

# **CPT202 Assignment 2**

## **Group Report for Software Engineering Group Project**

2023/2024 Semester 2

Project C: Online Booking Service for a Sports Centre

2024 May 12

Group number: 6

Submitted by: Mingyuan Li 2145618

Group Members:

- |    |               |         |
|----|---------------|---------|
| 1. | Jinai Ge      | 2035693 |
| 2. | Mingyuan Li   | 2145618 |
| 3. | Chengze Liu   | 2142808 |
| 4. | Yixuan Wang   | 2145543 |
| 5. | Yue Wang      | 2141845 |
| 6. | Fuyu Xing     | 2143763 |
| 7. | Baiyan Zhang  | 2142564 |
| 8. | Zhenyang Zhao | 2142074 |

## Catalogue

<b>Introduction .....</b>	<b>2</b>
<b>Problem Statement and Aims of the Project .....</b>	<b>2</b>
<b>Project Scope .....</b>	<b>2</b>
<b>User Characteristics and Assumptions and Dependencies .....</b>	<b>2</b>
<b>Project Risk .....</b>	<b>2</b>
<b>Architectural Design .....</b>	<b>3</b>
<b>System Architecture .....</b>	<b>3</b>
<b>Frontend System.....</b>	<b>4</b>
<b>Backend System.....</b>	<b>4</b>
<b>High-level Database Design .....</b>	<b>4</b>
<b>Software Design.....</b>	<b>5</b>
<b>Software Modules .....</b>	<b>5</b>
<b>High-level Process Flow .....</b>	<b>6</b>
<b>Software Support Services.....</b>	<b>6</b>
<b>Software Configuration and Production Environment .....</b>	<b>7</b>
<b>Software Testing.....</b>	<b>8</b>
<b>Unit Testing .....</b>	<b>8</b>
<b>Integration Testing .....</b>	<b>8</b>
<b>System Testing.....</b>	<b>9</b>
<b>Acceptance Testing.....</b>	<b>9</b>
<b>Conclusion and Future Work .....</b>	<b>9</b>
<b>References .....</b>	<b>10</b>
<b>Appendix .....</b>	<b>11</b>

## Introduction

### Problem Statement and Aims of the Project

As university sports activities become increasingly diverse, schools are also attempting to build new sports facilities and offer more sports programs for student recreation. However, with the increase in activities and facilities comes an increase in sports facility bookings. The traditional manual ticketing system for reservations has become highly inefficient and prone to errors. In terms of such a system, many universities and institutions have already designed various sports center management systems [1]. However, a common **problem** with some of these systems is that they cannot effectively manage and filter these activities due to the diversity of sports activities. Furthermore, these systems tend to have relatively limited functionality and fail to attract customers, thus lacking commercial value. In this project, the "Online Booking Service for a Sports Centre", we **aim** to design a web-based sports center booking system for all students and staff. In addition to basic requirements such as user registration, activity classification, and activity booking, we hope to record some user sports and booking data in the system. Ultimately, we aim to encourage more students and staff to participate in various sports activities, increasing their enthusiasm for sports and maintaining their physical health.

### Project Scope

Up to now, the project has finished some basic functionalities to meet the user requirements, which are shown in Appendix 1. It is clear to see that this system supports **user registration, user login, booking activities by immediate payment, checking wallet, shopping cart, make payments from checking or coupon and administrative features like maintaining sports activities, orders and so on**. The system is accessible to all the students and staff.

### User Characteristics and Assumptions and Dependencies

For **Students**: Students may have diverse interests in sports and fitness activities, ranging from individual workouts to team sports. Many students are familiar with online booking systems and comfortable navigating web applications. Also, Students interested in sports and fitness, seek opportunities to participate in various activities offered by the sports center. Often juggling academic commitments and extracurricular activities, they seek convenience and efficiency in booking sports facilities. Considering that students do not have independent financial resources, they would require discounts such as coupons to book sports activities.

For **Administrators**: Personnel responsible for managing the online booking system, overseeing user accounts, configuring settings, maintaining some information about the activities, and addressing technical issues. Administrators may require access to analytics and reporting tools to monitor usage patterns, track facility utilization, and assess the effectiveness of promotions or programs. Ability to customize booking rules, pricing, and availability based on seasonal demands, special events, or facility maintenance schedules. These user characteristics should guide the design and development of the online booking service to ensure it meets the needs and expectations of all stakeholders involved.

#### Assumptions and Dependencies:

1. Internet Access: Users have reliable internet access to use the online booking system.
2. User Registration: Users are willing to register and provide the necessary information to create an account on the platform.
3. Payment Integration: Users have access to payment methods accepted by the system for booking payments. This system uses balance or coupons.
4. Data Security: User data, including personal information and booking details, is securely stored and protected from unauthorized access [2].
5. Maintenance and Support: Adequate resources are available for ongoing maintenance, updates, and user support for the booking system. For example, administrators can maintain the information on the coupons.
6. Regulatory Compliance: The system complies with relevant data protection laws and regulations governing online transactions and user privacy, which is stated clearly when logging in [3].

### Project Risk

Potential risks associated with the project include technical challenges during development, user adoption issues, and data security concerns. Mitigation strategies will be implemented to address these risks and ensure the successful delivery of the project.

## Architectural Design

Architectural design requirements significantly impact the overall structure, performance, security, availability, and maintainability of the system. Our project focuses on developing an online sports centre booking system, based on these architectural design principles: **performance**, **security**, **scalability**, **data consistency**, **legal**, and **compliance**. By adhering to these principles, our projects have a reasonable response time; valid registration, login security; scalable vertical expansion capability; timely data updates, and appropriate user legal notices.

Our software adopts the MVC framework, which includes the user system and the administrator system, consisting of 10 systems and a reasonable cache system. The user system supports login and registration as well as subsequent wallet, booking, and coupon systems. The administrator system supports the management of coach, sport, coupon, and stadium systems. Our system architecture diagram is shown in Figure 1.

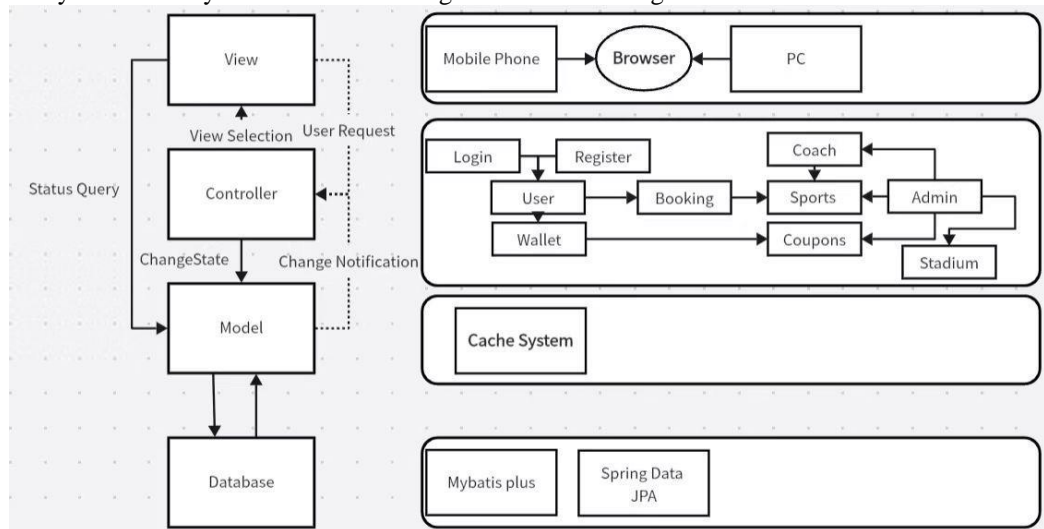


Figure 1: System architecture diagram

## System Architecture

### Architecture Style: MVC

#### Model

Our Model components, crucial for managing business data and logic, are implemented in the system through Java classes, such as `UserInfo`, `BookingInfo`, `SportActivity`, etc. These classes interact with the database and are responsible for data access, update, and delete operations. Using Spring Data JPA and MyBatis Plus, the model can effectively interact with the MySQL database to ensure data consistency and integrity.

#### View

The View layer manages the user interface and front-end logic, utilizing HTML, CSS, and JavaScript, and is enhanced by Bootstrap for responsive design. The view receives user inputs and passes them to the controller for processing and updates the user interface based on the returned data.

#### Controller

Acting as an intermediary between the model and the view, the controller processes all user input, calls the model to execute the requested business logic, and then selects the appropriate view to respond. Each controller component, such as `UserController` and `BookingController`, **encapsulates** specific logic to ensure that the application's request processing is structured and organized.

### Additional Architecture

#### Cache System

We design a cache system that allows to cache of frequently accessed data to reduce database access time and improve database performance.

#### Architecture Benefits:

**Modularity:** The MVC architecture provides a clear separation of responsibilities, and the responsibilities of each layer are clear, which is conducive to the organization and maintenance of code.

**Flexibility and maintainability:** Each component is independent, easy to modify and upgrade independently, and does not interfere with other parts, reducing the difficulty of system maintenance.

**Extensibility:** As the project scales, new views and controllers can be easily added, or the back-end logic changed without affecting the front-end user experience.

