CPT202 2023/2024 Semester 2

Final Report for Software Engineering Group Project

<Project Title> **Online Booking Service for a Sport Centre**
<Date> **2024.5.23**

Group number: **Group 7**
Student Name: **Bangxu Tian**
Student ID: **2144648**

Contents

I Introduction

Currently, sports serve as a healthy way to relieve fatigue in university life, strengthen the body, and balance study and entertainment for both teachers and students. To address the inefficiency of the current sports center, our group designed a web-based Sports Center Online Booking System. This system allows staff and students to book venues conveniently, and it includes an administrator mode to manage sports activities, view ticket orders, and monitor the venue's income, ensuring efficient operation.

As shown in Fig.1 and Fig.12 in Appendix A, our group adopted MVC (Model-View-Controller) software development process, using JAVA and Maven to build a web-based app. The system architecture consists of three parts: front-end, back-end, and database. The front end uses CSS, HTML, and JavaScript for user-friendly pages and interactive features. The back-end, built with the SpringBoot framework, handles data fetching and database interactions via RESTful APIs. We chose MySQL for reliable data storage and efficient queries, with database design ensuring user information security and data persistence through JPA(Java Persistence API). We also finished unit and integrated testing with Junit and some integrated tests. Finally, we deployed the system to Aliyun Server.

I played an essential role in this group, mainly responsible for the related modules of the administrator, covering the contents of front-end, back-end, database and Junit tests. Specifically, the functions that I am responsible for include **generating daily reports** (including sports, sales, and ticket selling), modifying the user's information (such as whether they are an administrator), **viewing the user's booking status**, **viewing the user's personal information** (including personally identifiable information and body information), and **selecting coupons** in the booking step. These functions involve efficient interaction between the front-end and the back-end, as well as storage and query of the database. During the development process, my team members and I worked closely to ensure the interfaces' consistency to realize the full functions of the system.

The report will be divided into a couple of sections. First, the Software Development Process section will discuss how Scrum facilitates the software development process and effective communication. The Software Design section then describes the design and implementation of all the PBIs that I developed individually. Third, the Change Management section will consider how to handle requirements changes in a Scrum project. The section on legal, social, ethical, and professional issues will be considered in the project and possible solutions to these issues will be discussed. Next, the conclusion part will summarize what I learned from the project and suggest improvements for future projects. Finally, References and Appendix will add some evidence and images to support the contents of the entire report.

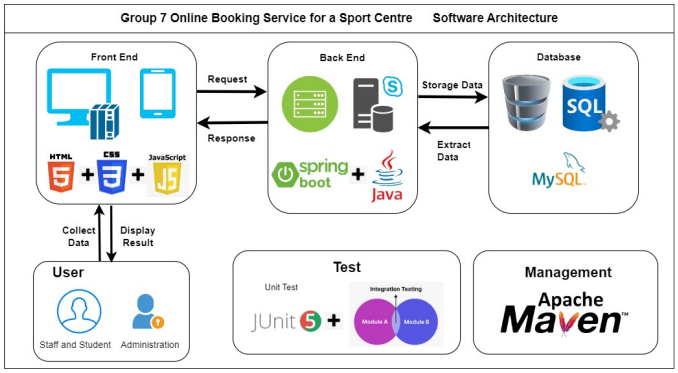


Fig. 1: Project Technology Architecture

II Software Development Process

Scrum is an agile development method that addresses the challenge that traditional software cannot respond quickly to changing requirements. As an iterative and incremental agile framework, in Scrum, each member has its own place and each organization member works independently and proactively to form a highly productive project team [1], [2]. For our group, the Scrum method is very suitable for developing this online booking system for the university sports center. It promotes the **software development process** and **effective communication**, which is mainly reflected in the following aspects.

A. Product Backlog Documentation

Product backlog(PBI) lists the work items that need to be coordinated and completed by the entire team [3]. It prioritizes the work requirements, user stories, acceptance criteria, and other miscellaneous items that must be completed [4]. It is an excellent guide to product development. During the group's development, we rigorously discussed and elicited PBI for this system early in development. First, we set them as the highest priority PBIs based on the basic requirements in the requirement file. For example, for users' functions, registering users, viewing and updating personal information, and booking tickets are the most essential to-do items. For administrators, viewing all booking information, viewing and modifying sports activities, and viewing personal information are the most urgent items to be involved and developed.

Our group has three judged criteria: urgency and importance of user needs, complexity and risk of technical implementation, and business value. For example, the user login and registration function is the primary function of the system, it is urgent and vital, so we give it a high priority; the User booking function is the most core part of the entire system, and has a very high business value: the priority is very high. At the same time, in the **Sprint Review** of each Sprint iteration, our Product Master and team members actively discussed the changes in the requirements listed above and made constant updates and adjustments.

As shown in Fig.13 in Appendix B, each PBI has been assigned different values according to Priority, Effort and

Business Value, specified Deadlines, Status and other information, and written user stories, acceptance criteria and other details according to user requirements. Overall, PBI helps us systematically manage and track all system requirements and functions, and its prioritization and details ensure that our team's work is always aligned with the needs of our users.

B. Precise Assignment of Roles

1) Product Owner

In a Scrum-framework-based team with 7-9 people, there are typically 2 leaders and 5-7 developers [5]. The Product Owner is responsible for the content of priorities of the PBI, ensuring that the team is always developing the features that are most valuable to users and the business. In the early stage of the development of the group, our product owner collected and clarified user requirements from stakeholders (in this case, Teachers and TAs), wrote user stories, created acceptance criteria together with other team members, and added them to the PBI. Acting as a bridge between the development team and the Teacher, he introduced the acceptance test standards in the online booking system for the developers **in detail**. For example, gather student and faculty feedback on the reservation system and write user stories about requirements such as "Register Users", "View and Update Personal Information" and "Generate Daily Report".

2) Scrum Master

Scrum Master ensures the correct implementation of the Scrum framework. In our group, it is likely to be a leader role for **all developers**, which is held by me. I developed a strict Sprint Iteration process to continuously track the progress, and timely seek and solve any problems that hinder the progress. For instance, when designing the **booking module**, there was a problem that the modules of two sports (Basketball and Football) could not be connected. I organized an additional technical discussion, inviting the developers of the two sports to explain their respective implementation methods and interface specifications in detail. This meeting revealed the inconsistency of API requirements for the two sports, and it led to a consensus on modifying the data structure and API design, ultimately resolving the issue.

