College of Software, Jilin University

Course Report of
Software Architecture, 2022

Title： Architecture design for a Blockchain technique based personal information management system

Date： 2022/9/25

Team No.： Team 23

Members:

序号	学号	姓名	贡献 百分比	手写/电子签名
1	13200430	霍鑫壮	27%	霍鑫壮
2	13200432	周鑫	21%	周鑫
3	14201016	文翔宇	18%	文翔宇
4	23200816	王俊博	17%	王俊博
5	55200401	刘子萱	17%	刘子萱

Content

1. Introduction

1.1 Antecedent knowledge

1.1.1 Blockchain

Blockchain is a chain composed of blocks after block. Each block contains certain information, which is connected to the chain in its own chronological order. Compared with traditional networks, blockchain has two core characteristics: first, data is difficult to tamper with, and second, decentralization. Based on these two characteristics, the information recorded by blockchain is more authentic and reliable, which helps to solve the problem of people's mistrust.

1.1.2 Personal Information Management System

The personal information management system provides a framework for organizations to maintain and improve compliance with data protection laws and best practices. It has further deepened the information security management system in the protection of personal information in order to strike a reasonable balance between the use and protection of personal data and reduce the risk of organizational operation and compliance.

1.2 Purpose

This document outlines the overall architecture of the system, and explains different aspects of the system from different perspectives by using various graphics, so that readers of this document can better understand the system. In addition, readers of this document should be divided into: 1. system analysis, 2. system design, 3. system deployment, 4. system testing, etc. More attention should be paid to: 1. Chapter IV, Chapter VI, 2. Chapter V, 3. Chapter VII, Chapter VIII, and 4. Chapter VI.

2. Architectural Representation

This document uses UML to describe, visualize, construct, and document all artifacts involved in this document. This document uses the standard 4+1 view format to describe different aspects of the system: use case view, logical view, process view, deployment attempt, and physical view. Each part uses multiple graphics to describe its detailed functions, and uses the form of model+text description to describe the purpose, potential requirements, functional requirements, deployment design and other aspects of the system.

3.Architectural Goals and Constraints

3.1 Goals

3.1.1 Practicability

The system should not only meet all existing functional requirements, but also be able to adapt to future potential functional requirements, or it can easily update the system to meet new requirements.

3.1.2 Reliability

The system can not only keep stable and error free for a long time, but also support the sudden situation of a large number of concurrent accesses in a short time, and can automatically repair as soon as possible in case of an exception to prevent a larger loss caused by time or data errors.

3.1.3 Security

The personal information data stored in the system shall be encrypted with a special method to prevent security losses caused by data leakage due to hackers, system errors, etc. The system users should be strictly granted different levels of authority. Only the level personnel who meet certain conditions can operate on the data to some extent to ensure the security of the data.

3.1.4 Compatibility and Extensibility

The system shall ensure that the access to the system on different platforms will get the same results, and have good scalability for future potential functional requirements.

3.1.5 Specialty

The design of this system meets the requirements of relevant commercial software standards issued by the state.

3.1.6 Maintainability

The system shall provide simple and powerful function maintenance module, so that the maintenance team of the company adopting the system can conveniently carry out data backup and daily management.

3.2 Constraints

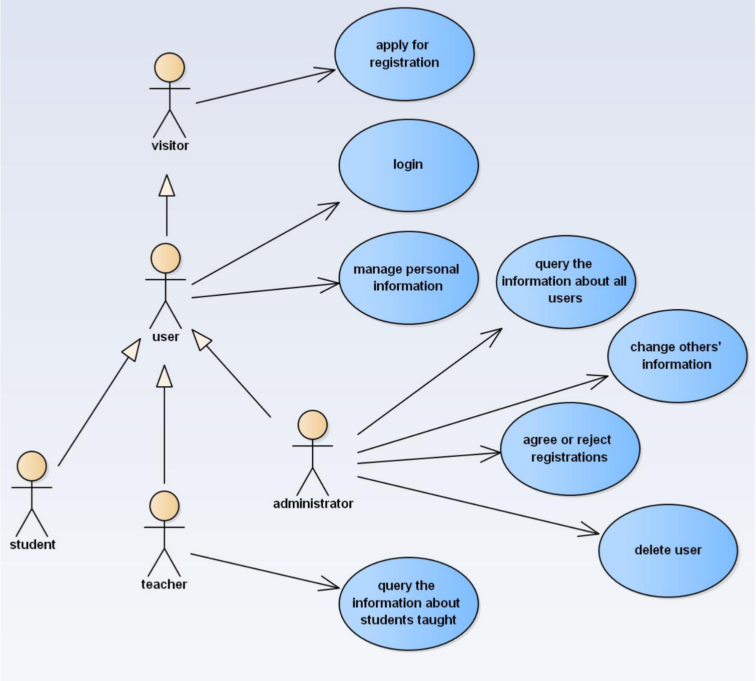
1.The system does not support the access of unknown users. All users must register in advance. Users who have not registered and logged in cannot use the system.

