**CPT202 2023/2024 Semester 2**

**Final Report for Software Engineering Group Project**

<Online Booking Service for a Sport Centre>

*<2024/5/18>*

Group number: <Group 5>

Student Name:

Student ID:

Project URL: http://120.26.207.141:8080/Login&Register.html

Contents

[**Introduction** 3](#_Toc166966539)

[**Software Development Process** 3](#_Toc166966540)

[**Software Design** 4](#_Toc166966541)

[**Change Management** 7](#_Toc166966542)

[**Legal, social, ethical, and professional issues** 7](#_Toc166966543)

[**Conclusion** 8](#_Toc166966544)

[**References** 8](#_Toc166966545)

[**Appendix** 9](#_Toc166966546)

# **Introduction**

As the university sports center grows busier, the manual handling of booking requests has become time-consuming and prone to errors, leading to double-bookings and schedule conflicts [1]. Therefore, as shown in figure 1, we developed a university sport booking system that not only provides fundamental online booking capabilities through date and time but also offers convenient order querying and management, along with customizable user homepages. Additionally, our project features a user-friendly administrative interface that simplifies the management of activities, items, and instructors, as well as the querying of orders and members. The system includes visualized statistical data that can be filtered and printed directly, aiding administrators in analysis and reporting. In this project, my primary responsibility was to handle the registration and login functions, ensuring that only internal university personnel could access the app, which was essential to maintain security and exclusivity within our campus community.

图示

描述已自动生成

Figure 1. System Flow Process

L

This report will cover the software development process, detailing our use of the Scrum framework, individual contributions to software design, the handling of change management, and some ethical considerations, which provides the project's progression from concept to implementation.

# **Software Development Process**

In our software development process, we implemented the agile Scrum framework, which significantly facilitated project progress and effective communication. Our team consisted of nine developers, each acting as a product owner to ensure the accuracy and completeness of requirements. Additionally, one member took on the role of the team leader, further streamlining our development efforts.

Before starting development, we carried out a requirement analysis and recorded a product backlog document. During the development process, we followed a weekly sprint cycle and successfully completed five iterations, culminating in the delivery of a fully sports booking system. In each sprint cycle, we followed four Scrum activities: Sprint Planning, Sprint Execution, Sprint Review, and Sprint Retrospective.

Utilizing the Scrum framework in our software development process, our team experienced significant benefits in fostering collaboration. Through regular sprint activities, each member was actively involved in the project, ensuring effectively communicating and timely error handing.

For example, during the first sprint, our team held a sprint planning meeting on March 25th, where we analyzed the PBIs in detail. We decided to prioritize establishing the database and designing the web prototype, which lay the foundation for the essential functionalities for both users and administrators. Then we assigned development roles among the team: four members focused on the user side and five on the administrator side. Developers then specified the PBIs each needed to complete during this sprint, after which collaborated on negotiating the design style of the web pages. Subsequently, we discussed several uncertain issues, such as whether our login API should use mobile phone number registration or campus information registration. Finally, the team leader summarized our sprint goals, the PBIs each developers needed to complete in this sprint, the integration of frontend and backend, and the design of the database and prototype.

Subsequently, the sprint execution phase took place from March 26th to April 1st. We initially planned to conduct daily scrum meetings every two days to promptly address any issues and prevent slow progress. However, due to the developers' other assignments and courses, we adjusted the scrum meeting schedule to every Monday combining with sprint meeting. As most of the developers were novices, we provided some time for learning frontend and backend development. Thus, we canceled the first meetings to give everyone sufficient time to learn. At our first scrum meeting, developers expressed uncertainty in the frontend page design style. Therefore, we coordinated to standardize the color scheme, layout, and the design of some reusable components to ensure consistency across the web.

Finally, we held conduct the Sprint Review and Sprint Retrospective. During the Sprint Review, we evaluated the development outcomes and found that we had only implemented a portion of the Product Backlog Items (PBIs) during the first sprint. In the Sprint Retrospective, we reflected on why we did not meet all our objectives. One reason was that our developers, being amateurs, required additional time to learn web development knowledge. Additionally, due to unfamiliarity with the process, there was a poor estimation regarding the number of PBIs that could be realistically handled. To address this, developers who were already familiar with the development process conducted a crash course for the others, helping everyone to become more accustomed to web development.

Overall, the Scrum agile development method facilitated a rapid and structured approach to our software development, enabling efficient communication and preventing delays in addressing issues. It effectively planned our time allocation, with each team member actively participating in the development process.

