Software Engineering

Final Project Report

Human Resource Management System

[HRMS]



**Ahmed Ehab Saleh 5833**

**Omar Ahmed Abdulaziz 5960**

**Youssef Mohamed Aboelsaad 5950**

SWE GROUP 8

# 1) User Requirements

**1.1** - The HRMS shall include the functionality to add, delete, and modify employees, while incorporating important information about each employee.

**1.2** - The Admin or HR Manager shall have the ability to manage different modifications related to the employees. These include promotions, movements between departments and retirements/departures.

**1.3** - The Admin or HR Manager shall oversee the salaries of all employees and all related transactions such as increases, bonuses etc.

# 2) System Requirements

**2.1** –

Each employee shall have a record in the database that includes their SSN, first name, last name, phone number, department, and salary.

**2.2** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | Add a new employee. |
| Description | This function adds a new employee's data that has been hired to the HRMS. |
| Inputs | Employee's SSN, first name, last name, phone number, department, and salary. |
| Source | Entered by the system user. |
| Output | A record containing the employee's SSN, first name, last name, phone number, department, and salary |
| Destination | System database. |
| Action | A new employee record is added to the database that includes the employee’s SSN, first name, last name, phone number, department, and salary. |
| Requires | Employee's SSN, first name, last name, phone number, department, and salary. |
| Pre-condition | Inputted SSN must be unique. |
| Post-condition | All employee details get added to the system. |
| Side-effects | None. |

**2.3** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | Delete an existing employee |
| Description | This function deletes an employee's record who retired or got fired from the database. |
| Inputs | Employee's SSN. |
| Source | Entered by the system user. |
| Output | None. |
| Destination | None. |
| Action | Employee record is deleted from the database. |
| Requires | Employee’s SSN. |
| Pre-condition | Inputted SSN must exist. |
| Post-condition | All employee details get deleted from the system. |
| Side-effects | None. |

**2.4** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | Modify an existing employee. |
| Description | This function modifies an employee's record, first an employee is searched by their SSN and all their record data appears which is then modified by the system user. |
| Inputs | Employee's SSN, followed by the data to be modified. |
| Source | Entered by the system user. |
| Output | A record containing the employee's modified data that includes their SSN, first name, last name, phone number, department, and salary. |
| Destination | System database. |
| Action | Employee record is fetched using the SSN inputted. This record is then updated with the required modifications to the first name, last name, phone number, department, and salary fields. |
| Requires | Employee’s SSN as well as current and new first name, last name, phone number, department, and salary so record can be updated. |
| Pre-condition | Inputted SSN must exist. |
| Post-condition | Employee’s SSN must not change. |
| Side-effects | None. |

**2.5** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | Manage employee promotions |
| Description | Manage the promotion of an employee from one position to a higher position/rank, as ordered by HR Manager or Admin. |
| Inputs | Employee SSN, current position, new position. |
| Source | Entered by system user. |
| Output | An updated record of this employee with a new and higher position. |
| Destination | System database. |
| Action | Employee record is fetched using the SSN inputted, the current position of this employee is also fetched. This record is then updated with the new position for the employee. |
| Requires | Employee’s SSN as well as current and new position so record can be updated. |
| Pre-condition | Inputted SSN must exist as well as new position must be of a higher rank than previous position. |
| Post-condition | Employee’s position is updated with the higher rank. |
| Side-effects | None |

**2.6** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | Manage employee movements between departments |
| Description | Manage the migration of employees from one department to the other, as instructed by HR Manager. |
| Inputs | Employee SSN, New Department |
| Source | Entered by system user. |
| Output | An updated record in Employee table of the current employee with new department. Record of this employee in old department table deleted and a new record in of the employee in the table of the new department. |
| Destination | System database. |
| Action | Employee record is fetched using SSN inputted, the current department of the employee is updated. Employee record is also fetched from department’s table. Deleted from the old department and inserted in the new department table. |
| Requires | Employee’s SSN and new department. |
| Pre-condition | Employee SSN must exist, new department must be different than old department. |
| Post-condition | Employee’s department is updated, and department’s tables updated accordingly |
| Side-effects | None |

**2.7** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | Manage employee retirement/departure |
| Description | Manage the departure or retirement of an employee. |
| Inputs | Employee SSN |
| Source | Entered by system user. |
| Output | An updated employee table with the record of the departed/retired employee deleted. |
| Destination | System database. |
| Action | Fetch employee record using SSN and perform deletion operation on this record from the database. |
| Requires | Employee SSN to fetch the record. |
| Pre-condition | Employee SSN must exist. |
| Post-condition | Employee table updated with this employee removed. |
| Side-effects | None |