This design not only ensures the functionality and efficiency of the system but also supports future growth and change, making it a robust architectural choice that meets the needs of modern Web applications.

## Frontend System

We select the following components and technology to help us with front-end development.

### HTML/CSS/JavaScript

We use these technologies to build the foundation of web pages, HTML for structured content, CSS for style design, and JavaScript for page interactivity. We build user-friendly interfaces for each PBI, with efficient navigation and pop-up pages for a rich human-computer interaction experience.

### Bootstrap

We leverage this front-end framework which supports the development of responsive web pages. It provides a rich set of prefabricated components and a powerful grid system, allowing for quickly building a stable, responsive user interface. It helps us to simplify the front-end development work and speeds up the development speed.

### jQuery

We use this JavaScript library that helps us to simplify HTML document traversal, event handling, animation, and Ajax interactions. Its ease of use and extensive plugin library can greatly improve the development efficiency.

## Backend System

### Spring Boot

Our backend system is built entirely on Spring Boot. It is an open-source Java-based framework for creating microservices, simplifying Spring-based application development, and automatically configuring project infrastructure. It streamlines the deployment and development process, with many runtime dependencies built in, support for various plug-ins, and easy integration with external systems. It helps us to automatic configuration and standalone deployment capabilities are ideal for rapid development.

### Spring Data JPA & MyBatis Plus

We use Spring Data JPA and MyBatis Plus for database and back-end interaction. Spring Data JPA support automatically connects the data access layer to the JPA implementation. MyBatis Plus is an enhancement to MyBatis that simplifies CRUD operations. Spring Data JPA is for developers who prefer to use object-relational mapping (ORM), while MyBatis Plus provides more control and flexibility for complex query situations [4]. Using these two technologies together, we can optimize database operations and meet different business requirements.

### MySQL

We use MySQL for database support, which provides the automatic configuration capabilities we need and the full ACID (atomicity, Consistency, isolation, persistence) transaction model, which guarantees data consistency. Moreover, the MySQL documentation is more detailed, which helps us quickly solve database build problems.

### Swagger

We use Swagger which simplifies API development and helps us design, build, and document RESTful Web services. Swagger automatically generates API documentation and an interactive API console. It enables us to clearly understand and test the API when we develop front-end functions, ensuring the correct implementation of the interface [5]. Our swagger detailed information can be checked in Appendix 2.

## High-level Database Design

Our project develops a web-based system for university staff and students to book sports activities. The Entity-Relationship Diagram (ERD) shown in Figure 2 provides a detailed view of the database designed to ensure the data organization meets the system requirements. Here is a detailed explanation of the entities and their relationships:

1. **User Information (user\_info):** Stores detailed information about registered users, including name, gender, date of birth, position, educational background, etc. User ID (id) serves as the primary key, linking to other tables such as booking information and wallet information.
2. **Coach Information (coach\_info):** Records basic information about coaches and their activity times. Coach ID (id) is used to link to specific sports activities (sport\_activity).
3. **Stadium Information (stadium):** Includes names of sports venues and their remaining space. Venue ID (id) connects to the sports activity table, showing where each activity takes place.
4. **Sports Activity (sport\_activity):** Records detailed information about sports activities, such as the name of the activity, start and end times, ticket prices, and coaches involved. Sports Activity ID (id) serves as the primary key and connects to coach information, booking information, and venue tables.

5. **Booking Information (booking\_info):** Contains details of user bookings, such as the activity booked, date, time, price, and booking status (e.g., confirmed, canceled). Booking ID (id) serves as the primary key and links to user and wallet information tables.
6. **Wallet Information (wallet\_info):** Financial details of users, such as wallet balance. Wallet ID (id) links to user and booking information tables to handle payment transactions.
7. **Coupon Information (coupon\_info):** Details of coupons such as name, start and end times, and usage status. Linked to the user ID, allowing users to apply coupons during bookings.

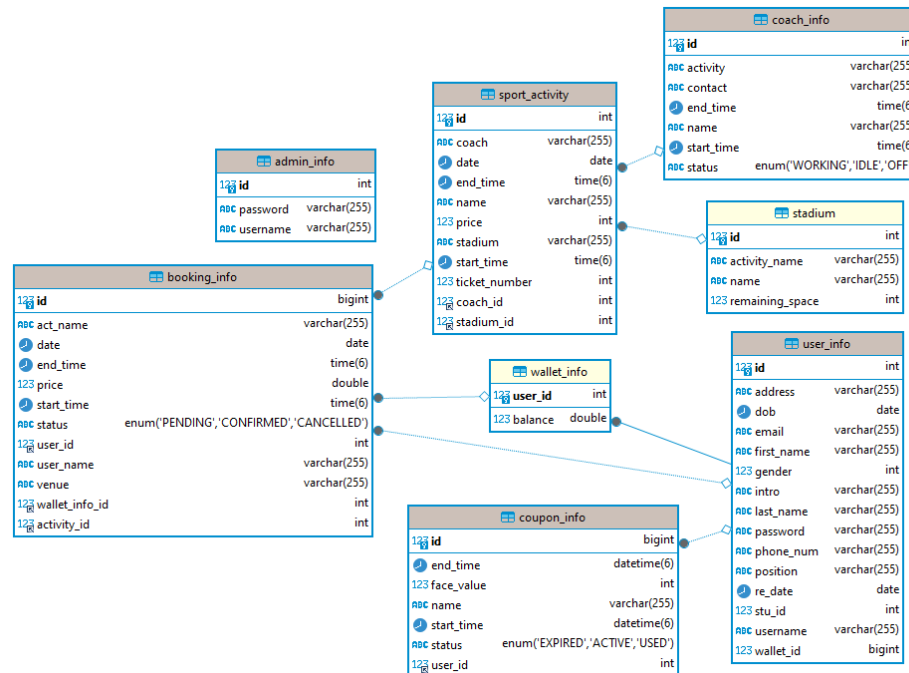


Figure 2: ERD

### Relationships and Their Importance

- **User to Booking:** The relationship between users (user\_info) and bookings (booking\_info) is central, ensuring users can book specific sports activities.
- **Sports Activity to Venue:** Each sport activity (sport\_activity) takes place in a specific venue (stadium), reflecting the direct relationship between venues and activities.
- **Coach to Sports Activity:** Coaches (coach\_info) are directly linked to specific sports activities, indicating who is responsible for coaching each activity.

### Structural Justification

The ERD structure effectively supports system functionalities such as user registration, activity browsing, booking, and payment processing. Additionally, each entity is equipped with essential attributes to support detailed reporting and statistical functions, meeting the needs of administrators to monitor and maintain the system.

## Software Design

### Software Modules

From a code logic perspective, the system is modularized for enhanced maintainability, scalability, and reusability. Each module has a distinct function and operates independently yet interacts through defined interfaces. For example, the login function and registration function on the login page are two independent modules. Moreover, the booking sports activity module only needs to know how to manage bookings, without paying attention to the logic of payment processing and user information. In this system, each module is independent and interacts through defined interfaces. For instance, after the user completes the booking, the booking management module will call the payment processing module to process the user's payment. This design can make the system more modular, reduce coupling between modules, and improve the readability and maintainability of the code.

Beyond modularity, software design methods like object-oriented design and componentization can also enhance software quality and maintainability. Object-oriented design treats software systems as collaborative objects with unique responsibilities, as evident in our system's development. For instance, sports activities are objects with specific attributes (activity name, venue) and methods (create, cancel reservation). Booking objects have attributes like booking time, username, and sports activities. Componentization, another method, divides systems into reusable, independent parts. In the sports center reservation system, these components could be user management (handling user info, login), administrator management (managing reservation info, coach info, sports activities), and payment management (processing payments). Besides, to keep attributes type-safe and provide readability, we use enumeration classes for the states of booking, coach, and coupon

## High-level Process Flow

### The activity flow of use cases

The system facilitates registration, login, sports activities browsing, booking and payments, and personal information modification. Users first register with personal information (ID, username, email, etc), then use these to access the homepage. Here, users can review available sports activities, view information, booking records, and wallet information. For the view of the personal information function, users can click the "Information" button to enter the user information page, where they can also modify personal information by clicking the "Update" button (enter modification information in the pop-up modal box). For the view reservation records function, users can click the "Booking" button to enter the user reservation page, where they can choose to click "All", "Unpaid" and other buttons to filter the order information they want to see. After clicking on the "PayAllByBalance" option, a payment prompt appears. For viewing wallet function, when the user clicks the "Wallet" button on the homepage, the user wallet page will display, allowing users to view their coupon information and wallet balance. For the booking sports activity function, users can select sports activities and make payments on the payment page. The basic flowchart of the system is as follows:

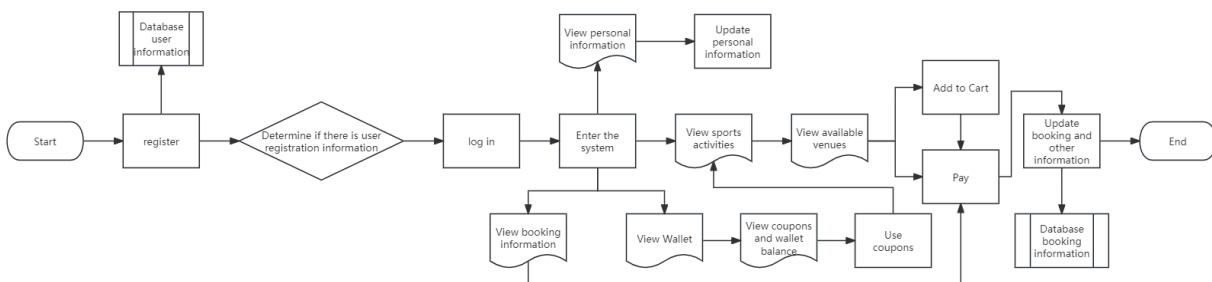


Figure 3: Flowchart of the system

Aside from the user interface, the system can also be managed by administrators. For access, the administrator must initially log in using their username and password. On the homepage, they can manage sports activities, stadiums, coaches, bookings, user information, coupons, and view reports (including monthly ticket counts and sales).