3) Development Team Member

The development team includes front-end design, back-end development and database management. Traditionally, each member is responsible for tasks in their area of expertise, collaborating in daily stand-ups and Sprint planning meetings [6]. However, in this project, according to the requirements of **Coursework Handbook**, to enable everyone to participate, we decided to develop in modules. Each member was responsible for **all the development aspects** of a certain module. For example, under the unified requirements, the Personal Center section, each team member separately develops identity information, fitness information, coupons, password protection pages, and booking history with front end, back end, and database content. Reasonable PBI compliance also ensures the integrity and consistency of each module.

C. An Iterative Sprint Process

In this project, we go through three iterative Sprint processes, as shown in Fig.2 and Fig.14 in Appendix C. Each Sprint lasts two and a half weeks, including **Sprint Planning**, **Sprint Execution**, **Sprint Review**, and **Sprint Retrospective**. The specific project Gantt diagram is shown in Fig.18. It should be noted that in the Sprint process of our group, we emphasized **Daily Stand-Up**. At weekly meetings, each team member reports on the week's progress, plans for the coming week, and obstacles. This approach ensures that information is synchronized among team members, and issues can be identified and resolved promptly, avoiding schedule delays. Specifically, in the Daily Stand-up of Execution of Sprint2, we found that the "user personal center" PBI is too large, which may not be fully completed in Sprint2. After a brief negotiation, we made decision to break up this part of the PBI, delivered it to the developers, and finally completed it within the specified time frame.

In addition, we use several communication and collaboration tools to support our development process. For example, instant communication through the WeChat Group is not limited to formality, and anyone can raise any questions anytime. These tools improve the team's communication efficiency, ensure all information is transparent and traceable, and help us maintain efficient collaboration and progress through each iteration.

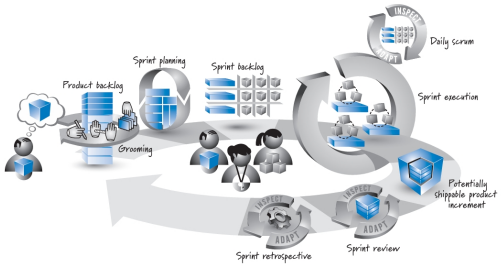


Fig. 2: Sprint Iteration Process

III Software Design

In this project, 40 PBIs were identified and each person was responsible for the entire design and implementation process for 5 of them. The implementation of each PBI has gone through detailed requirements analysis and technical design. It has been documented in the development process, using UML diagrams to support and explain.

A. Generate Daily Report

As an administrator of an online stadium booking system, it is critical to examine operational data and key business metrics. Therefore, an important PBI in this project is to generate a daily report on the stadium, which includes detailed

data on several sports, their sales, ticket numbers, etc. Through detailed data, administrators can understand users' booking preferences and continuously optimize the stadium's ticketing structure.

Specifically, for instance, during our pilot test, through the reports generated by the system, I found that people are more interested in fitness. Check regularly and get more tickets for fitness than for any other sport. As a wise decision maker, he/she may want to consider adding some other fitness related activities in the future (see PBI for details: Manage court number of sports activity and PBI: Manage limit of participants which are both delivered by my group-member **Ming.Wang21**), or adjust the total number of tickets and opening hours of fitness sports.

1) Logical Flow

When the administrator clicks the "Download Daily Report" button in the front-end interface and requests to get the daily report generated in the back-end, this operation will be sent to the back-end controller through the front-end **AXIOS** and **JavaScript** files. When the Controller receives the request, the system processes and invokes the methods: A series of Controllers will start generating the report process. As different level shown in Fig.3, first, the **ReportService** class extracts the relevant data from calling the methods of Controller **ProductDao** and **OrderDao** in the **data access layer**, using **MySQL** statements to filter all the order information for today from the database **in database layer**. After obtaining the relevant information in the database, the Controller in the **data service layer** will extract the data to generate an Excel table, which includes the status of all orders and details. The Sequence diagram for an administrator to download daily report is shown in Fig.15 in Appendix D .

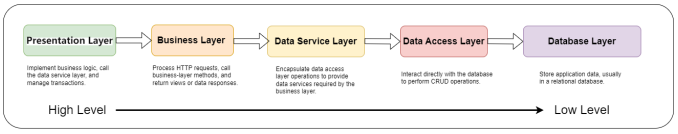


Fig. 3: Different Controller Layers

In this project, information such as the unique identity order number (id), date, sport type, quantity, venue, and user name for each ticket is recorded in detail in the database. However, as a Daily Report for revenue and maintenance purposes, all tickets of the same type are grouped together in the design process. This Excel table shows the sales, order quantity, and sport type for each sport.

2) Database

The design and implementation of the database still remain core to data support in the generation of the daily reports. Among the tables of numerous databases—see the details of the database contained in the Appendix F and a detailed E-R figure—this PBI is mainly related to two tables: "*tb_product*" and " tb_order". The "*tb_product*" table stores information of all sports including all sports and their venues, ticket prices, and the status. The "*tb_order*" table records all the order information, including order ID, discount ID, payment method, order creation time, use time, number of venues, order price, sport type, user name, and number of reservations. This report calculates the day sales with numbers of reservations and filters order data on that day according to '*create_time*' and '*use_date*' fields.

To design the database table, special attention was given to the principles of **database normalization** and **data integrity** [7], [8]. Each field has a clear definition with a clearly stated purpose. For instance, the primary key is the 'id' field. "*bigint*" is the type of field, and it has the self-increment property for each record to make sure that every record is unique. That is, in the scope of the problem: The 'product' field holds a product name, and its type is '*varchar(255)*', being not null, unique to prevent redundant data.

1) Database Normalization: Normalization avoids redundant data and keeps the integrity of data. For example, "*tb_order*" is a table, 'id' is a unique key representing each order, and '*discount_id*' is a foreign key related to information on the discount. This ensures that data items are held where they most logically should be, making the database more efficient and easier to administer.
2) Data Integrity: It was deemed consistent in that it preserved referential integrity despite its partial enforcement. For example, an addition of a foreign key constraint to the id in "*tb_product*" referring to the latter into the former, "*tb_order*," would ensure the existence of the product ID in the order within "*tb_product*." This will guarantee the absence of stand-alone or invalid order records and, in so doing, assure the reliability of data.