**2.8** -

|  |  |
| --- | --- |
| *HRMS/Management Software* | |
| Function | View employees’ salaries. |
| Description | This function allows the system user to see all the employees' salaries. |
| Inputs | None. |
| Source | None. |
| Output | A table containing the employees' SSN, first name, last name, department, and salary. |
| Destination | System's user interface. |
| Action | Employees’ records are fetched. Then, these records’ SSN, first name, last name, department, and salary are displayed. |
| Requires | None. |
| Pre-condition | None. |
| Post-condition | None. |
| Side-effects | None |

# 3) Suggested Type of Software Process:

**Extreme Programming (XP)** which is the best-known and most widely used agile method. It aims to produce higher quality software, and higher quality of life for the development team. Where the key practices include Pair Programming and **Test-Driven Development (TDD).**

# 4) Division of Phases:

1. **Planning:**the first stage, usually a customer explains their requirements to the development team however in our case there is no customer so we made up requirements that seemed fit for a simple HRMS system.
2. **Design** was done at the same time as planning, we focused our designs on simplicity in order to avoid further confusions and to avoid unnecessary complexities and redundancies in the future.
3. **Coding,** the phase during which the actual code is created by implementing specific XP practices. Most of the work was done while being on the same Discord call at all times to simulate being one team in the same room together while working.
4. **Testing**is a core element of extreme programming. Test-driven development was the approach we used for this project where we would write a failing test case and then write code to the point of succeeding this initial test case and again for a further couple of test cases that would seem plentiful for a unit to pass its test.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5) Our System and Application’s suggested Architecture

is **Model-View-Controller [MVC]** where our system is divided into three separate components:

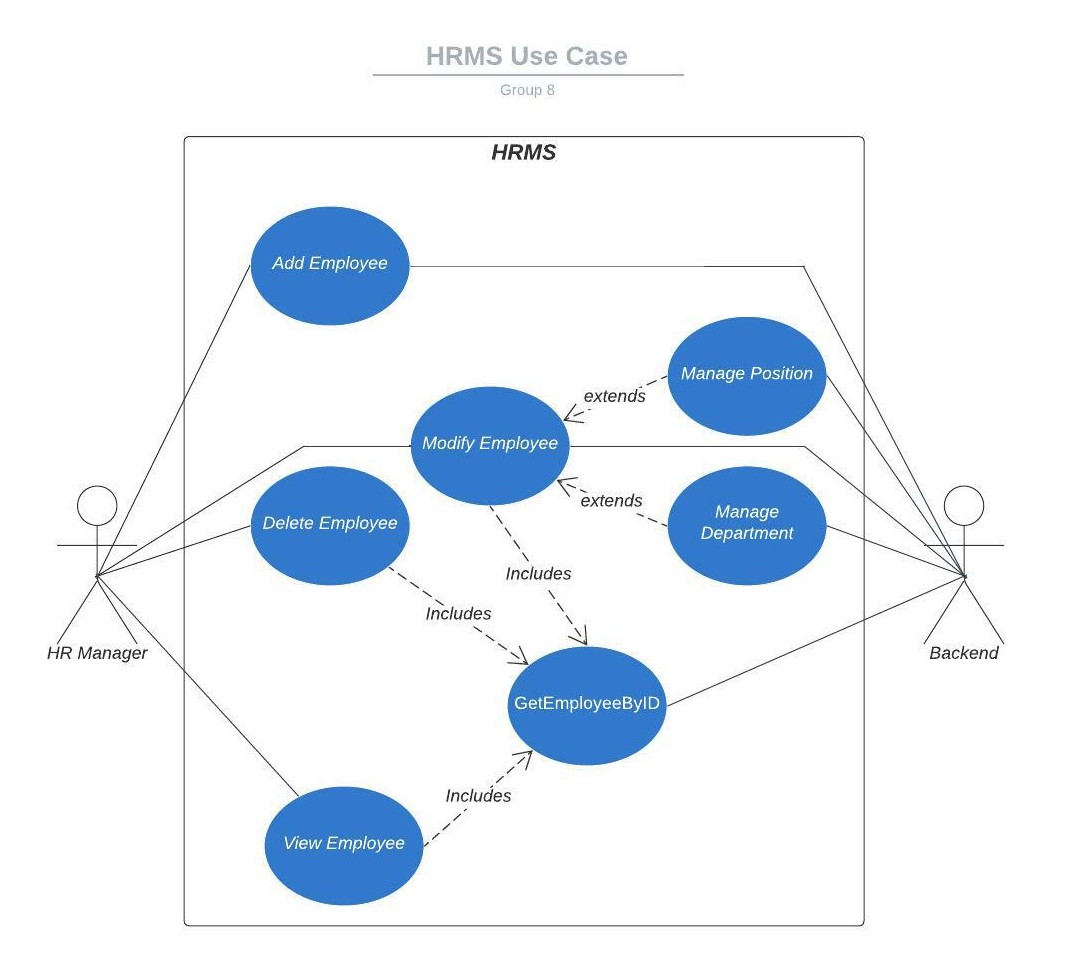
1. **Model:** which is represented by our application’s database
2. **View:** which is represented by the visuals of the app from it’s front-end and User Interface.
3. **Controller:** which is the interfacing between the user’s actions on the clickable buttons and other interactive elements in the application’s View.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| Functional Requirements | Non-functional Requirements |
| Allow Deletion and Addition of Employees | Perform with virtually no delays |
| Allow promotion, demotion and department migration seamlessly | Handle a capacity of over 600 Employees |
| The Manager should be able to view the details of any employee | To be fairly reliable and available for >98% of the day (Max 30 mins down time per day) |
| The HR Manager can edit the salaries of the employees | Usability isn’t a great issue since the HR Manager will be fairly trained |
| Provides validation for adding new employees | Must be fairly secure |

