Library Seating Reservation System

Team leader: William Wang

Member: Alan Li, Scott Dai ,

Andy Jiang ,Abbot Xiong

Mike Zhang

1.Customer Problem Statement ---------------------------------------------------------- 8

1.a Problem Statement --------------------------------------------------------------- 8

2.Glossary of Terms-------------------------------------------------------------------------12

3. System Requirements ------------------------------------------------------------------- 14

3.a Enumerated Functional Requirements ----------------------------------------- 14

3.b Enumerated Nonfunctional Requirements ------------------------------------- 15

3.c User Interface Requirements ----------------------------------------------------- 17

4. Functional Requirements Specification ----------------------------------------------- 21

4.a Stakeholders ------------------------------------------------------------------------ 21

4.b Actors and Goals ------------------------------------------------------------------- 21

4.c Use Cases ---------------------------------------------------------------------------- 22

4.d System Sequence Diagrams ------------------------------------------------------- 26

5.Effort Estimation using Use Case Points------------------------------------------------30

6. Domain Analysis---------------------------------------------------------------------------36

7. Interaction Diagrams ---------------------------------------------------------------------- 37

8. Class Diagram and Interface Specification ----------------------------------------------39

8.a Class Diagram -------------------------------------------------------------------39

8.b Data Types and Operation Signatures----------------------------------------------40

8.c Traceability Matrix--------------------------------------------------------------------42

9. System Architecture and System Design ------------------------------------------------45

9.a Architectural Styles------------------------------------------------------------------45

9.b Identifying Subsystems-----------------------------------------------------------46

9.c Mapping Subsystems to Hardware----------------------------------------------46

9.d Persistent Data Storage-----------------------------------------------------------47

9.e Network Protocol------------------------------------------------------------------47

9.f Global Control Flow---------------------------------------------------------------48

9.g Hardware Requirements ---------------------------------------------------------49

10.Algorithms and Data Structures -------------------------------------------------------50

11.User Interface Design and Implementation -----------------------------------------51

12.Design of Tests --------------------------------------------------------------------------54

13.History of Work -------------------------------------------------------------------------56

14.Summary of change ---------------------------------------------------------------------57

15.References --------------------------------------------------------------------------------57

**Work breakdown**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Open Fire | | | | | |
| Responsibility  levels |  | William | Scott | Andy | Mike | Abbot | Alan |
| Problem Statement |  |  | 50% |  |  | 50% |
| System Requirements | 50% |  |  |  |  | 50% |
| Enumerated Functional Requirements |  |  |  |  | 100% |  |
| Enumerated Nonfunctional Requirements | 40% | 60% |  |  |  |  |
| User Interface Requirements | 20% |  |  | 80% |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Open Fire | | | | | |
| Responsibility  levels |  | William | Scott | Andy | Mike | Abbot | Alan |
| Stakeholders |  |  |  | 100% |  |  |
| Actors and Goals |  |  |  | 100% |  |  |
| Use Cases | 20% | 20% | 20% |  | 20% | 20% |
| System Sequence Diagrams | 100% |  |  |  |  |  |
| Preliminary Design |  | 50% | 50% |  |  |  |
| User Effort Estimation |  | 50% | 50% |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Open Fire | | | | | |
| Responsibility  levels |  | William | Scott | Andy | Mike | Abbot | Alan |
| Interaction Diagrams |  |  |  |  |  | 100% |
| Class Diagram | 50% |  |  |  |  | 50% |
| Data Types and Operation Signatures |  |  |  |  |  | 100% |
| Traceability Matrix | 100% |  |  |  |  |  |
| Architectural Styles |  |  |  | 100% |  |  |
| Mapping Subsystems to Hardware |  |  |  |  | 100% |  |
| Persistent Data Storage |  | 50% | 50% |  |  |  |
| Network Protocol |  | 50% | 50% |  |  |  |
| Global Control Flow |  | 50% | 50% |  |  |  |
|  | William | Scott | Andy | Mike | Abbot | Alan |
|  |
| Hardware Requirements |  |  |  | 100% |  |  |
| Identifying subsystems |  |  |  |  | 100% |  |

1. **Customer Problem Statement**

1.1 **Problem Statement**

The library is a public place where documents are collected and organized and students are provided to study. It is an important resource for university study. Learning and making full use of the library's academic resources is the basic skill that every college student must master. However, as a librarian, I found that many study seats are only books, no one. Many students just put school supplies on the table, no one is sitting, causing students to complain to the library many times about the seat.0 And such a situation will have a very bad impact. First of all, such a situation will affect student learning. Students who want to study can't study hard without sitting. Second, such a situation will affect the order of the library. There are items in the seat but no one is wasting resources. In the end, such a situation will hurt the reputation of the library and will make everyone feel that library management is not working.

The above problems all reflect the problems of our library and cannot be delayed for too long. As a librarian, we are very pleased with the enthusiasm of students for learning. Because students love to learn, especially for students who want to take a postgraduate degree, seat is to avoid the way they can not find a seat. Even when the library opens, there are many students who go in and place their books to occupy seats. However, some of the occupied students took up the seat but did not return, causing other students to find no place to sit. This kind of behavior is selfish behavior. Students who can't find a seat are reflected in our librarians, which makes us very difficult. Therefore, in order to prevent this uncivilized behavior from happening and give students a better learning atmosphere and environment for the library, we have implemented some programs.

The first option, regional management. Each floor of the library is divided into several areas, each area is arranged for management. This kind of supervision program can effectively take care of the seats in each area. Once the occupants are found not to return, they can place their items in the custodial office and clean the table to the next person who needs to be used. However, this program requires a large number of people, so we decided to use the students as temporary workers to solve personnel problems. In the second program, the students of the postgraduate study arrange the study place independently. We found that many college students have plans for postgraduate study, so we believe that the students in the postgraduate study can be arranged in the library's self-study room to reduce the occupancy. For the above two scenarios, we try to simulate them.

The simulation results are not satisfactory. The problem with the first solution is the relationship between cost and students. In terms of cost, there are a large number of student workers, so more money is needed for them. And these student workers can also use their rights to take seats for others, which is out of our minds. The problem with the second option is that there is no substantial change. During the appointment of the postgraduate study room seating activity, the number of appointments exceeded the number of seats we expected. After we determined that the line was full, there were still a lot of students stranded. This is beyond our imagination and is also pleased with the learning power of today's college students.

The failure summary of the two programs prompted us to design a seat reservation system to solve the problem. The library seat reservation system is to allow students to be bound to the seat. When the student leaves the library, the seat is unlocked for use by the next classmate. In terms of cost, the price of our production system and machine is lower than the labor cost of the first solution. Moreover, the machine will only provide seats for students who enter the library and swipe their cards, and there will be no problem of inequality in distribution. We also advocate that the postgraduate study room coexists with this system, which not only solves the problem of too many students, but also can reasonably allocate the seats of the library. This is the library seating reservation system we need.

With the creation of the seat reservation system, the library's seating resources can be utilized. If the student wants a seat, he or she must choose a seat in the selection machine and can only carry the student card to complete. The student information is then bound to the seat for a limited period of time. In other words, this seat is the student's lifetime. This will prevent students from sitting in the seat. During the use of the seat by the student, if someone finds something else in the seat, the item can be carried to other public areas and the item must be completed. If the student finds someone else sitting in the seat, they need to declare that they have reserved the seat and asked the occupant to leave. If the occupant does not agree, he can ask for help from a nearby service person. The system can well meet the personal preferences of the majority of students, they can choose their favorite location. It is convenient and fast, and it can reflect the humanized service of the library. The seat reservation system then has precise management of the library's seats. The administrator can view the seat information at any time and record the flow of people on the day to facilitate finding and meeting the work needs. Finally, in the case of a large number of people, it can avoid the situation of random seating. Because students who need a seat must go to the landline to make a seat reservation

1. **Glossary of Terms**

1.Capsule:A specific design pattern that represents the encapsulated control threads in the system. A package is a class that has been given a stereotype that has a specific set of required and defined associations and characteristics.

2.Cardinality:The number of elements in the element set. Contrast: multiplicity.

Causal analysis:Track down the cause of the problem and determine the solution.

3.CBD:Component-based development

4.CCB:Change control committee

5.CDR:Key design review

6.CGI:Common gateway interface

7.Change control board (CCB):The role of the CCB is to provide a centralized control mechanism to ensure that each change request is properly considered, approved, and coordinated.

Change management:Control and track the activity of artifact changes. See also scope management.

8.Change request (CR):A general term for any request by a stakeholder to change an artifact or process. The information recorded in the change request is information about the current issue, the proposed solution, and the origin and impact of its cost. See also extension request, defect.

9.Checklist

10.Checkpoints:A set of conditions that a well-organized artifact should have. A question and answer form that should be answered affirmatively can also be used.

11.Class:A description of a set of objects with common attributes, operations, methods, relationships, and semantics. A class can use a set of interfaces to specify the set of operations it provides to its environment. See interface.

12.Class diagram:A diagram showing a set of descriptive (static) model elements, such as classes, types, and their content and relationships.

13.Class hierarchy:The relationship between classes that share a single inheritance. All Java classes inherit from the Object class.

14.Class library:A collection of classes.

15.Class method class method

16.Classifier:A mechanism for describing behavioral and structural characteristics. Classifiers include interfaces, classes, data types, and artifacts.

17.Client:A classifier that requests services from other classifiers. Contrast: supply (supplier).

18.Client/server:An interaction model in distributed data processing, where a program at one location issues a request to a program at another location and waits for a response. The program that makes the request is called the client program, and the answering program is called the service program.

**3.System Requirements**

3.1**Enumerated Functional Requirements**

Students can use the seat management system to find spare seats on the touch screen to lock the seat and release the seat when they leave. The login is done by swiping the campus card.

