

**Project Title: Airline booking system**

Course code: CPS\_2232

Course Name: Data Structure

Student Group List:

Student Name:Zheng Qinyuan Student Number: 1194184

Student Name: Zhang Yuyang Student Number: 1194181

Student Name: Zhang Xue Student Number: 1163317

Student Name: Zhao Minghe . Student Number: 1196759

Student Name: Jiang Qiwei Student Number: 1196688

**Professor**: Dr Ken Ehimwenma, Ph.D.

Date:12/17/2023

**Abstract**

Our project presents the development of a web application for airline seat reservation and purchase, emphasizing a Model-View-Controller (MVC) architecture. The application offers a user-friendly interface for customers to search, select, and purchase seats for different classes: First class, Business, and Economy. The shopping process involves multiple steps, including flight search, seat approval by the airline manager, and secure payment processing. The system employs JavaServer Pages (JSP) as the View component, providing dynamic and responsive user interfaces. Additionally, the project utilizes the MyBatis framework for efficient database operations and incorporates a service layer to abstract business logic

1. **Introduction**

Modern travel demands efficient and user-friendly solutions for airline seat reservation and purchase. This project addresses this need by developing a robust web application utilizing a classic Model-View-Controller (MVC) architecture. The system streamlines the seat purchasing process into three key steps, ensuring a smooth and secure experience for users.

The initial step involves searching for available flights based on criteria such as plane name, departure location, destination, departure time, arrival time, and total seats. Customers can then select the desired seat type, with prices displayed dynamically. Before purchasing, the airline manager must approve the selected seats, ensuring a controlled and regulated booking process.

The second step focuses on the airline manager's role, who reviews and approves seats added by the admin. Only approved seats are made available for customer purchase, ensuring that the process adheres to regulatory standards. Once approved, customers can proceed to view flight information and move forward in the booking process.

The third and final step encompasses the customer's approval of the itinerary, leading to the payment page. Here, the total price is displayed, and upon payment confirmation, the seat is marked as sold. Utilizing JavaServer Pages (JSP) for dynamic content generation, the application provides a seamless user experience.

This project leverages technologies such as MyBatis for efficient database operations, adheres to the principles of a classic MVC architecture, and incorporates a service layer for encapsulating business logic. The subsequent sections delve into the methodology, code structure, and key components of the web application, offering a comprehensive understanding of the system's design and functionality.

The provided text describes two UML diagrams illustrating all classes in our project. The first diagram depicts relationships among Employee, Dept, and Job classes related to airport employee entries. The second diagram encompasses document and user interactions, enabling customers to input and confirm information for ticket reservations. Additionally, it illustrates associations, aggregations, and directed associations between plane classes (FeiJi, FeijiZuoWei, FeijiZuoWeiYuDing, FeijiPass) and users. The Notice class represents operational notifications across the entire airport. This summary captures the essence of the UML diagrams without including visual representations.

The interfaces UML diagram in our project defines interface specifications for various modules, facilitating loosely coupled communication between them. By standardizing methods, properties, and events, it achieves system scalability and version management. The diagram provides clear module contracts, enhancing the project's maintainability and security. For example the rainService provides the solutions for the case that the plane taking off facing the raining situation. And each interface related to the its Servicelmpl Classes. At the same time, these Servicelmpl Classes are also linked to Dao Classes, in which Classes and methods are created for use by Servicelmpl Classes.

1. **Related Works**

In recent years, various works have emerged in the field of computer science, each contributing a unique perspective and implementation details to the field of reservation systems. Although these references mainly focus on different fields, certain aspects resonate with the goals of our aviation oriented projects.

In terms of medical scheduling, Akshay et al. (2019) proposed an intelligent system that integrates IoT devices for efficient reservation management, which inspired our goal of simplifying the process of aircraft reservation management systems and provided inspiration for simplifying the processes in our system. And they introduced the concept of model checking in the context of air ticket booking, providing insights into validation and validation procedures crucial for ensuring the correctness of the booking system. Elizabeth Rani et al. (2021) demonstrated a remote laboratory with Moodle reservation system, which allows for remote reservation of laboratory usage time. This provides insights for designing booking systems that go beyond specific fields and also provides a reference for us to create HTML css.

