CPT202 2023/2024 Semester 2

Final Report for Software Engineering Group Project

2023/2024 Semester 2

Project C: Online Booking Service for a Sports Centre

2024 May 24

Access our website at: http://8.149.129.247:8080/index.html

Access our project on GitHub at: https://github.com/Sushang-Li/CPT202_Group_6

Group number: 6

Student Name: Mingyuan Li

Student ID: 2145618

Contents

1. Introduction

As university sports activities diversify, schools are building new facilities and offering more booking options. However, the increase in sports activities leads to an increase in miscellaneous information in software, making the traditional booking system inefficient and not user-friendly [1]. Additionally, high data loads and inefficient code will affect the capacity and responsiveness of the system [2]. To address this problem, our project aims to develop a **lightweight booking system** with efficient software design and an intuitive user interface (UI) design to enhance user-friendliness and overall effectiveness.

The final solution we developed is an integrated sports center booking system based on the Model View Controller (MVC) architecture. It includes the user system and administrator system, consisting of 10 subsystems and a reasonable caching system for tracking user login identities (see Appendix A for detailed architecture). Specifically, our software encompasses essential functionalities, including user registration, login, sports activities booking, booking management, and administrative oversight of all core elements like user information, sports activities details, and bookings. **As the group leader**, my contributions are pivotal in creating and configuring the initial project framework, guiding team members in using Git for collaborative development, determining the UI design framework, and ensuring timely progress throughout the project. Beyond my leadership role, I also made significant technical contributions, completing five product backlog items (PBI) facilitating the functionalities as an administrator for **1.) viewing coach information, 2.) adding coaches, 3.) managing coaches, 4.) checking booking information, 5.) maintaining booking information.**

This report is structured into five sections as follows: The **Software Development Process** explains the Scrum framework utilized in projects, the **Software Design** describes the design and implementation of five PBIs independently developed The **Change Management** illustrates the logic for handling requirement changes within my responsibilities in the project and strategies for managing potential requirement changes across the entire project in future work. The **Legal, Social, Ethical, and Professional Issues** explores the impact of these five issues on the project and presents possible solutions we consider. The **Conclusion** summarizes the personal insights and knowledge acquired in software engineering through the project and identifies potential improvements for future iterations.

2. Software Development Process

Scrum is an agile framework that enhances the responsiveness and flexibility of software projects [3]. The main advantages of using Scrum include improved team collaboration, enhanced adaptability to changes, and a strong focus on delivering high-quality product increments [4]. These benefits make Scrum an ideal choice for our project, ensuring that we can commit to development tasks on schedule and respond effectively to requirement changes. Therefore, in our project, we utilized the five Scrum events to complete three sprints, which encompassed all core functionalities. Adhering to the project rules, each team member independently completed full-stack development tasks. Moreover, the enhanced communication facilitated by Scrum systematically helped us manage these conflicts and synchronize progress. To better illustrate our Scrum development process, the following details our five Scrum events.

2.1 Product Backlog Grooming

According to Francisca [5], Product Backlog Grooming is a continuous process throughout the project, involving the identification of PBIs, prioritization, effort estimation, and the removal or updating of outdated PBIs. In our project, grooming began with a 4-hour meeting group leader organized before development to identify the necessary PBIs based on customer specifications and system users' needs.

The meeting identified the forty core PBIs (Appendix B) and generated a detailed document listing each PBI's name, priority, sprint, and owner. By following the clear objectives outlined in the grooming document, task conflicts can be minimized, and active participation can be enhanced.

In subsequent weeks, we conducted weekly online meetings, in which we regularly updated our grooming document in terms of the owners and the necessity of PBIs based on active communication. For example, following the suggestions from teammates, I decided to remove the "Admin Blocked Users" PBI from the original division due to time constraints. Also, I extended new PBI "Admin Add Coaches" and "Admin Maintain Coaches" PBI as my assigned work mainly focused on the coach model.

2.2 Sprint Planning

Sprint planning is a crucial process where the team defines the goals for the upcoming sprint [6]. In our three sprints, we determined the PBIs to be completed for the current sprint based on the Assessment 1 PBI files. We also iterated the acceptance criteria of these PBIs through communication to enhance the quality of the software design. Through planning before each sprint, time waste and task conflicts are significantly reduced, enhancing team collaboration efficiency and systematic project management.

During the planning of three sprints, my PBI files assisted in selecting the current PBIs and refinement of tasks. Specifically, during Sprint 1 planning, I added the coach's working status attribute after consulting with the team member responsible for sports activity design. In Sprint 2 planning, I defined administrator's the booking information needs after discussions with booking design teammates.

2.3 Sprint Execution

As stated by Andrean [7], Sprint Execution involves the systematic development and testing of planned tasks within a set time frame. In our sprints, we independently developed the PBIs based on their descriptions and acceptance criteria and validated the development through unit tests. In the end, effective execution provides timely progress feedback for the project.

As it comes to the execution of my personal PBIs, for example, I followed four stages for each of them: Development, Testing, Review, and Demo. Specifically, after completing the back-end logic development and UI, I conducted unit tests and compared the results with the PBI file. Finally, I used screenshots to quickly validate the table design with the front-end architecture teammate.