The main application of the program has the following characteristics: 1. Open the inquiry service for students, students can query, lock and release the seat through the touch screen; 2. Get the entry and exit information of the students through the touch screen terminal. The system automatically completes the user's seat lock and release seat action based on this information;

1. **Check the seat**

Students can log in to the seat management system to check if there are available seats through any touch screen terminal connected to the campus network. The system provides two ways to query, one is to query according to the classroom seating layout, and the other is to customize the filtering method. Can meet the different needs of students

2. **Lock the seat**

When there are still seats available in the venue, students can choose between quick match (automatic system assignment) or self-selection (manual selection) of the seat. After the seat is successfully determined, the system will provide an electronic voucher to completely solve the seat problem through the "voucher" method.

The voucher can be printed on the touch screen or taken with a mobile phone.

3. **Cancel the seat**

If you need to cancel your seat, you can check the record in ‘My Seat’ and cancel it within the specified time. The administrator can change the number of cancellations and how long before they can be cancelled.

4. **Release the seat**

When the student ends using the seat, leave the library to automatically release the seat

**Management system**

The administrator can set various data of the system through the management system, and can store and delete seat information.

User system

Users can achieve the following functions through the system:

1 check the seat;

2. Lock the seat;

3. Cancel the seat;

4. Release the seat.

3.2.**Enumerated Nonfunctional Requirements**

1. Easy to manage (reasonable allocation of seats, improve work efficiency)

2. Processing information fast

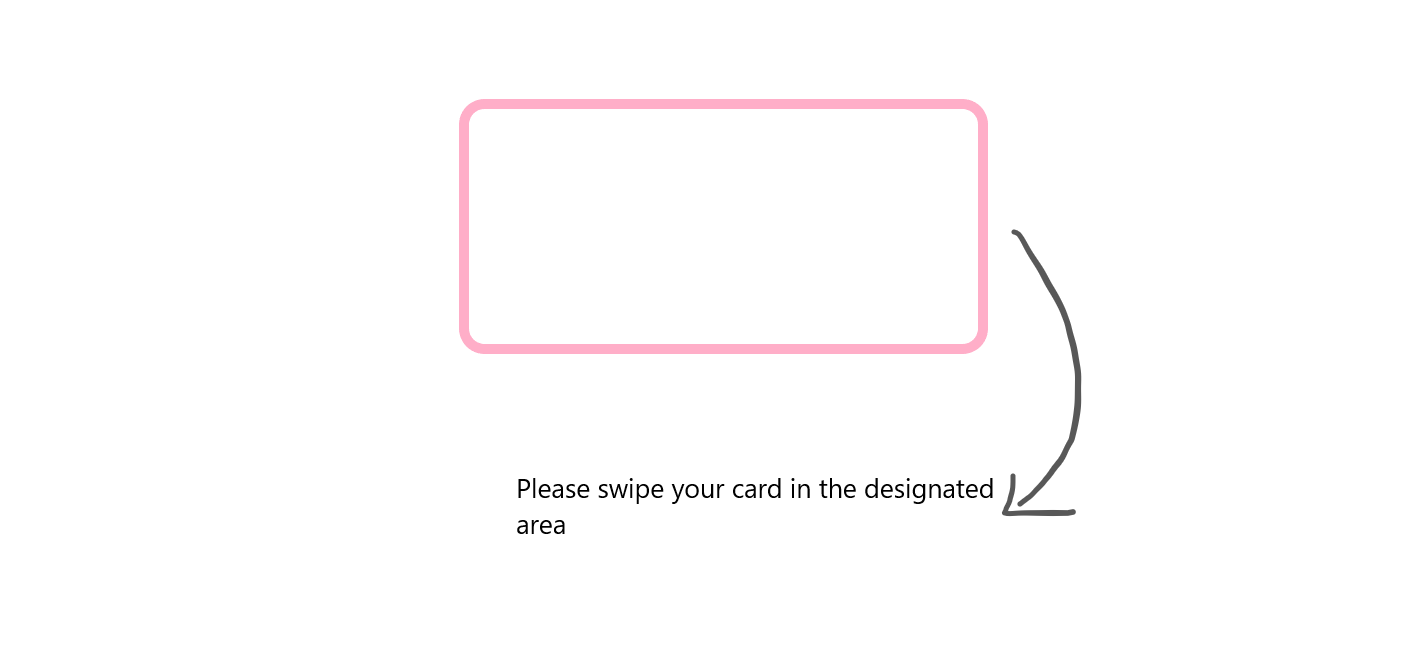
3. Saving labor

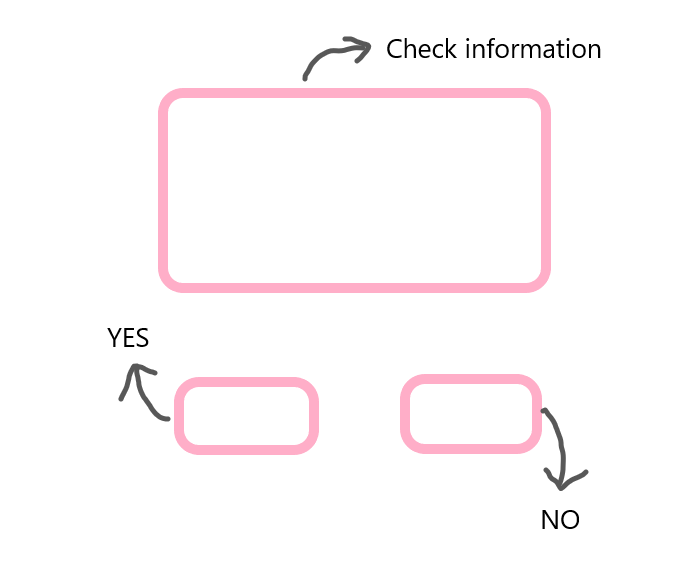
4. Information is not public

5. **Applicable to other similar libraries**

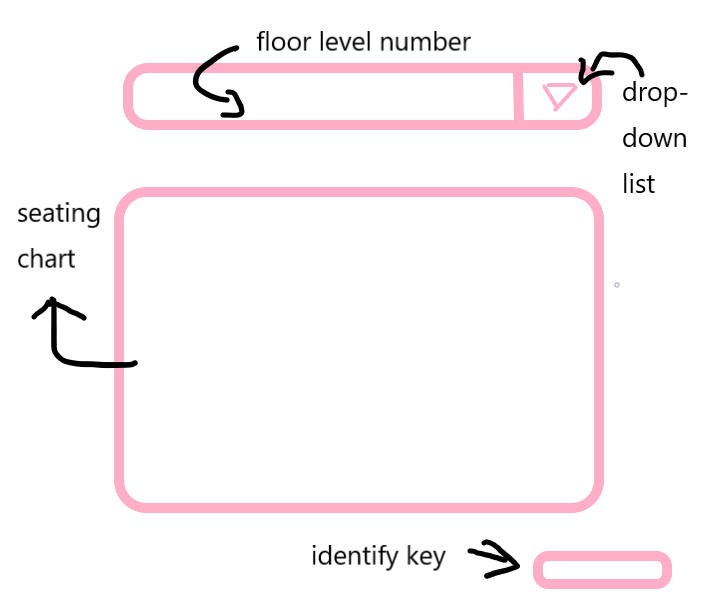
Many libraries do not have a reasonable system for assigning seats, which leads to students sitting in a mess. This system allows the library to have more seats available to accommodate more students. Most libraries require a lot of human resources to collect empty seats. The information is wasting a lot of unnecessary time. It can also be applied to places like libraries, such as restaurant ordering, bank queuing appointments. Some libraries' information is open to the public. People can check other people's information at any time. The information in this system is not public. Only those with sufficient authority can check private information (such as ID number, gender, age) and ensure that Information security. Many library systems process information very slowly, and users need to wait to get results. This system can process information quickly. If the information is correct, the user can quickly enter the software for use. Otherwise, the user will quickly get the information that he or she entered incorrectly.

3.3.**User Interface Requirements**

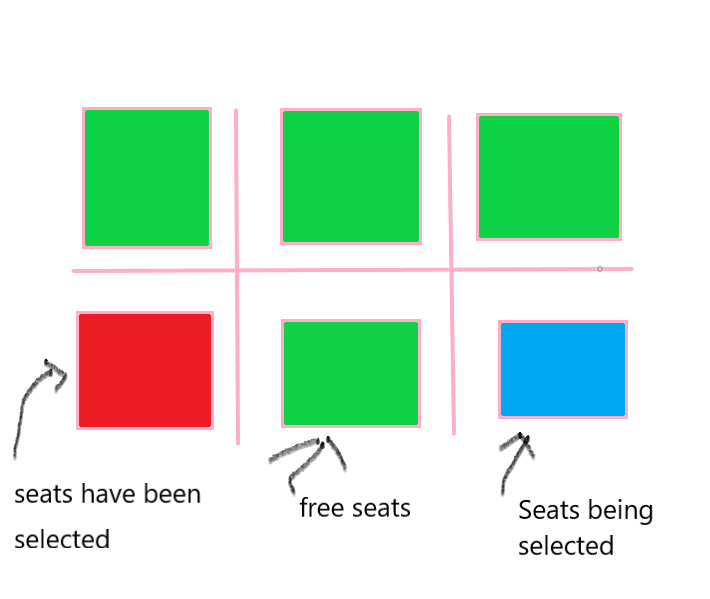




The above picture shows the student swiping the card, and the student can confirm the information on this page.



Students can see the number of library levels currently viewed in the top bar. The initial default is 1 floor. The inverted triangle on the right side can be popped up with a drop-down box. You can select the floor, the lower part is the library seat display, and the lower right corner is the confirmation button. After selecting the seat, the student clicks the button to submit the data. If the background shows that someone has selected the seat, a pop-up window will pop up to show that the seat has been selected, and after a few seconds, the page will automatically bounce back to the page again to select the page data. It will be refreshed based on the data in the database. If the selection is successful, the page will pop up and the selection will succeed. After a few seconds delay, it will automatically bounce back to the student landing page.



This is an example of selecting a seat page in the above picture. The seat that has been selected will display red, the free seat will display green, and when the green seat is selected by the user, it will display blue.

