EIE4432 Report

Concert Ticket Selling System

Group member:

Li Hongjin 21097395D

Zeng Mengyuan 21096444D

Table of Content:

Introduction

In this project, we are required to do the system of concert ticket selling. Totally we created login, registration (let the users register in the system), detail, Event Dashboard Dashboardpage for the admin), Event_user(the main page to users), the seat page for users to book tickets and payment pages for users to check whether the user's payment is successful or not. On the main page, there also a button called about me when the users click on the button, the page will change to the own page, and will t the user design their data they want to show in the system. For the admin, we designed two pages which are called management and seat_management. The page management lets the admin upload new concerts that will be held in the future and the seat management page helps the admin to design the concerts that will be held in different kinds of venue and design the price of tickets of different tickets.

Tech Stack :

Register:

In the registration page, we first added the text field to let the user enter their basic information and the image to let them upload their image from the local file. We use bootstrap to make the page look beautiful also we added the JavaScript file to the registration page to make user interact with server easily.

Check whether username or password is empty:

```
if (username.trim() === '' || password.trim() === '') {
    alert('Username and password cannot be empty');
    return;
}
```

Check whether the second input password is same as the first one:

```
if (password !== confirmPassword) {
    alert('Password mismatch!');
    return;
}
```

Also the same method to check whether the other text field is empty or not.

In html, I add a <p> that contains the alert of register successfully or register fail. In js file I add a judgement of whether the registration is successful or not:

```
if (role === "admin") {
    if (password !== "adminpass" || username !== "admin") {
        document.getElementsByClassName("fail")[0].classList.remove("d-none");
        document.getElementById("fail_ms").textContent = "Registration failed!";
    }
}
else{
    document.getElementsByClassName("successful")[0].classList.remove("d-none");
    document.getElementById("successful_ms").textContent = "Rgistration succeeded!";
```

When the register is successful the web will post the message and request to the auth/register.

```
fetch('/auth/register', {
method: 'POST',
body: formData,
})
.then(function (response) {
    if(role === "user"){
        return response.json();
    }
    else if(role === "admin" || username === "admin" ||password === "adminpass"){
        return response.json()
    }
})
.then(function (data) {
    if (data.status === 200 || data.status === 'success') {
    window.location.href = '/login.html';
    } else {
    alert(data.message);
    }
})
```

On the backend, we created a path to auth/login to send requests and read files.

On the backend, we create some method called init_userdb(), update_user(), validate_user(username, password) these functions help us reuse the same code easily in different post address.

In the init_userdb() this function called to read the file of database and load the data into a variable. Use username as an index to load different data.

In the update_user() method, this function change or add new information to the database. If it can find the username we looking for, it will return to whole data of the index we look for. If it cannot find the username, it will add the new information with the index of the username to the database.

In the validate_user() method, this function check and return the data in the user.

After that we post to the address auth/register :

We check some special detail if not match the requirements status 4xx and some wrong message to alert user, if the requirement is match it will return the data:

```
//检查username 和 password 是否为空:
if (!username || !password) {
  return res.status(400).json({ status: 'failed', message: 'Missing fields' });
}
//检查username是否小于三:
if (username.length < 3) {
  return res.status(400).json({ status: 'failed', message: 'Username must be at least 3 characters' });
}
//检查username是否存在:
if (users.has(username)) {
  return res.status(400).json({ status: 'failed', message: 'Username ' + username + ' already exist' });
}
//检查password是否小于八:
if (password.length < 8) {
  return res.status(400).json({ status: 'failed', message: 'Password must be at least 8 characters' });
}
//检查role:
if (role !== 'Admin' && role !== 'user') {
  return res.status(400).json({ status: 'failed', message: 'Role can only be either `Admin` or `user`' });
}
```

Finally, we export the route and import it to index.js.

Own (the page that can let the user upload their information):

In this page I added some title to let the user know which kind of information they can upload in this system:

```
<div class="card mt-4">
    <div class="card-body">
        <p class="card-text" id="Name">nickName: <span id="nickName"></span></p>
        <p class="card-text" id="first">FirstName: <span id="firstName"></span></p>
        <p class="card-text" id="last">LastName: <span id="lastName"></span></p>
        <p class="card-text" id="Email">Email: <span id="email"></span></p>
        <p class="card-text" id="birthday">Birthday: <span id="Birthday"></span> </p>
    </div>
    <button type="button" class="btn btn-primary" id="editButton">Update</button>
</div>
```

After that I added an overlay that can be input by the users and let it cannot see originally:

```css
.overlay {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-color: rgba(0, 0, 0, 0.5);
    display:flex;
    align-items: center;
    justify-content: center;
    z-index: 999;
    display: none;
}
```

Then I added an eventlistener that when the upload button was clicked the overlay will appear and user can upload the information they want.

```
    // 更新卡片内容
    nickName.textContent = nickNameInput.value;
    firstName.textContent = firstNameInput.value;
    lastName.textContent = lastNameInput.value;
    email.textContent = emailInput.value;
    birthday.textContent = birthdayInput.value;

    // 隐藏弹出层
    overlay.style.display = "none";
});

// 显示/隐藏输入框
editButton.addEventListener("click", function() {
    modal.style.display = "block";
});

// 隐藏输入框
saveButton.addEventListener("click", function() {
    modal.style.display = "none";
```

