COMP9900 Information Technology Project

Group : 9900 – H16P – PowerRangers

ProjectReport

*August*5, 2022

[Scrum Master]

Yanfeng Chen Front-end z5243188 yan.chen8@student.unsw.edu.au

[Group Members]

Haoyu Wang	Back-end	z5242694	z5242694@ad.unsw.edu.au
Yu Jia	Back-end	z5225083	yu.jia@student.unsw.edu.au
Zexuan He	Front-end	z5230400	zexuan.he@student.unsw.edu.au
Zhaoyang Wang	Front-end	z5292757	z5292757@ad.unsw.edu.au

1. Overview **3**
1.1 Background 3
1.2 Software Architecture 4
2.Functionality **4**
2.1 User identification 5
2.2 Event listings page 5
2.3 View the event page 6
2.4 Creation and management activities 7
2.5 Search and recommendation systems 8
2.6 Coupon Placement 9
2.7 Order Trading System 10
3. Third party functions **11**
3.1 Front End 11
3.2 Back End 15
4.Challenges in the development process **17**
4.1 Front-end challenges 17
4.2 Back-end challenges 18
5.User documentation **20**
5.1 Overview 20
5.2 Frontend 20
5.3 Back-end setup 22
5.4 Browser Access 22
5.5 Function Manual 23
6.References **26**

1. Overview

1.1 Background

There are many ways to spend your leisure time, and almost everyone has spent their free time by watching shows or movies. There are many people who have been passionate about a certain celebrity or a certain type of show, and buying tickets to the show is obviously a must. For users, having a website that brings together a variety of shows at different times and places can provide a lot of convenience. Likewise, for businesses selling tickets to shows, they need a platform that helps them find consumers. That's why we designed this site, so that users can see the availability of tickets for various shows at any time, and bring together different merchants on this site so that consumers can spend money according to their own preferences.

This type of e-commerce website is not uncommon, and consumers can choose from a variety of platform websites when they want to buy performance tickets. We believe that, as an excellent ticketing e-commerce trading platform, it is important for consumers to be able to buy the most desired tickets with the least amount of money, such as the ability to receive coupons, and the recommendation system on the home page are more important; for merchants, not only is it need a tool that can publish goods on the website, but also to be able to operate and manage customers, as well as marketing of goods, which is the platform that provides help to merchants. For example, Ticketmaster (https://www.ticketmaster.com/) does not have a user recommendation system or a coupon system. The design of this website is very simple, but users want to complete the whole process of buying tickets, which requires many The design of this website is very simple, but users want to complete the whole process of buying tickets, it is necessary to take many steps, here is also the point we consider in our website need to be optimized.

These observations allowed us to focus on optimizing the user experience in the process of developing the features. For example, on the merchant side, we designed features that allow merchants to send mass emails and SMS messages to consumers, so that they can regularly maintain their relationships with consumers and increase their repurchase rate. We also designed the event listings page on the consumer side to filter by price, city, event type, and other dimensions, so that consumers can more efficiently find shows that they want to buy.

1.2 Software Architecture

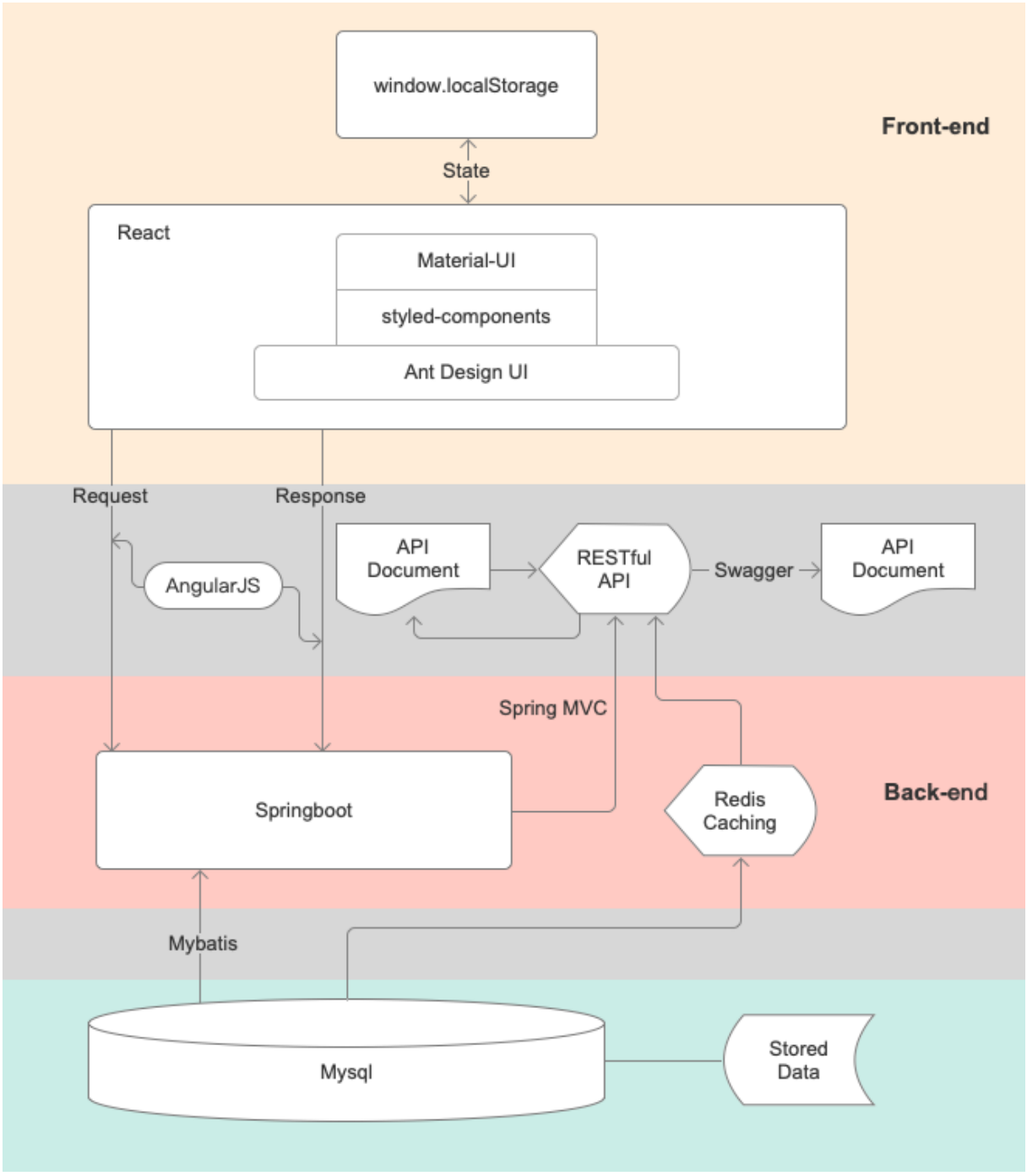

Figure 1:Front and back-end functional diagram

2.Functionality

The selected topic for our project was a ticketing website, and we completed all the functions specified in the project, while adding the following innovations: a recommendation system, a coupon management system, and merchant data analysis functions, which will be described in detail in the

subsequent feature descriptions.

2.1 User identification

As a ticket trading website, users can be divided into two main categories according to their claims of using the website: host users and customer users. In addition, based on the user's status, we will divide the users into guest users and authenticated users. In general, there are three types of users that may exist in the website.

(1) Guest users. That is, users who are not registered or logged in, and they can only stay at the login screen without completing the login or registration process, and cannot view the content of the website.

(2) Hosting users. Users who can become host users must not be guest users, they have already completed the registration and login operations and have chosen to log in to the website as host users. This means that their main purpose of using the site is to sell tickets as a merchant, and they will provide consumable content to every visitor who comes to the site.

(3) Custom Users. Again, users who can become custom users must have completed the registration and login process and have chosen to interact as a custom user. This means that their primary purpose for coming to the site is to browse for tickets or purchase tickets, and that they play a role in consuming content on the site and providing the impetus for merchants to post ticket items for shows.

Users who want to register a website identity need to define a user name and enter their email address, we will send a verification code to the user email address, after successful verification, users set their login password to login and access the website with this account.

