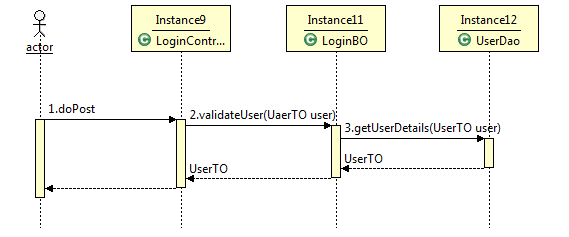
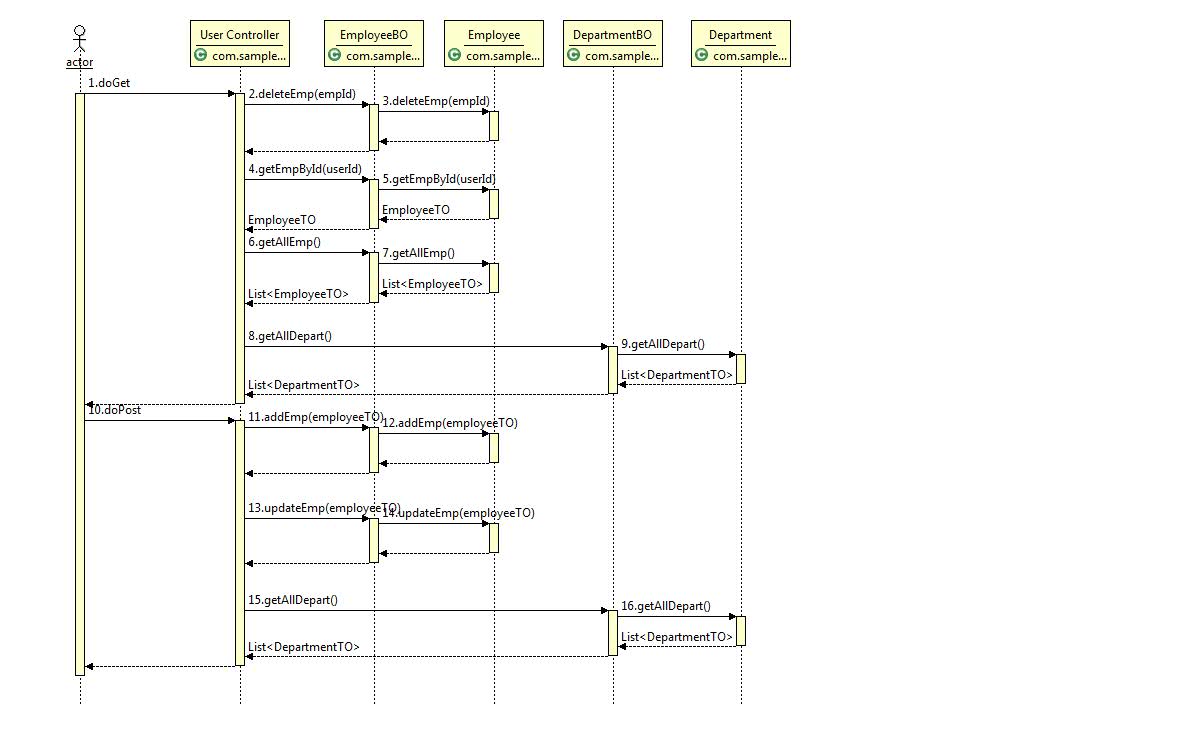
## Flow of Events – Design