# **Software Design**

In the project, we employ the MVC architecture for our software design. Utilizing Spring's MVC framework, the backend is organized into four main distinct layers (shown in Appendix 1): The Model layer, represented by Pojos, mirrors our database schema and encapsulates the data model; the Mapper layer maps these models to the database; the Service layer manages business logic; and the Controller layer handles incoming requests from clients and delivers responses. We also structured several layers for enhanced functionality. The utils layer contains utility classes like JwtUtils for JSON Web Token management. The exception layer includes a GlobalExceptionHandler for centralized error management, while the interceptor layer contains the LoginCheckInterceptor to secure and authenticate user sessions effectively. In addition, The Result class is used to standardize responses for API interactions. It includes an integer response code (1 for success, 0 for error), a descriptive message string (msg), and an optional data object. The class provides methods for generating successful responses (success()) and error responses (error(String msg)), ensuring consistency of responses in the project.

The entire system is segmented into several key modules. By decoupling each module, the database design adheres to the 3rd Normal Form to ensure efficient data handling and integrity (shown in Appendix 1).

In this booking system development, my main responsibility lies in the "Account Management Module" in yellow region of figure 1.

The PBIs I handled are shown in Appendix 2 which include front-end page design, back-end functionality implementation, and database integration of registration and login. PBI002 and PBI003 will be used as the example to introduce the detailed design process.

图示

描述已自动生成图形用户界面

描述已自动生成

Figure 2. PBI002 and PBI003

**PBI002—User Registration**

**Design goal**

The PBI is to restrict app access to members only, ensuring secure and exclusive usage. University verification is set to ensure only staff and students can register, after which members can log in and utilize the app.

**Design Process**

Initially, I considered using phone numbers and verification codes for managing entries in the member table. However, to enhance security and ensure that only internal students and staff could access the system, I introduced the login\_information table instead, which ensures that only users with verified IDs and passwords can access or be added to the member table. This approach greatly increases the security of our system by simulating calls to the central campus information database API to verify registration information.

**Implementation**

**UI Design**

I will introduce the page design of PBI002 and PBI003 together.

In the initial prototype of the webpage design, the registration and login interfaces were separated, as shown in appendix 3. During the implementation, I realized that this design was not only complex, involving five different pages, but also lacked centralized functionality, which may troubles users. Therefore, in the implementation phase, we integrated these functions into a single interface to facilitate ease of use.

As shown in figure 2(b) and 2(c), the page combines both registration and login functionalities. All registration fields are centrally placed in the right-hand section for easy visibility and interaction. On the left-hand side, a welcoming graphic of the Sport Centre is designed to attract users. This side also includes a navigational “Login” button allowing users to switch to the login page if they already have an account, enhancing user experience by providing a seamless transition between logging in and registering.

The sliding effect is implemented by binding JavaScript event listeners to "Register" and "Login" buttons. When clicked, listener trigger jQuery animate() that animate the sliding effect.

**Database**

**login\_information entity**: It includes the attributes id, password, real\_name, gender, and identity, with id serving as the primary key. It serves as authorization to verify and authenticate users, ensuring only verified IDs and passwords can access the system.

**member entity**: It includes the attributes id, alias, personalized\_signature, phone, creation\_date, update\_date, and member\_id, with member\_id serving as the primary key. The foreign key is the id from the login\_information entity. This entity stores customized member information and tracks updates and entry creations, effectively separating user authentication from user profile management.

**Function**

When a user enters their ID and password and clicks the "Registration" button, a PUT request is sent. On the server side, the request is handled by a method mapped with the @PutMapping annotation. This method first checks the member table for the existing user ID. If the ID is already registered, it returns error information "ID already registered". Next, the method queries the login\_information table to check if the ID exists. If it does not, it returns error "ID does not exist". If the ID is found, the method then verifies whether the corresponding password matches the one provided. If there is a match, registration is successful, and it returns success, recording the ID in the member table. If the password does not match, it returns error "Incorrect password".

**PBI003—Member Login**

**Design goal**

The PBI is to implement a login feature that enables registered members to access the booking system.

**Design process**

In PBI002, I use campus information for identity verification. Similarly, after registration, members will continue to use their campus credentials to log in, reducing the inconvenience of having to create and remember new ID and passwords.

**Implementation**

**Database**

It uses member and login\_information entities introduced above.

**Function**

When a user enters their ID and password and clicks the "Login as user" button, a POST request is sent. On the server side, the request is handled by a method mapped with the @PostMapping annotation. This method first checks the member table for the provided ID. If the ID does not exist, it returns an error with the message "User ID does not exist". Next, the method queries the login\_information table to retrieve the record corresponding to the provided ID. If the ID is found, it then verifies whether the provided password. If there is a match, the login is successful, and the method call JwtUtils to issue a JWT token and returns a success result along with the JWT to frontend. If the password does not match, it returns an error with the message "User ID or password is incorrect". The JWT will expire after 12h.