2.4 Sprint Review

As Necmettin explains [8], Sprint review involves presenting completed work, gathering feedback, and adjusting the PBI to ensure continuous delivery of high-value software. In our project, the review process is completed through weekly online meetings or WeChat messages. Since the team uses GitHub for project synchronization, members can easily review the completion of PBI locally and provide accurate feedback based on commit records. With frequent reviews and feedback, we can quickly adjust development details at the end of sprints to achieve the desired designs.

In the review, as the group leader, I coordinated the PBI results of team members, collected their feedback, and provided valuable insights to correct exceptional PBIs. For instance, in the "Admin Maintain User Information" PBI, it was evident that the modified user information did not align with the actual updated values. After receiving my feedback, the owner promptly fixed the bug. Additionally, based on my teammate's feedback, I identified a display error when there were no bookings in "Admin Add Coaches".

2.5 Sprint Retrospective

Sprint retrospective involves the team reflecting on the past sprint to identify successes, challenges, and opportunities for improvement in future sprints [9]. In our project, during the weekly meeting, we discussed the successes and drawbacks of the development process and proposed potential improvement plans collaboratively. This ensures smoother future sprints due to the technology stack overlap.

During the retrospective, the improvement plan I propose involves PBI time management, the use of third-party libraries, and adherence to code convention requirements.

3. Software Design

In the project, my software design and development involve administrative functionalities, encompassing five PBIs: **"Admin View Coach", "Admin Add Coach"**, **"Admin Maintain Coach"**, **"Admin View Bookings"**, and **"Admin Maintain Bookings"**. By interacting with the Coach and Booking systems, these PBIs enable administrators to throughout managing coach and booking information.

3.1 Design Overview and Justification

1. **Architecture Design**: To maintain consistency with the team, my part also employed the MVC as the architecture backbone. Specifically, I have fully designed the model, view, and controller architectures for the entire coach system, and increment the view and controller architectures for the booking system. This architecture design was chosen for its reusability (promotes code reuse), testability (powerful testing support), and scalability (modular structure supports growth).
2. **Module Design**: My backend module includes models, mappers, and controllers: models represent data structures, mappers handle data transfer between the database and controller, and controllers manage HTTP requests and responses. My frontend module consists of HTML, CSS, and JavaScript: HTML structures the content, CSS styles the appearance, and JavaScript adds interactivity. This design enhances maintainability and scalability. Refer to Appendix C for the class diagram.
3. **Database Design**: My database design includes two main tables: `booking_info` and `coach_info`. Additionally, I implemented a foreign key relationship between `coach_info` and `sport_info` to ensure the integrity and validity of the sports activity associated with a specific coach.
4. **API Design**: My API design adheres to consistent naming conventions (prefixed with `/api`), utilizing GET for retrieving data and POST for creating or updating resources. These APIs serve to manage front-end requests for coach and booking information.
5. **Error Handling Design**: For the flow control in both the controller and JavaScript, I meticulously designed precise error messages, facilitating rapid diagnosis and resolution of issues. This approach enhances the overall efficiency and effectiveness of the debugging process.
6. **Testing Design**: My testing focuses on coach and booking API methods, including unit testing to verify components and integration testing to ensure seamless interaction between different units.
7. **UI Design**: To ensure consistency with the administrative UI designed by my teammates, I adopted the same sidebar design. The coach and booking tables' UI utilize Bootstrap and jQuery, using these libraries for critical functionalities like sorting and searching while ensuring responsiveness.

3.2 Implementation Overview and Justification

1. **Module Implementation**: My module structure uses the standard Spring Boot directory structure: The `CoachController`, `CoachMapper`, and `CoachModel` that I designed are organized into their respective folders: controller, mapper, and model. All front-end modules are located within the static directory. This implementation aligns with the project convention, which effectively lowers access barriers for team members, ensuring a more inclusive and collaborative development environment.
2. **Database Implementation**: The creation of tables and initialization of columns in the database is facilitated by the `@Entity` annotation, with primary keys defined using `@Id` (see code example in Appendix F). This approach ensures a consistent database structure across all team members.
3. **API Implementation**: The APIs are implemented using `@GetMapping` and `@PostMapping` annotations. Each API is designated with a specific name that corresponds to its functionality.

4. **Error Handling Implementation**: Error detection in the back-end is determined by the return value of the mapper which interacts with the database. Besides, in the front-end, errors are captured using the `.catch` method. This approach ensures robust error handling and improves system reliability.
5. **Testing Implementation**: My testing process utilizes `SpringBootTest`. For unit testing, each method is tested using the `@Test` annotation of JUnit. For integration testing, the interactions between multiple units are examined using the `@SpringBootTest` annotation.
6. **UI Implementation**: The UI implementation involves popup pages and tables. CSS is used to define the page appearance like page scope and location, HTML is used to structure the content of tables, and JavaScript is responsible for data population and managing interactions with the back-end.