Lakshmipathi and Lekha T. (2007) demonstrated the BOOKiiIT venue reservation system, providing a well-defined, efficient, visual, and user-friendly system for spatial management. This provides important references for the availability and design principles of our aviation booking platform. At the same time, Lu (2007) introduced intelligent hospital appointments and drug scheduling supported by appointment systems based on near-field communication (NFC) in the context of healthcare, which provided inspiration for our optimization scheduling process and greatly helped with the approval priority and processing efficiency of airline managers.

McTavish&Sankararayanan (2010) conducted in-depth research on agile parking systems established by the Internet of Things, providing insights into the integration of emerging technologies and agile practices into aviation booking systems by updating the amount of parking spaces owned or occupied by customers through cloud servers. Sakthimohan et al. (2021) explored an intelligent agent based hotel search and booking system that provides users with the highest priority targets by collecting detailed information such as available facility lists, prices, customer experiences, and transportation. This reveals the role of collaborative systems and airline managers in seat approval, providing a reference for our system retrieval. As shown in the picture below, the system selects the most suitable location for the customer.

文本

描述已自动生成

Sankarananrayanan&Wani (2014) analyzed the e-commerce strategies of airlines in ticket distribution, providing a perspective to enhance the user experience and distribution strategies in our booking system. Singh&Shah (2020) introduced VIHARI, a multi-agent system for air ticket booking, which is suitable for the collaboration and efficiency of our aircraft booking management system.

Uran et al. (2007) conducted an in-depth study on an automated airline booking prediction system using blockchain technology, which ensures data security and centrally helps users book airline service tickets and regularly track update status. This provides automated insights consistent with our project goals, helping us analyze customer behavior and provide effective results. At the same time, it provides a reference for our customers and stakeholders to have an efficient user interface. Wei et al. (2009) introduced the online reservation system BOOKAZOR, which is a web-based reservation and scheduling application used to make appointments in designated geographic areas such as living rooms, hospitals, and architect streams. This provides us with a reference for creating a centralized platform that aligns with our goal of efficiently managing airplane reservations.

According to their process, we made the system as shown in the figure below. The system required admin to edit the flight information first, and then the manager would review the flight and finally the customer could make the purchase.

文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

文本

描述已自动生成

**III.Methodology**

The project outlines various aspects of a web application designed using a classic Model-View-Controller architecture, along with additional insights into the project's structure and technologies used. Here's a methodical breakdown focusing on the code structure and JSP files:

1. MVC Architecture:

- Model: Represents the application data domain. In Java, this typically involves classes that encapsulate data with getters and setters, allowing objects to be easily passed around and manipulated.

- View: In this context, JSP files are used as the View component. They dynamically generate HTML content server-side before sending it to the client. JSPs use tag libraries and expressions to embed backend logic into HTML.

- Controller: The controller manages the application logic, routing requests to appropriate handlers, which then update the model or select a view.

2. JSP Files Overview:

- JSP files, such as `list.jsp`, `add.jsp`, `update.jsp`, etc., are used to create dynamic web pages. For example:

- `list.jsp`: displays a list of items, such as departments, employees, or flights.

- `add.jsp` and `update.jsp`: These are used to add new entries or update existing ones in the system.

3. Persistence Layer:

- The use of MyBatis indicates an abstraction layer for database operations, where SQL queries are often mapped to Java methods for data retrieval and manipulation.

- DAO (Data Access Object): Interfaces define the contract for the operations that can be performed on each domain model, such as `DeptDao` for department operations.

4. Service Layer:

- The service layer abstracts the business logic from the rest of the application. It interacts with the DAO layer to fetch, persist, and modify data.

5. Configuration:

- The application uses configuration classes like `ApplicationContextProvider` and XML files (`applicationContext.xml`, `springmvc-config.xml`, `web.xml`) to set up the Spring Framework, defining beans, and wiring dependencies.

6. Common Utilities:

- Common constants and utility classes help maintain clean code by centralizing configuration values and reusable code snippets.

