# Online Booking Service for a Sport Centre


***Group 5***

| | |
|---|---|
| *Jingxin Zha* | *2141508* |
| *Liangyuting Zhang* | *2142253* |
| *Xian Qian* | *2142302* |
| *Xin He* | *2145653* |
| *Xujia Ning* | *2141856* |
| *Yi Ni* | *2142298* |
| *Yutong Hu* | *2145001* |
| *Zhenyu Dai* | *2143271* |
| *Zimeng Li* | *2141564* |

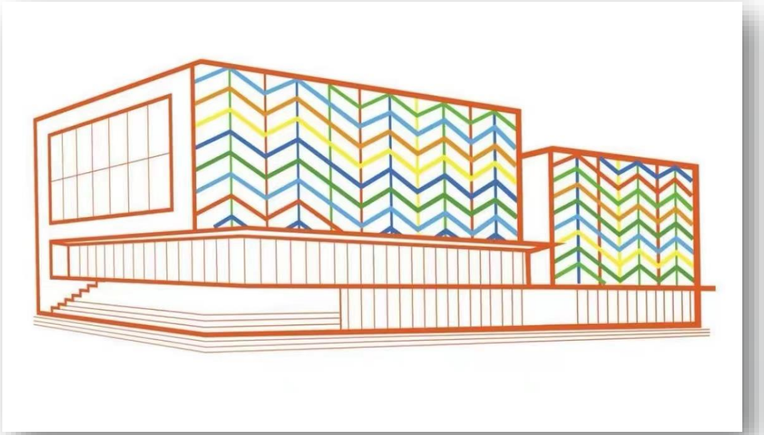
# Overview /CONTONTS

- RART·ONE -

# Introduction

01

# Background



The development of ***a web-based online booking service system*** for a university's sport center

## Aims of Our Project

- A ***high-quality, efficient, user-friendly and safe*** system that enables ***convenient managements*** and ***booking services***
- ***All customer specifications***

# Background

## User Characteristics

**Staff and Student Users**

*unique booking needs*

- A convenient and efficient booking process
- Real-time access to activity information
- Clear activity descriptions
- Photos for informed decisions
- Secure payment methods with order confirmation

**Sport Center Administrators**

*expect an intuitive interface*

- Access statistics data
- Manage activities, items, members, instructors and orders

## Project Risks

***Technical Risks:*** bugs in software development, security vulnerabilities [1]
***Time Risks:*** time constraints
***Requirements Risks:*** inadequate understanding of requirements, changes in requirements
***User Acceptance Risks:*** not satisfied with the functionality or performance of the system
***Security Risks:*** users' personal information leakage, payment information leakage [2]
***Compliance Risks:*** non-compliance with law [1]

- RART·TWO -

# Architectural Design

02

# *Requirements Influence Architectural Design*

- Several **key requirements** influence the architectural design of our web-based sport center system:

**Reliability**

**Performance**

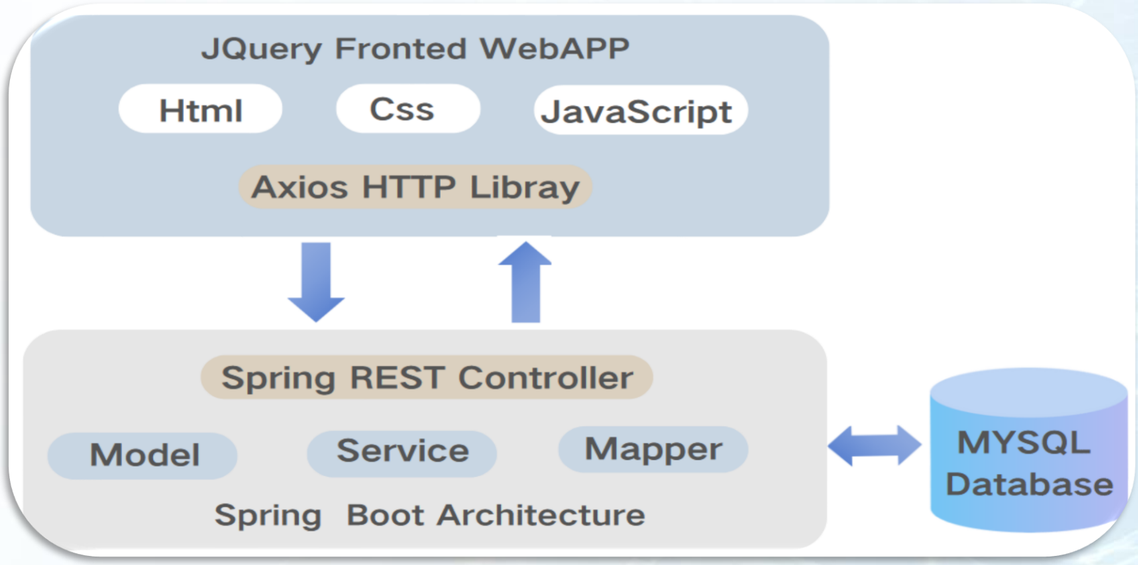
**Usability**

**Security**

# Overview of Architectural Design & Frontend System

**Architecture Components**

**Frontend:** HTML, CSS, JavaScript
**Backend:** Spring Boot [3]
**Database:** MySQL




**Frontend Technologies & Justification**

**HTML, CSS, JavaScript**

**JQuery**

Managing stateful user interactions

Enhances **scalability and maintainability** of the frontend [4]

**Ajax**

Promise-based **HTTP** client

Handling backend requests more efficiently

Request and response interception

**Streamlining error handling and improving code organization**

# Backend System

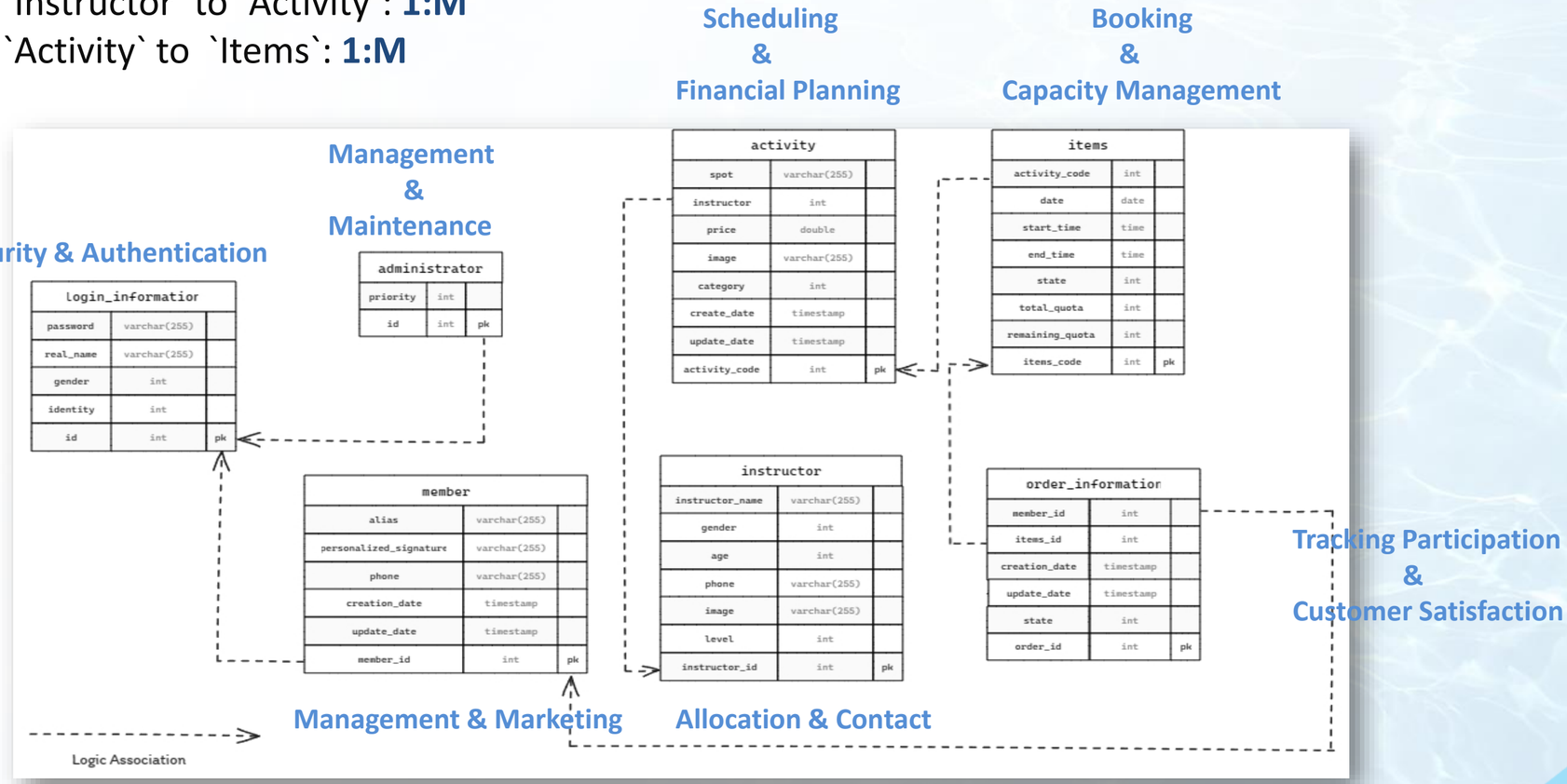
**Backend Technologies & Justification**