# **Change Management**

In our Scrum project, we prioritized agility and responsiveness in managing Product Backlog Items (PBIs). Emphasizing the Agile principle of adapting to change over rigidly following a plan, we adapted PBIs based on ongoing feedback and evolving requirements. For example, we removed the PBI designed to create different administrative interfaces and functions based on administrator priority, due to its complexity and the time required for implementation. Our iterative development approach allowed for frequent reassessment and refinement of PBIs, ensuring that changes were effectively communicated and collaboratively understood across the team. This strategy-maintained alignment and facilitated timely adjustments in our development process.

My PBIs also underwent some changes. Initially, I planned to separate the verification and customization processes into three distinct pages as shown in Appendix 3. However, I realized that creating three separate pages was time-consuming and potentially redundant, which could lead to user frustration. Therefore, I decided to integrate these two PBIs into one page as shown in figure 2(b) and figure 2(c).

# **Legal, social, ethical, and professional issues**

From the legal standpoint, the university sports booking system needs to handle members’ information and booking data. To ensure compliance with data protection and privacy laws [2], we employ protocol, obtaining users' explicit consent during registration and login as shown in figure 3(a), If the user does not check the "I have understood and agreed to all terms" checkbox, they will not be able to proceed to the next step.

To secure user data, we utilize SQL precompiling and parameterized queries to prevent SQL injection attacks, safeguarding our database as shown in figure 3(b).

For preventing unauthorized access, we issues JWT (figure 3(c)). JWT provides a simple way to send secure and self-contained information as a JSON object to confirm information authenticity and integrity [3]. Issued JWT will expire after 12 hours.

Through the measures mentioned above, we provide the utmost protection of user data privacy and security at a legal level, ensuring that our system complies with all relevant laws and regulations.



Figure 3. Legal, social, ethical, and professional issues

# **Conclusion**

The application of Scrum Agile Methodology significantly streamlined our software development cycle, allowing us to complete tasks efficiently and meet deadlines effectively. Immersing myself in this process provided me with invaluable experience and a deep understanding of what comprehensive software development truly entails.

I have mastered using Spring Boot to simplify the creation, execution, debugging, and deployment of web. Additionally, the MVC architecture has established a robust framework for organizing projects, clearing the integration of independent front-end, back-end, and database components, which has enhanced my ability to manage complex software architectures efficiently.

Some improvements can be implemented to solid the booking app. The current system often experiences slow responses during large, popular sporting events that accommodate multiple reservations. Therefore, implementing scalable solutions such as load balancing and optimizing database transactions is necessary to ensure the system can handle high concurrency effectively. Additionally, implementing a message queuing system can help manage task distribution and service communication more efficiently, which allows asynchronous processing of tasks that helps in decoupling components and smoothing out peak loads.

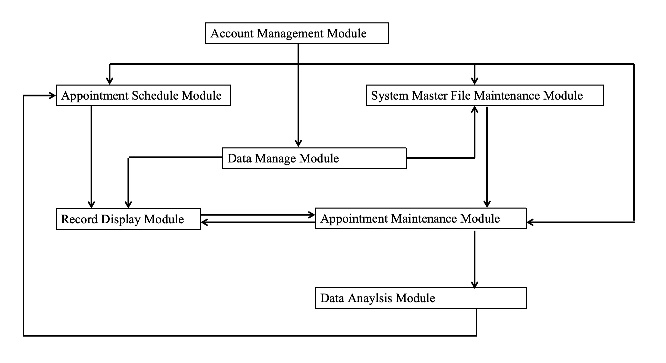
# **References**

1. K. Kowalczyk, A. M. Kowalczyk, A. Zwirowicz-Rutkowska, and M. Bednarczyk, “E-Tourism Based on Geoinformation Applications for Seniors: Requirements and Design Guidelines,” Quaestiones Geographicae, vol. 42, no. 3, pp. 101-113, Sep. 2023, doi:10.14746/quageo-2023-0026.
2. Custers, Bart, et al. *EU personal data protection in policy and practice*. Vol. 29. The Hague, The Netherlands: TMC Asser Press, 2019.
3. Jones, Michael, John Bradley, and Nat Sakimura. *Json web token (jwt)*. No. rfc7519. 2015.

# **Appendix**

Appendix 1

Diagrams for sports activity booking system.



Backend Layers

System Architecture

图示, 表格

描述已自动生成

ER Diagram

图示

描述已自动生成

Use Case

Appendix 2

PBI sequence diagrams for sports activity booking

图示

描述已自动生成

User Registration

图形用户界面, 文本, 应用程序

描述已自动生成

Member Login

图片包含 图形用户界面

描述已自动生成

Administrator Login

图形用户界面, 应用程序

中度可信度描述已自动生成图示, 日程表

描述已自动生成

Setting Personal Information

Protocol Consent

图示

描述已自动生成

Log Out

Appendix 3

Prototyping of Account management module

图形用户界面, 应用程序

描述已自动生成

Prototyping