Report

Contents

Assumption and the user and system requirements

Assumption

Assume we are the user of this system we hope the system can be easy to use it. Also, the system UI looks very clear. And we will input all the data correctly.

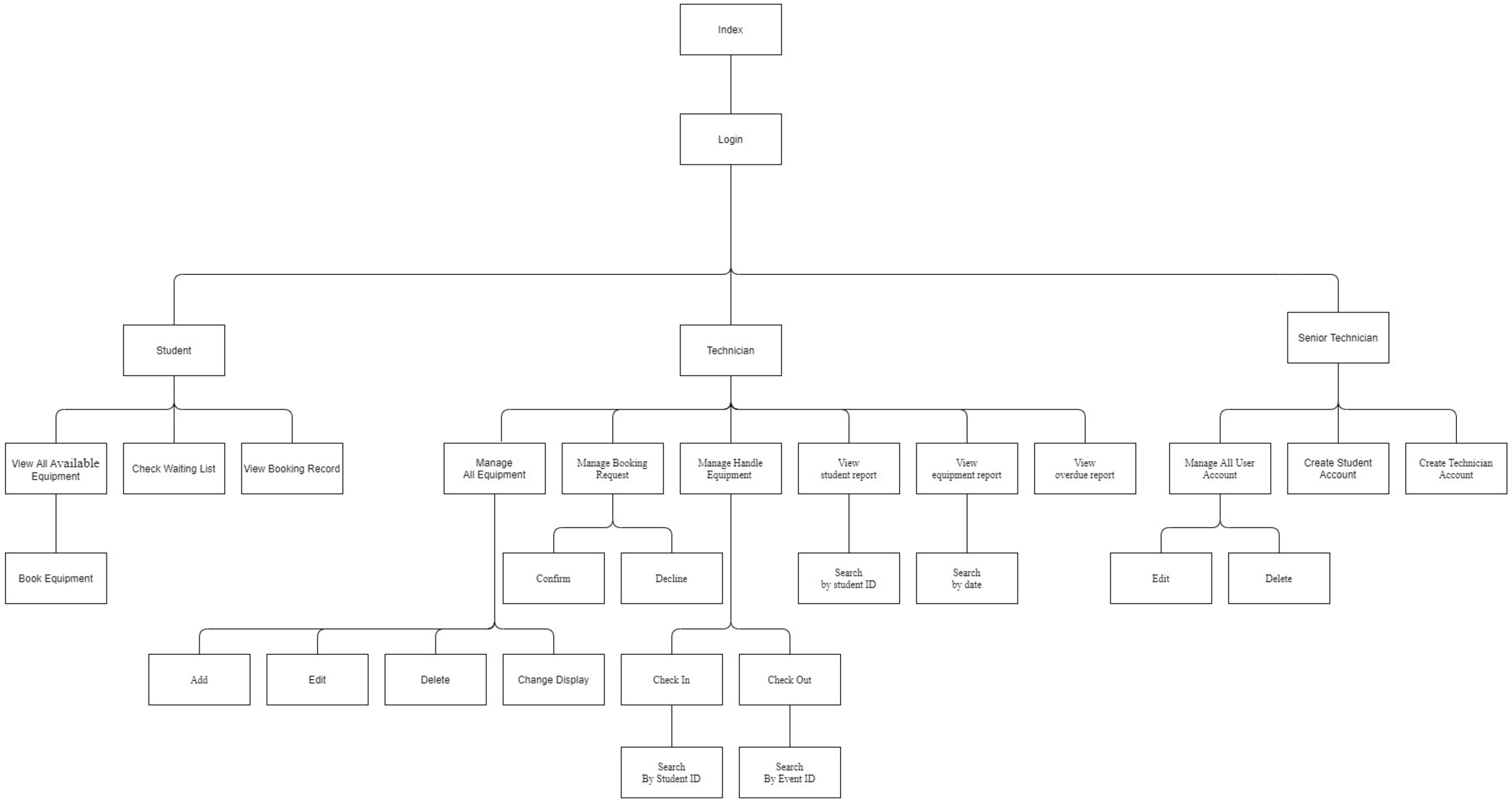
Users

1. Student
2. Technicians
3. Senior Technicians

System requirements

Users	System requirements
Student	● Login account ● Logout account ● View all available Equipment ● Create a new booking record ● View personal record
Technicians	● Login account ● Logout account ● Manage all Equipment (display, edit, delete, add) ● Manage Student booking request (confirm, decline) ● Manage handle equipment (Search by event ID / Student ID, check in / out) ● View student report of borrowing record (Search by student ID) ● View equipment report of utilization rate record (Search by date) ● View overdue report of no delivered equipment record
Senior Technicians	● Login account ● Logout account ● Manage all User account (delete, edit) ● Create student account ● Create Technician account