The design principles within the database make the system capable of supporting the creation of a proper and reliable daily report. They are salient in maintaining the integrity and consistency of data that is required for the system to work.

In addition, I designed some additional data constraints and checking mechanisms. As shown in Fig.4, for example, the design of the available field can be quickly distinguished between currently available and unavailable products and can only be set to 0 or 1, indicating the available state of the status; The index design of '*use_date*' field not only improves the efficiency of filtering order data by date but also verifies the validity of order according to time. In addition, the field with a standard name and clear table structure is also conducive to the maintenance and expansion of the system.

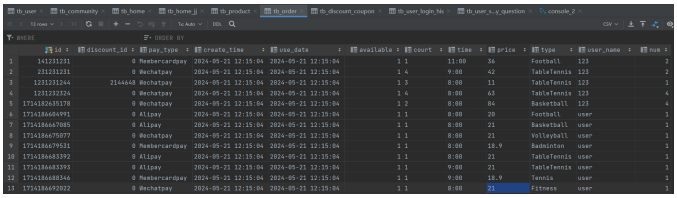


Fig. 4: *tb_product* Screenshot

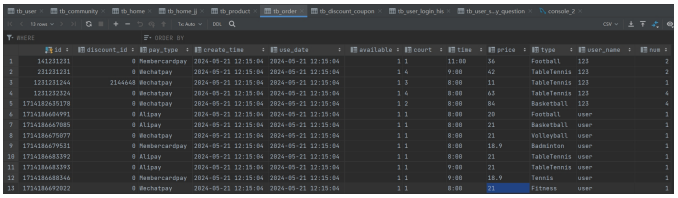


Fig. 5: *tb_order* Screenshot

3) States of the Objects

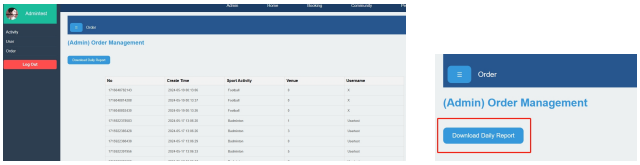
The main objects involved in generating daily reports are **Products** and **Orders**. Their status and changes have a direct impact on the generation of daily reports. In a system, not only the state at the database level is critical but also the state changes at the back-end level and the front-end GUI level.

While managing all sports, each sport has a status field that represents its current state. This status directly affects whether the ticket corresponding to the sport can be booked by the user typically and whether it will appear in the daily operation report. For example, when the product status is ”available”, the user can order the product , and the order will be recorded and added to the sales. Moreover, the daily report will count and display the information about these tickets. Conversely, if the product status is ”unavailable,” the user cannot book the product and its usage will not be shown in the report. This state change is not only reflected at the database level, through manual action on the administrator side, but also timely updates on the user GUI interface to prevent users from trying to reserve products that are no longer available.

4) User Interface

In the PBI that generates daily reports, the user interface (UI) is designed to make it easy for administrators. With its intuitive and **user-friendly** UI design, administrators can quickly generate and download daily operational reports, thereby improving the system’s usability and user experience.

As shown in the red box in Fig.6, the administrator can visually see some information about the order through the GUI interface. Moreover, the user interface includes a ”Download Daily Report” button, which is in a prominent position and easy to find and click. Users click the button, the system will call the back-end to generate a daily operational report and provide a download link. The design is straightforward, and users do not need to go through the complicated operation process to get the required report with just one click. Fig.16 in Appendix D is the sample grid of daily report in Excel.



(a) (b)

Fig. 6: Order Interface of Administrator

Furthermore, the style of the entire interface remains consistent, and the color matching and layout design follow a unified visual specification to make the user feel comfortable and smooth during the operation. In this project, I decided to use orange as the primary color for the common user interface and blue as the main color for the administrator interface to have distinguishes between them. Also, both navigation layout maintains the same: Fig.7 and 8 depict the same navigation structure and different color patterns for the two different user types. Administrators will see an additional ”Admin” button than ordinary users in the upper navigation bar. This consistent and aesthetically pleasing design improves user satisfaction and the professional image of the system.

Fig. 7: Orange Navigation for Users

Fig. 8: Blue Navigation for Admins

B. Select Coupon

In this project, the ”Select Coupon” is an important experience-enhancing PBI for users when booking stadium tickets. I have added a coupon selection step to the user’s ticket booking process. When the user makes a reservation on the front-end interface, the system extracts a list of available coupons from the back-end database. It presents them to the user on the interface. The user can select one of the coupons, and the selection will be recorded and applied to the total amount of the order when the order is submitted, and the corresponding discount will be made. In the design process. I analyzed the user needs in detail and realized a user-friendly and efficient coupon selection interface.

1) Logic Flow

After the user has selected the number of tickets, click ”Next”. The ”**Select Coupon**” button will appear in the lower right corner of the interface to select coupons in a **pop-up mode**. If the coupon is selected, the front end records the coupon information selected by the user and sends this information to the back end when the user submits the order. The order submission request is processed by the **Order-Controller**. During this process, the **OrderService** verifies the validity of the coupons, including checking whether the coupon is expired and if applicable to the current order. If the coupon is valid, the system will call to get the type of coupon (for example, a fee reduction coupon or a discount coupon), calculate the amount after the discount, apply it to the total amount of the order, and send it back to the front end for the next payment operation. Appendix E Fig. 17a and Fig.17b elaborates the sequence diagram and state machine diagram

of this process respectively, to demonstrate the logical flow visually.

2) Database

The database design also plays a crucial role in supporting the "select coupons" feature. In this "Select Coupons" PBI, the relevant database table is "*tb_discount_coupon*", based on the previous database design concept, In addition, I used the concepts **index optimization** and **data range constraint** to optimize the database design to deliver this PBI.

1) Index Optimization: Fig.9 is a screenshot of "*tb_discount_coupon*". Index fields can improve query efficiency, especially when filtering and selecting valid coupons. For example, federated indexes (start_time, end_time) can significantly improve the performance of querying coupons within the validity period. In addition, checking constraints ensures data accuracy and consistency.