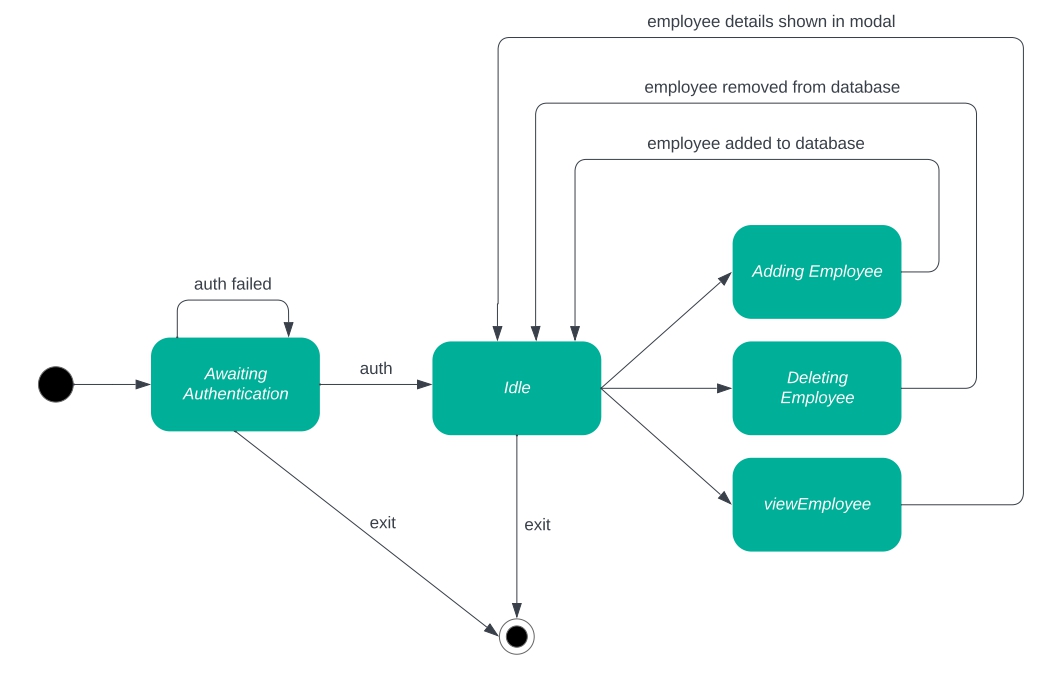
# 6) System Modelling:

## **6.1** - Use Case Diagram: The diagram is meant to depict the user’s possible interactions with the view and the database as well as how these interactions affect the content of the database and what’s viewed in the UI. There is no indication of time in this diagram.