### Main process design (the process of booking sports activities)

The system's principal function is to book sports activities. Users need to select their desired sports activities from the homepage, following which they're led to the corresponding booking page. Here, they can either directly pay or add to their cart after agreeing to the user conditions. On selecting "Pay", users are offered the choice of paying through a wallet or coupons. After payment completion, the booking page updates with the related order information stating a 'Pending' sequence. Post finishing the sports activity booking, this order status shifts to 'Confirm'. Users can cancel their orders any time before the 'Confirm' status. More UMLs (Activity diagram, State diagram, Class diagram) of the system is shown in the Appendix 3, Appendix 4, Appendix 5, Appendix 6, Appendix 7, Appendix 8, Appendix 9.

### Abnormal process design

The system features specific safeguards against anomalies including login, payment, and information retrieval issues, etc. For instance, if a user inputs an incorrect or non-existent ID and password on the login page, the system will encourage them to either register or re-enter accurate details. If the user's selected payment method, such as wallet or coupon, lacks sufficient funds, the system will issue a prompt. Key user operations will trigger a system pop-up inquiry about continuation, thus mitigating operational mishaps and ensuring system stability and reliability.

## Software Support Services

### Database

The project involves the creation, configuration, monitoring, and maintenance of a database using spring-boot as a



framework. By utilizing Mybatis-plus, database creation becomes simplified through the addition of corresponding dependencies. This enables the use of annotations in the entity, mapper, and controller classes to achieve table creation. The project manages various types of data, including user, administrator, coach, sports activity, wallet balance, venue, coupons, and order information. The implementation allows for easy addition, deletion, and retrieval of information, with MySQL being used to create and manage the database. Additionally, data security measures are in place to ensure that only authorized users with personal database passwords can access the database.

### **Security**

Our project establishes security measures to ensure the protection of user passwords. During user registration, passwords must be at least 6 characters long and cannot contain repeated or consecutive characters to prevent theft. Additionally, the system also offers a password reset service for users who forget their password, ensuring that new passwords adhere to the security guidelines. If a user attempts to set a non-compliant password, the system will provide a reminder outlining the password requirements.

### **Webpage Navigation**

The project consisted of a web navigation service focusing on front-end design. A navigation bar located at the top of each page allows users to easily access desired pages through hyperlinks. The home page features a carousel display of available sports, complemented by a "Learn More" button that redirects to the booking page for detailed information. On mouse hover, the text on the homepage zooms in to show clickable hyperlinks to the appropriate pages. When selecting a time and location for a sporting event, the system reminds the user that they must review and agree to the venue rules before paying for the event or adding the event to their shopping cart. Upon entering the payment page, if the user's wallet balance is insufficient, clicking on "Confirm Payment" will bring up a pop-up reminder. The admin pages have a sidebar on the left for quick navigation. In addition, all buttons on the page have a color that represents their functions; all popup pages have the appropriate close button.

### **Coding Structure and Convention**

Our project uses GitHub to centralize file **organization**. Team member uploads the contributions and merges them into the main branch following peer review and approval. We have implemented coding **standards** across the project to ensure consistency. This includes uniform indentation, predefined bracket positioning, and standardized comment placement. Corresponding classes are created in the controller and model layers of this system to make its declarations uniform.

Uniform **statement** representation is crucial for ensuring consistent execution across various scenarios. This is achieved by standardizing the use of conditional statements and encapsulating related statements within functions. Our project adheres to strict **naming** conventions to enhance code readability and maintainability. We employ camelCase for naming variables, functions, and front-end file names; and using a standard class naming strategy for naming the Java classes.

Our project uses MVC architecture, so our design is encapsulated in three components to achieve **encapsulation** specification and avoid development interference: model, view, and controller. They correspond to the modal, static, and controller folders. To manipulate the database, we also have the mapper folder to manage the database mapping.

### **Software Configuration and Production Environment**

#### **Development Environment**

In our development, we use IntelliJ IDEA to develop the back end, and IntelliJ IDEA or VScode to develop the front end. The systems we used include Windows and MacOS. The project uses JDK17, MySQL 8.0.36 for the database, Postman or Swagger for back-end Api debugging, and MySQL Workbench or DBeaver for database management.

#### **Dependency**

Our software configuration employs six key dependencies for streamlined development. We use "spring-boot-starter-data-jpa" for database integration, "spring-boot-starter-web" for web development, "spring-boot-devtools" to enhance developer productivity, "MYSQL-connector-j" ensures MySQL connectivity, and "spring-boot-starter-test" supports the testing. Moreover, we use "springdoc-openapi-starter-webmvc-ui" to generate the Swagger API interface and enhance frontend developers' ability to efficiently interact with backend APIs. And we used "mybatis-plus-spring-boot3-starter" to significantly improve database development efficiency. Additionally, "Lombok" is employed to reduce boilerplate code in Java objects, thereby enhancing code maintainability and simplifying our codebase.

#### **Dependency Configuration**

After adding the dependencies, we did the following dependency configuration to achieve the best development process. We configure Hibernate to automatically update the schema, ensuring that database structures are always up to date. To get consistent session management, we set the JDBC session schema to be initialized at every start. Besides, we activate JPA Auditing with remote restart capabilities targeted at specific paths to reload code changes dynamically. Moreover, to make MyBatis-plus more controllable, we set it to output directly to the console.



## Production Environment

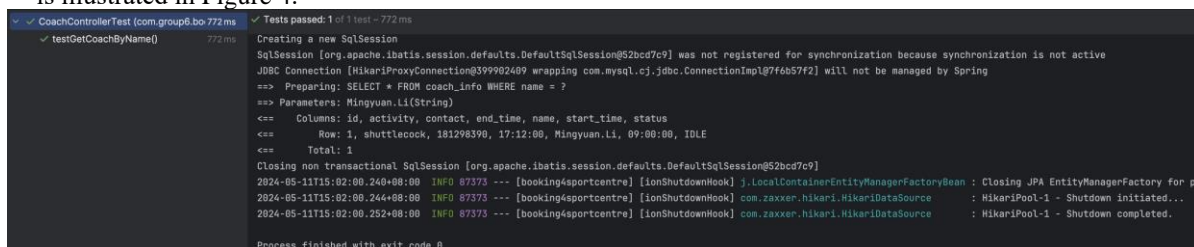
In the production environment, we standardized the configuration of JDK and MySQL to match the development environment. Then, we packaged the entire project in JAR format utilizing Maven's lifecycle packaging function. Subsequently, based on our server configuration, the JDBC session schema initialization was performed upon each startup. After that, we executed the JAR program via the command line to launch the project, ensuring consistent session management across the runtime environment. Finally, we conducted testing on the cloud server to verify that the project's behavior has the same results observed during local tests, which means our server configuration is valid.

## Software Testing

### Unit Testing

The unit testing conducted for the 'CoachController' class encompasses five methods:

1. **testViewCoachInfo():** Verify that calling 'viewCoachInfo' without input returns a non-null list of seven "CoachInfo" objects utilizing 'assertNotNull'.
2. **testDeleteCoachInfo():** Confirm the effectiveness of the 'deleteCoachInfo' method by deleting a coach by ID and verifying the non-null outcome and accurate deletion confirmation.
3. **testAddCoachInfo():** Test the 'addCoachInfo' method by passing a complete "CoachInfo" object and validating the addition with 'assertNotNull' and 'assertEquals'.
4. **testUpdateCoachInfo():** Check the update functionality for existing coach information by modifying details with a "CoachInfo" object and confirming the update is correct and non-null.
5. **testGetCoachByName():** Ensure 'getCoachByName' fetches coach details accurately by invoking it with "Mingyuan. Li" and verifying the response using 'assertNotNull' and 'assertFalse'. Console output for this method is illustrated in Figure 4.



```
CoachControllerTest (com.group6.bo.772ms) Tests passed: 1 of 1 test - 772ms
  testGetCoachByName() 772ms
    Creating a new SqlSession
    SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@52bcd7c9] was not registered for synchronization because synchronization is not active
    JDBC Connection [HikariProxyConnection@399902409 wrapping com.mysql.cj.jdbc.ConnectionImpl@7f6b57f2] will not be managed by Spring
    ==> Preparing: SELECT * FROM coach_info WHERE name = ?
    ==> Parameters: Mingyuan.Li(String)
    <== Columns: id, activity, contact, end_time, name, start_time, status
    <== Row: 1, shuttlecock, 181298390, 17:12:00, Mingyuan.Li, 09:00:00, IDLE
    <== Total: 1
    Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@52bcd7c9]
    2024-05-11T15:02:00.240+08:00 INFO 87373 --- [booking@sportcentre] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for po
    2024-05-11T15:02:00.244+08:00 INFO 87373 --- [booking@sportcentre] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...
    2024-05-11T15:02:00.252+08:00 INFO 87373 --- [booking@sportcentre] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.
    Process finished with exit code 0
```

Figure 4: Console output of a unit testing

Similar unit tests have been performed for each method within various classes of the application, including 'AdminLoginController', 'AdminUserInfoController', 'BookingController', 'CoachController', 'CouponController', 'ReportController', 'SportActivityController', 'StadiumController', 'UserController', 'UserInfoController', 'UserLoginController', and 'WalletController'.

