Software Design and Architecture Assignment

Yiyun Zhai 930945 (zhaiy4)

Mincheng Li 889903 (minchengl)

(Highlight in green: updated part. Highlight in red: new added part.)

Overview

We have chosen the Human Resource Management (HRM) System as our enterprise application. As an enterprise system, the HRM system would have a different number of users depending on the size of the enterprise that uses this system. For large enterprises, the user number of the HRM system would be large. Therefore, HRM system would also need to process tons of data every day and should handle concurrency issues when multiple users accessing the same resources. And there should be several kinds of users in the system, such as administrator, department manager, employee, etc. This results in different interfaces and permissions for different user modes. Nowadays, different companies have different types of work attendance regulation. For business logic aspect, we plan to deliver flexible work-hours computation. The administrators could set up different ways of attendance recording to fit their requirements. At the same time, the system would compute the attendance records and give out the related statistical result.

Features

There are two types of roles designed in this system, respectively, Administrator and Normal User. Administrator refers to the staff who is responsible for managing this system and this type of user would have more permissions of operating the system. Normal user refers to the employee who would have limited permission.

For functions of the system, there will be two main features which are Department Management and Employee Management. The details of these two features will be described below.

- Feature A: Department Management

Department Management feature is mainly used for managing the structure of different companies. For administrators, they would have permission to create a new department, edit the departments' information and delete any departments. However, for normal users, they would have any permissions to manage the departments' information. They only have the right to view information about different departments.

- Feature B: Employee Management
  The purposes of the Employee Management feature is to manage employees' personal information and attendance records. Therefore, for administrator uses, they would create an archive for a new employee, edit the existing employees' personal information and delete any employees' archives. In addition, administrators also could view the attendance records and the related statistical result. For normal users, they could clock on or clock off on this system and manage some of their own personal information.

The user case diagram for the whole system is shown below.

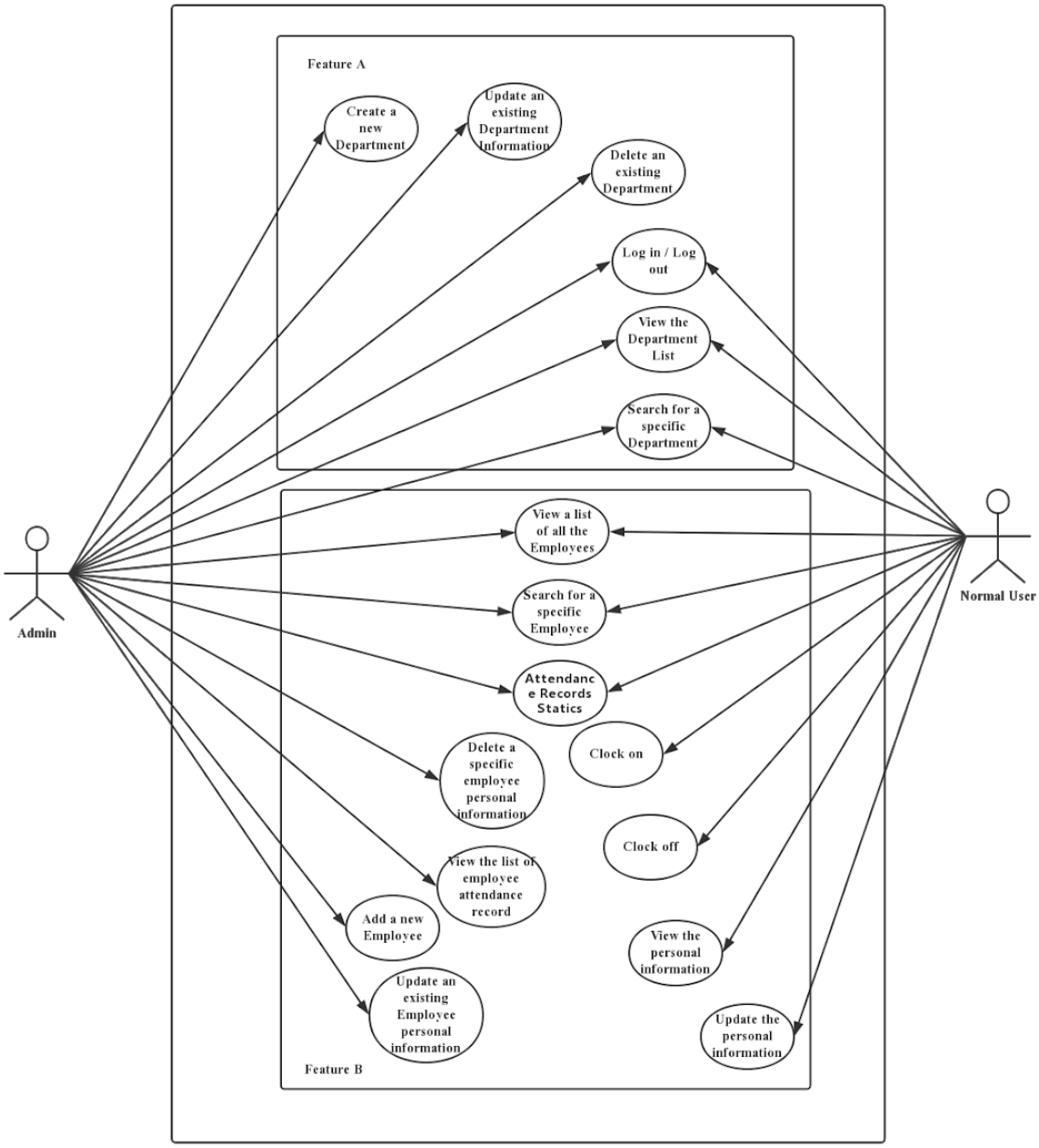

Figure 1 Use Case Diagram

Detailed Use Cases

- Feature A: Department Management

Use Case 1: Create a new department

ID	1.1
Description	Admin user can create new departments at Department Management page
Primary Actor	Admin
Precondition	User has already logged in as Admin
Postcondition	System returns to Department Management page with newly created data displayed on the system
Main Scenario	1. The system jumps to Department Management page 2. Click on “Add” button 3. Enter the information into corresponding fields 4. Click the ‘Save’ button
Extensions	Prompts an error message “Failed creation”

Use Case 2: Update an existing department information

ID	1.2
Description	Admin user can update existing departments at Department Management page
Primary Actor	Admin
Precondition	1. User has already logged in as Admin 2. There is at least one department in the list
Postcondition	The system returns to Department Management page with updated data displayed on the system
Main Scenario	1. The system jumps to Department Management page 2. Select a department and click the corresponding ‘Edit’ button

	3. Enter the information into corresponding fields 4. Click the ‘Save’ button
Extensions	Prompts an error message “Failed edit”

Use Case 3: Delete an existing department

ID	1.3
Description	Admin user can delete existing departments at Department Management page
Primary Actor	Admin
Precondition	1. User has already logged in as Admin 2. There is at least one Department in the list
Postcondition	System returns to Department Management page with updated data displayed on the system
Main Scenario	1. The system jumps to Department Management page 2. Select a department and click the corresponding ‘Delete’ button
Extensions	Prompts an error message “Failed delete”

Use Case 4: View the Department List

ID	1.4
Description	Admin and Normal User can view Department List at Department Management page
Primary Actor	Admin, Normal User
Precondition	User has already logged in as Admin or Normal User
Postcondition	The system displays existing Department List
Main Scenario	1. The system jumps to Department Management page
Extensions	Prompts an error message “Department information not

	found”

Use Case 5: Search for a specific Department

ID	1.5
Description	Admin and Normal User can search for a specific department at Department Management page
Primary Actor	Admin, Normal User
Precondition	User already logged in as Admin or Normal User
Postcondition	System returns to Department Management page with query result displayed on the system
Main Scenario	1. System jumps to Department Management page 2. Enter the information into searching input box 3. Click the ‘Search’ button
Extensions	Prompts an error message “Department information not found”

- Feature B: Employee Management

Use Case 6: Add a new employee

ID	2.1
Description	Admin user can add a new employee at Employee Management page
Primary Actor	Admin
Precondition	User has already logged in as Admin
Postcondition	The system returns to Employee Management page with newly created data displayed on the system
Main Scenario	1. The system jumps to Employee Management page 2. User click on “Add” button 3. Enter the corresponding information 4. Click ‘Save’ button

Extensions	Prompts an error message “Failed creation”

Use Case 7: Update an existing employee personal information

ID	2.2
Description	User can edit an existing Employee information at Employee Management page
Primary Actor	Admin
Precondition	1. User has already logged in as Admin 2. There is at least one Employee in the list
Postcondition	The system returns to Employee Management page with updated data displayed on the system
Main Scenario	1. The system jumps to Employee Management page 2. Select an employee and click the corresponding ‘Edit’ button 3. Enter the information into corresponding fields 4. Click the ‘Save’ button
Extensions	Prompts an error message “Failed edit”

Use Case 8: View a list of all the Employees

ID	2.3
Description	Admin and Normal User can view the Employee List at Employee Management page
Primary Actor	Admin, Normal User
Precondition	User has already logged in as Admin or Normal User
Postcondition	The system displays existing Employee List
Main Scenario	1. The system jumps to Employee Management page
Extensions	Prompts an error message “Employee information not found”

Use Case 9: Search for a specific employee

ID	2.4
Description	Admin can search for a specific employee based on the keywords they enter.
Primary Actor	Admin, Normal User
Precondition	User has already logged in as Admin or Normal User
Postcondition	The original Employee List changes to the list of search results
Main Scenario	1. The user enters the Employee Management page 2. The user enters the name of the employee 3. Click the ‘search’ button
Extensions	Prompts an error message “There is no related employee”

Use Case 10: Delete a specific employee personal information

ID	2.5
Description	Admin can delete a specific employee’s personal information.
Primary Actor	Admin
Precondition	User has already logged in as Admin
Postcondition	The deleted employee item has been removed from the Employee list
Main Scenario	1. Select an employee and click the corresponding “delete” button 2. The system displays a pop-up window to confirm whether delete operation would be performed. 3. Click the “confirm” button 4. The pop-up window disappears 5. The system returns to the Department Management page and reloads the data
Extensions	Prompts an error message “The delete operation fails to perform. Please try again.”

Use Case 11: View the list of employee attendance record in The System

ID	2.6
Description	The admin can view all the attendance records
Primary Actor	Admin
Precondition	User already logged in as Admin
Postcondition	The system displays existing Attendance Record List
Main Scenario	1. The system enters the Attendance Record Page and loads the Attendance Record data
Extensions	Prompts an error message “Attendance record information not found”

Use Case 12: Clock On

ID	2.7
Description	Normal User can clock on
Primary Actor	Normal User
Precondition	User has already logged in as Normal User
Postcondition	User successfully records the working hours
Main Scenario	1. User Click the ‘Clock on’ button 2. The system enters the result page to display the operation result and time
Extensions	Prompts an error message “Fail to Clock on. Please try again.”