**User Effort Estimation**

a. Open the login interface;

b. Click on the username;

c. Enter the username;

d. Click on the password;

e. Enter the password;

f. Click OK to enter the seat selection interface;

g. Select a floor;

h. choose a seat;

i. Click OK to complete the reserved seat;

j. Exit the login interface.

Mouse clicks required for the task: 7

User interface navigation and document input 2 parts

**4.Functional Requirements Specification**

**Stakeholders**

●Users ○ Users can check the use of seats in a timely and convenient manner to prevent seats from being found.

●Librarian ○ Work on seat management, personnel management, and maintenance management

**Actors and Goals**

  ● User (initiated): ○ The user uses the username and password to perform online library selection activities on the platform.

● Database (participation): ○ Database provides data support for the system, enabling students to query library seating information

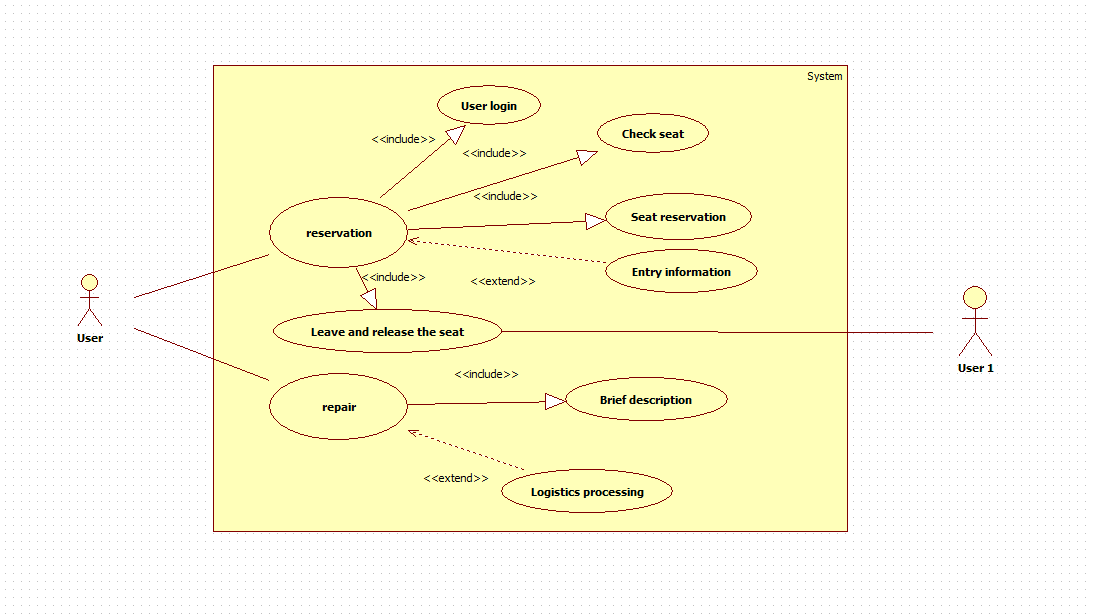
● Library login system (participation): ○ The library system will send student information to the system when the student leaves the library, so that the system can update the database in time.

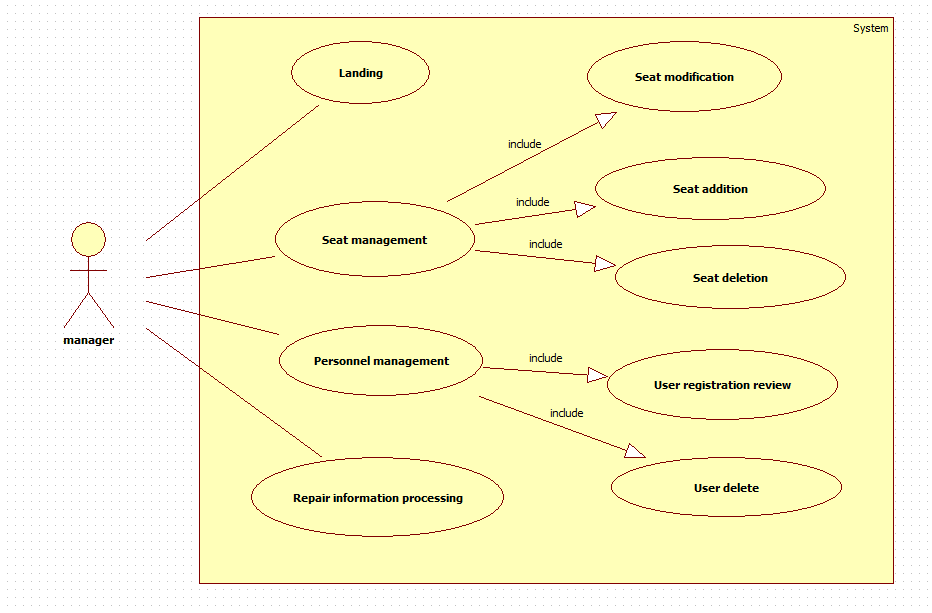
**Casual Description**

The user enters the seat selection system, and the personal information is logged in through the background for review. After the login is successful, the database feedback seat information is selected and locked by the user, and the seat is released through the background when the user leaves. If the seat or other functions are damaged, a repair system is provided, and the user asks a question, which is processed by the background and then manually processed.

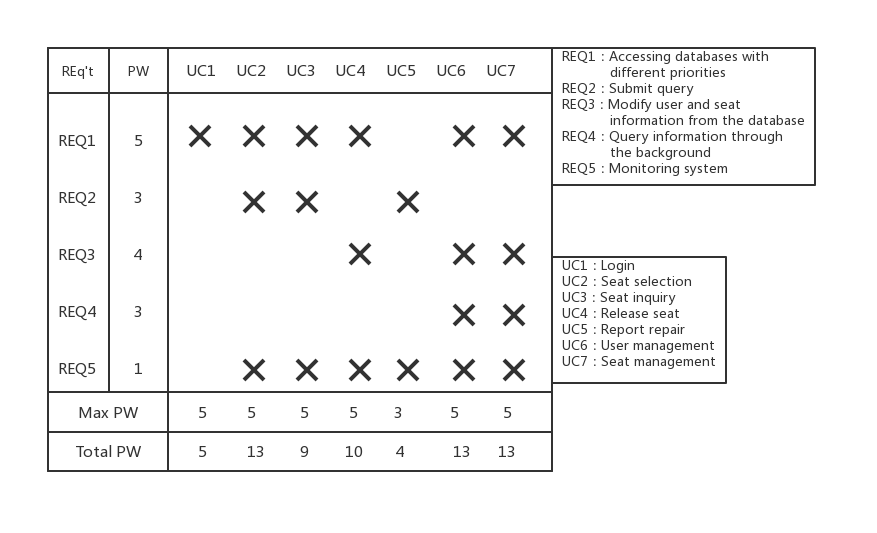
The administrator enters the system through the background login and is responsible for seat management management and warranty information processing. The addition and deletion are modified by calls between different data controls, and monitoring and feedback activities are performed in the background.

