

**HUMAN RESOURCE MANAGEMENT SYSTEM**

**Software Design Document**

– Hochiminh City, July 2023 –

| **#** | **Student ID** | **Student Name** | **Dedicated Percent (%)** |
| --- | --- | --- | --- |
| 1 | SE160052 | Mai Xuân Quân | 18% |
| 2 | SE151166 | Nguyễn Thanh Tùng | 17% |
| 3 | SE151143 | Nguyễn Lam Trường | 16% |
| 4 | SE150939 | Lê Hiếu Nghĩa | 16% |
| 5 | SE150307 | Nguyễn Thành Tài | 17% |
| 6 | SE150101 | Hồ Bảo Anh | 16% |
| **SUM** | | | 100% |

**Table of Contents**

[**I. Overview 6**](#_heading=h.gjdgxs)

[1. Code Solution 6](#_heading=h.30j0zll)

[2. Database Schema 7](#_heading=h.1fob9te)

[**II.Use case 9**](#_heading=h.p5rrcpzqvcn)

[1. Diagram 9](#_heading=h.68mez6khmqd0)

[2. Description 9](#_heading=h.scj6rw5zkxpj)

[**III. Code Designs 48**](#_heading=h.3znysh7)

[1. Create Employee 48](#_heading=h.7pkb8oqzqki3)

[a. Class Specifications 48](#_heading=h.pe7o3ja8h99l)

[b. Activity Diagram(s) 49](#_heading=h.ikqtqzs04x45)

[c. Sequence Diagram(s) 49](#_heading=h.9m7djukxbs6c)

[2. Update Employee 50](#_heading=h.83xj5yj5v8yb)

[a. Class Specifications 50](#_heading=h.v1er8dj2qap9)

[b. Activity Diagram(s) 51](#_heading=h.msmsf1jo26d6)

[c. Sequence Diagram(s) 52](#_heading=h.gz24ohrli9rq)

[3. Check Attendance 52](#_heading=h.xusocyiy69gb)

[a. Class Specifications 52](#_heading=h.yuzkkkaj7gei)

[b. Activity Diagram(s) 53](#_heading=h.tlus8vd3pv6z)

[c. Sequence Diagram(s) 54](#_heading=h.wfjzdaukxie8)

[4. Manage Employee’s Tax request 54](#_heading=h.u7zn5o2ifsre)

[a. Class Specifications 54](#_heading=h.tllpivnn58kw)

[b. Activity Diagram(s) 55](#_heading=h.i98rryvikofg)

[c. Sequence Diagram(s) 55](#_heading=h.he2tj5v2ih1z)

[5. View Employee Information 56](#_heading=h.ams9854cl9ne)

[a. Class Specifications 56](#_heading=h.vxtby77l4yqe)

[b. Activity Diagram 56](#_heading=h.7jigk03apf1r)

[c. Sequence Diagram 56](#_heading=h.8qot4d1aq47e)

[6. Change information request 57](#_heading=h.g4g0actgvx5p)

[a. Class Specifications 57](#_heading=h.9f52zh75mb1c)

[b. Sequence Diagram(s) 57](#_heading=h.ps7nbaq3mv43)

[7. View Attendance 58](#_heading=h.jcqtykxcas1p)

[a. Class Specifications 58](#_heading=h.32q4985d7apt)

[b. Sequence Diagram(s) 58](#_heading=h.uog6my5rugav)

[8. Get the OT income information 59](#_heading=h.kue14c7pnqn8)

[a. Class Specifications 59](#_heading=h.8fnff55j74m8)

[b. Activity Diagram 59](#_heading=h.d0hznq1kr33u)

[c. Sequence Diagram 60](#_heading=h.rbtcyygcz34t)

[9.Confirm update profile 60](#_heading=h.czcrcsrifxlo)

[a)Class Specifications 60](#_heading=h.hj9jifuudpuj)

[b)Activity Diagram(s) 61](#_heading=h.8ij3kf7kx9yt)

[c)Sequence Diagram(s) 62](#_heading=h.ux41lgs9jamm)

[10.OT request 62](#_heading=h.5o8mpr3cas36)

[a)Class Specifications 62](#_heading=h.cdpl8eokdula)

[b)Activity Diagram(s) 63](#_heading=h.m1w7dd9x921r)

[c)Sequence Diagram(s) 64](#_heading=h.l3w3timpy087)

[11. Application for leave 64](#_heading=h.va5zus9tay2e)

[a)Class Specifications 64](#_heading=h.869mtiajcmtv)

[b)Activity Diagram(s) 65](#_heading=h.tjmajca3wbg7)

[c)Sequence Diagram(s) 66](#_heading=h.b79binu5fswf)

[12. Resignation letter 66](#_heading=h.fxcos54qmrsj)

[a)Class Specifications 66](#_heading=h.d2n4wxshthpz)

[b)Activity Diagram(s) 67](#_heading=h.nypc6e7zgu90)

[c)Sequence Diagram(s) 68](#_heading=h.x5ahfkx11yri)

[13. Login 69](#_heading=h.2et92p0)

[a. Class Specifications 69](#_heading=h.3dy6vkm)

[b. Sequence Diagram(s) 69](#_heading=h.1t3h5sf)

[14. Logout 70](#_heading=h.ajoshwzhderd)

[a. Class Specifications 70](#_heading=h.rhxe9fg3fnig)

[b. Sequence Diagram(s) 70](#_heading=h.xaiybzh10qnz)

[c. Database queries 71](#_heading=h.g056n4nx29sw)

[15. Approve HR Request 71](#_heading=h.7lcsp5x1jyn6)

[a. Class Specifications 71](#_heading=h.ek9eba6rxjrv)

[b. Sequence Diagram(s) 72](#_heading=h.52itxq6rfy1)

[c. Database queries 72](#_heading=h.9wvim4hfhx7v)

[16. Request leave 72](#_heading=h.2qykf3a4dzp7)

[a. Class Specifications 72](#_heading=h.fwy0776tja4o)

[b. Activity Diagram(s) 73](#_heading=h.t5o4pot0yyek)

[c.State Diagram(s) 74](#_heading=h.1ndwnc12knuc)

[d. Sequence Diagram(s) 75](#_heading=h.gydev6yftaei)

[17. Request resignation 75](#_heading=h.3c7tlytb69oy)

[a. Class Specifications 75](#_heading=h.ecnhvb3p64zo)

[b. Activity Diagram(s) 76](#_heading=h.6pywj1z7jcys)

[c.State Diagram(s) 78](#_heading=h.66w3siugzbf5)

[d. Sequence Diagram(s) 78](#_heading=h.z6hfzbrwi2kj)

[18. Request update attendance 78](#_heading=h.e6fvwmcp01v2)

[a. Class Specifications 78](#_heading=h.a037ktwtiyd1)

[b. Activity Diagram(s) 80](#_heading=h.7jqnp1f3yj3x)

[c.State Diagram(s) 81](#_heading=h.yw9lodv3uou9)

[d. Sequence Diagram(s) 81](#_heading=h.4p7swnmoa687)

[19. Request change work department 81](#_heading=h.rkds2jmevgix)

[a. Class Specifications 81](#_heading=h.3tf4kail1x2h)

[b. Activity Diagram(s) 83](#_heading=h.w3zuvpgifpix)

[c.State Diagram(s) 83](#_heading=h.c36lw32jnlrc)

[d. Sequence Diagram(s) 84](#_heading=h.iyymlcon8na9)

[20. Request tax support 84](#_heading=h.ji8cp7rzf55q)

[a. Class Specifications 84](#_heading=h.ricgjvwlpfv7)

[b. Activity Diagram(s) 86](#_heading=h.9q07c6c711zn)

[c.State Diagram(s) 86](#_heading=h.wg1ltssp1jep)

[d. Sequence Diagram(s) 87](#_heading=h.ly4cd0dv9vt9)

[21. Request OT 87](#_heading=h.holb47rk1ezv)

[a. Class Specifications 87](#_heading=h.2l35dz75jmjg)

[b. Activity Diagram(s) 89](#_heading=h.i678nxqkqdmg)

[c.State Diagram(s) 89](#_heading=h.wnx3d2apd6tp)

[d. Sequence Diagram(s) 90](#_heading=h.vo6uzz6hhcrq)

[**IV. Class Diagram 90**](#_heading=h.ewwhet7h0gw9)

[1. Diagram 90](#_heading=h.9yqecdawjf95)

[2. Class Diagram Specifications 90](#_heading=h.1pz5ghuik8zi)

[2.1 EmployeeController Class 90](#_heading=h.jgajjq5ne8am)

[2.2 EmployeeRepository Class 91](#_heading=h.skprwqqnoa6w)

[2.3 WorkDepartmentRequestController Class 91](#_heading=h.cwu3gfa949b)

[2.4 WorkDepartmentRequestRepository Class 91](#_heading=h.r65biwx72u5b)

[2.5 ResignationController Class 92](#_heading=h.4tlloqaau7xb)

[2.6 ResignationRepository Class 92](#_heading=h.2bdynnwjunpb)

[2.7 OTController Class 92](#_heading=h.lh7ne77m4hx)

[2.8 OTRepository Class 92](#_heading=h.sonbp9idqoxm)

[2.9 TaxController Class 92](#_heading=h.yvugfplny7nf)

[2.10 TaxRepository Class 92](#_heading=h.6e9e1914wbq4)

[2.11AttendanceController Class 92](#_heading=h.9iosiukth8s7)

[2.12 AttendanceRepository Class 93](#_heading=h.5p4zr321s5w3)

[2.13 LeaveRequestController Class 93](#_heading=h.vl9txcceh5f9)

[2.14 LeaveRequestRepository Class 93](#_heading=h.idpdmia9nigh)

[**V. Database Tables 94**](#_heading=h.17dp8vu)

[1. Employee 96](#_heading=h.3rdcrjn)

[2. Department 96](#_heading=h.el5xlwf4oi4u)

[3. ChangeWorkDepartmentRequest 96](#_heading=h.g2dlktjqfq5w)

[4. ResignationRequest 97](#_heading=h.niad0eli32me)

[5. UpdateEmployeeInforRequest 97](#_heading=h.3e4bvluevy72)

[6. TaxRequest 98](#_heading=h.x6df6us5yyyd)

[7. Attendance 99](#_heading=h.9xgne171c9pt)

[8. LeaveRequest 99](#_heading=h.2f7ejy3dvpss)

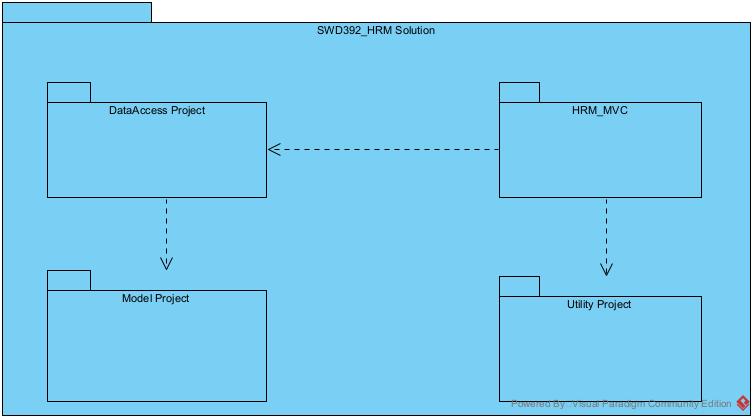
[9. OTRequest 100](#_heading=h.nrp3mwcqm00j)

[10. UpdateAttendanceRequest 100](#_heading=h.cel5k9d6azom)

# 

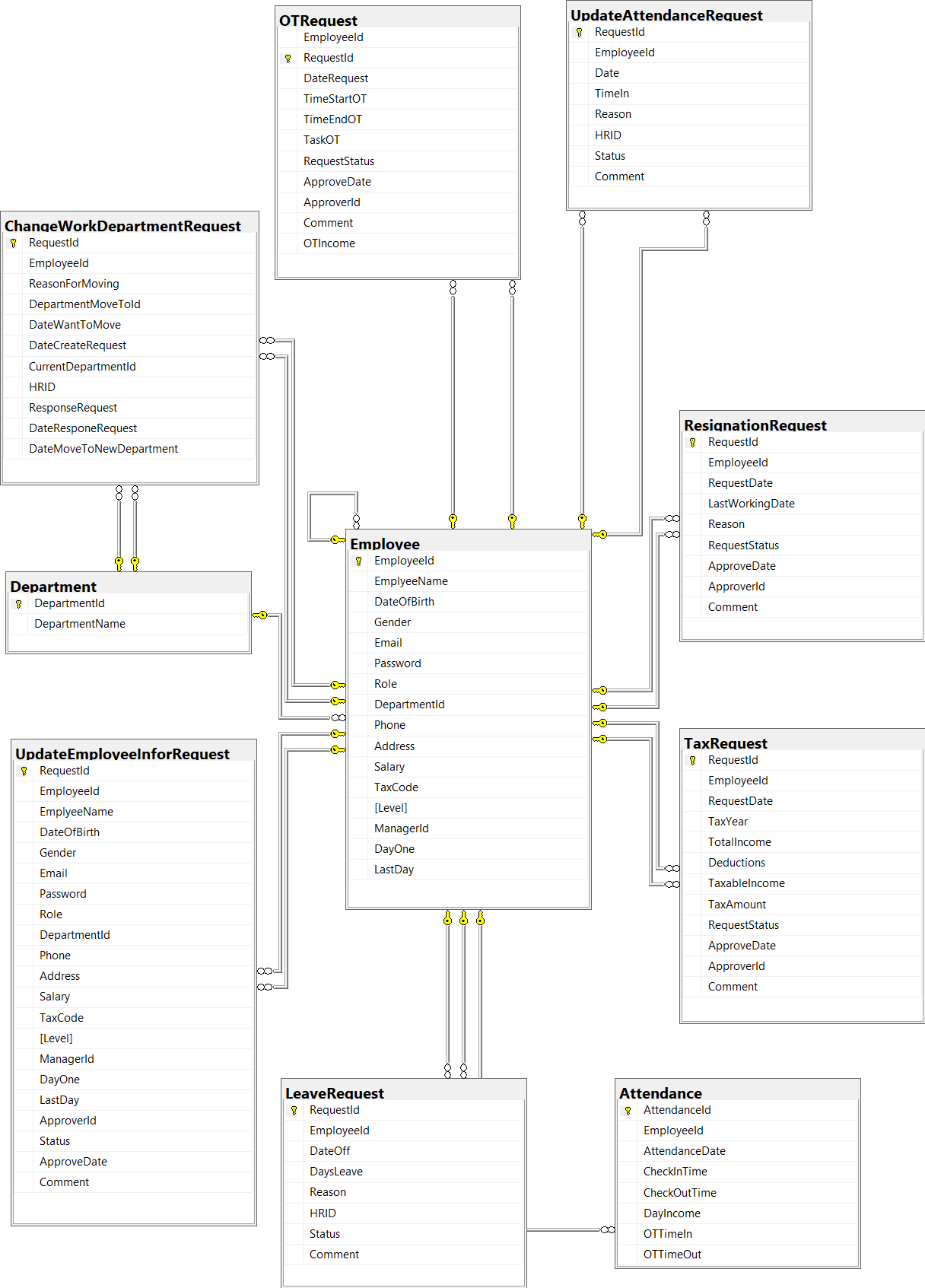
# I. Overview

## 1. Code Solution



| **No** | **Project** | **Description** |
| --- | --- | --- |
| *01* | *SWD392\_HRM Solution* | *Contains all projects code of system* |
| *02* | *Model Project* | *Contains all model classes mapping with database tables and DbContext class to interact with database using Entity Framework* |
| *03* | *DataAccess Project* | *Contains all repository classes* |
| *04* | *HRM\_MVC Project* | *Major part of solution which implements all logics and pages of web application* |
| *05* | *Utility Project* | *Provide reusable code of solution* |

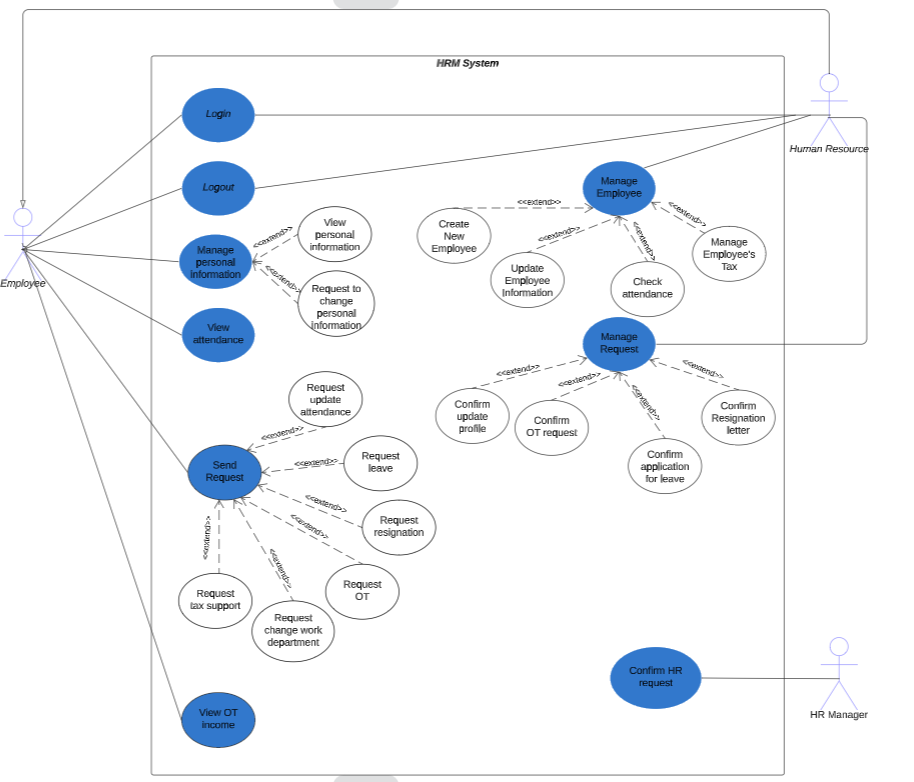
## 2. Database Schema

****

| **No** | **Table** | **Description** |
| --- | --- | --- |
| *01* | *Employee* | *Table contains the employee’s information*  *- Primary keys: EmployeeId*  *- Foreign keys: DepartmentId, ManagerId* |
| *02* | *Department* | *Table contains the department’s information*  *- Primary keys: DepartmentId* |
| *03* | *ChangeWorkDepartmentRequest* | *Table contains the change work department requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: DepartmentMoveToId, CurrentDepartmentId, EmployeeId, HRID* |
| *04* | *ResignationRequest* | *Table contains resignation requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: EmployeeId, ApproverId* |
| *05* | *UpdateEmployeeInforRequest* | *Table contains update personal information requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: EmployeeId, ApproverId* |
| *06* | *TaxRequest* | *Table contains update tax information requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: EmployeeId, ApproverId* |
| *07* | *Attendance* | *Table contains attendance information of employees*  *- Primary keys: AttendanceId*  *- Foreign keys: EmployeeId* |
| *08* | *LeaveRequest* | *Table contains leave requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: EmployeeId* |
| *09* | *OTRequest* | *Table contains overtime requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: EmployeeId* |
| *10* | *UpdateAttendanceRequest* | *Table contains update attendance requests from employees*  *- Primary keys: RequestId*  *- Foreign keys: EmployeeId* |