Use Case 13: Clock Off

ID	2.7
Description	Normal User can clock off.

Primary Actor	Normal User
Precondition	User has already logged in as Normal User
Postcondition	User successfully records the working hours
Main Scenario	3. User Click the ‘clock off’ button 4. The system enters the result page to display the operation result and time
Extensions	Prompts an error message “Fail to Clock off. Please try again.”

Use Case 13: View the personal information in The System

ID	2.8
Description	Normal User can view his/her personal information.
Primary Actor	Normal user
Precondition	User has already logged in as Normal user
Postcondition	The system displays the personal information of the current user
Main Scenario	1. The system enters the personal information page and loads the related data
Extensions	Prompt an error message “Fail to display your personal information.”

Use Case 14: (Employee) Update the personal information

ID	2.8
Description	Normal User can update some of his/her personal information.
Primary Actor	Normal user
Precondition	User has already logged in as Normal user The information which could be modified by the employee

	is the information which is in editable text fields.
Postcondition	Personal information of the current user has been updated successfully
Main Scenario	1. The system enters the personal information page and loads the related data 2. Click the “edit” button 3. Modify the related information 4. Click the “save” button
Extensions	Prompts an error message “Fail to update your personal information. Please try again.”

Use Case 14: Attendance Record Statics

ID	2.8
Description	Admin and Normal user can view the statistical result of attendance records.
Primary Actor	Admin and Normal user
Precondition	User has already logged in as Admin or Normal user
Postcondition	The statistical result of attendance records is successfully displayed
Main Scenario	1. The system enters the statistics page
Extensions	Prompt an error message “Fail to display the data. Please try again.”

High-Level Architecture

High-Level Design

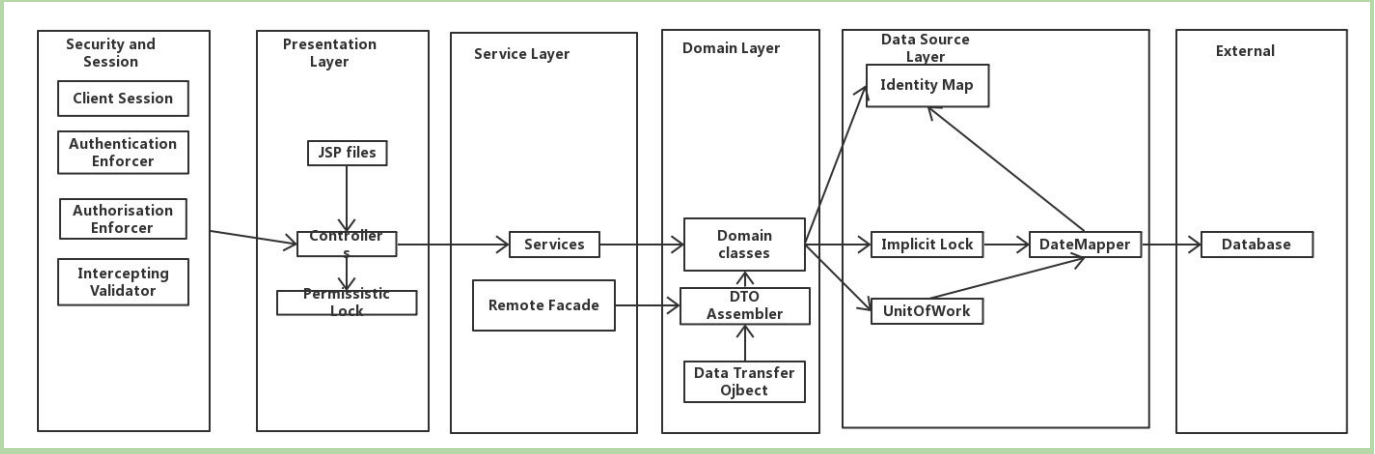

Figure 2 High Level Design Diagram

The high-level architecture is considered as a five-layer model, including security & session layer, presentation layer, service layer, domain layer and data source layer. The main function of the presentation layer is to present the user interface. It is responsible for collecting user input, passing it to the next layer (service layer), and returning processed data to the user interface. Except for the basic functions of the presentation layer, it also cooperate with security & session layer. It is responsible for managing the information stored in the session and invoke the corresponding methods to finish the input invalidation and authorisation.

When the presentation layer receives the requests from the user, the system will invoke the corresponding methods in the service layer to finish each request. Thus, the main function of the service layer is to handle the simple request directly sent from the presentation layer, and call methods in the domain layer. It is responsible for implementing the detailed functions by combining different methods in the domain layer.

The main function of the domain logic layer is to handle the sub-functions from the service layer, including different operations on each model. Therefore, the models for each object used in this project will be contained in this layer. In addition, this layer also consists of Data Transfer Object and Assembler classes for each entity class, which could be used for distribution.

The main function of the data source layer is to finish truly interaction with database and provide interfaces for the domain layer. This layer consists of the data mapper classes, unit of work classes and identity map classes of each model. In addition, the implicit lock is also implemented in this layer.

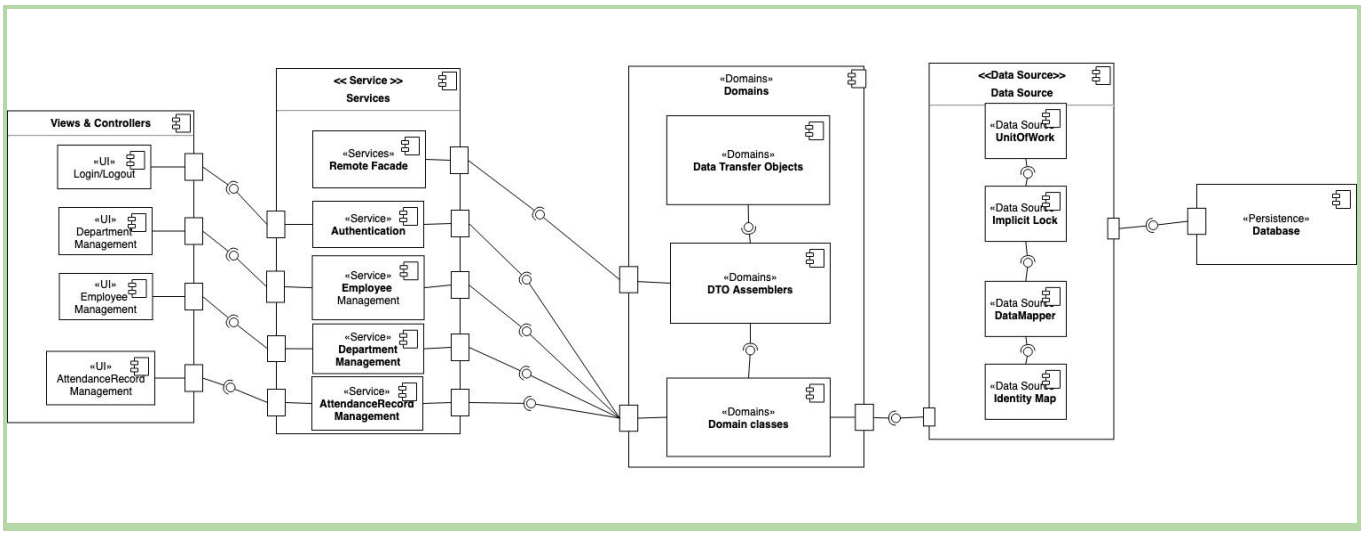

Figure 3 System Component Diagram

Figure 4 Class Diagram

Pattern Implementation

Presentation Layer

The presentation layer is a component that provides user interface and interacts with service layer. Therefore, on this layer, it contains Page Controller, Template View and Pessimistic off-line lock.

The presentation pattern selected in this system is Model View Controller (MVC). The view section includes all JSP files, and the controller section includes controller servlets in “servlet” package.

- **Page controller**

Page controller will create a controller servlet for each request of JSP file. Compared with front controller, it has a little too much redundant. And front controller is easy to extend, it is more appropriate for system with huge functions. However, consider that creating a controller for each operation is not very time-consuming and more importantly, it allows developers to clearly distinguish and understand. Therefore, the page controller would be better for this system.

For implementation, there is a single controller servlet for each action. Each servlet handles GET and POST requests and is responsible for passing request to service layer.

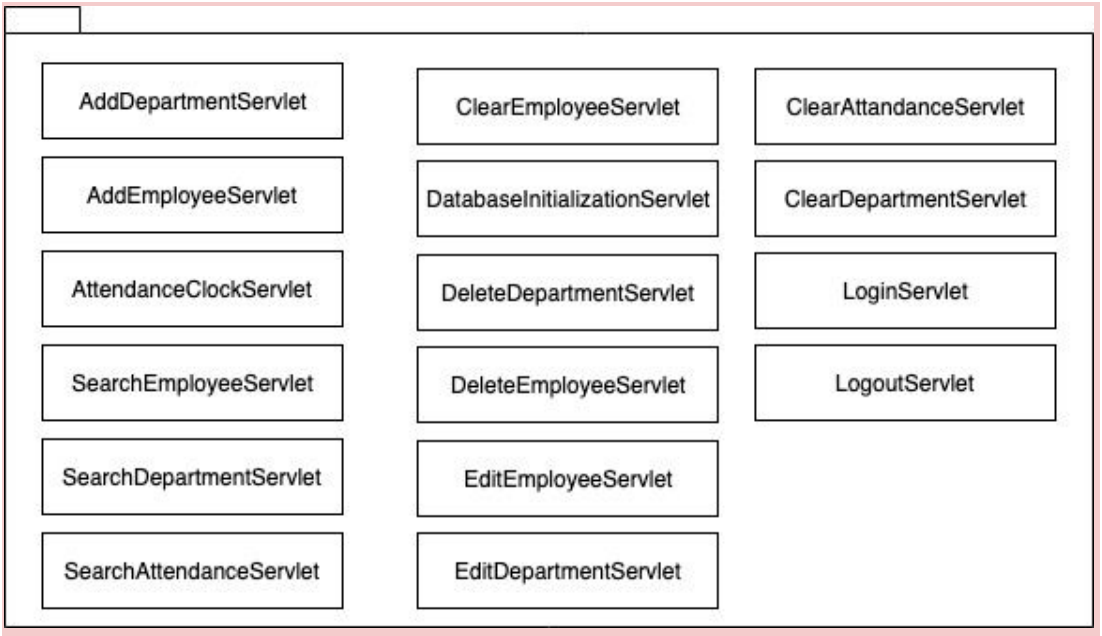

Figure 5 Servlet Package Diagram

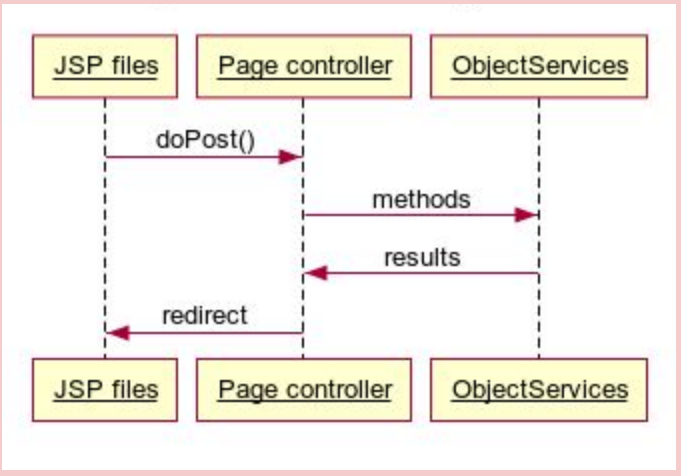

Figure 6 PageController Interaction Diagram

- **Template view**

Template View, which is one of the most common approach, is used in this system. It is easy to implement with dynamic information contained in web pages. It is easy to design and reduces the programming effort. And the disadvantage of this pattern, which is that embedding Java code into the html code leads to the higher difficulty of testing, could also be overcome by carefully coding. Overall, this pattern is the most suitable for this system.