**Use Case Diagram**

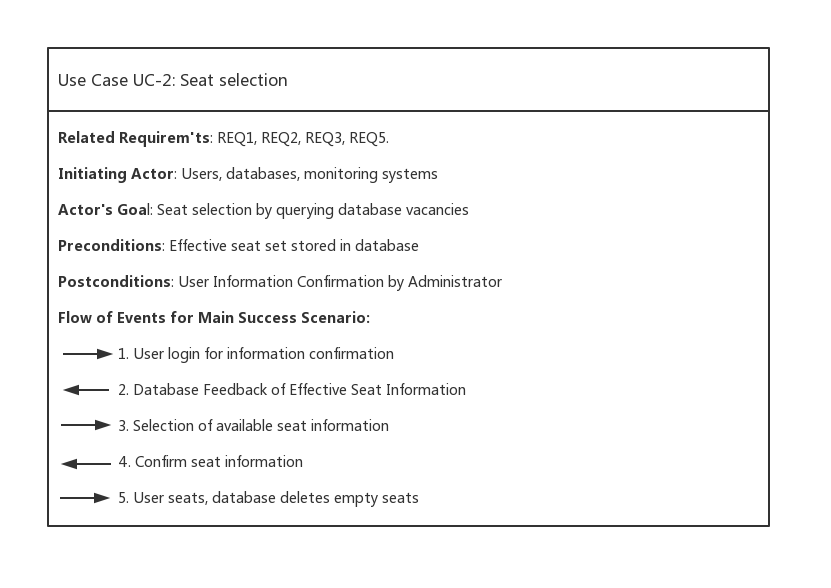


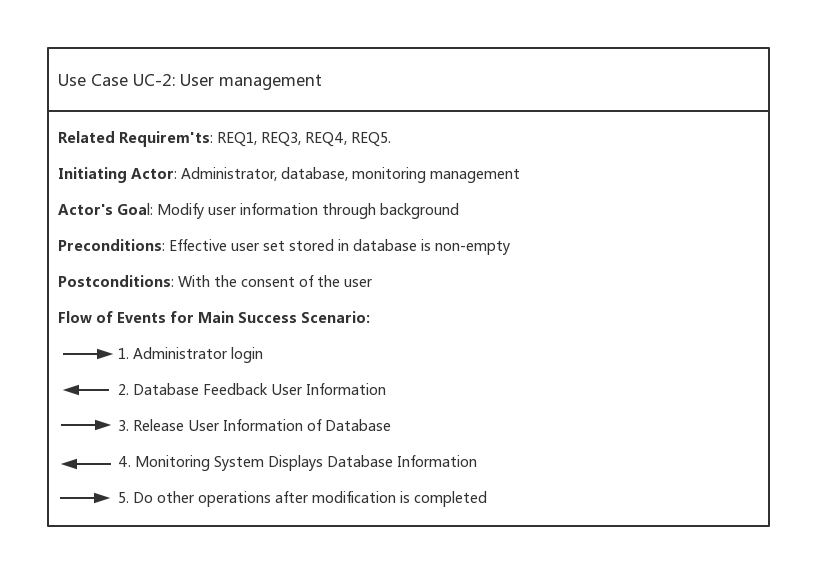


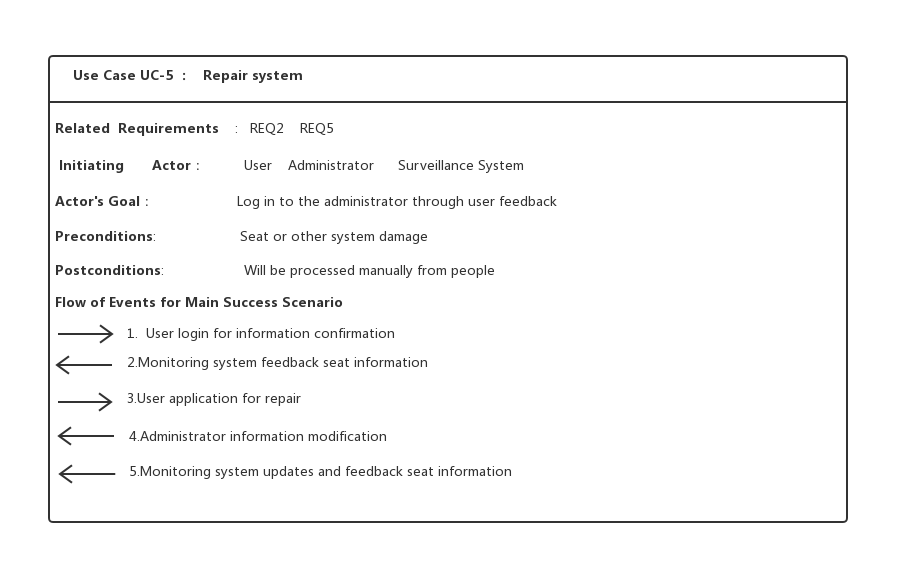
**Traceability Matrix**



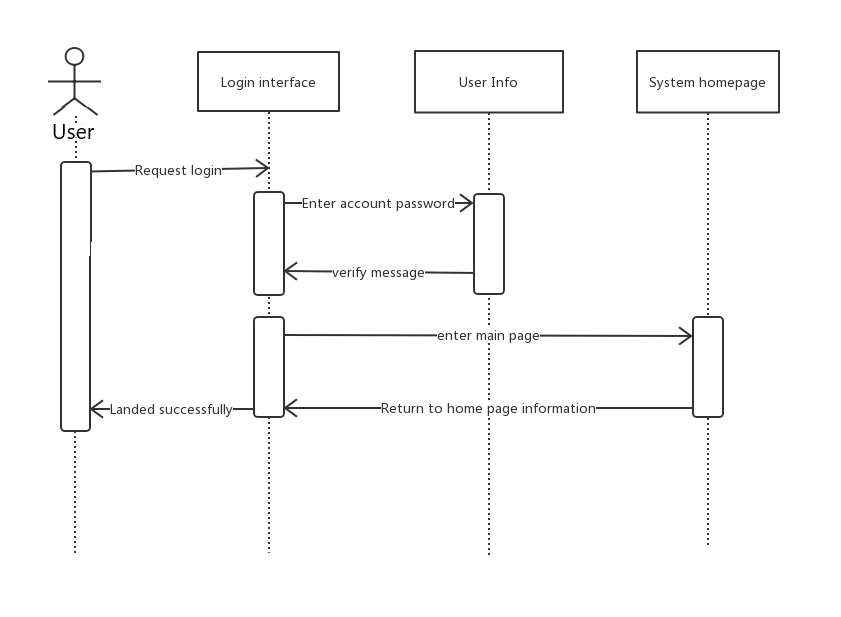
**Fully-Dressed Description**

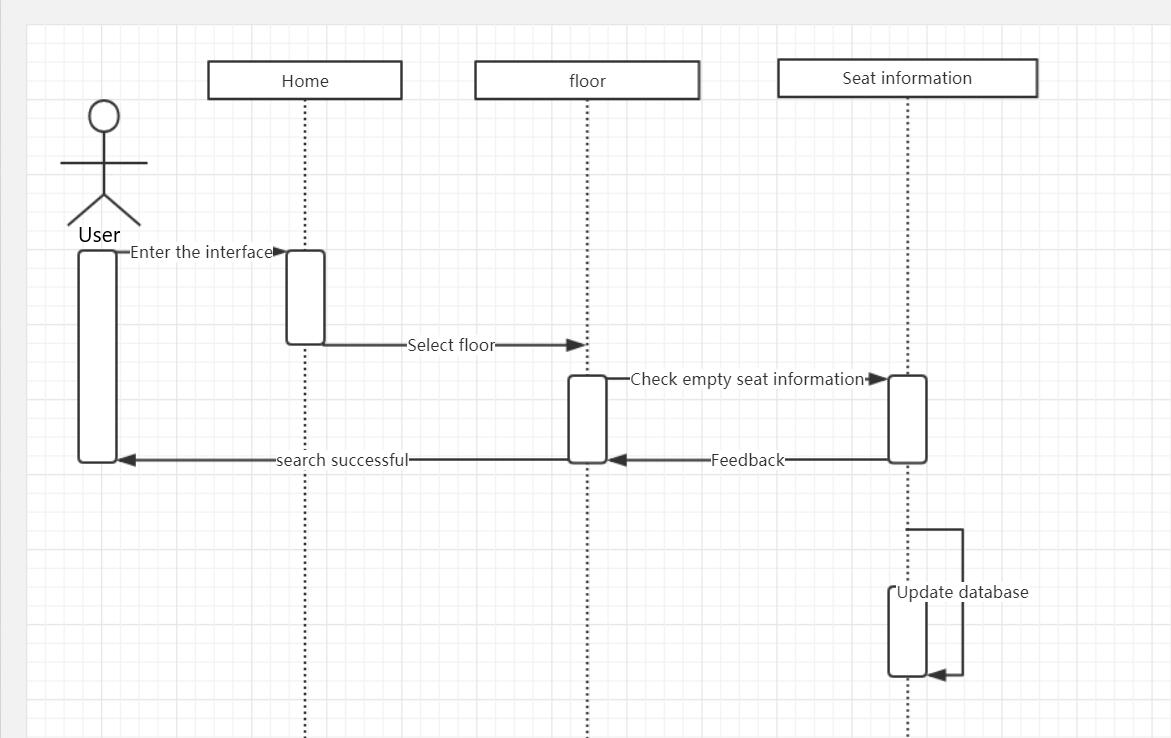


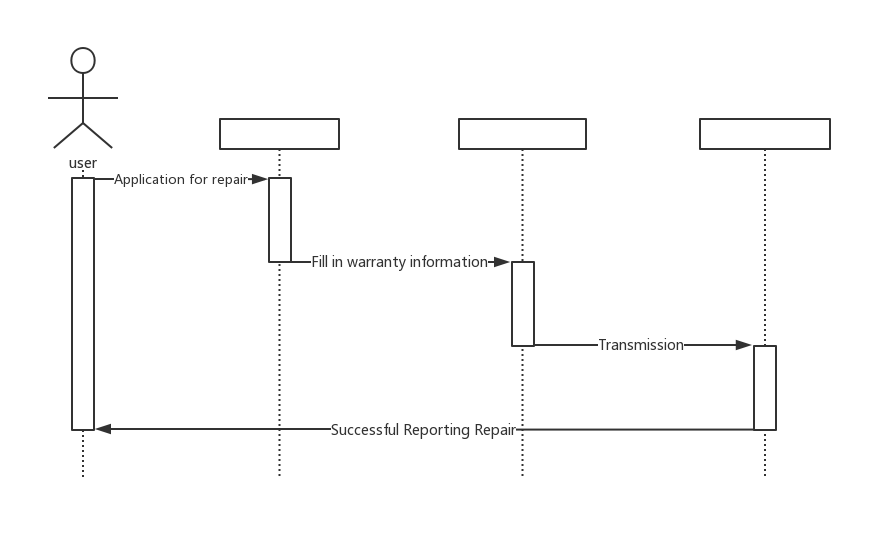
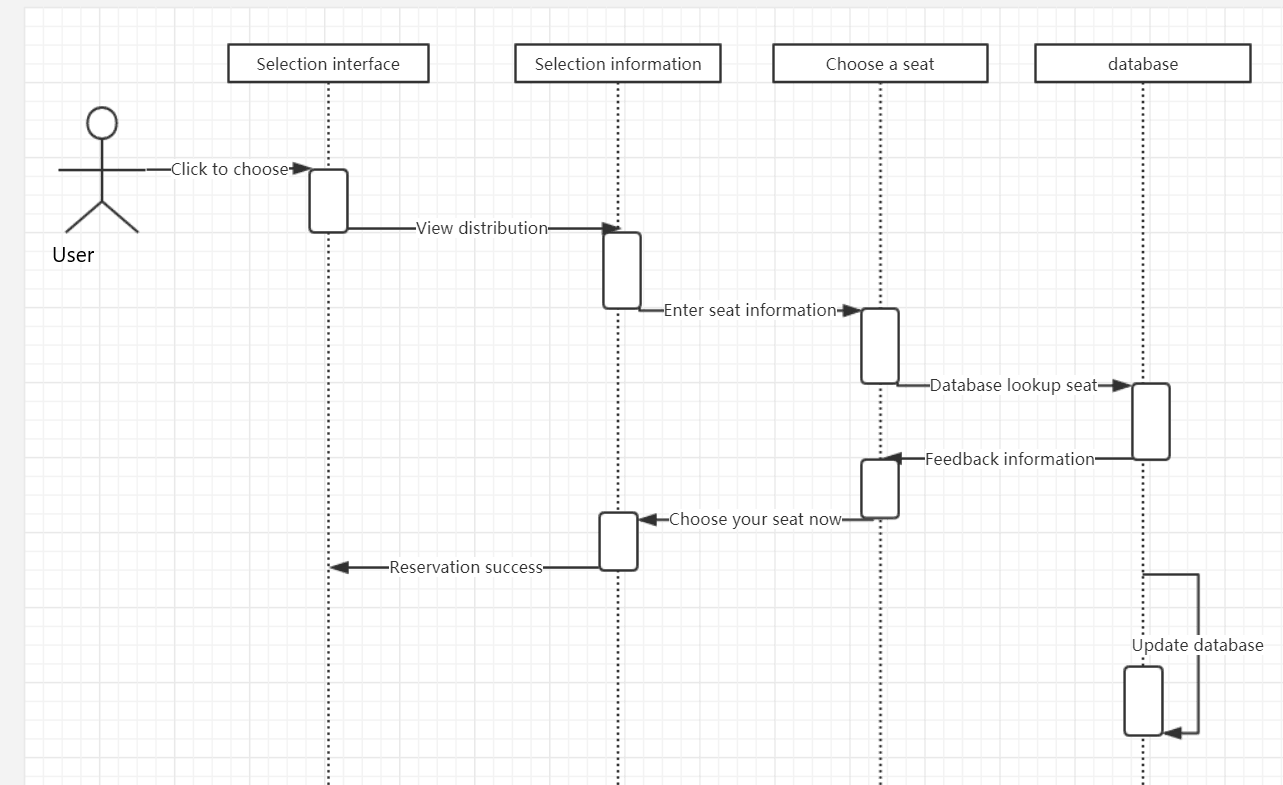