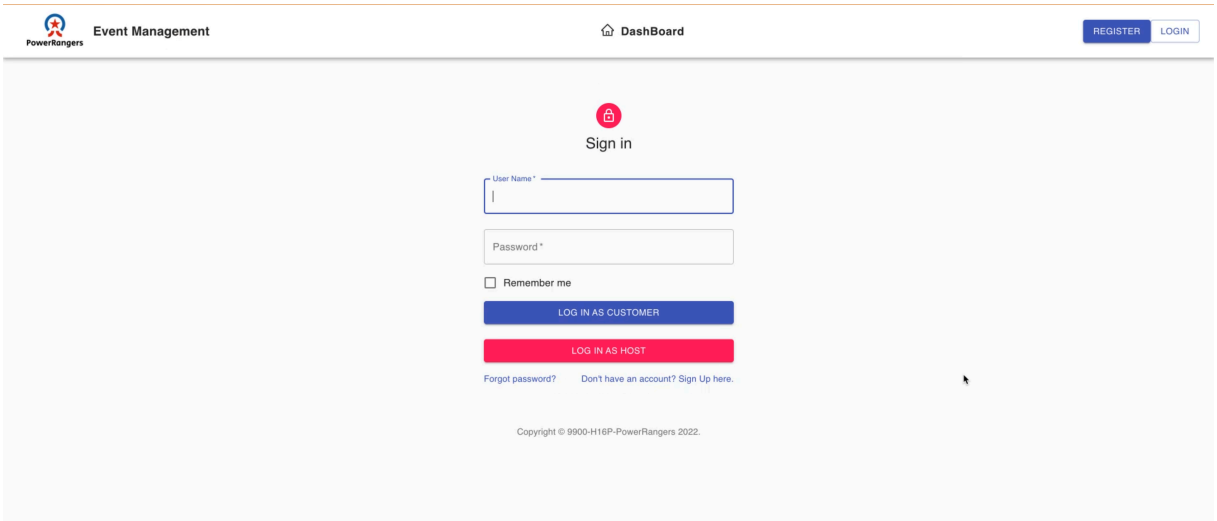

figure 2:Login screen

2.2 Event listings page

When the user logs in successfully, he or she will be taken to the website's events home page. Here all the performances available for trading in the site will be listed. The cards of these events will display the event picture, event name, event tag, event price and other information, and the user can get a general idea of the event on the home page.

In addition, users can also filter and find them in various ways according to their needs.

(1) Users can search according to the name and description of the event. Once users input the specific content and search, the website will filter out the events that match the search results and show them to users.

(2) Users can also filter by price range, type tag, and event time, and filter out the events that meet their desires for viewing.The search system will be described in more detail later.

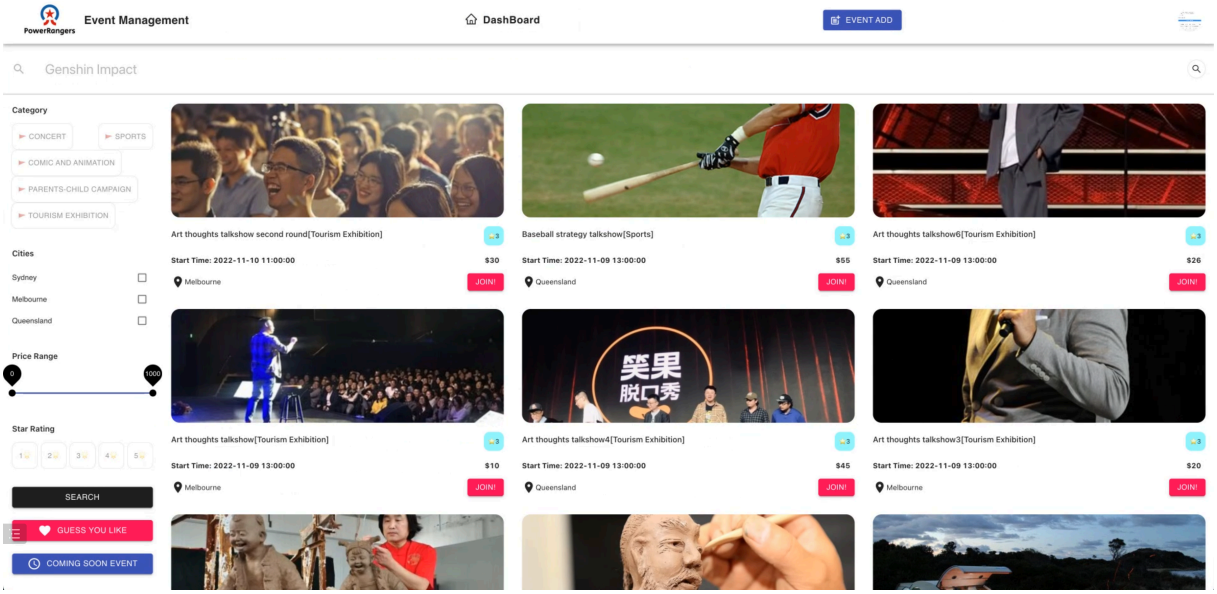

figure 3:Main page

2.3 View the event page

When the user clicks on a specific event, it will go to the detail view page of this event. In this page, the remaining available quantity of the event, the price, the offer, the time and place of the event, and the rating of the event by other users are displayed and the average score is calculated.

When a user wants to buy this event, the user has to choose the quantity they want to buy first, and the backend will calculate the price for the user. When the user selects more quantity than the remaining stock of this event, the user will not be able to complete the purchase and the front-end of the website will pop up an error alert for the user.

If a coupon exists for this event, the user can collect the coupon on this page, and the user can use the coupon to reduce the price during the transaction after receiving it successfully.

Users can also rate the event and write their own comments. Once the user completes the comment, the score will be calculated and averaged with other users' scores for the event and displayed on the detail page of the event.

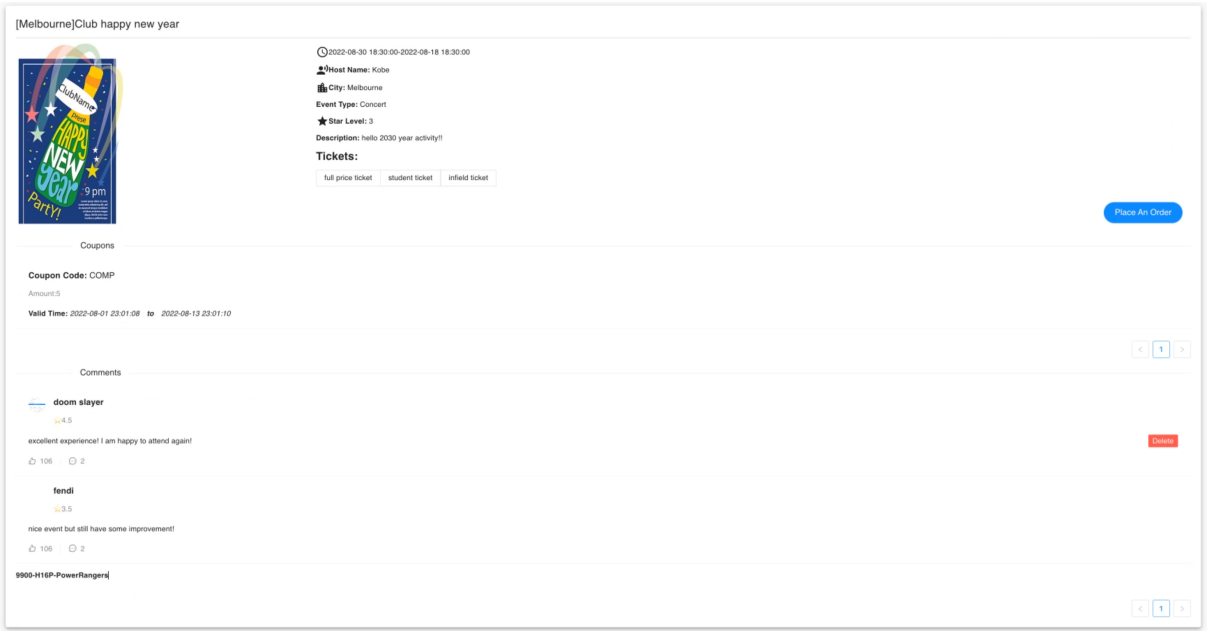

figure 4:event page

2.4 Creation and management activities

If the user chooses to log in as host, then the user can create events and manage and edit them.

After the user has successfully logged in, the event can be created via the create button in the top-bar of the website. Users can also click on their avatar button to access their personal center and manage the events they have created.

(1) Create event: After the host user logs in successfully, click the create button to enter the event creation process, the user needs to submit the name, main picture, performance time, description, ticket time, ticket price and number of tickets available, and also can choose whether to generate the event offline verification code according to the need, after the information format is verified, the event can be successfully created. After successful creation, the backend will generate and record a unique ID for each event, and if the user chooses to generate an event offline QR code, the backend will record the event and reserve the QR code inventory according to the number of tickets available entered by the user.

(2) Manage events: After the host user creates a successful event, he can modify and stop the created event in the personal center page. In the personal center page, the events created by the host user are displayed. For each event that is being sold, the user can edit it and end it early. For each event that has been ended (after the sale time), the user can delete it. When the user clicks the edit button, it will enter the editing process of this event, and the information that the user filled in last time will be automatically filled in the front of the web page, where the user finishes saving after making changes, and then the event in the web page will be updated to the latest content information.