3.3 Example 1: Comprehensive Analysis of "Admin View Booking"

The "Admin View Booking" enables the administrator to view all bookings, providing a comprehensive overview of the current booking status. The detailed PBIs can be checked in Appendix D.

Logical flow: The administrator clicks the "Booking" button, prompting the web to send a GET request to the controller. The controller then calls the `getAllBookings` in the mapper, which queries the database for all data in the `booking_info` table. The database returns the booking list to the mapper, which encapsulates the booking information in a Java list and sends it back to the controller. The controller forwards this list to the web application, displaying all booking information or a "No bookings available" message if none exists. This workflow is illustrated in Figure 1's sequence diagram.

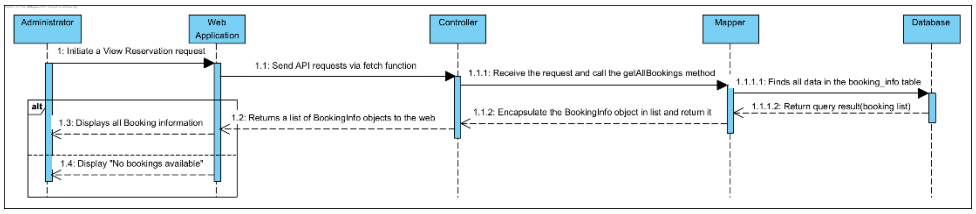


Figure 1: Sequence Diagram of "Admin View Booking"

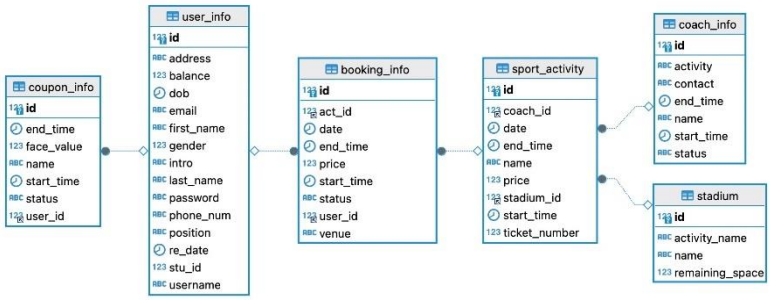


Figure 2: ERD of our Project

Database: The `booking_info` table stores booking details, including `id` (unique identifier), `act_id` (activity identifier), `date` (booking date), `start_time` (start time), `end_time` (end time), `price` (cost), `status` (current booking status), `user_id` (user identifier), and `venue` (activity location). The SQL statement `SELECT * FROM booking_info` is used in the mapper to retrieve all booking information. More details can be checked in the Entity Relationship Diagram (ERD) shown in Figure 2.

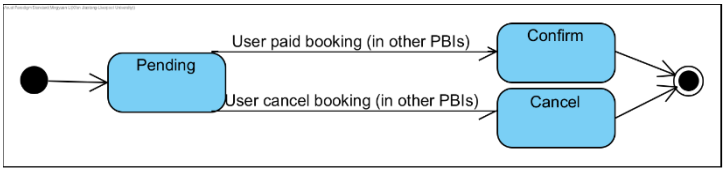


Figure 3: State Machine Diagram of Bookings

States of the Objects: For each booking, an enumeration class is employed to define its state, comprising three statuses: Pending, Confirm, and Cancel (Figure 3). Upon booking generation, it enters the Pending state. The Pending transitions to Confirm after payment. The modify button remains available in both the Pending and Confirm. However, once it into Cancel, the administrator can no longer modify it.

UI: I implemented the booking table UI using `dataTables` of Jquery. Each row corresponds to an individual booking entry, with column headings and filter buttons provided at the top of the table. A search bar is in the upper right corner for querying information, and the lower left corner displays the number of entries on the current page. When there is no booking information, a message will span the entire row, informing that there is currently no booking. This UI prototype is provided in Appendix J.

3.4 Example 2: Comprehensive Analysis of "Admin Maintain Coach"

The "Admin Maintain Coach" enables the administrator to modify or delete coach information and provide full management rights over coaches. The detailed PBIs can be checked in Appendix E.

Logical flow: When a change coach request is made, the web application sends an update coach API request to the controller, which calls the `updateCoachInfo` method. This method uses the coach mapper to find and update the coach information in the database. Then the mapper returns an update number indicating the status. The controller subsequently creates the update message according to the number and transmits it back to the web. Next, execute a method with analogous logic to query the current coach and update the coach table. The above workflow is illustrated in the sequence diagram shown in Appendix H. Moreover, the deletion function follows the same pipeline, as detailed in Appendix I

Database: The coach_info table includes `id` (Primary Key), `activity`, `contact`, `start_time`, `end_time`, `name`, and `status`. Implement the status using an enumeration class, encompassing the states: Working, Idle, and Off. The SQL `SELECT * FROM coach_info WHERE name = #{name}` is used in the mapper to find a coach by name. Check more details in the ERD shown in Figure 2.

States of the Objects: Each coach has three states corresponding to Working, IDLE, and OFF. Initially, each coach is in the OFF state and can be transitioned to the IDLE state, enabling subsequent entry into the Working state. These state transitions can be viewed in Appendix G.