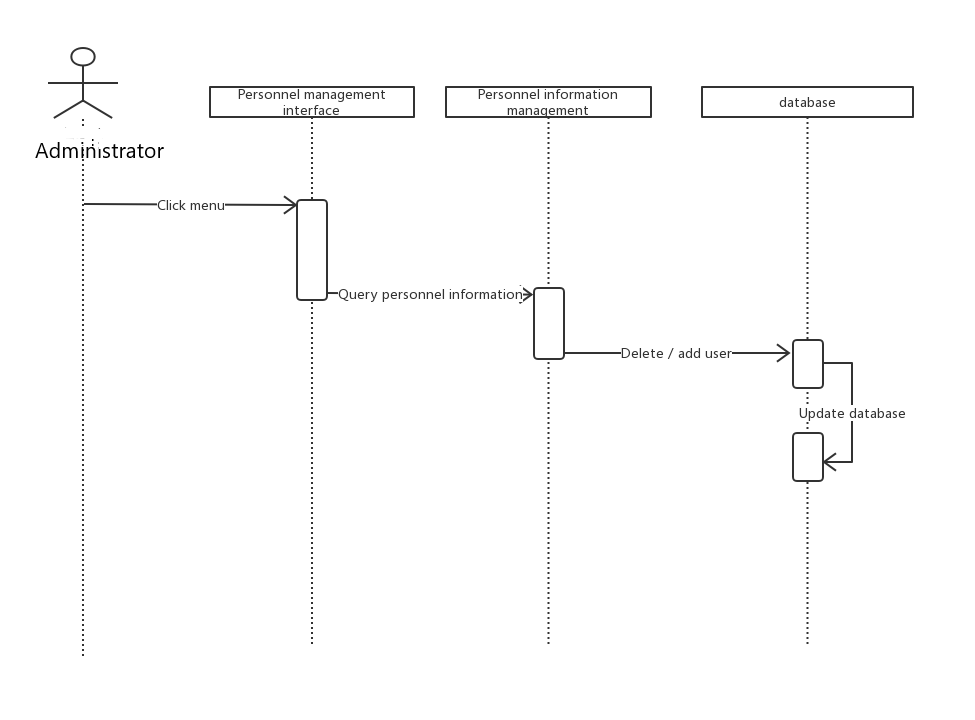
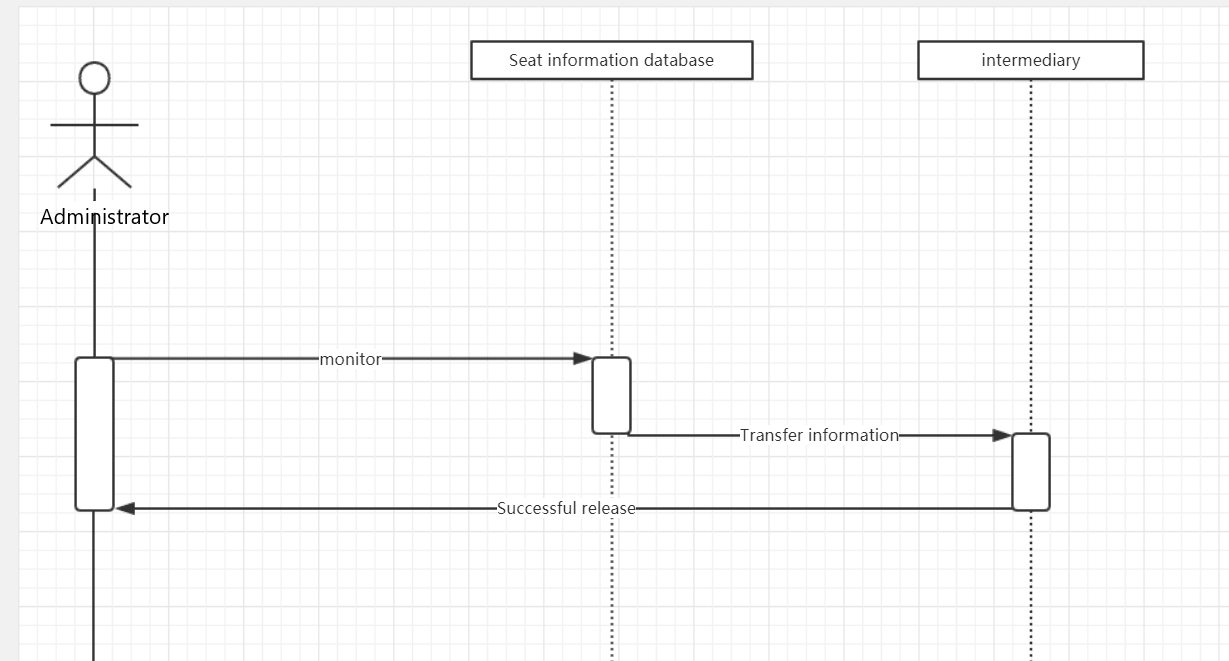


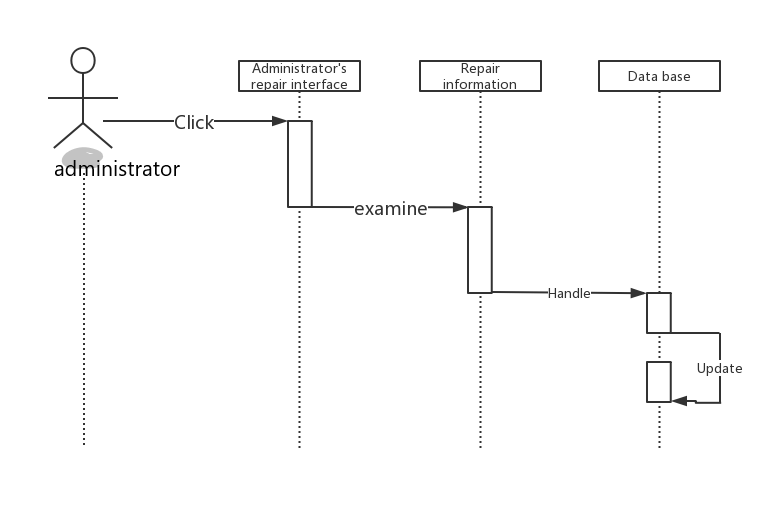
**System Sequence Diagrams**











**5.Effort Estimation using Use Case Points**

a. Open the login interface;

b. Click on the username;

c. Enter the username;

d. Click on the password;

e. Enter the password;

f. Click OK to enter the seat selection interface;

g. Select a floor;

h. choose a seat;

i. Click OK to complete the reserved seat;

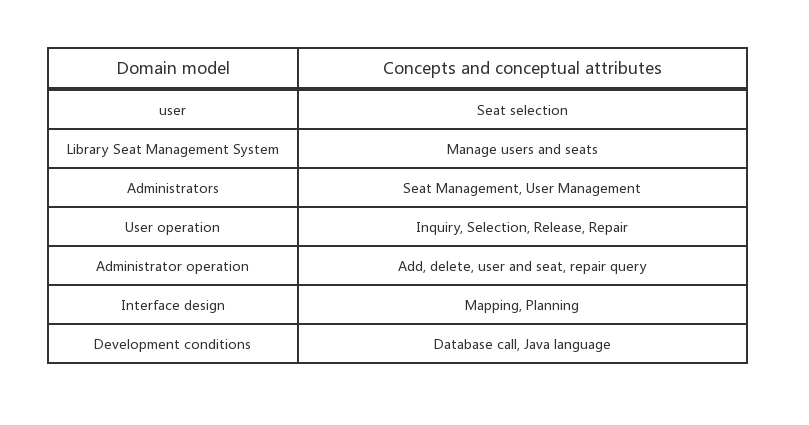
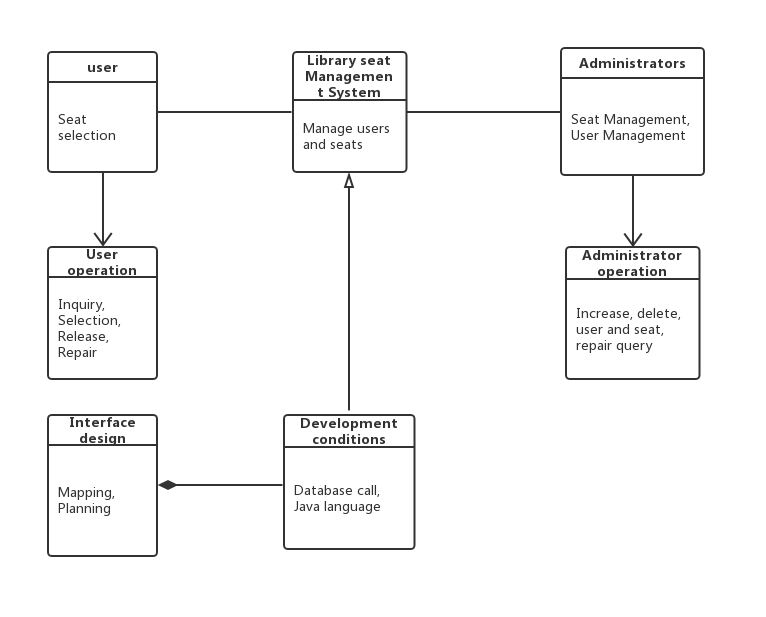
j. Exit the login interface.