2.The system is suitable for mobile devices, laptops, personal computers and other ways, and can be accessed in Windows, Linux, IOS and other environments.

3.The system uses mysql database for database and navicat software for database visualization tools.

4.During functional design, all functional, efficiency and cost requirements specified in use case view and logical view must be considered.

4. Use-Case View: Architecturally-Significant Use Case


apply for registration
visitor
login
manage personal information
user
query the information about all users
change others' information
agree or reject registrations
administrator
student
delete user
teacher
query the information about students taught

4.1 Actors of the personal information management system

visitor:When using the system for the first time, you are a visitor and can perform the application registration use case to become another participant.

user:After successful registration, a user can log in to the system and manage personal information.

student:A more common type of user, with only a few basic functions.

teacher:A special class of users can query information about the students they teach.

administrator:The most special kind of users, the global management and maintenance of the system.

4.2 Use Case for visitor

4.2.1 Apply for registration

Use Case：apply for registration

Participant：visitor

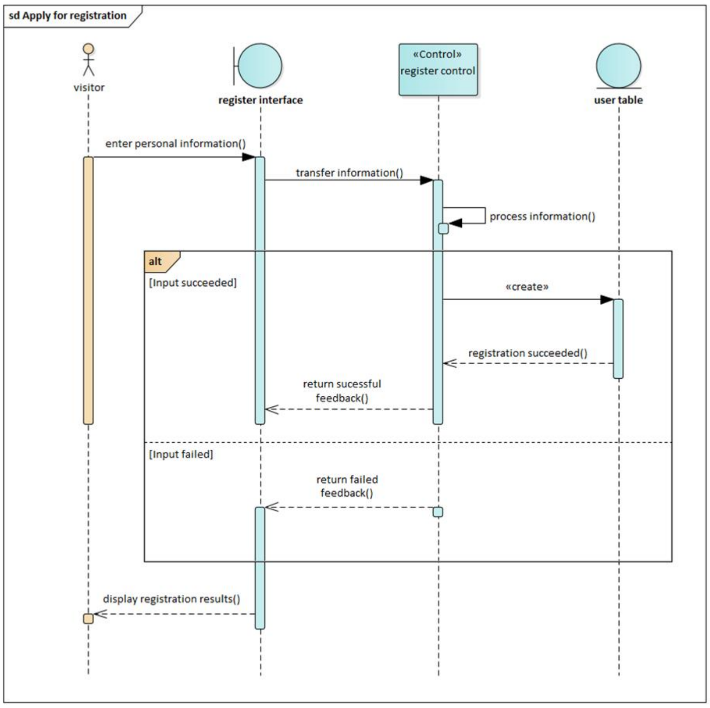
Brief description：In this use case, the user applies for registration and enters personal information into the system.

Main flow:

1. The user enters the registration interface and enters his/her personal information.
2. The system verifies the information entered by the user and creates a user table for it.
3. The registration interface displays successful registration.

alternative flow：

1.1The user entered illegally, and the system displays registration failure.



4.3 Use Case for user

4.3.1 Login

Use Case: Login

Participant: user

Brief description: This use case describes how users log in before using the system

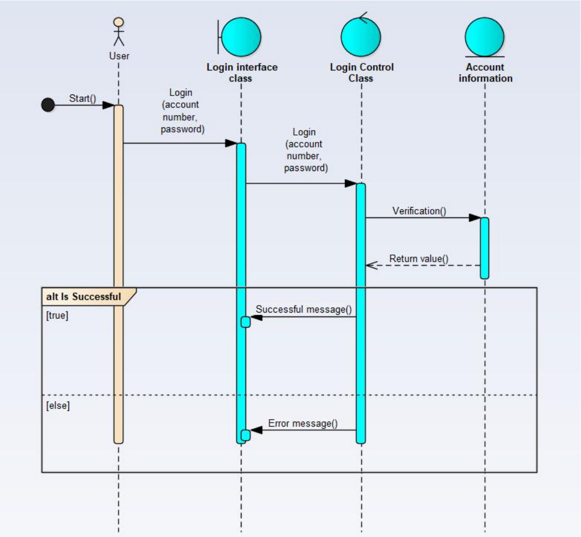
Main processes:

1. The user enters the system and enters the account and password

2. The user logs in successfully

Alternative process:

1.1. The user enters an incorrect account or password, and returns to the first step



4.3.2 Manage personal information

Use Case：Manage personal information

Participants：user/student/teacher/adminstrator

Brief description：This use case describe how users manage their personal information.