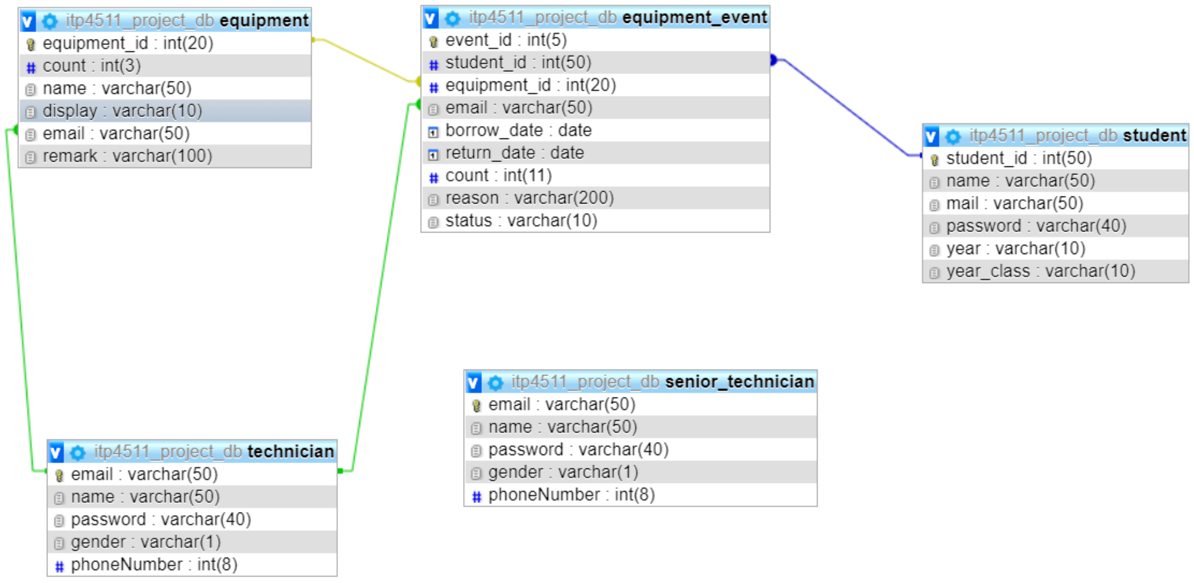
Site map


Index
Login
Student
Technician
Senior Technician
View All Available Equipment
Check Waiting List
View Booking Record
Book Equipment
Manage All Equipment
Manage Booking Request
Manage Handle Equipment
View student report
View equipment report
View overdue report
Confirm
Decline
Search by student ID
Search by date
Add
Edit
Delete
Change Display
Check In
Check Out
Search By Student ID
Search By Event ID
Manage All User Account
Create Student Account
Create Technician Account
Edit
Delete

System structure on how MVC Model is applied

First, our system will get the user's requirements. Next, the server will define the user's requirements and decide which page should be redirected for the users. Also, the page data will generate in the server first. Sometimes it will execute some SQL to change the database data first. Then, when the server finished handling those requests, the system will decide which page is suitable for the user's identity and redirect to it. So, this system is applied to the MVC model by handle the user's request in the server, which isn't handled from the user's side.

Database structure



itp4511_project_db **equipment**
equipment_id : int(20)
count : int(3)
name : varchar(50)
display : varchar(10)
email : varchar(50)
remark : varchar(100)

itp4511_project_db **equipment_event**
event_id : int(5)
student_id : int(50)
equipment_id : int(20)
email : varchar(50)
borrow_date : date
return_date : date
count : int(11)
reason : varchar(200)
status : varchar(10)

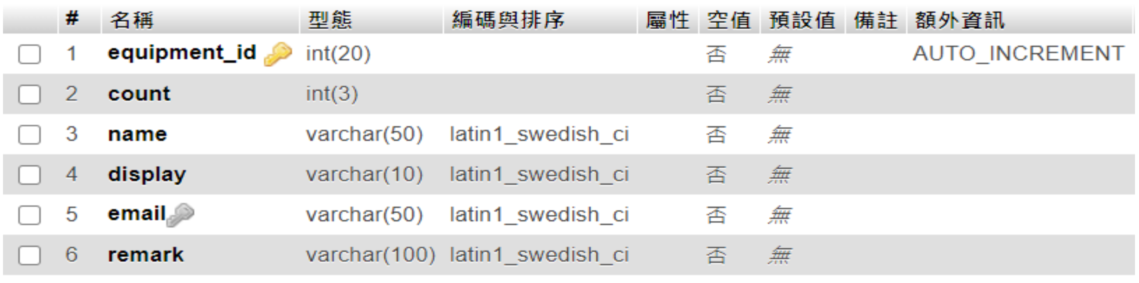
itp4511_project_db **student**
student_id : int(50)
name : varchar(50)
mail : varchar(50)
password : varchar(40)
year : varchar(10)
year_class : varchar(10)

itp4511_project_db **senior_technician**
email : varchar(50)
name : varchar(50)
password : varchar(40)
gender : varchar(1)
phoneNumber : int(8)

itp4511_project_db **technician**
email : varchar(50)
name : varchar(50)
password : varchar(40)
gender : varchar(1)
phoneNumber : int(8)

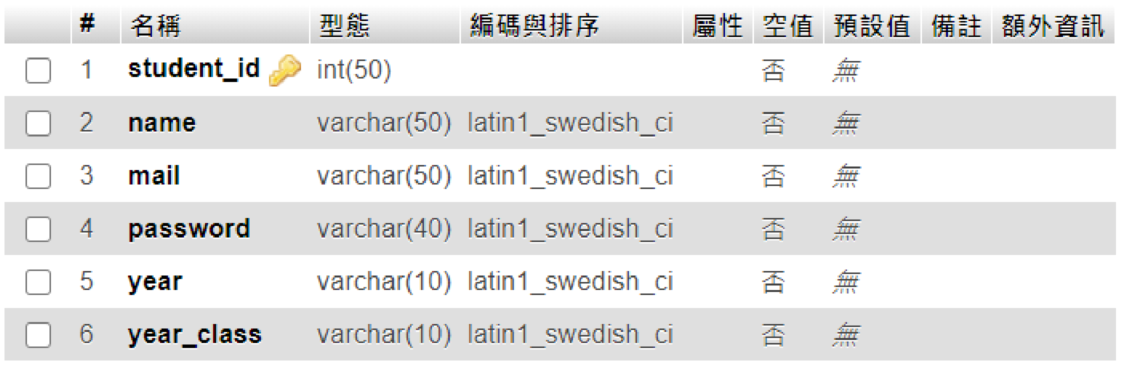
Table Data Type

Equipment Table



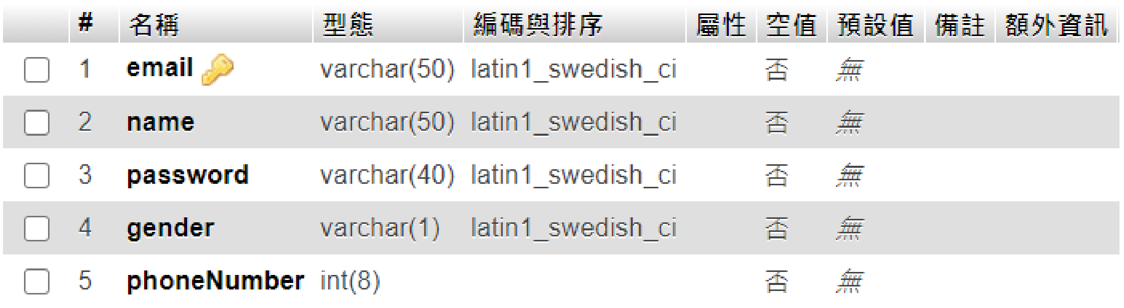
	#	名稱	型態	編碼與排序	屬性	空值	預設值	備註	額外資訊
☐	1	**equipment_id**	int(20)			否	無		AUTO_INCREMENT
☐	2	**count**	int(3)			否	無		
☐	3	**name**	varchar(50)	latin1_swedish_ci		否	無		
☐	4	**display**	varchar(10)	latin1_swedish_ci		否	無		
☐	5	**email**	varchar(50)	latin1_swedish_ci		否	無		
☐	6	**remark**	varchar(100)	latin1_swedish_ci		否	無		