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor name** | **Description of relevant characteristics** | **Complexity** | **Weight** |
| Admin | Admin is interacting with the system via a graphical user interface (when managing users on the central computer). | Complex | 3 |
| User | Tenant is interacting through a text-based user interface | Average | 2 |
| User and Seat management | User and Seat management isused for interaction between administrators and users | Simple | 1 |
| Database | Database is another system interacting through a protocol. | Average | 2 |

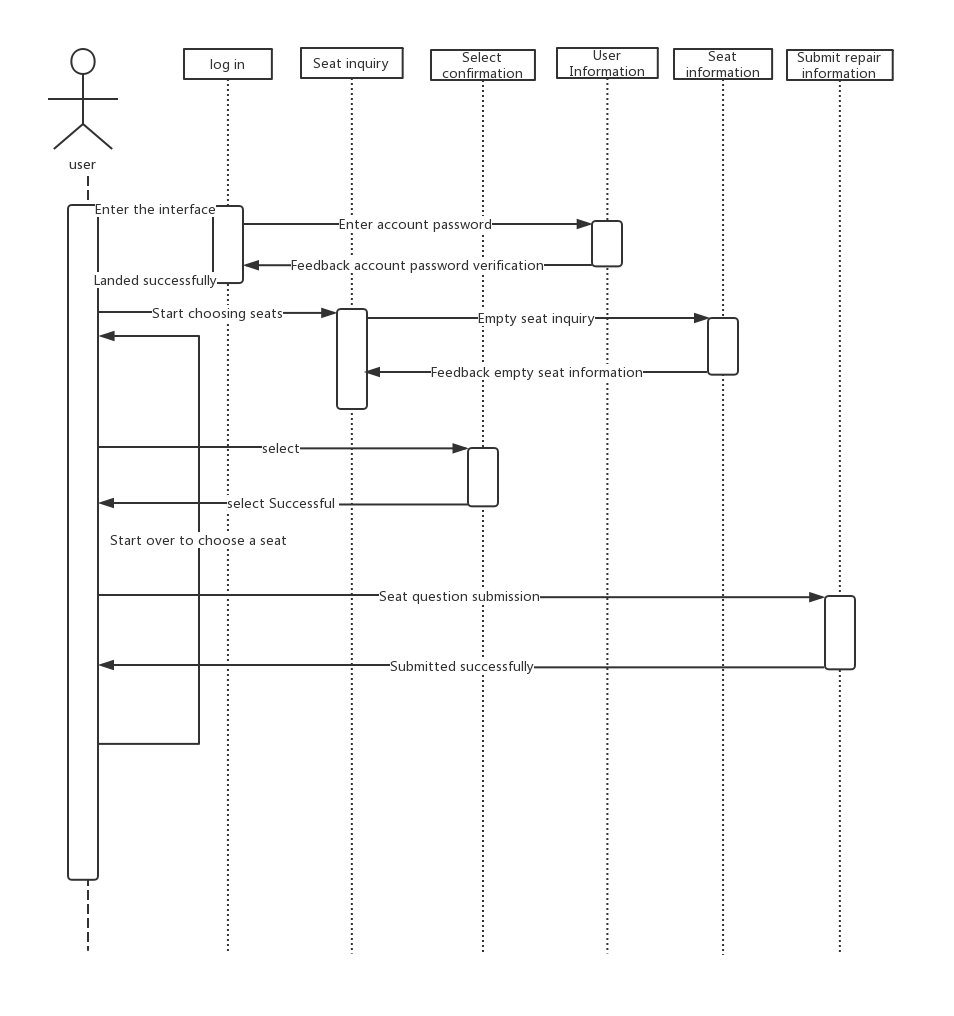
|  |  |  |  |
| --- | --- | --- | --- |
| **Use case** | **Description** | **Category** | **Weight** |
| Login (UC‑1) | Simple user interface. | Simple | 5 |
| Seat select(UC‑2) | Simple user interface. | Simple | 5 |
| Seat inquiry (UC‑3) | Average user interface. | Average | 10 |
| Release seat (UC‑4) | Complex user interface. | Average | 10 |
| Report repair (UC‑5) | Simple user interface. | Simple | 5 |
| User management (UC‑6) | Complex user interface. . | Average | 10 |
| Seat management(UC‑7) | Complex user interface. | Average | 10 |

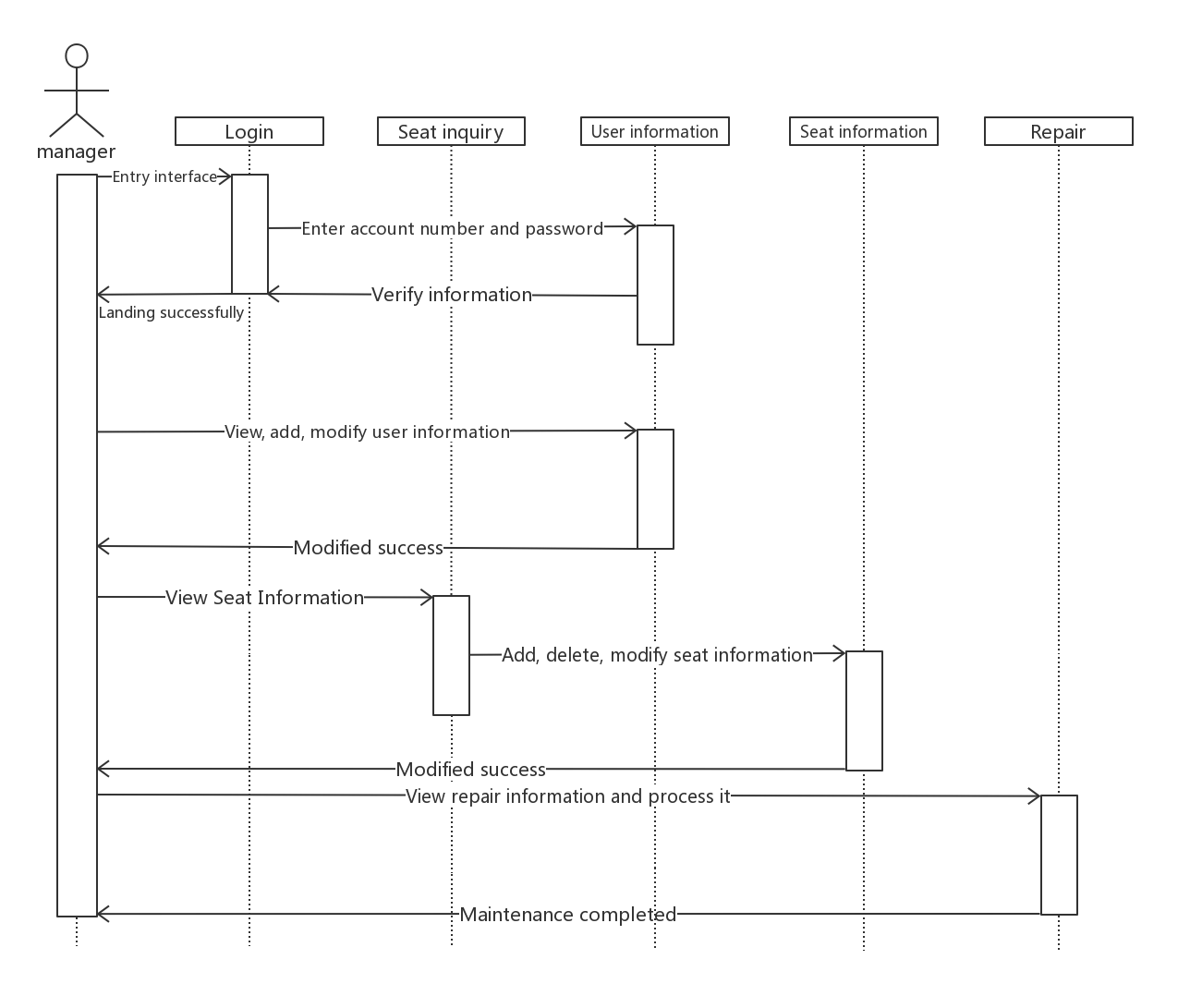
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Technical factor** | **Description** | | | **Weight** | | **Perceived Complexity** | **Calculated Factor (WeightPerceived Complexity)** | |
| T1 | Can only be operated on the internal network | | | 2 | | 0 | 2x0 = 0 | |
| T2 | Refresh at any time | | | 1 | | 7 | 1x7 = 7 | |
| T3 | User expects efficiency | | | 1 | | 1 | 1x1 = 1 | |
| T4 | Simple and average | | | 1 | | 1 | 1x1 = 1 | |
| T5 | No requirement for reusability | | | 1 | | 0 | 1x0 = 0 | |
| T6 | Ease of install is moderately important | | | 0.5 | | 3 | 0.5x3 = 1.5 | |
| T7 | Ease of use is very important | | | 0.5 | | 7 | 0.5x5 = 2.5 | |
| T8 | No portability concerns beyond a desire to keep database vendor options open | | | 2 | | 2 | 2x2 = 4 | |
| T9 | Easy to change minimally required | | | 1 | | 1 | 1x1 = 1 | |
| T10 | Concurrent use is required | | | 1 | | 0 | 1x0 = 0 | |
| T11 | Security is a significant concern | | | 1 | | 7 | 1x5 = 5 | |
| T12 | No direct access for third parties | | | 1 | | 0 | 1x0 = 0 | |
| T13 | No unique training needs | | | 1 | | 0 | 1x0 = 0 | |
| Technical Factor Total: | | | | | | | 23 | |
| **Environmental**  **factor** | | | **Description** | **Weight** | | **Perceived Impact** | | | **Calculated Factor (Weight**  **Perceived Impact)** | |
| E1 | | | Familiar with the development process | 1.5 | | 6 | | | 1.5 ✖ 6 = 9 | |
| E2 | | | Application problem experience | 0.5 | | 6 | | | 0.5 ✖ 6 = 3 | |
| E3 | | | Paradigm experience | 1 | | 3 | | | 1✖ 3 = 3 | |
| E4 | | | Lead analyst capability | 0.5 | | 2 | | | 0.5 ✖ 2 = 1 | |
| E5 | | | Motivation | 1 | | 3 | | | 1✖ 3 = 3 | |
| E6 | | | Stable requirements | 2 | | 5 | | | 2✖ 5 = 10 | |
| E7 | | | No part-time staff will be involved | 1 | | 0 | | | 1✖ 0 = 0 | |
| E8 | | | Difficult programming language | 1 | | 0 | | | 1✖ 0 = 0 | |
| Environmental Factor Total: | | | | | | | | | 30 | |