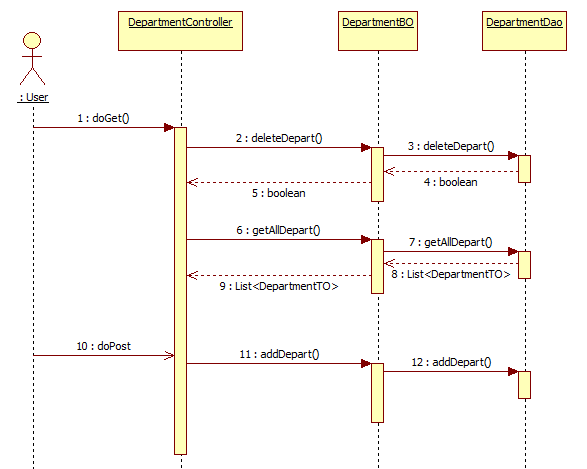
### Login Sequence



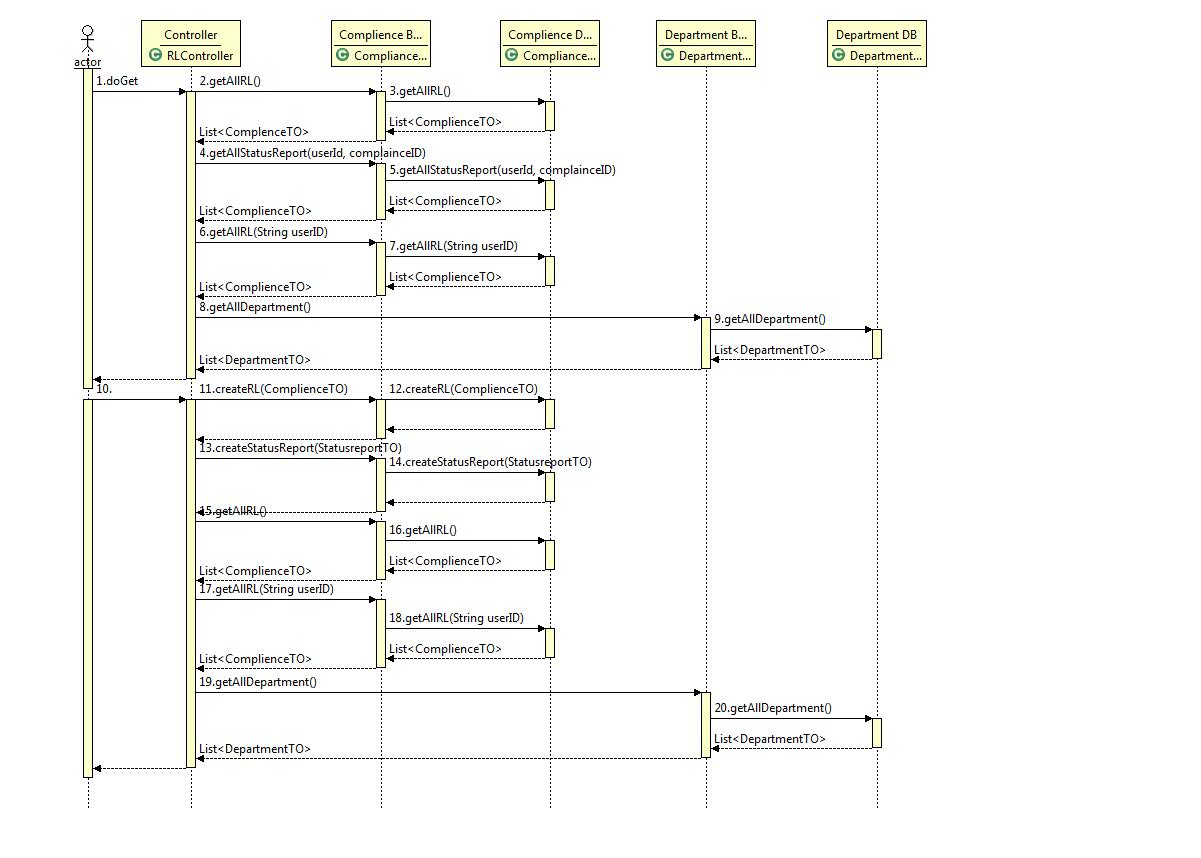
### Employee Sequence



### Department Sequence

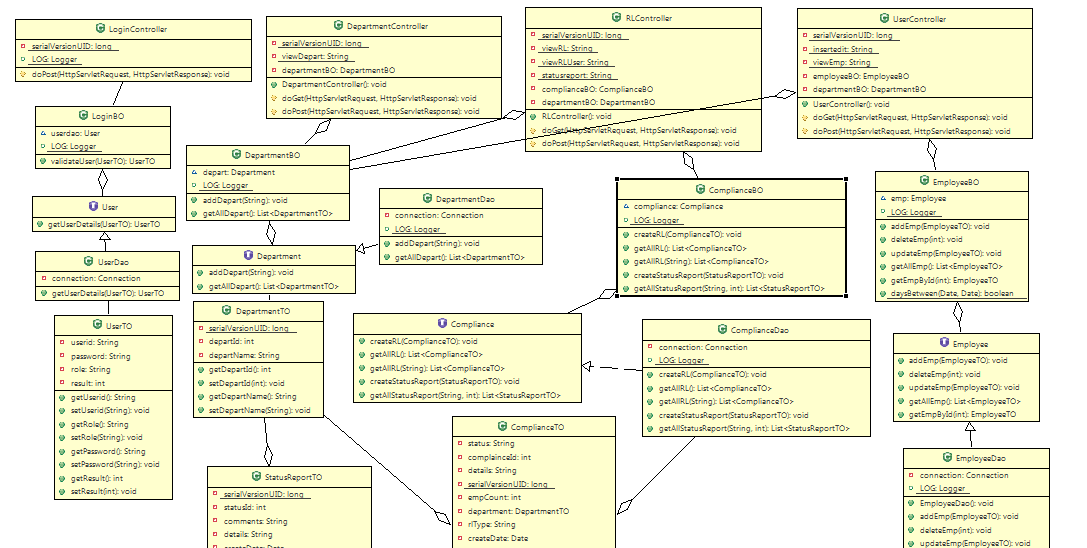


### RL Sequence



# Object Model

## System Object Model



## UI Controls

**Login Employee**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| User ID | Textbox | No | Yes | No | Mandatory | Varchar | 11 |
| Password | Textbox | No | Yes | No | Mandatory | Varchar | 30 |

**Add/Edit Employee**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| First Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 30 |
| Last Name | Textbox | No | Yes | Only Alphabets | Mandatory | Varchar | 30 |
| DOB | Textbox | No | Yes | Valid Date | Mandatory | Date | - |
| Email | Textbox | No | Yes | Only Alphanumeric | Mandatory | Varchar | 50 |
| Department | Combo box | No | Yes | - | Mandatory | Varchar | 30 |

**Add Department**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| Department Name | Textbox | No | Yes | Alphanumeric | Mandatory | Varchar | 30 |

**Create RL**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| Regulation/Legislation Type | Combo box | No | Yes | - | Mandatory | Varchar | 15 |
| Regulation/Legislation Details | Text Area | No | Yes | - | Mandatory | Varchar | 45 |
| Creation Date | Textbox | No | Yes | Valid Date | Mandatory | Date | - |
| Department | Combo box | No | Yes | - | Mandatory | Varchar | 30 |