For implementation, there are scriptlets embedded in JSP files. The scriptlets' function include authentication, authorization, and interaction with other layers.

- **Pessimistic off-line lock**

For a human resource management system, it has a large number of users, large storage data, high concurrency possibility. So for the system, preventing the concurrent problem is indispensable. The optimistic off-line lock works by rolling back transactions when a data conflict is detected. Transaction rollback will increase the system calculation burden, bring loss of user data and increase the security risk. So it is better to avoid conflicts than to detect and fix it. Although pessimistic locks are more complicated to implement, it is necessary for this kind of enterprise systems.

For implementation, the class "LockManager" is created to limit the number of users reading and writing data at the same time. When a user sends a read or write request, he acquires a read or write lock, which is released after the read or write operation is complete. Other users who send read and write requests during this period will be

rejected by the server. The following diagram is an example of adding a new department to the system using the pessimistic off-line lock.

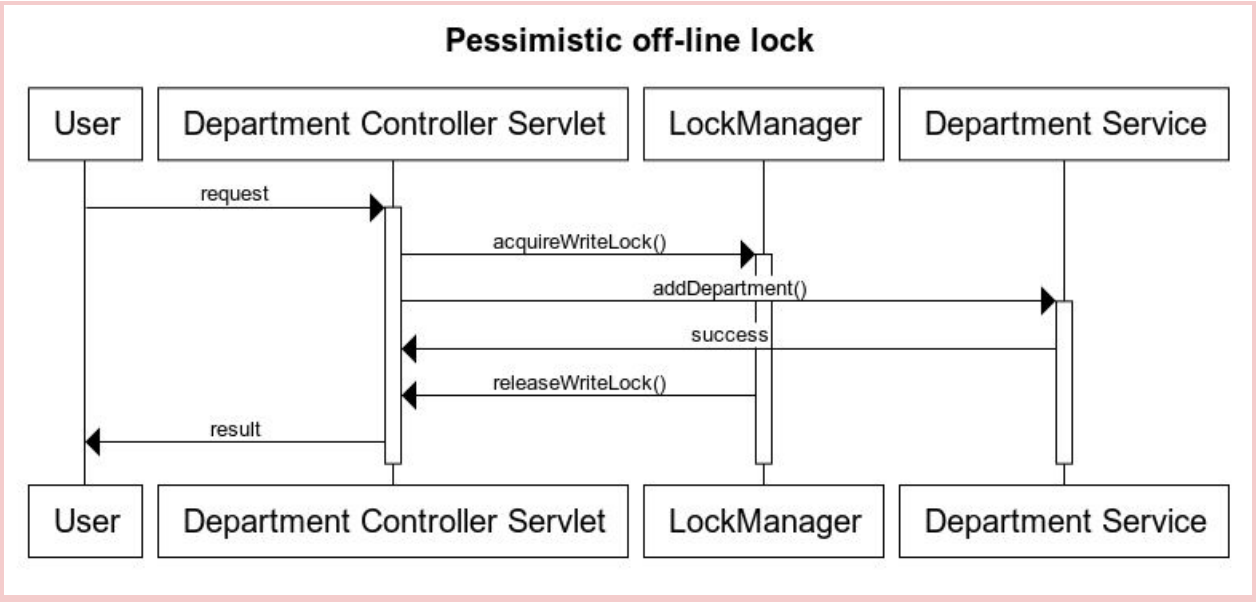

Figure 7 Pessimistic off-line lock Interaction Diagram

Domain Layer & Data Source Layer

- **Domain model and inheritance patterns**

The domain diagram shows the relationship between the entity classes. For this system, there are five main models, respectively, User, Employee, Admin, Department and Attendance Record. User model is implemented as an abstract model and it shares some basic properties and functions. Employee model and Admin model inherit from the User model. Employee model add a private property which refer to the department where the employee works in while Admin model add a list of Department model. For the Department model, except for the basic perproties, which are departmentID, name, phone number and location, there are two arraylist properties, which are used for storing the data of the administrators and employees of the departments. The Employee model and the Department model forms a one-to-many relationship, which means that one department could have different numbers of employees. At the same time, there also exists the one-to-many relationship between the Employee model and the AttendanceRecord model. However, in this system, one admin is allowed to manage more than one departments while one department could have at least one administrators. Therefore, for the relationship between the Admin model and the Department model, it is a many-to-many relationships.

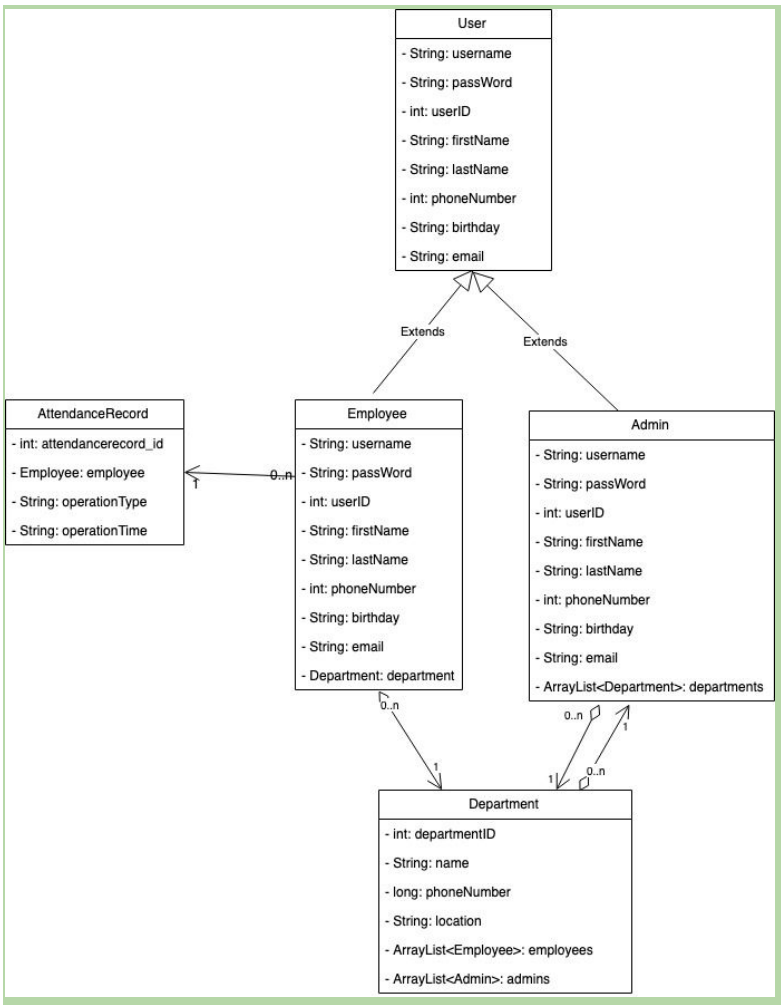

Figure 8 System Domain Diagram

- **Unit of work**

For Employee, Admin and Department models, they all have a unit-of-work class to help handle all the operation of insert, modify and delete data. There are three ArrayList in each unit-of-work class to respectively record the objects which need to be newly inserted, modified and deleted. When the commit method is called, the corresponding operations will be performed on the objects in these ArrayLists.

- **Identity Map**

To reduce the frequency of reading data from the database, an identity-map class is created to help keep and maintain the data read from the database before. Therefore, when an item needs to be searched from the database, the system will first search the item in the corresponding identity map. If there is not correct item found, the system will turn to query in the database.

- **Data Mapper**

Data Mapper of each model will eventually finish all the interactions with the database. As mentioned above, when a user wants to insert, delete or modify any objects, the methods in the unit-of-work classes will be called. However, once the commit method is called, the system will call the methods in the Data Mappers to truly update the information in the database.

- **Lazy Load**

The implementation of Lazy load pattern is used for loading the information that the system needs but not all the information. Therefore, in this system, for the properties of each model, when their getter methods are called but they do not contain any meaningful values, the system will help load value from the database for these variables.

- **Identity field**

To maintain the relations between each object and its database record, the id of each object is set as the same as the id of the record in the database.

- **Foreign key mapping and Embedded value**

Different from the User model, the Employee model has a property which refers to a department object. Therefore, in the database, the table of the Employee model contains a foreign key from the table of the Department Model. Similarly, in the Department model, there is an ArrayList property to store all its employees' information. Thus, they form a one-to-many relationship.

- **Association table mapping**

The relationship between the Admin model and Department model is more complex. In this system, it is allowed that an administrator could manage different departments and a department could have different administrators. They have a kind of many-to-many relationships. To implement this, there is an ArrayList, which is used to store all its departments, in the Admin model. For the Department model, there is also an ArrayList used for storing all its administrators' information. To more clearly record their relationship, in the database, this kind of relationship is not recorded in the tables of the Admin model and the Department mode. A new table named 'admin_department_table' is created for this kind of relationship.

- **Implicit lock**

The main difference between implicit lock and pessimistic lock is the implementation architecture layer for read/write locking and unlocking. The implementation layer of

implicit lock is at data mapper layer, which is much lower than pessimistic lock is. This means that read/write locks are acquired as late as possible and released as early as possible. It will shorten the time users have to wait for read/write access.

For implementation, the class “LockingMapper” is created to handle read/write lock at data mapper layer. To be specific in this system, “DepartmentLockingMapper” and “DepartmentDataMapper” implements interface “IDepartmentMapper”. “DepartmentLockingMapper” is responsible for acquiring locks. After receiving read/write request, it acquires lock and registers object to “unitofworkDepartment” directly. The following diagram shows an example of adding a new department using implicit lock.