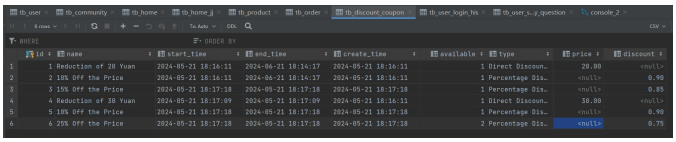


Fig. 9: Discount Coupon Database Screenshot

2) Data Range Constraint: When the user selects a coupon, the front end will request via **AXIOS** to retrieve the available coupon from the back end. Query the "*tb_discount_coupons*" table to extract all valid coupons, ensuring that the users can only select available coupons with validation. After choosing the coupon, and then an order is submitted, the system verifies the validity and availability of the coupon again, calculates the final discount, and applies it to the order. The front end interface will update and show the fee. During the design, the type field distinguishes between direct discounts and percentage discounts and instructs the system to apply the correct discount. For example, the available fields are constrained to predefined values (1 for available, 2 for unavailable), and the discount fields are constrained to a range of 0.01 to 1 with a maximum of two decimal places, representing the degree of discount. This makes it easy to do so to prevent invalid data insertion.

Through this comprehensive database design, the *"tb_discount_coupon"* table can effectively support the realization of the "select coupons" function, ensuring that the system can accurately and efficiently process coupon data, and provide users with a good use experience.

3) States of the Objects

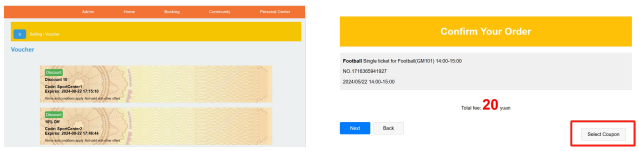
In the implementation of the "select coupons" function, the main objects involved include **Discount coupons** and **Orders**. Their states and changes directly impact the functional realization and user experience of the system. Among them, the specific introduction of the "Order" part has been reflected in the previous paragraphs.

For coupons, when a coupon is issued to the user account and stored in the user database, the detailed information will contain the start time and the end time, and the back-end program will check if the current time is between the start time and the end time. If "Yes", the coupon status field is marked as "1" in the database; Otherwise, it will be marked as 2. When the relevant module interface of the personal center of the client enters, the back-end system will extract all the user coupons with the status of "1" from the database and display their value on the front-end GUI. Of course, the database is also updated regularly, and the status of coupons may become unavailable due to expiration, use, or disable by an administrator (the **'available'** field is 2). Meanwhile, the relevant fields in the database will be marked as 2. They will no longer be displayed on the front-end interface, preventing users from selecting expired coupons.

4) User Interface

In the process of this PBI, the UI design enhances the user experience. Users can check their own coupons in the Personal Center-Voucher interface (as shown in Fig.10a). This design makes it easy for users to manage and view their coupons, ensuring they can quickly find and use them when needed. The interface of these coupons details the offer information, unique code, expiration time, etc, so that users can clearly understand the conditions of use and expiration of each coupon.

As seen in Fig.10b, the user can see a "Select Coupon" button in the lower right corner of the booking screen. After clicking, the system will pop up a pop-up window to select the coupon. This design ensures that users can easily apply coupons before confirming discount orders. Users need to decide and click the "Confirm" button on the coupon they want to apply for. That is, with intuitive UI design, many steps of user action can be reduced, which generally enhances user convenience. By using pop-up Windows, skipping between pages is avoided, and user operation is smoother and more convenient.

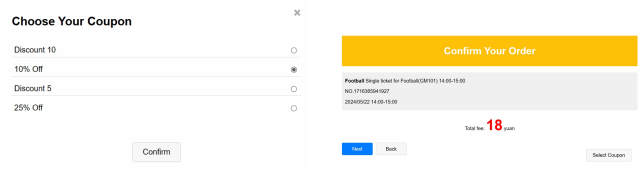


(a) (b)

Fig. 10: Coupon Interface(a) and Select Coupon Button(b)

IV Change Management

Given that this project adopts **Scrum** based on agile methodology, its flexibility is the central feature of this project. Moreover, our **Sprint Iteration** plays a significant role in changing the requirement. For example, during **Sprint Execution 1**, in Daily Stand-up, our team found an unclear point in the requirement: For the development of the coupon feature, we found that users could use multiple coupons at once. The Scrum Master (I) decided to limit users to one coupon at a time



(a) (b)

Fig. 11: Select Coupon with Pop-up Window(a) and New Cost(b)

due to the difficulty of interlacing the use of coupons and the violation of the requirements in the original PBI. This change in requirements was raised and discussed in daily meetings. Eventually, our team decided to adjust the development plan to adjust this feature. A Scrum Master coordinated the work of team members to ensure that the team could quickly adapt to change and continue to work efficiently.

In addition, in the **Sprint Retrospective 2**, we invited some users to test the features and gain feedback. In this review meeting, **Bowen.Fang21**, my group member, proposed the types of sports that could be added: The user hoped to add ”swimming” and ”table tennis” based on the six existing sports. We discussed the possibility of such an addition at the Sprint Retrospective 2, modifying the PBI and its time allocation in light of the reality, adding both sports in the next round (**Sprint Execution 3**), ensuring that the stadium offers a full range of sports.

V Legal, Social, Ethical, and Professional Issues

In our projects, legal, social, ethical, and professional issues are considered in multiple aspects of the design and implementation of the system. Here are some key problems we considered during the project's development and their solutions.

A. Legal Issues

First, concerning legal issues, our project ensures that the protection of user data complies with relevant laws and regulations, such as the General Data Protection Regulation (GDPR). However, the protection of user information in the process of user registration and information storage still needs more effort. For example, when users enter password data as a form to submit to the back-end, we should use a reasonable password encryption method (such as AES) and a reliable information transmission protocol (TLS/SSL) to ensure the security of password transmission. We have also designed a detailed privacy policy to inform users at the time of registration how their data will be used and protected. User data is anonymized and stored to a minimum in the system to ensure data privacy compliance.

B. Social and Ethical Issues

From a social and ethical perspective, our system needs to ensure that it is fair and just for all users. When designing system features, we take special care to avoid discrimination or bias. For example, when there are not enough remaining tickets for a specific time period in a particular sport, multiple users are ready to purchase these tickets simultaneously. The system needs to use fair, transparent algorithms to ensure that ticket purchases do not change based on personal status (such as whether or not it is an administrator).