Student Table



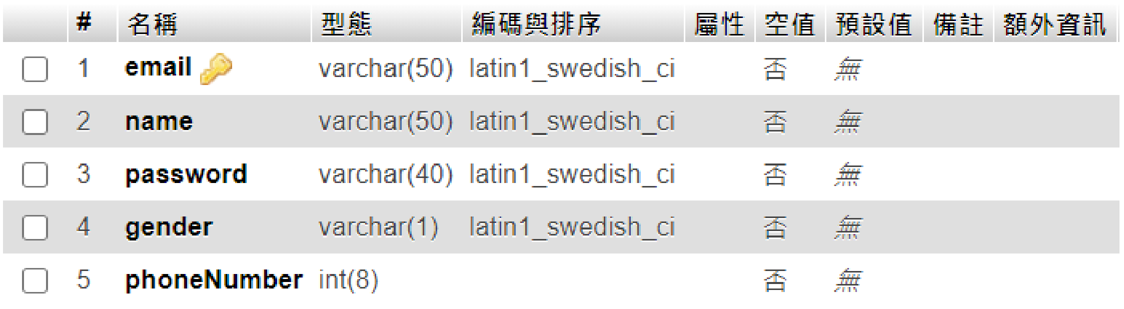
	#	名稱	型態	編碼與排序	屬性	空值	預設值	備註	額外資訊
☐	1	**student_id**	int(50)			否	無		
☐	2	**name**	varchar(50)	latin1_swedish_ci		否	無		
☐	3	**mail**	varchar(50)	latin1_swedish_ci		否	無		
☐	4	**password**	varchar(40)	latin1_swedish_ci		否	無		
☐	5	**year**	varchar(10)	latin1_swedish_ci		否	無		
☐	6	**year_class**	varchar(10)	latin1_swedish_ci		否	無		

Technician Table



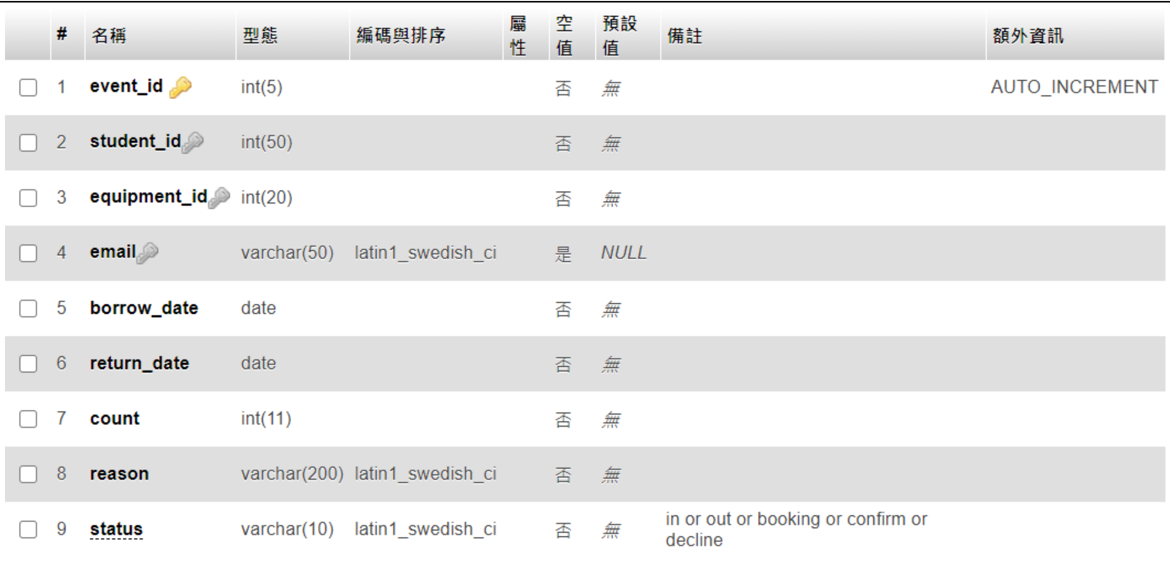
	#	名稱	型態	編碼與排序	屬性	空值	預設值	備註	額外資訊
☐	1	**email**	varchar(50)	latin1_swedish_ci		否	無		
☐	2	**name**	varchar(50)	latin1_swedish_ci		否	無		
☐	3	**password**	varchar(40)	latin1_swedish_ci		否	無		
☐	4	**gender**	varchar(1)	latin1_swedish_ci		否	無		
☐	5	**phoneNumber**	int(8)			否	無		

Senior_Technician Table



	#	名稱	型態	編碼與排序	屬性	空值	預設值	備註	額外資訊
☐	1	**email**	varchar(50)	latin1_swedish_ci		否	無		
☐	2	**name**	varchar(50)	latin1_swedish_ci		否	無		
☐	3	**password**	varchar(40)	latin1_swedish_ci		否	無		
☐	4	**gender**	varchar(1)	latin1_swedish_ci		否	無		
☐	5	**phoneNumber**	int(8)			否	無		

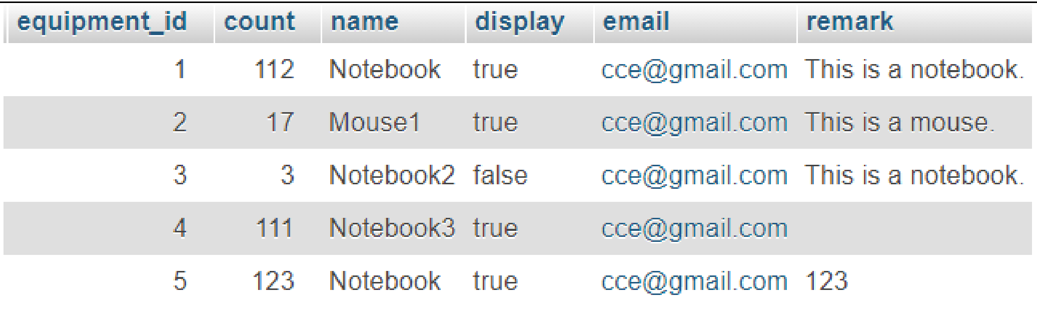
Equipment_event Table



	#	名稱	型態	編碼與排序	屬性	空值	預設值	備註	額外資訊
☐	1	event_id	int(5)			否	無		AUTO_INCREMENT
☐	2	student_id	int(50)			否	無		
☐	3	equipment_id	int(20)			否	無		
☐	4	email	varchar(50)	latin1_swedish_ci		是	NULL		
☐	5	borrow_date	date			否	無		
☐	6	return_date	date			否	無		
☐	7	count	int(11)			否	無		
☐	8	reason	varchar(200)	latin1_swedish_ci		否	無		
☐	9	status	varchar(10)	latin1_swedish_ci		否	無	in or out or booking or confirm or decline	