# II.Use case

## Diagram

****

## Description

| **ID** | **Use Case** | **Actors** | **Use Case Description** |
| --- | --- | --- | --- |
| UC-01 | Login | *Employee* | When Employees want to get into a website to use all functions, they will be forced to login first by their Emails and Passwords. |
| UC-02 | Logout | Employee/Manager | Employees want to exit the account of the system and delete all sessions. |
| UC-03 | View Personal Information | Employee/Manager | This use case allows users to access the system to retrieve their basic personal information. Also allow them to submit a petition to correct the incorrect information |
| UC-04 | View Attendance | Employee | This use case allows users to access the system to retrieve their attendance information for a period of time |
| UC-05 | Request update attendance | Employee | The "Request Update Attendance" use case involves the employee submitting a request to update their attendance record. This use case allows employees to request modifications or corrections to their recorded attendance information, such as clock-in/out times, breaks, or any other attendance-related details. |
| UC-06 | Request leave | Employee | The "Request Leave" use case involves an employee submitting a request for taking leave from work. This use case allows employees to request time off for various reasons, such as vacation, personal matters, medical leave, or any other approved leave type. |
| UC-07 | Request resignation | Employee | The "Request Resignation" use case involves an employee submitting a formal request to resign from their current employment position. This use case allows employees to initiate the resignation process, notify their intention to leave the organization, and provide necessary information for further actions related to the resignation. |
| UC-08 | Request OT | Employee | The "Request Overtime" use case involves an employee submitting a request to work additional hours beyond their regular working schedule. This use case allows employees to formally communicate their need for overtime work and provide relevant information for the approval and processing of the request. |
| UC-09 | Request change work department | Employee | The "Request Change Work Department" use case involves an employee submitting a request to change their assigned work department within the organization. This use case allows employees to communicate their desire for a department transfer and provide necessary information for the request to be reviewed and processed. |
| UC-10 | Request tax support | Employee | The "Request Tax Support" use case involves an employee requesting assistance or support related to tax-related matters from the organization. This use case enables employees to seek guidance, clarification, or resolution for tax-related issues, ensuring compliance and addressing any concerns they may have. |
| UC-11 | View OT income | Employee/Manager | This use case allows users to access the system to retrieve their overtime hours information for a period and their income from there. |
| UC-12 | Approve HR request | HR Manager | This use case describes the process of confirming an HR request by the HR Manager. The HR Manager plays a critical role in reviewing and approving HR requests from the HR department or employees to ensure accuracy and compliance with relevant regulations and procedures. |
| UC-13 | Create New Employee | HR | The "Create New Employee" use case involves the process of onboarding a new employee into an organization's workforce. This use case outlines the necessary steps to create a comprehensive employee profile within the organization's HR or personnel management system. The primary objective is to gather and record accurate information about the new employee, ensuring a smooth transition into their role and integration into the organization. |
| UC-14 | Update Employee Information | HR | The "Update Employee Information" use case involves the process of modifying and updating an existing employee's information in the organization's HR or personnel management system. This use case outlines the necessary steps to ensure accurate and up-to-date employee data, reflecting any changes in personal, professional, or organizational details. |
| UC-15 | Check attendance | HR | The "Attendance Management" use case involves the process of tracking and managing the attendance of employees within an organization. This use case outlines the steps to record and monitor employee attendance, enabling accurate timekeeping, attendance analysis, and payroll processing. |
| UC-16 | Employee's Tax | HR | The "Employee Tax Management" use case involves the process of managing and ensuring compliance with tax obligations related to employees within an organization. This use case outlines the necessary steps to handle various aspects of employee taxation, including tax withholding, reporting, and documentation. |
| UC-17 | ViewAllHRRequests | HR Manager | This use case will display all HR requests in the system, including confirmed or not, so that the HR Manager can control all requests and choose a specific one to respond to. |
| UC-18 | ApproveHRRequest | HR Manager | This use case allows the HR Manager to decide HR requests by changing its status (Approve/Decline) and adding some comment into it. |

| *ID and Name:* | ***UC-01 Login*** | | |
| --- | --- | --- | --- |
| *Created By:* | *Nghĩa* | *Date Created:* |  |
| *Primary Actor:* | Employee | *Secondary Actors:* |  |
| *Description:* | Employees must login to access it by their emails and passwords. | | |
| *Trigger:* | *When they go to the website and want to use a function inside.* | | |
| *Preconditions:* | *Employees must remember their emails and passwords.* | | |
| *Postconditions:* | *Email and Password must be checked correctly.* | | |
| *Normal Flow:* | ***1.0 Login success:***   1. ***Employee/Manager go to the website with the default page is Login Page.*** 2. ***Input email and password into input field.*** 3. ***Go straight to the Home Page.***   ***2.0 Login fail:***   1. ***Employee/Manager go to the website with the default page is Login Page.*** 2. ***Input email and password into input field.*** 3. ***Go back Login Page*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* |  | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* | *Almost all humans in the company must use this everyday.* | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-02 Logout*** | | |
| --- | --- | --- | --- |
| *Created By:* | *Nghĩa* |  |  |
| *Primary Actor:* | Employee/Manager |  |  |
| *Description:* | Logged account (Employee/Manager) wants to exit his/her account from the website to go home or change to another account and all their job/ information will be removed from the session. | | |
| *Trigger:* | *When they want to remove all their information from a website.* | | |
| *Preconditions:* | *Login already* | | |
| *Postconditions:* |  | | |
| *Normal Flow:* | ***1.0 Logout flow:***   1. ***Click on logout button*** 2. ***Confirm logout*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* |  | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* | *Almost all humans in the company must use this everyday.* | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | **UC-03 View personal information** | | |
| --- | --- | --- | --- |
| *Created By:* | Bảo Anh | | |
| *Primary Actor:* | Employee/HR | | |
| *Description:* | This use case allows users to access the system to retrieve their basic personal information. Also allow them to submit a petition to correct the incorrect information | | |
| *Trigger:* | When they want to view their personal information. | | |
| *Preconditions:* | Login already | | |
| *Postconditions:* |  | | |
| *Normal Flow:* | 1. The user clicks on the avatar on the header 2. The profile screen will be shown | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* |  | | |
| *Priority:* | High | | |
| *Frequency of Use:* | Sometimes | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | **UC-04 View attendance** | | |
| --- | --- | --- | --- |
| *Created By:* | Bảo Anh | | |
| *Primary Actor:* | Employee/HR | | |
| *Description:* | This use case allows users to access the system to retrieve their attendance information for a period of time | | |
| *Trigger:* | When they want to view their attendance information. | | |
| *Preconditions:* | Login already | | |
| *Postconditions:* |  | | |
| *Normal Flow:* | 1. The user clicks on the Check Attendance in the submenu. 2. The user choice the period of date that they want to view 3. Click to search button | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* |  | | |
| *Priority:* | High | | |
| *Frequency of Use:* | Almost all humans in the company must use this everyday. | | |
| *Business Rules:* | The date in “from” must be before the date “to” | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-05* Request update attendance** | | |
| --- | --- | --- | --- |
| *Created By:* | *Tài* |  |  |
| *Primary Actor:* | Employee |  |  |
| *Description:* | The Employee will send Request update attendance | | |
| *Trigger:* | *Employee identifies an attendance discrepancy: The employee notices an error or discrepancy in their recorded attendance, such as missing punches, incorrect time entries, or inaccurate attendance data.* | | |
| *Preconditions:* | *Existing attendance records: There should be existing attendance records for the employee that require updates or corrections. Without any existing records, there would be no need for an attendance update request.*  *Employee authentication: The employee must be authenticated and have the necessary permissions to access and submit attendance update requests.* | | |
| *Postconditions:* | *Updated attendance records: The attendance records for the employee are updated based on the requested changes. This could involve modifying the dates, times, or types of attendance entries, such as marking an absence as approved leave or adjusting clock-in/out times.*  *Notification to the employee: The employee is notified about the outcome of their attendance update request. This could be in the form of a confirmation message, email, or notification within the system, informing them whether the request was approved or denied.* | | |
| *Normal Flow:* | ***Employee opens the "Request Update Attendance" page or feature within the system.***  ***The system displays the attendance update form, which includes fields for entering the requested changes, such as date, time, type of attendance, and any relevant notes or comments.***  ***Employee fills out the form with the necessary information, specifying the desired updates to their attendance records.***  ***Employee submits the attendance update request by clicking the "Submit" or similar button.***  ***The system validates the submitted data, ensuring that all required fields are filled and the requested changes are within the allowed parameters.***  ***If the validation is successful, the system proceeds to process the attendance update request.*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* | Invalid or incomplete data: If the employee submits the attendance update form with missing or invalid data, such as leaving required fields empty or providing incorrect date or time values, the system may display an error message indicating the need for valid data entry.  Unauthorized access: If the employee does not have the necessary permissions or access rights to request an attendance update, the system may deny the request and display an error message indicating insufficient privileges. | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* |  | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-06* Request leave** | | |
| --- | --- | --- | --- |
| *Created By:* | *Tài* |  |  |
| *Primary Actor:* | Employee |  |  |
| *Description:* | The Employee will send request leave | | |
| *Trigger:* | *The employee requests to take leave.* | | |
| *Preconditions:* | *The employee must be authenticated and logged into the system.*  *The employee must have the necessary permissions or role to request leave.*  *The system must be accessible and functioning properly.*  *The required information for the leave request, such as start date, end date, and reason, must be known or provided by the employee* | | |
| *Postconditions:* | *The leave request is recorded and stored in the system.*  *The employee's leave balance is updated to reflect the leave request.*  *The leave request is sent for approval to the appropriate authority.*  *The employee receives a confirmation or acknowledgment of the leave request submission.*  *The employee is notified of the status of the leave request (e.g., approved, pending, rejected).* | | |
| *Normal Flow:* | ***The employee opens the leave request page or form.***  ***The system displays the leave request form to the employee, which includes fields for entering details such as leave start date, leave end date, reason for leave, etc.***  ***The employee fills out the required information in the leave request form.***  ***The employee submits the leave request form.***  ***The system validates the entered information to ensure it meets the required criteria (e.g., valid dates, minimum notice period).***  ***If the entered information is valid, the system records the leave request and updates the employee's leave balance.***  ***The system sends a notification or confirmation to the employee, indicating that the leave request has been successfully submitted.*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* | invalid input: If the employee provides invalid or incomplete information in the leave request form (e.g., missing required fields, entering an invalid date format), the system may display an error message and prompt the employee to correct the information.  System error: In case of unexpected system errors, such as database connectivity issues or system crashes, the system may display a generic error message and prompt the employee to try again later or contact technical support. | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* |  | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-07* Request resignation** | | |
| --- | --- | --- | --- |
| *Created By:* | *Tài* |  |  |
| *Primary Actor:* | Employee |  |  |
| *Description:* | The Employee will send Request resignation | | |
| *Trigger:* | *Employee's Resignation Decision* | | |
| *Preconditions:* | *The employee must be authenticated and logged into the system.*  *The employee must be currently employed by the organization: The employee must be an active member of the organization and currently hold a position within it.*  *The system must be accessible and functioning properly.*  *The required information for the Resignation request, such as start date, end date, and reason, must be known or provided by the employee* | | |
| *Postconditions:* | *The leave request is recorded and stored in the system.*  *The employee's leave balance is updated to reflect the leave request.*  *The leave request is sent for approval to the appropriate authority.*  *The employee receives a confirmation or acknowledgment of the leave request submission.*  *The employee is notified of the status of the leave request (e.g., approved, pending, rejected).* | | |
| *Normal Flow:* | ***The employee opens the leave request page or form.***  ***The system displays the leave request form to the employee, which includes fields for entering details such as leave start date, leave end date, reason for leave, etc.***  ***The employee fills out the required information in the leave request form.***  ***The employee submits the leave request form.***  ***The system validates the entered information to ensure it meets the required criteria (e.g., valid dates, minimum notice period).***  ***If the entered information is valid, the system records the leave request and updates the employee's leave balance.***  ***The system sends a notification or confirmation to the employee, indicating that the leave request has been successfully submitted.*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* | invalid input: If the employee provides invalid or incomplete information in the leave request form (e.g., missing required fields, entering an invalid date format), the system may display an error message and prompt the employee to correct the information.  System error: In case of unexpected system errors, such as database connectivity issues or system crashes, the system may display a generic error message and prompt the employee to try again later or contact technical support. | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* |  | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-08* Request OT** | | |
| --- | --- | --- | --- |
| *Created By:* | *Tài* |  |  |
| *Primary Actor:* | Employee |  |  |
| *Description:* | The Employee will send Request OT | | |
| *Trigger:* | *The trigger for the "Request OT" use case is when an employee decides to request overtime work.* | | |
| *Preconditions:* | *The employee must be currently employed by the organization.*  *The employee must have the necessary authorization or permission to request overtime work.*  *The employee must have completed their regular work hours or have valid reasons for requesting overtime beyond their regular schedule.* | | |
| *Postconditions:* | *The employee receives acknowledgement or confirmation of their overtime request submission.*  *The request is forwarded to the appropriate authority or department responsible for reviewing and approving overtime requests.*  *The employee may receive updates or notifications regarding the status of their overtime request, such as approval, rejection, or pending review.*  *If approved, the overtime work schedule is communicated to the employee, including the specific date, time, and duration of the overtime work.* | | |
| *Normal Flow:* | ***The Employee initiates the overtime request process by accessing the RequestOT page or feature in the system.***  ***The system displays the RequestOT form, prompting the Employee to provide the necessary details related to their overtime request, such as the date, start time, end time, reason, and any additional notes.***  ***The Employee fills out the RequestOT form with the required information.***  ***The Employee submits the completed RequestOT form.***  ***The system receives the submitted form and validates the data provided by the Employee.***  ***If the validation is successful, the system proceeds with processing the overtime request.***  ***The system records the overtime request in the database or appropriate storage.***  ***The system generates an acknowledgement or confirmation message, which is sent to the Employee, informing them that their overtime request has been received and is under review.***  ***The system forwards the overtime request to the relevant authority or department responsible for reviewing and approving overtime requests.***  ***The reviewing authority evaluates the overtime request based on factors such as availability of resources, operational needs, employee workload, and compliance with organizational policies.***  ***If the overtime request is approved, the reviewing authority notifies the Employee of the approval and provides details regarding the approved overtime work schedule, including the specific date, start time, end time, and any additional instructions or guidelines.***  ***The Employee acknowledges the approval and confirms their availability for the approved overtime work.*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* | Overtime request rejection: If the reviewing authority determines that the overtime request cannot be approved based on organizational policies, operational constraints, or other factors, the system may notify the Employee about the rejection and provide a reason or explanation for the decision. The Employee may have the opportunity to revise the request or discuss the matter with the appropriate authority.  Invalid or missing data: If the Employee submits the RequestOT form with invalid or incomplete data, such as missing required fields or providing incorrect information, the system may generate an error message and prompt the Employee to correct the errors before resubmitting the form. | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* |  | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-09* Request change work department** | | |
| --- | --- | --- | --- |
| *Created By:* | *Tài* |  |  |
| *Primary Actor:* | Employee |  |  |
| *Description:* | The Employee will send Request change work department | | |
| *Trigger:* | *Employee requests a change in their work department.* | | |
| *Preconditions:* | *There must be an available position or vacancy in the target department.*  *The employee must be an active member of the organization.*  *The employee must have the necessary qualifications or skills required for the target department* | | |
| *Postconditions:* | *The employee's request for a change in the work department is submitted and recorded in the system.*  *The request undergoes the necessary review and approval process by the appropriate authorities.*  *If the request is approved, the employee's work department is updated to the new assigned department.*  *The employee is notified of the outcome of their request, whether it is approved or denied.*  *If the request is denied, the employee's work department remains unchanged.* | | |
| *Normal Flow:* | ***The employee initiates the process by expressing their desire to change their work department.***  ***The employee accesses the "Request Change Work Department" feature through the appropriate interface or application.***  ***The system presents the employee with a form or interface to enter the necessary information related to their work department change request.***  ***The employee fills out the required fields, including their current department, desired department, reason for the change, and any additional details.***  ***Once the employee completes the form, they submit the request.***  ***The system receives the request and validates the entered information.***  ***The request is then routed to the appropriate authority or department responsible for reviewing and approving work department changes.***  ***The reviewing authority assesses the request, considering factors such as organizational needs, available positions, qualifications, and any other relevant criteria.***  ***If the request is approved, the system updates the employee's work department to the newly assigned department.***  ***The system generates a notification or confirmation message to inform the employee of the approved work department change.***  ***The employee receives the notification and is informed about their new work department assignment and any additional instructions, if applicable.***  ***If the request is denied, the system generates a notification or message indicating the denial, providing reasons for the decision, if available.***  ***The employee receives the denial notification and is informed that their work department will remain unchanged.***  ***The process concludes, and the employee's work department status is updated accordingly.*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* | Invalid or incomplete request: If the employee submits an incomplete or invalid request form, the system may prompt them to provide the missing or corrected information before proceeding.  Unauthorized access: If an unauthorized employee attempts to access the "Request Change Work Department" feature, the system should deny access and display an appropriate error message.  Work department change not allowed: There may be certain business rules or restrictions in place that limit or prohibit work department changes. If the requested department change is not allowed, the system should inform the employee and provide an explanation.  Request review and approval process: The review and approval process may involve multiple steps and different authorities. If there are delays or bottlenecks in the approval process, the system should track and communicate the status to the employee, ensuring transparency and managing their expectations.  Conflicting work department requests: In situations where multiple employees request a change to the same department or if there are limited available positions, conflicts may arise. The system should handle such conflicts by applying predetermined rules or criteria to resolve them and notify the affected employees accordingly. | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* |  | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | ***UC-10* Request tax support** | | |
| --- | --- | --- | --- |
| *Created By:* | *Tài* |  |  |
| *Primary Actor:* | Employee |  |  |
| *Description:* | The Employee will send Request tax support | | |
| *Trigger:* | *the employee's recognition of the need for tax support and their decision to seek assistance from the organization's tax support services.* | | |
| *Preconditions:* | *The employee must be a registered member of the organization.*  *The employee must have a valid employee account with appropriate access privileges.*  *The organization must have established tax support services or a designated department responsible for handling tax-related inquiries and assistance.* | | |
| *Postconditions:* | *The employee's request for tax support is recorded and logged in the system.*  *The tax support team is notified about the employee's request and takes appropriate action.*  *The employee receives acknowledgment of their request for tax support.*  *The tax support team provides assistance and guidance to the employee regarding their tax-related inquiries or concerns.*  *The employee's tax-related issues are addressed and resolved to the best extent possible within the scope of the organization's tax support services.*  *The employee receives necessary documentation, advice, or instructions from the tax support team to fulfill their tax obligations effectively.* | | |
| *Normal Flow:* | ***The employee initiates the process by accessing the "Request Tax Support" feature or page.***  ***The system presents the tax support request form to the employee.***  ***The employee fills out the required information in the tax support request form, including their contact details, specific tax-related questions or concerns, and any relevant supporting documentation.***  ***The employee submits the completed tax support request form.***  ***The system validates the form data for completeness and correctness.***  ***If the form data is valid, the system proceeds to record the tax support request and assigns a unique identifier.***  ***The system generates a confirmation message or notification to acknowledge the employee's tax support request and provides them with the assigned request ID for future reference.***  ***The system forwards the tax support request to the appropriate tax support team or personnel responsible for handling such requests.***  ***The tax support team receives the request and reviews the details provided by the employee.*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* | Invalid Form Data: If the employee submits an incomplete or incorrect tax support request form, the system displays an error message highlighting the missing or invalid information and prompts the employee to correct the form before resubmitting.  Technical Failure: If there is a technical failure or system error during the submission or processing of the tax support request, the system displays an error message informing the employee about the issue and advises them to try again later or contact technical support for assistance. | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* |  | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| *ID and Name:* | **UC-11 View OT income** | | |
| --- | --- | --- | --- |
| *Created By:* | Bảo Anh | | |
| *Primary Actor:* | Employee/Manager | | |
| *Description:* | This use case allows users to access the system to retrieve their overtime hours information for a period and their income from there. | | |
| *Trigger:* | When they want to view their OT income information. | | |
| *Preconditions:* | Login already | | |
| *Postconditions:* |  | | |
| *Normal Flow:* | 1. The user clicks on the Income in the submenu. 2. The user chooses the from date and the to date on the headline. 3. The system will be shown the list of attendance. | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* |  | | |
| *Priority:* | High | | |
| *Frequency of Use:* | Almost all humans in the company must use this everyday. | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