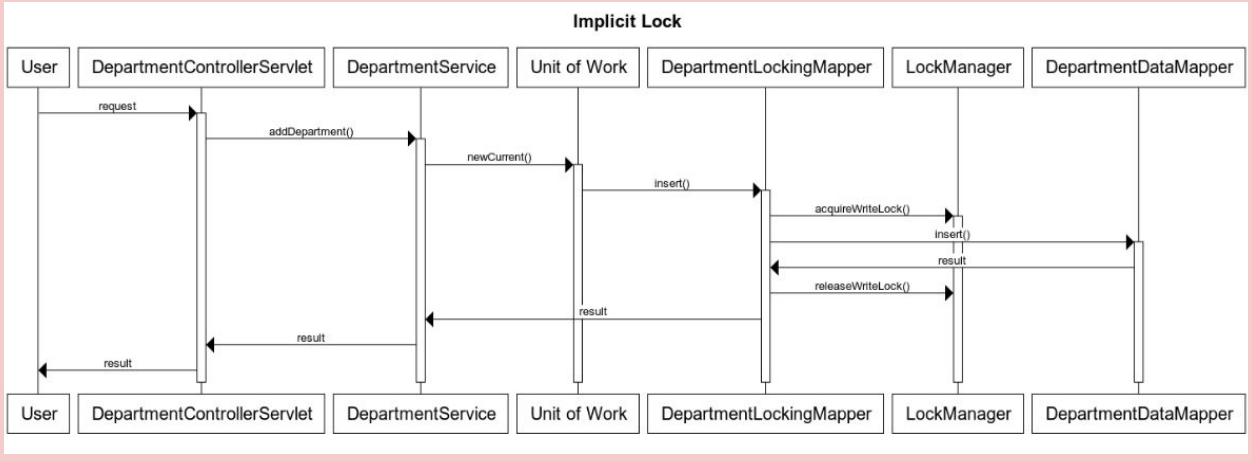

Figure 9 Implicit Lock Interaction Diagram

- **Data Transfer Object**

For some small system, the data transfer between the front-end and back-end might be not too big. However, for enterprise systems, the number of data which need to transfer will be much larger. Sometimes, some functions might need different data for one transaction, which might lead to send requests and get responses for multiple times. However, this situation might lead to a waste of resources and time. Therefore, packaging all the required data into one object would be a good way and Data transfer Object is designed to implement this idea.

To implement this pattern, two different classes are needed to be implemented for one domain class and these two classes are respectively DTO class and Assembler class. For DTO class, it mainly responses for implementing the getter and setter methods. These methods are corresponding to those of the corresponding domain class. Then the DTO class will also implement the methods of serialization and deserialization. For

the Assembler class, it mainly focuses on the data transfer between the DTO class and the corresponding domain class. The interaction diagram of this pattern can refer to the Remote Facade Diagram.

Service Layer

The service layer is a component that connects the presentation layer and domain logic layer. The service layer consists of four classes which correspond to domain objects respectively.

Class “UserService” handles login and authentication request. “AttendanceService”, “DepartmentService”, “EmployeeService” handle the functions of the three subsystems. They are all used for receiving parameters from the presentation layer, and calls function in domain layer. They are used to simply handle direct requests from the presentation layer.

- **Remote facade**

Remote facade mainly provides a coarse-grained facade to reduce the number of sending requests and receiving responses and then achieve the goal of improving the efficiency of the data transfer between the frontend and backend.
In this system, the Remote facade finishes its functions mainly by invoking the assembler methods.

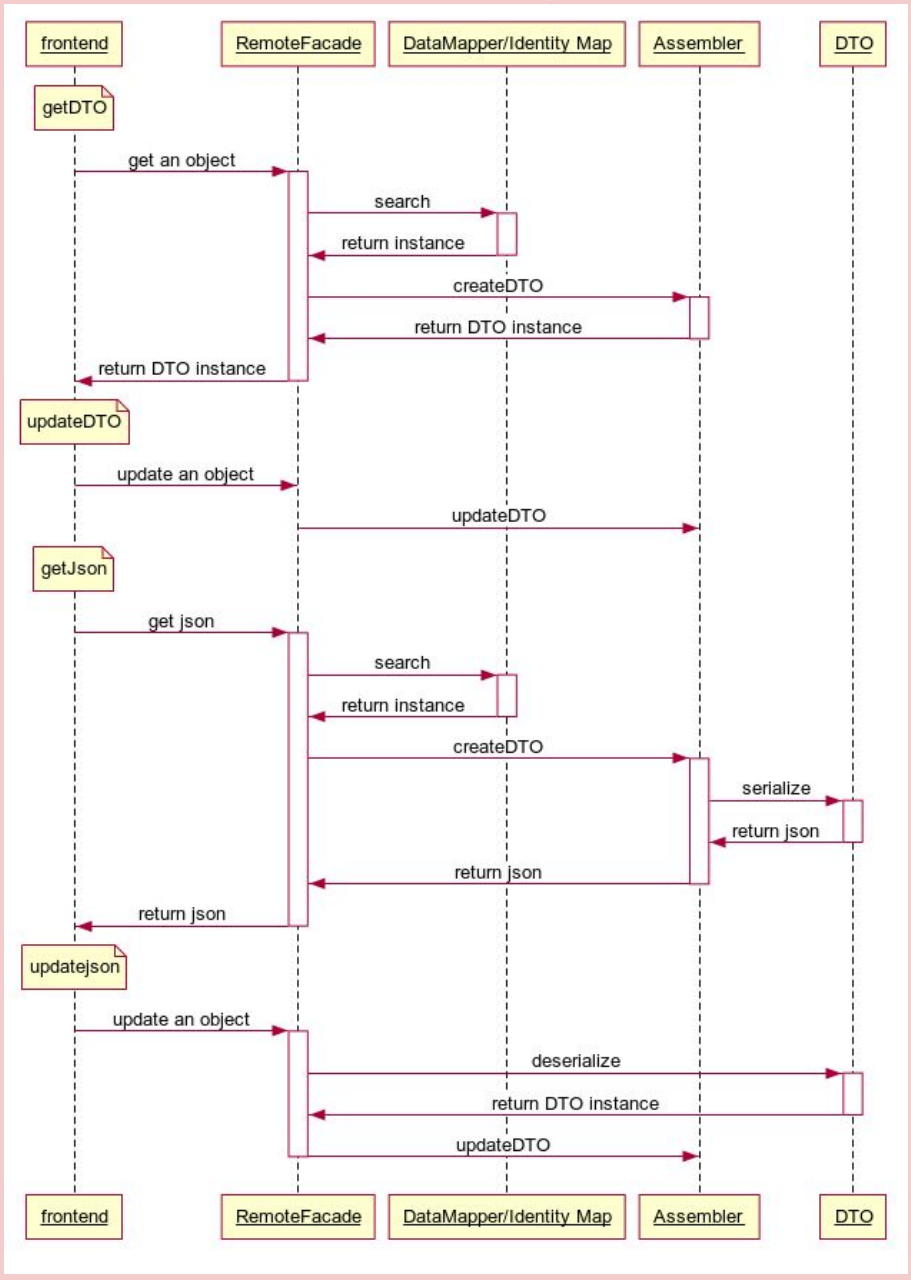

Figure 10 Remote Facade Interaction Diagram

Security & Session Layer

- **Client session**

Since HTTP connection is stateless, it is necessary to store some user information locally. Consider the session information that need to be stored within this system, there are not too many session states need to be stored. Important session states include: user name, user type and user permissions. Using client session could make the implementation simple compared with more complicated database session and server

session. Besides, it works well with server clusters, reduce the workload of server and database. There are drawbacks to using client session state, including poor scalability and safety issues. Since there is no high performance and security concerns, client session is sufficient to meet current requirements. Thus, client session state is chosen in this system.

For implementation, HTTP session is used. Session attributes are set in HTTP session in controller servlet. And then JSP files get HTTP sessions when needed. The method of URL parameter is insecure because user information is directly exposed in URL. The following diagram shows a general process of client session usage.

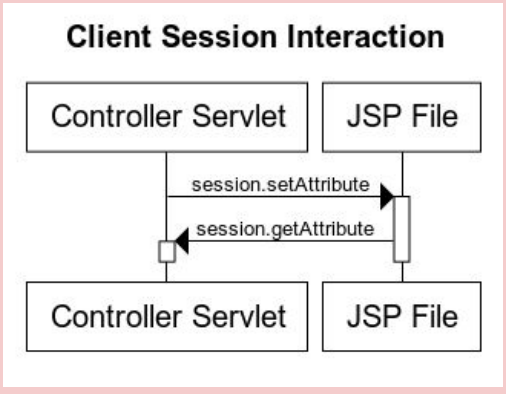

Figure 11 Client Session Interaction Diagram

- **Authentication enforcer**

Authentication enforcer verifies that each request sent is from an authenticated entity. It is also obvious necessary for human resource systems. Each user's identity is different. If a user acts as someone else, that can be a problem. For this system, unauthenticated users should not be able to open the "departmenManagement" page. In contrast, authenticated users should not be able to see the login page. Login to the system with username and password is the simplest example of authentication. The existence of authentication enforcer requires more than just the ability to authenticate the login operation. It needs to store a user's authenticated identity somewhere and manage the user's view with the identity as the user uses the system.

For implementation, the external library "shiro" is used. It could help store user information, and could be called whenever controller servlet or JSP files need it. The class "AppRealm" initialize the authentication enforcer when the system starts. Then in presentation layer, use functions in "shiro.SecurityUtils" to get authentication information. Using external libraries can reduce manual implementation and reduce

code redundancy. The following diagram shows the process of user login and turn to index page.

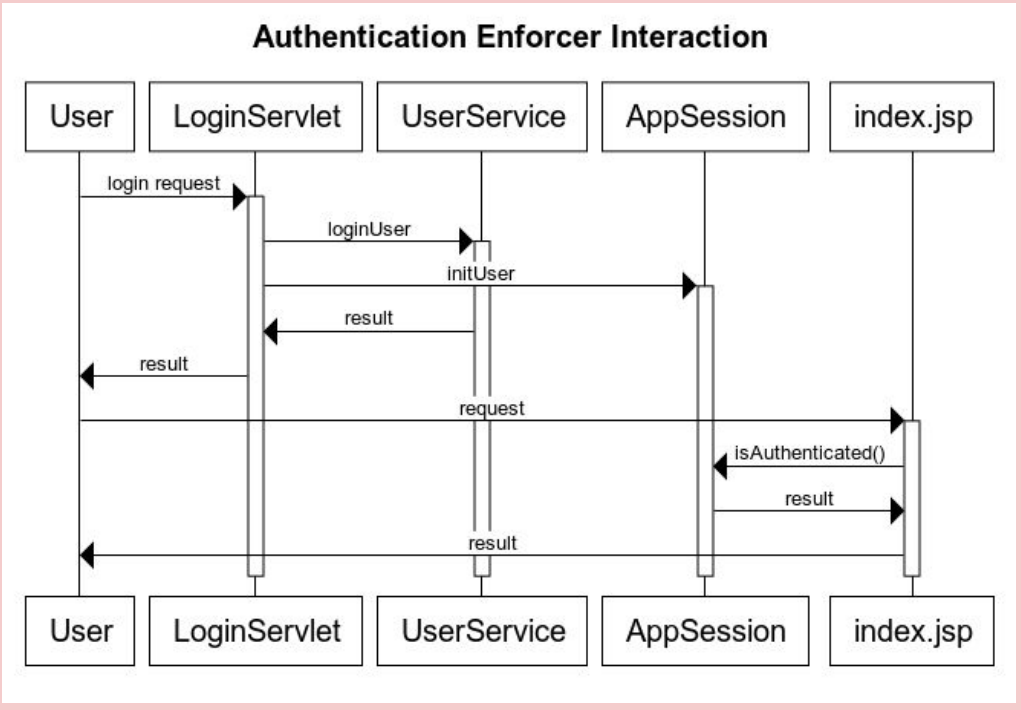

Figure 12 Authentication Enforcer Interaction Diagram

- **Authorisation enforcer**

Authorization enforcer verifies that each request is being performed on behalf of an authorized entity. Each user in this system has a different role which can be admin or employee. An admin should have the permission to add, edit, delete, view all the departments and employees. An employee should have permission to view the information above, but not to modify the data. For example, if a user has logged in as an employee, he is not supposed to see “add”, “edit”, “delete” buttons. If he opens the “addDepartment” page, he will also be judged to have no permission when he submits the form and the operation will be refused. The usage of authorization enforcer provides a centralized encapsulation of authorization mechanisms which helps handle the view and action of users based on user permissions. This pattern becomes even more essential as the enterprise system has more user roles.