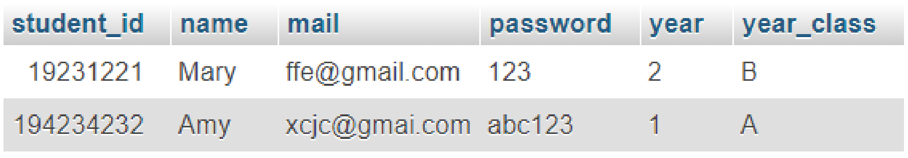
Simple Date

Equipment Table



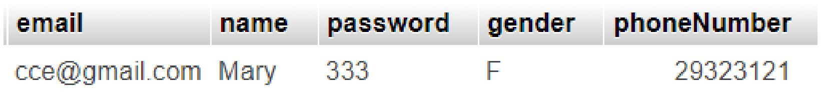
equipment_id	count	name	display	email	remark
1	112	Notebook	true	cce@gmail.com	This is a notebook.
2	17	Mouse1	true	cce@gmail.com	This is a mouse.
3	3	Notebook2	false	cce@gmail.com	This is a notebook.
4	111	Notebook3	true	cce@gmail.com	
5	123	Notebook	true	cce@gmail.com	123

Student Table



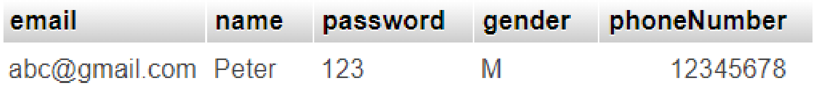
student_id	name	mail	password	year	year_class
19231221	Mary	ffe@gmail.com	123	2	B
194234232	Amy	xcjc@gmai.com	abc123	1	A

Technician Table

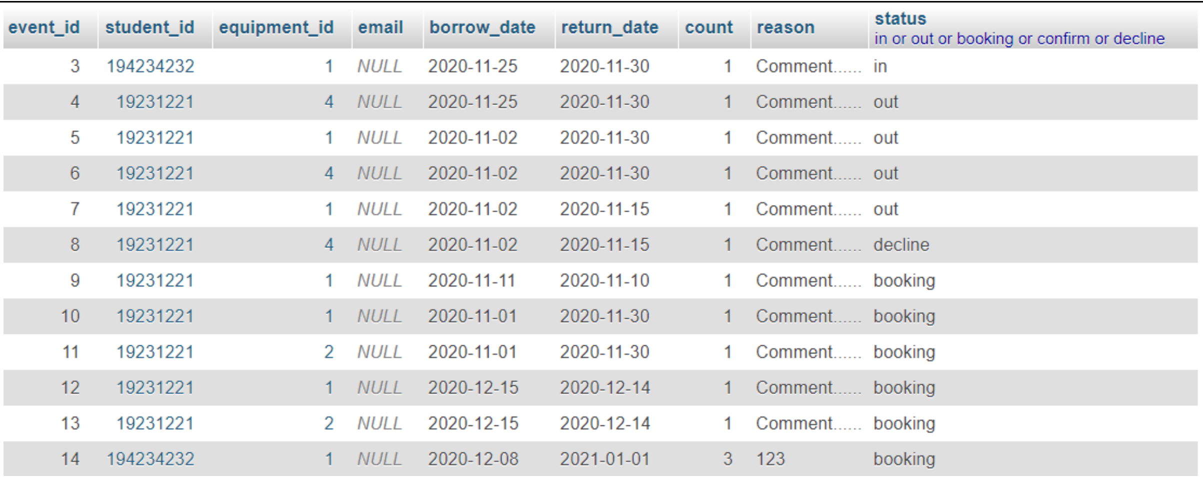


email	name	password	gender	phoneNumber
cce@gmail.com	Mary	333	F	29323121

Senior_Technician Table

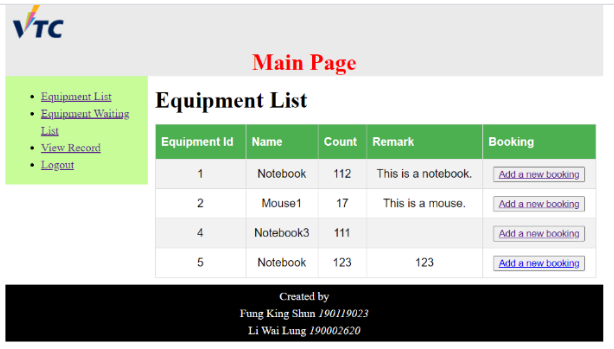


email	name	password	gender	phoneNumber
abc@gmail.com	Peter	123	M	12345678

Equipment_event Table	

event_id	student_id	equipment_id	email	borrow_date	return_date	count	reason	status in or out or booking or confirm or decline
3	194234232	1	NULL	2020-11-25	2020-11-30	1	Comment......	in
4	19231221	4	NULL	2020-11-25	2020-11-30	1	Comment......	out
5	19231221	1	NULL	2020-11-02	2020-11-30	1	Comment......	out
6	19231221	4	NULL	2020-11-02	2020-11-30	1	Comment......	out
7	19231221	1	NULL	2020-11-02	2020-11-15	1	Comment......	out
8	19231221	4	NULL	2020-11-02	2020-11-15	1	Comment......	decline
9	19231221	1	NULL	2020-11-11	2020-11-10	1	Comment......	booking
10	19231221	1	NULL	2020-11-01	2020-11-30	1	Comment......	booking
11	19231221	2	NULL	2020-11-01	2020-11-30	1	Comment......	booking
12	19231221	1	NULL	2020-12-15	2020-12-14	1	Comment......	booking
13	19231221	2	NULL	2020-12-15	2020-12-14	1	Comment......	booking
14	194234232	1	NULL	2020-12-08	2021-01-01	3	123	booking

Brief description (1 or 2 pages only) on the major characteristics and design of your application