| ID and Name: | **UC-12 Approve HR request** | | |
| --- | --- | --- | --- |
| Created By: | Hiếu Nghĩa |  |  |
| Primary Actor: | HR Manager |  |  |
| Description: | This use case describes the process of confirming an HR request by the HR Manager. The HR Manager plays a critical role in reviewing and approving HR requests from the HR department or employees to ensure accuracy and compliance with relevant regulations and procedures. | | |
| Trigger: | The trigger for this use case is when the HR Manager receives an HR request that requires confirmation. | | |
| Preconditions: | * The HR Manager confirms the HR request. * The confirmation result is recorded and communicated to the relevant HR department or employee. | | |
| Postconditions: | * The HR department determines the accurate amount of taxes to be deducted from the employee's salary. * The calculated tax amount is recorded and used for payroll processing and tax reporting purposes. | | |
| Normal Flow: | 1. The HR Manager receives a notification or accesses the HR system to view pending HR requests for confirmation. 2. The HR Manager reviews the detailed information of the HR request, including purpose, content, request date, and other relevant details. 3. The HR Manager evaluates the HR request based on regulations, procedures, policies, and other relevant factors. 4. The HR Manager decides to confirm or reject the HR request. 5. If the request is confirmed:   a. The HR Manager records the confirmation information in the system.  b. The HR Manager notifies the relevant HR department or employee about the confirmation result.   1. If the request is rejected:   a. The HR Manager records the rejection reason in the system.  b. The HR Manager notifies the relevant HR department or employee about the rejection result and provides the reason. | | |
| Alternative Flows: | If the HR Manager requires additional information or consultation from relevant parties:  a. The HR Manager contacts the corresponding HR department or employee to request additional information or consultation.  b. The HR Manager proceeds with the review of the HR request after receiving the necessary information or consultation. | | |
| Exceptions: | Technical failure: If there is a system error or technical issue preventing the confirmation process:  a. The system displays an error message indicating technical failure.  b. The HR Manager can contact the technical support team to resolve the issue or try again later. | | |
| Priority: | High | | |
| Frequency of Use: | Depends on the volume of HR requests and the organization's processes. | | |
| Business Rules: | * Only the HR Manager has the authority to confirm or reject HR requests. * Specific regulations and procedures may apply when reviewing HR requests. | | |
| Other Information: | * The HR system should provide a user-friendly interface and detailed information about HR requests requiring confirmation. * Proper security measures should be in place to protect the confidentiality and integrity of HR information. | | |
| Assumptions: | * The HR Manager has the knowledge and authority necessary to review and confirm HR requests. * The HR system is functional and accessible to perform the HR request confirmation process. | | |

| ID and Name: | **UC-13 Create New Employee** | | |
| --- | --- | --- | --- |
| Created By: | Lam Trường |  |  |
| Primary Actor: | HR |  |  |
| Description: | This use case describes the process of creating a new employee record in the HR management system. The HR department is responsible for adding new employees to the organization's workforce. | | |
| Trigger: | The trigger for this use case is the hiring of a new employee. | | |
| Preconditions: | * The HR department has appropriate access rights and permissions to create new employee records. * The necessary information for creating a new employee record is available, including personal details, contact information, job title, department, and other relevant information. | | |
| Postconditions: | * A new employee record is successfully created in the HR management system. * The employee's information is stored in relevant databases, systems, and reports. | | |
| Normal Flow: | 1. The HR department logs into the HR management system. 2. The HR department navigates to the "New Employee" or "Add Employee" section. 3. The system presents a form to enter the employee's information. 4. The HR department fills in the required information for the new employee, including personal details, contact information, job title, department, etc. 5. The HR department verifies the accuracy of the entered information. 6. The HR department submits the form to create the new employee record. 7. The system validates the entered information and creates a new employee record in the employee database. 8. The system generates a confirmation message indicating the successful creation of the new employee record. | | |
| Alternative Flows: | If there are mandatory fields that are not filled in or contain invalid data:  a. The system highlights the missing or invalid fields and displays an error message.  b. The HR department corrects the fields and resubmits the form. | | |
| Exceptions: | Technical failure: If there is a system error or technical issue preventing the creation of the new employee record:  a. The system displays an error message indicating technical failure.  b. The HR department can try again later or contact the technical support team for assistance. | | |
| Priority: | High | | |
| Frequency of Use: | Moderate to High (Depending on the organization's hiring rate) | | |
| Business Rules: | * Only authorized HR personnel should have access to create new employee records. * Certain fields may have specific validation rules or restrictions (e.g., date format, character limits) that must be followed. | | |
| Other Information: | * The system should provide a user-friendly interface to make it easy for the HR department to create new employee records. * Proper security measures should be in place to protect the confidentiality and integrity of employee data. | | |
| Assumptions: | * The HR department has the necessary information and documentation to accurately create a new employee record. * The HR management system is functional and accessible for performing the create process. | | |

| ID and Name: | **UC-14 Update Employee Information** | | |
| --- | --- | --- | --- |
| Created By: | Lam Trường | Date Created: |  |
| Primary Actor: | HR | Secondary Actors: |  |
| Description: | This use case describes the process of updating employee information by the HR department. The HR department needs to make changes to employee records such as personal details, contact information, job title, department, and other relevant information. | | |
| Trigger: | This use case describes the process of updating employee information by the HR department. The HR department needs to make changes to employee records such as personal details, contact information, job title, department, and other relevant information. | | |
| Preconditions: | * The HR department must have appropriate access rights and permissions to update employee information. * The employee whose information needs to be updated must already exist in the system. | | |
| Postconditions: | * The employee's information is successfully updated in the system. * The updated information is reflected in relevant databases, systems, and reports. | | |
| Normal Flow: | 1. The HR department logs into the HR management system. 2. The HR department navigates to the employee information update section. 3. The system presents a list of employees for whom information can be updated. 4. The HR department selects the employee whose information needs to be updated. 5. The system displays the employee's current information. 6. The HR department makes the necessary changes to the employee's information. 7. The HR department verifies the changes and ensures their accuracy. 8. The HR department submits the updated information. 9. The system validates the updated information and stores it in the employee database. 10. The system generates a confirmation message indicating the successful update of the employee's information. | | |
| Alternative Flows: | 1. If the HR department selects an employee who does not exist in the system:   a. The system displays an error message indicating that the employee does not exist.  b. The HR department can choose to search for the correct employee or cancel the update process.   1. If there are mandatory fields that are not filled in or contain invalid data:   a. The system highlights the missing or invalid fields and displays an error message.  b. The HR department corrects the fields and resubmits the updated information. | | |
| Exceptions: | Technical failure: If there is a system error or technical issue, preventing the update process from completing:  a. The system displays an error message indicating the technical failure.  b. The HR department can try again later or contact the technical support team for assistance. | | |
| Priority: | Medium | | |
| Frequency of Use: | High (HR departments frequently update employee information as needed) | | |
| Business Rules: | * Only authorized HR personnel should have access to update employee information. * Certain fields may have specific validation rules or restrictions (e.g., date format, character limits) that must be followed. | | |
| Other Information: | * The system should provide a user-friendly interface to make it easy for the HR department to update employee information. * Proper security measures should be in place to protect the confidentiality and integrity of employee data. | | |
| Assumptions: | * The HR department has the necessary knowledge and training to update employee information accurately. * The HR management system is functional and accessible for performing the update process. | | |

| ID and Name: | **UC-15 Check attendance** | | |
| --- | --- | --- | --- |
| Created By: | Lam Trường |  |  |
| Primary Actor: | HR |  |  |
| Description: | This use case describes the process of checking the attendance of employees by the HR department. The HR department needs to track and monitor the attendance of employees to ensure compliance with attendance policies and record accurate attendance data. | | |
| Trigger: | The trigger for this use case is the need to check the attendance of employees, which can be on a daily, weekly, monthly, or ad-hoc basis. | | |
| Preconditions: | * The HR department has appropriate access rights and permissions to check employee attendance. * The attendance tracking system is in place and properly configured to record employee attendance. | | |
| Postconditions: | * The HR department obtains accurate attendance data for the selected time period or date. * The attendance data can be used for various purposes such as payroll processing, performance evaluation, and attendance reporting. | | |
| Normal Flow: | 1. The HR department logs into the HR management system. 2. The HR department navigates to the attendance tracking section or module. 3. The HR department selects the desired time period or specific date for which they want to check attendance. 4. The system retrieves and displays a list of employees along with their attendance status (present, absent, late, etc.) for the selected time period or date. 5. The HR department reviews the attendance data and makes any necessary notes or annotations. 6. The HR department may export or generate attendance reports for further analysis or documentation purposes. | | |
| Alternative Flows: | If there is a need to check the attendance of a specific employee:  a. The HR department can search for the employee by name or employee ID.  b. The system retrieves and displays the attendance data specifically for the selected employee. | | |
| Exceptions: | Technical failure: If there is a system error or technical issue preventing the retrieval of attendance data:  a. The system displays an error message indicating the technical failure.  b. The HR department can try again later or contact the technical support team for assistance. | | |
| Priority: | Medium | | |
| Frequency of Use: | High (Attendance tracking is typically a regular task for HR departments) | | |
| Business Rules: | * Only authorized HR personnel should have access to check employee attendance. * Attendance policies and rules should be clearly defined and communicated to employees. | | |
| Other Information: | * The attendance tracking system should be reliable and capable of accurately recording and presenting attendance data. * The system should provide relevant filters and search options to facilitate the HR department's navigation and retrieval of attendance data. | | |
| Assumptions: | * Employees are responsible for accurately recording their attendance through a reliable system or process. * The HR management system is functional and accessible for performing the attendance checking process. | | |

| ID and Name: | **UC-16 Employee's Tax** | | |
| --- | --- | --- | --- |
| Created By: | Lam Trường |  |  |
| Primary Actor: | HR |  |  |
| Description: | This use case describes the process of calculating an employee's tax obligations by the HR department. The HR department needs to determine the appropriate amount of taxes to be deducted from an employee's salary based on their income, tax regulations, and other relevant factors. | | |
| Trigger: | * The trigger for this use case is the need to calculate an employee's tax obligations, typically on a regular basis such as monthly or bi-monthly. | | |
| Preconditions: | * The HR department has appropriate access rights and permissions to calculate employee taxes. * The necessary information for tax calculation is available, including employee's income, tax brackets, deductions, allowances, and other relevant data. | | |
| Postconditions: | 1. The HR department determines the accurate amount of taxes to be deducted from the employee's salary. 2. The calculated tax amount is recorded and used for payroll processing and tax reporting purposes. | | |
| Normal Flow: | 1. The HR department accesses the tax calculation module or system. 2. The HR department selects the employee for whom tax calculation is to be performed. 3. The system retrieves the employee's income information and applicable tax rules. 4. The HR department verifies the accuracy of the income and other relevant data. 5. The HR department applies the relevant tax brackets, deductions, and allowances to calculate the employee's taxable income. 6. The HR department calculates the tax amount based on the applicable tax rates and formulas. 7. The HR department verifies the calculated tax amount and ensures its accuracy. 8. The calculated tax amount is recorded in the employee's payroll record for deduction purposes. | | |
| Alternative Flows: | If there are specific tax exemptions or adjustments applicable to the employee:  a. The HR department applies the exemptions or adjustments as per the tax regulations and guidelines. | | |
| Exceptions: | Technical failure: If there is a system error or technical issue preventing the tax calculation process:  a. The system displays an error message indicating the technical failure.  b. The HR department can try again later or contact the technical support team for assistance. | | |
| Priority: | High | | |
| Frequency of Use: | * Regular (Tax calculation is performed on a recurring basis as part of payroll processing) | | |
| Business Rules: | * Only authorized HR personnel should have access to calculate employee taxes. * Tax regulations and rules should be followed accurately to ensure compliance with tax laws. | | |
| Other Information: | * The tax calculation system should be up-to-date with the latest tax regulations and rates. * Proper security measures should be in place to protect the confidentiality and integrity of employee tax information. | | |

| *ID and Name:* | ***UC-17 Approve HRRequest*** | | |
| --- | --- | --- | --- |
| *Created By:* | *Nghĩa* |  |  |
| *Primary Actor:* | HR Manager |  |  |
| *Description:* | The HR Manager will accept all HR requests and decide to approve or decline those requests. | | |
| *Trigger:* | *The HR Manager reads the HR request and decides to approve or decline that request by changing its status.* | | |
| *Preconditions:* | *Login first and user must be HR Manager* | | |
| *Postconditions:* |  | | |
| *Normal Flow:* | ***1.0 Login success and role must be HR Manager:***   1. ***View all HR requests*** 2. ***Select a request to view detail*** 3. ***Change status request by approve or decline*** 4. ***Send back request to HR*** | | |
| *Alternative Flows:* |  | | |
| *Exceptions:* |  | | |
| *Priority:* | *High* | | |
| *Frequency of Use:* | *Depends on how much HR in company.* | | |
| *Business Rules:* |  | | |
| *Other Information:* |  | | |
| *Assumptions:* |  | | |

# III. Code Designs

## 1. Create Employee

### a. Class Specifications

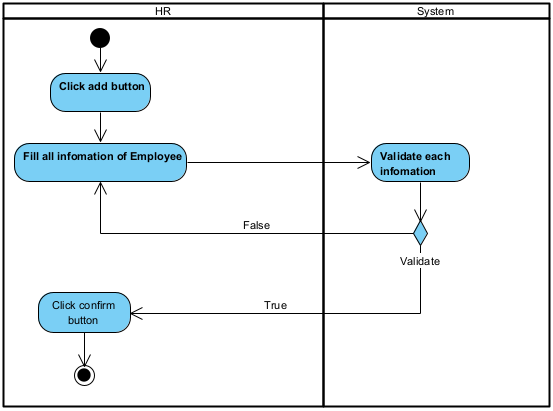
***Employee Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *createEmployee()* | *Creates a new employee record in the database.* |

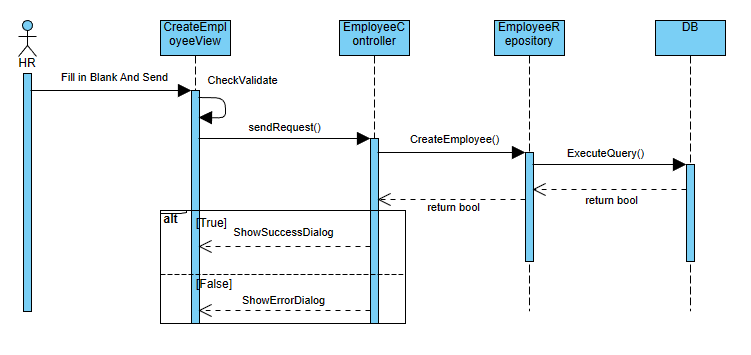
Method 01: createEmployee()

* Description: This method is responsible for creating a new employee record in the database. It takes the necessary input parameters and performs the following steps:
  1. Validate the input parameters: Ensure that the provided employee details are valid and complete.
  2. Generate a unique employee ID: Generate a unique identifier for the new employee.
  3. Create a new employee object: Create a new instance of the Employee class with the provided details and the generated employee ID.
  4. Save the employee record: Persist the employee object to the database.
  5. Return the result: Return a boolean value indicating whether the employee creation was successful or not.