1. **Domain Analysis**



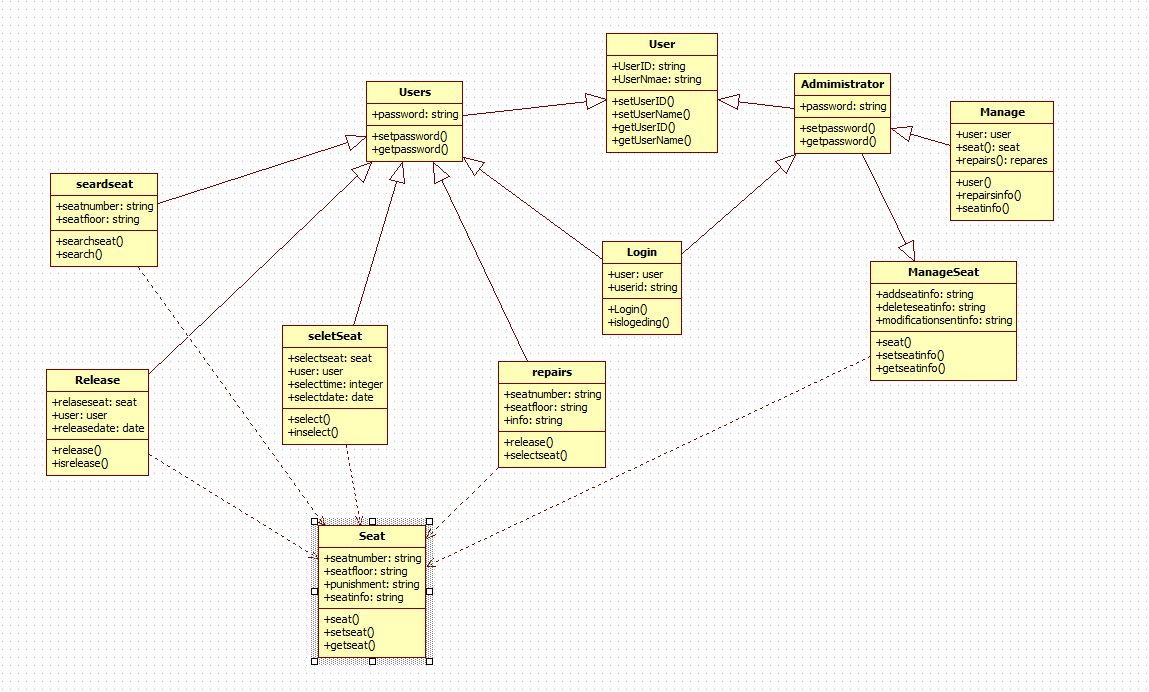
1. **Interaction Diagrams**





**8.Class Diagram and Interface Specification**

**Class Diagram**



**Data Types**

|  |  |
| --- | --- |
| Data Types | Function |
| Int | 32-bit, the maximum data storage capacity is 2 to the 32th power minus 1, and the data range is negative 2 to 31 power to positive 2 to 31 power minus one. |
| float | 32-bit, the data range is 3.4e-45~1.4e38. When directly assigning, you must add f or F after the number. |
| char | 16-bit, storing Unicode code, assigned with single quotes. |
| byte | 8 bits, the maximum amount of stored data is 255, and the stored data range is between -128 and 127. |
| short | 16 bits, the maximum data storage is 65536, and the data range is between -32768 and 32767. |
| long | 64-bit, the maximum data storage capacity is 2 to 64th power minus 1, and the data range is negative from 2 to 63 to the positive 2 to 63 to minus 1. |
| double | 64-bit, the data range is 4.9e-324~1.8e308, you can add d or D or not. |
| boolean | There are only two values for true and false. |

**Operation Signatures**

|  |  |
| --- | --- |
| class | Function |
| User | Get the value of the user attribute name by the method getName(), and set the value of the user attribute name by the method setName() |
| Login | By verifying the user account password, it is determined whether the login can be successful. |
| Student | Read student information. |
| SearchSeat | Query the database seat information to display the available seats. |
| Repairs | Create repair information and confirm seat information. |
| Amininistrator | Read administrator information. |
| ManageSeat | The administrator adds, deletes, and modifies seat information. |
| Seat | Store seat information and record seat number, floor, and information. |

**Traceability Matrix**

|  |  |  |
| --- | --- | --- |
| Login | Accessing databases with different priorities | Determine the permissions of the consumer to distinguish different priorities |
| User seat operation | Accessing databases with different priorities | The logon user data is divided by priority to avoid program deadlock. |
| Submit query | Upload input instructions to the server for further action |
| Monitoring system | Global monitoring to avoid delays in database docking or management problems caused by system vulnerabilities |
| Modify user and seat information form the database | According to the instructions yong entered, the data will be called |
| Report repair | Submit query | Upload input instructions to the server for further action |
| Monitoring system | Global monitoring to avoid delays in database docking or management problems caused by system vulnerabilities |
| User and Seat management | Accessing databases with different priorities | Modify information with administrator privileges |
| Modify user and seat information form the database | The administrator operates on the information and usage information |
| Query information through the background | Hierarchical partitioning |
| Monitoring system | Monitor inappropriate operations |

|  |  |
| --- | --- |
| **OCL Notation** | **Meaning** |
| OPERATIONS ON ALL OCL COLLECTIONS | |
| c->size() | Returns the number of elements in the collection c. |
| c-> doGetLoginController () | Returns true if c has no elements, false otherwise. |
| c1->MySeatController(c2) | Returns true if every element of c2 is found in c1. |
| c1-> UserReviewController (c2) | Returns true if no element of c2 is found in c1. |
| c->boolean flag(id) | Returns true if the Boolean expression expr true for all elements in c. As an element is being evaluated, it is bound to the variable id, which can be used in id. This implements universal quantification ∀. |
| c-> MySeatDao (var1, var2 | expr) | Same as above, except that expr is evaluated for every possible *pair* of elements from c, including the cases where the pair consists of the same element. |
| c->exists(var | expr) | Returns true if there exists at least one element in c for which expr is true. This implements existential quantification ∃. |
| c-> JDBCUtil (var | expr) | Returns true if expr evaluates to a different value when applied to every element of c. |
| c-> UserService (expr) | Returns a collection that contains only the elements of c for which expr is true. |
| OPERATIONS SPECIFIC TO OCL SETS | |
| s1->information(s2) | Returns the set of the elements found in s1 and also in s2. |
| s1-> Message(s2) | Returns the set of the elements found either s1 or s2. |
| s->Seat(x) | Returns the set s without object x. |
| OPERATION SPECIFIC TO OCL SEQUENCES | |
| seq->first() | Returns the object that is the first element in the sequence seq. |

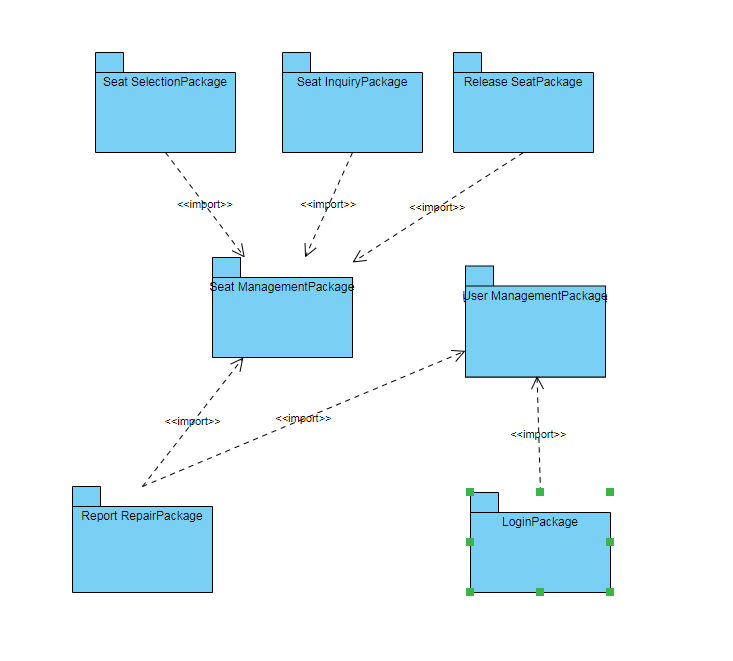
**Accessing Collections in OCL**

**9.System Architecture and System Design**

**Architectural Styles**

Class driver architecture - the architectural style is mainly depends on the class to manipulate data call, reaction, as well as to the changes in the database data, as a result of our system is an interactive system, mainly depends on the user to the operation of the database, and call the database data, the system must be able to make a timely response to the operation of the user.

**Identifying Subsystems**



**Mapping Subsystems to Hardware**

Need to run on at least two computers, seat selection, seat inquiry, release seat, report repair running on one computer, user management and seat management running on another computer

**Persistent Data Storage**

The library reservation seat system requires documents to have a long save time, so the group decided to adopt the relational database model. The suffix used in this mode is named **.mdf** as the main file, which is used to save the main data in the database. Another subfile is a log file with a suffix of **.ldf**, which records the database log and the database's operation record. The combination of the two will perfectly run the persistent data storage of the library management system.