Seat and detail page:

When the user clicks the book button on the main page, they can see the seat pages where they can choose a seat to book. On this page there are many small squares which represent the seat in the venue and when user click on it the square will become green, when user book the seat successfully the square will become red and unable to click：

All of this is achieved by js file:

```
for (var j = 0; j < cols; j++) {
    var seat = document.createElement("td");
    seat.dataset.row = i;
    seat.dataset.col = j;
    if(j >= Math.floor(cols/2)-1 && j<= Math.floor(cols/2)
        seat.dataset.price = 500;
    }else{
        seat.dataset.price = 450;
    }
    seat.addEventListener("click", function() {
        var row = parseInt(this.dataset.row);
        var col = parseInt(this.dataset.col);

        if (seatMap[row][col] === 0) {
            seatMap[row][col] = 1;
            this.classList.add("selected");
        } else if (seatMap[row][col] === 1) {
            seatMap[row][col] = 0;
            this.classList.remove("selected");
        }

        updateSelectedSeat();
    });

    row.appendChild(seat);
    seatMapRow.push(0);
}
```

In this code you can see I also create a function to update Selected Seat information: In this function, the selected message also will be stored in a list and sessionstorage to let the detail page read the message and load it successfully:

```
var payButton = document.getElementById("pay");
payButton.addEventListener("click", function() {
  var seats = sessionStorage.getItem("seats");

  var detailUrl = "detail.html?message=" + encodeURIComponent(seats);
  window.location.href = detailUrl;
});
```

Also in the detail html show the data:

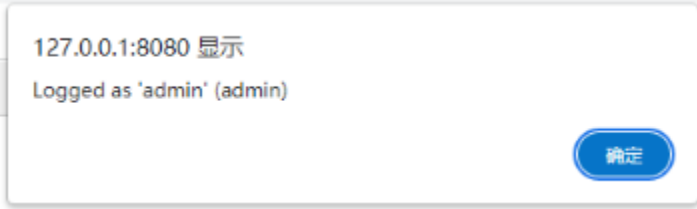
```
var seatCard = document.getElementById("SeatNumber");
var seatprice = document.getElementById("price");
// 在detail.html中
var queryParams = new URLSearchParams(window.location.search);
var seats = queryParams.get("message");
var price = sessionStorage.getItem("price");

// 将座位信息显示在卡片中
seatCard.textContent = "SeatNumber:  " + seats ;
seatprice.textContent = "Price: " + price;
```

For login page, I created two roles. And after login, there are different pages for different roles. For admin, you can see the management page, while the user can not see the management page. The following code, you can see that if the role is different, go to the page is different.

```
fetch('http://127.0.0.1:8080/auth/login', {
  method: 'POST',
  body: formData
})
  .then(response => response.json())
  .then(data => {
    if (data.status === "success") {
      alert(`Logged as '${data.user.username}' (${data.user.role})`);
      if (role === 'admin') {
        window.location.href = '../Event Dashboard.html';
      }
      else if (role === 'user') {
        window.location.href = '../Event_user.html';
      }
    } else if (data.status === "failed") {
      alert(data.message);
    } else {
      alert("Unknown error");
    }
  })
  .catch(error => {
    console.log('Error:', error);
  });
```

For the login account, the user was created on the previous register page. If there is no account, click the register buttonto register. I realized login by searching for user information and if the user exists and entered the correct id and password prompt login. As described in status (401):

```
if (role==='' || username==='' || password==='' ) {
  return res.status(400).json({ status: 'failed', message: 'Missing fields' });
}

const user = await validate_user(username, password, role);
if (user) {
  return res.status(201).json({ status: 'success', user: { username: user.username, role: user.role } });
} else {
  return res.status(401).json({ status: 'failed', message: 'Incorrect username and password' });
}
```



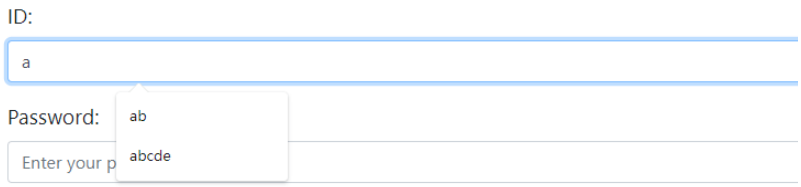
If the password is incorrect or there is no account, the corresponding statement appears in the console.

```
try {
  const user = users.get(username);
  if (!user) {
    console.log('User does not exist.');
    return false;
  }

  if (user.password !== password) {
    console.log('Password mismatch.');
    return false;
  }
```

```
if (!username || !password) {
  alert('Username and password cannot be empty');
  return;
}
```