### b. Activity Diagram(s)



### c. Sequence Diagram(s)



## 2. Update Employee

### a. Class Specifications

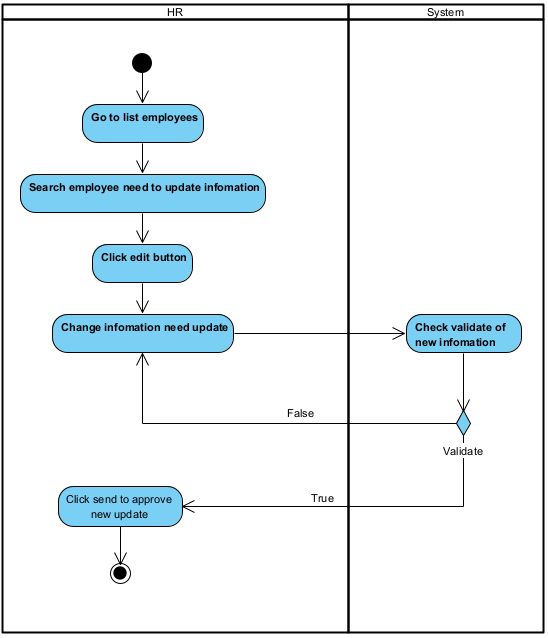
***Employee Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *updateEmployeeInfor()* | *Updates the information of an existing employee in the database.* |

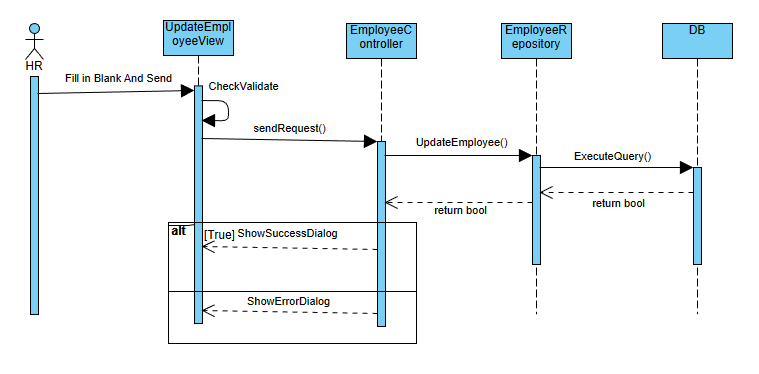
Method 01: updateEmployee ()

* Description: This method is responsible for updating the information of an existing employee in the database. It takes the employee ID and the updated details as input and performs the following steps:
  1. Validate the input parameters: Ensure that the provided employee ID is valid and that at least one updated detail is provided.
  2. Check if the employee exists: Query the database to check if an employee with the provided ID exists.
  3. Update the employee information: Update the employee record in the database with the provided details.
  4. Return the result: Return a boolean value indicating whether the employee information was successfully updated or not.

### b. Activity Diagram(s)



### c. Sequence Diagram(s)



## 3. Check Attendance

### a. Class Specifications

***Employee Class***

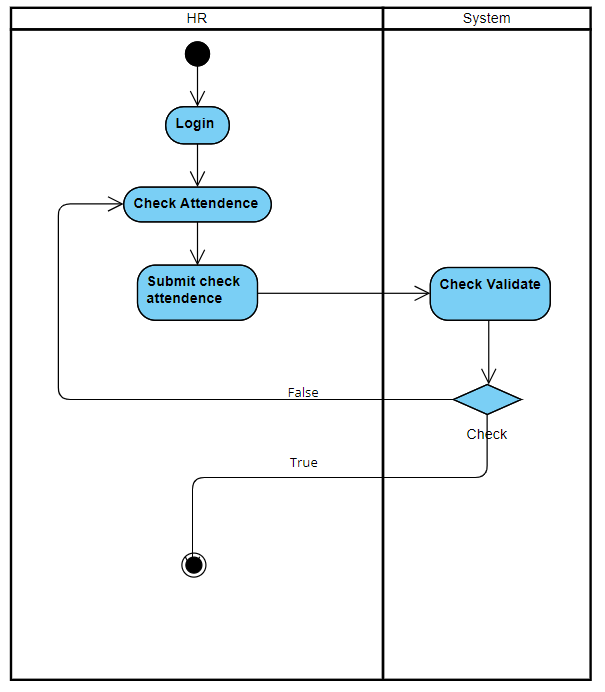
| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *checkAttendance()* | *Retrieves the attendance information for an employee.* |

Method 01: checkAttendance()

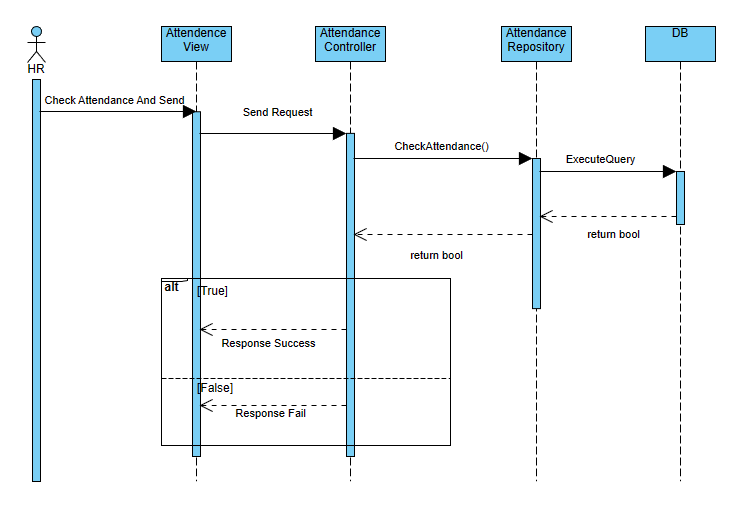
* Description: This method is responsible for retrieving the attendance information for a given employee. It takes the employee ID and the date range as input and performs the following steps:

1. Validate the input parameters: Ensure that the provided employee ID is valid and the date range is properly formatted.
2. Query the attendance records: Retrieve the attendance records from the database based on the employee ID and the specified date range.
3. Return the attendance information: Return the retrieved attendance records as a result.

### b. Activity Diagram(s)

****

### c. Sequence Diagram(s)



## 4. Manage Employee’s Tax request

### a. Class Specifications

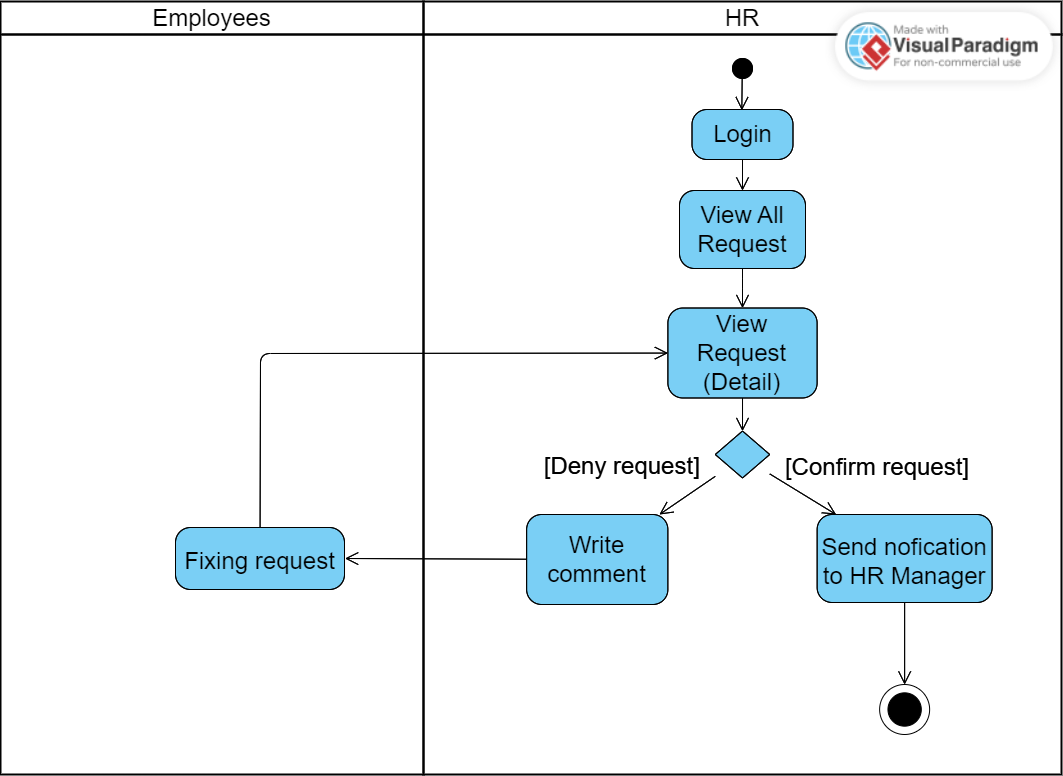
***Employee Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *getEmployeeTax()* | *Retrieves the tax information for an employee.* |

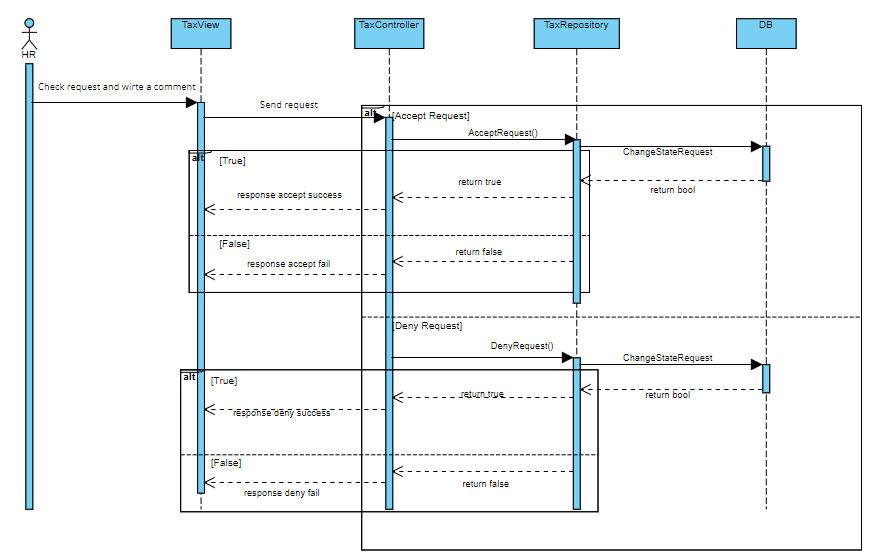
Method 01: getEmployeeTax()

* Description: This method is responsible for retrieving the tax information for a given employee. It takes the employee ID as input and performs the following steps:
  1. Validate the input parameter: Ensure that the provided employee ID is valid.
  2. Query the database: Retrieve the tax information for the employee from the database based on the employee ID.
  3. Return the tax information: Return the retrieved tax information as a result.

### b. Activity Diagram(s)



### c. Sequence Diagram(s)



## 5. View Employee Information

### a. Class Specifications

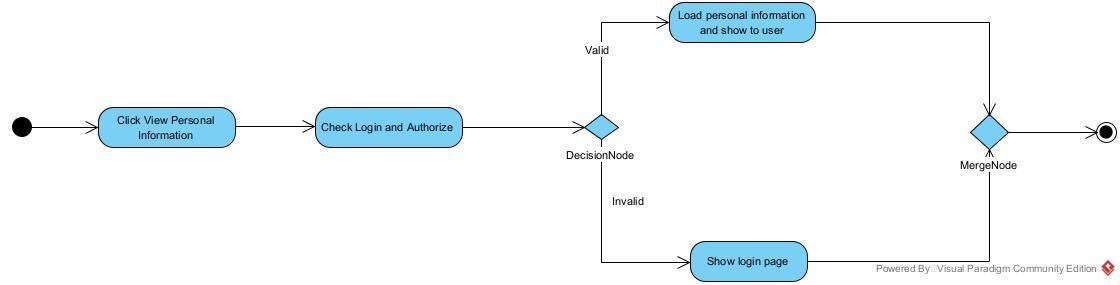
***Employee Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getEmployeeInfo() | Get the employee information record in the database. |

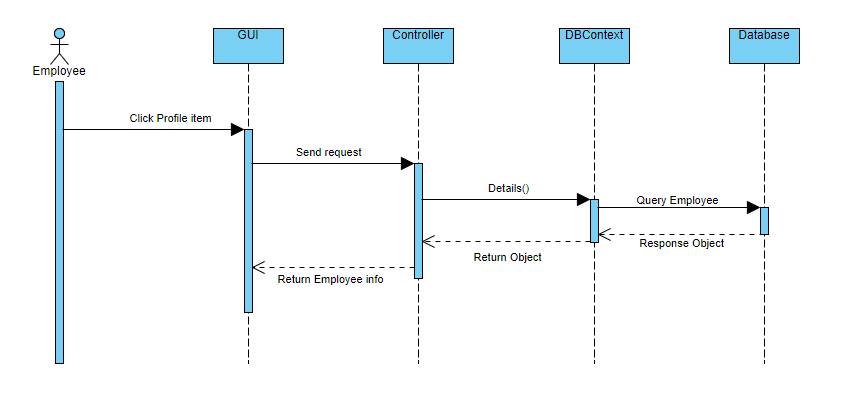
Method 01: getEmployeeInfo()

* Description: This method is responsible for getting the employee information record in the database. It takes the necessary input parameters and performs the following steps:
  1. Validate the input parameters: Ensure the provided employee’s ID is valid and complete.
  2. Query employee information: Query the employee in the database by employee’s ID.
  3. Return the result: Return an employee object if exists or return null if not exist.

### b. Activity Diagram



### c. Sequence Diagram



**c. Database queries**

SELECT \* FROM Employee WHERE employeeID=?

## 6. Change information request

### a. Class Specifications

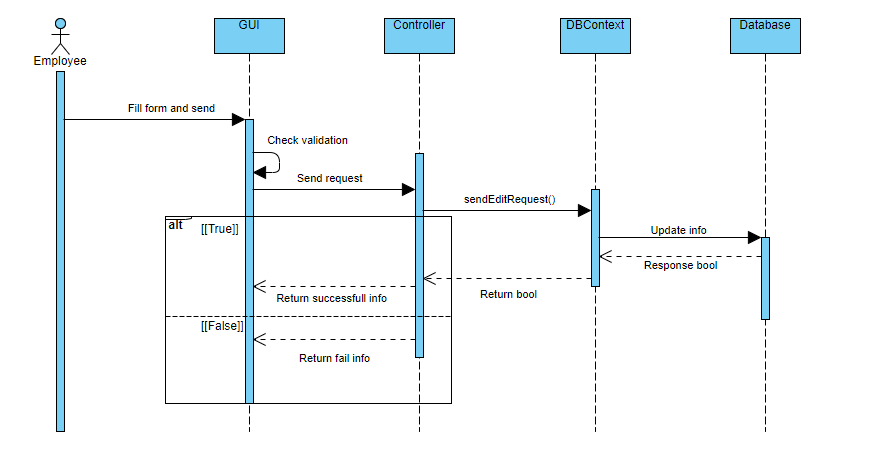
***Employee Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | createChangeInfoRequest() | Create the change information request of an existing employee in the database and send it to the HR department. |

Method 01: createChangeInfoRequest()

* Description: This method is responsible for creating the change information request of an existing employee in the database. It takes the employee ID and the updated details as input and performs the following steps:
  1. Validate the input parameters: Ensure that the provided employee ID is valid and that at least one updated detail is provided.
  2. Check if the employee exists: Query the database to check if an employee with the provided ID exists.
  3. Generate the new request ID: Generate a unique identifier for the new request.
  4. Create a new request object: Create a new instance of the Request class with the provided details.
  5. Return the result: Return a boolean value indicating whether the change information request was successfully create or not.

### b. Sequence Diagram(s)



**c. Database queries**

INSERT INTO ChangeInfoRequest VALUES(?,?,?,?)

## 7. View Attendance

### a. Class Specifications

***Employee Class***

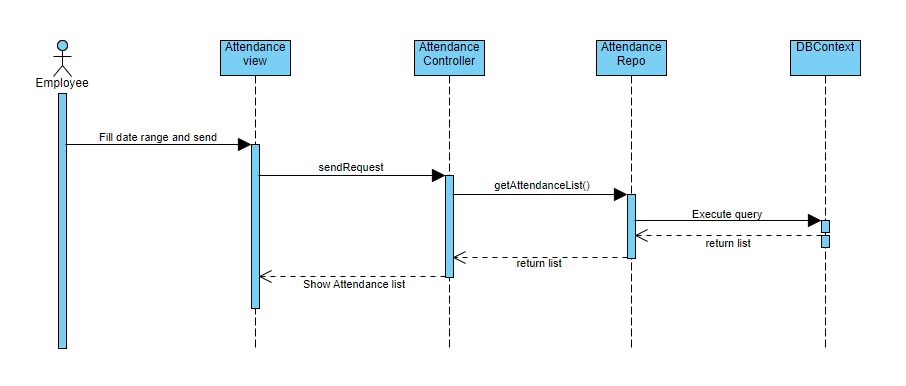
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAttendanceList() | Get the list of attendance information of an employee |

Method 01: getAttendanceList()

* Description: This method is responsible for retrieving all the attendance information for an employee. It takes the employee ID and the specified date range as input and performs the following steps:

1. Validate the input parameters: Ensure the provided employee ID is valid and the date range is formatted correctly.
2. Query the attendance records: Retrieve the records from the database based on the employee ID and the specified date range.
3. Return the attendance information: Return the retrieved attendance records as a result.