Main flow：

1.Users choose to query, modify or add personal information.

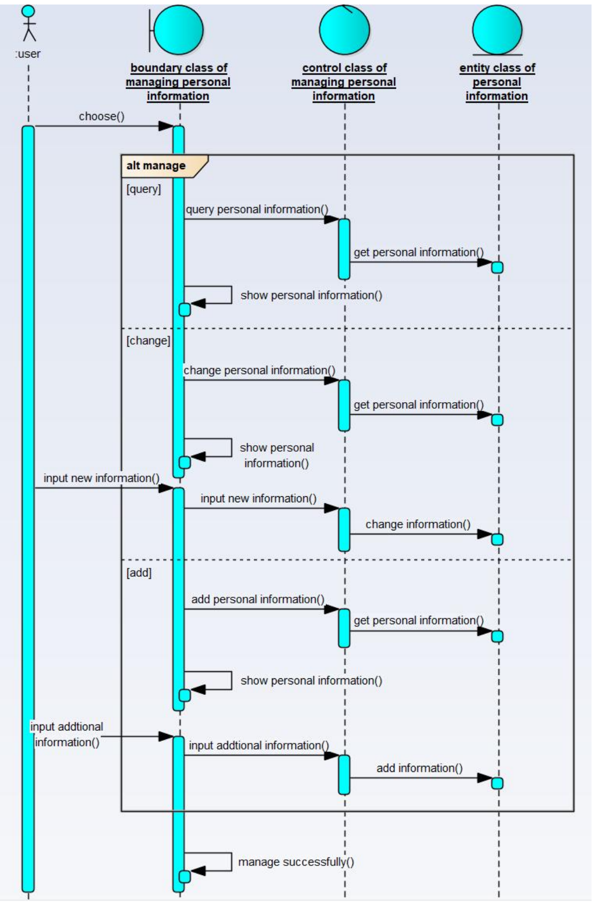
2.After you click the search option, the screen displays your personal information.

3.The message "Successful completion" is displayed.

Alternate flow：

2a. Click the Modify option, and the Modify interface pops up. Users can modify the existing information.

2b. Click the Add option to pop up the Add interface. Users can add existing information.



:user
boundary class of managing personal information
control class of managing personal information
entity class of personal information
choose()
alt manage
[query]
query personal information()
get personal information()
show personal information()
[change]
change personal information()
get personal information()
show personal information()
input new information()
input new information()
change information()
[add]
add personal information()
get personal information()
show personal information()
input addtional information()
input addtional information()
add information()
manage successfully()

4.4 Use Case for teacher

4.4.1 Query the information about students taught

Use Case: Query the information about students

Participant: teacher

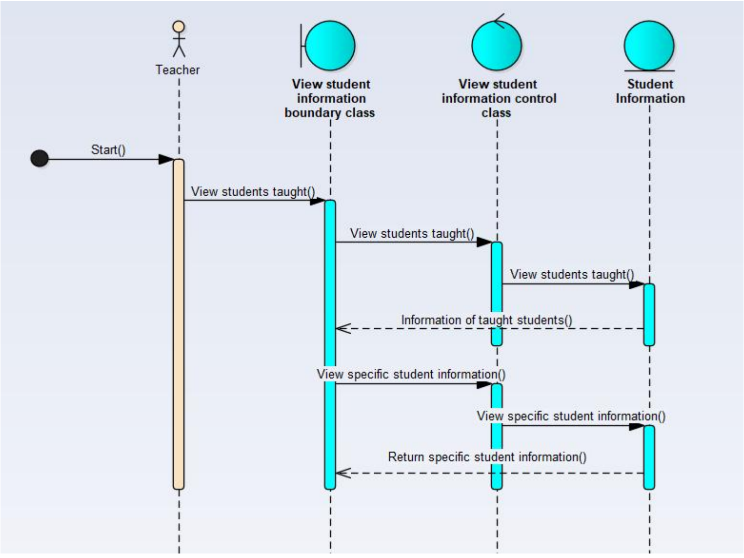
Brief description: This use case describes how a teacher can view the information of his students when using the system