I also made a remember ID option, check the box and remember the user id the next time you log in.

```
const rememberedId = localStorage.getItem('remember');
  if (rememberedId) {
      idInput.value = rememberedId;
      rememberIdCheckbox.checked = true;
  }

  loginButton.addEventListener('click', () => {
      const role = document.getElementById('role').value;
      const id = idInput.value;
      const password = document.getElementById('password').value;

      if (!id || !password) {
          alert('ID and password cannot be empty');
          return;
      }

      // 如果用户勾选了记住ID选项，则将ID保存到本地存储
      if (rememberIdCheckbox.checked) {
          localStorage.setItem('rememberedId', id);
      } else {
          localStorage.removeItem('rememberedId');
      }
  })
```



For Event Dashboard page, for admin, there is a navigation box at the top of the page to manage events.

```
<ul class="nav nav-pills nav-fill mt-2">
    <li class="nav-item col-2">
      <a class="nav-link active" aria-current="page" href="Event Dashboard.html">Event Dashboard</a>
    </li>
    <li class="nav-item col-2">
      <a class="nav-link" href="management.html">Event Management</a>
    </li>
    <li class="nav-item col-2">
      <a class="nav-link" href="seat management.html">Seat Management</a>
    </li>
</ul>
<hr class="border border-primary border-2 opacity-50">
```

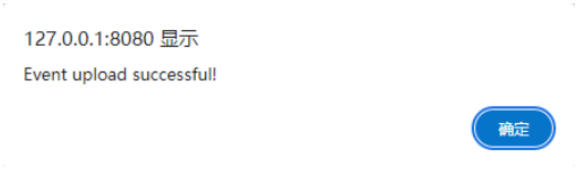
This page features a "card" format to present the concert list. You can see it in this list, including title, description, time, and so on. I also designed for search concert events.

```
<div class="row justify-content-center g-4">
    <div class="col-md-2">
        <div class="card" style="width: 18rem;">
            <img src="image/Clockenflap.png" class="card-img-top" alt="...">
            <div class="card-body">
                <h5 class="card-title">Clockenflap </h5>
                <p class="card-text">As a world-class all-round cultural and music festival, Clockenf
                <p class="card-text" id="date1">Date: 2/12/2023</p>
                <p class="card-text" id="venue1">Venue: Central Harbourfront Space</p>
                <p class="card-text" id="musician">musician: Clockenflap band</p>
                <a href="./seat.html" class="btn btn-primary mx-3">Book</a>
```

For the Event management page, I designed a picture box in the left side of the page, and when the user selects the file to upload the image, the image appears in the picture box. The code is shown in the following figure:

```
fileInput.addEventListener('change', function(event) {
  const file = event.target.files[0];

  if (file) {
    const reader = new FileReader();

    reader.addEventListener('load', function() {
      const image = document.createElement('img');
      image.src = reader.result;

      preview.innerHTML = '';
      preview.appendChild(image);

      placeholder.style.display = 'none';
    });

    reader.readAsDataURL(file);
  } else {
    placeholder.style.display = 'block';
  }
});
```

Then fill in the concert details on the right and click the submit button. At this point, a prompt appears, and it go to the Event Dashboard page.



I get the information filled in on this page and define it and write it into the new card.

```
document.addEventListener('DOMContentLoaded', () => {
  // 解析URL参数
  const urlParams = new URLSearchParams(window.location.search);
  const imageDataURL = urlParams.get('image');
  const name = urlParams.get('name');
  const date = urlParams.get('date');
  const price = urlParams.get('price');
  const venue = urlParams.get('venue');
  const musician = urlParams.get('musician');
```

Create a new card:

```
const card = document.createElement('div');
card.classList.add('card');
```

Then add the new card to Event Dashboard and create an empty write dashboard in Event Dashboard.

```
const dashboard = document.getElementById('dashboard');
if (dashboard) {
  dashboard.appendChild(card);
} else {
  console.error('Cannot find dashboard element.');
}
```

In this way, when clicking submit button, go to the Event Dashboard page and a creat new concert event.

For Seat Management page, create different seat map for different venues. First create a drop-down box and write to a different venues. I wrote three different finals so created three seating maps. When the venue changes, update the seating diagram:

```
document.getElementById('venue').addEventListener('change', function() {
  const selectedVenue = this.value;
  const seatMap = seatMaps[selectedVenue];
  updateSeatMap(seatMap);
});
```

Update the seating chart and clear the last option,

```
function updateSeatMap(seatMap) {
  const seatMapContainer = document.getElementById('seatMap');

  seatMapContainer.innerHTML = '';
```

Create svg and seats:

```
const svg = document.createElementNS('http://www.w3.org/2000/svg', 'svg');
svg.setAttribute('width', '800');
svg.setAttribute('height', '450');

// 创建座位
for (let i = 1; i <= seatMap.totalSeats; i++) {
  const seat = document.createElementNS('http://www.w3.org/2000/svg', 'rect');
  seat.setAttribute('class', 'seat');
  seat.setAttribute('x', (i % 10) * 30);
  seat.setAttribute('y', Math.floor((i - 1) / 10) * 30);
  seat.setAttribute('width', '25');
  seat.setAttribute('height', '25');
  seat.setAttribute('data-seat-number', i);
```