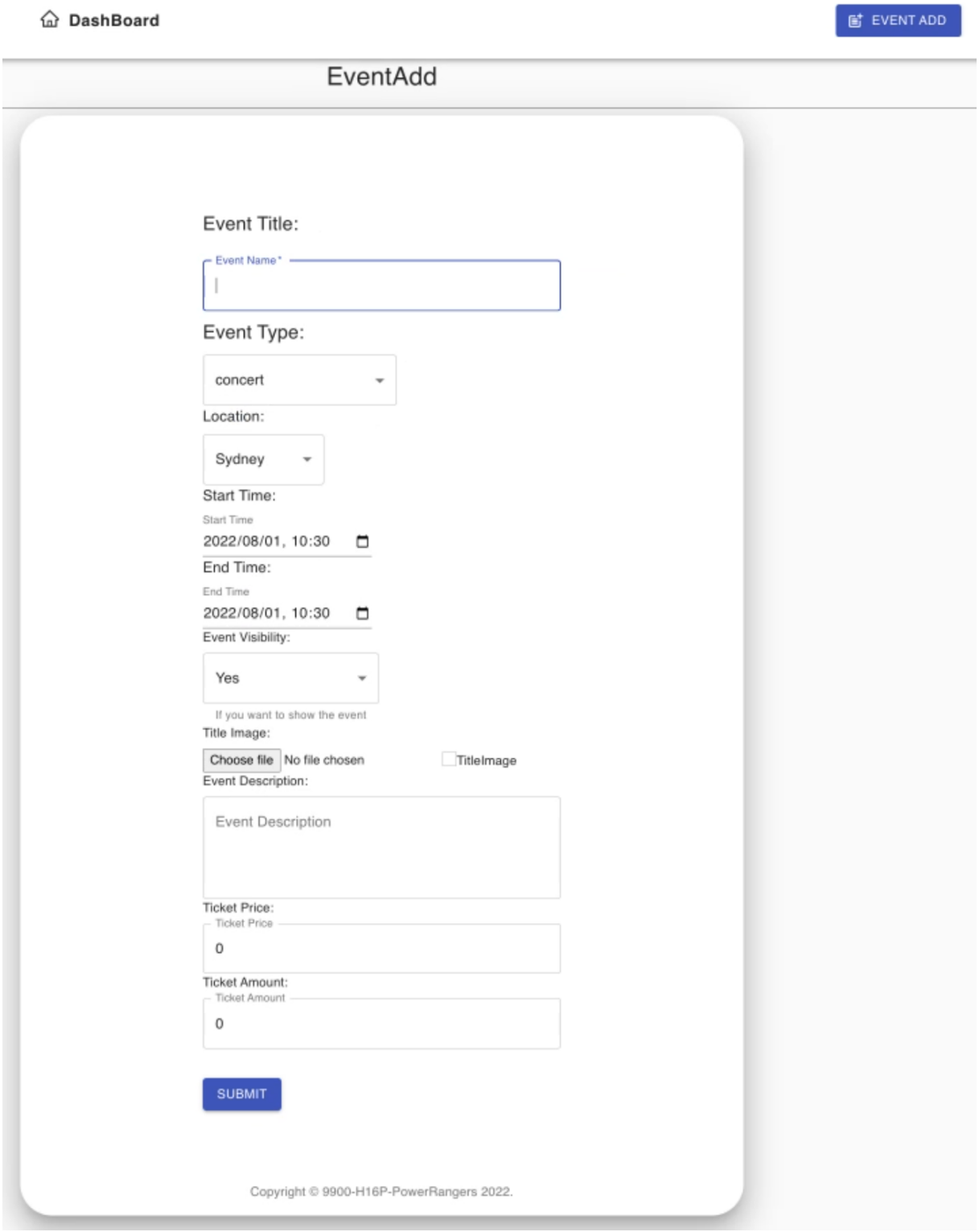

figure 5:Event creation page

2.5 Search and recommendation systems

We have additionally designed a search system and a recommendation system. There is a global search box and an event-specific search box on the home page of the website. In addition, there is a guess you like button, users click this button, the system will recall some event results that want to recommend to

users based on user behavior data.

(1) Search system: The search function is available for both host users and customer users. Users can filter and retrieve based on event name, event description, event type, and host name that created the event. The user input will be matched to the corresponding ID in the background, and the regular matching result based on the ID will be recalled and displayed to the user. The whole recall logic we follow the cosine theorem:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}.$$

(2) Recommendation system: When the user clicks the Guess Your Favorite button on the home page, the system will sort the recalled results according to the user's consumption record. If the user has no consumption record, the system will randomly recommend 10 popular events to show to the user. If the user has consumption records, then the system will recommend according to the type of events that the user has consumed, and the similar events will be sorted first.

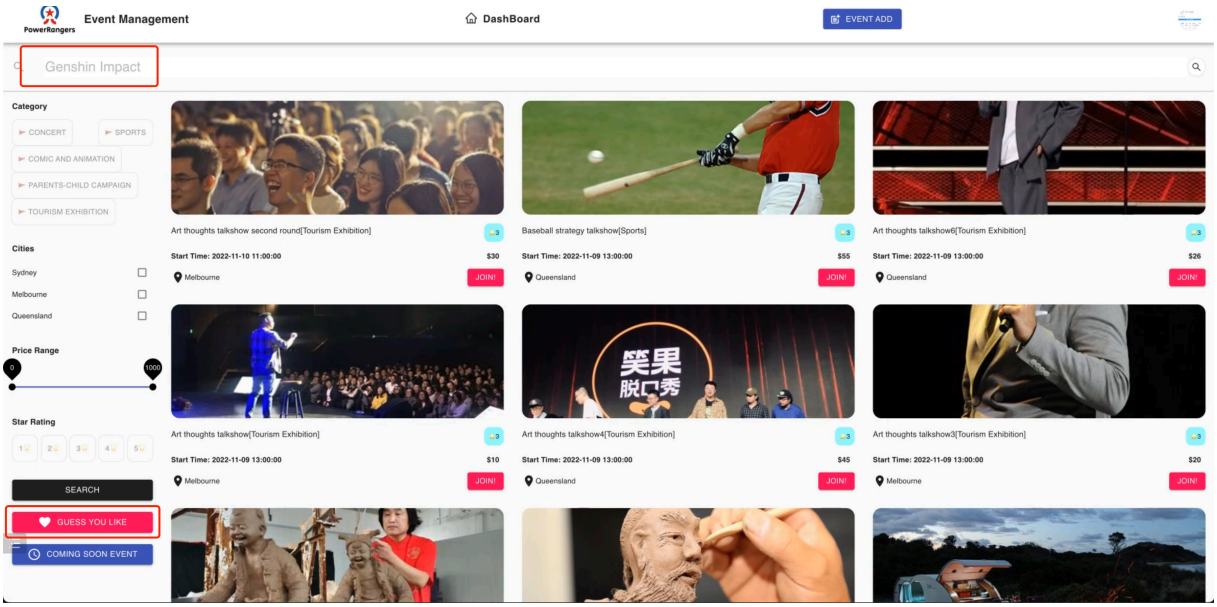

figure 6:Search Recommendation System

2.6 Coupon Placement

In order to have a closer link between host and customer, we designed a coupon system where host users can create coupons for a certain activity or send targeted coupons to certain users; similarly, customer users can receive coupons in a certain activity or get coupons directly from host users.

(1) Host gives coupons to customers: host can give coupons to customers in two ways. One is to issue coupons to this activity by clicking Create Coupon in the Activity Management page of Personal Center. The host user needs to set the threshold of coupon usage, the amount of reduction and the number of coupons issued. Any customer user can receive coupons in this campaign and use the discount when purchasing. Another way is to issue coupons to certain users in the customer

management page by clicking the Issue Coupon button, then the coupons will be issued directly to the accounts of these users.

(2) Customer users can get coupons: Customer users can get coupons for a specific event by clicking the Get Coupon button on the detail page of that event. If the remaining stock is 0, the user will fail to get the coupon. At the same time, users can also check all the coupons they currently have in the coupon page of their personal center. In the process of transaction, you can use the coupons you have and get price reduction.

In order to have a closer link between host and customer, we designed a coupon system where host users can create coupons for a certain activity or send targeted coupons to certain users; similarly, customer users can receive coupons in a certain.