UI: The coach table employs the same UI framework as the booking table. At the far right of each row, there are "Modify" and "Delete" buttons for the administrator to manage coaches. Clicking "Modify" opens a page with input fields pre-filled with the current information of the selected row. After making changes, click "Update Coach" to save the update and redirect back to the table page. Clicking "Delete" opens a confirmation page that highlights the coach's name. If there are no coaches, a message spans the entire row, indicating the absence of coaches. The UI prototype is in Appendix K and Appendix L.

4. Change Management

According to Ahmed Mubark [10], in the Scrum of our project, effectively managing requirement changes is crucial for ensuring the successful delivery of desired outcomes. These changes persist throughout the entire duration of our project, affecting every team member's contribution.

4.1 Handling Requirement Changes

For the project, we plan to develop an advanced report-generation module in the initial phase. This module will offer administrators comprehensive statistics on sports activity sales, current venue status, coach status, and user behavior analysis. However, during development, we discovered that the time and database support for creating the complex report generation module were limited due to constraints in third-party library support. Therefore, we changed the requirement to develop a basic monthly reports module focusing on statistics on sales and tickets.

For my part, in designing the "Admin Maintain Coach" PBI, my initial intention was to enable the modification of multiple coach records simultaneously. However, during the development phase, I encountered the challenge of ensuring data consistency due to the atomicity of update operations. Thus, I adjusted the requirements to permit only single-record updates per operation.

4.2 Managing Potential Requirement Changes

As the project goes on, we will continue to explore more solid password recovery features. Currently, we have implemented a basic password retrieval function using user IDs. However, users may need a higher level of security; we therefore plan to enhance the password recovery feature by developing an email-based retrieval method. Meanwhile, a retrieval code provided during registration can also be an alternative as the backup in addition to email retrieval.

5. Legal, Social, Ethical, and Professional Issues

We considered legal, social, ethical, and professional issues in our project, identifying potential solutions.

Legal Issues: Legal issues refer to the laws and policies that must be adhered to during the development and delivery of a project. In our project, we encountered data security challenges, especially regarding the protection of sensitive user information such as phone numbers in their profiles. To address this, MySQL's built-in encryption features can be used to secure our data. Additionally, in the collection and processing of user data, we can provide users with a clear privacy statement and obtain their consent.

Social Issues: Social issues affect individuals' well-being and resource access when using software. In our project, we encountered the issue of user accessibility obstacles, particularly considering the elderly staff among our users. To solve this, we designed an intuitive and user-friendly UI (e.g., larger fonts) to minimize the cognitive load and reliance on memory for our users.

Ethical Issues: Ethical issues related to ethical standards and codes of conduct in the development and use of projects. In our project, we faced a user privacy issue due to the need to collect book information and payment details. A possible solution can be implementing data minimization practices to ensure that only essential information is collected and providing users with full control over their data.

Professional Issues: Professional issues refer to challenges related to the skills, conduct, and standards expected in a professional setting. In our project, we faced development efficiency issues as some team members were not sufficiently familiar with the Spring Boot framework. To address this, we can train all team members with a high-quality full-stack development online tutorial at the beginning.

6. Conclusion

In conclusion, the software we developed is a sports center booking system based on the MVC architecture. I played a pivotal role in establishing the project framework, guiding Git usage, designing the UI, ensuring project progress, and completing key PBIs. Through this project, I have learned the process of conducting reliable and efficient collaborative development using Scrum, understand the simplicity and clarity of the MVC architecture, and gained experience in full-stack development. Additionally, I have gained an understanding of designing systems with low coupling and high cohesion, and the method of adhering to standardized code conventions.

In the future, I aim to increment multi-line data modification capabilities in my work to improve management efficiency. Furthermore, we will optimize the password retrieval function by incorporating email retrieval or retrieval codes to enhance system security. We will also design a responsive UI for optimal user experience on various devices. Besides, to address legal and ethical issues, we will implement advanced data encryption methods and show users the personal information collection list.

References

[1] T. Q. Sheng, K. Subaramaniam, and A. S. Bin Shibghatullah, "OwnBook: Developing Mobile Application For Badminton Court Booking System," in *Proc. 2023 IEEE 3rd Int. Maghreb Meeting of the Conf. on Sci. and Techn. of Automatic Control and Comp. Eng. (MI-STA 2023)*, Benghazi, Libya, 2023, pp. 419-424. doi: 10.1109/MI-STA57575.2023.10169223.

[2] Anshika, Kumar, A., Thakur, S., Thakur, K., & Singh, G. (2023). Enhanced Efficiency and Customer Engagement Through the Online Shopping Arcade with Vendor Recommendation and Pre-Booking System. In Proceedings of the *4th IEEE 2023 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS 2023)* (pp. 532-537). Greater Noida, India. doi:10.1109/ICCCIS60361.2023.10425280.