### b. Sequence Diagram(s)



**c. Database queries**

SELECT \*

FROM Attendance

WHERE EmployeeID=? AND Date >=? AND Date <=?

## 8. Get the OT income information

## 

### a. Class Specifications

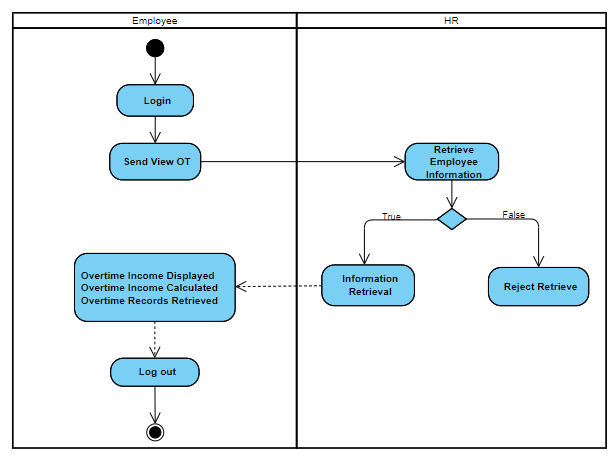
***Employee Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getOTIncome() | Retrieves the OT income information of an employee. |

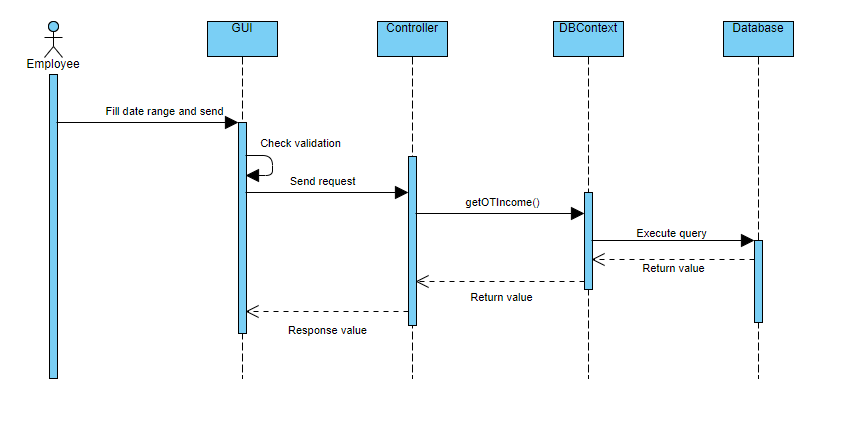
Method 01: getEmployeeIncome()

* Description: This method is responsible for retrieving the OT income information of an employee. It takes the employee ID as input and performs the following steps:
  1. Validate the input parameter: Ensure that the provided employee ID is valid.
  2. Query the database: Retrieve the OT income information of the employee from the database based on the employee ID.
  3. Return the tax information: Return the retrieved OT income information as a result.

### b. Activity Diagram



### c. Sequence Diagram



**c. Database queries**

SELECT \*

FROM OTRequest

WHERE EmployeeID=?

## 9.Confirm update profile

### a)Class Specifications

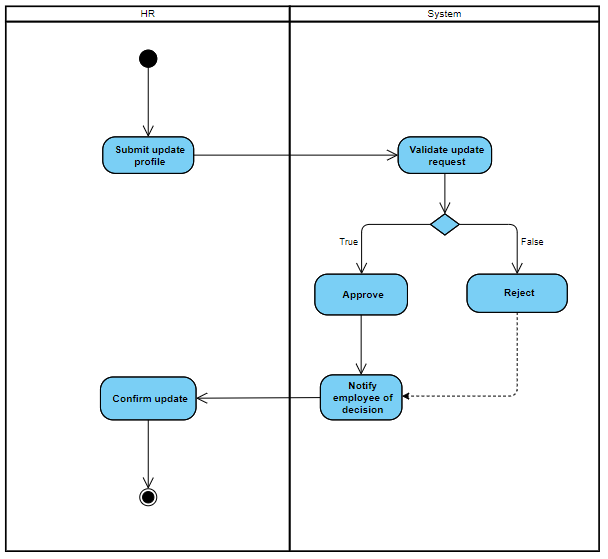
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | confirmUpdateProfile() | This method is responsible for confirming the update of an employee's profile. |

Method: confirmUpdateProfile()

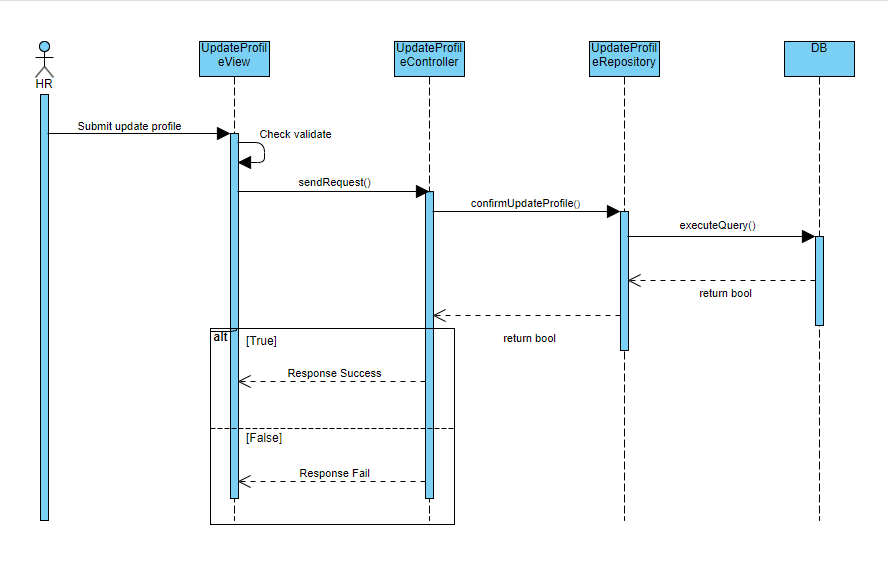
Description: This method is responsible for confirming the update of an employee's profile. It takes the necessary input parameters and performs the following steps:

* Validate the input parameters: Ensure that the provided employee details are valid and complete.
* Retrieve the employee record: Fetch the employee record from the database using the provided employee ID.
* Update the employee details: Update the relevant fields of the employee record with the new information provided.
* Save the updated record: Persist the modified employee record to the database.
* Return the result: Return a boolean value indicating whether the profile update was successful or not.

### b)Activity Diagram(s)



### c)Sequence Diagram(s)



**d)Database queries**

## 10.OT request

### a)Class Specifications

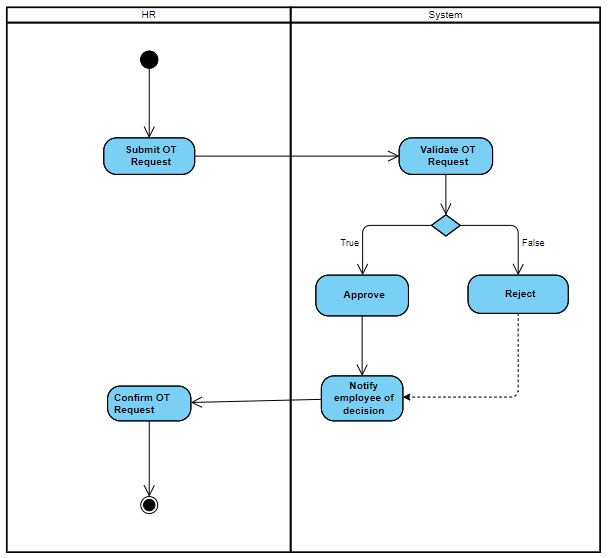
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | submitOTRequest() | This method is responsible for submitting an overtime (OT) request for an employee. |

Method: submitOTRequest()

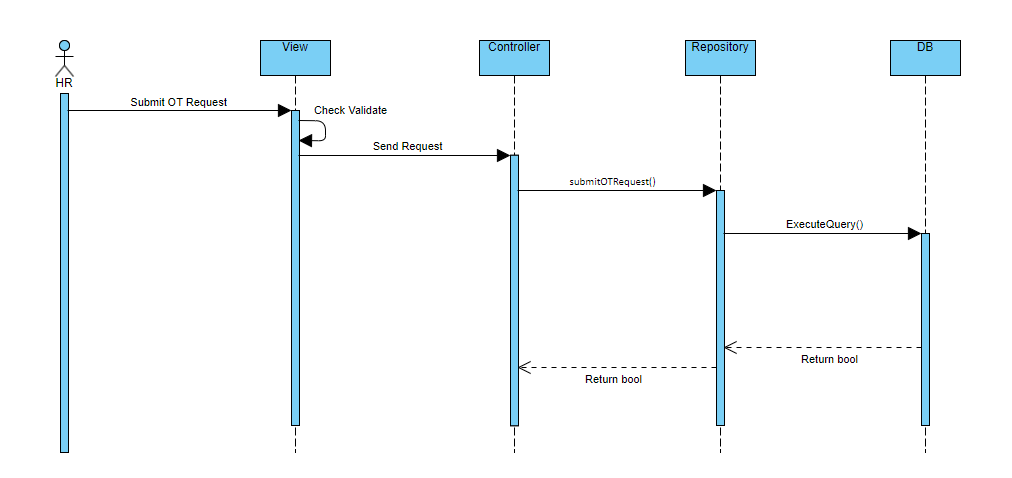
Description: This method is responsible for submitting an overtime (OT) request for an employee. It takes the necessary input parameters and performs the following steps:

* Validate the input parameters: Ensure that the provided OT details are valid and complete.
* Retrieve the employee record: Fetch the employee record from the database using the provided employee ID.
* Create a new OT request object: Create a new instance of the OTRequest class with the provided details and associate it with the employee.
* Submit the OT request: Save the OT request object to the database.
* Return the result: Return a boolean value indicating whether the OT request submission was successful or not.

### b)Activity Diagram(s)



### c)Sequence Diagram(s)



**d)Database queries**

## 11. Application for leave

### a)Class Specifications

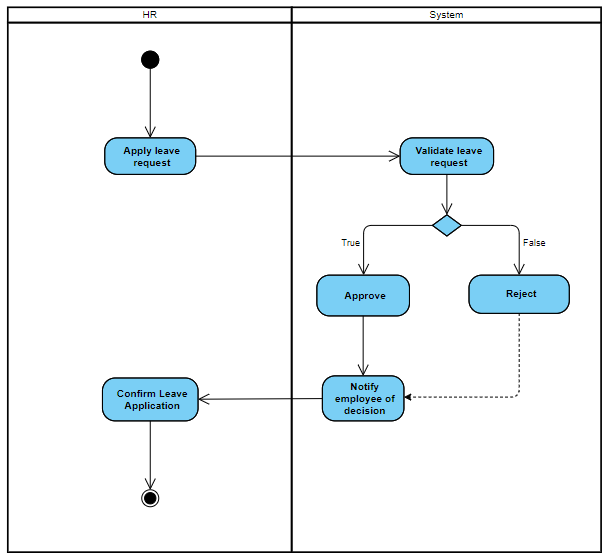
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | applyForLeave() | This method is responsible for processing an employee's application for leave. |

Method: applyForLeave()

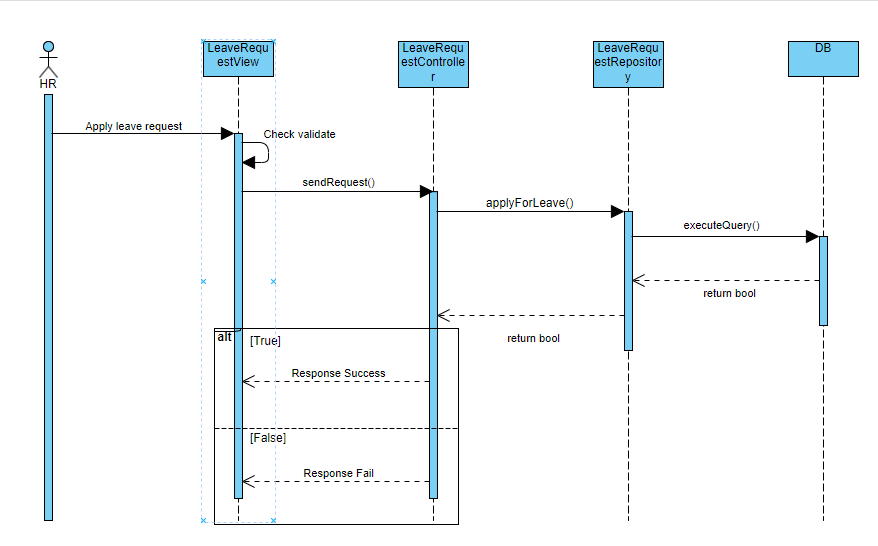
Description: This method is responsible for processing an employee's application for leave. It takes the necessary input parameters and performs the following steps:

* Validate the input parameters: Ensure that the provided leave details are valid and complete.
* Retrieve the employee record: Fetch the employee record from the database using the provided employee ID.
* Create a new leave application object: Create a new instance of the LeaveApplication class with the provided details and associate it with the employee.
* Submit the leave application: Save the leave application object to the database.
* Return the result: Return a boolean value indicating whether the leave application was successfully submitted or not.

### b)Activity Diagram(s)



### c)Sequence Diagram(s)



**d)Database queries**

## 12. Resignation letter

### a)Class Specifications

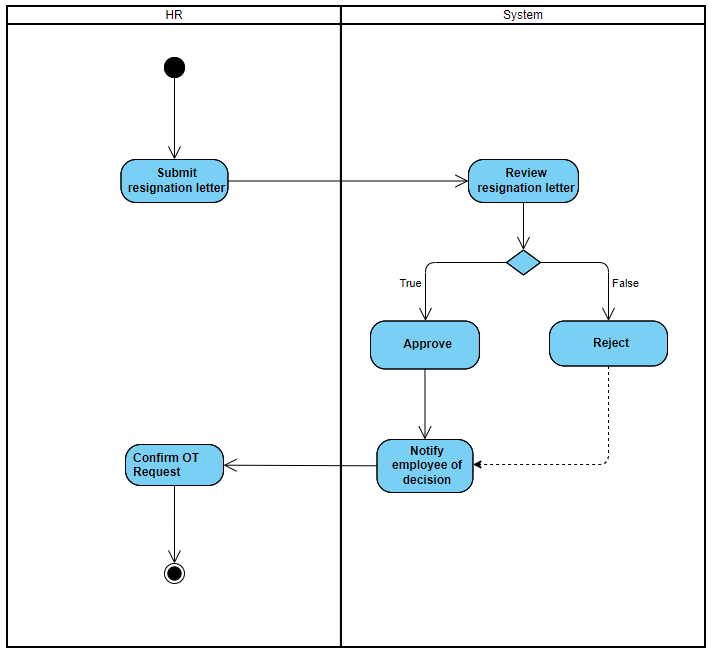
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | submitResignationLetter() | This method is responsible for processing an employee's resignation. |

Method: submitResignationLetter()

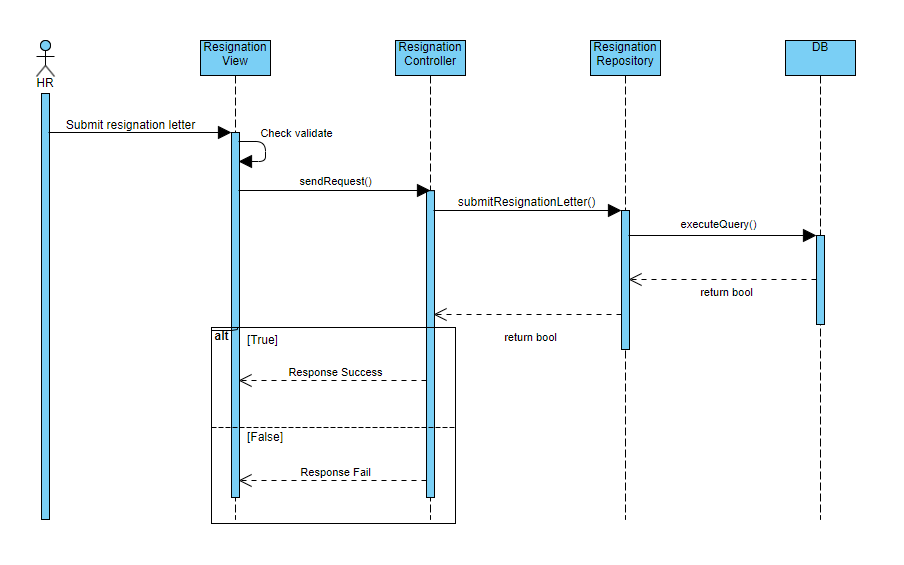
Description: This method is responsible for processing an employee's resignation. It takes the necessary input parameters and performs the following steps:

* Validate the input parameters: Ensure that the provided resignation details are valid and complete.
* Retrieve the employee record: Fetch the employee record from the database using the provided employee ID.
* Create a new resignation letter object: Create a new instance of the ResignationLetter class with the provided details and associate it with the employee.
* Submit the resignation letter: Save the resignation letter object to the database.
* Return the result: Return a boolean value indicating whether the resignation letter was successfully submitted or not.