For implementation, “shiro” is also available for this pattern. It stores user roles when they login, and the authorization check method could be called in controller servlet and JSP files. The initialization process is similar to authentication enforcer it is generally partnered with authorization enforcer. Using this pattern also eliminate the existence of

redundant code, allow greater reuse. Besides, the security information is encapsulated, and it is easier to change permissions and scale.
The following diagram shows the process of requesting to add a new department after logged in.

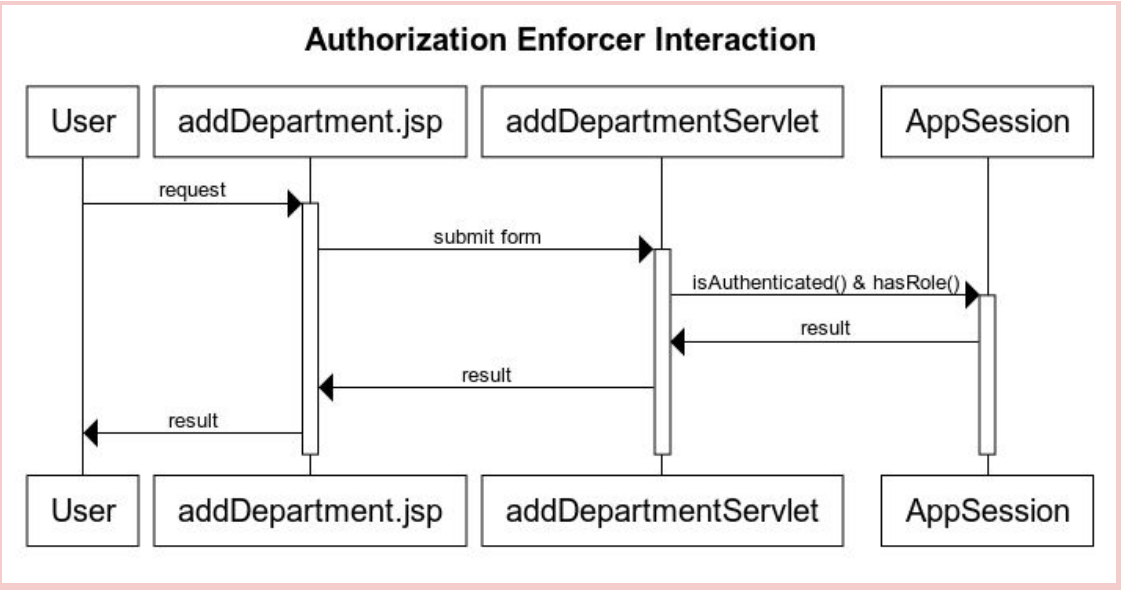

Figure 13 Authorization Enforcer Interaction Diagram

- **Intercepting validator**

Intercepting validator validates that each request is validated as well-formed and non-malicious content. For example, when adding a new employee to the system, there is a “email” property. If the user enters illegal email, it might create troubles for admins who need to use the data later. Besides, if the length of the string exceeds the data format in database table, it would cause a system crash. This pattern can effectively prevent malicious input data to harm the system. Although it could be complex to implement, and might affect the system performance, it is still necessary for a human resource system that store important data.

For implementation, regular expression judgement is used in this system. Instead of using an external library, this method is easier and more convenient for this system. Thus, for each form that would be submitted, there is a page controller that handled the input data. A regular expression judgement is added for each input. For the “email” property mentioned above, there are limitations for the input. The length of the string, whether the string is empty, the format of the string would be judged in the controller servlet. If the input data is illegal, there will be a pop-up window warning the user, and the operation will not be processed.

The following diagram shows the process of adding a new employee using intercepting validator.

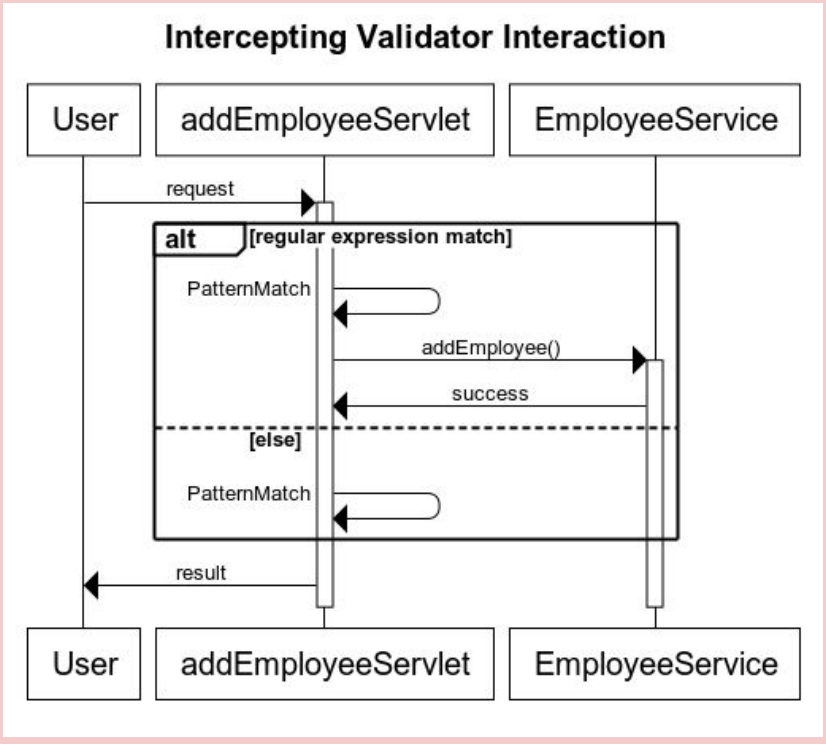

Figure 14 Incepting Validator Interaction Diagram

System Function Implementation

- **Log in**

When an admin or an employee wants to log in to the system, they will first enter their username and password on the user interface. The LoginServlet will store the username in session for later use. And the presentation layer processes the login request, and send a POST request to the next layers. The data source layer first query the database to see if the username exists. Then check whether the username matches the password. Ultimately, the underlying data source layer will returns whether the query is successful, and the user type (admin or employee).

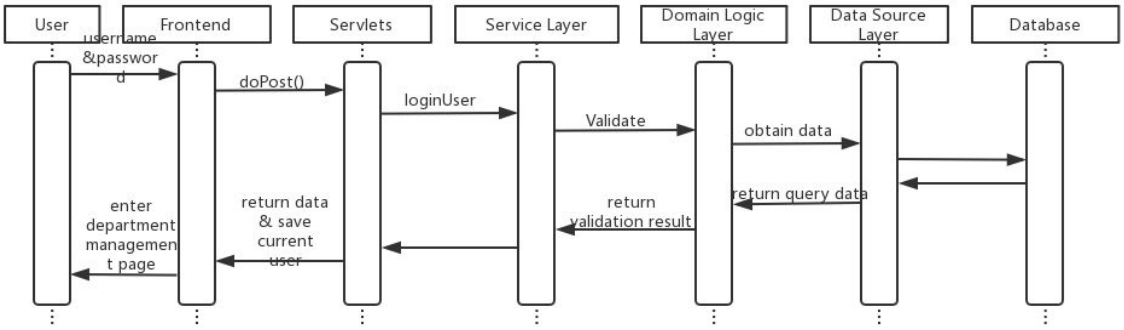

Figure 15 System Login Sequence Diagram

- View all the departments

After the user logins to the system, the homepage will be a list of all department, displaying the username from the session. Both admin and employee users can access the view of the list. The presentation layer calls getAllDepartment() function in service layer. And service layer access required data from Domain Logic Layer. Based on the input parameters, the system will first check whether the required data exists in the identity maps, otherwise, it will send requests to the database via Data Source Layer.

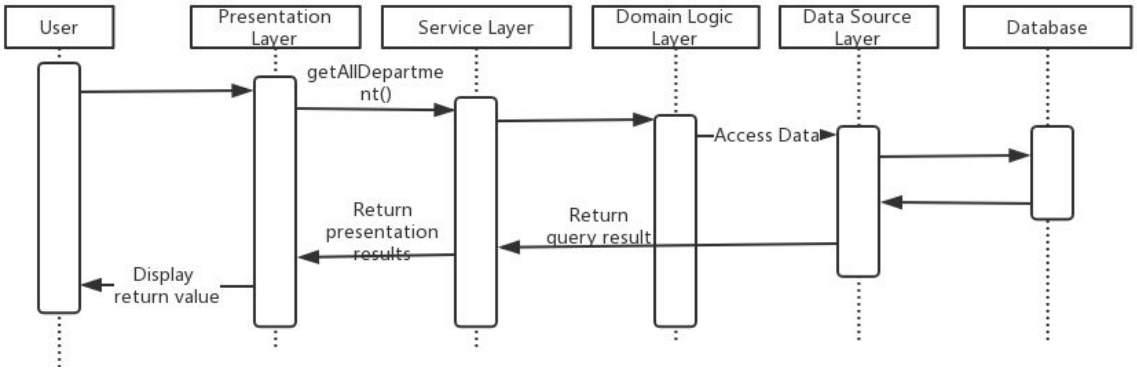

Figure 16 System View All Department Sequence Diagram

- Add A Department

The department management function is only available for admin users. When an admin wants to create a new department for the system, he will first enter the information for the department on the user interface. And the presentation layer processes the creation request and calls the functions on the service layer. The service layer call the registerNew method from the unit-of-work class of Department model, and then commit to data source layer. Data mapper looks in the database to search if there is existing department that conflict with user input, and returns whether the creation operation is successful or not. Finally the interface will be redirected to the department list page with newly added data.