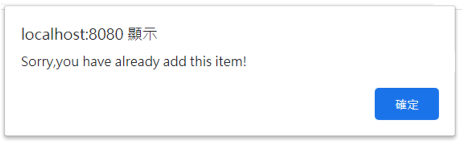

Main Page

- Equipment List
- Equipment Waiting List
- View Record
- Logout

Equipment List

Equipment Id	Name	Count	Remark	Booking
1	Notebook	112	This is a notebook.	Add a new booking
2	Mouse1	17	This is a mouse.	Add a new booking
4	Notebook3	111		Add a new booking
5	Notebook	123	123	Add a new booking

Created by
Fung King Shun *190119023*
Li Wai Lung *190002620*


localhost:8080 顯示
Sorry,you have already add this item!

a) Complete the user requirements

As you can see, on the left hand-side is all the requirements of the user. This is a student account, so we provided all the function which only student can do it.

b) Consistent design and easy to use

The page is very clear, because we use table to display the data and use css to design. So that, user can be easy to use and understand.

c) Smooth navigation with the application

For the navigation, we use navigation bar to change to page. Left hand-side is the bar labeled all the hyperlink. User can click the link to change the page.

d) Tidy Page Layout with logical and related graphics

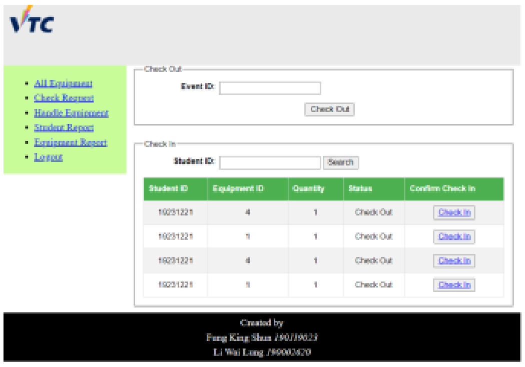
We use so many html codes to design like: h1, color, button, li, text size… Therefor, user easy to know this is a button, textbox, and hyperlink.

e) Error-free implementation

To handle the error like input error, system error, we display a message box to the user. So, user can understand why appeared error.

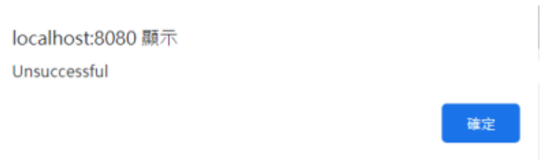
f) Creativity

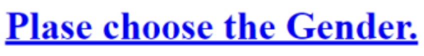
In order not to be complicated, we used the simplest method. But at the same time, all functions must be retained.



Student ID	Equipment ID	Quantity	Status	Confirm Check In
19231221	4	1	Check Out	Check In
19231221	1	1	Check Out	Check In
19231221	4	1	Check Out	Check In
19231221	1	1	Check Out	Check In

Created by
Fung King Shun *190119023*
Li Wai Lung *190003620*


localhost:8080 顯示
Unsuccessful


Plase choose the Gender.

a) Complete the user requirements

This is the technician account we provided all the function which the technician can do it which you can see on the left-hand side.

b) Consistent design and easy to use

This page combines form and table which used CSS to design. Users can use this function by their intuition easily.

c) Smooth navigation with the application

For the navigation, we put the navigation bar on the left-hand side of all pages. We will use that to change to the page. Users can click the link to call those functions.

d) Tidy Page Layout with logical and related graphics

We use CSS to design the form and table which can let the user think this page is very tidy.

e) Error-free implementation

If there is an error like an input error, system error, we display an alert box or a message to the user. For example, when the user inputs incorrect data, we will use the alert box to tell the user is it the action successful.

f) Creativity

We use the simplest CSS to design those pages which make those pages not complicated. On the other hand, we retained all functions.

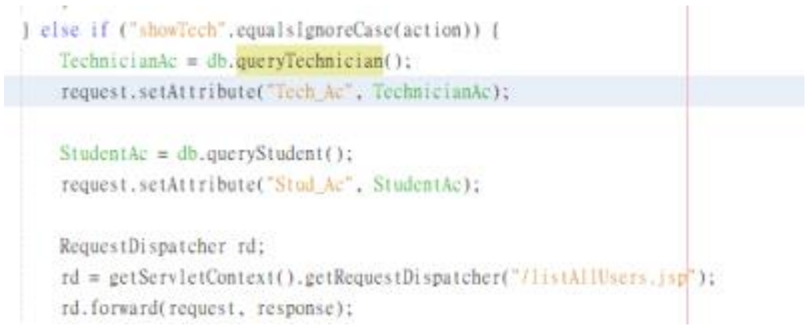
Conclusions

After completed the project, we know that the MVC architecture that separates business logic, presentation layer and data. M stands for Model V stands for View and C stands for controller. In Model, this is the data which consists of the business logic of the system. It consists of classes which have the connection to the database. And we use JDBC to connect the Database fetches the data and sends to the view layer. In view, it consists of HTML, JSP and CSS. To shows the data on UI of the application. In controller, it is the interface between View and Model. It receives the requests from the view layer and processes the requests and does the necessary validation for the request. So, using MVC model can be easy to maintain, extend and test.

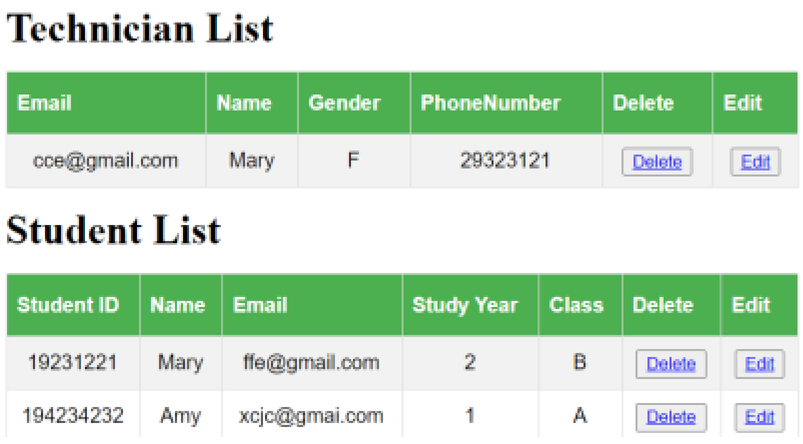
Skill checklist

A. Use JSP/servlets to dynamically generate HTML pages

We use servlets to call Db to catch the data and show the table



```
} else if ("showTech".equalsIgnoreCase(action)) {
    TechnicianAc = db.queryTechnician();
    request.setAttribute("Tech_Ac", TechnicianAc);

    StudentAc = db.queryStudent();
    request.setAttribute("Stud_Ac", StudentAc);

    RequestDispatcher rd;
    rd = getServletContext().getRequestDispatcher("/listAllUsers.jsp");
    rd.forward(request, response);
```