**Status Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Data Element** | **Control type** | **Default Values** | **Editable Field** | **Restrictions** | **Mandatory/**  **Non Mandatory** | **Data Type** | **Data**  **Size** |
| Comments | Text Area | No | Yes | - | Mandatory | Varchar | 15 |
| Creation Date | Textbox | No | Yes | - | Mandatory | Date |  |

## UI Design

### Login Employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Login Details** | | |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **User ID :** |  |  | |  |
|  |  |  |  |  |  |
|  | **Password :** |  |  |  |  |
|  |  |  |  |  |  |
|  |  | **Login** | |  |  |

### Add/Edit Employee

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Employee Details** | | |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **First Name :** |  |  | |  |
|  |  |  |  |  |  |
|  | **Last Name :** |  |  |  |  |
|  |  |  |  |  |  |
|  | **DOB** |  |  | |  |
|  |  |  |  |  |  |
|  | **Email Address:** | |  | |  |
|  |  |  |  |  |  |
|  | **Department:** |  |  | |  |
|  |  |  |  |  |  |
|  |  | **Submit** | |  |  |

### View Employee

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Emp Id** | **First Name** | **Last Name** | **DOB** | **Email** | **Department** | **Action** |  |
| 11 | Vishal | Shah | 10/31/1983 | [ramya.rathinakumar@Capgemini.com](mailto:ramya.rathinakumar@Cognizant.com) | ADM | Update | Delete |
| 12 | Amol | Patil | 8/11/1983 | [aparna.seshadri@capgemini.com](mailto:aparna.seshadri@cognizant.com) | DCX | Update | Delete |
| 13 | Dinesh | Sharma | 10/19/1981 | [bharath.kumar@capgemini.com](mailto:bharath.kumar@cognizant.com) | ADM | Update | Delete |