**Network Protocol**

The network protocol used by the library seat reservation system is JDBC. The Java database connection architecture is the standard way for Java applications to connect to databases. With the JDBC API, team members don't have to write a program for accessing the Sybase database, write a program for accessing the Oracle database, or write another program for accessing the Informix database. Team members simply need to write a program using the JDBC API, which can send SQL calls to the appropriate database. At the same time, the combination of the Java language and JDBC makes it easy to write different applications for different platforms. It can be run on any platform by writing a program. As an API, JDBC provides a standard interface for program development. JDBC uses existing SQL standards and supports standards for connecting to other databases. JDBC implements all of these standards-oriented targets and interfaces with simple, strict type definitions and high performance implementations.

Once the JDBC connection is established, it can be used to transfer SQL statements to the database it is involved in. JDBC does not impose any restrictions on the type of SQL statement being sent. This provides a great deal of flexibility in allowing the use of specific database statements or even non-SQL statements. The method that executes the SQL statement is executeQuery. This method is used to execute a SELECT statement.

**Global Control Flow**

The library seating reservation system is executed in a "linear" manner, and each user needs to perform the same operation, waiting for the next user or step in the loop.

The library reservation seat system is a real-time system, which is periodic. It can start to enter the system login interface at 6 o'clock every morning, and end all user operations as of 10 o'clock in the evening. Then organize the data for all the operations that took place today.

Our system uses single-process multi-thread mode. The objects with independent control threads are only administrators. The synchronization between threads can query all the information in the database at the same time. The information is correctly carried out in the next step, and the information error returns to the first step. You need to do the previous work again. After the step is over, the next user needs to operate the same steps. Data synchronization between threads.

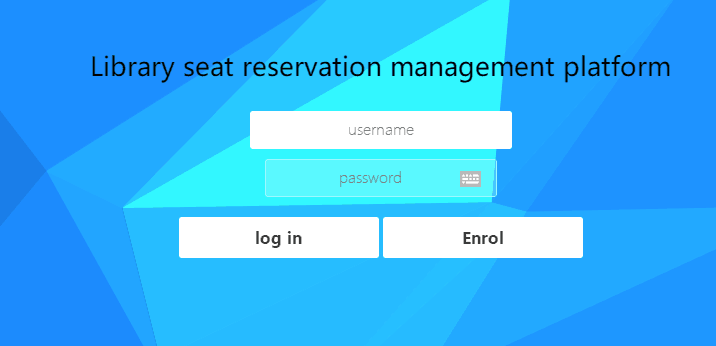
**Hardware Requirements**

Hardware requirements: 2 GM for 2.4 GHz dual core processor or higher processor RAM

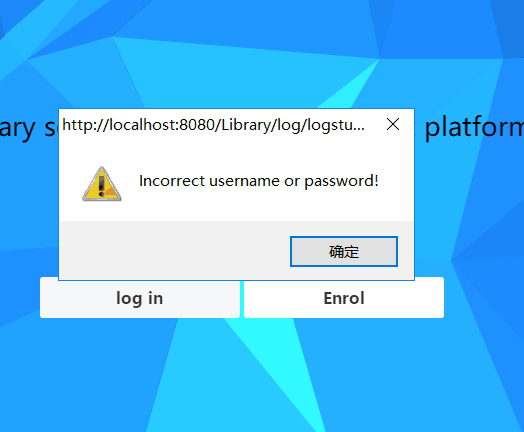
LAN connection Windows operating system 7 screen Minimum 720P Minimum hard disk space: 128MB

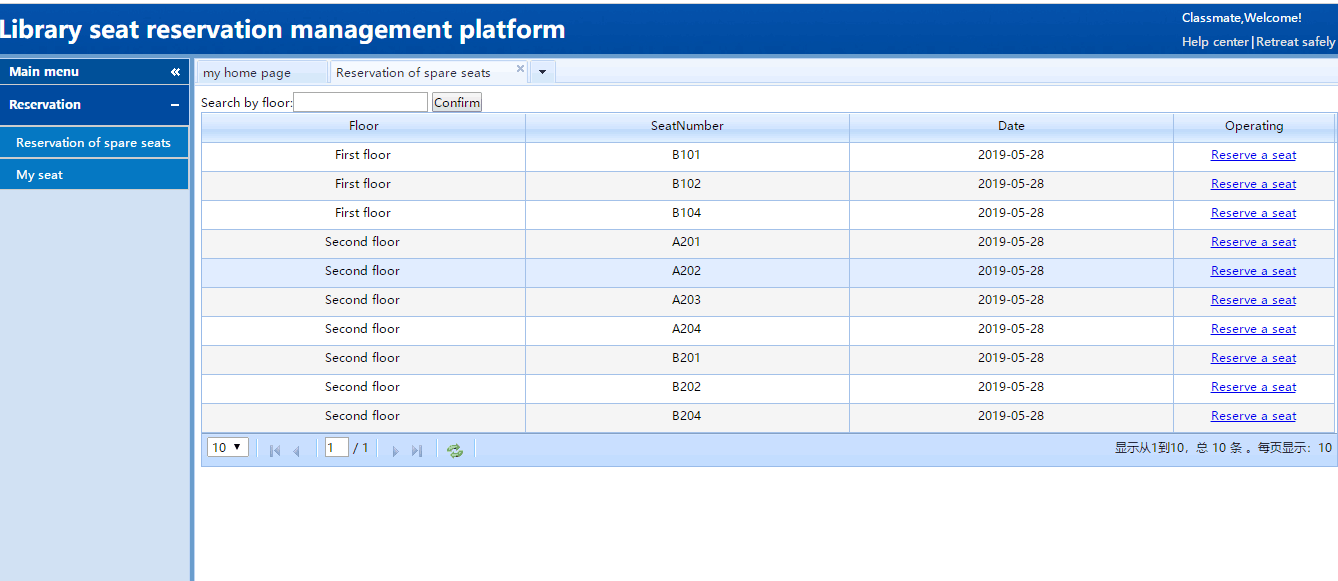
1. **Algorithms and Data Structures**

This program only involves the addition, deletion and change of the database, and does not involve algorithm operations.

**11.User Interface Design and Implementation**

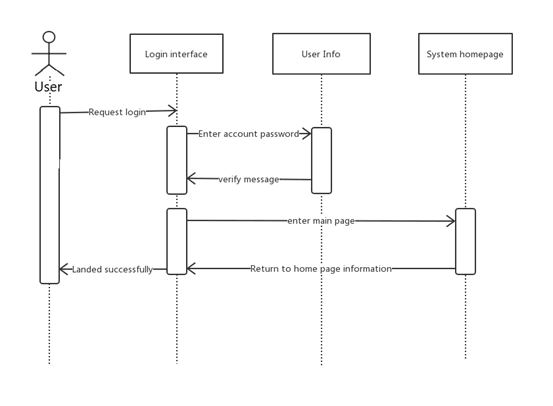
This is the "login page" of the library seating reservation system. After the student swipes the card into the library, he will go to the library seat reservation system machine to make a seat reservation. Students will see the "login page". Students need to enter the student's student number in the Username field, and the student enters the password in the password field (the password is usually the last six digits of the student's ID number). After the input is completed， click "ok" to proceed to the next step.



This is the "Logon Failure" interface. One of the situations where students enter their username and password in the "Login Page". The reason for this is that the student's username and password do not match, that is, the student's username or password input error. At this time, students will click on the "Back" button at the bottom right of the page to return to the previous "login page" and re-enter the student's username and password.This is the interface for "selecting a seat." After the student successfully logs in to the “Login Page”, he will automatically jump to the “Select Seat” page. On this page, students can click on the seating chart below to select the seat number they want. If the student does not want to choose a seat on the first floor, then click the inverted

**12.Design of Tests**

Test case：



|  |  |  |  |
| --- | --- | --- | --- |
|  | Features | Input | Output |
| System login test | Determine whether the user identity is legal | Username, password, permission | Display corresponding subsystem |

Test: Name: System Operation Login Test  
Purpose: Test the system operator interface.  
Content: account password input, plausibility check, legality check, system operation interface display control

Range:The test cases in this test plan specification can basically cover all the cases and basically reflect whether the software has errors.

Test design specification:

|  |  |  |
| --- | --- | --- |
| Input and test cases | Expected output | Reason for selection |
| Enter username, don't enter password | Show no password entered | password can not be blank |
| Do not enter a username, enter a password | Show no username entered | Username can not be empty |
| Enter a username or password that does not match | Display password is incorrect | Username and password must match to log in |
| Enter unregistered username | Show that the user does not exist | Login must be a registered user |
| Enter matching username and question | Show login success | Legal login request |

**13.History of Work**

A.Identify team project goals

B.Assign tasks among team members to identify tasks that team members need to accomplish

C.Team members write their own part of the code based on their assigned tasks

3.14~3.28

Willam Write a seat selection code

Alan User registration code

Andy Seat information add code

Abbott User submits repair information code

Mike User seat release code

Scott Check registered user information code

3.29~4.11

Willam Seat confirmation code

Alan User login code

Andy Seat information removal code

Abbott The administrator releases the seat code for the repair

Mike Web page design

Scott Delete user information code

4.12~4.25

Willam Seat query code

Alan Administrator login section

Andy Seat information modification code

Abbott

Mike Web page design

Scott Search user information code

D.Team members add databases based on their task needs

4.04~4.19

Create a database and add database data

E.Team members complete code connection and database connection together

5.02~5.09 Team members integrate their own code

5.10~5.15 Integrate all code and connect to the database

5.16 Display results

F. Fix bugs and improve each team member code

5.17~5.22 Solve the problems found

**14.Summary of changes**

On April 18, the Use Case of the Administrator section was added.

On April 21, the design illustration of the report1 page was changed.

May 10, updated the database environment

May 17, changed the bug that users can view the admin interface

**15.References**

<https://github.com/orgs/library-seating-reservation-system/projects/1>