Technician List

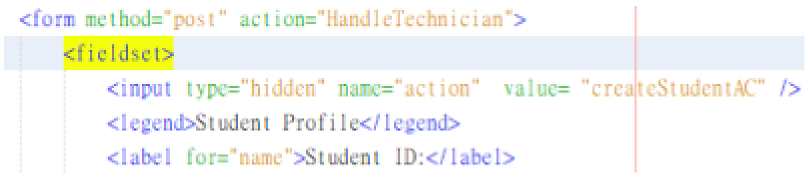
Email	Name	Gender	PhoneNumber	Delete	Edit
cce@gmail.com	Mary	F	29323121	Delete	Edit

Student List

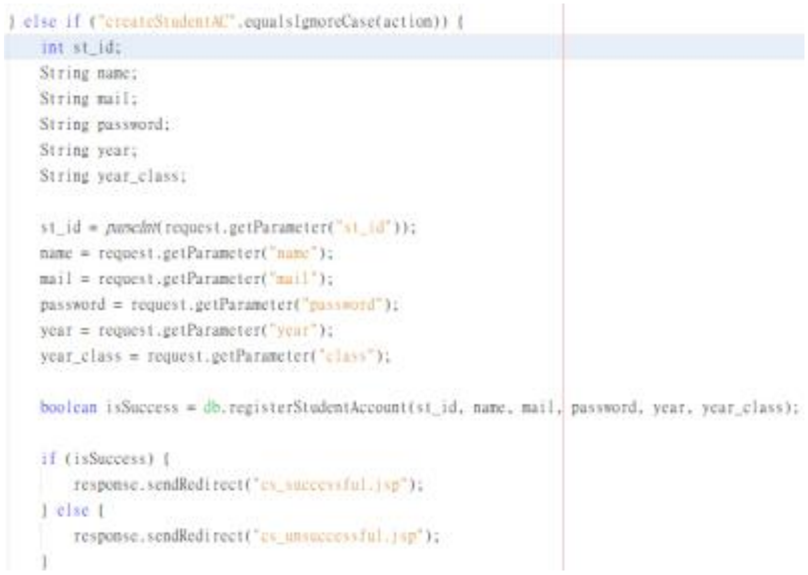
Student ID	Name	Email	Study Year	Class	Delete	Edit
19231221	Mary	ffe@gmail.com	2	B	Delete	Edit
194234232	Amy	xcjc@gmai.com	1	A	Delete	Edit

B. Use JSP/servlets to accept user inputs from browser

We use servlets to catch the data which the user input.



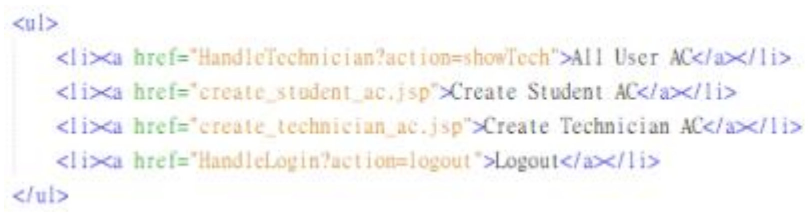
```
<form method="post" action="HandleTechnician">
    <fieldset>
        <input type="hidden" name="action"  value= "createStudentAC" />
        <legend>Student Profile</legend>
        <label for="name">Student ID:</label>
```



```
} else if ("createStudentAC".equalsIgnoreCase(action)) {
    int st_id;
    String name;
    String mail;
    String password;
    String year;
    String year_class;

    st_id = parseInt(request.getParameter("st_id"));
    name = request.getParameter("name");
    mail = request.getParameter("mail");
    password = request.getParameter("password");
    year = request.getParameter("year");
    year_class = request.getParameter("class");

    boolean isSuccess = db.registerStudentAccount(st_id, name, mail, password, year, year_class);

    if (isSuccess) {
        response.sendRedirect("cs_successful.jsp");
    } else {
        response.sendRedirect("cs_unsuccessful.jsp");
    }
```

C. Use JSP Action

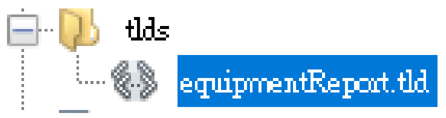
We use JSP Action to call the function

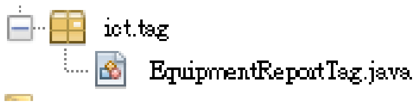


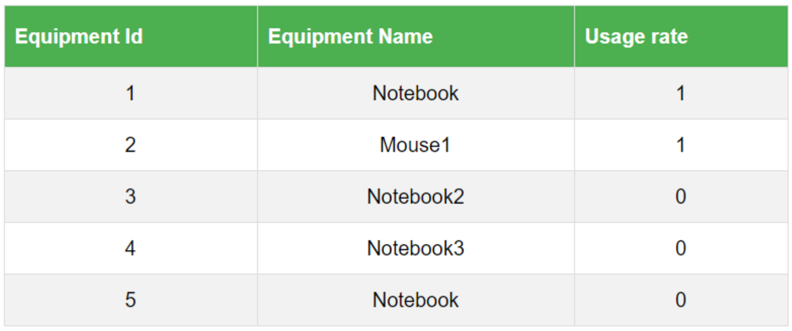
```
<ul>
    <li><a href="HandleTechnician?action=showTech">All User AC</a></li>
    <li><a href="create_student_ac.jsp">Create Student AC</a></li>
    <li><a href="create_technician_ac.jsp">Create Technician AC</a></li>
    <li><a href="HandleLogin?action=logout">Logout</a></li>
</ul>
```