Main processes:

1. The teacher enters the system and clicks to view the options
2. The teacher selects the students to view
3. View information

Alternative process:

nothing



Teacher
View student information boundary class
View student information control class
Student Information
Start()
View students taught()
View students taught()
View students taught()
Information of taught students()
View specific student information()
View specific student information()
Return specific student information()

4.5 Use Case for administrator

4.5.1 Query all users

Use Case: query all users

Participant: administrator

Brief description: The manager can query by entering the job number of the query object.

Main flow:

1.The manager enters the query interface and enters the job/student number of the query object.

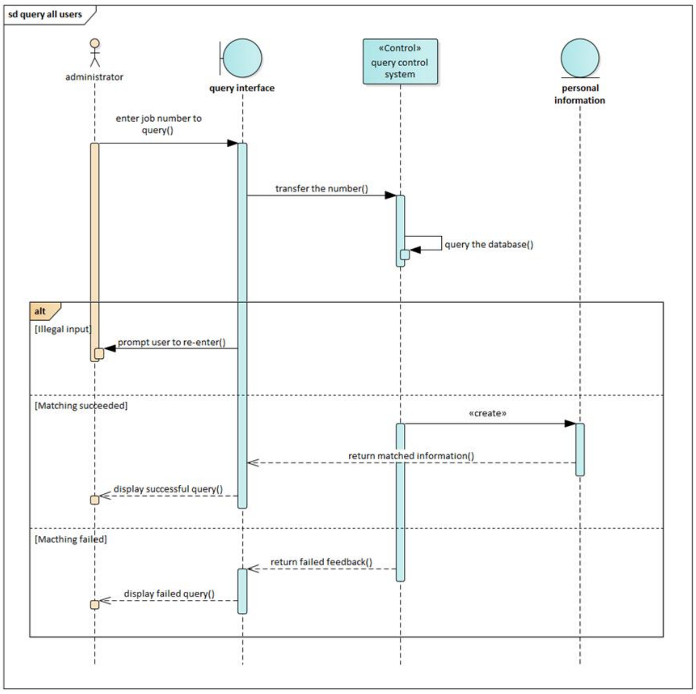
2.System search database.

3.The successfully matched information will be returned to the query interface, and the query success will be displayed.

Alternative flow:

1.1.The administrator enters illegally, and the system prompts to re-enter.

1.2.The system searches the database but has no matching information, and returns the information failed to query to the interface.



4.5.2 Delete registrations

Use Case：Delete registrations

Participants：administrator

Brief description：In this use case, the administrator can delete the user.

Main flow：

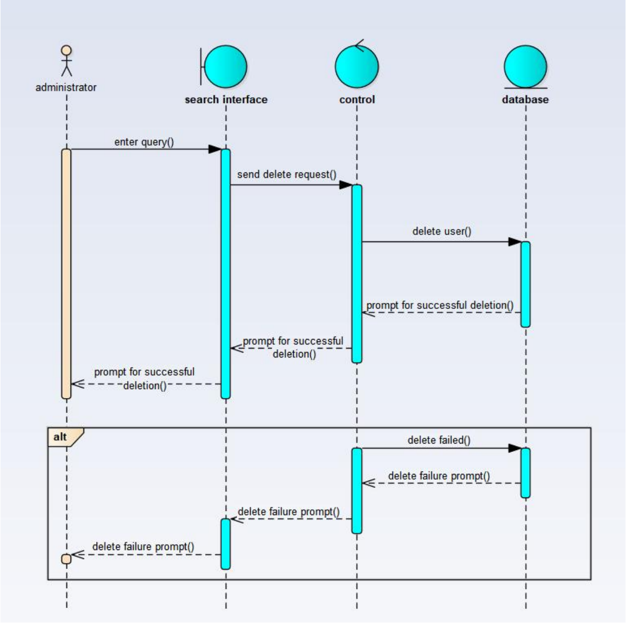
1. The administrator enters the search interface

2. Administrator input query

3. Delete User

Alternate flow：

a.When the query statement does not conform to the specification, the system will feed back that the content entered by the user is incorrect.



4.5.3 Agree or reject registrations

Use Case： Agree or reject registrations

Participants： adminstrator

Brief description： This use case describes the administrator's handling of a registration request.