C. Professional Issues

Within the group, we have emphasized confidentiality many times. In addition to submitting the work, each member of our team ensures that all information involved in the project process is only used for project development purposes and will not be disclosed to a third party. Through the above considerations, we improve the system's legitimacy, enhance user loyalty and satisfaction, and comply with relevant laws and regulations.

VI Conclusion

This report details my work as a developer of this online sports booking system. Throughout this project, I improved my knowledge and skills and gained valuable practical experience. By participating in the entire development process of the project, I gained insight into the complete software development life cycle of requirements analysis, design, implementation, testing, and deployment. In the process, I solidified my understanding of the Scrum framework and agile development methods. I learned how to effectively deal with changing requirements and ensure project flexibility and scalability.

I also gathered valuable experience in accomplishing a task with other team members. I am well-equipped to design and implement complex system functions, including coordinated front-end and back-end development, database design and optimization, user interface design, and improving user experience. Further, I learned how to consider legal, social, ethical, and professional issues in my projects to realize system compliance and user trust.

In the future, the project needs to be upgraded in both functional and non-functional aspects. In terms of functions, the system can add more human-centric design. For example, add classification and sorting methods in the process of viewing all of their coupons, so that users can view their own coupons in different order and priority; In the order interface, add the reservation of sports equipment, so that users can book not only the venue, but also the equipment, and further improve the user experience. At the non-functional level, the team will further enhance programming techniques, and experiment with more advanced deployment methods (such as implementing continuous integration/continuous deployment (CI/CD)), to improve product quality, and enhance delivery speed.

References

[1] J. Sutherland, N. Harrison, and J. Riddle, “Teams that finish early accelerate faster: A pattern language for high performing scrum teams,” in *2014 47th Hawaii International Conference on System Sciences*, 2014, pp. 4722–4728.

[2] R. Kurnia, R. Ferdiana, and S. Wibirama, "Software metrics classification for agile scrum process: A literature review," in *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2018, pp. 174–179.
[3] T. Sedano, P. Ralph, and C. Péraire, "The product backlog," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 200–211.
[4] K. Chakravarty and J. Singh, "A deep dive into product backlog prioritization technique in scrum," in *2023 OITS International Conference on Information Technology (OCIT)*, 2023, pp. 77–81.
[5] C. Unger-Windeler and K. Schneider, "Expectations on the product owner role in systems engineering - a scrum team's point of view," in *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2019, pp. 276–283.
[6] D. S. Park and J. Y. Noh, "The effect of sprint duration to the velocity in a large-scale embedded software project," in *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2023, pp. 1–5.
[7] K. Kumar and S. K. Azad, "Database normalization design pattern," in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, 2017, pp. 318–322.
[8] S. Fakhraee and F. Fotouhi, "Effective keyword search over relational databases considering keywords proximity and keywords n-grams," in *2011 22nd International Workshop on Database and Expert Systems Applications*, 2011, pp. 190–194.

VII Appendix

A. Model-View-Controller(MVC) Architecture

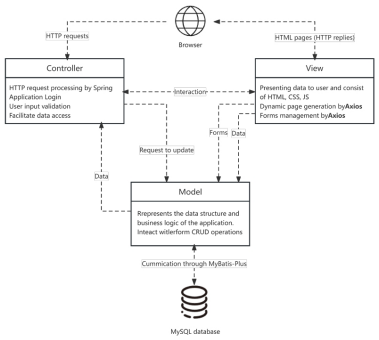


Fig. 12: MVC Architecture Diagram

B. Screenshot of Sample PBI

Product Backlog Item

PBI #	TBXB1				
Title	User Information Update				
Creation Date	3.30	**Status**	New	**Sprint**	Sprint 2
Priority	1	**Effort**	4	**Business Value**	8

Story

As a registered logged in user, I want to update my personal information so that I can keep account information up to date.

Description

1. Motivation and Background
   - This feature allows users to update their personal information and ensure the accuracy of the user profile.
   - This feature also helps database maintenance to keep users up to date.
2. Feature Overview
   - Users can update the dynamic information (body information and part of personal information), such as profile picture, IP address, height, weight, etc.
   - For things like username and email, you should display the number of changes made by the user.
   - This update feature does not include information that would require a complex

Fig. 13: Screenshot of Sample PBI

C. Sprint Process

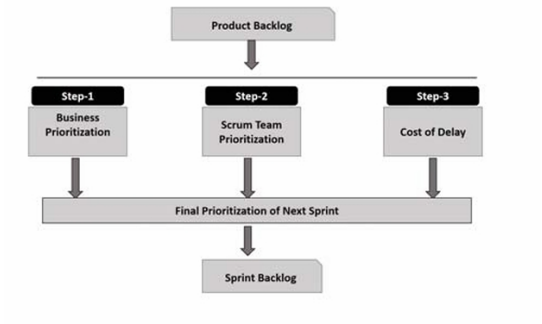


Fig. 14: Key Artifacts of Scrum [4]

D. Generating Daily Report

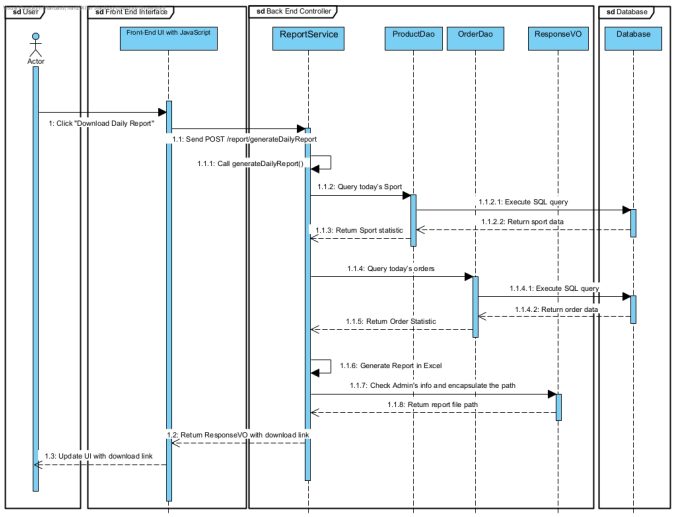
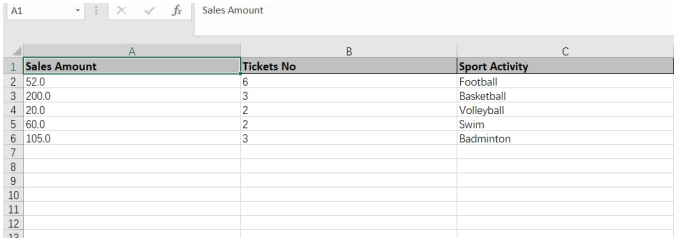