[3] A. M. Turcios-Esquivel, E. G. Aviles-Rabanales, and G. Sayeg-Sanchez, "Use of technology and Scrum as an agile methodology to favor the development of balanced teamwork enrichment skills in higher education subjects," in *IEEE Global Engineering Education Conference (EDUCON)*, Salmiya, Kuwait, 2023, pp. American University of Kuwait (AUK); et al.; IEEE; IEEE Education Society; IEEE Region 8; KIPCO, May 2023, doi: 10.1109/EDUCON54358.2023.10125262.

[4] H. Fawareh, E. F. Al-Qbelat, and M. N. Al-Refai, "The Use of SCRUM Methodology for Final Year Undergraduate Project During Corona Pandemic," in *2022 13th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 2022, pp. 162-166, doi: 10.1109/ICICS55353.2022.9811142.

[5] F. Ribeiro, A. L. Ferreira, A. Tereso, and D. Perrotta, "Development of a grooming process for an agile software team in the automotive domain," in *Advances in Intelligent Systems and Computing*, vol. 745, Naples, Italy, 2018, pp. 887-896, doi: 10.1007/978-3-319-77703-0_86.

[6] R. Mesquita, R. Nascimento, L. Souza, and M. Lucena, "Using the iStar framework for planning and monitoring sprints in scrum projects: An experience report," in *CEUR Workshop* Proceedings, vol. 2490, Salvador, Brazil, 2019.

[7] A. Taufiq, T. Raharjo, and A. Wahbi, "Scrum evaluation to increase software development project success: A case study of digital banking company," in *2020 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Virtual, Depok, Indonesia, 2020, pp. 241-246, doi: 10.1109/ICACSIS51025.2020.9263235.

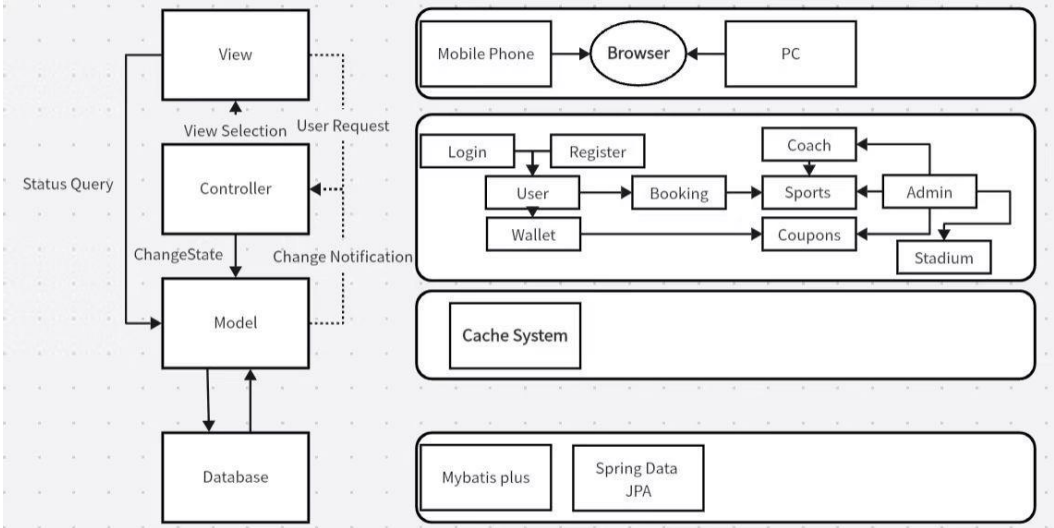
[8] N. Ozkan, S. Bal, T. G. Erdogan, and M. S. Gok, "Scrum, Kanban or a Mix of Both? A Systematic Literature Review," in Proceedings of the *17th Federated Conference on Computer Science and Intelligence Systems (FedCSIS)*, Sofia, Bulgaria, 2022, pp. 883-893, doi: 10.15439/2022F143.

[9] J. Werewka and A. Spiechowicz, "Enterprise architecture approach to SCRUM processes, sprint retrospective example," in Proceedings of the *17th Federated Conference on Computer Science and Information Systems (FedCSIS)*, Prague, Czech Republic, 2017, pp. 1221-1228, doi: 10.15439/2017F96.

[10] A. M. Alsalemi and E.-T. Yeoh, "A survey on product backlog change management and requirement traceability in agile (Scrum)," in *2015 9th Malaysian Software Engineering Conference (MySEC)*, Kuala Lumpur, Malaysia, 2015, pp. 189-194, doi: 10.1109/MySEC.2015.7475219.