- **Spring Boot**
  - Box Functionalities
  - **Simplifies** the **Development** of New Services
- **MyBatis Plus**
  - Perform Database Operations
  - Customize SQL Queries
  - **Enhance Data Access** with Increased **Precision**
- **Lombok**
  - Automatically Generate Standard Code
  - Reducing  Manual Coding
  - Enhancing Code **Readability and Maintainability**
  - Minimizing the Verbosity of Java Applications
  - Reducing Repetitive Code
  - Enhances **Code Clarity**
- **JWT**
  - Authentication and Authorization
  - **Robust Security Measures [5]**

# High-level Database Design

**Entity Relationships**

`Login_information` to `Member` & `Administrator`: **1:1**
`Member` to`Order_information`: **1:M**
`Items` to `Order_information`: **M:1**
`Instructor` to `Activity`: **1:M**
`Activity` to `Items`: **1:M**

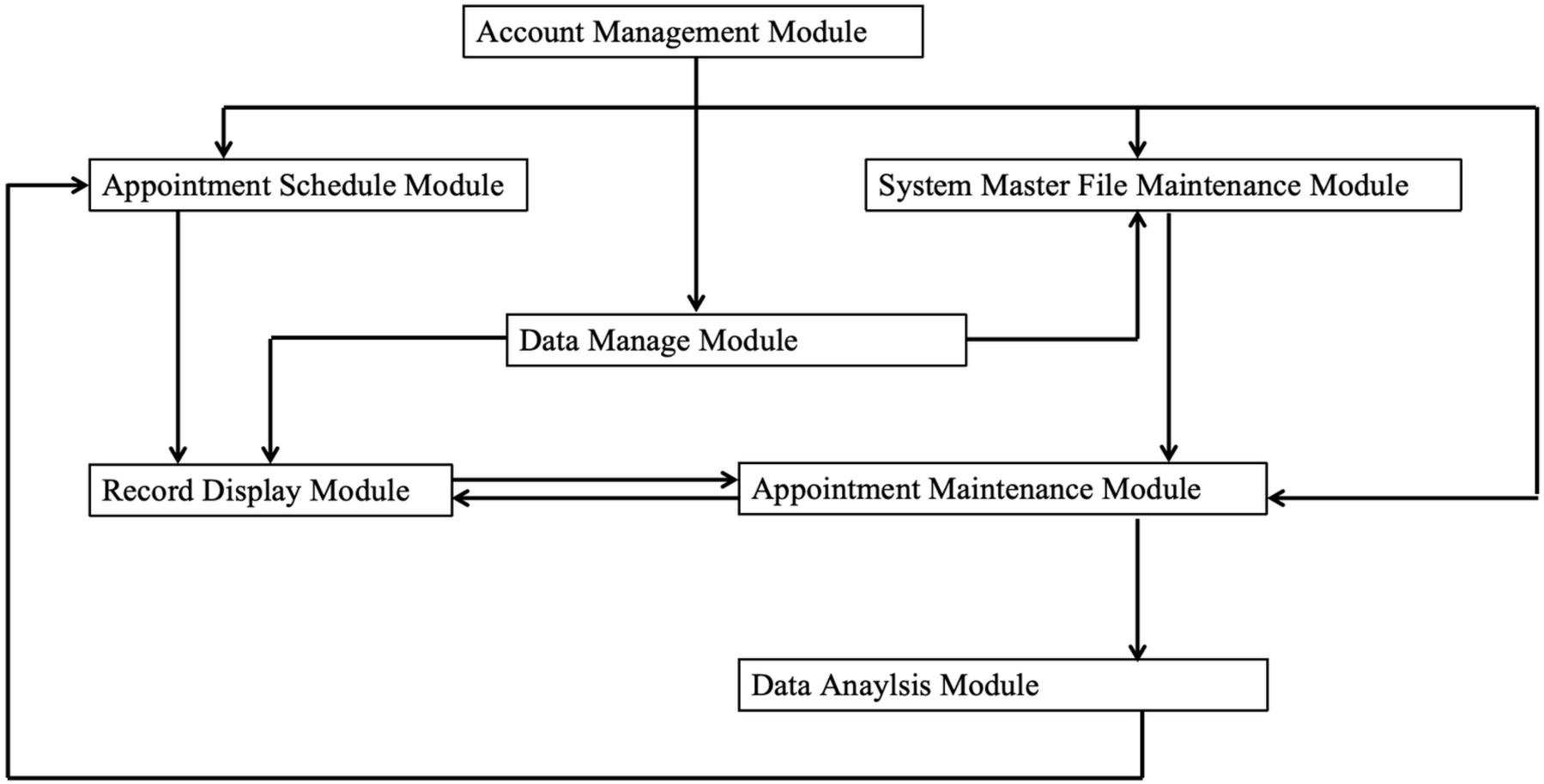


- RART·THREE -

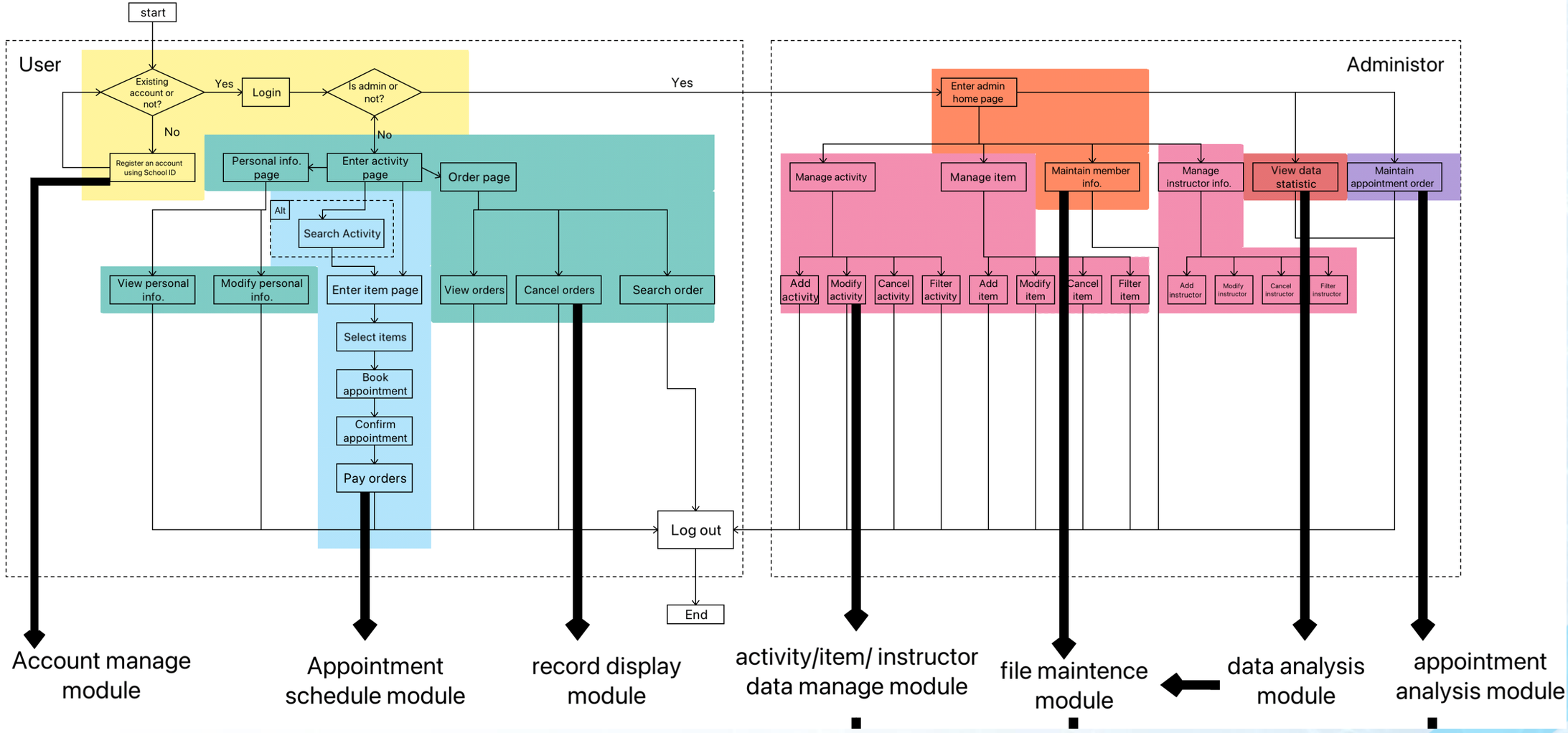
# Software Design

03