7. Domain Classes:

- Domain classes like `Dept` encapsulate the data fields, such as `id`, `name`, and `remark`, with their accessors and mutators.

8. Dynamic SQL Providers:

- These are used to construct SQL queries dynamically based on certain conditions, which can reduce the need for hard-coded SQL strings and make the code more adaptable to changes.

**UML diagrams**

**A black and white screen with many squares

Description automatically generated**

**Results**

Here are the three steps of the workflow:

1. The Airline Admin will set the price of the seats. There are three types of seats:

first class

Business

economy

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

2.The Airline Manager will see the seat list added by Admin when logging in.

Then The Airline Manager needs approval.

Only seats approved by The Airline Manager can be purchased by customers.

After the customer logged in, the customer did not see the aircraft of feijihapi2 just added.

A screenshot of a computer

Description automatically generated

When the manager logs in, the manager sees that the plane of feijihapi2 just added does not have a pass status:

A screenshot of a computer

Description automatically generated

He can click on the **✓**  mark to pass:

A screenshot of a computer

Description automatically generated

After the manager passes the feijihapi2 aircraft just added, the customer can see the feijihapi2 aircraft just added on the purchase page:

A screenshot of a computer

Description automatically generated

3.When the customer approves the itinerary, the customer will enter the payment page, where the total price will be displayed. When the customer presses the payment button, the seat is marked as sold.

The aircraft of feijihapi2 just added shows "not purchased"

A screenshot of a computer

Description automatically generated

When the customer clicks the operation button, he will enter the purchase page

A screenshot of a computer

Description automatically generated

After submitting the information, enter the confirmation page

A screenshot of a computer

Description automatically generated

When the purchase is completed, it will display "purchased"：

A screenshot of a computer

Description automatically generated

After the seat is sold, the customer will receive an email containing the flight itinerary.

A screenshot of a computer

Description automatically generated

**IV.Reports and Analysis**

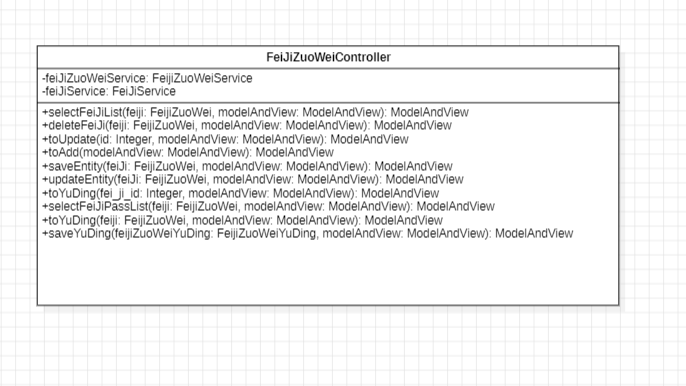
This project describes a simple and effective Java based Air ticket purchase. It uses JSP Files Overview MVC Architecture, Dynamic SQL Providers, and other methods

1. Good Characteristics

Simplicity: Each function is clearly defined and uncomplicated, making the code simple to comprehend and modify.

Efficiency: Using the classic Model View Controller (MVC) architecture, the project has a clear organizational structure that is easy to understand and maintain.

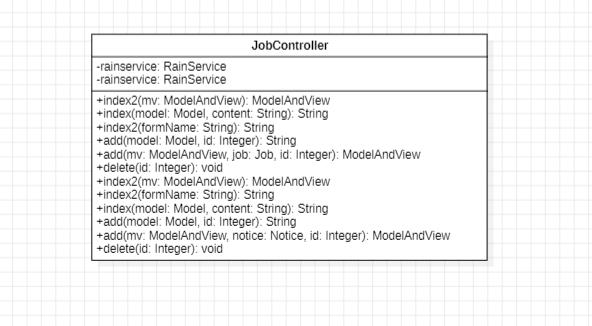
**package** com.rain.service.impl;  
  
**import** com.rain.dao.FeiJiDao;  
**import** com.rain.domain.FeiJi;  
**import** com.rain.service.FeiJiService;  
**import** org.springframework.stereotype.Service;  
  