Appendix

A System Architecture



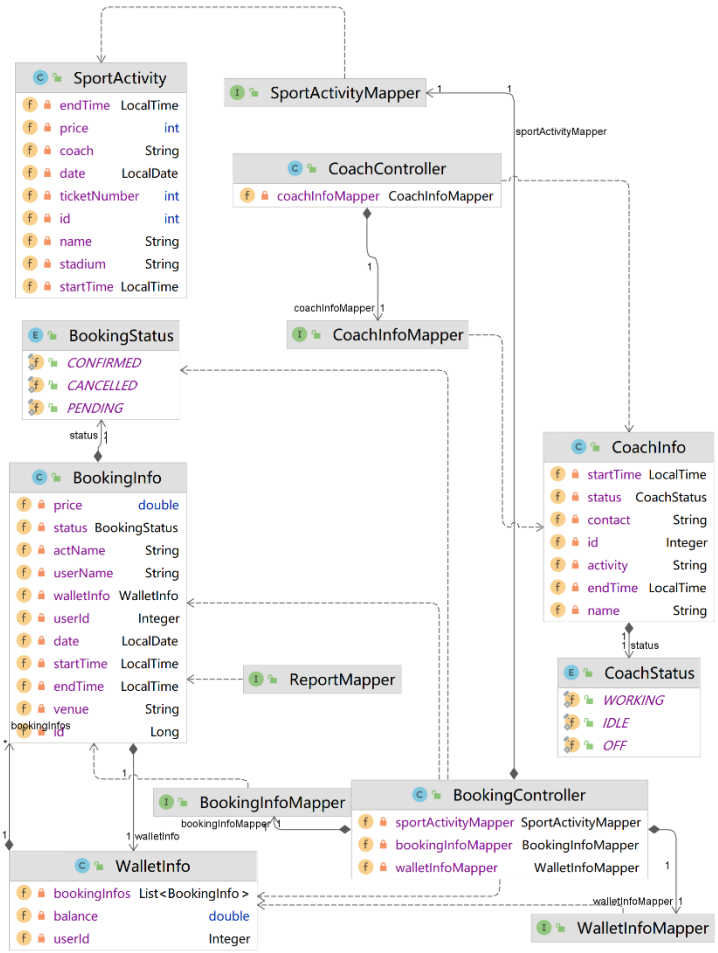
Appendix A: System architecture diagram

B PBI Grooming Result

PBI #	PBI Title	Priority	Owner	Sprint
1	Display Booking Details on Payment Page	1	Fuyu Xing	1
2	Display Payment Method on Payment Page	1	Fuyu Xing	1
3	Display Wallet Balance on Wallet Page	2	Fuyu Xing	2
4	Administrators Manage Coupons	2	Fuyu Xing	2
5	User Views Coupons	3	Fuyu Xing	3
6	Admin View Coach	1	Mingyuan Li	1
7	Admin Add Coach	1	Mingyuan Li	1
8	Admin View Booking	2	Mingyuan Li	2
9	Admin Maintain Coach	2	Mingyuan Li	2
10	Admin Maintain Booking	3	Mingyuan Li	3
11	Display Sports on Homepage	2	Yue Wang	2
12	User Views Sports Information	1	Yue Wang	2
13	User Pays Tickets on Booking Page	1	Yue Wang	1
14	User Searches Information on Booking Page	2	Yue Wang	2
15	Display Stadium Property on Booking Page	3	Yue Wang	3
16	Admin View User Information	2	Yixuan Wang	2
17	Admin Maintain User Information	2	Yixuan Wang	2
18	Admin Delete User Information	2	Yixuan Wang	2
19	Admin View Monthly Ticket Quantity Report	3	Yixuan Wang	3
20	Admin View Monthly Revenue Report	3	Yixuan Wang	3
21	Users View Personal Information	1	Jinai Ge	1
22	Users Modify Personal Information	2	Jinai Ge	1
23	User Homepage Carousel of Popular Daily Activities	3	Jinai Ge	3
24	Users View the Reservation Record	2	Jinai Ge	2
25	Users Cancel the Reservation Record	2	Jinai Ge	2
26	Admin View Sports Activities	1	Chengze Liu	1
27	Admin Maintain Sports Activities	1	Chengze Liu	2
28	Admin Maintain Sports Stadiums	1	Chengze Liu	2
29	Logout Functions for Users and Administrators	2	Chengze Liu	3
30	Admin View Sports Stadiums	2	Chengze Liu	3
31	User View Bookings	1	Baiyan Zhang	1
32	User Delete Bookings	1	Baiyan Zhang	1
33	User Pay One Unpaid Booking	2	Baiyan Zhang	2
34	User Pay All the Bookings	2	Baiyan Zhang	2
35	User Add One Booking to Shopping Cart	3	Baiyan Zhang	3
36	User Login Function	1	Zhenyang Zhao	2
37	Admin Login Function	1	Zhenyang Zhao	2
38	User Agreement Function	2	Zhenyang Zhao	2
39	User Registration Function	1	Zhenyang Zhao	3
40	Balance Recharge Function on Wallet Page	3	Zhenyang Zhao	3

Appendix B: The Latest of the PBI Grooming Table

C Class Diagram of My Work



Appendix C: Class Diagram of My Work

D Detailed PBIs of "Admin View Booking"

Appendix D: Product Backlog Item of "Admin View Booking"

PBI #	8
Title	Admin View Booking

Creation Date	25 March 2024	**Status**	New	**Sprint**	1
Priority	1	**Effort**	12h	**Business Value**	3/5

Story

As an administrator, I want to be able to view all the bookings, to know the current situation of booking, and then effectively manage the booking records.

Description

1. The system should provide a dedicated page where administrators can access a comprehensive list of all bookings.
2. The list of bookings should include pertinent details such as their price, status, activity name, user information, wallet information, date, start and end times.

