weight(self):

    #define weights for each review and rating for this book


def remove_book(self):

   complaints = 0 #check the number of complaints, if number of complaints = 3 , then remove book

   reading_count= 0  #f no one read book for a certain period of time then remove the book
```

5. System GUI Screens:

Registered User Page



MainWindow - [Preview]

User Page

Search:

Top 5 Books:

1. Book 1
2. Book 2
3. Book 3
4. Book 4
5. Book 5

Reading History:

1. Book1
2. Book2

Profile

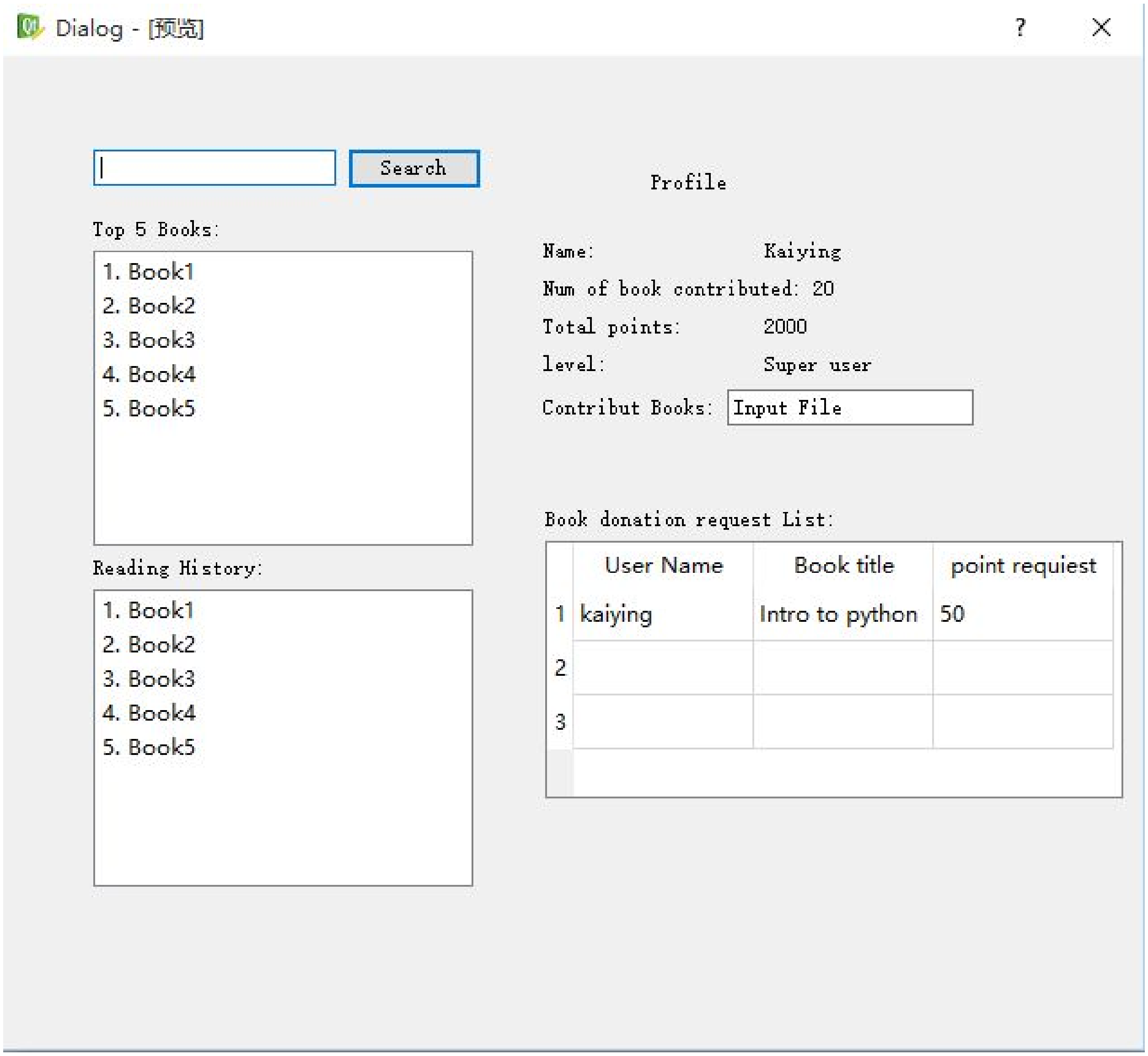
Name: Joseph Fan

Num of books contributed: 43

Total points: 1000