### Integration Testing

We leveraged the Spring Boot Test framework to conduct integration testing on five controllers, validating their internal and external interactions and functionalities.

1. **AdminUserInfoControllerTest:** In this testing, we sequentially tested the complete workflow of user management. Starting with adding a user, we then verified the addition by querying the user. Next, we modified the user's profiles and checked these changes by querying the user by id. Finally, we tested deleting the user, ensuring each step was contingent on the success of the previous operations.
2. **BookingControllerTest:** In this testing, we tested the entire booking process which involves the wallet modal. The testing began with creating a booking, followed by querying to confirm its successful creation, then we updated the booking details. Subsequently, confirmed the booking which successfully deducted the balance in the wallet for the correct price. Finally verified the deletion of the booking.
3. **SportActivityControllerTest:** In this testing, we tested the entire activities management process which involves the Coach and Stadium modal. Starting by adding a coach and a stadium, followed by creating an activity that utilized the added coach and stadium information. We conducted a query of all activities to ensure their proper adding. Next, we modified the ticket count of an activity, followed by querying a specific activity by its ID. Then, we updated other details of the activity and deleted the activity.

We also tested the complete process of adding, deleting, modifying, and checking **CoachController** and **StadiumController**. The testing results showed that our controllers passed all testing correctly, test results are shown in Figure 5, and more results can be found in Appendix 10.1, 10.2, 10.3, 10.4, 10.5.

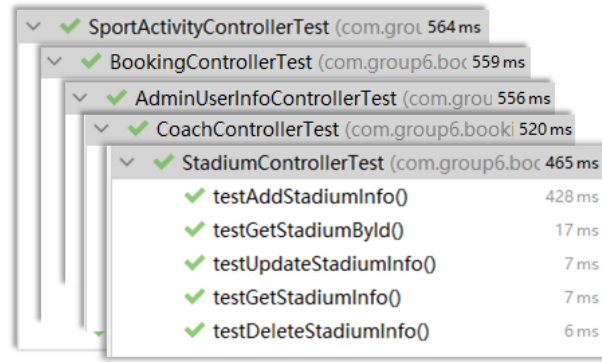


Figure 5: Integration testing results

## System Testing

1. **Functional Integrity:** Our software is divided into a user system and an administrator system, including 10 sub-systems. These tests are conducted to verify all user-facing functions, including user interactions, data input, and processing outcomes. The administrator system focuses on master file management and admin-level operations.
2. **Performance:** This testing focuses on software response time, including front-end and back-end response time.
3. **Security:** Security testing focuses on login security, the validity of 2 password security logics, and the feasibility of retrieving passwords.
4. **Compatibility:** The software is tested on multiple browsers (Edge, Safari, Firefox), and all functions can be used without error.

The system testing results can be checked in Table 1:

Test Category	Functional Integrity	Performance	Security	Compatibility
Results	All functions performed as expected	Response time within limits	All security principles are valid	Compatible with all tested browsers

Table 1: System testing results

## Acceptance Testing

After we deployed the software on the Alibaba Cloud server, we invited four real users to participate in this test, assuming the roles of both administrators and system users, covering the complete process from administrator backend operations to user system interactions, conducted across various desktop devices using browsers like Edge, Firefox, and Safari. The detailed results can be access in the Appendix 11.

1. **Administrator Operations:** Including login, logout, adding, modifying, and deleting entries in the system's master files, as well as viewing orders, user information, and statistical reports.
2. **User Operations:** Including registration, login, logout, updating personal information, booking activities, and updating bookings.

**Test Results (Table 2):**

Test Category	Total Tests	Tests Passed	Tests Failed	Pass Rate	Test Coverage
Results	31	31	0	100%	100%

Table 2: Acceptance testing results

## Conclusion and Future Work

In **conclusion**, our software engineering group has successfully designed, developed, implemented, and tested the basic service of the online booking system of the sports center. Throughout the project lifecycle, we adhered to industry-standard software engineering practices, including requirements gathering, design, implementation, testing, and deployment. Specifically, we completed the front-end, back-end, and database full-stack development of 10 systems under the MVC framework, which showed robust support services and excellent code convention. However, we have yet to address the issue of integrating specific activities with centralized payment solutions.

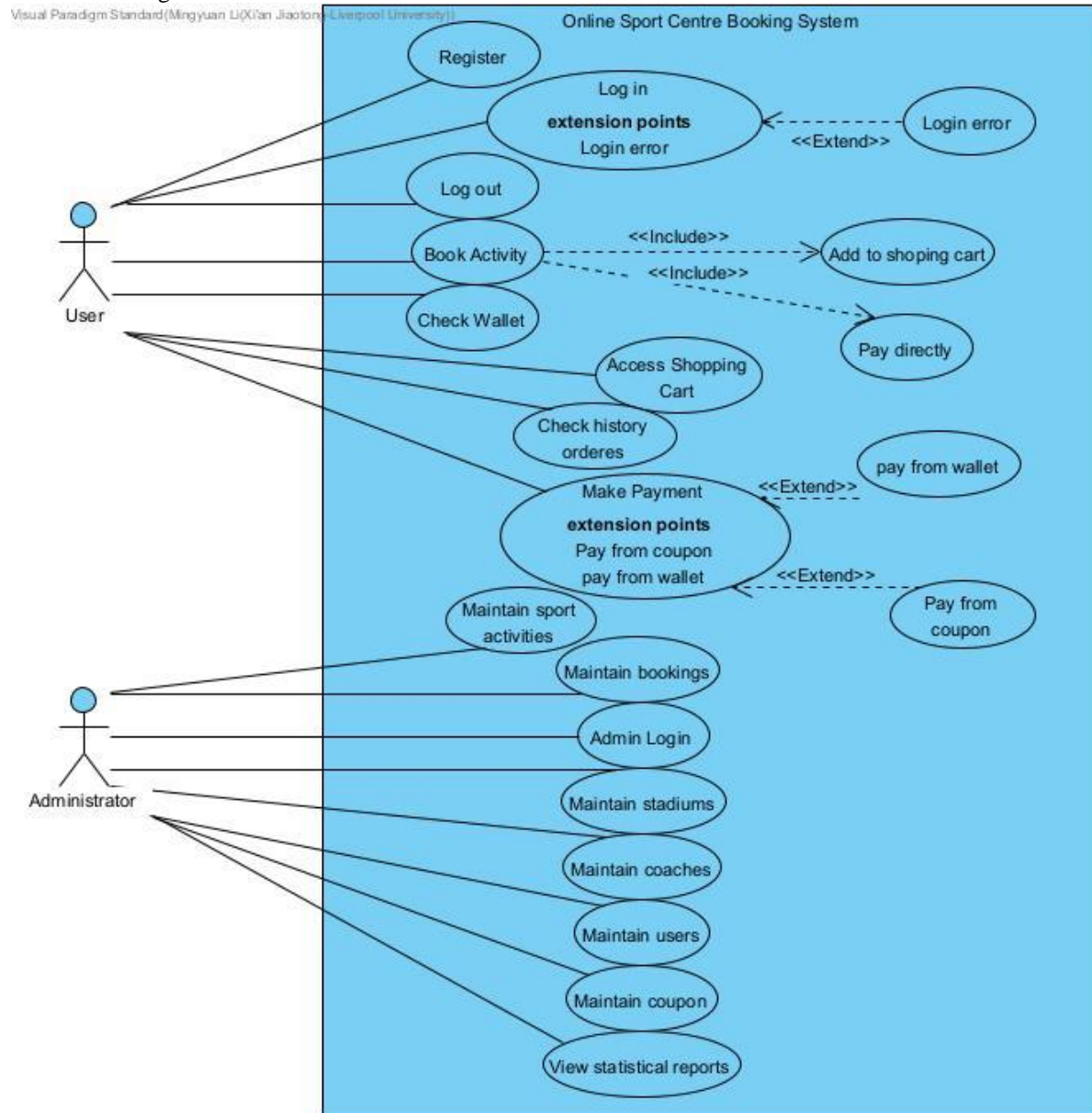
In the **future**, the group will polish the interface of the booking to make it more navigational and user-friendly. Besides, we will improve the payment for the bookings, including selecting specific activities for centralized payment. Starting on May 14, we will start the development of the next sprint to optimize the booking interface, and on May 21, we will start to solve the centralized payment issue.