### Add/View RL

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | **Create RL Document** | | | |  |
|  |  |  | |  |  |  |
|  |  |  | |  |  |  |
|  | **Regulation/Legislation Type :** |  | |  | |  |
|  |  |  | |  |  |  |
|  | **Regulation/Legislation Details:** |  | |  |  |  |
|  |  |  | |  |  |  |
|  | **Creation Date:** |  | |  | |  |
|  |  | |
|  | **Department:** |  | |  | |  |
|  |  |  | |  |  |  |
|  |  | **Submit** | | |  |  |
|  |  |  | |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RL Id | RL Type | Description | Creation Date | Department | Status |
| 4 | Federal | aaaaa | 2013-May-14 | Finance | Open |
| 5 | Federal | ddddd | 2013-May-14 | Finance | Open |

### View Complience

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Complience Id | Complience Details | Complience Type | Creation Date | Department Name |
| 4 | ddds | Federal | 2013-May-14 | Finance |
| 5 | sdfsd | Federal | 2013-May-14 | Finance |

### Add/View Department

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Department Details** | | | | |  |  |  |
|  |  |  | | |  |  |  |  |
|  |  |  | | |  |  |  |  |
|  | **Name :** |  | | |  |  |  |  |
|  |  |  | | |  |  |  |  |
|  |  |  | | |  | |  |  |
|  |  | **Submit** |  | | | |  |  |
|  |  |  | | |  | |  |  |
|  |  |  | | |  |  |  |  |
|  |  |  | | |  |  |  |  |
|  | Department Id | | | | Department Name | | |  |
|  | 11 | | | | IT | | |  |
|  | 12 | | | | HR | | |  |
|  | 13 | | | | Finance | | |  |
|  |  | | |  |  |  |  |  |

### Add/View Comments

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Add Comments** | | |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **Comment** |  |  | |  |
|  |  |  |  |  |  |
|  | **Creation Date** |  |  |  |  |
|  |  |  |  |  |  |
|  |  | **Submit** | |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Status Report ID | Comments | Creation Date | Department |
| 4 | Federal | 2013-May-14 | Finance |
| 5 | Federal | 2013-May-14 | Finance |

## 

## Component Descriptions

|  |  |  |
| --- | --- | --- |
| **Class** | **Method** | **Program Logic** |
| LoginController | **doPost**  **Parameters**  HttpServletRequest request, HttpServletResponse  **Return Type**  void | * Get the userid, password from request. * Set the user details obtained in the UserTO object. * Call the validateUser() method of LoginBO class. * Set the userId and role in session. * If the obtained role is “ADMIN” then forward to UserController for listing users otherwise redirect to RLController * Forward the response to the view page using a RequestDispatcher if the operation is successfully completed. Else redirect to the error page. |
| LoginBO | **validateUser**  **Parameters**  UserTO user  **Return Type**  UserTO | * Call the getUserDetails() method of UserDao. * Verify the result parameter of UserTO. * If result =0 the invalid user. * If result=1 then invalid password. * Otherwise successful login. |
| UserDao | **getUserDetails**  **Parameters**  UserTO user  **Return Type**  UserTO | * Create a connection object. * Build the delete query. * Create a prepare statement and pass the input parameter userid to the prepare statement. * Execute the query * Iterate through the result set * Get password and role from each result set. * Compare the password given by user and whatever retrived fromm database. * If the passwords are same set the result as 2 else 1. * Set result and role to UserTO. * Return the userTO |
| UserController | **doGet**  **Parameters** HttpServletRequest request, HttpServletResponse response  **Return Type**  *void* | * Get the ‘action type’ from the request. * If the ‘action type’ is ‘delete’ call the deleteEmp() method of the EmployeeBO class by passing the employee id whose information needs to be deleted as the input parameter. * If the ‘action type’ is ‘edit’ call the getEmpById() method of the EmployeeBO class by passing the employee id whose information needs to be updated as the input parameter. * If the ‘action type’ is ‘listUser’ call the getAllEmp() method of the EmployeeBO class. * Set the Employee List obtained in request. * Retrieve the Department list by calling the getAllDepart() method of DepartmentBO class. * Set the Department List obtained in request. * Based on the Action type forward the response to the appropriate page using a RequestDispatcher. * If there is an error while performing the operation redirect to the Error page. |
| **doPost**  **Parameters**  HttpServletRequest request, HttpServletResponse  **Return Type**  void | * Get the ‘action type’ from the request. * Get the firstName, lastName, dob, email, departID, departNm and userid from request. * Set the Employee details obtained in the EmployeeTO object. * Set the Department details obtained in the DepartmentTO object. * If the userid obtained from request is null call the addEmp() method else call the updateEmp() method of EmployeeBO class. * Retrieve the employee list by calling the getAllEmp() method of the EmployeeBO class. * Set the employee list in request. * Retrieve the department list by calling the getAllDepart () method of the DepartmentBO class. * Set the department list in request. * Forward the response to the view page using a RequestDispatcher if the operation is successfully completed. Else redirect to the error page. |
| EmployeeBO | **addEmp**  **Parameters**  EmployeeTO employee  **Return Type**  void | * Retrieve the DOB from the EmployeeTO instance and compare the date with the current date. * If the difference between the DOB and the current date is greater than 24 call the addEmp() of EmployeeDao by passing the employee instance as input parameter. * Else set the message throw a BusinessException. |
| **deleteEmp**  **Parameters**  int empId  **Return Type**  void | * Call the deleteEmp() of EmployeeDao by passing the empId. |
| **updateEmp**  **Parameters**  EmployeeTO employee  **Return Type**  void | * Retrieve the DOB from the EmployeeTO instance and compare the date with the current date. * If the difference between the DOB and the current date is greater than 24 call the updateEmp() of EmployeeDao by passing the employee instance as an input parameter. * Else set the message throw a BusinessException. |
| **getAllEmp**  **Parameters**  None  **Return Type**  List<EmployeeTO> | * Call the getAllEmp() of EmployeeDao |
| **getEmpById**  **Parameters**  int userId  **Return Type**  EmployeeTO | * Call the getEmpById () of EmployeeDao by passing the user id. |
| EmployeeDAO | **addEmp**  **Parameters**  EmployeeTO employee  **Return Type**  void | * Create a connection object. * Build the insert query. * Create a prepare statement and pass the input parameters, first name, last name, dob, email and department id to the prepare statement. * Execute the query. |