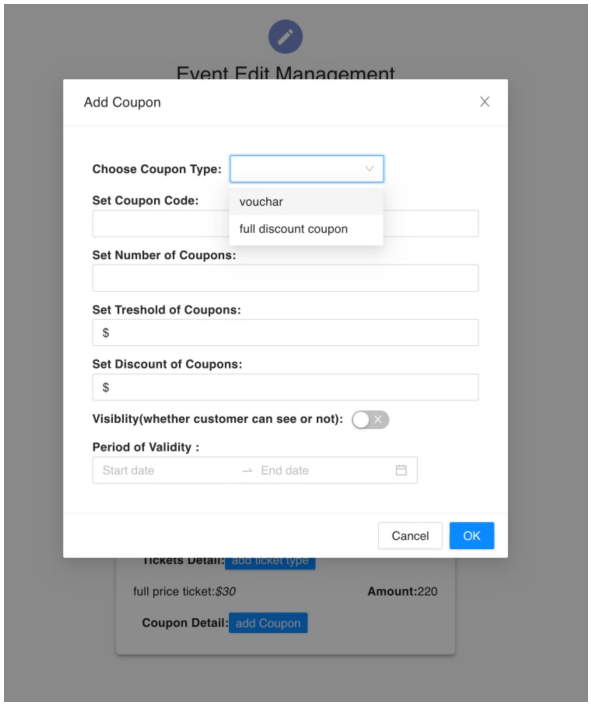

figure 7:Coupon Management

2.7 Order Trading System

Customer users can evoke the transaction process by clicking the participate button on the home page and event details page of the website.

The user can select the number of tickets he wants to buy and enter the order payment screen, and the system will calculate the amount to be paid by the user. Here the user can choose the payment method they want to use, such as entering their bank card number or directly scanning the QR code to pay.

Once the user has paid successfully, this order will be recorded into the system backend and user record. Users can view their transaction records and transaction details in their personal center. The backend will also record user behavior and use it in the recommendation system and host's data analysis.

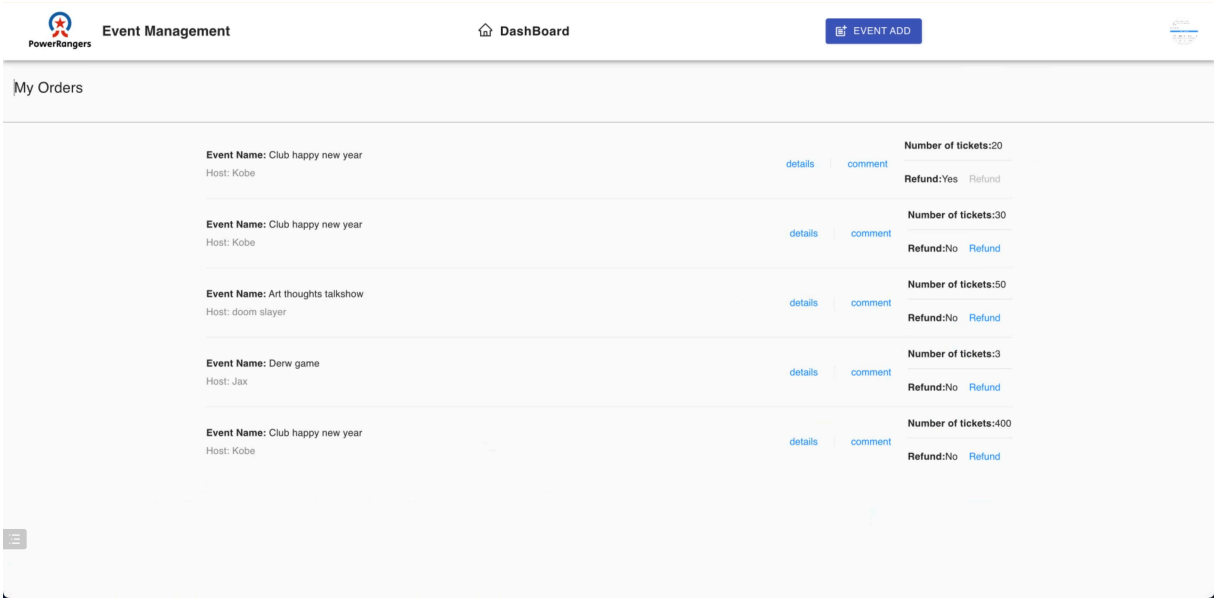

figure 8:Order Management System

3. Third party functions

3.1 Front End

On the front end we use a system framework of react and nodejs.

3.1.1 React

Where react is based on an internal facebook project, also known as React.js or ReactJS, is a JavaScript library for building user interfaces. Originally used to build Instagram's website and open sourced in May 2013, React is high performance and its declarative, componentized nature makes writing code easy. As the React community grows, more and more people are getting into learning and developing React, making it possible to develop not only web apps, but also desktop apps, TV React can also be used to build UI's. Users can pass many types of parameters in React, such as declarative code to help render UI's, static HTML DOM elements, dynamic variables, or even interactive application components.

React features the following[1].

1. Declarative design: React uses a declarative paradigm that makes it easy to describe applications.

2. Efficient: React minimises interaction with the DOM by emulating it.

3. Flexible: React works well with known libraries or frameworks.

Advantages:

1. Fast: During UI rendering, React enables local updates to the actual DOM through micro-operations in the virtual DOM.

2. Cross-browser compatibility: The virtual DOM helps us solve the cross-browser problem by providing us with a standardised API that is fine even in IE8.

3. Modularity: Write separate modular UI components for your application so that when something goes wrong with one or some components, it can be easily isolated.

4. Unidirectional data flow: Flux is an architecture for creating unidirectional data layers in JavaScript applications, which was conceptualised by Facebook with the development of the React view library.

5. Isomorphic, pure javascript: Because search engine crawlers rely on server-side responses rather than JavaScript execution, pre-rendering your application helps with search engine optimisation.

6. Good compatibility: you can use RequireJS to load and package, while Browserify and Webpack are suitable for building large applications. They make those tough tasks less daunting.

Cons: React is not a complete framework in itself, so it needs to be deeply coupled with ReactRouter and Flux if a large project requires a complete set of frameworks.

We discussed this within the group and after assessing the overall difficulty of the project and the cost of learning to use the tools, we chose react as the final system framework.

3.1.2 NodeJS

After deciding to use react as the most basic framework capability, we chose a node.js-based environment because Node.js provides a good development base and package management[2]:

Base class libraries: jquery, lodash, etc.

front-end frameworks: react, angular, etc.

Back-end frameworks: express, koa, etc.

testing-related: karma, jasmine, etc.

build-related: gulp, webpack, etc.

easy-to-use package management tools: npm.

NodeJS also has the following features: single-threaded, non-blocking I/O, event-driven

Single-threaded: in server-side languages like Java, PHP or .net, a new thread is created for each client connection. Each thread consumes about 2MB of memory, which means that theoretically, a server with 8GB of memory can connect to a maximum of about 4000 users at the same time. For a web application to support more users, the number of servers needs to be increased, and the hardware cost of the web application of course goes up. Instead of creating a new thread for each client connection, Node.js uses just one thread. When a user connects, an internal event is triggered, allowing Node.js programs to be macroscopically parallel as well, through a non-blocking I/O, event-driven mechanism. With Node.js, a server with 8GB of RAM can handle over 40,000 simultaneous user connections. The benefits of single-threading, along with the fact that the operating system no longer has the time overhead of creating and destroying threads at all.

Non-blocking I/O: When accessing a database to obtain data, it takes a while. In a traditional single-threaded processing mechanism, after the database access code is executed, the entire thread is paused and waits for the database to return results before executing the code that follows. In other words, the I/O blocks the execution of the code, significantly reducing the efficiency of the execution of the program. Because of the non-blocking I/O mechanism in Node.js, the code that accesses the database is executed immediately, and the code that follows it is executed in a callback function, thus improving

the efficiency of the execution of the program. When a certain I/O is executed, the thread performing the I/O operation will be notified in the form of an event, and the thread will execute the callback function for this event. In order to handle asynchronous I/O, threads must have event loops that constantly check for unhandled events and handle them in turn. In blocking mode, a thread can only handle one task and multiple threads are required to increase throughput. In non-blocking mode, a thread is always performing a computation and the CPU core utilisation of this thread is always 100%.

event-driven: In Node, client requests to establish connections, submit data, etc., trigger corresponding events. In Node, only one event callback function can be executed at a time, but in the middle of executing an event callback function, it is possible to switch to another event (e.g. a new user connects) and then return to continue executing the callback function for the original event. js is written in C++ (V8 is also written in C++). Nearly half of the underlying code is used to build the event queue and callback function queue. The server's task scheduling is done with event drivers, using a single thread that is tasked with handling a very large number of tasks.

Based on these three features, we found that: NodeJS is good at I/O, not good at computation. And for our development of an online ticketing system with high concurrency and low computation, the NodeJS framework was chosen for development.

3.1.2 UI Component Library

3.1.2.1 material UI

After our discussion, we finally chose Material UI as our front-end UI library: Material-UI is a collection of React components that implements Google's Material Design principles, using Material-UI can make our pages more colorful and animated (in line with Material Our original intention for the online ticketing system was to bring the "real world interactive experience to the screen", removing the impurities and randomness of reality, retaining the original and purest form, spatial relationships, changes and transitions of the design, and working with the flexible nature of the virtual world to restore the closest to the real Material Design's design principles are as follows[3].