3. There should be a sorting functionality to organize the list of bookings based on criteria such as their names.
4. There should be a searching functionality to find a certain booking based on the searching content.

Acceptance Criteria

1. Given that I am a valid administrator, when I am on the administrator homepage, then I can access the "Booking Information" page by clicking on the “Booking” button on the side of the homepage.

Scenario 1: There **exist** booking information:

- Given that I in the "Booking Information" page, when there exists valid booking information in the database, then I should see a list of all existing bookings displayed on the page.
- Given that I am viewing the list of bookings, when I click on the "Sort by Activity" option on the side of the page, then the list should be rearranged based on the activities of the booking, making it easy to check bookings for the same activity.
- Given that I am viewing the list of bookings, when I click on the "Sort by Status" option on the side of the page, then the list should be rearranged based on the bookings 's status.
- Given that I am viewing the list of bookings, when I click the search bar on the top of the list, then I should be able to search for specific bookings based on their names.

Scenario 2: There is no booking information:

- Given that I am a valid administrator, when I navigate to the "Booking Information" page in the system and there is no booking information, then I should see “There is no booking information yet. Please add coach first. ” on the page.

E Detailed PBIs of "Admin Maintain Coach"

Appendix E: Product Backlog Item of "Admin Maintain Coach"

PBI #	9
Title	Administrator Maintain Coaches

Creation Date	25 March 2024	**Status**	New	**Sprint**	2
Priority	2	**Effort**	24h	**Business Value**	4/5

Story

As an administrator, I want to modify the coaches, so that I can manage the coaches information to support the activities.

Description

1. The system should provide a dedicated page where administrators can access a comprehensive list of all coaches.
2. There should be a button on the far right of each row of coaches that will take the administrator to the “Modify Coaches” page.

3. On the “Modify Coaches” page, the administrator can modify the full names, contact information, working slots, working status and the specific activities they are qualified to coach.
4. The administrator can click “Cancel” to return after entering “Modify Coaches” page.
5. There should be a “Delete” button on the far right of each row of coaches that will take the administrator to the “Delete Confirm” page.
6. On the “Delete Confirm” page, the administrator can see a highlight of the coach’s name. And confirm button to proceed.
7. The administrator can click “Cancel” to return after entering “Delete Confirm” page.

Acceptance Criteria

1. Given that I am a valid administrator, when I am on the administrator homepage, then I can access the " Coaches Information" page by clicking on the “Coaches” button on the side of the homepage.

Scenario 1: There **exist** coaches:

- Given that I am a valid administrator, when I navigate to the "Coaches" page, then I should see a list of all existing coaches displayed on the page.
- Given that I am on the "Coaches" page, when I click the modify button on the far right of each row of order, then I will see a popup page: “Modify Coaches” and show some input blocks (pre-filled) that are the elements I can modify.
- Given that I am in the “Modify Coaches” page, when I input all the content I want to change, then I can click the “Confirm” button to save my modify and proceed.
- Given that I am on the "Coaches" page, when I click the delete button on the far right of each row of order, then I will see a popup page: “Delete Confirm” and see the highlighted coach name that I select.
- Given that I am in the “Delete Confirm” page, when I click the “Confirm” button, then I will delete the coach I select and back to "Coaches" page.

Modify Scenario 1: The modification is in the **correct** form.

- Given that I am on the “Modify Coaches” page, when I confirm my modification by clicking the “Confirm” button, then I will be back to the " Coaches" page and check that my change is active.

Modify Scenario 2: The modification is in the **incorrect** form.

- Given that I am on the “Modify Coaches” page, when I confirm my modified content by clicking the “Confirm” button, then there will display “Input form is wrong, please fill in the correct information.” nearby the “Confirm” button and let me try to fill in the required field.

Scenario 2: There is **no** coach:

- Given that I am a valid administrator, when I navigate to the "Activity Bookings" page, then I should see “There is no coaches yet. Please add coaches.” on the page.

2. Given that I am on the “Modify Coaches” page, when I click the “Cancel” button, then the system will close the “Modify Coaches” page and back to the "Coaches" page.
3. Given that I am on the “Delete Confirm” page, when I click the “Cancel” button, then the system will close the “Delete Confirm” page and back to the "Coaches" page.

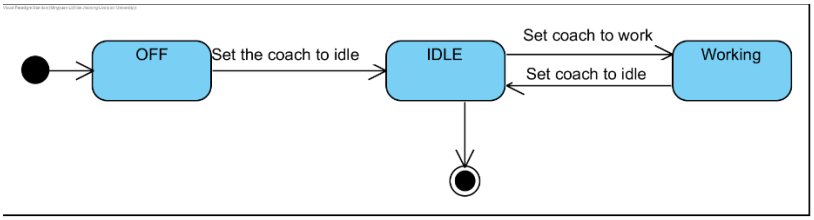
F Code Screenshot of Coach Table Implementation

```java
@Entity
@TableName("coach_info")
@Data
public class CoachInfo {
    no usages
    @Id
    @TableId(type = IdType.AUTO)
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    no usages
    private String name;
    no usages
    private String contact;
    no usages
    private LocalTime startTime;
    no usages
    private LocalTime endTime;
    // CoatStatus: WORKING, IDLE, OFF
    no usages
    @Enumerated(EnumType.STRING)
    private CoachStatus status;
    // Maybe need join query
    no usages
    private String activity;
}
```