Main flow：

1.The administrator chooses to process the registration application.

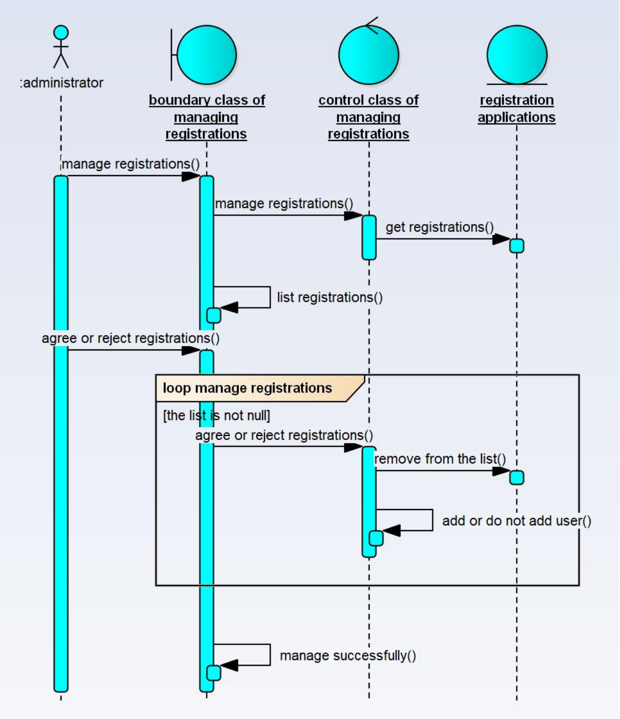
2.The system displays a list of all current applications.

3.The administrator agrees or rejects each item.

4.The processing is complete when the list is all processed.

Alternate flow：

2a. If the system has no application, end the use case.


:administrator
boundary class of managing registrations
control class of managing registrations
registration applications
manage registrations()
manage registrations()
get registrations()
list registrations()
agree or reject registrations()
loop manage registrations
[the list is not null]
agree or reject registrations()
remove from the list()
add or do not add user()
manage successfully()

4.5.4 Modify other person's information

Use Case： Modify other person's information

Participants： adminstrator

Brief description： In this use case, the administrator can modify other person's information

Main flow：

1. The administrator enters the search interface

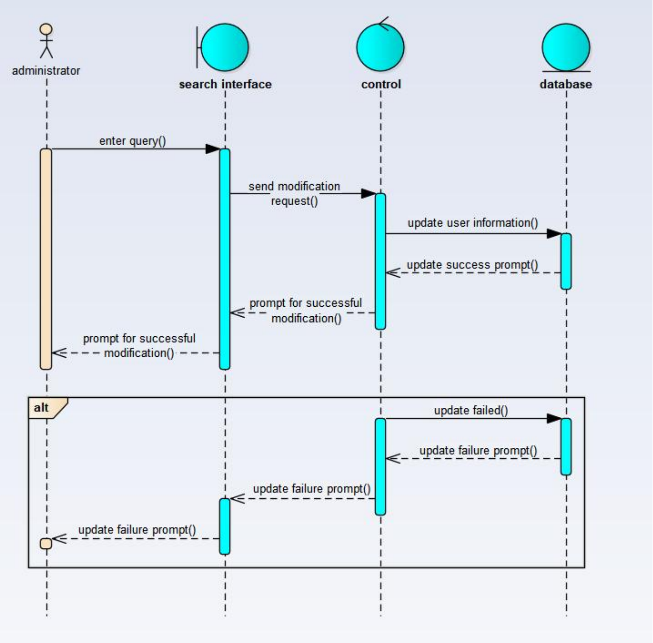
2. Administrator input query

3. Modify information

Alternate flow：

a.When the query statement does not conform to the specification, the system will feed back that the content entered by the user is incorrect.