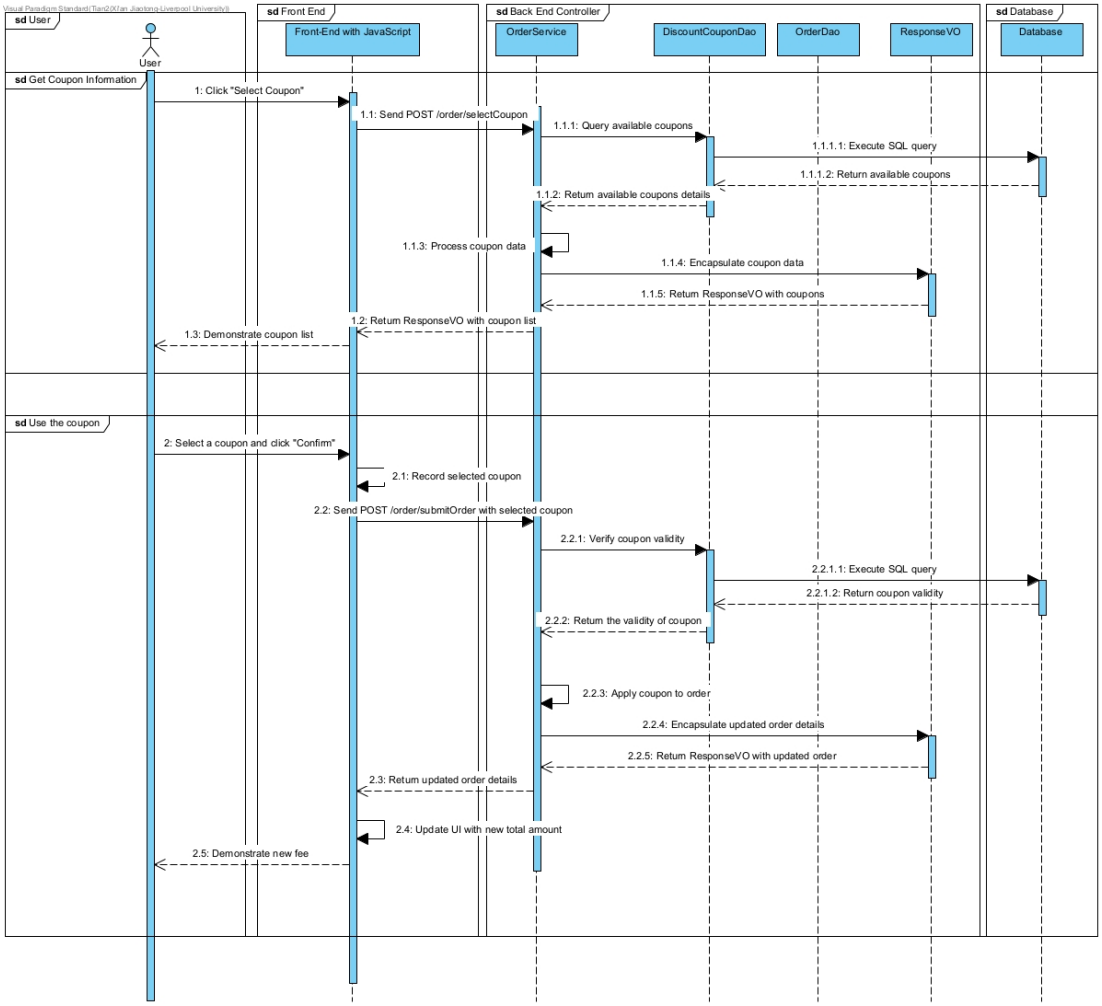
Fig. 15: Sequence Diagram of Generating Daily Report



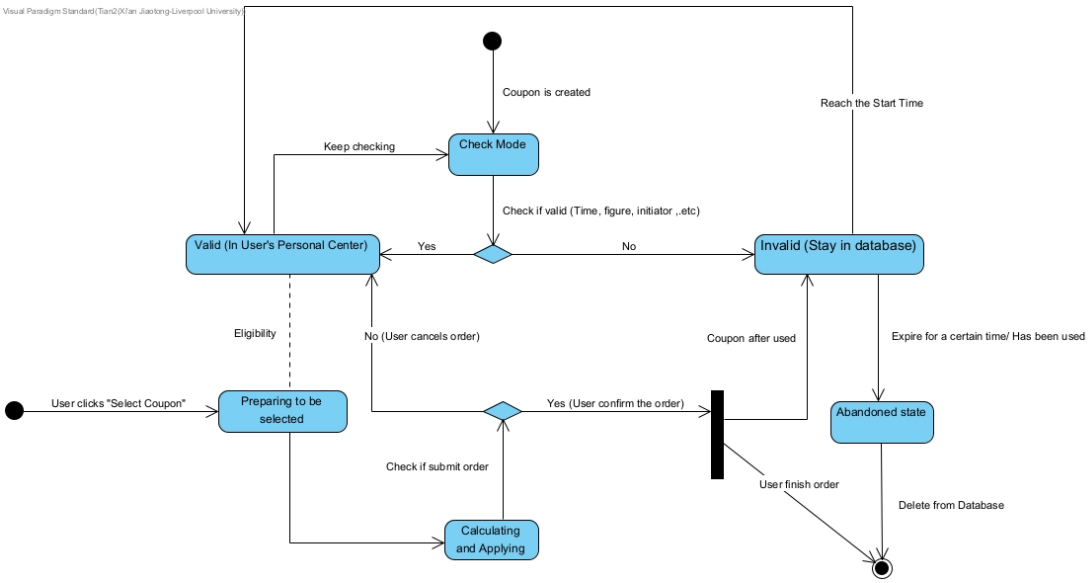
	A	B	C
1	**Sales Amount**	**Tickets No**	**Sport Activity**
2	52.0	6	Football
3	200.0	3	Basketball
4	20.0	2	Volleyball
5	60.0	2	Swim
6	105.0	3	Badminton
7			
8			
9			
10			
11			
12			