# Software Modules



# High-level Process Flow



# *Software Support Services*

**Database Related Services**

- Use **MySQL** to store information
- ORM Tools: Simplify Database Interactions with **MyBatis Plus**

**Security Related Services**

- JWT Implementation [6]
- Request Interception

**Webpage Navigation Related Services**

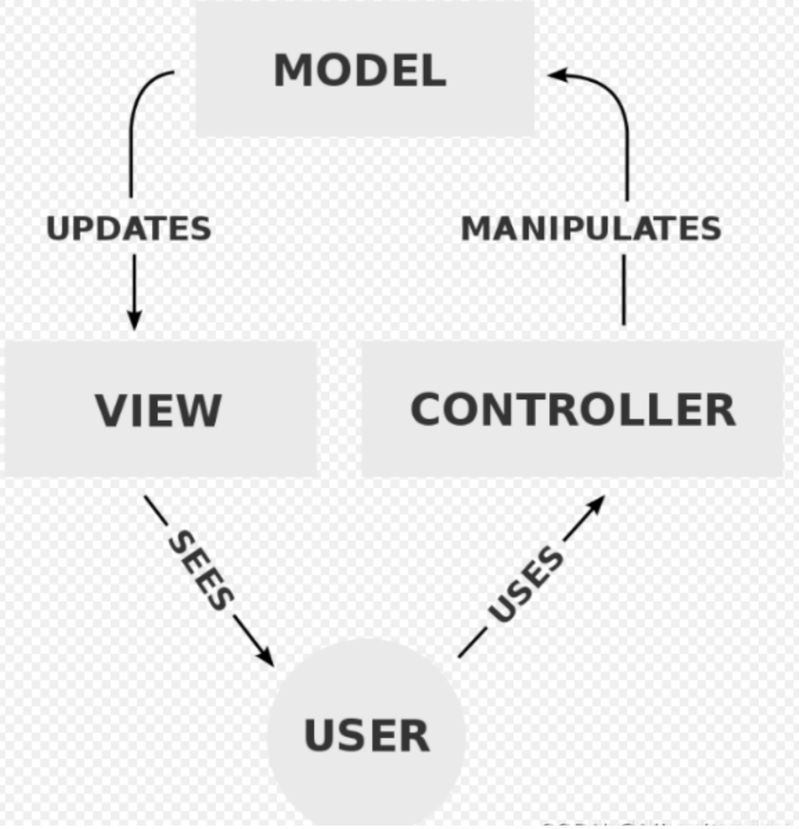
- Using **Spring Boot** to develop RESTful style Web services [7]
- Use **Spring Security** combined with **Spring MVC** to achieve access control