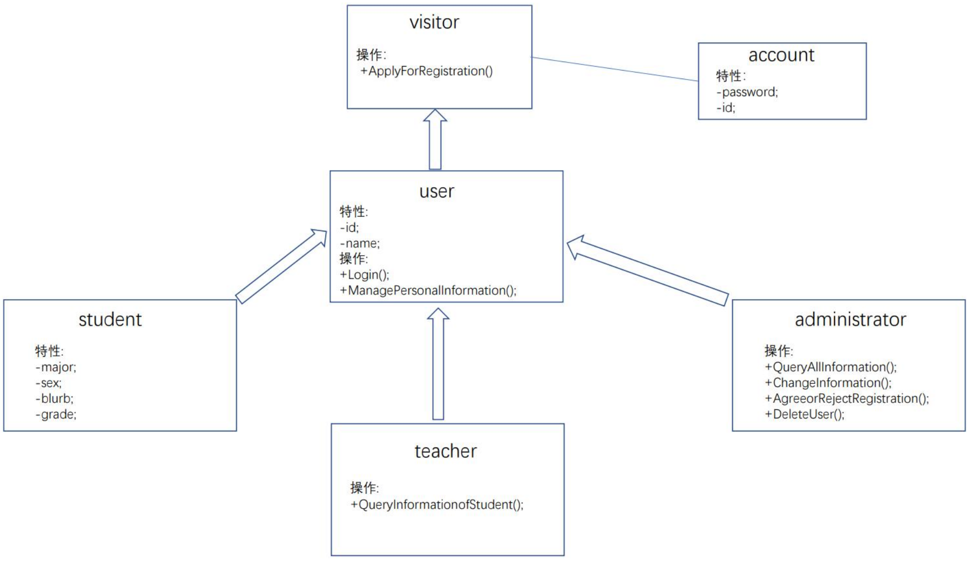
Sequence Diagram:



5. Logical View: Architecture Overview

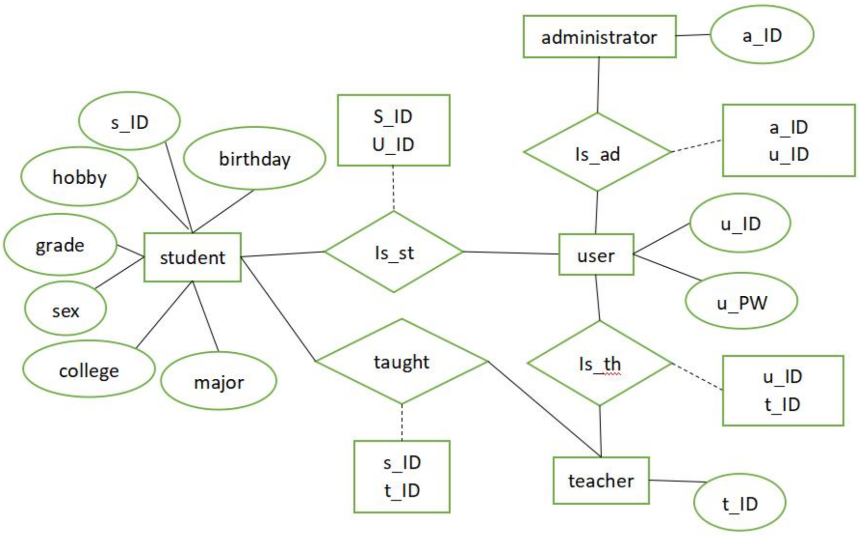
5.1 Class Diagram

The main classes related are as follow, The visitor class and the account class are related relationships, the user class and the visitor class are generalized relations, and the student class, teacher class and administrator class and user class are generalized relationships.



5.2 Database Construction

5.2.1 Entity Relationship Diagram



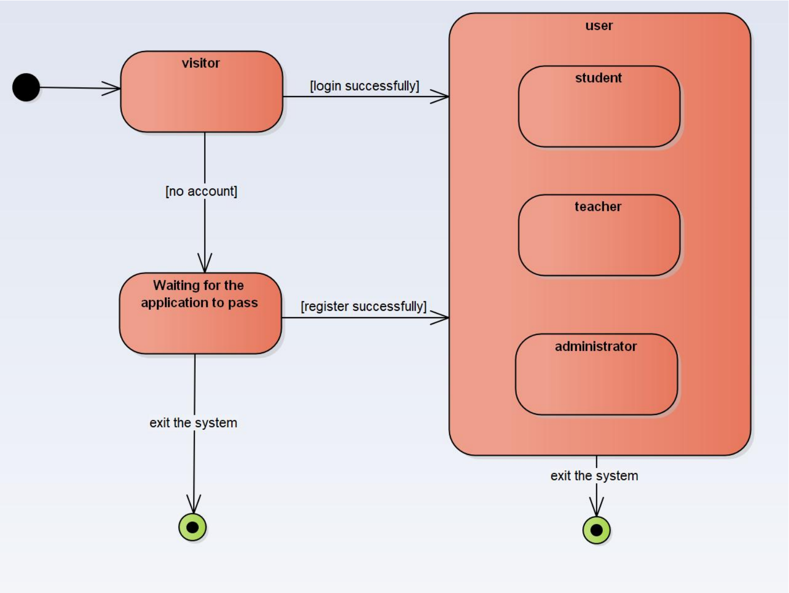
5.2.2 Relational schema

student(birthday,s_ID,hobby,grade,sex,college,major)
user(u_ID,u_PW)
teacher(t_ID)
administrator(a_ID)
Is_st(s_ID,u_ID)
taught(s_ID,t_ID)
Is_ad(a_ID,u_ID)
Is_th(u_ID,t_ID)