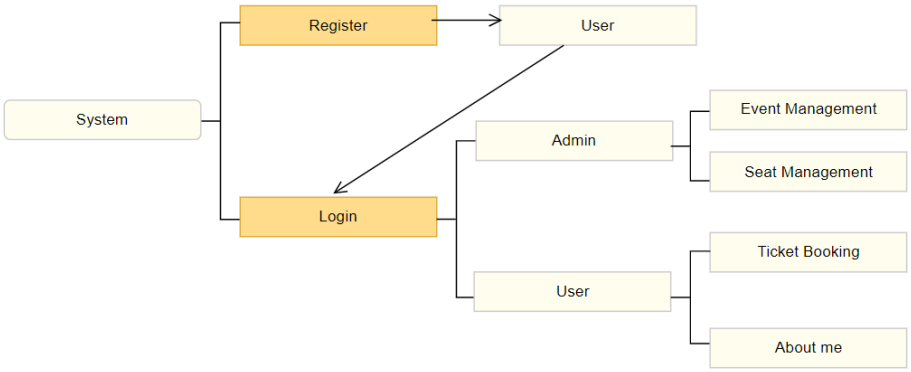
When clicking the seat, judge whether the seat is occupied, if occupied, show the seat number and occupancy information.

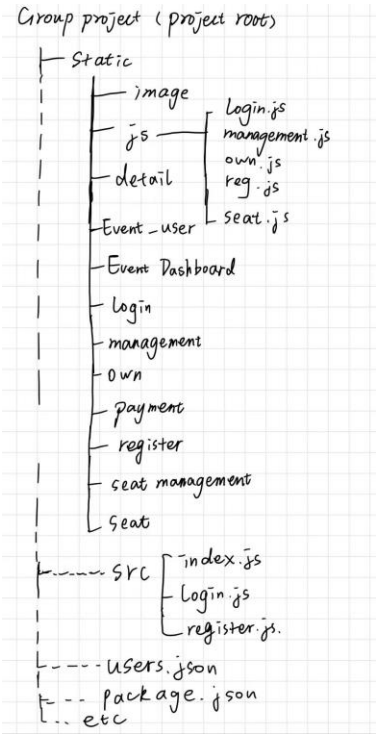
```
if (seat.classList.contains('occupied')) {
  alert('The seat number you selected is ' + seatNumber+'. Seat ' + seatNumber + ' has been occupied.');
} else {
  alert('The seat number you selected is ' + seatNumber);
}
```

Seats appear red if occupied and green if available.

```
.seat {
  fill: rgb(73, 180, 73);
}

.occupied {
  fill: rgb(248, 100, 100);
}
```

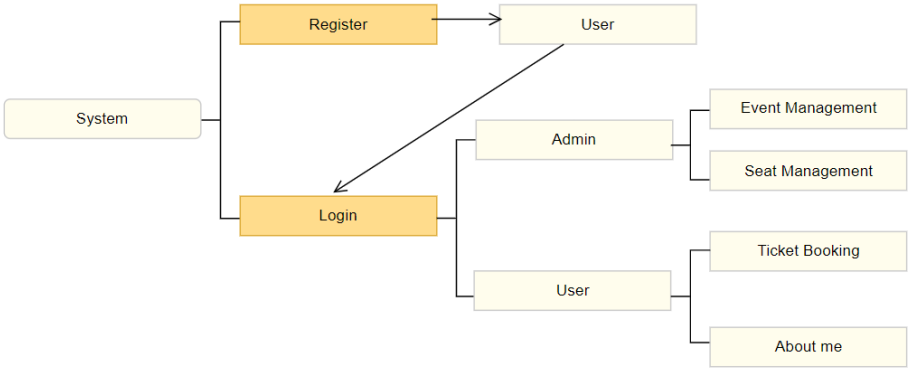
Website Structure :





When we enter the web page, we can choose to register or log in. Register can only register user accounts, not admin. For Login, there are two role options. Select admin to perform event management and seat management. Select user to book tickets and view my information.

Use Case Diagram :



Application Features：

	features	
features		Completed?
User Account Registration	User registration page: support standard information including user ID, password, nickname, email, gender, birthdate, etc.	Y
	Validation checks for input value, i.e., Password Policy and Duplicate User ID	Y
	Feedback message on validation failure and registration success/failure	Y
	Password stored in a database with password hashing	Y
	An admin doesn't need to register an account.	Y

features		
features		Completed?
User Login	Two roles: admin and users	Y
	Use user ID and password to log in. The admin uses the default ID “admin” and password “adminpass” to log in	Y
	User logout	Y

features		
features		Completed?
Event Dashboard	Show the list of events with event details (date/time, title, venue, description)	Y
	At least one event	Y
	Ticket sales for each event (Real-time ticket availability)	N

features		
features		Completed?
Ticket Booking Page	Select seats from the SVG seat map	Y
	View available and occupied seats in the SVG-based seat map (use different colors)	Y
	Real-time price calculation	Y

	features	
	features	Completed?
Payment Page	Show event details, seat selection and total price	Y
	Input payment details (details could be fake)	Y
	Feedback on payment status	Y
	how electronic tickets to buyers after successful purchas	Y
	Update the order status upon successful payment	N