● physicality is metaphorical (creating a distinctive touch through systematic kinetic effects and sensible use of space)

● sharp, graphic and well thought out (a basic graphic design specification is developed to enhance the user experience)

● Meaningful animations (sensible and meaningful animations that capture the user's attention and maintain a continuous experience throughout the system).

The design principles are in line with our pre-conceived product experience and style: we wanted to give users the ultimate real-world experience online.

In addition, material UI is suitable for small teams or individual projects to quickly build front-end interfaces, is cheap to learn, and is open source and free.

3.1.2.2 Ant Design

In addition to material UI, we also use the Ant Design component library, which has the following advantages[4].

1.Ant Design language support js and ts.

2. Provides a variety of UI components with good experience: various charts, paging, menus, pop-up boxes, selectors The mobile version also has a component library with a wide range of application scenarios, including list refresh effects, pop-up selectors, tab tabs, etc.

3.There are special versions and mainstream development frameworks used in conjunction with.

(1) Angular (front-end development framework maintained by Google)

(2) Vue (a progressive framework for building user interfaces)

(3) for their own corporate framework, Ant Design has opened up the design specifications to facilitate users to better implement their own version of the technology stack

4. Provides mobile version: Ant Design Mobile a UI component library based on Preact / React / React Native

3.1.3 Third party libraries

We use the following libraries:react moment, styled-components, react-router-dom

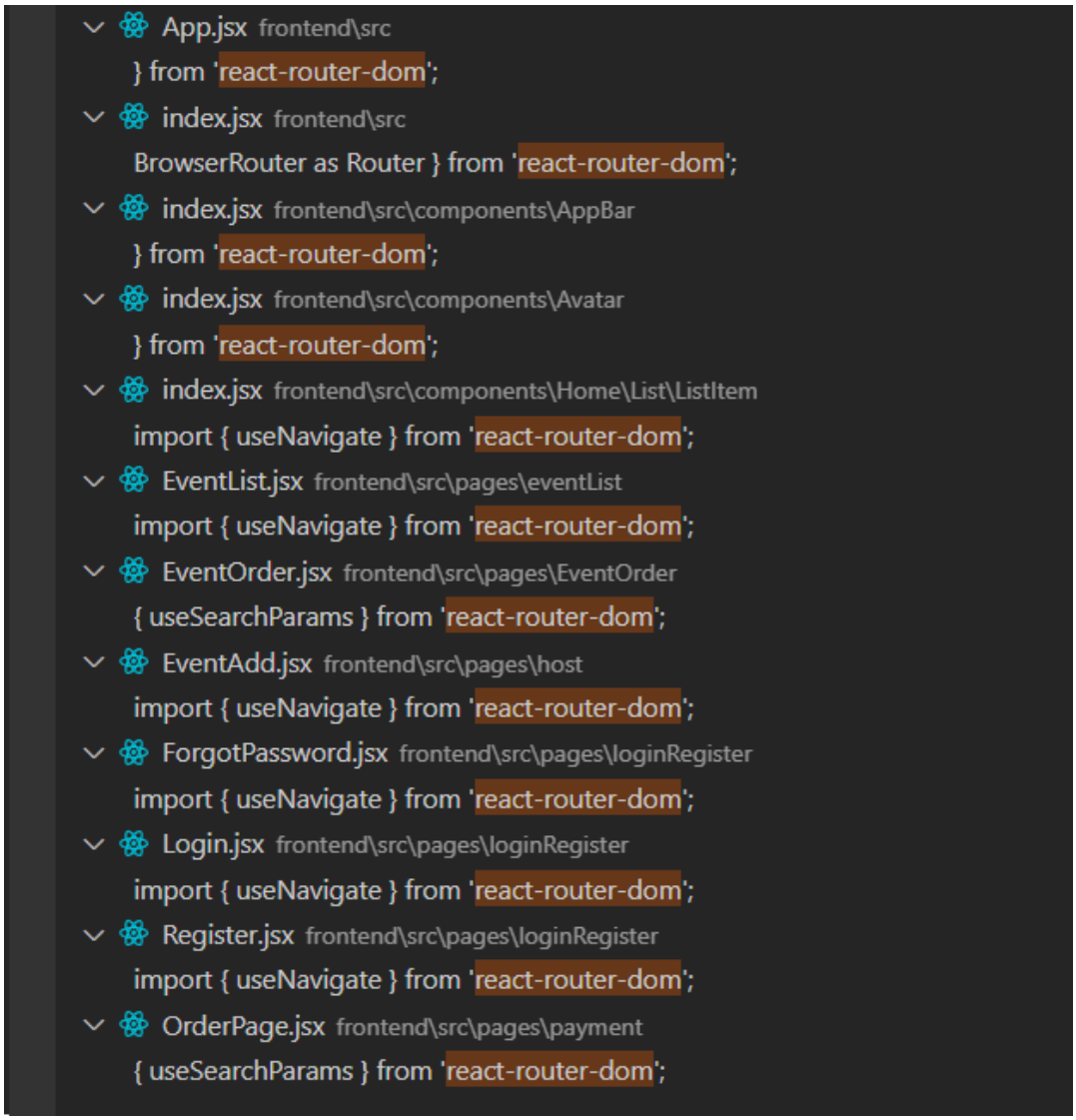

figure 9:libraries

3.2 Back End

On the backend we use SpringBoot 2.0, a holistic Java-based development framework. Spring Boot is a new framework provided by the Pivotal team and is designed to simplify the initial build and development of new Spring applications. The framework uses a specific approach to configuration so that developers no longer need to define sample configurations. To put it in my terms, Spring Boot is not really a new framework, it is configured to use many frameworks by default, just as maven integrates all jar packages, Spring Boot integrates all frameworks[5].

Spring Boot simplifies the development of Spring-based applications and allows you to create a standalone, product-level Spring application with a small amount of code.

It offers the following benefits.

- Use the Spring project boot page to build a project in seconds
- Easy export of various forms of services such as REST API, WebSocket, Web, Streaming, Tasks
- Very clean security policy integration
- Support for relational and non-relational databases
- Support for runtime embedded containers such as Tomcat, Jetty
- Powerful development packages with hot start support
- Automatic dependency management
- Self-contained application monitoring
- Support for various IEDs, such as IntelliJ IDEA, NetBeans

The disadvantage is that it is more integrated and less easy to understand the underlying layers during use.

3.2.1Building RESTful APIs

The idea of separating the front and back ends; based on several design patterns: the Proxy pattern, the Singleton pattern, the Factory (Spring IoC Dependency Injection) pattern, and the Decorator pattern.

The design of the back-end project structure is based on the Spring MVC idea (Proposal can be seen)

In the overall framework, we use the idea of front-end and back-end separation: the front-end and back-end are developed independently, on two different servers, and need to be deployed independently. They belong to two different projects, using two different code bases and different developers, and the front-end and back-end engineers agree on the interaction interface to achieve simultaneous development. The front-end only needs to focus on the style of the page and the parsing and rendering of dynamic data, while the back-end focuses on the specific business logic[6].

Based on the following design patterns.

1. Proxy pattern

provides an alternative way of accessing the target object; that is, through a proxy object to access the target object. The advantage of this is that additional functional operations can be enhanced on top of the target object implementation, i.e. extending the functionality of the target object. This type of design pattern is a structural pattern. It also plays the role of intermediary isolation: in some cases, a

client class does not want to or cannot directly reference a delegate object, and the proxy class object can play the role of intermediary between the client class and the delegate object, which is characterised by the proxy class and the delegate class implementing the same interface. The proxy pattern has the following advantages: 1.clear responsibilities. 2.high scalability. 3.intelligent.

2. Singleton pattern

The singleton pattern has the following advantages: 1. control the use of resources, through thread synchronization to control concurrent access to resources; 2. control the number of instances generated to achieve the purpose of saving resources; 3. as a communication medium, that is, data sharing, it can be used to achieve communication between two unrelated threads or processes without establishing a direct link.

3. Factory pattern

The factory in this case refers to Spring IoC dependency injection: the factory pattern can be used to make the code structure clear and effectively encapsulate changes; to simplify the front and back-end calls to API calls; and to reduce the coupling to a single encapsulated code set.

4. Decorator pattern

The decorator pattern allows us to add and improve the functionality of the previous MVP at a later stage, for example, the ticketing system and the "Guess what" column are developed independently without coupling each other.

The design of the back-end project structure is based on the Spring MVC idea: Spring MVC is an application development framework based on the MVC architecture used to simplify web application development and is part of Spring. modules, namely Model, View and Controller. Model: used to encapsulate business logic processing (java class); View: used to present data and manipulate the interface (Servlet); Controller: used to coordinate the view and model (jsp); Processing flow: the view sends the request to the controller, which selects the corresponding model to process; the model gives the processing result to the controller, which selects the appropriate view to present the processing result.