### b)Activity Diagram(s)



### c)Sequence Diagram(s)



**d)Database queries**

## 13. Login

### a. Class Specifications

**EmployeeController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *checkLogin(string Email, string Password) : EmployeeDTO* | *Using Email and Password to check login by sending these information to Repository* |
|  |  |  |

**EmployeeRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *checkLogin(string Email, string Password): EmployeeDTO* | *Using Email and Password to check Employee in the database.* |

### 

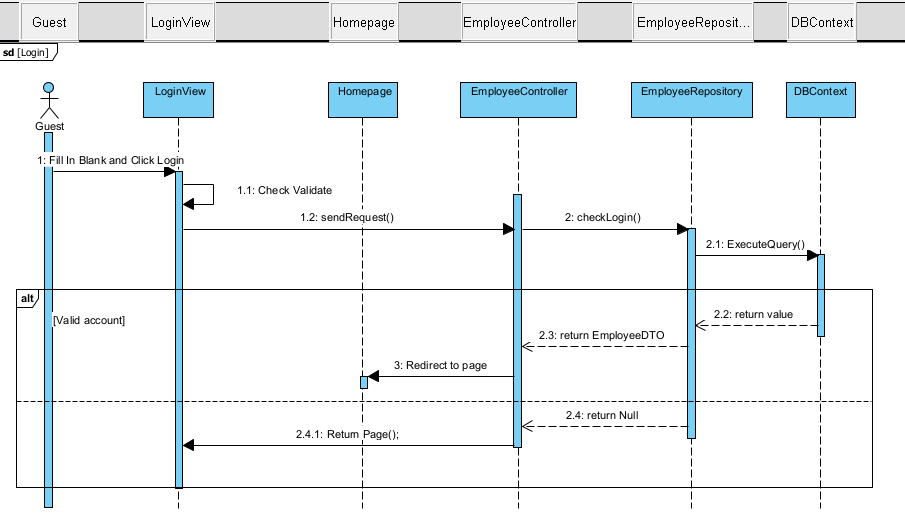
Method 01(Employee): *checkLogin(string Email, string Password): EmployeeDTO*

* Description: This method is responsible for authoring a guest to know if they are an employee or not by checking their email and password and sending it to the Repository and check in database.

Method 01(EmployeeRepository): *checkLogin(string Email, string Password): EmployeeDTO*

* Description: This method is responsible for authoring a guest to know if they are an employee or not by checking their email and password.
* First, we must check validate input to make sure that no empty input and wasting time to access Database
* Then, send a request to the Controller by form in view that contains Email and Password input. Controller will use EmployeeRepository to activate the DBContext, to access the database and use code query SELECT to check the account.
* After execution, if it returns null, that means there is no account using that email and password, or the password was wrong while the email is correct. And it will redirect back to Login View again.But if it returns a valid EmployeeDTO, then redirect it to Homepage so the employee can use web functions.

### b. Sequence Diagram(s)



## 14. Logout

### a. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

#### Employee Class

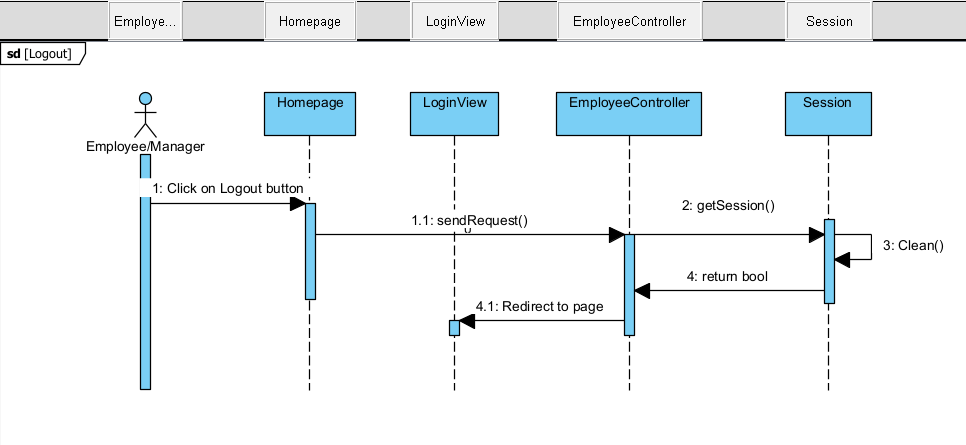
*[Provide the detailed description for the class methods]*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *logout()* | *Exit account and delete all account’s information in session.* |
|  |  |  |

Method 01: logout()

* Description: This method is simply exit the current account in session and delete all data at the moment

### b. Sequence Diagram(s)



### c. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

## 15. Approve HR Request

### a. Class Specifications

*[Provide the description for each class and the methods in each class, following the table format as below]*

#### EmployeeRepository Class

*[Provide the detailed description for the class methods]*

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *GetAllHRRequest()* | *View request information from HR by specific ID.* |
| *02* | *ApproveRequest(int id, string type)* | *Update HR requests by changing request status and notes.* |

Method 01: *GetAllHRRequest*(): List<RequestDTO>

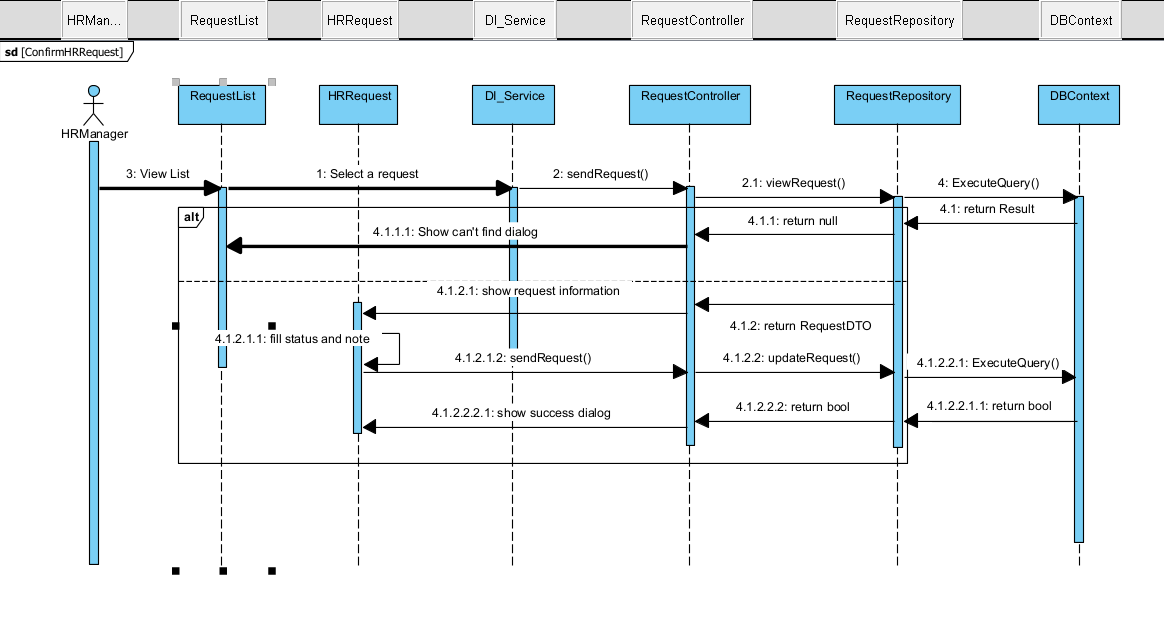
* After select a specific request from HR by requestID, then load all information of it into screen view

Method 02: *ApproveRequest(int id, string type)*

* HR Manager can update request status by Approve, Denial and add Notes into that request for HR to read it and fix their requests.

### b. Sequence Diagram(s)

*[Provide the sequence diagram(s) for the feature, see the sample below]*



### c. Database queries

*[Provide the detailed SQL (select, insert, update...) which are used in implementing the function/screen]*

SELECT \* FROM [TypeRequest] WHERE requestID = ?;

UPDATE Requests SET statusRequest = ?, noteRequest = ? WHERE requestID = ?;;

## 16. Request leave

### a. Class Specifications

***LeaveRequest Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *CreateLeaveRequest(leaveRequestData)* | *This method is responsible for creating and submitting a leave request. It facilitates the communication and coordination between various components involved in the leave request creation process.* |

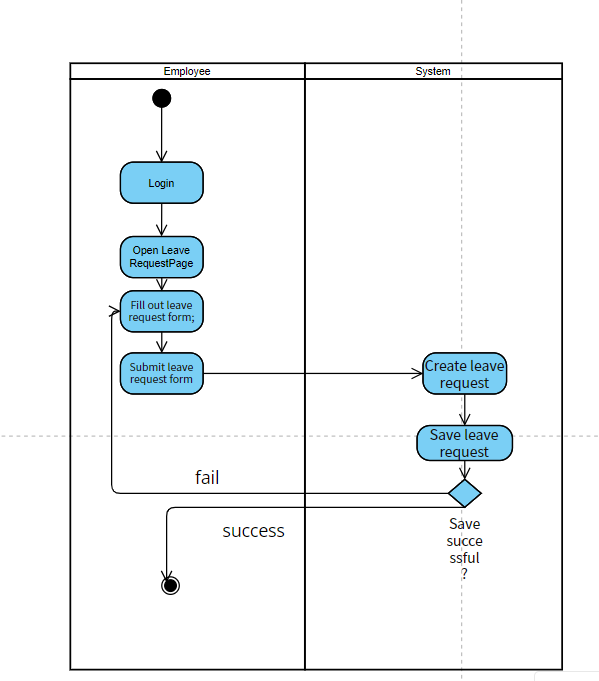
Method 01:*CreateLeaveRequest(leaveRequestData)*

* Description:OpenLeaveRequestPage(): The Employee initiates the leave request process by opening the LeaveRequestPage.
* DisplayLeaveRequestForm(): The LeaveRequestPage displays the leave request form to the Employee.
* SubmitLeaveRequestForm(): The Employee fills out the leave request form and submits it.
* CreateLeaveRequest(leaveRequestData): The LeaveRequestPage sends the leave request data to the LeaveRequestController to create the leave request.
* CreateLeaveRequest(leaveRequestData): The LeaveRequestController forwards the leave request data to the LeaveRequestRepository to create the leave request.
* SaveLeaveRequest(leaveRequestData): The LeaveRequestRepository attempts to save the leave request data to the DbContext (database or appropriate storage).
* Save successful: If the save operation is successful, the DbContext sends a success response to the LeaveRequestRepository.
* Leave request created: The LeaveRequestRepository notifies the LeaveRequestController that the leave request has been successfully created.
* Leave request created: The LeaveRequestController informs the LeaveRequestPage that the leave request has been created.
* Leave request created: The LeaveRequestPage notifies the Employee that the leave request has been created.

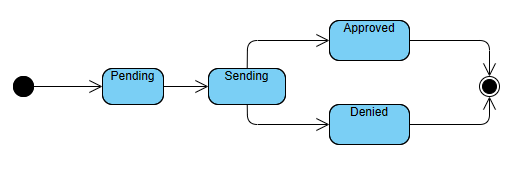
(Additional steps leading up to save failure)

* SaveLeaveRequest(leaveRequestData): The LeaveRequestRepository attempts to save the leave request data to the DbContext.
* Save failed: If the save operation fails, the DbContext sends a failure response to the LeaveRequestRepository.
* Leave request creation failed: The LeaveRequestRepository notifies the LeaveRequestController that the leave request creation has failed.
* Leave request creation failed: The LeaveRequestController informs the LeaveRequestPage that the leave request creation has failed.
* Leave request creation failed: The LeaveRequestPage notifies the Employee that the leave request creation has failed.

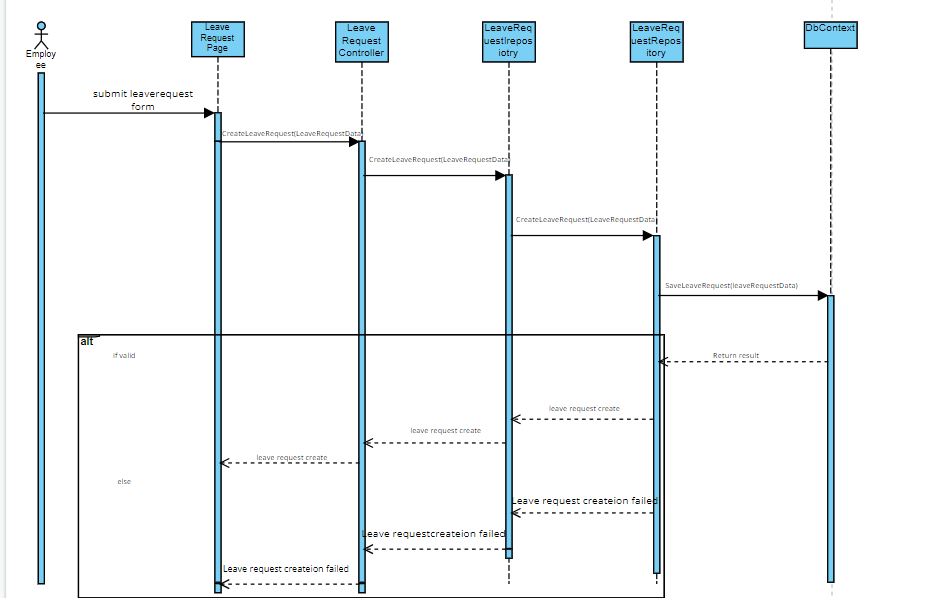
### b. Activity Diagram(s)



### c.State Diagram(s)



### d. Sequence Diagram(s)

****

## 17. Request resignation

### a. Class Specifications

ResignationRequest Class

| No | Method | Description |
| --- | --- | --- |
| 01 | requestResignation(requestData) | This method is responsible for creating and submitting a resignation request. It facilitates the communication and coordination between various components involved in the resignation creation process. |

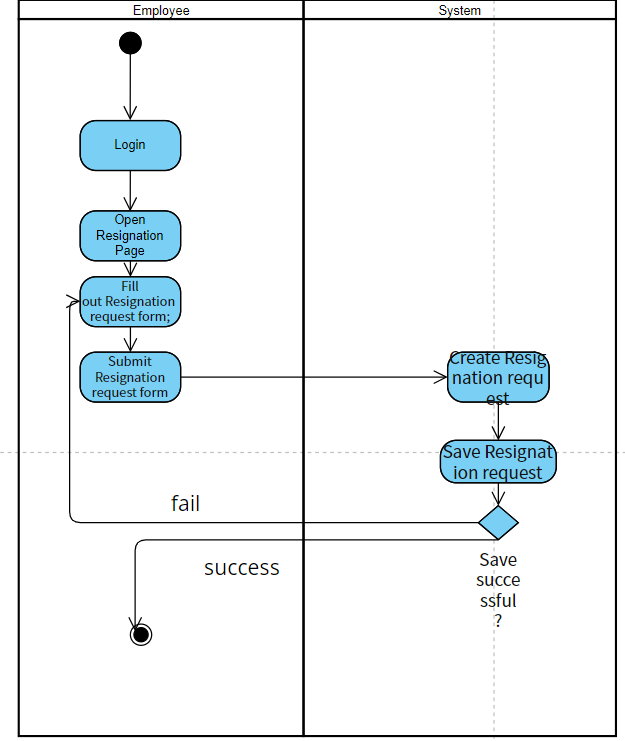
Method 01: requestResignation()

1. Description:Resignation Page(): The Employee initiates the Resignation request process by opening the Resignation Page
2. DisplayResignationForm(): The Resignation Page displays the leave request form to the Employee.
3. SubmitResignationForm(): The Employee fills out the Resignation request form and submits it.
4. CreateResignation(ResignationData): The Resignation Page sends the Resignation request data to the ResignationRequest Controller to create the Resignation request.
5. CreateResignation(ResignationData) : The ResignationRequest Controller forwards the Resignation request data to the ResignationRepository to create the Resignation request.
6. SaveResignation(ResignationData) : The ResignationRepository attempts to save the leave request data to the DbContext (database or appropriate storage).
7. Save successful: If the save operation is successful, the DbContext sends a success response to the ResignationRepository .
8. Resignation request created: The ResignationRepository notifies the ResignationRequest Controller that the leave request has been successfully created.
9. Resignation request created: The ResignationRequest Controller informs the LeaveRequestPage that the leave request has been created.
10. Leave request created: The Resignation Page notifies the Employee that the Resignation request has been created.

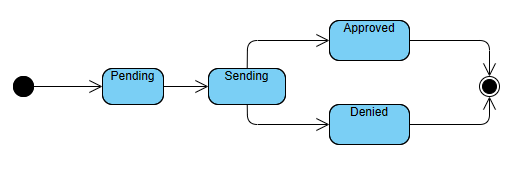
(Additional steps leading up to save failure)

* SaveResignation(ResignationData): The ResignationRepository attempts to save the Resignation request data to the DbContext.
* Save failed: If the save operation fails, the DbContext sends a failure response to the ResignationRepository .
* Resignation request creation failed: The ResignationRepository notifies the ResignationRequest Controller that the Resignation request creation has failed.
* Resignation request creation failed: The ResignationRequest Controller informs the Resignation Page that the Resignation request creation has failed.
* Resignation request creation failed: The Resignation Page notifies the Employee that the Resignation request creation has failed.