**Full Stack Development Services**

1. Front-End Management
2. Back-End Development
3. Database Management

# Coding Structure and Convention

## 1. MVC Architecture:




## 2. Layered Architecture:

**Service Layer:** combines the data operations provided by DAO to form a complete business process.

**Data Access Layer:** interact with data sources and perform CRUD operations.

**Dependency Injection:** automated injection of dependencies (e.g. database connections) into components that need them [8].

## 3. Convention:

Naming Conventions、 Commenting、 Error Handling

- RART·FOUR -

# Software Testing

04

# Unit Testing

All unit tests passed, confirming that individual components behave correctly under various scenarios.

## Process

1. Identify the unit of code.
2. Write test cases.
3. Prepare the test environment, configure the appropriate test framework and simulation tools.
4. Execute test cases, run test codes and record execution results.
5. Review test results.

## Test Data and Test Results

BookingAppApplicationTests (com.cpt202cw)
testAddItems()

```java
@Test//Verifies that new items can be added and that the system accepts the JSON format and returns an OK status.
public void testAddItems() {
    String url = "http://localhost:8080/items/addItems";
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(MediaType.APPLICATION_JSON);
    String jsonItems = "[{\"activityCode\":101, \"date\":\"2024-04-10\"}]";
    HttpEntity<String> request = new HttpEntity<>(jsonItems, headers);
    TestRestTemplate restTemplate = new TestRestTemplate();
    ResponseEntity<String> response = restTemplate.postForEntity(url, request, String.class);
    assertEquals(HttpStatus.OK, response.getStatusCode());
}
```

BookingAppApplicationTests (com.cpt202cw)
testGetItemById()

```java
@Test//Tests fetching an item by its ID to ensure it returns the correct item details.
public void testGetItemById() {
    String url = "http://localhost:8080/items/1";
    TestRestTemplate restTemplate = new TestRestTemplate();
    ResponseEntity<String> response = restTemplate.getForEntity(url, String.class);
    assertEquals(HttpStatus.OK, response.getStatusCode());
    assertNotNull(response.getBody());
}
```