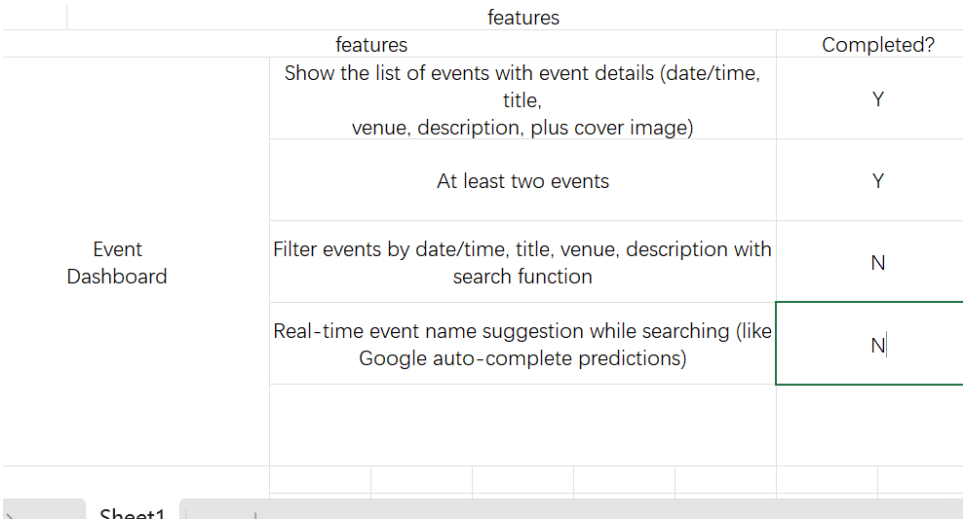
	features	
	features	Completed?
Seat Management Page (Admin only)	Create SVG-based seat maps (at least 40 seats) for all venues	Y
	View current available and occupied seats in the SVG-based seat map (use different colors)	Y
	All seats with the same price	Y

	features	
	features	Completed?
Event Management Page (Admin only)	Create new events (date/time, title, venue, description)	Y

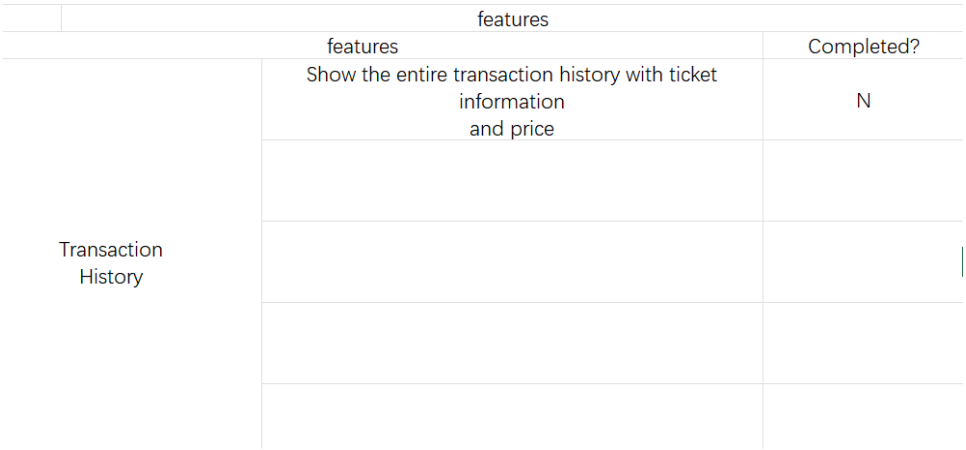
	features	
	features	Completed?
User Account Registration	User registration page: support profile image on top of other standard information.	Y

	features	
	features	Completed?
User Login	Choice to remember login user-id	Y
	Forget password feature	Y
	The user will be automatically logged out after a certain period of idle time	N

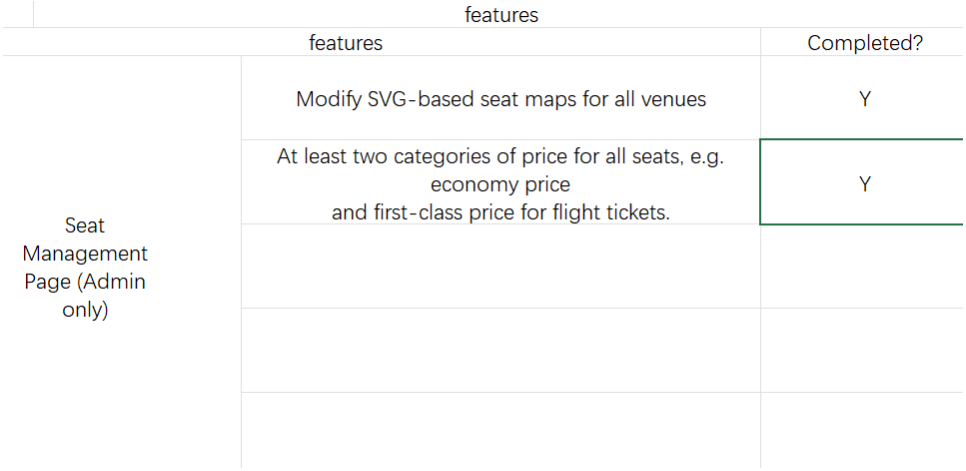
	features	
	features	Completed?
User Account Management	View user profile	Y
	Update user profile, including nickname, password, email and profile image	Y