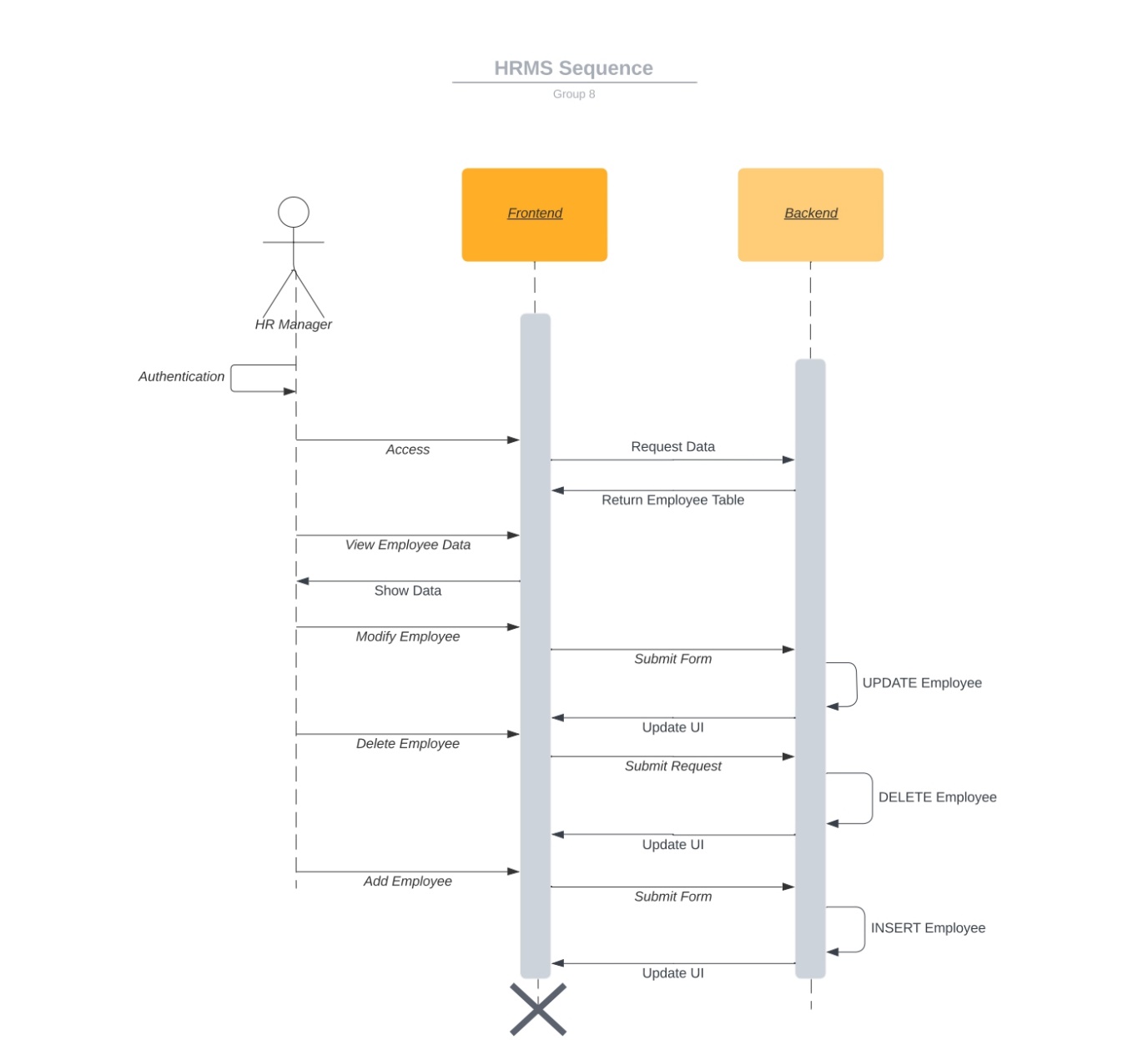


## **6.2** - Activity Diagram: It shows the different actions and how they interconnect between components.

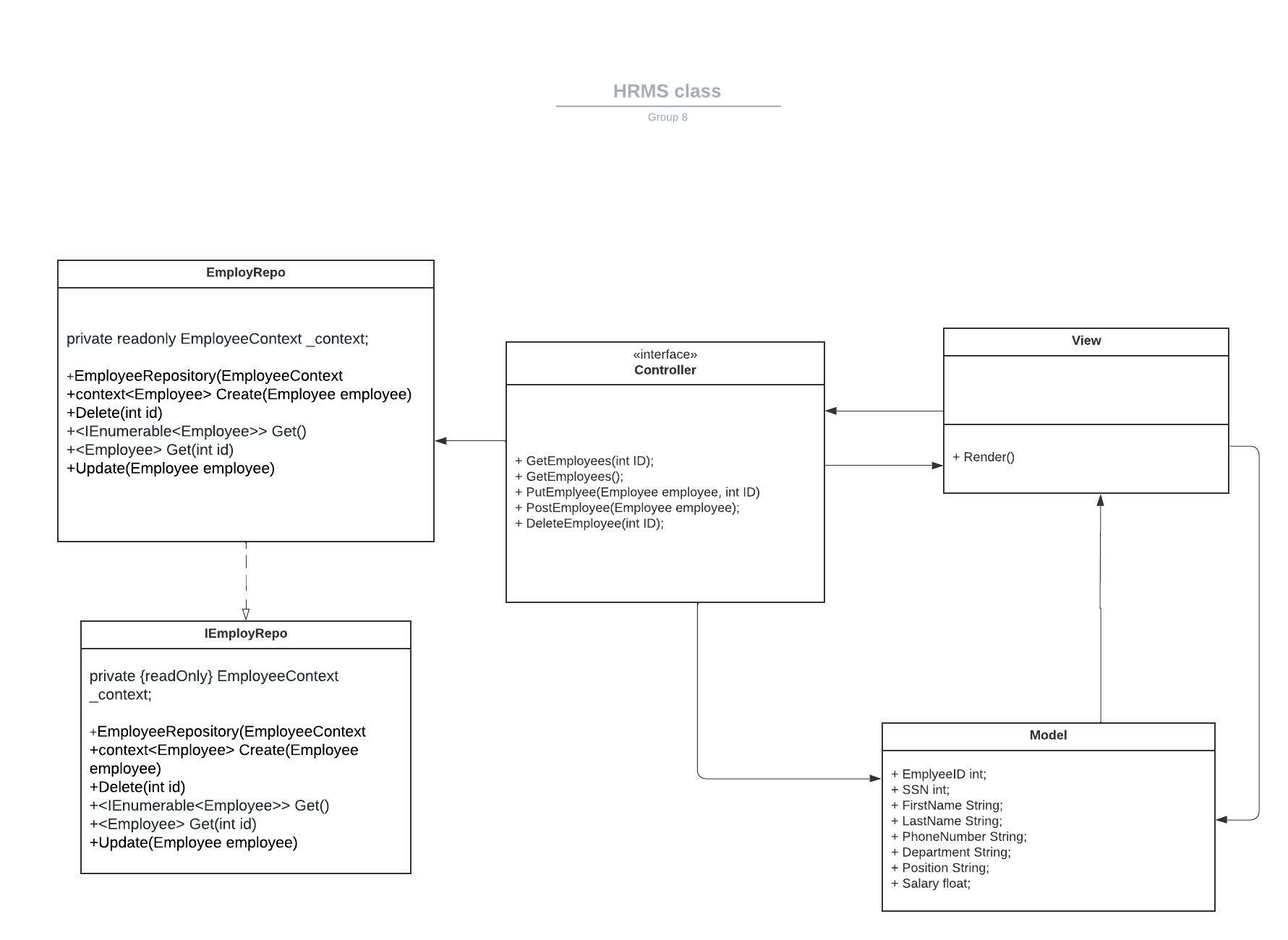
## **6.3** – State Machine Diagram: Shows how the system as a whole switch from one state to another through different actions and also shows which actions are available from which system states.



## **6.4** – Interaction Sequence Diagram: shows how the different system components interact with respect to timing and sequence, which also shows the formation of transactions which are a series of operations which either occur atomically and succeed or do not occur at all.



## **6.5** – Class Diagram: it represents the static structure of the system through the classes written in it, as well as all class attributes, operations, methods and the relationships between the objects derived from the respective classes.



7) Design Description

i) Frontend: React:

The Front-End of the HRMS Software was implemented using React. It consists of two main pages: One which views all of the employees, and the other which consists of a form to add a new employee to the database.

Viewing all the employees is done through sending a GET request to the application API’s in the backend, requesting the data of all employees and outputting them in a table. Each employee record in the table also has three buttons. One to DELETE the employee, one to VIEW the employee data in a card, and one to EDIT the employee. DELETE operation is done by sending a DELETE request to the backend API with the specified employee ID. EDIT operation is done by sending a PUT request with the employee ID, with the body consisting of JSON object of the newly modified employee data.

The addition of a new employee in the database is done by the user by filling out a form with all the new employee data. A POST request is then sent to the backend API with a JSON object in the body of the request, adding a new employee to the database