Fig. 16: Screenshot of Daily Report in Excel

E. Select Coupon



(a) Sequence Diagram of Selecting Coupon



(b) State Machine Diagram for Selecting Coupon

Fig. 17: Diagrams for Selecting Coupon

F. Sprint Gantte Chart

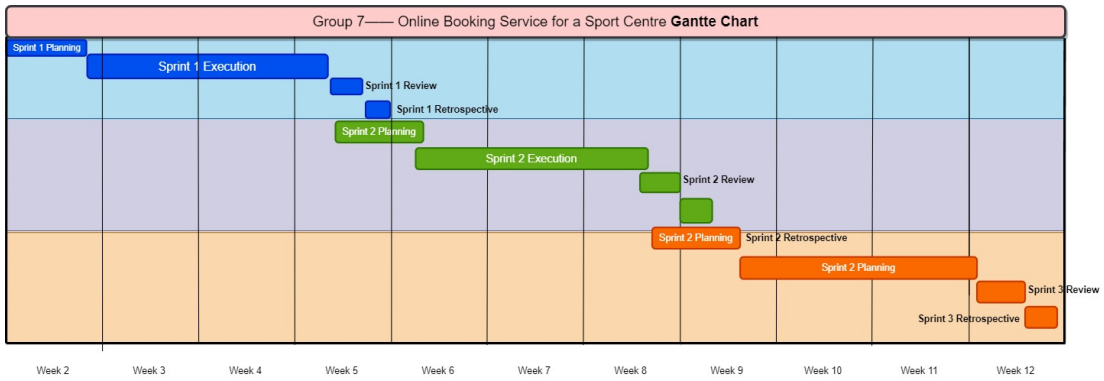


Fig. 18: Project Gantte Chart of Sprints

G. Database Entity-Relationship (E-R) Diagram

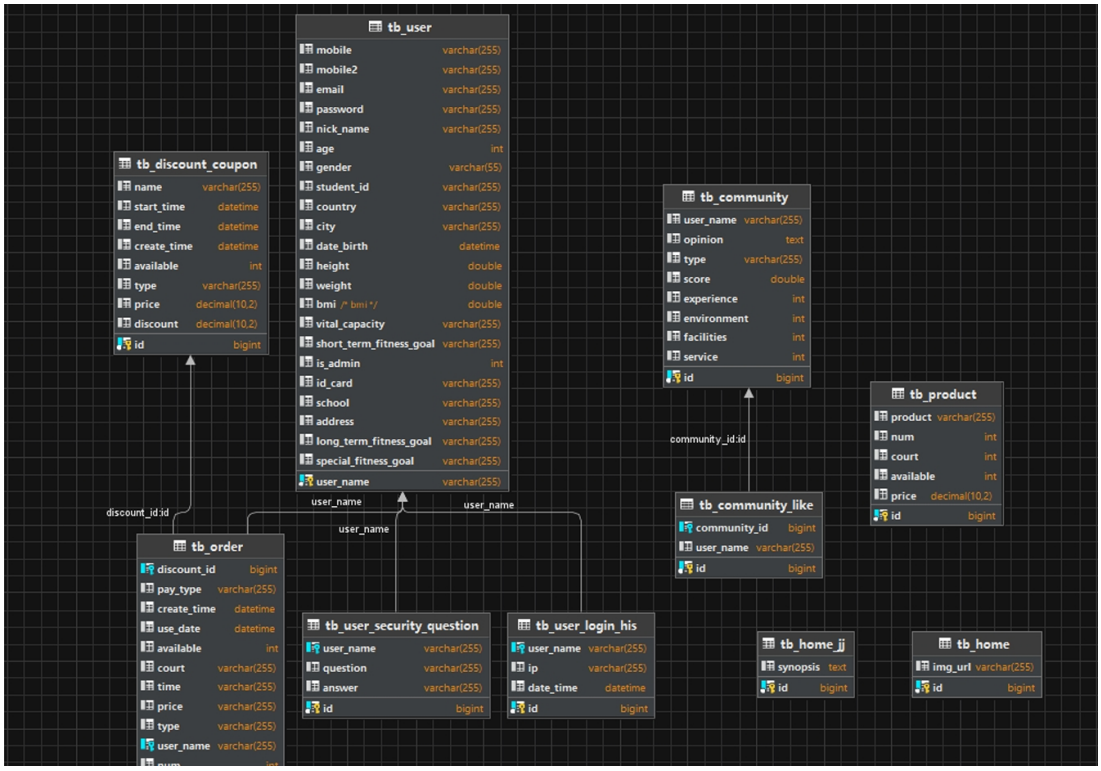


Fig. 19: Database Entity-Relationship (E-R) Diagram

H. Screenshot for Partial Code

Admin.js | body_information_modify.js | order.js

```
var body_information_saveChanges = document.getElementById("body_information_saveChanges");
var body_information_cancelChanges = document.getElementById("body_information_cancelChanges");
var Leave_any_way = document.getElementById("Leave_any_way");
var cancel_modal = document.getElementById("Cancel_button");


var cancel_span = document.getElementsByClassName("close")[0];
var cancel_span2 = document.getElementById("pop_up_cancel");


body_information_cancelChanges.onclick = function () {
    cancel_modal.style.display = "block";
}
// click <span> (x), close pop-up window
cancel_span.onclick = function () {
    cancel_modal.style.display = "none";
}
cancel_span2.onclick = function () {
    console.log('aaaaaa');
    cancel_modal.style.display = "none";
}
Leave_any_way.onclick = function () {
    history.back()
}
```