## References

- [1] G. I. Alshammare, M. S. B. A. Halim, and G. A. A. Alsheikh, "Online Booking Services Assisted by Technology to Improve Customer Loyalty in Jordanian Five-Star Hotels," *International Journal of Professional Business Review*, vol. 7, no. 3, 2022. [Online]. Available: <https://search-ebscohost-com-s.elink.xjtlu.edu.cn:443/login.aspx?direct=true&db=edselc&AN=edselc.2-52.0-85140753490&site=eds-live&scope=site>. [Accessed: 12-May-2024]
- [2] F. Liao, L. Hu, J. Wang, Z. Wang, and J. Gan, "Research and suggestions on scientific data security standards," *Kexue Tongbao/Chinese Science Bulletin*, vol. 69, no. 9, pp. 1142-1148, 2024. [Online]. Available: <http://dx.doi.org/10.1360/TB-2023-0228>
- [3] A. Berger et al., "Towards Automated Regulatory Compliance Verification in Financial Auditing with Large Language Models," in *2023 IEEE International Conference on Big Data (BigData)*, pp. 4626–4635, 2023. [Online]. Available: <https://search-ebscohost-com-s.elink.xjtlu.edu.cn:443/login.aspx?direct=true&db=edsee&AN=edsee.10386518&site=eds-live&scope=site>. [Accessed: 12-May-2024]
- [4] Y. Z. Li, S. Gao, J. Pan, B. F. Guo, and P. F. Xie, "Research and application of template engine for web back-end based on Mybatis-PLUS," in *Procedia Computer Science*, vol. 166, Kunming, China, 2020, pp. 206-212. [Online]. Available: <http://dx.doi.org/10.1016/j.procs.2020.02.052>
- [5] Y. Tashtoush, M. N. Alrashdan, O. Salameh, and M. Alsmirat, "Swagger-based jQuery Ajax validation," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019*, Las Vegas, NV, United States, 2019, pp. 69-72. [Online]. Available: <http://dx.doi.org/10.1109/CCWC.2019.8666542>

## Appendix

1. Our source code is available at: [https://github.com/Sushang-Li/CPT202\\_Group\\_6](https://github.com/Sushang-Li/CPT202_Group_6)
2. Access our website at: <http://8.149.129.247:8080/index.html> (Impermanence, only for CPT202 grading)
3. Use case diagram



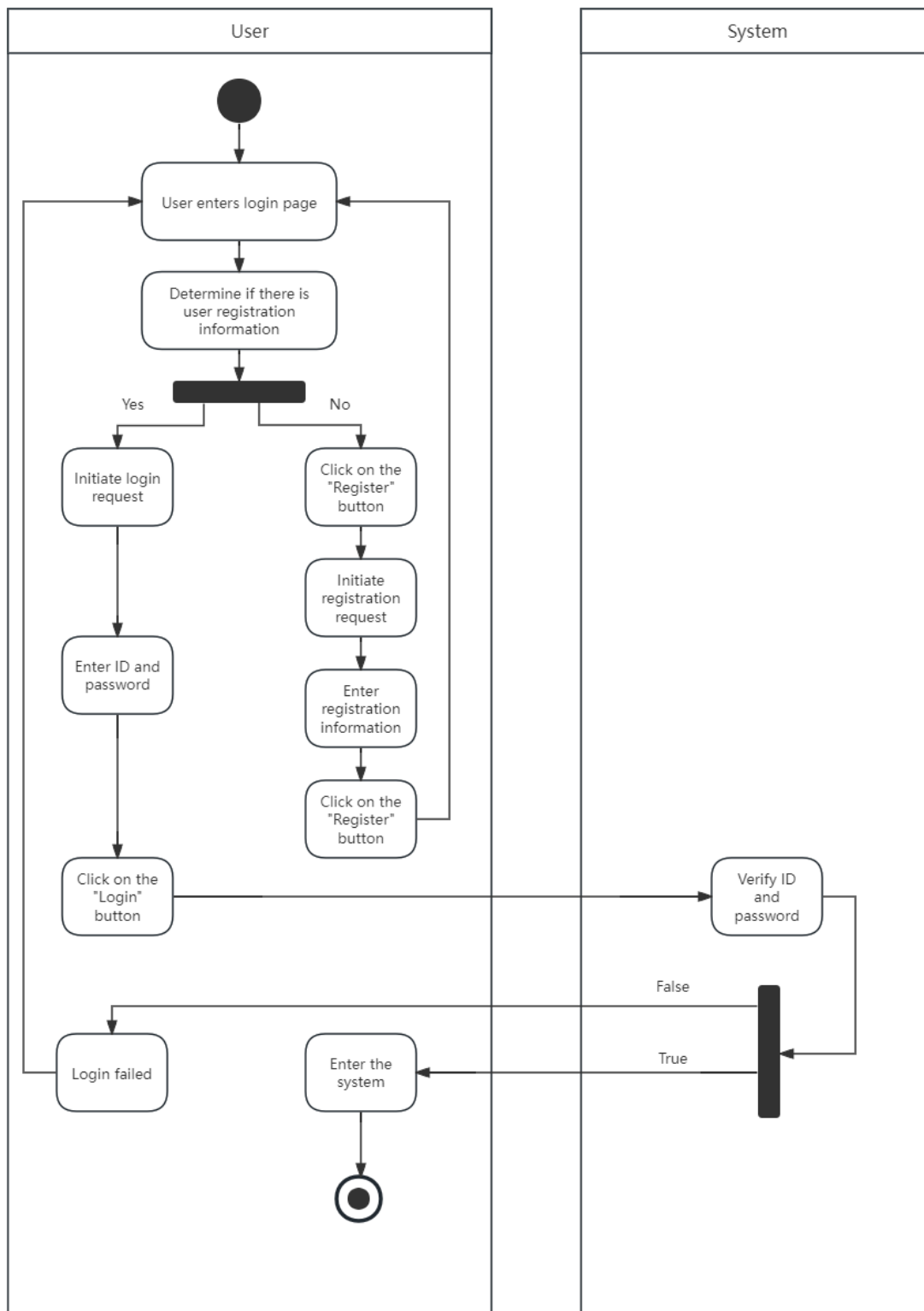
Appendix 1: Use case diagram

#### 4. Apis visualization in the swagger

<b>stadium-controller</b>		^
GET	/api/stadium/{id}	▼
PUT	/api/stadium/{id}	▼
POST	/api/stadium/add	▼
GET	/api/stadium	▼
DELETE	/api/stadium/delete/{id}	▼
<b>sport-activity-controller</b>		^
PUT	/api/sportActivity/{id}	▼
POST	/api/sportActivity/updateTicketNumber	▼
POST	/api/sportActivity/add	▼
GET	/api/sportActivity	▼
GET	/api/sportActivity/updateBookActivity	▼
GET	/api/sportActivity/getActivity/{id}	▼
GET	/api/sportActivity/allActivities	▼
DELETE	/api/sportActivity/delete/{id}	▼
<b>booking-controller</b>		^
GET	/api/bookings/{id}	▼
PUT	/api/bookings/{id}	▼
DELETE	/api/bookings/{id}	▼
POST	/api/bookings/updateWalletInfoId	▼
POST	/api/bookings/processArray	▼
POST	/api/bookings/confirm	▼
POST	/api/bookings/api/createBooking	▼
POST	/api/bookings/addOnBooking	▼
GET	/api/bookings	▼
GET	/api/bookings/updateOnBooking/{id}	▼
GET	/api/bookings/getAllBookings	▼
GET	/api/bookings/deleteOnBooking/{id}	▼
GET	/api/bookings/data/{date}	▼
GET	/api/bookings/admin_get_BookingInfo	▼
<b>wallet-controller</b>		^
POST	/api/wallet	▼
POST	/api/wallet/recharge	▼
POST	/api/wallet/pay	▼
GET	/api/wallet/{userId}	▼
GET	/api/wallet/balance/{userId}	▼
<b>user-info-controller</b>		^
POST	/api/user/updateInfo	▼
GET	/api/user/information	▼
<b>coupon-controller</b>		^
POST	/api/update_coupon	▼
POST	/api/coupons/user/{id}	▼
POST	/api/add_coupon	▼
GET	/api/get_coupon_by_status	▼
GET	/api/get_coupon_by_name	▼
GET	/api/coupons	▼
GET	/api/coupons/user/{userId}	▼
DELETE	/api/delete_coupon/{id}	▼
<b>coach-controller</b>		^
POST	/api/update_coach	▼
POST	/api/delete_coach	▼
POST	/api/coach_by_name	▼
POST	/api/add_coach	▼
GET	/api/coachInfo	▼
<b>admin-user-info-controller</b>		^
POST	/api/updateUser	▼
POST	/api/addUser	▼
GET	/api/getUserById/{id}	▼
GET	/api/UserInfo	▼
DELETE	/api/deleteUser	▼
<b>report-controller</b>		^
GET	/api/report	▼