**import** javax.annotation.Resource;  
**import** java.util.List;  
  
  
@Service  
**public class** FeiJiServiceImpl **implements** FeiJiService {  
 @Resource  
 **private** FeiJiDao **feiJiDao**;  
  
 */  
 \* 通过ID查询单条数据  
 \*  
 \** ***@param id*** *主键  
 \** ***@return*** *实例对象  
 \*/* @Override  
 **public** FeiJi queryById(Integer id) {  
 **return this**.**feiJiDao**.queryById(id);  
 }  
  
 @Override  
 **public** List<FeiJi> selectFeiJi(FeiJi feiJi) {  
 **return this**.**feiJiDao**.selectList(feiJi);  
 }  
  
  
 */  
 \* 新增数据  
 \*  
 \** ***@param feiji*** *实例对象  
 \** ***@return*** *实例对象  
 \*/* @Override  
 **public** FeiJi insert(FeiJi feiji) {  
 **this**.**feiJiDao**.insert(feiji);  
 **return** feiji;  
 }  
  
 */  
 \* 修改数据  
 \*  
 \** ***@param feiji*** *实例对象  
 \** ***@return*** *实例对象  
 \*/* @Override  
 **public** FeiJi update(FeiJi feiji) {  
 **this**.**feiJiDao**.update(feiji);  
 **return this**.queryById(feiji.getId());  
 }  
  
 */  
 \* 通过主键delete数据  
 \*  
 \** ***@param id*** *主键  
 \** ***@return*** *是否success  
 \*/* @Override  
 **public boolean** deleteById(Integer id) {  
 **return this**.**feiJiDao**.deleteById(id) > 0;  
 }



Modularity: The project is composed of multiple classes, each representing a distinct entity or function (Book, User, and BookManagementSystem). This design improves the code's maintainability.

1. Weaknesses

Error Handling: The system lacks robust error handling at the moment. It does not account for situations in which the user may enter incorrect information, which could result in unintended behaviour.



Functionality: Although the system is fully functional. But if there is an input error, there is no way to return to the previous interface, and you can only exit and perform seat selection again

1. Improvements

Error Handling: This program can increase the number of confirmations of purchase information to prevent customers from discovering purchase errors after purchasing tickets. Resulting in the need for ticket refund or repurchase

Functionality Expansion: It is possible to add ticket purchase history in the system, and multiple language translations can be added to enable people from different countries to use this program smoothly, making it more comprehensive