|  |  |  |
| --- | --- | --- |
|  | **deleteEmp**  **Parameters**  int empId  **Return type**  void | * Create a connection object. * Build the delete query. * Create a prepare statement and pass the input parameter employee id to the prepare statement. * Execute the query |
| **updateEmp**  **Parameters**  EmployeeTO employee  **Return Type**  void | * Create a connection object. * Build the update query. * Create a prepare statement and pass the input parameters, first name, last name, dob, email and department id to the prepare statement. * Execute the query |
| **getAllEmp**  **Parameters**  None  **Return Type**  List<EmployeeTO> | * Create a connection object. * Build the retrieve query. * Create a prepared statement. * Execute the query * Iterate through the resultset obtained and set the user id, first name last name, dob, email in the EmployeeTO object. * Set the department id and department name in the DepartmentTO object. * Set the DepartmentTO object in the EmployeeTO object. * Add the Employee object to a ArrayList object. * Return the Employee list obtained to the calling method. |
| **getEmpById**  **Parameters**  int userId  **Return Type**  EmployeeTO | * Create a connection object. * Build the retrieve query. * Create a prepared statement and pass the userid as the input parameter. * Execute the query * Iterate through the resultset obtained and set the user id, first name last name, dob, email in the EmployeeTO object. * Set the department id and department name in the DepartmentTO object. * Set the DepartmentTO object in the EmployeeTO object. * Add the Employee object to a ArrayList object. * Return the Employee list obtained to the calling method. |
| DepartmentController | **doGet**  **Parameters**  HttpServletRequest request, HttpServletResponse response  **Return Type**  void | * Call the deleteDepart() method of DepartmentBO class by passing the department Id as the input parameter. * Retrieve the Department list by calling the getAllDepart() method of DepartmentBO class. * Set the Department List obtained in request. * Forward the response to the appropriate page using a RequestDispatcher. * If there is an error while performing the operation redirect to the Error page |
| **doPost**  **Parameters**  HttpServletRequest request, HttpServletResponse response  **Return Type**  void | * Retrieve the department name from request. * Call the addDepart() method of DepartmentBO class by passing the department name as the input parameter. * Retrieve the Department list by calling the getAllDepart() method of DepartmentBO class. * Set the Department List obtained in request. * Forward the response to the appropriate page using a RequestDispatcher. * If there is an error while performing the operation redirect to the Error page. |
| DepartmentBO | **addDepart**  **Parameters**  String departNm  **Return Type**  void | * Call the addDepart() of DepartmentDao by passing the departNm. |
| **deleteDepart**  **Parameters**  int departId  **Return Type**  boolean | * Call the deleteDepart() of DepartmentDao by passing the departId. |
| **getAllDepart**  **Parameters**  None  **Return Type**  List<DepartmentTO> | * Call the getAllDepart () of DepartmentDao. |
| DepartmentDAO | **addDepart**  **Parameters**  String departNm  **Return Type**  void | * Create a connection object. * Build the insert query. * Create a prepare statement and pass the input parameter department name to the prepare statement. * Execute the query. |
|  | **deleteDepart**  **Parameters**  int departId  **Return Type**  boolean | * Create a connection object. * Build the delete query. * Create a prepare statement and pass the input parameter department Id to the prepare statement. * Execute the query. |
|  | **getAllDepart**  **Parameters**  None  **Return Type**  List<DepartmentTO> | * Create a connection object. * Build the retrieve query. * Create a prepared statement. * Execute the query * Iterate through the resultset obtained and set department id and department name in the DepartmentTO object. * Add the DepartmentTO object to a ArrayList object. * Return the department list obtained to the calling method. |
| RLController | **doGet**  **Parameters**  HttpServletRequest request, HttpServletResponse response  **Return Type**  void | * Get the ‘action type’ and departlist from the request. * Get userId and role from session. * If role is not null and role is “ADMIN” then * Call getAllRL() method of ComplienceBO and set the return as records. * Else If the ‘action type’ is ‘createStatusRpt’ then * Get the “compId” from request * Call the getAllStatusReport() of ComplienceBO. Set the ComplieneTO list to session. * Else Call the getAllRL (userID) of ComplienceBO. Set the ComplieneTO list to session. * Based on the Action type forward the response to the appropriate page using a RequestDispatcher. * If there is an error while performing the operation redirect to the Error page. |