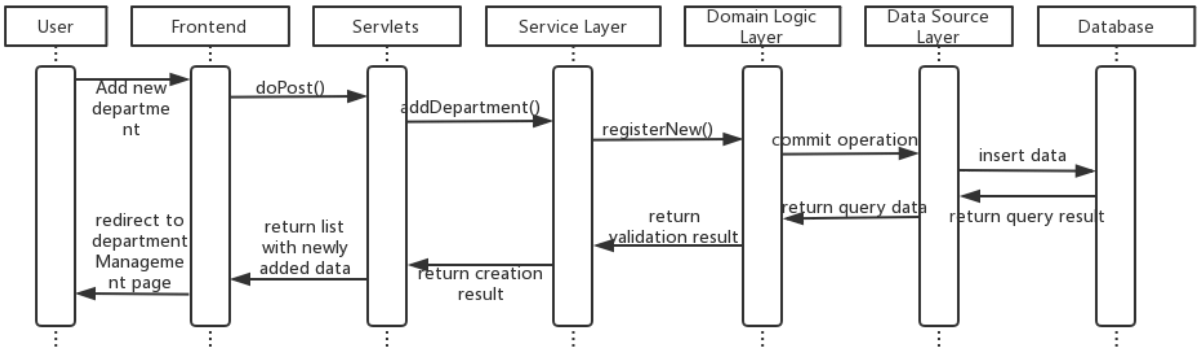

Figure 17 System Add Department Sequence Diagram

- Delete A Department

The delete function is similar with the add function above. When an admin wants to delete an existing department, he will click on the “Delete” button on a specific row. The DeleteDepartmentServlet will get the department_id from front-end, and pass the parameter to the service layer. The service layer then calls deleteDepartment method, and domain logic layer executes registerDeleted method from the unit-of-work class of Department model. The data mapper searches and verify that the corresponding department item exists in the database table. Ultimately, front-end will receive the result and will redirect to the updated department list.

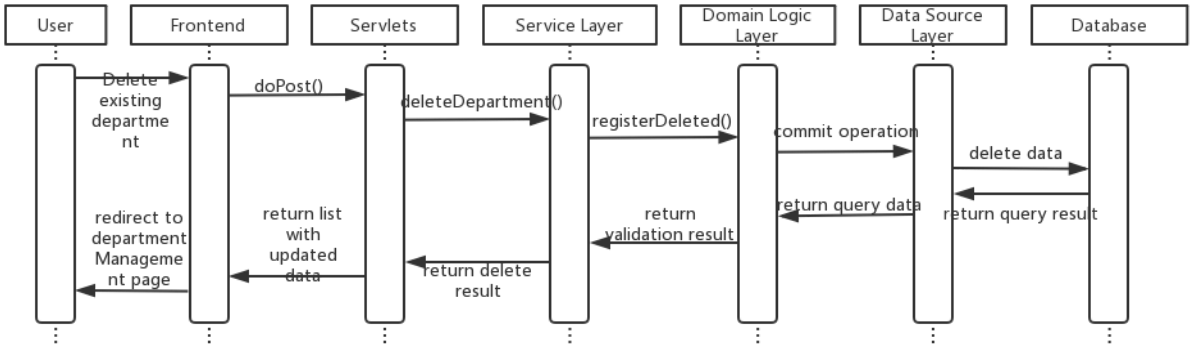

Figure 18 System Delete Department Sequence Diagram

- Edit A Department

When an admin wants to edit a department, he will click on the “Edit” button on a specific row. The EditDepartmentServlet will get the department_id from front-end, store it in session, and pass the parameter to service layer. The service layer calls search function to get the whole information of the selected department. Then the web page will be redirected to editDepartment page, with department ID, name and location sent back

from service layer. Then the admin can enter the data that he wants to change. After clicking “Confirm” button, the form will be submitted. The EditDepartmentServlet will handle the POST request. It calls editDepartment function in service layer, and service layer registerDirty from the unit-of-work class of Department model. And then the delete operation will be committed to the data source layer. Finally the web page will be redirected to the department list page with newly updated data.

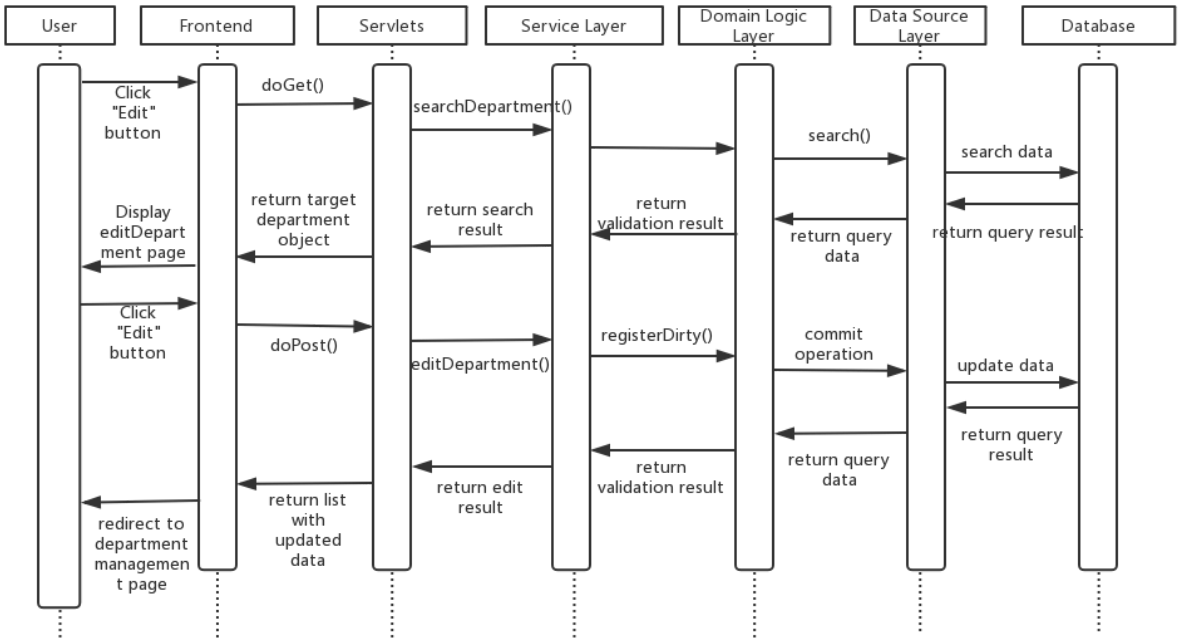

Figure 19 System Edit Department Sequence Diagram

- Search For A Department

Both admin and employee users can search for departments by identical ID and name of departments. The user first enter some inputs in the search bar, and clicks “search” button. The SearchDepartmentServlet stores the string that the user entered in session for later use, and calls searchDepartment function in service layer using the string parameter. The service layer then calls search function in data mapper in the domain logic layer. If the item is existing in the identity map and the system will directly return it as result Otherwise, the data mapper will finish a query operation in the database. The service layer then passes the list to the servlet for further process. The front-end gets the arraylist from servlet, and displays on the web page.

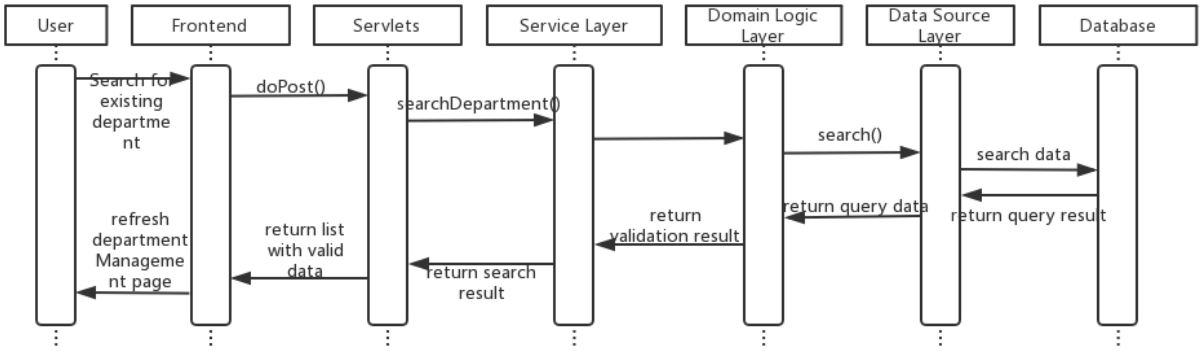

Figure 20 System Search Department Sequence Diagram

- View a list of all the Employees

For the Employee Management Subsystem, the functionality to display all the employee information is basic but important. When the user entered the Employee Management Subsystem, the system will invoke the corresponding function in the Employee Service, and then the corresponding query method in the Employee domain class will be invoked. The query method in the Employee domain class will query the database by using the methods in the Employee DataMapper, which is responsible for the interaction between Employee objects and the database. While getting data from the database, the Employee DataMapper will also invoke the Employee IdentityMap to check if these data have been in the Employee IdentityMap. If they were not in the identity map, they will be added into the identity map. After all the query operations, the EmployeeService will return a list of the Employee information to the frontend browser.

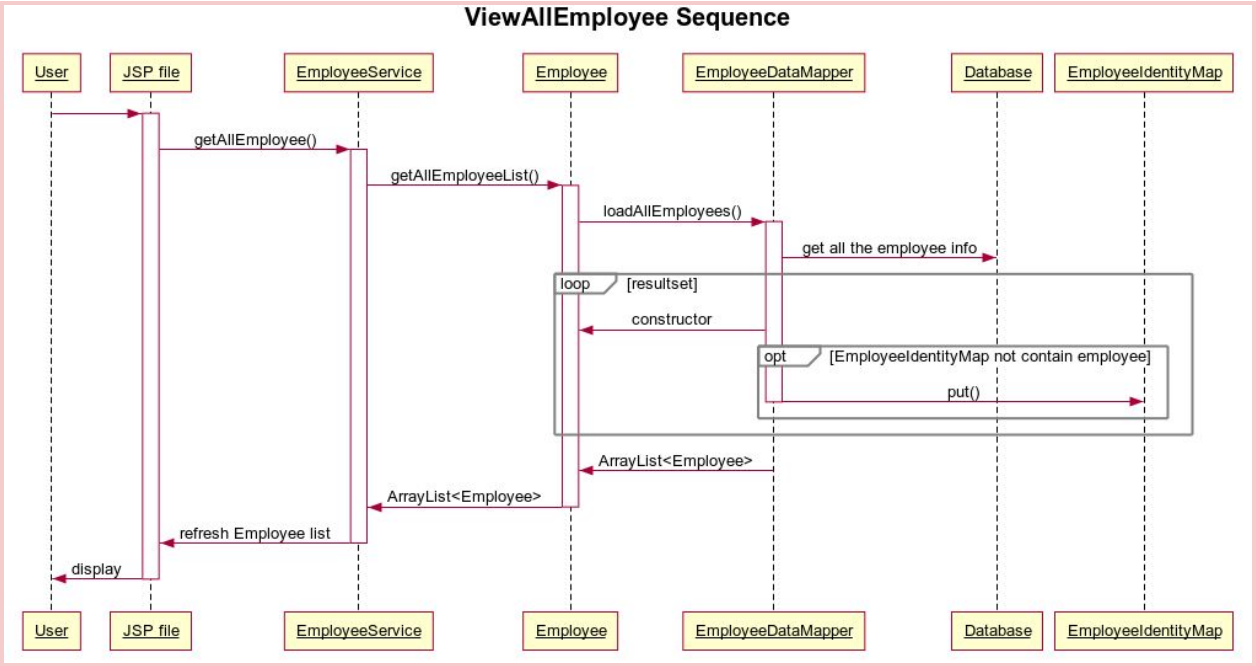

Figure 21 ViewAllEmployee Sequence Diagram

- **Search for a specific employee**

Except for viewing all the employee information, the system also allows its users to search for specific users. The users could enter the id of the user who they want to search for. The id entered will be delivery to the SearchEmployeeServlet as a parameter and at that time, the system will acquire Read Lock. Once the lock was acquired, the system will invoke the corresponding query method in the Employee Service. Similarly, the Employee Service will invoke the method in the Employee Domain Class. To query the required data, the system will first invoke the get method in the Employee IdentityMap. If the system could not find the correct data from the Employee IdentityMap, the system will query the data from the database via the method in the Employee DataMapper. Finally, the queried data will be sent back to the frontend browser.