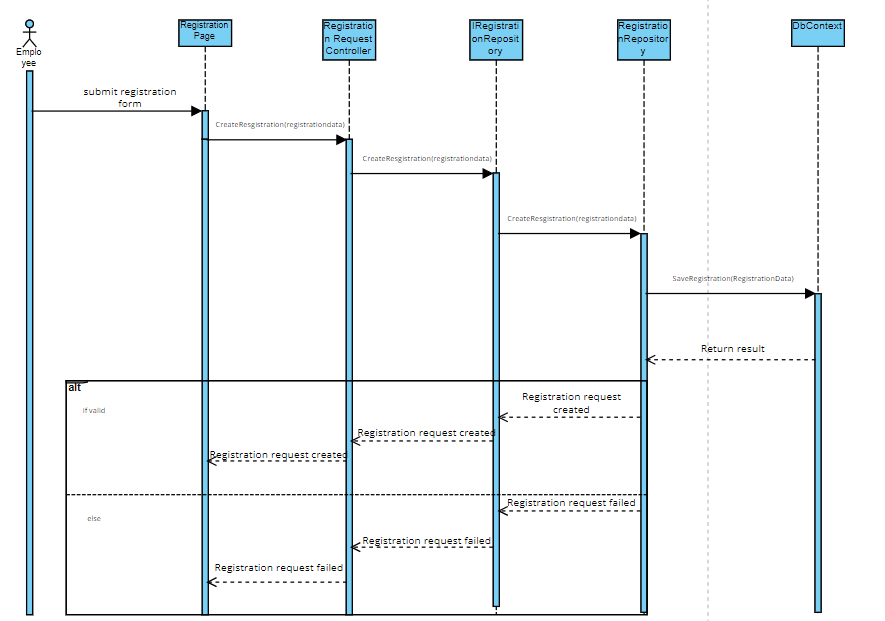
### b. Activity Diagram(s)



### c.State Diagram(s)



### d. Sequence Diagram(s)



## 18. Request update attendance

### a. Class Specifications

Attendance Class

| No | Method | Description |
| --- | --- | --- |
| 01 | requestUpdateAttendance(requestData) | This method allows an employee to request an update to their attendance record. It provides a means for employees to report any discrepancies or changes in their recorded attendance. |

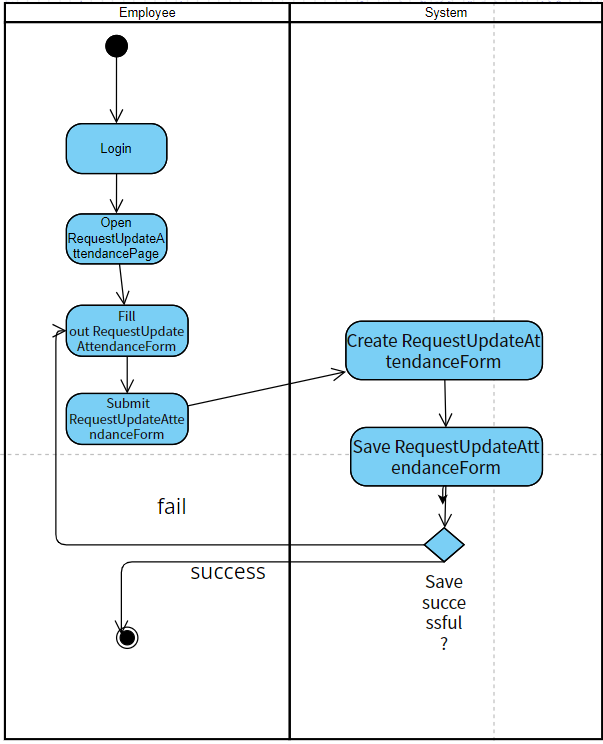
Method 01: requestUpdateAttendance

1. Description: RequestUpdateAttendancePage(): The Employee initiates the UpdateAttendance request process by opening the RequestUpdateAttendancePage
2. DisplayResignationForm(): The Resignation Page displays the leave request form to the Employee.
3. SubmitRequestUpdateAttendanceForm(): The Employee fills out the RequestUpdate request form and submits it.
4. CreateUpdateAttendanceRequest(requestData): The RequestUpdateAttendance Page sends the Update attendance request data to the RequestUpdateAttendanceControllerr to create the Update attendance request.
5. CreateUpdateAttendanceRequest(requestData) : The RequestUpdateAttendanceControllerr forwards the Update attendance request data to the RequestUpdateAttendanceRepository to create the Update attendance request.
6. SaveUpdateAttendanceRequest(requestData) : The RequestUpdateAttendanceRepository attempts to save the Update attendance data to the DbContext (database or appropriate storage).
7. Save successful: If the save operation is successful, the DbContext sends a success response to the RequestUpdateAttendanceRepository .
8. Update attendance request created: The RequestUpdateAttendanceRepository notifies the RequestUpdateAttendanceControllerr that the leave request has been successfully created.
9. Update attendance request created: The RequestUpdateAttendanceControllerr informs the RequestUpdateAttendance page that the leave request has been created.
10. RequestUpdateAttendance request created: The RequestUpdateAttendance page notifies the Employee that the Update attendance request has been created.

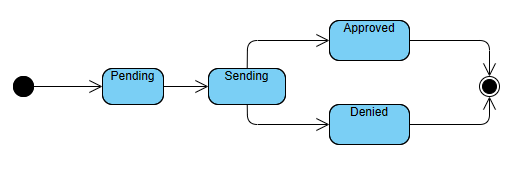
(Additional steps leading up to save failure)

* SaveUpdateAttendanceRequest(requestData) The RequestUpdateAttendanceRepository attempts to save the tUpdateAttendance request data to the DbContext.
* Save failed: If the save operation fails, the DbContext sends a failure response to the RequestUpdateAttendanceRepository .
* Update attendance request creation failed: The RequestUpdateAttendanceRepository notifies the RequestUpdateAttendanceControllerr that the Update attendance creation has failed.
* Update attendance request creation failed: The RequestUpdateAttendanceControllerr informs the RequestUpdateAttendancePage that the Update attendance creation has failed.
* Update attendance request creation failed: The RequestUpdateAttendancePage notifies the Employee that the Update attendance request creation has failed.

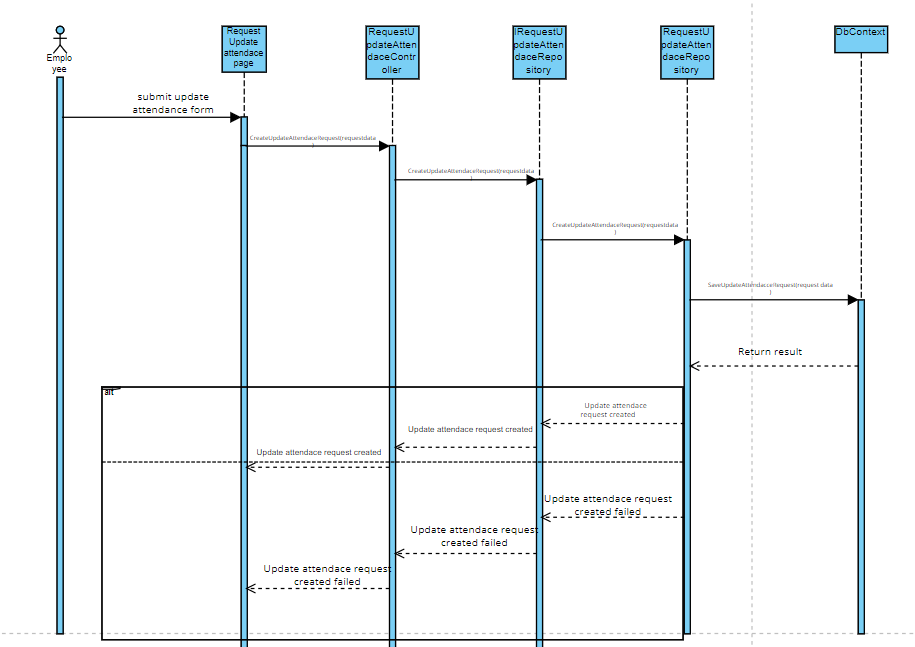
### b. Activity Diagram(s)



### c.State Diagram(s)



### d. Sequence Diagram(s)

****

## 19. Request change work department

### a. Class Specifications

ChangeWorkDepartmentRequest Class

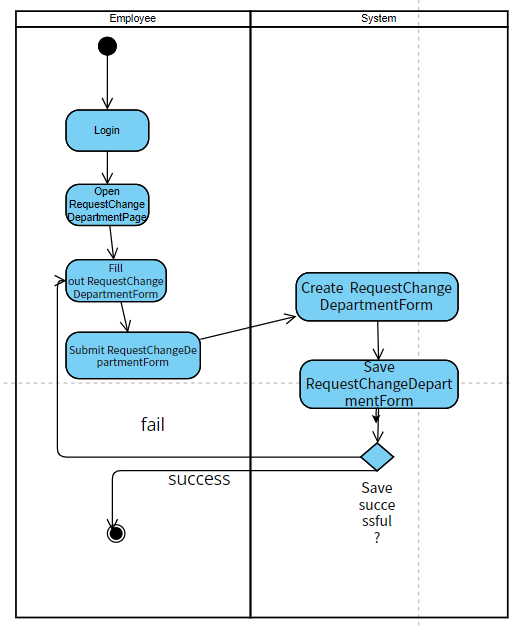
| No | Method | Description |
| --- | --- | --- |
| 01 | CreateChangeDepartmentRequest(requestData) | This method allows an employee to request a change in their work department within the organization. It provides a streamlined process for employees to communicate their desire to switch departments and initiate the necessary steps for the department change |

Method 01: CreateChangeDepartmentRequest(requestData)

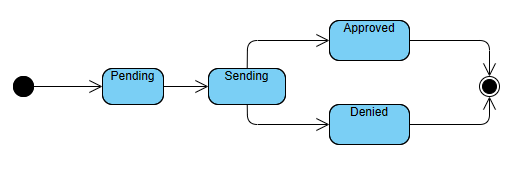
Description:

1. RequestChangeDepartmentPage(): The Employee initiates the Request Change Work Department process by opening the RequestChangeDepartmentPage.
2. DisplayChangeDepartmentForm(): The RequestChangeDepartmentPage displays the change department request form to the Employee.
3. SubmitRequestChangeDepartmentForm(): The Employee fills out the change department request form and submits it.
4. CreateChangeDepartmentRequest(requestData): The RequestChangeDepartmentPage sends the change department request data to the RequestChangeDepartmentController to create the change department request.
5. CreateChangeDepartmentRequest(requestData): The RequestChangeDepartmentController forwards the change department request data to the RequestChangeDepartmentRepository to create the change department request.
6. SaveChangeDepartmentRequest(requestData): The RequestChangeDepartmentRepository attempts to save the change department request data to the DbContext (database or appropriate storage).
7. Save successful: If the save operation is successful, the DbContext sends a success response to the RequestChangeDepartmentRepository.
8. Change department request created: The RequestChangeDepartmentRepository notifies the RequestChangeDepartmentController that the change department request has been successfully created.
9. Change department request created: The RequestChangeDepartmentController informs the RequestChangeDepartmentPage that the change department request has been created.
10. Change department request created: The RequestChangeDepartmentPage notifies the Employee that the change department request has been created.
11. (Additional steps leading up to save failure)
12. SaveChangeDepartmentRequest(requestData): The RequestChangeDepartmentRepository attempts to save the change department request data to the DbContext.
13. Save failed: If the save operation fails, the DbContext sends a failure response to the RequestChangeDepartmentRepository.
14. Change department request creation failed: The RequestChangeDepartmentRepository notifies the RequestChangeDepartmentController that the change department request creation has failed.
15. Change department request creation failed: The RequestChangeDepartmentController informs the RequestChangeDepartmentPage that the change department request creation has failed.
16. Change department request creation failed: The RequestChangeDepartmentPage notifies the Employee that the change department request creation has failed.

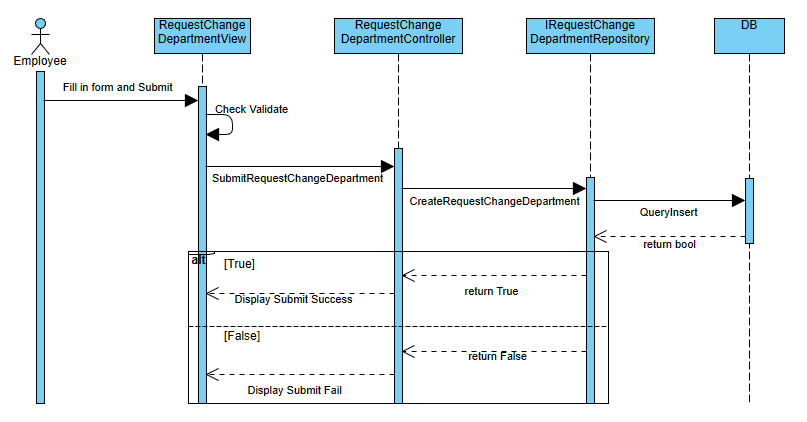
### b. Activity Diagram(s)



### c.State Diagram(s)



### d. Sequence Diagram(s)

****

## 

## 20. Request tax support

### a. Class Specifications

Tax Request Class

| No | Method | Description |
| --- | --- | --- |
| 01 | requestTaxSupport(requestData) | This method is responsible for handling the request to request tax support |

RequestTaxSupportPage(): The Employee initiates the RequestTaxSupport process by opening the RequestTaxSupportPage.

DisplayTaxSupportForm(): The RequestTaxSupportPage displays the tax support request form to the Employee.

SubmitRequestTaxSupportForm(): The Employee fills out the tax support request form and submits it.

CreateTaxSupportRequest(requestData): The RequestTaxSupportPage sends the tax support request data to the RequestTaxSupportController to create the tax support request.

CreateTaxSupportRequest(requestData): The RequestTaxSupportController forwards the tax support request data to the RequestTaxSupportRepository to create the tax support request.

SaveTaxSupportRequest(requestData): The RequestTaxSupportRepository attempts to save the tax support data to the DbContext (database or appropriate storage).

Save successful: If the save operation is successful, the DbContext sends a success response to the RequestTaxSupportRepository.

Tax support request created: The RequestTaxSupportRepository notifies the RequestTaxSupportController that the tax support request has been successfully created.

Tax support request created: The RequestTaxSupportController informs the RequestTaxSupportPage that the tax support request has been created.

Tax support request created: The RequestTaxSupportPage notifies the Employee that the tax support request has been created.

Additional steps leading up to save failure:

11. SaveTaxSupportRequest(requestData): The RequestTaxSupportRepository attempts to save the tax support request data to the DbContext.

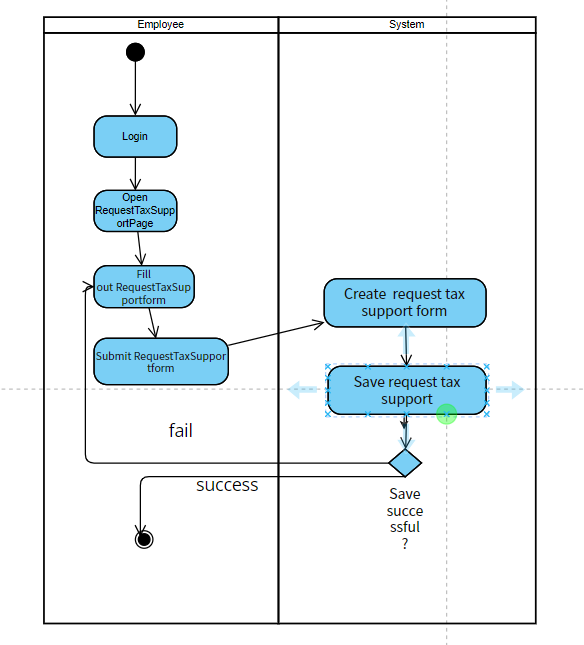
Save failed: If the save operation fails, the DbContext sends a failure response to the RequestTaxSupportRepository.

Tax support request creation failed: The RequestTaxSupportRepository notifies the RequestTaxSupportController that the tax support creation has failed.

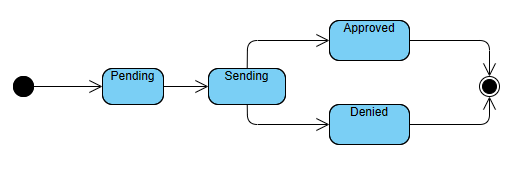
Tax support request creation failed: The RequestTaxSupportController informs the RequestTaxSupportPage that the tax support creation has failed.

Tax support request creation failed: The RequestTaxSupportPage notifies the Employee that the tax support request creation has failed.

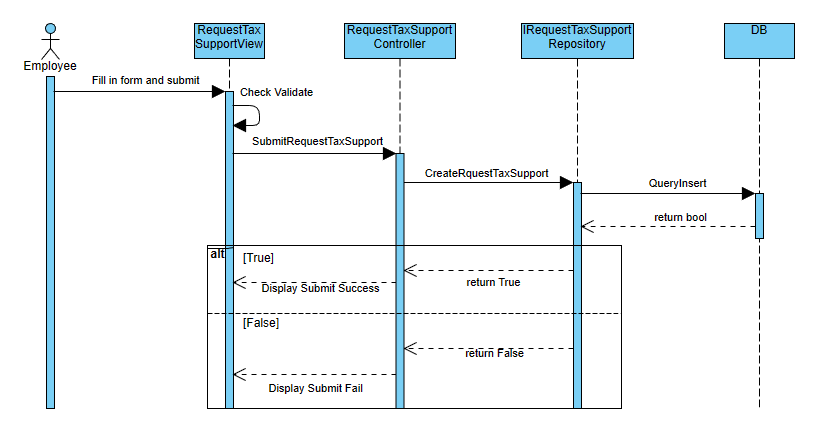
### b. Activity Diagram(s)

****

### c.State Diagram(s)



### d. Sequence Diagram(s)

****

## 21. Request OT

### a. Class Specifications

***OT Request Class***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *requestOT(reqestdata)* | This method is responsible for handling the request to request OT support |

RequestOTPage(): The Employee initiates the Request OT process by opening the RequestOTPage.

DisplayOTRequestForm(): The RequestOTPage displays the OT request form to the Employee.

SubmitOTRequestForm(): The Employee fills out the OT request form and submits it.

CreateOTRequest(requestData): The RequestOTPage sends the OT request data to the RequestOTController to create the OT request.

CreateOTRequest(requestData): The RequestOTController forwards the OT request data to the RequestOTRepository to create the OT request.

SaveOTRequest(requestData): The RequestOTRepository attempts to save the OT request data to the DbContext (database or appropriate storage).

Save successful: If the save operation is successful, the DbContext sends a success response to the RequestOTRepository.

OT request created: The RequestOTRepository notifies the RequestOTController that the OT request has been successfully created.

OT request created: The RequestOTController informs the RequestOTPage that the OT request has been created.

RequestOT request created: The RequestOTPage notifies the Employee that the OT request has been created.

(Additional steps leading up to save failure)

SaveOTRequest(requestData): The RequestOTRepository attempts to save the OT request data to the DbContext.