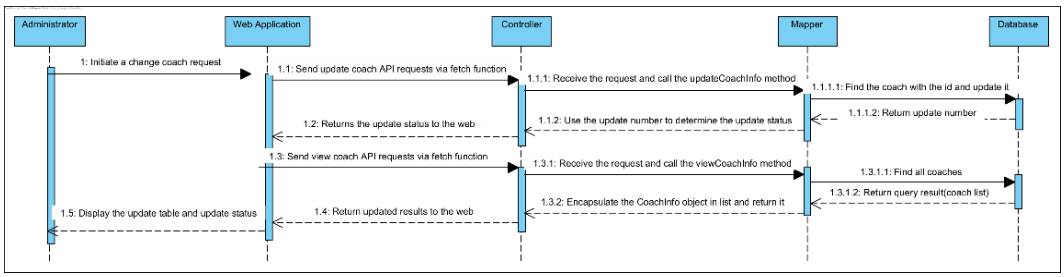
Appendix F: Code Screenshot of Coach Table Implementation

G State Machine Diagram of Coach



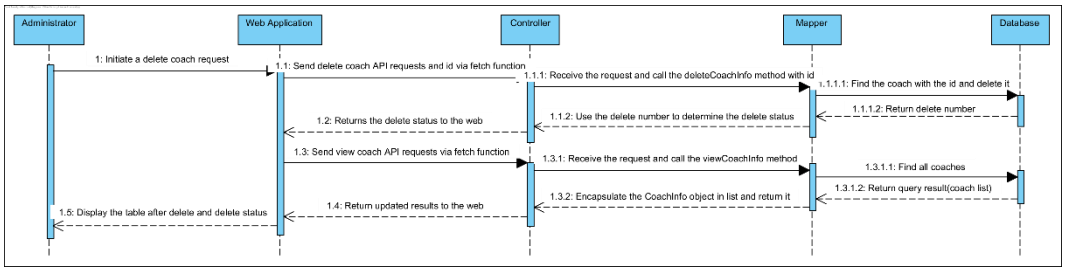
Appendix G: State Machine Diagram of Coach

H Sequence Diagram of Modifying Coach Information



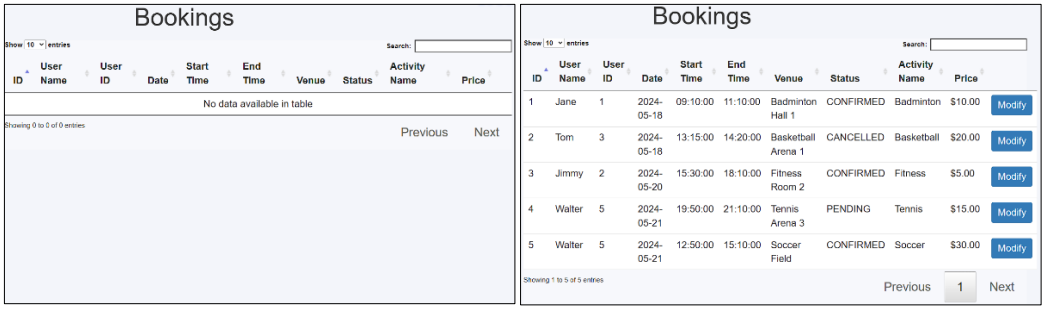
Appendix H: Sequence Diagram of Administrator Modifying Coach Information

I Sequence Diagram of Delete a Coach



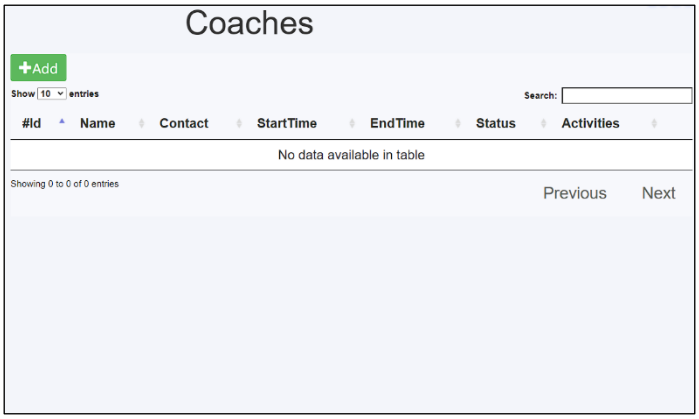
Appendix I: Sequence Diagram of Delete a Coach

J UI Prototype for "Admin View Booking"



Appendix J: UI Prototypes for "Admin View Booking": No Booking (Left) and Existing Booking (Right)

K UI Prototype for "Admin Maintain Coach"



Appendix K: UI Prototypes for "Admin View Booking" in the Absence of Coaches

Appendix L: UI Prototypes for "Admin View Booking" Including Its Popup Pages