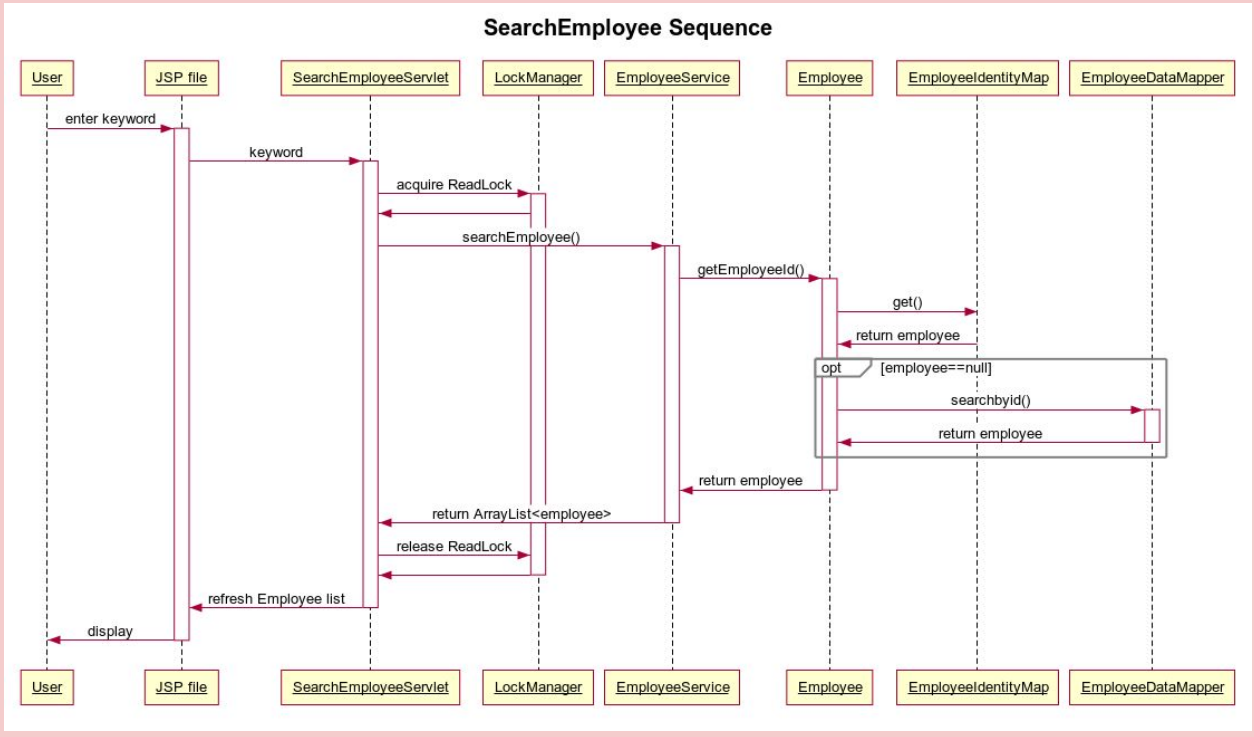

Figure 22 SearchEmployee Sequence Diagram

- **Delete a specific employee personal information**

For the admin users, they have permission to delete the profile of any employees. During the process of delete an existing employee, the system also needs to first acquire Write Lock. If the lock was required, the DeleteEmployeeServlet will invoke the delete method in the EmployeeService. Then the system will first check whether the

employee item exists and invoke the method in the unit of work. Once the unit of work commits, the Employee DataMapper will first check the related attendance records and execute the delete operation. When all these operations finished, the system will release the write lock and update the frontend browser.

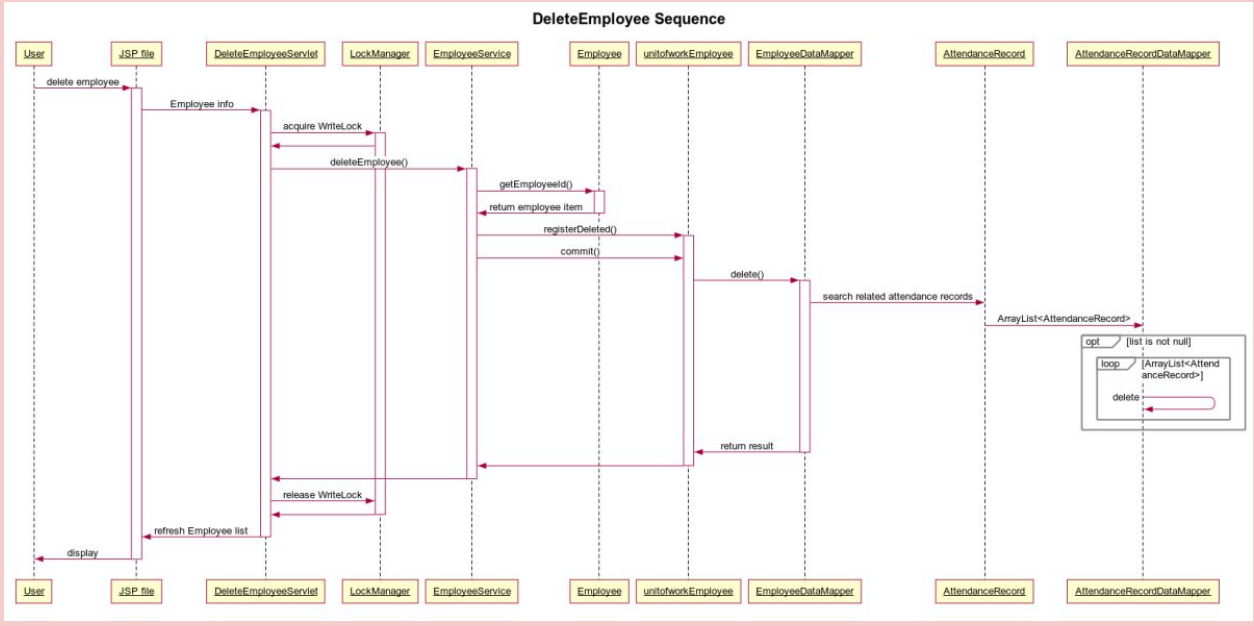

Figure 23 DeleteEmployee Sequence Diagram

- **Create a new employee**

  If the admin users insert a profile of a new employee, the system will get the inputs from the front end and send to the related servlet. Then after acquiring the write lock and validating the correctness of the inputs, the addEmployee method in the EmployeeService will be invoked. Before executing the data inserting operation, the system will first check the department in which the new employee works. If the department exists, the system will invoke the constructor of the Employee Domain Class and then the unit of work of the Employee will invoke registerNew method and commit. The Employee DataMapper will finish the insert operation and add this item into the identity map. Once the insert operation finished, the write lock will be released and the list of the employees displaying on the front end will be updated.

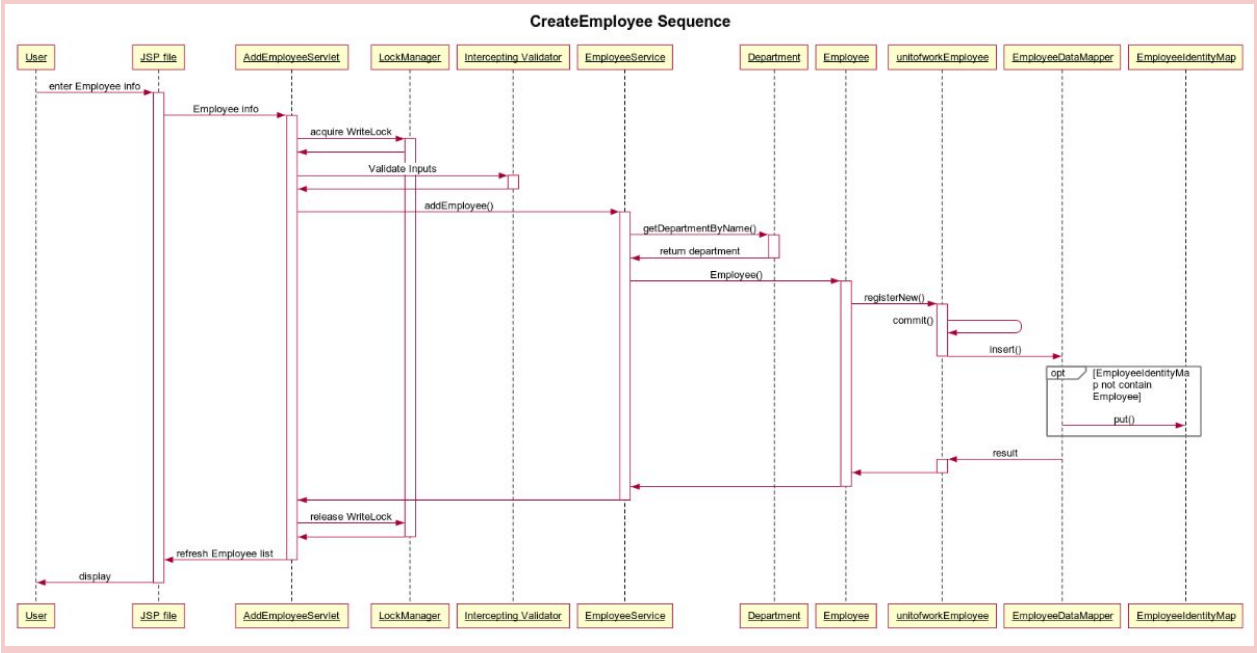

Figure 24 CreateEmployee Sequence Diagram

- **Update an existing employee**

  The process of updating an existing employee is similar to that of creating a new employee. They both acquire the write lock and validate the inputs. The differences between these two operations are that, for the update operation, the system mainly invokes the setter methods of the Employee Domain Class and then invoke the registerDirty method of unitofworkEmployee.