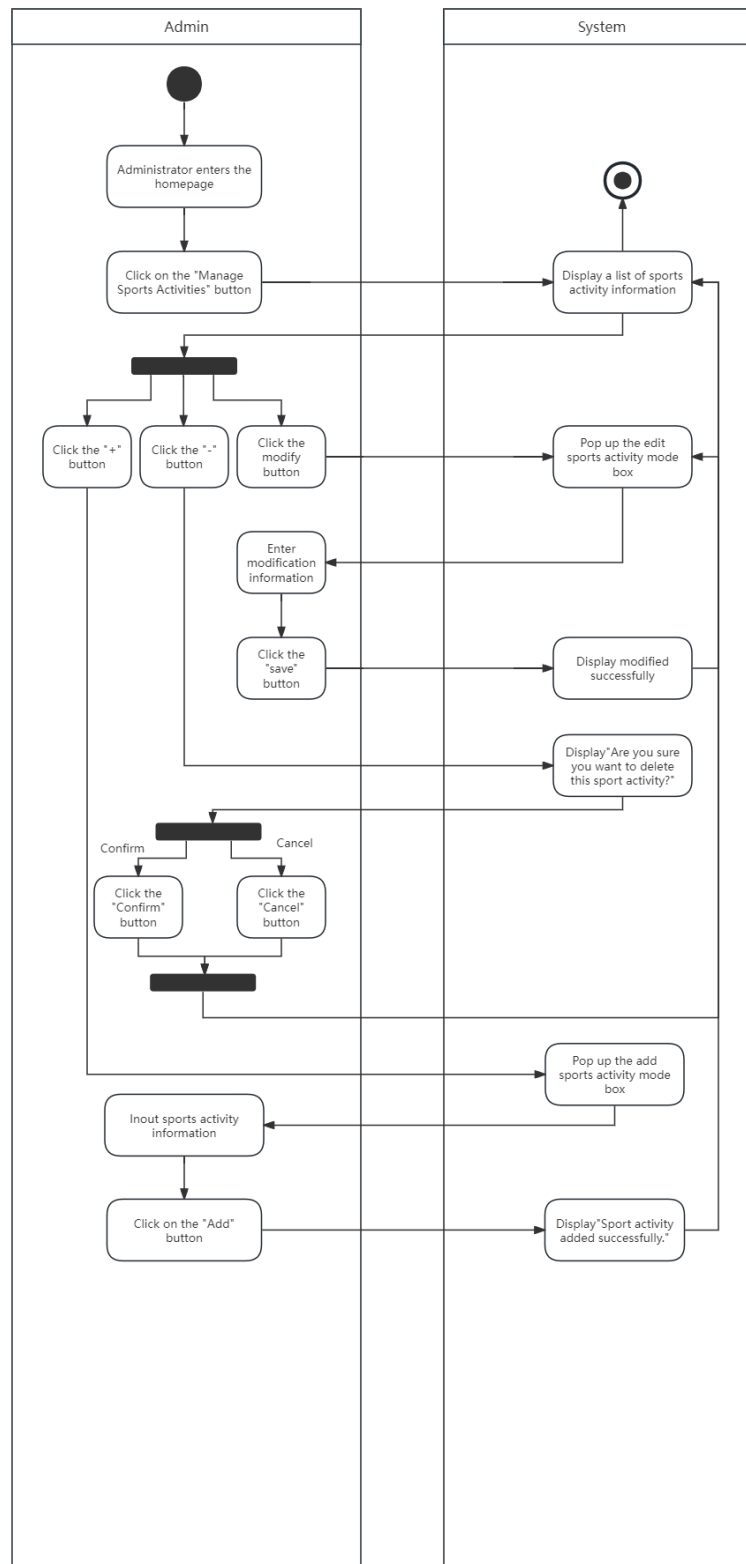
Appendix 2: Apis visualization in the swagger

## 5. User login and registration activity diagram



Appendix 3: User login and registration activity diagram

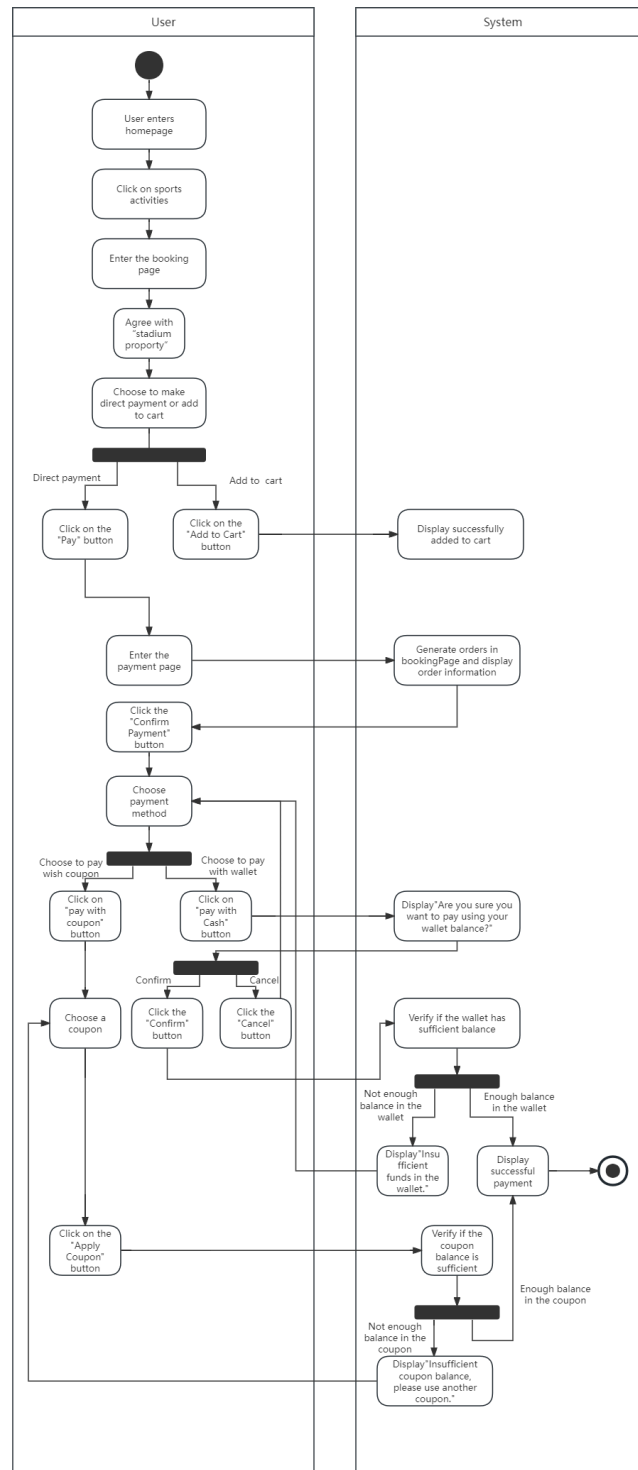
## 6. The administrator manages the sports activity diagram