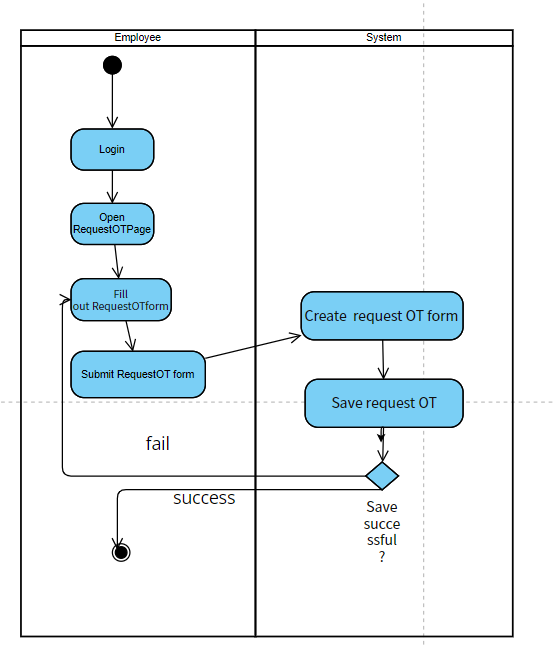
Save failed: If the save operation fails, the DbContext sends a failure response to the RequestOTRepository.

OT request creation failed: The RequestOTRepository notifies the RequestOTController that the OT request creation has failed.

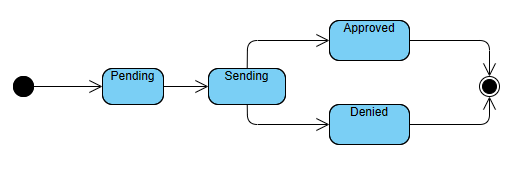
OT request creation failed: The RequestOTController informs the RequestOTPage that the OT request creation has failed.

OT request creation failed: The RequestOTPage notifies the Employee that the OT request creation has failed.

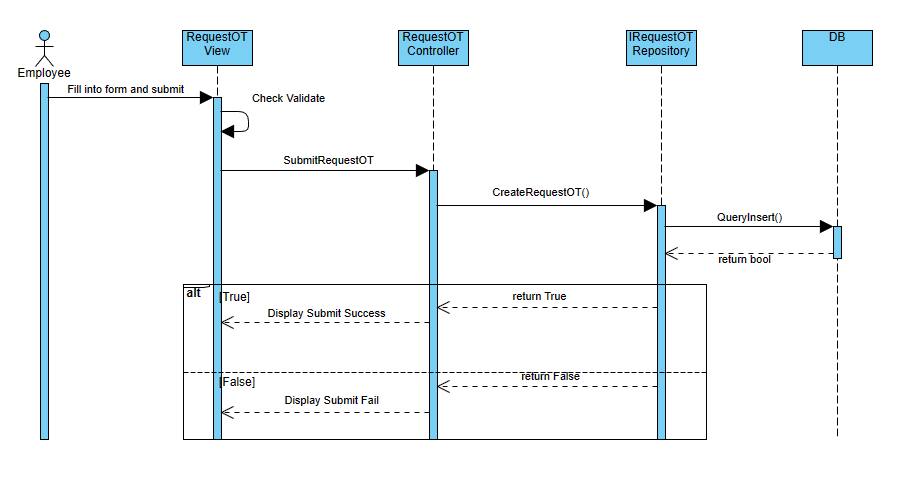
### b. Activity Diagram(s)

****

### c.State Diagram(s)



### d. Sequence Diagram(s)

****

# IV. Class Diagram

## Diagram

[Image link](https://drive.google.com/file/d/1_VpKA15Ovmng7gBnOhG_8JP7vmuhw7it/view?usp=sharing)

## Class Diagram Specifications

### 2.1 EmployeeController Class

| *No* | *Method* | *Description* |
| --- | --- | --- |
| *01* | *createEmployee(Employee emp) : bool* | *Creates a new employee with the provided employee data.* |
| *02* | *getEmployee(string id) : Employee* | *Retrieves the details of an employee based on the provided employee ID.* |
| *03* | *updateEmployee(Employee emp) : bool* | *Updates the information of an existing employee with the provided data.* |
| *04* | *deleteEmployee(string id) : bool* | *Deletes an existing employee based on the provided employee ID.* |
| *05* | *searchEmployee(string search): List[Employee]* | *Searches for employees based on the provided search criteria.* |
| *06* | *checkLogin(string email, password): EmployeeDTO* | *Using Email and Password to Login into systems,.* |
| *07* | *logout()* | *Exit account and delete all account’s information in session.* |

### 2.2 EmployeeRepository Class

| *No* | *Method* | *Description* |
| --- | --- | --- |
| *01* | *createEmployee(Employee emp) : bool* | *Creates a new employee record in the database with the provided employee data.* |
| *02* | *getEmployee(string id) : Employee* | *Retrieves the details of an employee from the database based on the provided employee ID.* |
| *03* | *updateEmployee(Employee emp) : bool* | *Updates the information of an existing employee in the database with the provided data.* |
| *04* | *deleteEmployee(string id) : bool* | *Deletes an existing employee from the database based on the provided employee ID.* |
| *05* | *searchEmployee(string search) : List[Employee]* | *Searches for employees in the database based on the provided search criteria and returns a list of matching employees.* |
| *06* | *checkLogin(string email, password): EmployeeDTO* | *Using a database to check if there are any Employee that have exactly the same email and password. Then return that Employee back.* |
| *07* | *viewHRRequest(int requestID) : RequestDTO* | *View request information from HR by specific ID.* |
| *08* | *updateHRRequest(RequestDTO): void* | *Update HR requests by changing request status and notes.* |

### 2.3 WorkDepartmentRequestController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | CreateChangeDepartmentRequest(requestData) | This method allows an employee to request a change in their work department within the organization |
|  |  |  |

### 2.4 WorkDepartmentRequestRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *SaveChangeDepartmentRequest(requestData)* | This method is responsible for update attendance for employee request to database |
|  |  |  |

### 2.5 ResignationController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | requestResignation(requestData) | This method is responsible for creating and submitting a resignation request. |
|  |  |  |

### 2.6 ResignationRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *saveResgination(requestData)* | This method is responsible for save leave request to database |
|  |  |  |

### 2.7 OTController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | requestOT(reqestdata) | This method is responsible for handling the request to request OT support |
|  |  |  |

### 2.8 OTRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *saveOTRequest(rquestData)* | This method is responsible for save OT request for employee request to database |
|  |  |  |

### 2.9 TaxController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | manageTaxRequest(string requsetId) : bool | Processes a tax request and determines whether it should be accepted or denied. |
| *02* | requestTaxSupport(requestData) | This method is responsible for handling the request to request tax support |

### 2.10 TaxRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | acceptTaxRequest(string requestId) : bool | Accepts a tax request with the provided request ID. |
| *02* | denyTaxRequest(string requestId) : bool | Denies a tax request with the provided request ID. |
| *03* | saveTaxSupportRequest(requestData) | This method is responsible for update attendance for employee request to database |

### 2.11AttendanceController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | checkAttendance(lst: List[Employee]) : bool | *C*hecks the attendance of a list of employees and returns a boolean indicating the attendance status. |
| *02* | requestUpdateAttendance(requestData) | This method allows an employee to request an update to their attendance record. |

### 2.12 AttendanceRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | markAttendance(string employeeId, DateTime checkTime) : bool | Marks the attendance of an employee for a specific date. |
| *02* | saveUpdateAttendaceRequest(requestData) | This method is responsible for update attendance for employee request to database |

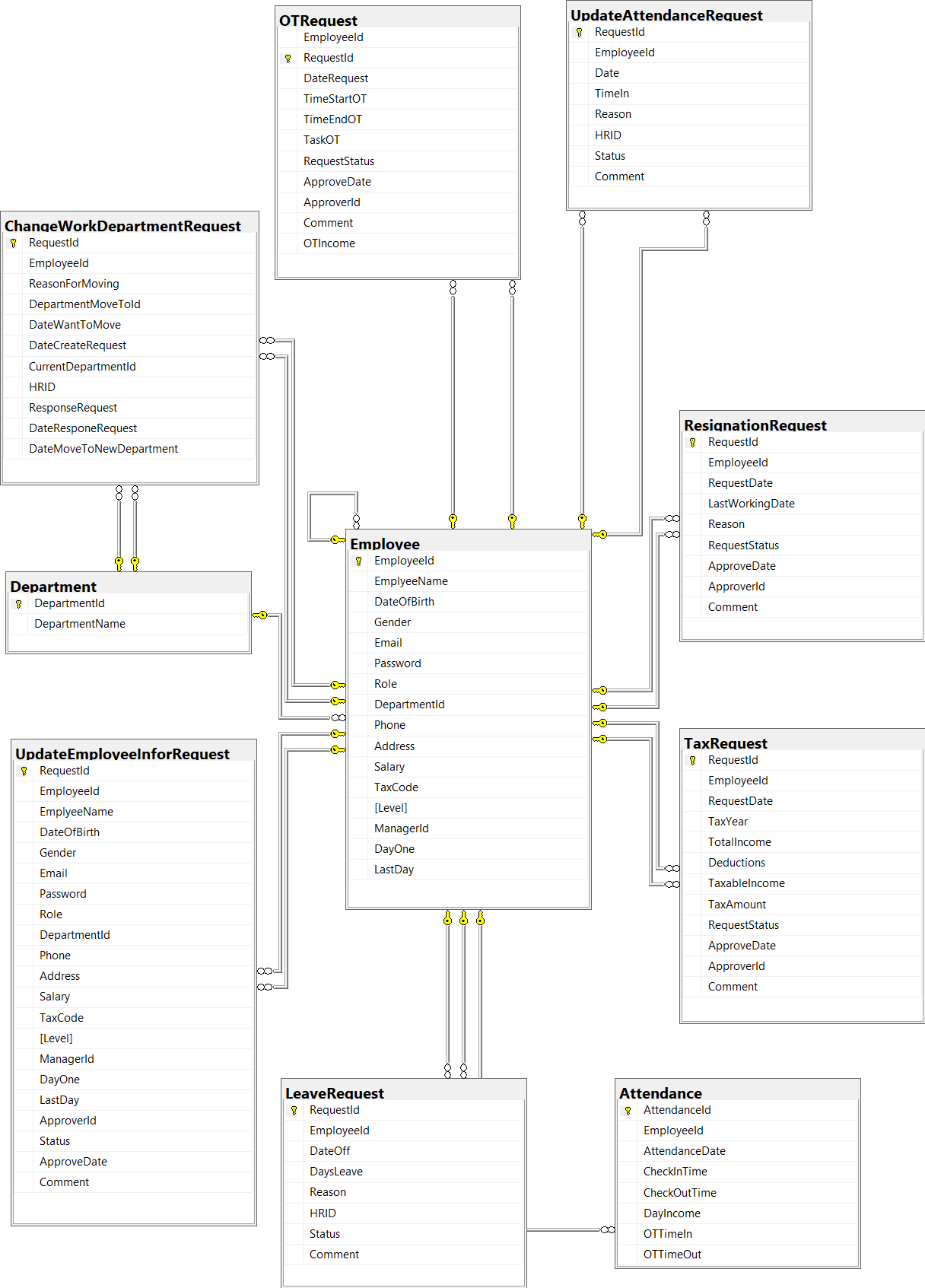
### 2.13 LeaveRequestController Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | CreateLeaveRequest(leaveRequestData) | This method is responsible for creating and submitting a leave request |
|  |  |  |

### 2.14 LeaveRequestRepository Class

| **No** | **Method** | **Description** |
| --- | --- | --- |
| *01* | *saveLeaveRequest(*leaveRequestData) | This method is responsible for save leave request to database |
|  |  |  |

# V. Database Tables

****

## 1. Employee

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | EmployeeId | nchar(10) | 10 | true | true | PK |  |
| 2 | EmployeeName | nvarchar(100) | 100 | false | false |  |  |
| 3 | DateOfBirth | date | date | false | false |  |  |
| 4 | Gender | int | 1 | false | false |  |  |
| 5 | Email | nchar(50) | 50 | false | false |  |  |
| 6 | Password | nchar(30) | 30 | false | false |  |  |
| 7 | Role | nchar(20) | 20 | false | false |  |  |
| 8 | DepartmentId | nchar(10) | 10 | false | false | FK | Reference to Department table |
| 9 | Phone | nchar(10) | 10 | false | false |  |  |
| 10 | Address | nvarchar(200) | 200 | false | false |  |  |
| 11 | Salary | decimal(18, 0) | decimal | false | false |  |  |
| 12 | TaxCode | nchar(20) | 20 | false | false |  |  |
| 13 | [Level] | nchar(20) | 20 | false | false |  |  |
| 14 | ManagerId | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 15 | DayOne | date | date | false | false |  | First working day |
| 16 | LastDay | date | date | false | false |  | Last working day |

## 2. Department

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | DepartmentId | nchar(10) | 10 | true | true | PK |  |
| 2 | DepartmentName | nvarchar(50) | 50 | false | false |  |  |

## 3. ChangeWorkDepartmentRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | ReasonForMoving | nvarchar(300) | 300 | false | false |  |  |
| 4 | DepartmentMoveToId | nchar(10) | 10 | false | false |  |  |
| 5 | DateWantToMove | date | date | false | false |  |  |
| 6 | DateCreateRequest | datetime | datetime | false | false |  |  |
| 7 | CurrentDepartmentId | nchar(10) | 10 | false | false |  |  |
| 8 | HRID | nchar(10) | 10 | false | false |  |  |
| 9 | ResponseRequest | bit | 1 | false | false |  |  |
| 10 | DateResponeRequest | datetime | datetime | false | false |  |  |
| 11 | DateMoveToNewDepartment | date | date | false | false |  |  |

## 4. ResignationRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | RequestDate | datetime | datetime | false | false |  |  |
| 4 | LastWorkingDate | date | date | false | false |  |  |
| 5 | Reason | nvarchar(300) | 300 | false | false |  |  |
| 6 | RequestStatus | nvarchar(20) | 20 | false | false |  |  |
| 7 | ApproveDate | datetime | datetime | false | false |  |  |
| 8 | ApproverId | nchar(10) | 10 | false | false |  |  |
| 9 | Comment | nvarchar(100) | 100 | false | false |  |  |

## 5. UpdateEmployeeInforRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | true | true | PK |  |
| 3 | EmployeeName | nvarchar(100) | 100 | false | false |  |  |
| 4 | DateOfBirth | date | date | false | false |  |  |
| 5 | Gender | int | 1 | false | false |  |  |
| 6 | Email | nchar(50) | 50 | false | false |  |  |
| 7 | Password | nchar(30) | 30 | false | false |  |  |
| 8 | Role | nchar(20) | 20 | false | false |  |  |
| 9 | DepartmentId | nchar(10) | 10 | false | false | FK | Reference to Department table |
| 10 | Phone | nchar(10) | 10 | false | false |  |  |
| 11 | Address | nvarchar(200) | 200 | false | false |  |  |
| 12 | Salary | decimal(18, 0) | decimal | false | false |  |  |
| 13 | TaxCode | nchar(20) | 20 | false | false |  |  |
| 14 | [Level] | nchar(20) | 20 | false | false |  |  |
| 14 | ManagerId | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 16 | DayOne | date | date | false | false |  | First working day |
| 17 | LastDay | date | date | false | false |  | Last working day |
| 18 | ApproverId | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 19 | Status | nvarchar(20) | 20 | false | false |  |  |
| 20 | ApproveDate | datetime | datetime | false | false |  |  |
| 21 | Comment | nvarchar(50) | 50 | false | false |  |  |

## 6. TaxRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | RequestDate | datetime | datetime | false | false |  |  |
| 4 | TaxYear | int | int | false | false |  |  |
| 5 | TotalIncome | decimal(18, 0) | decimal(18, 0) | false | false |  |  |
| 6 | Deductions | decimal(18, 0) | decimal(18, 0) | false | false |  |  |
| 7 | TaxableIncome | decimal(18, 0) | decimal(18, 0) | false | false |  |  |
| 8 | TaxAmount | decimal(18, 0) | decimal(18, 0) | false | false |  |  |
| 9 | RequestStatus | nvarchar(20) | 20 | false | false |  |  |
| 10 | ApproveDate | datetime | datetime | false | false |  |  |
| 11 | ApproverId | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 12 | Comment | nvarchar(50) | 50 | false | false |  |  |

## 7. Attendance

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | AttendanceId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | AttendanceDate | date | date | false | false |  |  |
| 4 | CheckInTime | datetime | datetime | false | false |  |  |
| 5 | CheckOutTime | datetime | datetime | false | false |  |  |
| 6 | DayIncome | decimal(18, 0) | decimal(18, 0) | false | false |  |  |
| 7 | OTTimeIn | datetime | datetime | false | false |  |  |
| 8 | OTTimeOut | datetime | datetime | false | false |  |  |

## 8. LeaveRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | DateOff | date | date | false | false |  |  |
| 4 | DaysLeave | int | int | false | false |  |  |
| 5 | Reason | nvarchar(300) | 300 | false | false |  |  |
| 6 | HRID | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 7 | Status | nvarchar(20) | 20 | false | false |  |  |
| 8 | Comment | nvarchar(50) | 50 | false | false |  |  |

## 9. OTRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | DateRequest | datetime | datetime | false | false |  |  |
| 4 | TimeStartOT | datetime | datetime | false | false |  |  |
| 5 | TimeEndOT | datetime | datetime | false | false |  |  |
| 6 | TaskOT | nvarchar(50) | 50 | false | false |  |  |
| 7 | RequestStatus | nvarchar(20) | 20 | false | false |  |  |
| 8 | ApproveDate | datetime | datetime | false | false |  |  |
| 9 | ApproverId | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 10 | Comment | nvarchar(50) | 50 | false | false |  |  |

## 

## 10. UpdateAttendanceRequest

| **#** | **Field name** | **Type** | **Size** | **Unique** | **Not Null** | **PK/FK** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | RequestId | int | int | true | true | PK |  |
| 2 | EmployeeId | nchar(10) | 10 | false | true | FK | Reference to Employee table |
| 3 | Date | datetime | datetime | false | false |  |  |
| 4 | TimeIn | datetime | datetime | false | false |  |  |
| 5 | Status | nvarchar(20) | 20 | false | false |  |  |
| 6 | Reason | nvarchar(300) | 300 | false | false |  |  |
| 7 | HRID | nchar(10) | 10 | false | false | FK | Reference to Employee table |
| 8 | Comment | nvarchar(50) | 50 | false | false |  |  |