| **doPost**  **Parameters**  HttpServletRequest request, HttpServletResponse response  **Return Type**  void | * Get the ‘action type’ and departlist from the request. * Get userId and role from session. * If role is not null and role is “ADMIN” then * Create the ComplianceTO and DepartmentTO with request parameters. * Set the DepartmentTO to ComplienceTO. * Call createRL () method of ComplienceBO passing the ComplianceTO . * Call getAllRL() method of ComplienceBO and set the return in session. * Else If the ‘action type’ is ‘statusrpt’ then * Create StatusReportTO with request parameters. * Call createStatusReport() method of ComplienceBO.\ * Get ‘ComplianceID’ from session. * Call getAllStatusReport() method of ComplienceBO by passinf userID and complienceID. Set the result to request as attribute. * Call getAllRL() method of ComplienceBO by passing the userID. * Set the result as request attribute. * Else * Call the getAllRL(userID) of ComplieneBO and set the result as to request as ‘complianceList’ attribute. * Call the getAllDepart() of DepartmentBO and set the return to request as departments attribute. * Call the getAllRL() method of ComplienceBO and set the return as “records” to request attribute. * Based on the Action type forward the response to the appropriate page using a RequestDispatcher. * If there is an error while performing the operation redirect to the Error page. |
| ComplianceBO | **createRL**  **Parameters**  ComplianceTO complianceTO  **Return Type**  void | * Call the createRL () of ComplianceDao. |
| **getAllRL**  **Parameters**  None  **Return Type**  List<ComplianceTO> | * Call the getAllRL () of ComplianceDao. * For each ComplienceTO in the list, ite checks * If empCount == stsCount then * Set status = “Closed” * Else * Set status = “Open” |
| **getAllRL**  **Parameters**  String userId  **Return Type**  List<ComplianceTO> | * Call the getAllRL (String userId) of ComplianceDao. |
| **createStatusReport**  **Parameters**  StatusReportTO statusReportTO  **Return Type**  void | * Call the getAllRL createStatusReport of ComplianceDao. |
| **getAllStatusReport**  **Parameters**  String userId,int complainceID  **Return Type**  List<StatusReportTO> | * Call the getAllStatusReport of ComplianceDao. |
| ComplianceDao | **createRL**  **Parameters**  ComplianceTO complianceTO  **Return Type**  void | * Create a connection object. * Build the insert query. * Create a prepare statement and pass the input parameter rlType, details, createDate, departIdto the prepare statement. * Execute the query. |
| **getAllRL**  **Parameters**  None  **Return Type**  List<ComplianceTO> | * Create a connection object. * Build the retrieve query. * Create a prepared statement. * Execute the query * Iterate through the resultset obtained and set complianceid, rlType, details, createDate, department\_id, department\_nm, , empcount, statuscount in the ComplianceTO object. * Add the ComplianceTO object to a ArrayList object. * Return the complience list obtained to the calling method. |
| **getAllRL**  **Parameters**  String userId  **Return Type**  List<ComplianceTO> | * Create a connection object. * Build the retrieve query. * Create a prepared statement. * Execute the query * Iterate through the resultset obtained and set complianceid, rlType, details, createDate, department\_id, department\_nm in the ComplianceTO object. * Add the ComplianceTO object to a ArrayList object. * Return the complience list obtained to the calling method. |
| **createStatusReport**  **Parameters**  StatusReportTO statusReportTO  **Return Type**  void | * Create a connection object. * Build the insert query. * Create a prepare statement and pass the input parameter comments, createDate, departId , userId, complienceId to the prepare statement. * Execute the query. |
| **getAllStatusReport**  **Parameters**  String userId,int complainceID  **Return Type**  List<StatusReportTO> | * Create a connection object. * Build the retrieve query. * Create a prepared statement. * Execute the query * Iterate through the resultset obtained and set statusrptid, comments, createDate,department\_id, department\_nm in the StatusReportTO object. * Add the StatusReportTOobject to a ArrayList object. * Return the StatusReportTO list obtained to the calling method. |