Appendix 4: The administrator manages the sports activity diagram

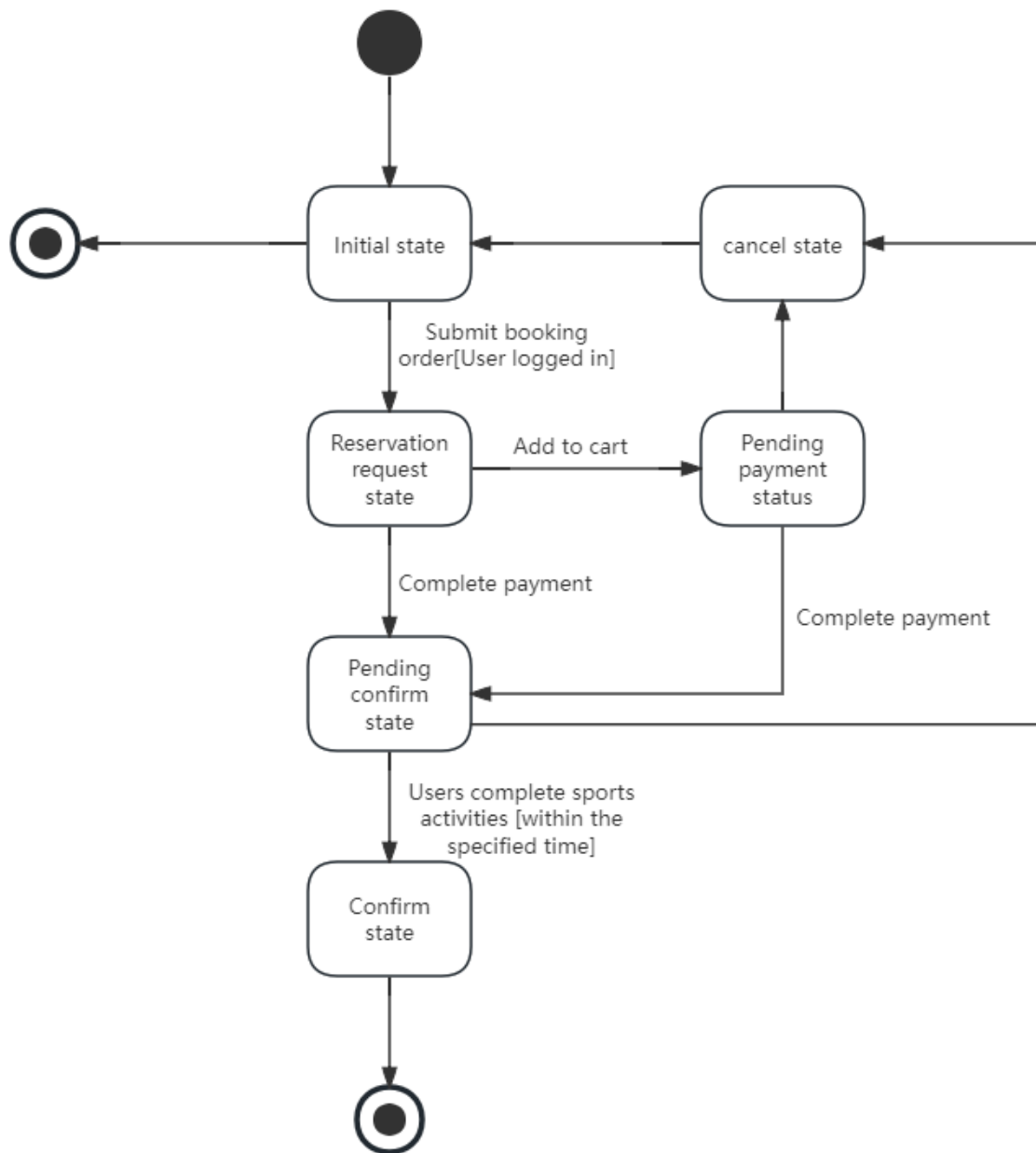


## 7. User booking activity diagram



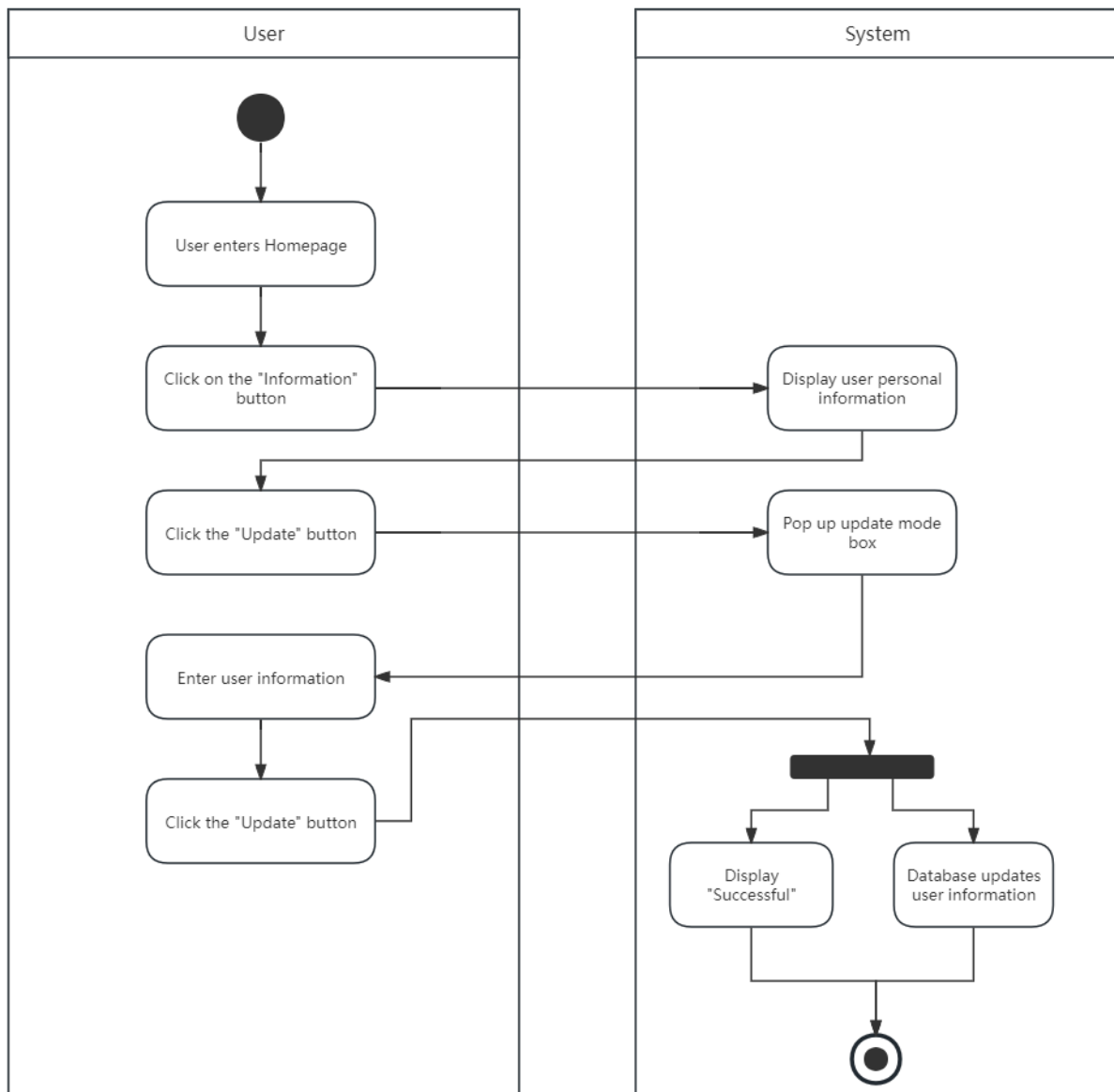
Appendix 5: User booking activity diagram