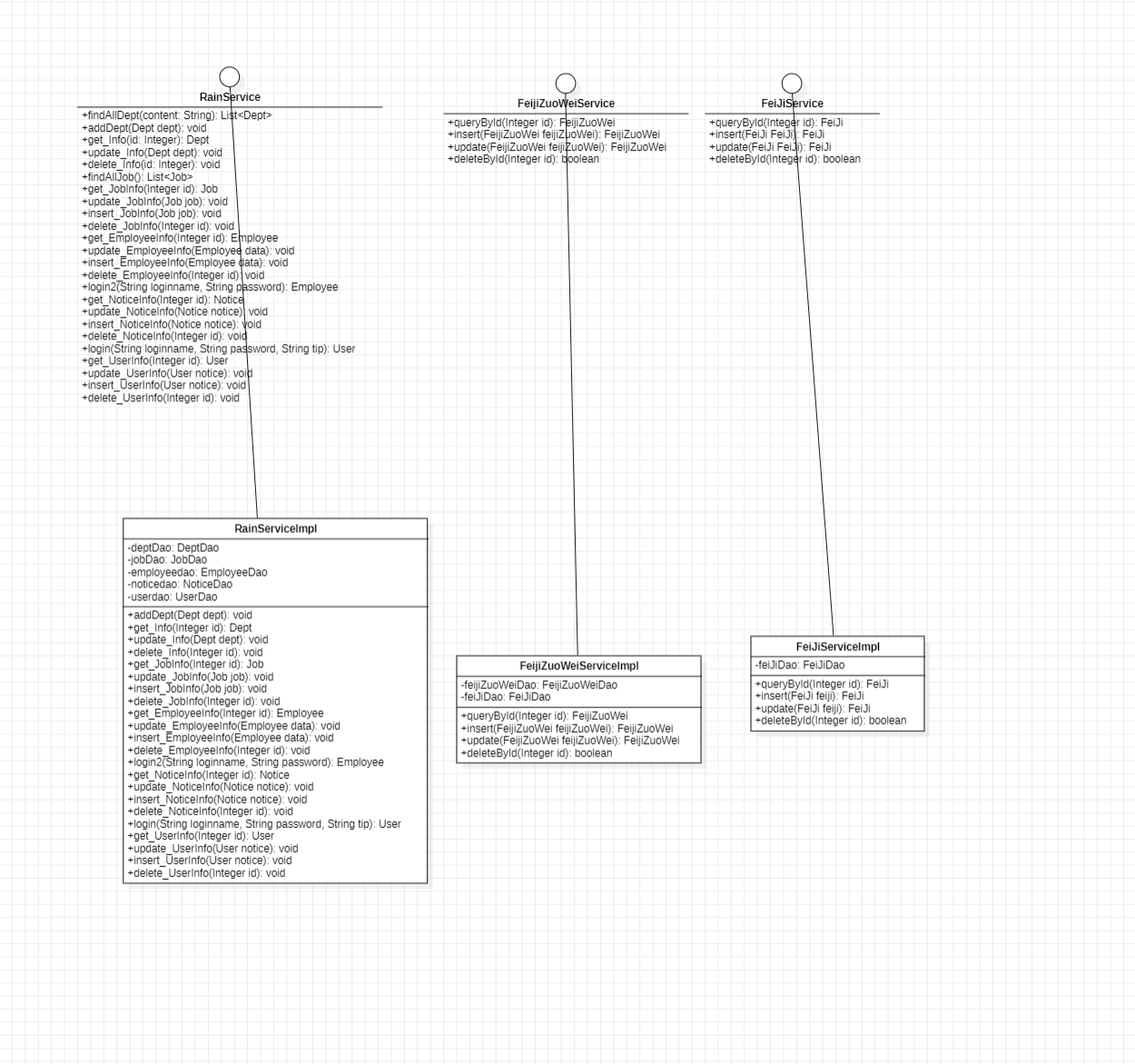
Passenger can directly search for the flight number to know the departure time of the plane



**V. Conclusions**

The Java-based Airplane Reservation Management System is a sophisticated project designed to streamline the process of managing air ticket purchases. Employing Java Server Pages (JSP) and the Model-View-Controller (MVC) architecture, the system effectively organizes and handles various aspects of flight reservation. At the core of its functionality, the system allows the Airline Admin to set prices for different seat classes - first class, business, and economy - providing the foundational structure for the reservation process.

A critical component of this system is the role of the Airline Manager. This role involves reviewing and approving the seat list set by the Admin. The system ensures that seats are not available for customer purchase until they receive the Manager's approval. This step is crucial in maintaining order and accuracy in the reservation process, as evidenced by the implementation in the 'feijihapi2' aircraft scenario, where new seats are not visible to customers until approved.



The customer interaction aspect of the system is designed with user-friendliness in mind. Customers can select flights, confirm their itineraries, and are seamlessly directed to a payment page. Upon completion of payment, the seat status is updated to "purchased," and the customer receives an email with their flight itinerary. This streamlined process is intuitive and efficient, enhancing the overall user experience.

In conclusion, while the Airplane Reservation Management System is well-structured and efficient, it would greatly benefit from enhancements in error management and expanded functionality. Implementing additional confirmation steps during the ticket purchase process and incorporating features like ticket purchase history and multilingual support would elevate the system's comprehensiveness and accessibility. Overall, the project demonstrates a commendable approach in managing airplane reservations and holds potential for further development to enhance its robustness and appeal to a wider audience.