features		
features		Completed?
Event Dashboard	Show the list of events with event details (date/time, title, venue, description, plus cover image)	Y
	At least two events	Y
	Filter events by date/time, title, venue, description with search function	N
	Real-time event name suggestion while searching (like Google auto-complete predictions)	N



features		
features		Completed?
Transaction History	Show the entire transaction history with ticket information and price	N



features		
features		Completed?
Seat Management Page (Admin only)	Modify SVG-based seat maps for all venues	Y
	At least two categories of price for all seats, e.g. economy price and first-class price for flight tickets.	Y

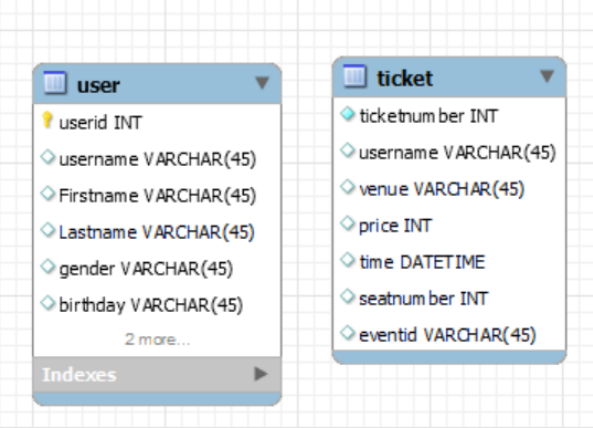
Advanced Features

Our system does not contain the advanced features.

Database Design

Description of tables and data fields of each table

We use mongodb to create our database.



user	
Column name	description
userid(pk)	It is the unique key that start with digit 6 to identify each access permission information
username(FK)	It is the field to identify the specific user
Firstname	It is the field to store the firstname of the user
Lastname	It is the field to store the lastname of the user
gender	It is the field to store the gender of the user
birthday	It is the field to store the birthday of the user
role	It is the filed to store wether the user is an admin or nomal user
email	It is the field to store the email of the user
password	It is the field to store the password of the user.

ticket	
ticketnumber(PK)	It is the unique key that will identify each ticket information
username	It is the field to store the user of the ticket
venue	It is the field to store the venue of the ticket
price	It is the field to store the price of the ticket
time	It is the field to store the time of the ticket
seatnumber	It is the field to store the seatnumber of the ticket

API Specification & Implementation:

In the javascript files we create more function which used a lot of time in the other part. The function can be reused and modified independently.

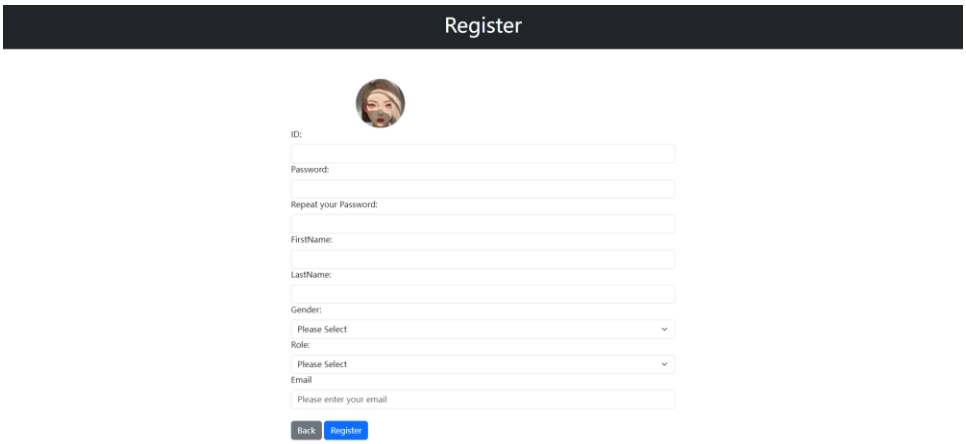
Also for the code we add more Comments to make the code understandable.

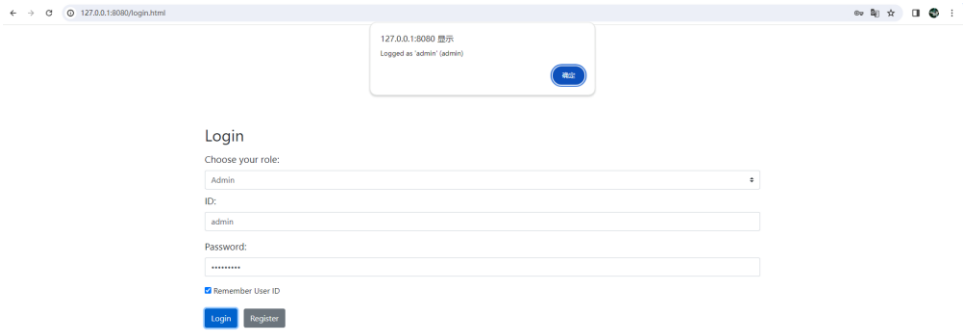
In the backend, we create a file login and add more paths to the same address auth and to the index, this makes the code efficiently