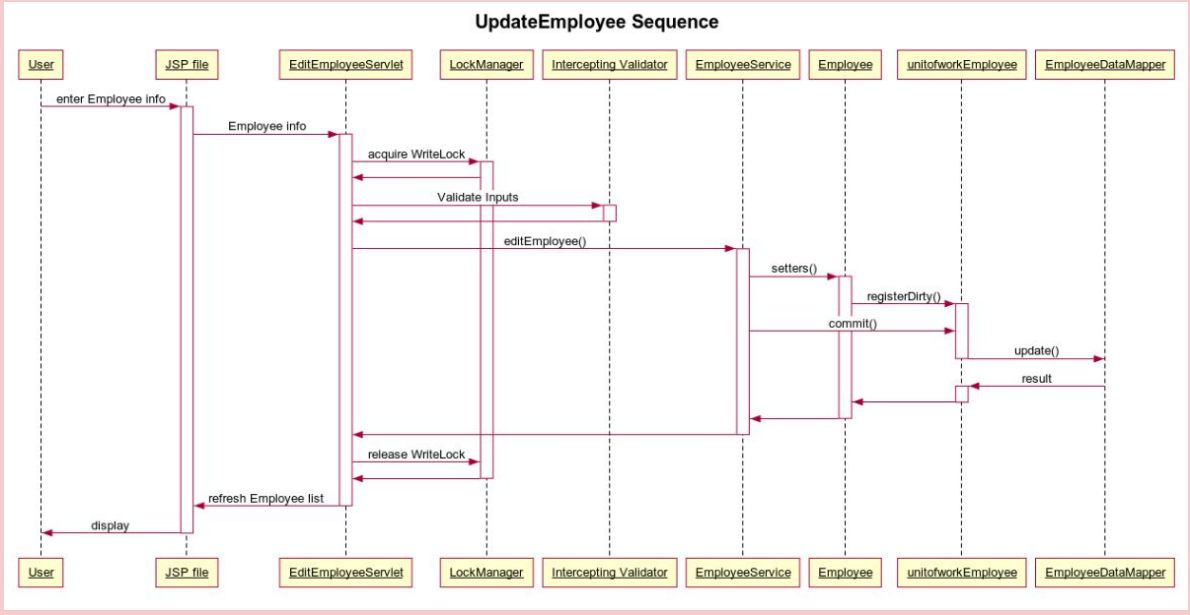

Figure 25 UpdateEmployee Sequence Diagram

- **View the list of employee attendance record in The System**

  For this system, except for the basic department management subsystem and employee management subsystem, the system also allows the employee users to clock on and clock off to achieve the goal of recording the attendance of the employee user. Therefore, the third subsystem will be the Attendance Record Management System. Same as the employee management system and the department management subsystem, the Attendance Record Management System will have a basic function of viewing all the list of attendance records. Similar to the function of viewing the list of all the employee, for this function, the system will call the corresponding methods of AttendanceService, AttendanceRecord Domain Class, AttendanceRecord DataMapper layer by layer and finally get all AttendanceRecord data from the database.

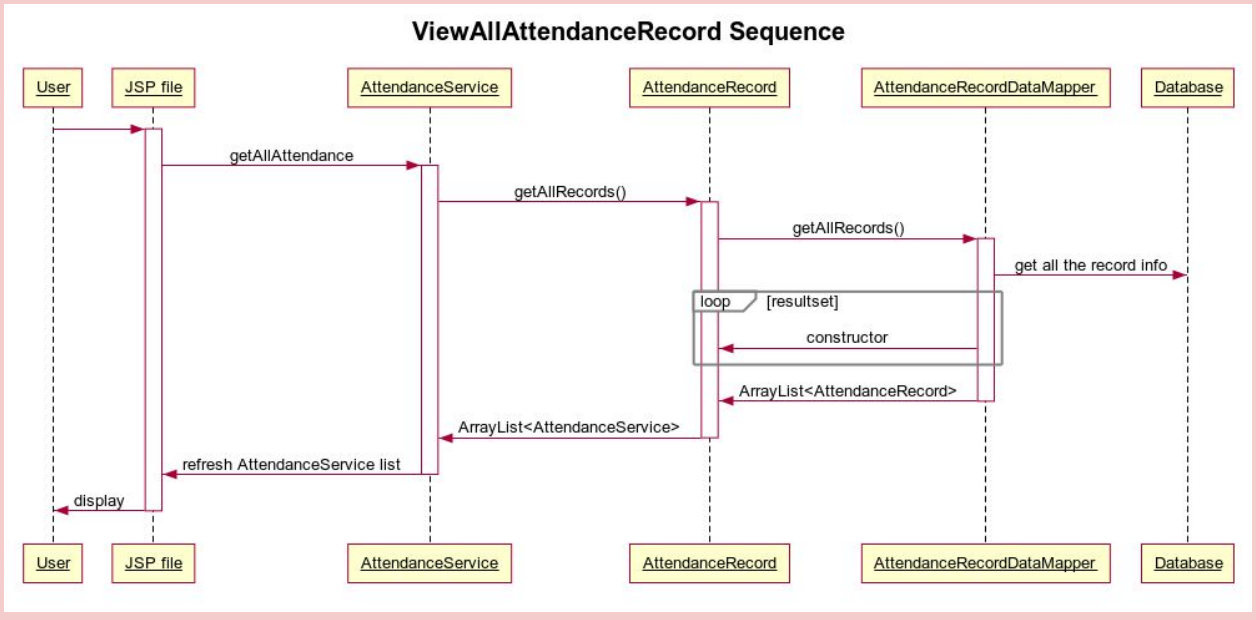

Figure 26 ViewAllAttendanceRecord Sequence Diagram

- Clock On/Clock Off

  To achieve the goal to record the attendance of the employee users, the system has the functions of clocking on and clocking off. In fact, these two functions are a kind of data inserting method for the AttendanceRecord object. Therefore, it also follows a similar process of inserting a new department or employee. The difference between the functions of inserting a new department or employee and this function is that, for clock-on and clock-off operations, the system will first invoke two different methods which are `clockon` method and `clockoff` methods in the corresponding service layer. However, the `clockon` method and `clockoff`

methods will use the same logic and finally insert their data into the same database table.

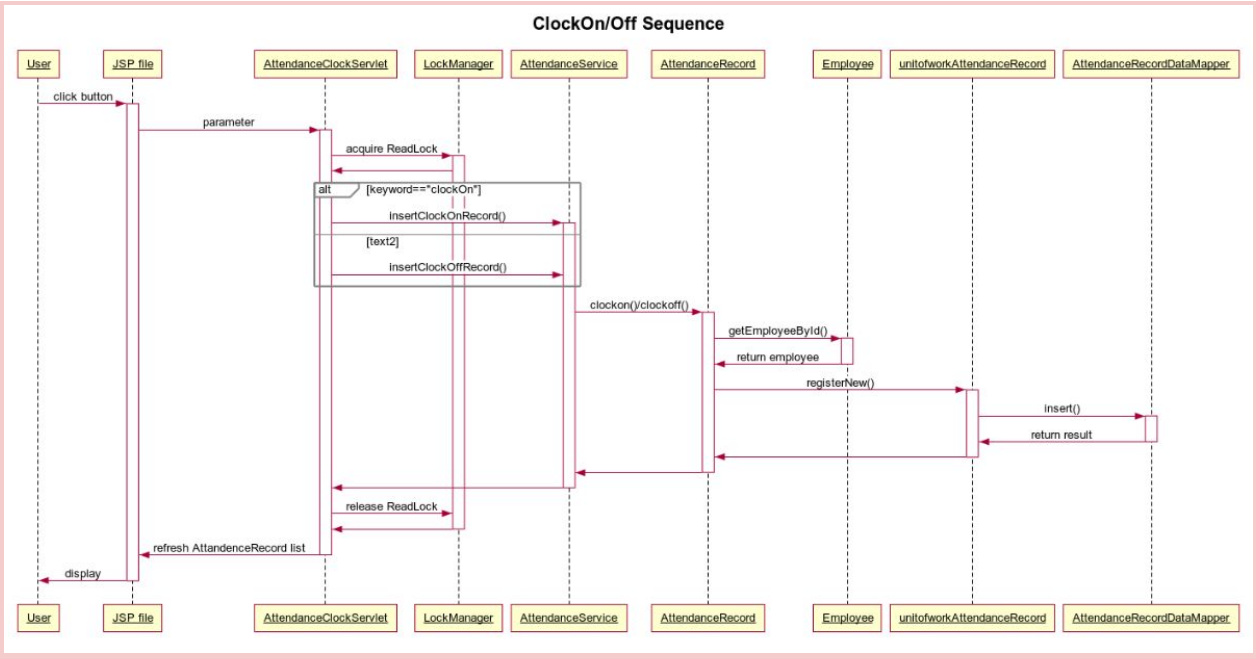

Figure 27 ClockOn/Off Sequence Diagram

Links

Heroku Demo: https://humanresourcesystem.herokuapp.com/
Heroku Project: https://dashboard.heroku.com/apps/humanresourcesystem/resources
Github Repo:
https://github.com/MinchengL/SWEN90007_PROJECT_ASSIGNMENT.git

Test case:

1. Login:
   User ID: 1 password: 1234 role: admin (can login as admin)
   User ID: 2 password: 5678 role: employee (can login as employee)
2. View department
   (can see department list after login)
3. Add department
   (login as admin first)
   Department name: ITSupport
   Phone number: 0412649364
   Location: DMD
   (*Input requirements: Department name should not exist in list before, phone number should match Australian phone number pattern. Otherwise: pop-up alert)
4. Delete department
   Click on “Delete” button of certain row
   (can see item deleted in department list)
5. Edit department
   a. Click on “Edit” button
   b. Enter changed information:
      Department name: HumanResource111
      Phone number: 0412452056
      Location: ERC
   c. Click on “Confirm” button
      (can see updated item in department list)
6. Search department
   a. Enter “1” in search bar
   b. Click on “Search” button
      (can see the search results)
   c. Click on “Clear” button
7. Add employee
   (*Input requirements: Department name should exist in department list already, phone number should match Australian phone number pattern, email address should be legal. Otherwise: pop-up alert)
8. Delete employee
   Click on “Delete” button of certain row
   (can see item deleted in employee list)
9. Edit employee
   (*Input requirements follow 7)

10. Search employee
    (*Search by employee ID)
11. View attendance record
    (can see attendance record list after login)
12. Search attendance record
    (*Search by employee ID)
13. Clock on & Clock off
    Click on “Clock on” or “Clock off” button
    (can see updated attendance record list)
14. Logout
    (Can not access system pages without authentication)