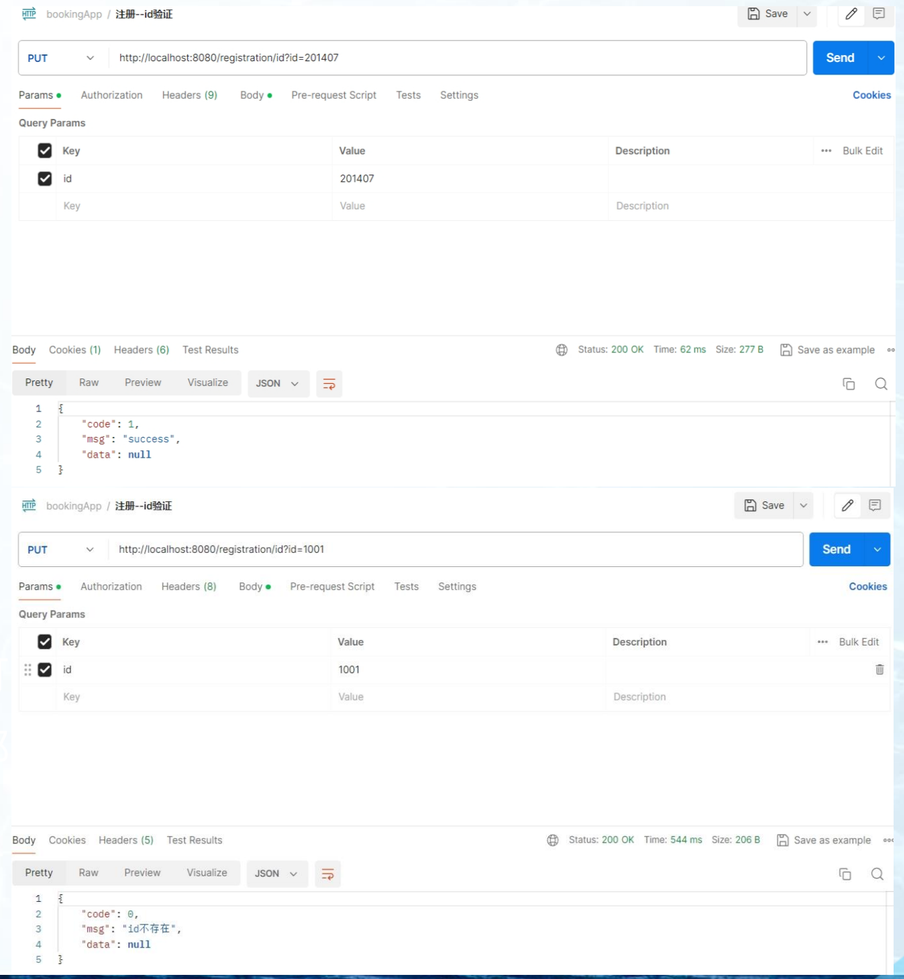
# Integration Testing

Integration tests validate the correct interaction between different modules, with no discrepancies found.

## Process

1. Identify the modules.
2. Write test cases.
3. Prepare the test environment.
4. Execute test cases, simulate interactions between different modules and record test results.
5. Check the test results.

## Test Data and Test Results



# Acceptance Testing

Acceptance tests confirm that the system meets the user requirements and behaves as expected under simulated real-world usage scenarios.

## Process

1. Determine acceptance criteria and acceptance test cases.
2. Write acceptance test cases.
3. Prepare the acceptance test environment.
4. Conduct acceptance test cases on behalf of the customers, simulate user operations and record test results.
5. Check test results.

### Acceptance Testing Example

Administrators' Acceptance for Data Statistics

### Objective

Confirm the system meets administrators' needs for data statistics.

### Test Case

Ensure that data statistics are accepted by the administrators.

### Input

System-generated statistical reports and visualizations.

### Expected Results

Data statistics are accepted by the administrators and meet their needs.

### Actual Results

As expected.

- RART·FIVE -

# Conclusion

05

# Conclusion

## Achievement

- **Comprehensive Web-based Online Booking System for the Sport Center**
- **Better Utilization of Scrum Framwork [9]**
- **Understanding of the Legal, Social, Ethical and Professional Considerations**
- **System Specification, System Design and Implementation, System Validation**