3.2.2 Extension Pack Listing

We use Swagger, Postman, SendCloud, lombok and other code packages.

Swagger: can generate documentation directly from the code, no longer need to write the interface documentation manually, greatly improve the development efficiency, can save the time of writing documentation; can be directly run, can test the API interface online; highly extensible, can support multiple languages

Postman: has an easy-to-use GUI, which greatly improves the efficiency of use; can save the history of API requests; and has the ability to use collections, environment variables, run tests and share collections without restrictions, suitable for the back-end verification of online ticketing systems; and can use collection Runner to automate tests; has flexible API monitoring, runtime, performance and accuracy; supports split-stack development.

SendCloud: We use sendcloud's ability to send activation authentication emails for registration and password reset; in subsequent iterations of the product we can also use its strong reach to promote part of the marketing campaign and complete the commercial closure of the product.

lombok: a very useful Java tool that can be used to help developers eliminate lengthy Java code, especially for the relatively simple back-end development of ticketing systems; simplifying the lengthy pipeline code and greatly improving development efficiency.

3.2.3Network Token Listing

We use json web tokens as web tokens in the user registration and login process.

JWT has the following advantages: it is scalable, in the case of distributed applications, the same type of web token session needs to be shared across multiple machines and can usually be stored in a database or redis. jwt does not require this. Stateless jwt does not store any state on the server side. one of the principles of the RESTful API is to be stateless, and when a request is made, a response is always returned with parameters and no additional impact. The user's authentication state introduces this additional impact, which undermines this principle. In addition jwt's load can store some common information for exchanging information, and using JWT effectively, can reduce the number of times the server queries the database[7].

3.3.3 Database enumeration

3.3.3.1 Main Database

MySQL is a relational database, mainly used for storing persistent data, storing data in the hard disk, which is slow to read. MySQL as a persistent storage relational database, the relative weakness is that each request to access the database, there is an I/O operation, if repeatedly and frequently access the database. First: it will take a lot of time to link the database repeatedly, which will lead to slow operation efficiency; second: repeated access to the database will also lead to high load on the database, so it is necessary to use caching to avoid the risk[8].

3.3.3.2 Caching database

The cache is a buffer for data exchange (cache). When the browser executes a request, it first looks in the cache, and if it exists, it gets it; otherwise, it accesses the database, and the cache has the feature of reading data faster. The Redis database is a cached database, used to store frequently used data, thus reducing the number of database accesses and improving operational efficiency.

4.Challenges in the development process

4.1 Front-end challenges

4.1.1 Slow page loading

During the development process, the problem of slow loading of the main page emerged, which could result in a long blank loading page, greatly affecting the user experience.

Solution: We optimized the logic of the front and back-end; the front-end added a "loading" page, but adding a new page would inevitably cause the browser to refresh, so in order to avoid affecting the

user experience here, we finally decided to use seNavigate as hooks to do the jumping between pages; the back-end optimized the SQL statements for registration and login to solve the slow Query.

4.1.2 Confusing code format

The front-end IDE vscode was buggy and the code optimisation plugin Prettier was installed but did not work, making the code format messy and affecting the development experience.

4.1.3 Data management

The design of the front and back-end interfaces was not reasonable, resulting in some interfaces not using the property of self-adding primary key in the database table, which increased the number of parameters and the complexity of database queries.

Solution: Optimize the interface design and rewrite the specific logic statements to make more use of the self-incrementing primary key.

4.1.4 Other Challenges & Solutions

4.1.4.1 Compatibility issues

The project has many dependencies, and the versions of each dependency may be incompatible, for example, the front-end react and material ui versions are incompatible, and the back-end java version and spring version are incompatible, resulting in the front-end and back-end not working together and spending a lot of time on configuring the environment.

Solution: Check the compatible versions of the respective dependencies and reinstall and reconfigure them.

4.1.4.2 Reusability issues

The initial development of the front-end did not determine the page layout and overall visual effect, resulting in some components such as the pop-up <module> not being designed in a reusable form, leading to subsequent repetitive development and inelegant code, as well as not ensuring the integrity of the component design. Solution: Use a reusable component, in this case useForm.jsx, to build a series of scenarios that involve filling out forms such as registering and logging in, adding activities and forgetting passwords[9].

4.2 Back-end challenges

4.2.1 Recommendation system

The recommendation system is divided into a recall and a sorting phase: recall sorts through the types of events that have been consumed to find those that match the requirements. Sorting returns the results after calculating the similarity between the consumption records and the individual recalled events.

4.2.2 Difficulties and solutions in database design

4.2.2.1 Database

The data available, especially the consumption records, were too small and many of the time descriptions were empty, which affected the accuracy of the recommendations.

Solution: Replace the character set with utf8mb4_unicode_cs, which is case-sensitive.

4.2.2.2 Forms

Constructing a complete system requires the use of many tables, and the links between each table (primary key, foreign key) may be complex, which may lead to inconsiderate table design, missing attributes or problems with redundant and useless attributes being left behind.

Solution: List the attributes that each table may have in excel and discuss and amend the structure of the table within the group.

4.2.3 Other challenge points encountered & solutions

4.2.3.1 Network Security

Both mysql and redis were deployed in the cloud, with the intention of ensuring data consistency, but server security issues led to hacking, loss of database data and system corruption.

Solution: Periodically perform backup/snapshot operations. The password generator is used to automatically generate complex passwords. The server sets 12-16 complex passwords consisting of uppercase, lowercase, special characters and digits. The super administrator is forbidden to log in and change the default port number of the remote login service. The security group function of the cloud host allows only service protocols and ports, but not all protocols and ports.

4.2.3.2 Global proxy

The back-end was not well built with a dynamic proxy-based back-end task to monitor, capture and handle error reporting from back-end services, resulting in many native, unprocessed and difficult to understand exceptions and error reporting being passed to the front-end, adding great difficulty to front-end development and debug.

Solution:Use the Spring Security module provided by the Spring development framework to build a global dynamic proxy task to catch and handle exceptions and error reporting.

4.2.3.3 Front and back-end collaboration

Separate front and back-end development, information exchange is not timely and not comprehensive, easy to cause interface design does not meet each other's requirements or new development interface frequent error problems, the need to introduce a better way to exchange information.

Solution: The back-end integrates swagger to provide the front-end with a visual interface for interface definition; the back-end uses postman to self-test the newly developed interfaces, and then delivers them to the front-end after passing.

4.2.3.4 Long query time

Due to the large number of database tables and the strong linkage between the tables, it is easy to have the problem of slow SQL queries caused by multi-table join queries, making the front-end request time long and user experience poor.

Solution: Optimize the sql query statement, replace the "where" keyword with "join" to improve the efficiency of sql statement execution; use redis to store hotspot data such as personal information of logged-in users, which can intercept part of the requests to the database causing congestion, but directly accessing redis to get the front-end data it wants at a particularly fast speed.

5.User documentation

5.1 Overview

Our website is an e-commerce platform for buying and selling performance tickets. The website can be accessed by accessing the vlab configuration environment, or directly through a shared local link from the cloud server.

5.2 Frontend

Prerequisite:Below list should be installed on local environment：

1. Node.js
2. npm
3. yarn
4. JDK 11
5. end setup
   a. Change to the root directory of project, which should show like below figure:

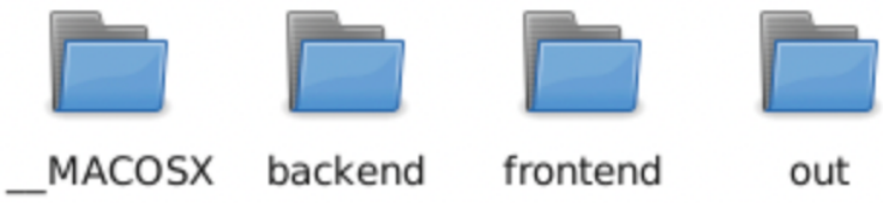

figure 10:root directory

b. Move into the frontend directory with:**Command:cd frontend**

c. In the frontend directory, install the required dependencies with:**Command:npm install** or

yarn install

It may take several minutes to complete and present information likes below figure:

```
wagner % npm install

added 2241 packages, and audited 2308 packages in 2m

70 vulnerabilities (14 low, 19 moderate, 33 high, 4 critical)

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

figure 11:present information

Note that if there is any unexpected error occur, which makes you cannot install some dependencies successfully,then you can also download the pre-installed dependencies called “node_modules” referring to the link:

https://drive.google.com/file/d/1Igd-qxeP9UnCZpY44c-NrYKnFPp5CN8r/view?usp=sharing ,

and unzip and extract the content in the current directory, then you can move the next step.

d. In the current directory, run the frontend project with:**Command: npm start** or **yarn install**

Then you will see the information likes below figure if everything goes well.