User test cases with screenshots :

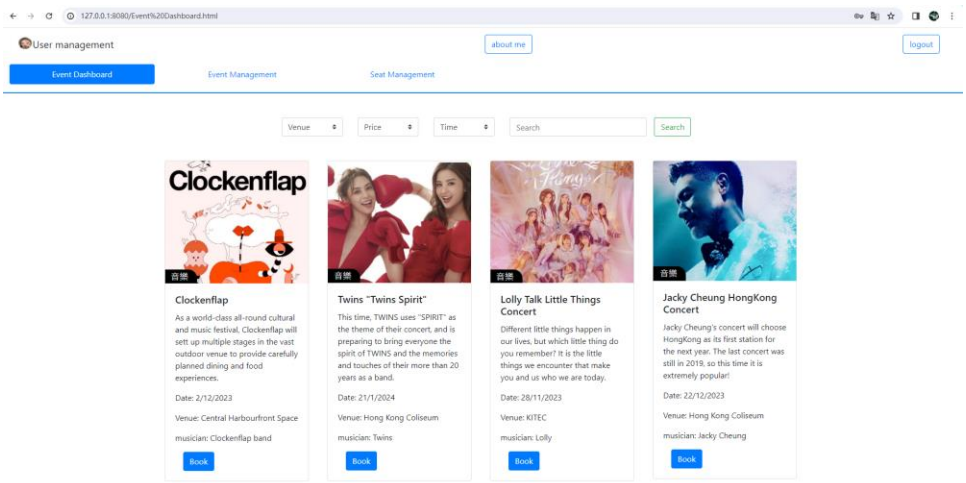
Login page:

Register:

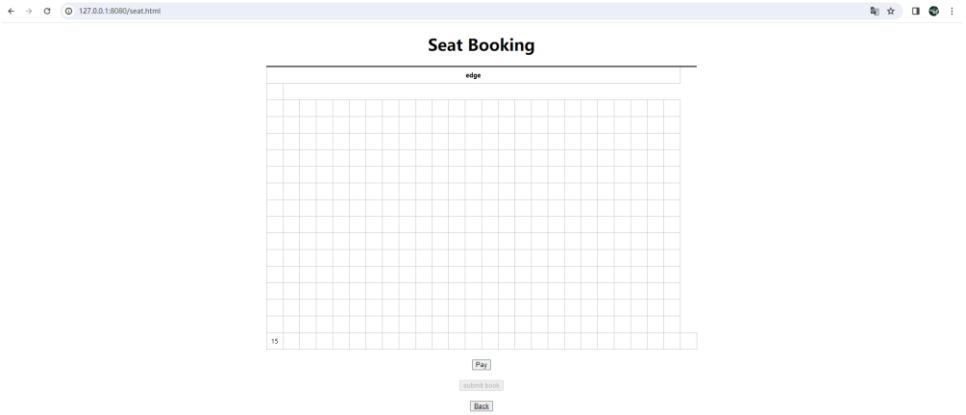


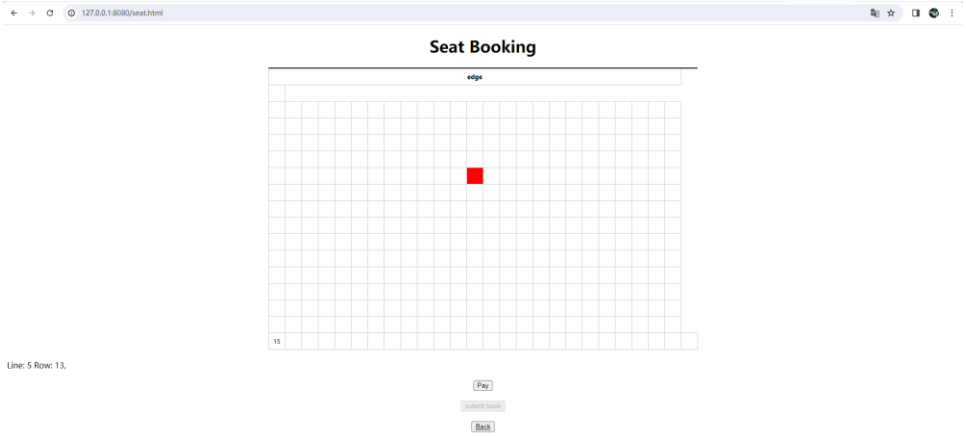


Event Dashboard:

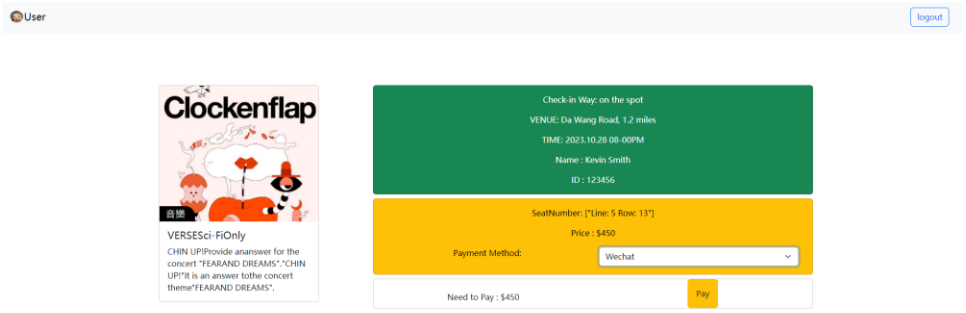


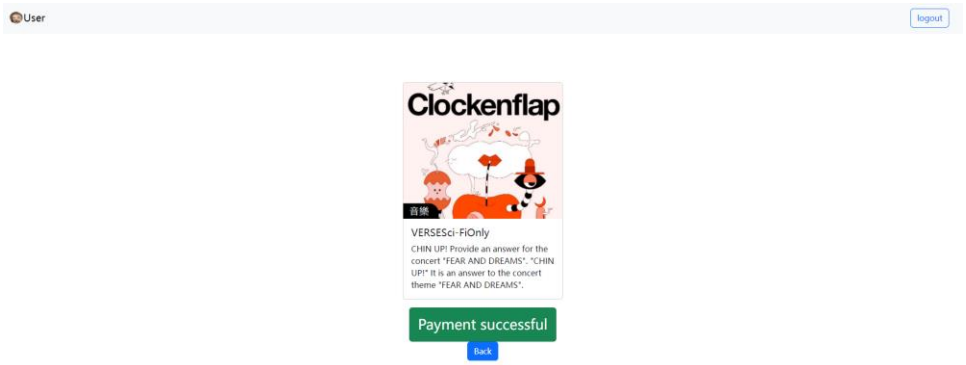
Ticket Book:



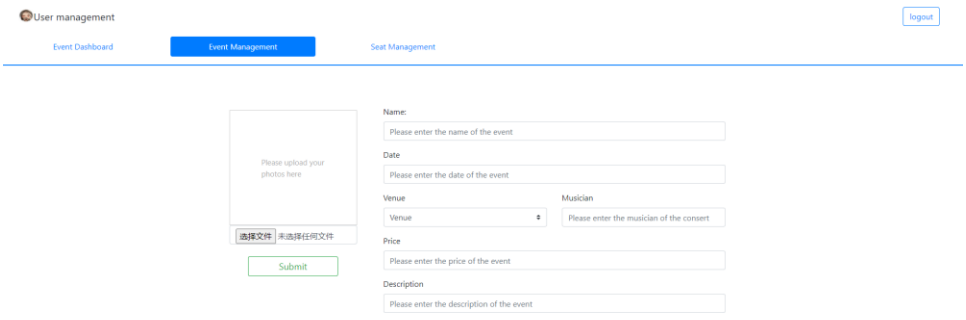


Payment:

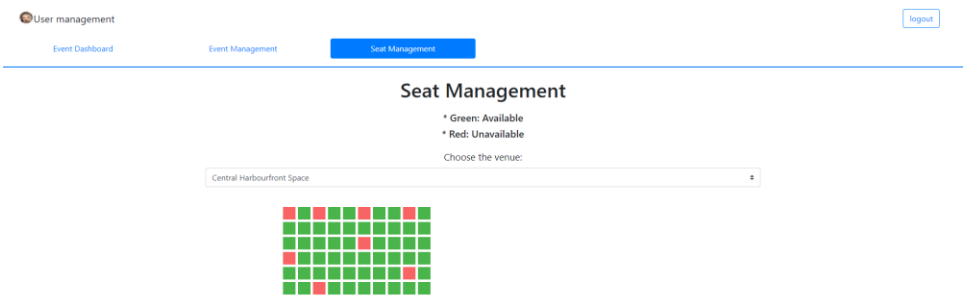




Event management:



Seat management:



Deployment:

This system can add more third-party payment links to the payment method, allowing users to pay for tickets more conveniently.

Conclusion:

This project was to make a Concert Ticket Selling System. In this process, we encountered many challenges, but finally successfully completed the project. Before starting the project, we completed the UI design drawing, determined the style of our website, and provided samples for the later production. We complete the website with a variety of technologies, including front-end (HTML, CSS, and JavaScript), back-end (Node.js and Express framework), and databases (MongoDB). For this system we created user login, registration pages, event management and seat management, etc.
We have realized some functions of the website according to the needs, including the display of concert information, user login and registration, ticket book processing and user information modification and other functions. The processing of the back-end JavaScript files is our biggest challenge. We finally completed some interactive functions through many modifications and debugging.
Through completing this concert website project, we all learned a lot of knowledge and skills about website development. We mastered the basic principles of front-end and back-end development better and learned how to interact with databases and manage data. We also improved our problem solving and teamwork. If we have the opportunity, we will learn more deeply and improve the system.

Work distribution：

Li Hongjin works for the registration part, own page, seat and detail page, payment successful page and the database.

ZengMengyuan works for the login part, Event Dashboard page, seat and event management(admin).

Li Hongjin: 50%

ZengMengyuan: 50%