6. PROCESS VIEW

6.1 State-chart Diagram

When the system is initially opened, it is in the state of visitors. If there is no account, it needs to register and apply. If the registration fails, it will exit the system; otherwise, it will enter the system successfully. If you already have an account, log in to the system. The system is in user state, including student, teacher, and administrator.

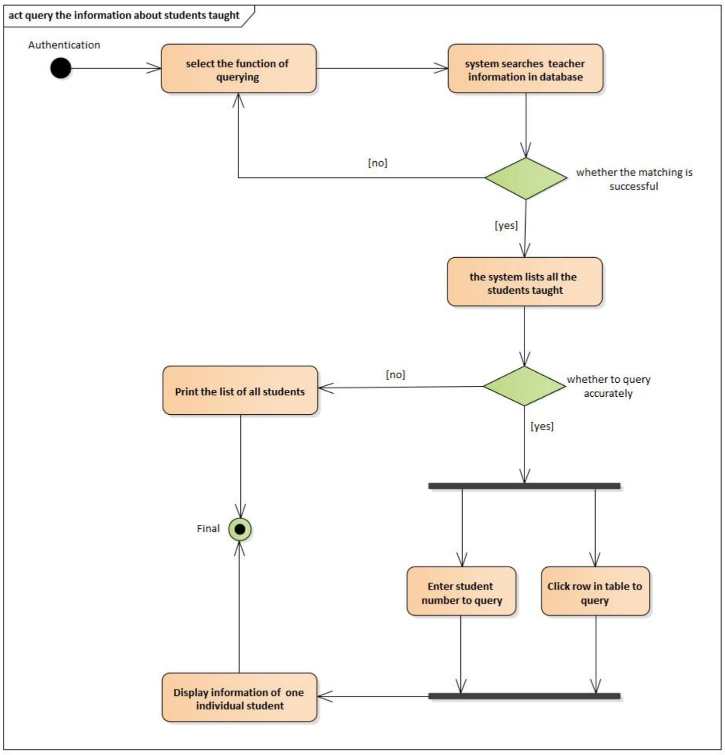


6.2 Activity Diagram

6.2.1 Query the information about students taught

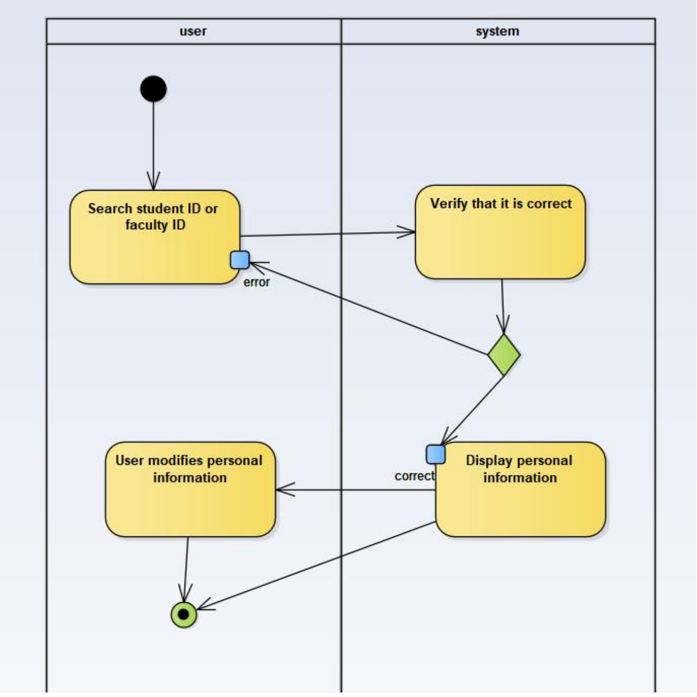
The function of querying student information is only authorized to teachers. Before using this function, you need to log in. At the same time, the system can only operate normally after successfully matching the teacher information in the database.