## 8. Order state diagram



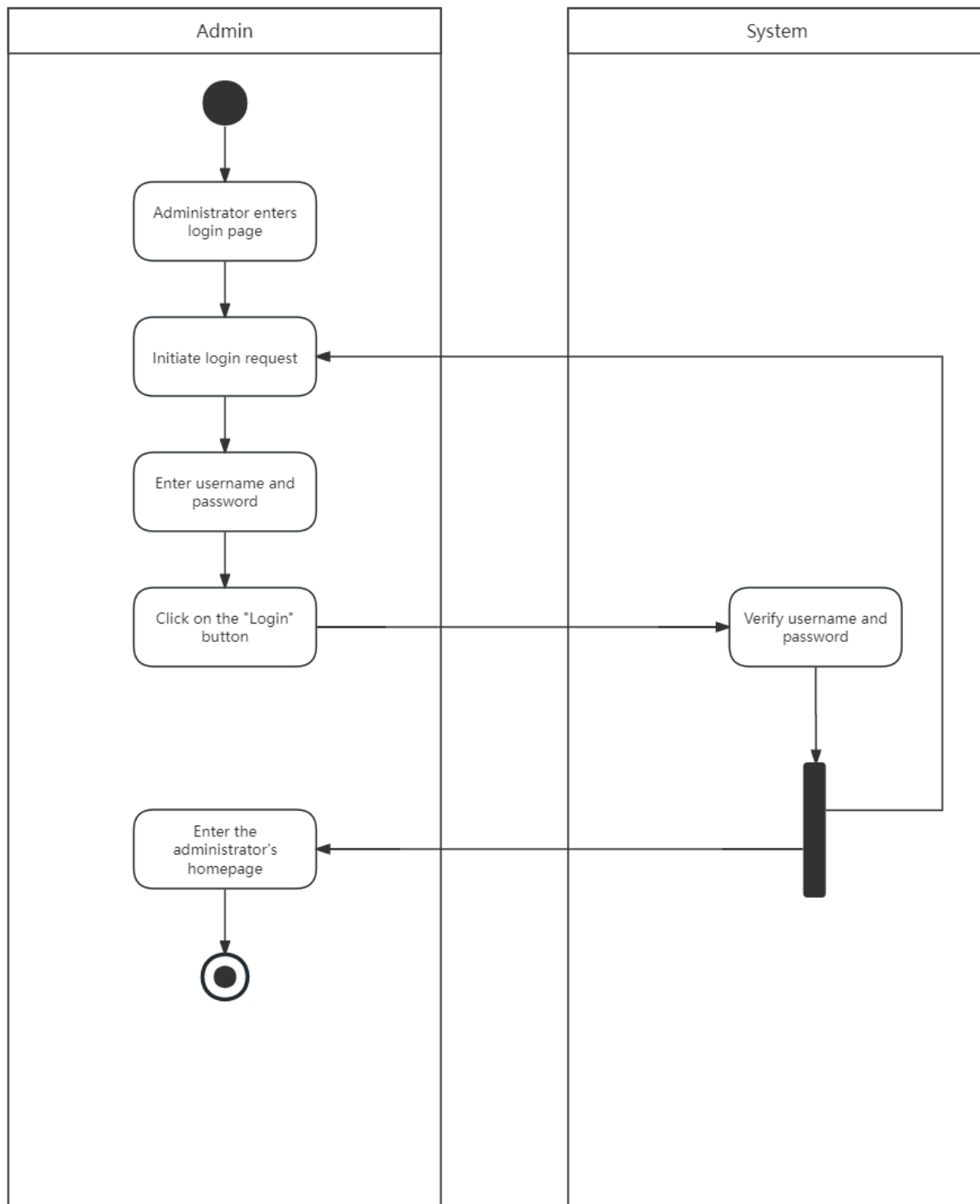
Appendix 6: Order state diagram

## 9. User view and modify user information activity diagram



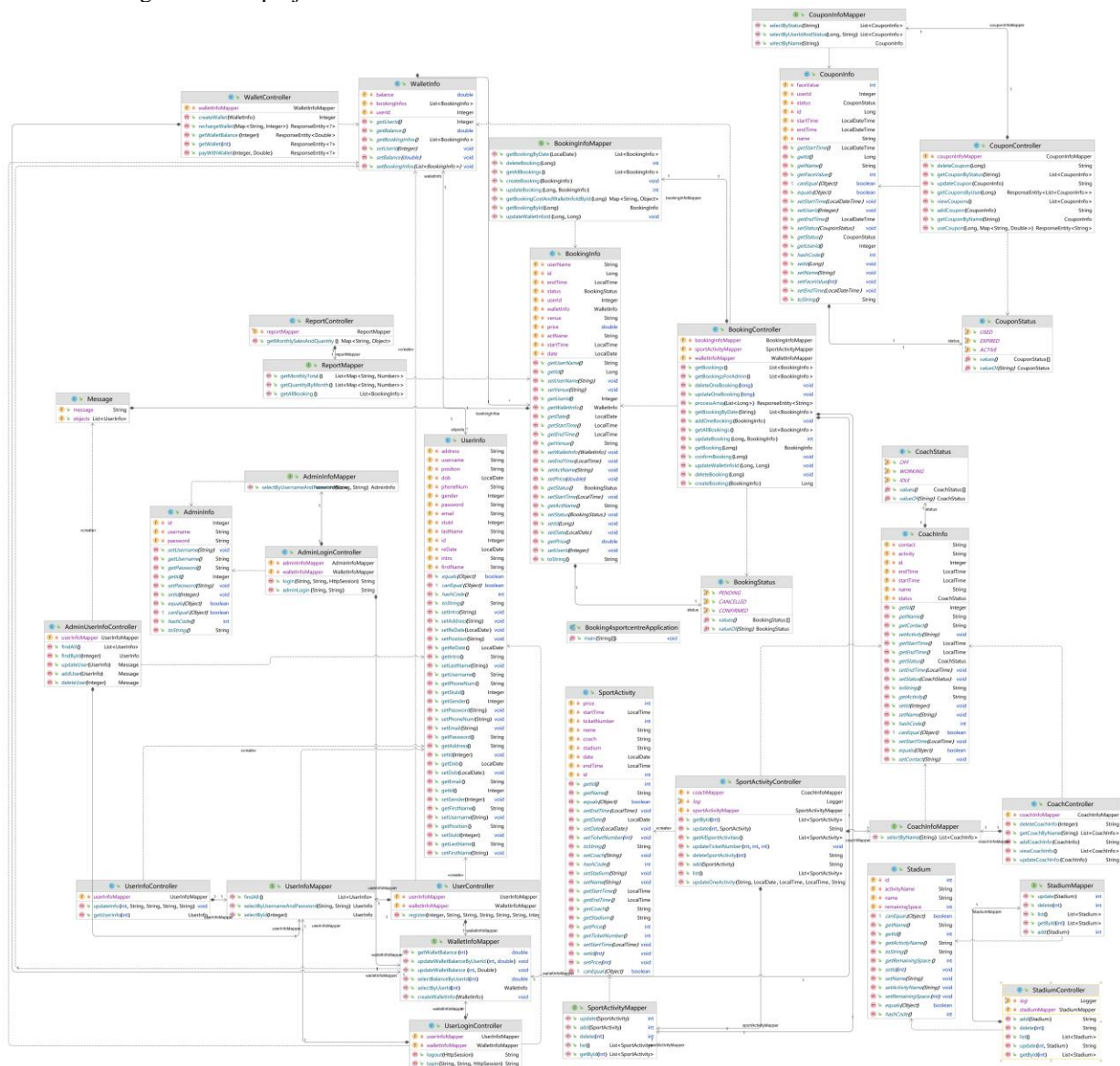
Appendix 7: User view and modify user information activity diagram

## 10. Administrator login activity diagram



Appendix 8: Administrator login activity diagram

## 11. Class Diagram of our project



## Appendix 9: Class Diagram of our project

## 12. Integration testing details

✓ StadiumControllerTest (com.group6.boc 465 ms)	
✓ testAddStadiumInfo()	428 ms
✓ testGetStadiumById()	17 ms
✓ testUpdateStadiumInfo()	7 ms
✓ testGetStadiumInfo()	7 ms
✓ testDeleteStadiumInfo()	6 ms

## Appendix 10.1: Integration testing of StadiumController

✓	✓	CoachControllerTest (com.group6.bookl	520 ms
✓		testDeleteCoachInfo()	430 ms
✓		testAddCoachInfo()	47 ms
✓		testUpdateCoachInfo()	18 ms
✓		testGetCoachByName()	18 ms
✓		testViewCoachInfo()	7 ms

Appendix 10.2: Integration testing of CoachController

✓	✓	SportActivityControllerTest (com.grou	564 ms
✓		testUpdateActivity()	455 ms
✓		testAddStadiumInfo()	9 ms
✓		testFindAll()	51 ms
✓		testGetActivityById()	4 ms
✓		testUpdateTicketNumber()	15 ms
✓		testListActivity()	6 ms
✓		testDeleteSportActivity()	6 ms
✓		testAddActivity()	8 ms
✓		testAddCoachInfo()	10 ms

Appendix 10.3: Integration testing of SportActivityController

✓	✓	BookingControllerTest (com.group6.bo	559 ms
✓		testGetBookings()	480 ms
✓		testConfirmBooking()	37 ms
✓		testCreateBooking()	6 ms
✓		testGetBookingByDate()	6 ms
✓		testGetBooking()	14 ms
✓		testUpdateBooking()	11 ms
✓		testDeleteBooking()	5 ms

Appendix 10.4: Integration testing of BookingController

✓	✓	AdminUserInfoControllerTest (com.grou	556 ms
✓		testAddUser()	475 ms
✓		testFindAll()	43 ms
✓		testUpdateUser()	23 ms
✓		testFindByld()	9 ms
✓		testDeleteUser()	6 ms

Appendix 10.5: Integration testing of AdminUsersInfoController

### 13. Complete acceptance testing results

ID	Description	Tester	Role	Result	Note
1	Log in	Yunjia.Qi	Administrator	Pass	Use 'admin1', 'password1'
2	Add a sport activity	Yunjia.Qi	Administrator	Pass	none
3	Amend a sport activity	Yunjia.Qi	Administrator	Pass	Alert "Are you sure you want to delete this sport activity?"
4	Delete a sport activity	Yunjia.Qi	Administrator	Pass	none
5	Search a sport activity by name	Yunjia.Qi	Administrator	Pass	none
6	Add a new coach	Rui.Lin	Administrator	Pass	none
7	Modify coach information	Rui.Lin	Administrator	Pass	none
8	Delete existing coach	Rui.Lin	Administrator	Pass	Alert "Are you sure you want to delete coach Mingyuan.Li?"
9	View all bookings and search by name	Rui.Lin	Administrator	Pass	none
10	Modify user booking information	Rui.Lin	Administrator	Pass	none
11	View registered users information	Rui.Lin	Administrator	Pass	none
12	Update user information	Rui.Lin	Administrator	Pass	none
13	View statistical report	Rui.Lin	Administrator	Pass	none
14	Log out	Rui.Lin	Administrator	Pass	none
15	Attempt to login with wrong username and password	Rui.Lin	Administrator	Pass	Use invalid data: '222', '000'
16	Sign up	Zimeng.Li	User	Pass	none
17	Log in	Zimeng.Li	User	Pass	none
18	Update Personal information	Zimeng.Li	User	Pass	none
19	Charge	Zimeng.Li	User	Pass	none
20	View sport activities	Zimeng.Li	User	Pass	none
21	Select one activity to book	Zimeng.Li	User	Pass	none
22	Search by coach and stadium on selected activity page	Zimeng.Li	User	Pass	none
23	View agreement property before payment	Feiyang.Peng	User	Pass	none
24	Pay for selected activity directly	Feiyang.Peng	User	Pass	none
25	Add activities to cart	Feiyang.Peng	User	Pass	Alert "Successful"
26	Select all bookings in cart and make payment	Feiyang.Peng	User	Pass	Alert "The total is ¥20, are you sure you want to pay?"
27	Views all bookings and filter by different status	Feiyang.Peng	User	Pass	none
28	Cancel existing bookings	Feiyang.Peng	User	Pass	Alert "Are you sure you want to cancel this booking?"
29	Log out	Feiyang.Peng	User	Pass	none
30	Attempt to pay with insufficient wallet balance	Feiyang.Peng	User	Pass	Display "Please charge before payment."
31	Attempt to pay with insufficient coupon balance	Feiyang.Peng	User	Pass	Display "Insufficient coupon balance, please use another coupon."

### Appendix 11: Complete acceptance testing results



14. Individual contribution from

## CPT202 Assignment 2

### Individual Contribution for Group Report and Presentation

**Group Number: 6**

Name	ID Number	Contribution (%)
1. Jinai Ge	2035693	12.5%
2. Mingyuan Li	2145618	12.5%
3. Chengze Liu	2142808	12.5%
4. Yixuan Wang	2145543	12.5%
5. Yue Wang	2141845	12.5%
6. Fuyu Xing	2143763	12.5%
7. Baiyan Zhang	2142564	12.5%
8. Zhenyang Zhao	2142074	12.5%

**Signed by all members:**

邢富玉 張柏岩 葛金艾 王悦

王艺璇 刘成泽 赵桢扬

李明远