**References**

Akshay, V., Kumar S., A., Alagappan, R. M., & Gnanavel, S. (2019). BOOKAZOR - an Online Appointment Booking System. 2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN), 1–6. <https://doi.org/10.1109/ViTECoN.2019.8899460>

Elizabeth Rani, G., Narasimha Murthy, G., Abhiram, M., Mohan, H., Singh Naik, T., & Sakthimohan, M. (2021). An Automated Airlines Reservation Prediction System Using BlockChain Technology. 2021 Sixth International Conference on Image Information Processing (ICIIP), 6, 224–228. <https://doi.org/10.1109/ICIIP53038.2021.9702587>

Lakshmipathi, J., & Lekha T., C. (2007). VIHARI: An Ontology Sevice Driven Multi-Agent System for Air Ticket Booking. 15th International Conference on Advanced Computing and Communications (ADCOM 2007), 675–680. <https://doi.org/10.1109/ADCOM.2007.118>

Lu, W. (2007). An Analysis of Airline E-commerce Strategies in Ticket Distribution. 2007 International Conference on Service Systems and Service Management, 1–5. <https://doi.org/10.1109/ICSSSM.2007.4280082>

McTavish, C., & Sankaranarayanan, S. (2010). Intelligent agent based hotel search & booking system. 2010 IEEE International Conference on Electro/Information Technology, 1–6. <https://doi.org/10.1109/EIT.2010.5612121>

Sakthimohan, M., Pasupulti, M. K., Gopa, V., Gosi, N., & Elizabeth, R. G. (2021). IOT Established Agile Parking System. 2021 3rd International Conference on Signal Processing and Communication (ICPSC), 637–641. <https://doi.org/10.1109/ICSPC51351.2021.9451732>

Sankarananrayanan, S., & Wani, S. M. A. (2014). NFC enabled intelligent hospital appointment and medication scheduling. 2014 2nd International Conference on Information and Communication Technology (ICoICT), 24–29. <https://doi.org/10.1109/ICoICT.2014.6914034>

Singh, H., & Shah, R. R. (2020). BOOKiiIT - Designing a Venue Booking System (Technical Demo). 2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM), 287–291. <https://doi.org/10.1109/BigMM50055.2020.00050>

Uran, S., Hercog, D., & Jezernik, K. (2007). Remote Control Laboratory with Moodle Booking System. 2007 IEEE International Symposium on Industrial Electronics, 2978–2983. <https://doi.org/10.1109/ISIE.2007.4375089>

Wei, Z., Dong, R., Luo, X., & Liu, F. (2009). Model Checking Airline Tickets Reservation System Based on BPEL. 2009 Third International Conference on Genetic and Evolutionary Computing, 248–251. <https://doi.org/10.1109/WGEC.2009.86>

**Appendix:**

**Contribution of Group Members**

Student Name: …Zhang Xue…. Student Number: …1163317….

Write code and Methodology (Screenshots/diagrams, Full Code Implementation)

Make UML of ApplicationContextProvider Class related beans, interceptor and util.common

Student Name: …Zhao Minghe……. Student Number: …1196750……

Write and Revise Reports and Analysis, Debugging code

Make CommonController DeptController EmployeeControllerFeiJiController

FeiJiZuoWeiController JobController NoticeController UserController

of Controller layer

Student Name: ……Zheng Qinyuan……. Student Number: …1194184…….

Write and Revise Abstract, Introduction, Debugging code

Create UML of Dept Document Employee FeiJi FeiJiPass FeijiZuoWei FeijiZuoWeiYuDing Job Notice User

of Domain layer

Student Name: ………Zhang Yuyang…. Student Number: …1194181…….

Write Related Works, References, Revise Abstract, Introduction, Reports and Analysis, Debugging code

Make UML of DeptDynaSqIProvider

EmployeeDynaSqlProvider JobDynaSqIProvider NoticeDynaSqIProvider UserDynaSqlProvider DeptDao

EmployeeDao FeiJiDao FeijiZuoWeiDao JobDao NoticeDao UserDao's UML

of Dao layer

Student Name: ………Jiang Qiwei…. Student Number: …1196688….

Write Conclusion and Appendix, Debugging code

Make UML of FeiJiServicelmpl FeijiZuoWeiServicelmpl RainServicelmpl FeiJiService FeijiZuoWeiService RainService

of service layer

**NB:** There shall be presentation by each member of a group for 5 mins.

***Dr. Ken Ehimwenma, Ph.D.***