It is worth noting that teachers can query the information of a single student in two ways.


act query the information about students taught
Authentication
select the function of querying
system searches teacher information in database
[no]
whether the matching is successful
[yes]
the system lists all the students taught
Print the list of all students
[no]
whether to query accurately
[yes]
Final
Enter student number to query
Click row in table to query
Display information of one individual student

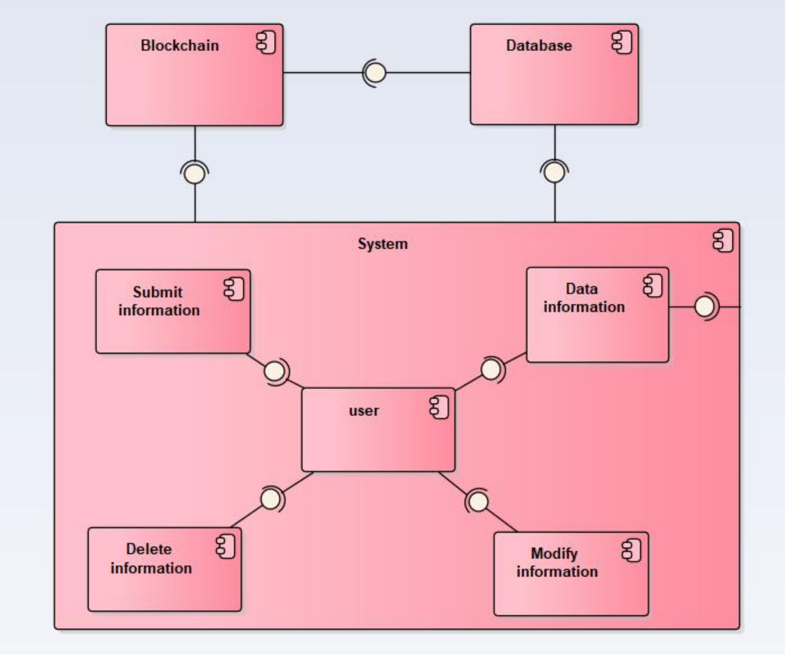
6.2.2 Manage personal information

The user first enters the management system, then searches his/her student number or faculty number and enters the password. The system verifies whether it is correct. If it is correct, the user can enter and modify his/her information;If there is an error, the system will prompt you to re-enter. After entering, users can choose to modify their own information and then exit; Or check and exit directly.


user
system
Search student ID or faculty ID
Verify that it is correct
error
User modifies personal information
correct
Display personal information

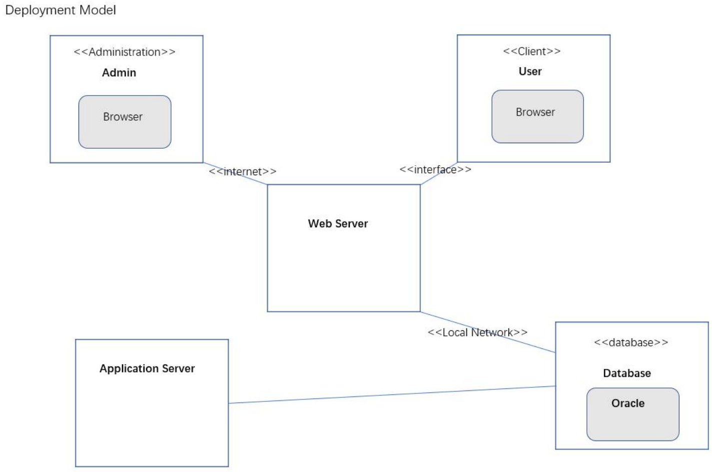
7. DEPLOYMENT VIEW

The component view shows the physical composition and the dependency relationship among those components of the system.



8. PHYSICAL VIEW

Deployment diagram of the architecture describes the various physical nodes for the most typical platform configurations.



9. REFERENCES

[1] Zheng Lulu, Zhao Jianchuan, Xu Yi, Hu Xunxun, Jiang Longsheng, Wang Hang. Personal Information Management System Based on Blockchain Technology [J]. Internet of Things Technology, 2018,8 (09): 100-101+103. DOI: 10.16667/j.issn.2095-1302.2018.09.025

[2] Shao Qifeng, Jin Chuqing, Zhang Zhao, Qian Weining, Zhou Aoying. Blockchain technology: architecture and progress [J]. Journal of Computer Science, 2018, 41 (05): 969-988