Contribut Books:

Super user page



Top 5 Books:

1. Book1
2. Book2
3. Book3
4. Book4
5. Book5

Reading History:

1. Book1
2. Book2
3. Book3
4. Book4
5. Book5

Name: Kaiying

Num of book contributed: 20

Total points: 2000

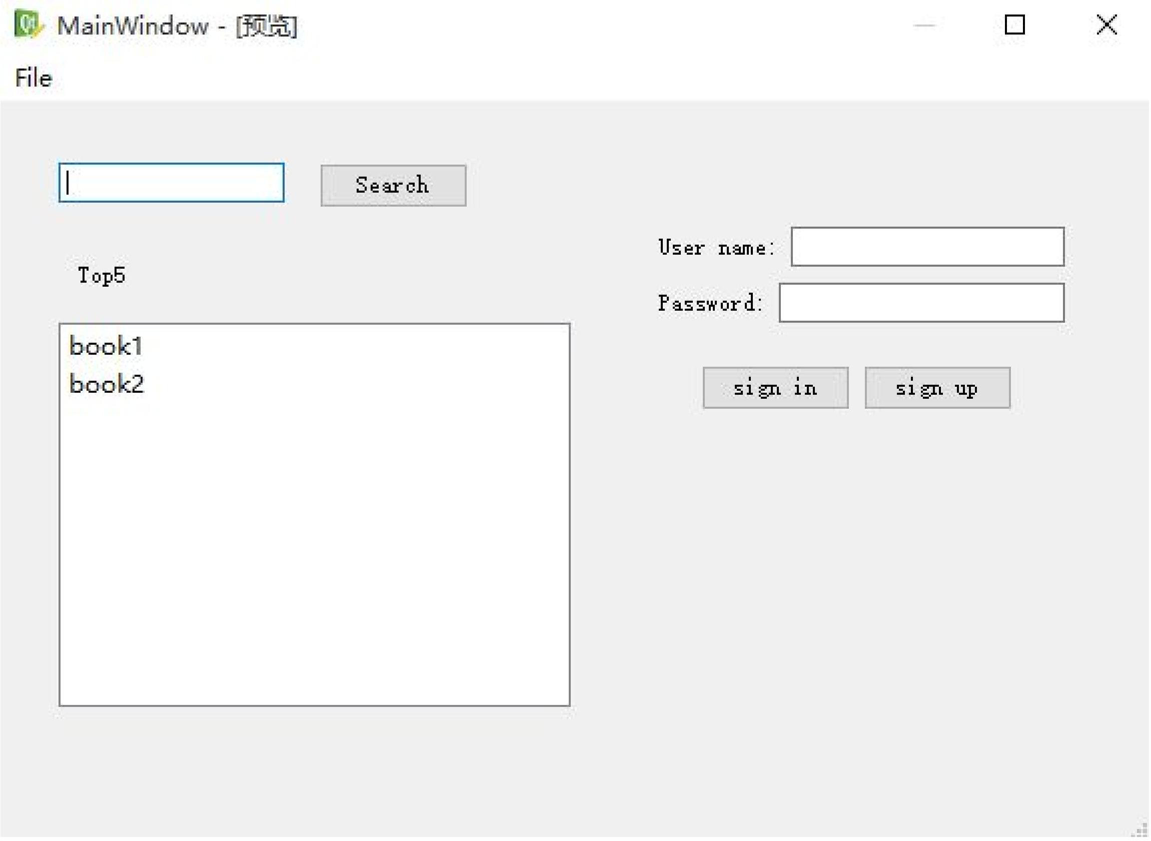
level: Super user

Contribut Books: Input File

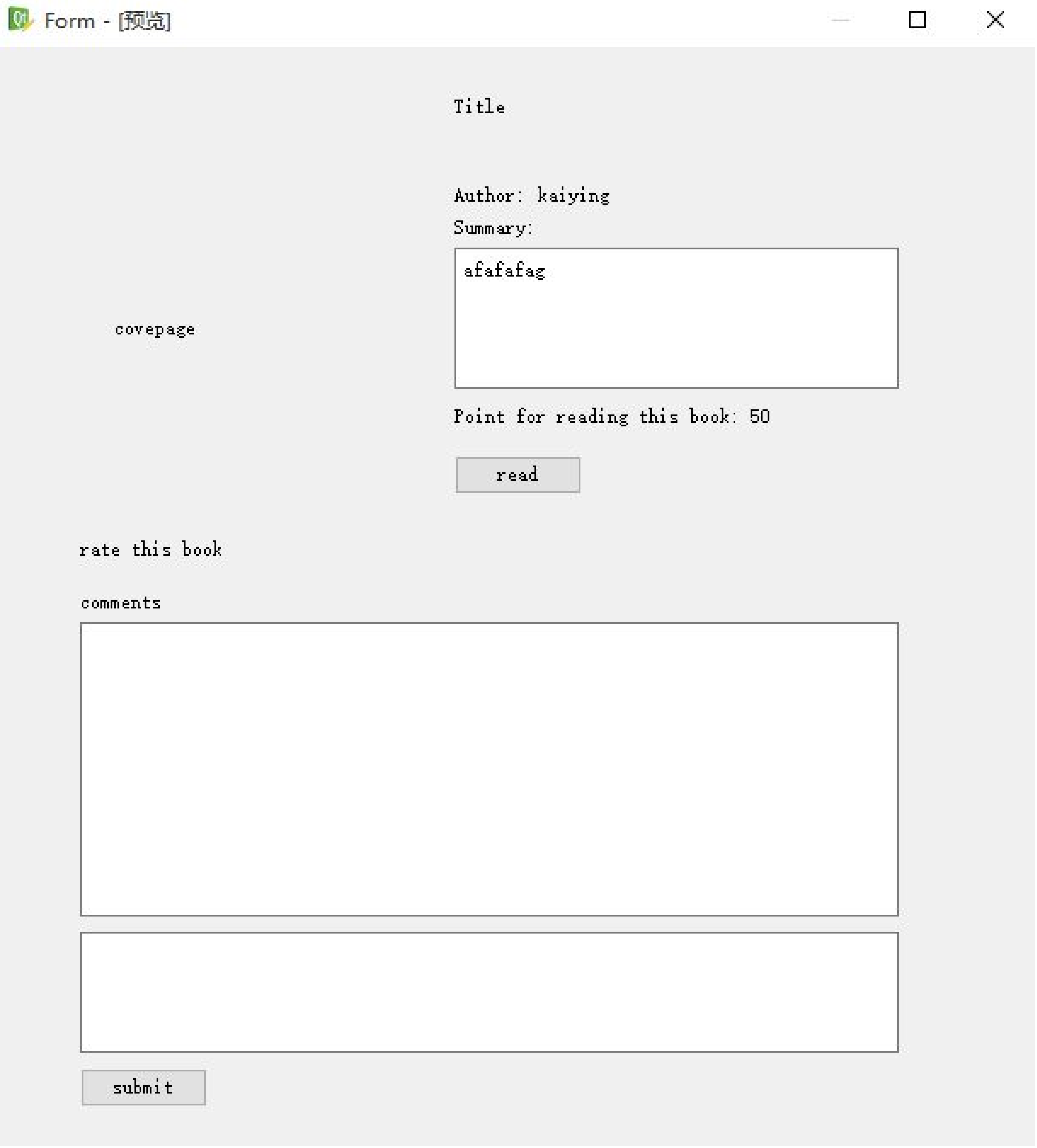
Book donation request List:

	User Name	Book title	point requiest
1	kaiying	Intro to python	50
2			
3			

Visitor page



Book page



Form - [预览]

Title

Author: kaiying

Summary:

afafafag

Point for reading this book: 50

rate this book

comments

6. Minutes of Group Meetings:

Meetings were held for 2 hours every week. Additionally we also had 2 hours of lab/week.

7. Updates to Project Specification Report:

In this report, we corrected the broken use case diagram that we made in the first specification report (please check the use case diagram in section 2 of this report). We also corrected contents in section 2 and section 3 of SRS report so that we only have the use case and a brief description in section 2 and a detailed description between users and use cases in section 3. The version 2 of Software requirement report is attached.