*Requirements engineering* *Front-end & back-end & database* *Testings*

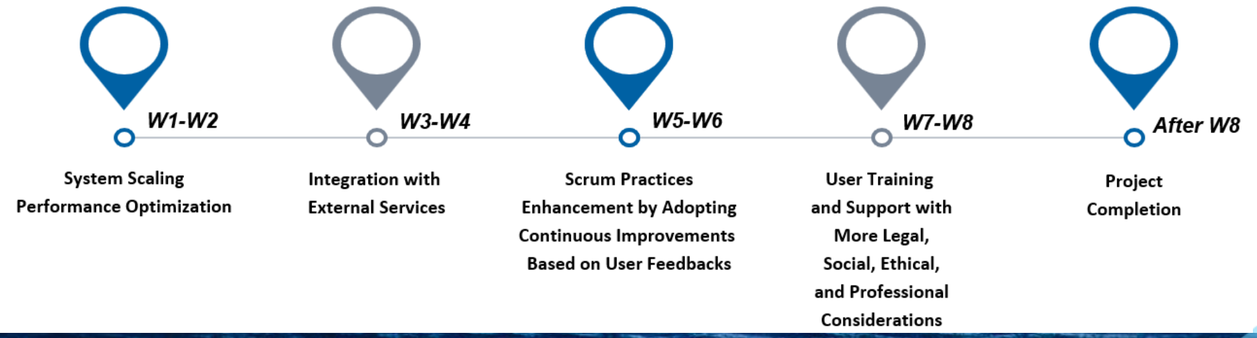
## Problems in Our Project

- **Device Presentation**
- **System Scalability and Flexibility**
- **Integration with External Services**

*Payment Tools*

## Future Works

**Related Timeline** (*Time is measured in weeks*) *1 Sprint = 2 Weeks*



# Reference List


[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] Z. Othman, K. A. F. A. Samah, N. H. M. Zain, and A. F. Zulkifli, “Optimizing Sports Center Recommendation System in Malaysia Through Content-Based Filtering Technique and Web Application,” *2023 IEEE 14th Control and System Graduate Research Colloquium (ICSGRC), Control and System Graduate Research Colloquium (ICSGRC)*, pp. 69-74, Aug. 2023, doi:10.1109/ICSGRC57744.2023.10215432.

[3] D. Zhao, Y. Liu, and B. Pei, “An Exploration of Architectural Design Factors with a Consideration of Natural Aspects Based on Web Crawling And Text Mining,” *Mathematics*, vol. 10, no. 23, pp. 4407-4407, Nov. 2022, doi: 10.3390 / math10234407.

[4] M. Jele and M. Dzie kowski, "The comparative analysis of Java frameworks: Spring Boot, Micronaut and Quarkus,” *Journal of Computer Sciences Institute*, vol. 21, no. 2544-0764, pp. 287-294, Dec. 2021, doi:10.35784/jcsi.2724.

[5] A. Bucko, K. Vishi, B. Krasniqi, and B. Rexha, “Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History,” *Computers*, vol. 12, no. 4, pp. 78-80, Apr. 2023, doi: 10.3390 / computers12040078.

[6] Y.Z. Li et al., "Research and Application of Template Engine for Web Back-end Based on MyBatis-Plus," *Procedia Computer Science*, vol. 166, pp. 206-212, Mar. 2020. doi:10.1016 / j. rocs. 2020.02.052.

[7] L.-Y. Kong et al., "Design and Realization of an Online Examination System Based on Maven Framework," *2022 3rd International Conference on Computer Science and Management Technology (ICCSMT)*, pp. 56-59, Jan. 2022, doi:10.1109 / ICCSMT58129.2022.00019.

[8] J. Zhao, "Application and Practice of MVC Architecture Pattern in On-the-job Internship Management System," *2022 International Conference on Networks, Communications and Information Technology (CNCIT)*, pp. 25-30, Feb. 2022, doi:10.1109 / CNCIT56797.2022.00013.

[9] S. Kalaivani, A. Senthilkumar, A. Prabhune, M. B. Durairaj, and S. S. Bhat, "Application of Scrum Framework and Low Code No Code Platform for Development and Implementation of In-Patient Electronic Visitor Management System to Optimise Hospital Operations,” *2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, pp. 1-6, Jan. 2024, doi: 10.1109/IITCEE59897.2024.10467670.


Thank You !