# Database Design

# Table Structure

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Table Name | Column Name | Column Discription | Data Type | Data Size | Primary Key | Other Constraint |
| DEPARTMENT | DEPARTMENT\_ID | Column is used for storing Department Id | NUMBER | 10 | YES |  |
| DEPARTMENT\_NAME | Column for storing Department Name | VARCHAR2 | 25 |  |  |
| COMPLIENCE | COMPLIENCEID | Column is used for storing compliance Id | NUMBER | 10 | YES | NOT NULL |
| RLTYPE | Column for storing rltype Name | VARCHAR2 | 15 |  |  |
| DETAILS | Column for storing details | VARCHAR2 | 250 |  |  |
| CREATEDATE | Column for storing date | DATE |  |  |  |
| DEPARTMENT\_ID | Column for storing Department Id | NUMBER | 10 |  | FOREIGN KEY (DEPARTMENT.DEPARTMENT\_ID ) |
| EMPLOYEES | EMPID | Column is used for storing user id. | NUMBER | 10 | YES | Not Null |
| FIRSTNAME | Column for storing the first name. | VARCHAR2 | 45 |  |  |
| LASTNAME | Column is used for storing last name | VARCHAR2 | 45 |  |  |
| DOB | Column is used for date of birth | DATE |  |  |  |
| EMAIL | Column is used for storing email | VARCHAR2 | 100 |  |  |
| DEPARTMENT\_ID | Column is used for storing department id | NUMBER | 10 |  | FOREIGN KEY (DEPARTMENT.DEPARTMENT\_ID ) |
| LOGIN\_MASTER | USERID | Column is used for storing user id. | NUMBER | 10 |  | FOREIGN KEY (EMPLOYEES.EMPID ) |
| PASSWORD | Column is used for storing password | VARCHAR2 | 30 |  |  |
| ROLE | Column is used for storing role | VARCHAR2 | 20 |  |  |
| STATUSREPORT | COMPLIENCEID | Column is used for storing compliance Id | NUMBER | 10 |  | FOREIGN KEY (COMPLIENCE.COMPLIENCEID), NOT NULL |
| STATUSREPORTID | Column is used for storing status report Id | NUMBER | 10 | YES | NOT NULL |
| EMPID | Column is used for storing emp id | NUMBER | 10 |  | FOREIGN KEY (EMPLOYEES.EMPID ),NOT NULL |
| COMMENTS | Column is used for storing comments | VARCHAR2 | 15 |  |  |
| CREATEDATE | Column is used for storing date | DATE |  |  |  |
| DEPARTMENT\_ID | Column is used for storing department Id | NUMBER | 10 |  | FOREIGN KEY (DEPARTMENT.DEPARTMENT\_ID ) |

## E- R Diagram (Relationship between Tables)

# 