D. Use Custom Tag (Taglib)

We use taglib to display the JSP



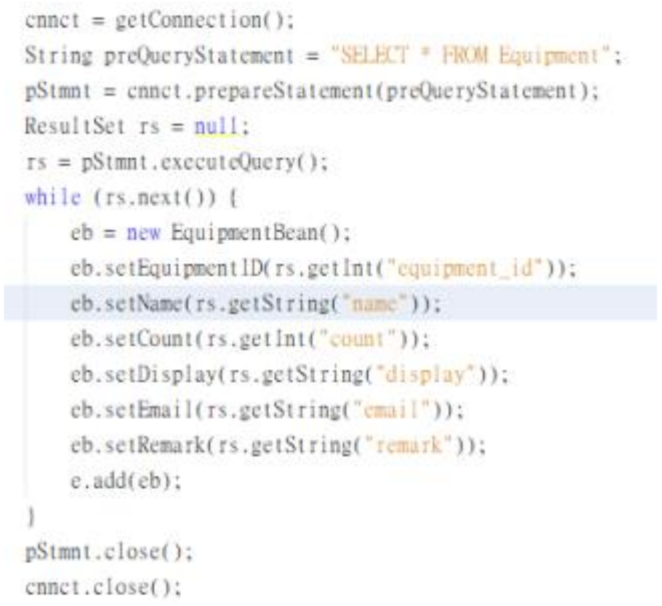




Equipment Id	Equipment Name	Usage rate
1	Notebook	1
2	Mouse1	1
3	Notebook2	0
4	Notebook3	0
5	Notebook	0

E. Use JavaBean

We use JavaBean to save the data which catch from the database



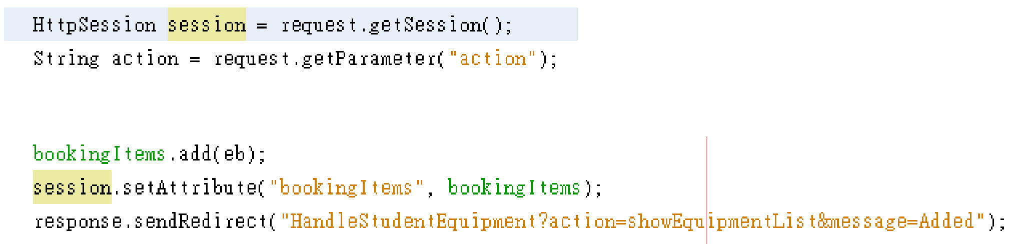
```
cnnct = getConnection();
String preQueryStatement = "SELECT * FROM Equipment";
pStmnt = cnnct.prepareStatement(preQueryStatement);
ResultSet rs = null;
rs = pStmnt.executeQuery();
while (rs.next()) {
    eb = new EquipmentBean();
    eb.setEquipmentID(rs.getInt("equipment_id"));
    eb.setName(rs.getString("name"));
    eb.setCount(rs.getInt("count"));
    eb.setDisplay(rs.getString("display"));
    eb.setEmail(rs.getString("email"));
    eb.setRemark(rs.getString("remark"));
    e.add(eb);
}
pStmnt.close();
cnnct.close();
```

F. Use JDBC for database connection

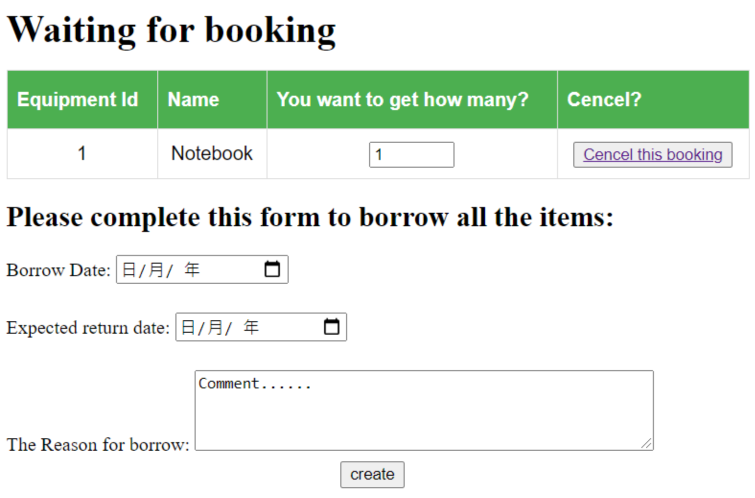
We use ict.servlets to call ict.db. And the ict.db will connect to the database and do those SQL.

G. Use session checking

The session is used in saving student borrowing request



```
HttpSession session = request.getSession();
String action = request.getParameter("action");


bookingItems.add(eb);
session.setAttribute("bookingItems", bookingItems);
response.sendRedirect("HandleStudentEquipment?action=showEquipmentList&message=Added");
```



Waiting for booking

Equipment Id	Name	You want to get how many?	Cencel?
1	Notebook	1	Cencel this booking

Please complete this form to borrow all the items:

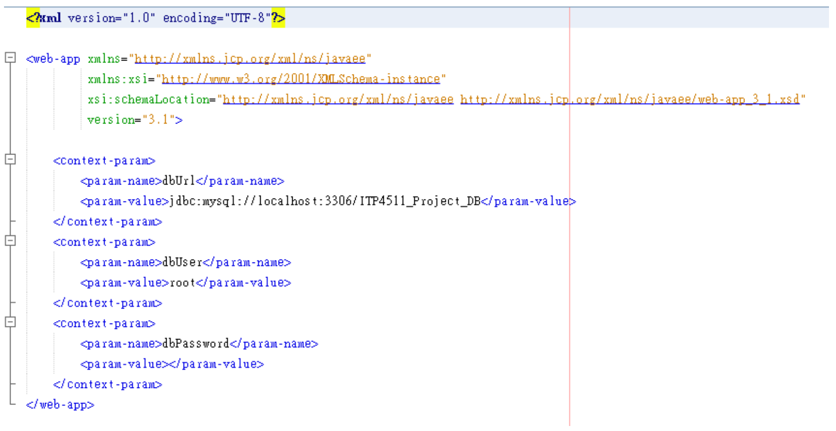
Borrow Date: 日/月/ 年

Expected return date: 日/月/ 年

The Reason for borrow: Comment......