```
  Line 58:    'ifDisplay' is assigned a value but never used         no-unused-vars
  Line 70:    'eventType' is assigned a value but never used         no-unused-vars
  Line 93:    'location' is assigned a value but never used          no-unused-vars
  Line 116:   'ticketType' is assigned a value but never used        no-unused-vars
  Line 277:   'handleSubmit' is assigned a value but never used      no-unused-vars
  Line 285:   'uploadIamge' is assigned a value but never used       no-unused-vars
  Line 287:  'base64' is assigned a value but never used           no-unused-vars
  Line 319:   'handleTypeChange' is assigned a value but never used  no-unused-vars

./src/pages/loginRegister/ForgotPassword.jsx
  Line 4:  'message' is defined but never used  no-unused-vars
  Line 5:  'request' is defined but never used  no-unused-vars

./src/pages/eventList/EventList.jsx
  Line 5:  'Modal' is defined but never used         no-unused-vars
  Line 57:  'data' is assigned a value but never used  no-unused-vars

./src/pages/payment/OrderPage.jsx
  Line 6:    'StarOutlined' is defined but never used               no-unused-vars
  Line 44:   'setSearch' is assigned a value but never used         no-unused-vars
  Line 246:  Expected to return a value at the end of arrow function  array-callback-return

./src/components/ProfileForm/index.jsx
  Line 6:   'Typography' is defined but never used          no-unused-vars
  Line 77:  'userDetail' is assigned a value but never used  no-unused-vars

./src/pages/EventOrder/EventOrder.jsx
  Line 7:  'setSearch' is assigned a value but never used  no-unused-vars

./src/components/AppBar/index.jsx
  Line 91:  'naviToEventEdit' is assigned a value but never used  no-unused-vars