Fig. 20: JavaScript Code Screenshot

```
@RestController
@RequestMapping("discountCoupon")
public class DiscountCouponController {

    @Autowired
    private DiscountCouponDao discountCouponDao;

    /**
     * Save
     *
     * @param discountCouponEntity
     * @return
     */
    @RequestMapping("save")
    public ResponseVO save(@RequestBody DiscountCouponEntity discountCouponEntity) {
        discountCouponDao.insert(discountCouponEntity);
        return ResponseVO.success();
    }

    /**
     * Query
     *
     * @param discountCouponEntity
     * @return
     */
    @RequestMapping("find")
    public ResponseVO find(@RequestBody DiscountCouponEntity discountCouponEntity) {
        QueryWrapper<DiscountCouponEntity> queryWrapper = new QueryWrapper<>(discountCouponEntity);
        List<DiscountCouponEntity> discountCouponEntities = discountCouponDao.selectList(queryWrapper);
        return ResponseVO.success().add("list", discountCouponEntities);
    }
```

Fig. 21: Screenshot of Back End Controller

body_information_modify.js | Voucher.css | Voucher.html | order.js

```
<div class="container">
    <aside class="sidebar">
        <!-- Sidebar content -->
        <div class="logo">
            <div class="circle-avatar">
                <img src="photo.png" alt="Logo Icon">
            </div>
            <span class="logo-text"></span>
        </div>
        <nav class="menu">
            <ul>
                <li><a href="personal_information_display.html">Personal Information</a></li>
                <li><a href="body_information_display.html">Body Information</a></li>
                <li><a href="password_main_page.html">Password Zone</a></li>
                <li><a href="ticket_history.html">Ticket history</a></li>
                <li class="active"><a href="Voucher.html">Voucher</a></li>

            </ul>
            <div class="menu-item log-off">
                <a href="/login/html/logtest.html">Log Out</a>
            </div>
        </nav>

    </aside>
    <section class="content">
        <!-- Content Header -->
        <header>
            <button id="toggleSidebar">☰</button>
            <span class="header-title">Setting / Voucher</span>
```

Fig. 22: Screenshot of HTML Code

Admin.js | body_information_modify.js | Voucher.css | order.js

```
body, html {
    height: 100%;
    margin: 0;
    font-family: Arial, sans-serif;
    background: #ecf0f1; /* 你的灰色背景 */

}

.container {
    display: flex;
    height: 100%;
}

.sidebar {
    width: 250px;
    background: #2c3e50;
    color: white;
    padding: 15px;
    min-height: 100vh;
    position: fixed;
    padding-bottom: 60px;
    left:0;
    top: 0;
    overflow-y: auto;

}
```

Fig. 23: Screenshot of CSS Code

I. Screenshot of Sprint Meeting

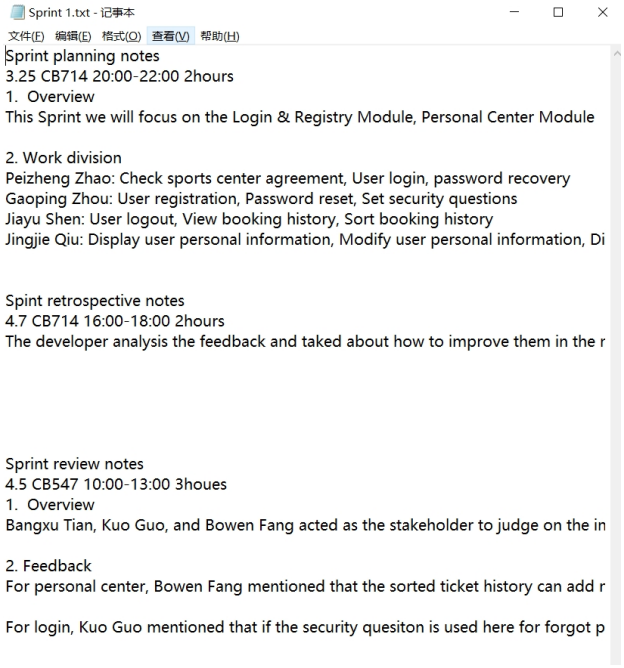


Fig. 24: Sprint Meeting Record 1

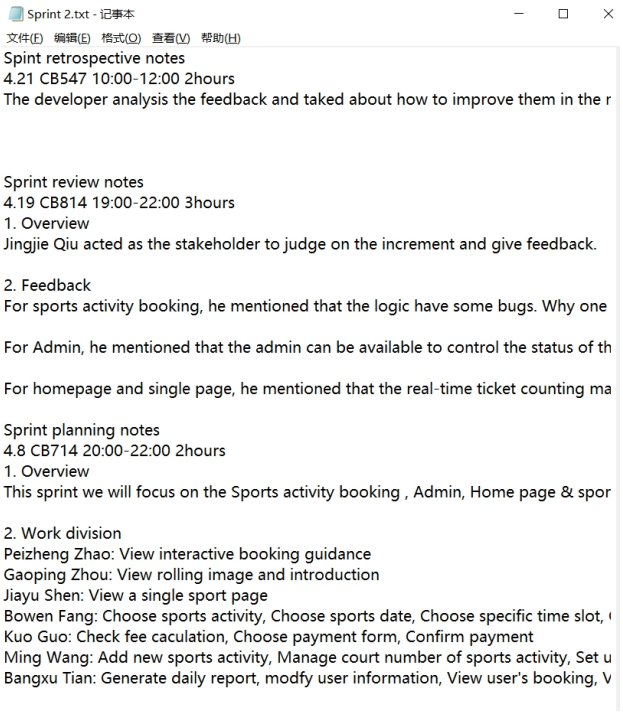


Fig. 25: Sprint Meeting Record 2

J. Screenshot of Team Cooperation

Fig. 26: First Time Team Work

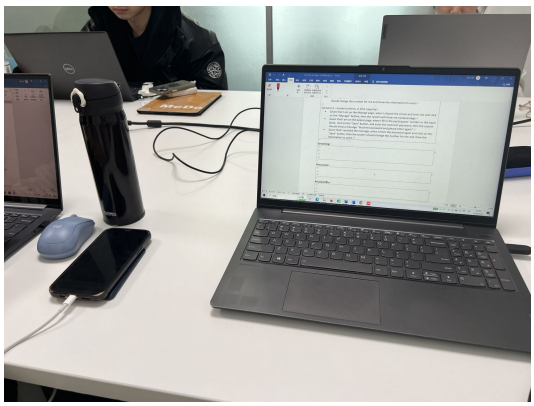


Fig. 27: Team Work in modifying PBI