H. Use login control

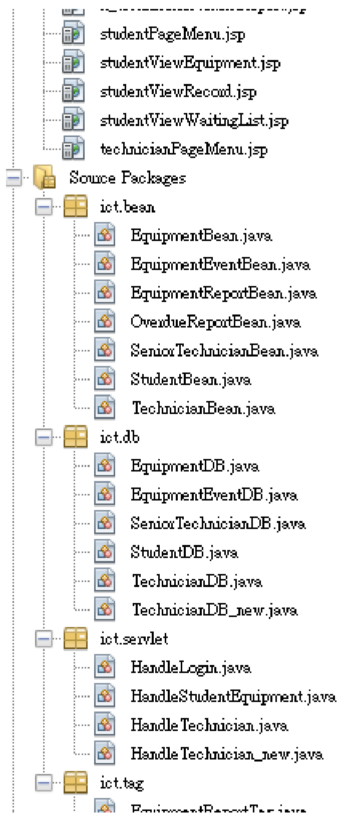
To login DB we created a web.xml to connected



```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

    <context-param>
        <param-name>dbUrl</param-name>
        <param-value>jdbc:mysql://localhost:3306/ITP4511_Project_DB</param-value>
    </context-param>
    <context-param>
        <param-name>dbUser</param-name>
        <param-value>root</param-value>
    </context-param>
    <context-param>
        <param-name>dbPassword</param-name>
        <param-value></param-value>
    </context-param>
</web-app>
```

I. Apply the MVC model

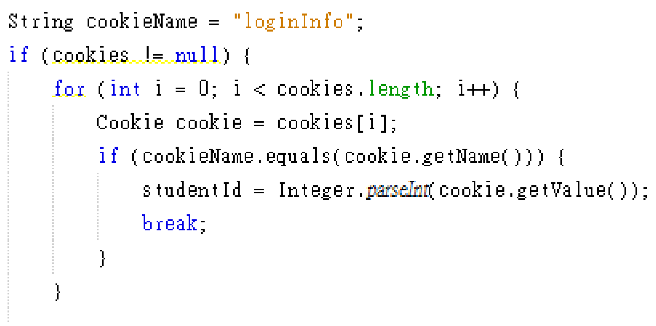
We use DB, JSP and servlet to apply the MVC



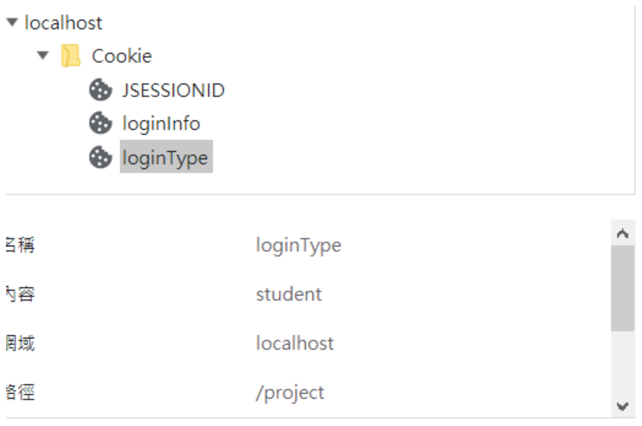
```
studentPageMenu.jsp
studentViewEquipment.jsp
studentViewRecord.jsp
studentViewWaitingList.jsp
technicianPageMenu.jsp
Source Packages
  ict.bean
    EquipmentBean.java
    EquipmentEventBean.java
    EquipmentReportBean.java
    OverdueReportBean.java
    SeniorTechnicianBean.java
    StudentBean.java
    TechnicianBean.java
  ict.db
    EquipmentDB.java
    EquipmentEventDB.java
    SeniorTechnicianDB.java
    StudentDB.java
    TechnicianDB.java
    TechnicianDB_new.java
  ict.servlet
    HandleLogin.java
    HandleStudentEquipment.java
    HandleTechnician.java
    HandleTechnician_new.java
  ict.tag
```

J. Other skills applied

We use Cooke to save the login state



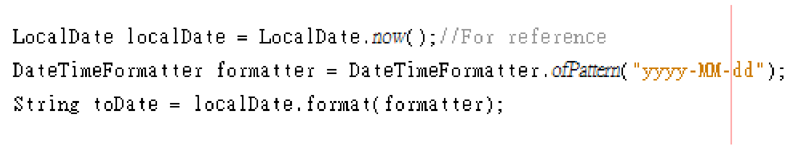
```
String cookieName = "loginInfo";
if (cookies != null) {
    for (int i = 0; i < cookies.length; i++) {
        Cookie cookie = cookies[i];
        if (cookieName.equals(cookie.getName())) {
            studentId = Integer.parseInt(cookie.getValue());
            break;
        }
    }
```



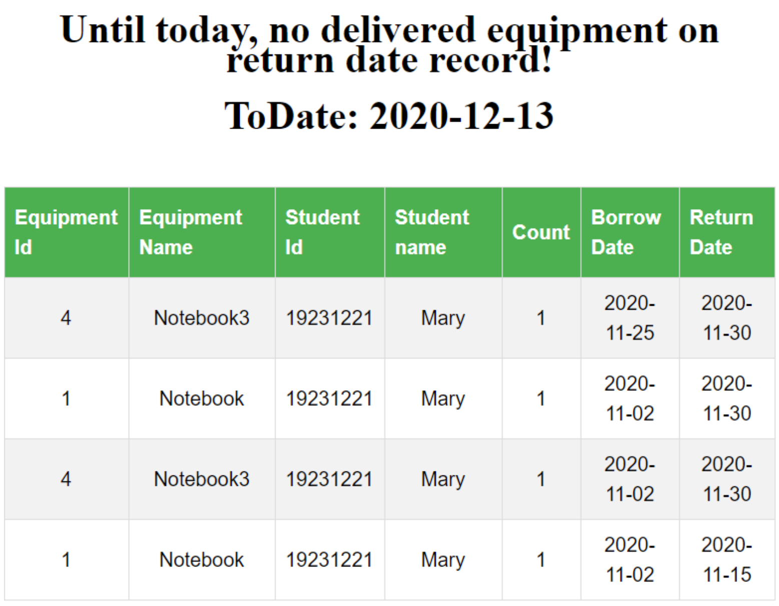
```
▼ localhost
  ▼ Cookie
      JSESSIONID
      loginInfo
      loginType
```

名稱	loginType
內容	student
網域	localhost
路徑	/project

In the report function in DB needs to calculating date time. So, we write java code to get the real time in java system.



```
LocalDate localDate = LocalDate.now();//For reference
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd");
String toDate = localDate.format(formatter);
```



Until today, no delivered equipment on return date record!

ToDate: 2020-12-13

Equipment Id	Equipment Name	Student Id	Student name	Count	Borrow Date	Return Date
4	Notebook3	19231221	Mary	1	2020-11-25	2020-11-30
1	Notebook	19231221	Mary	1	2020-11-02	2020-11-30
4	Notebook3	19231221	Mary	1	2020-11-02	2020-11-30
1	Notebook	19231221	Mary	1	2020-11-02	2020-11-15