Search for the keywords to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.
```

figure 12: frontend information

e. Turn on a browser and visit our website via:

http://localhost:3000/

Now the front-end project is running on the port 3000.

5.3 Back-end setup

f. Change to the root directory of project.

g. We have already packed the whole backend project into an executable file called "backend.jar", which you can move into its directory with:

Command: cd out/artifacts/backend_jar

h. To start to backend service via:

Command: java -jar backend.jar

Then you should see information likes below figure:

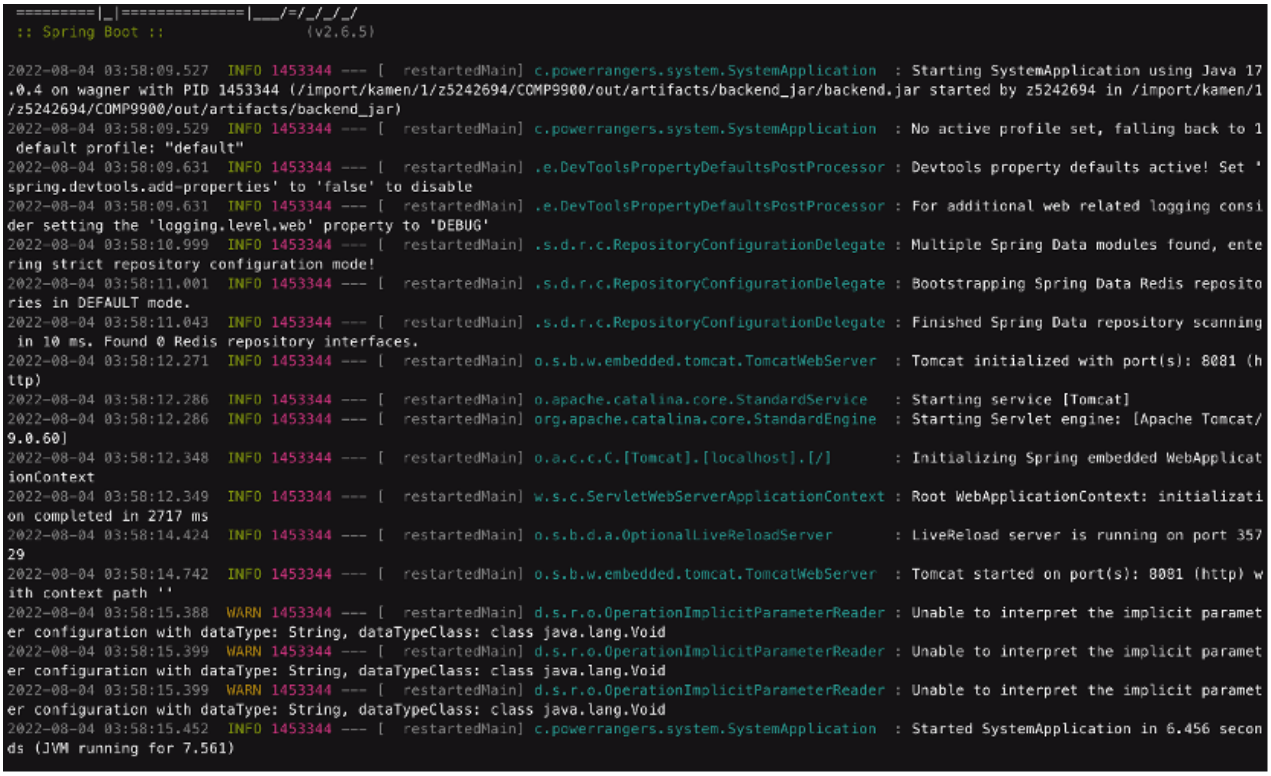

figure 13:backend information

Now the back-end service is running on the port 8081.

i. If everything goes well till now, the project has completely deployed and worked, enjoy!

5.4 Browser Access

If you want to see the interfaces information of this project, you can open a browser and visit: http://localhost:8081/swagger-ui/index.html

you can also access our website demo through: http://1.15.115.148:3000

5.5 Function Manual

5.5.1 Login and Registration

Before starting to access the site, the user needs to create an account. Clicking on the "Register" button will take the user through the process of registering for an account.

Users will need to fill in their personal information in the account registration form and verify their email address. Once registered, the user will be automatically logged into the site.

If the user forgets their password, they can reset their password by entering their email verification code. Go to the password recovery page by clicking on "Forgot Password", click on the verification code in the form, and then check your inbox to submit your new password to save it.

If you already have an account, you can log in through the login page, enter your personal information and password in this form, and select your identity to log in to the site.

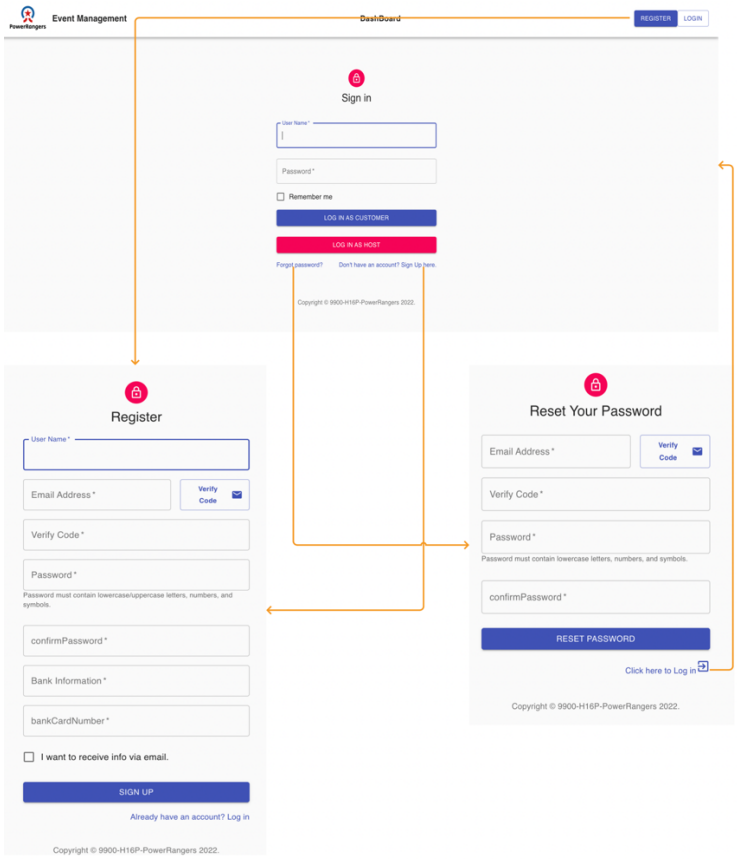

figure 14: Login and Registration

5.5.2 host user creation event

Once the user has chosen to log in to the site using the host identity, a create button will appear in the top navigation bar of the site after a successful login. The user clicks the Create button to enter the process of creating an event. The user fills in the form with the information about the event they want to create and submits the form, the event will be created successfully and will be displayed on the home page of the website for other users to view and purchase.

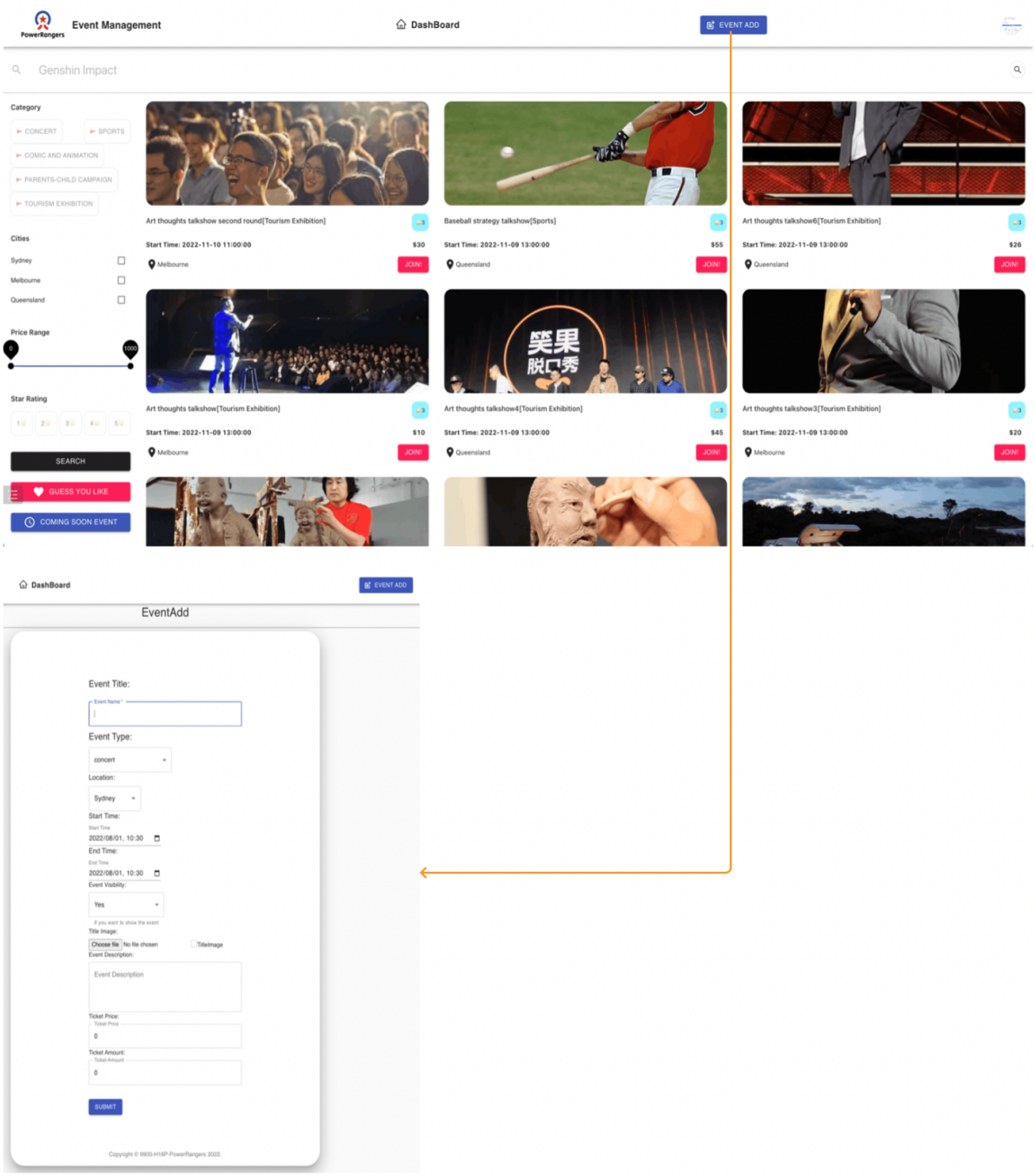

figure 15: host user creation event

5.5.3 Purchase and recharge

Users can click the join button on the event details page to make a purchase, and the user's account balance will be automatically deducted when the purchase is made. If the user's balance is insufficient, the user can top up the account by using a bank card or scanning a QR code.

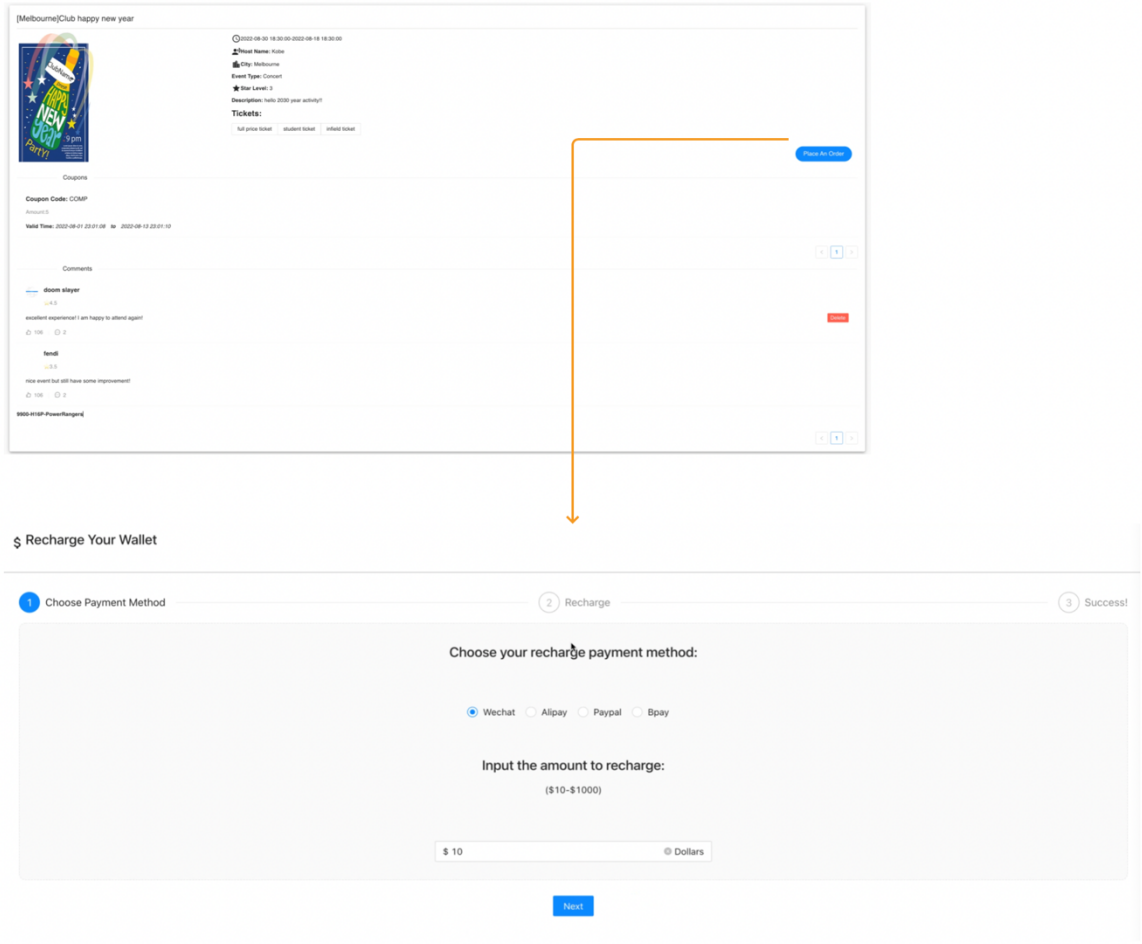

Figure 16: Purchase and recharge

6.References

[1] Banks, A. and Porcello, E., 2017. Learning React: functional web development with React and Redux. " O'Reilly Media, Inc.".

[2] Elrom, E., 2021. React Router and Material-UI. In React and Libraries (pp. 79-113). Apress, Berkeley, CA.

[3] Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C. and Deleuze, S., 2013. Spring boot reference guide. Part IV. Spring Boot features, 24.

[4] Pautasso, C., 2014. RESTful web services: principles, patterns, emerging technologies. In Web Services Foundations (pp. 31-51). Springer, New York, NY.

[5] Da Silva, M.D. and Tavares, H.L., 2015. Redis Essentials. Packt Publishing Ltd.

[6] Lee, M.H., 2019. Design and implementation of hybrid apps design based on spring MVC. Journal of the Korea Convergence Society, 10(3), pp.395-400.

[7] Widenius, M., Axmark, D. and Arno, K., 2002. MySQL reference manual: documentation from the source. " O'Reilly Media, Inc.".

[8] Di Sarno, C., Garofalo, A., Matteucci, I. and Vallini, M., 2016. A novel security information and event management system for enhancing cyber security in a hydroelectric dam. International Journal of Critical Infrastructure Protection, 13, pp.39-51.

[9] Kuncara, T., Putra, A.S., Aisyah, N. and Valentino, V.H., 2021. Effectiveness of the E-Ticket System Using QR Codes For Smart Transportation Systems. International Journal of Science, Technology & Management, 2(3), pp.900-907.