**HRMS**

**SPROJ Report**



**Aamina Mariam 22100062  
Javeria Tariq 22100230  
Ali Adnan Arif 22100303  
Mohammad Yousuf 22100289**

**Advisor: Waqar Ahmed  
School of Science and Engineering  
Lahore University of Management Sciences  
30th April 2022**

**Acknowledgement and Dedication**

**Certificate**

I certify that the senior project titled “**HRMS**” was completed under my supervision by the following students:

Mohammad Yousuf

Aamina Mariam

Javeria Tariq

Ali Adnan Arif

Talha Nasir

and the project deliverables meet the requirements of the program.

------------------------------------- Date:

**Advisor (Signature)**

------------------------------------- Date:

**Co-advisor (if any)**

# Table of Contents

[**List of Figures**](#_yee7368w64ir) **7**

[**Introduction**](#_7vo6q0i0fyf0) **8**

[Introduction](#_op4ej0s5g4c9) 9

[Objective and Scope](#_6hhccxbd1rqt) 9

[Development Methodology](#_sjrnedo2j0vl) 9

[Contributions](#_akn8ob8glfj1) 9

[**System Requirements**](#_o5vjn8vevwn) **10**

[System Actors](#_4t8qlam5cepv) 10

[Functional Requirements](#_lhl2g0cn1zdr) 10

[Non-functional Requirements](#_5i4vhdlcabqe) 10

[**System Architecture**](#_xbvd1l444kng) **11**

[Architecture Diagram](#_gjdgxs) 11

[Architecture Description](#_30j0zll) 11

[Justification of the Architecture](#_1fob9te) 11

[Tools and Technologies](#_3lmtf7t4ml2j) 11

[**Requirements Specifications**](#_553gbujj8ars)  **12**

[Use Cases](#_chy73oojciai) 12

[Class Diagram](#_2et92p0) 12

[Sequence Diagrams](#_tyjcwt) 12

[**Software Development Methodology and Plan**](#_tm5m7s741foq) **13**

[Software Process Selection](#_3dy6vkm) 13

[Gantt Chart](#_1t3h5sf) 16

[**Database Design and Web Services**](#_nnf0ejrt2e28) **17**

[Database Design](#_81owdbqqubwt) 17

[API Specification](#_4yqice3qsmiw) 17

[**System User Interface**](#_ohi58uvy54tv) **18**

[**Project Security**](#_iglhskglyv3f) **19**

[Project Threats](#_336t7n38bqd7) 19

[Potential Losses](#_712jx6dutuws) 19

[Security Controls](#_rm3xn5vfjsfw) 19

[Static and Dynamic Security Scanning Tools](#_ldji2g8jpkmx) 19

[**Risk Management**](#_fhxpoviibhgr) **20**

[Potential Risks and Mitigation Strategies](#_qn3c332q9xx) 20

[**Testing and Evaluation**](#_w9luo5ktfh1) **26**

[**Deployment Guidelines**](#_geq6fx6x129s) **27**

[**Conclusion**](#_utfw2ms24esa) **28**

[Summary](#_n708j8cay9ca) 28

[Challenges](#_rg8ibhnt6caz) 28

[Future](#_ns4klefbsiae) 28

[**Review checklist**](#_2s8eyo1) **29**

[**References**](#_gf7pdgip1xeu) **30**

# List of Figures

# Introduction

## Introduction

Our project streamlines the complex process of managing and recruiting employees in an organization. We provide a "one stop solution" forSmall and medium-sized enterprises (SMEs) that caters to the HR needs of an organization.

We have made a web application. The target user for this application can be any company that wants a Human Resource Management System with a CV ranking component.

## Objective and Scope

The Human resource management system covers many HR aspects from application to management to promotions. The software keeps track of an organization’s employees and provides analytics of their performance using relevant KPIs. The software combines a number of systems and processes to ensure the ease of management in human resources and business processes. The HRMS software helps HR professionals manage the modern workforce.

Our aim is to assist companies in running effectively and efficiently. The system is a suite of software that companies can use to regulate their internal HR functions. Employee data management, recruitment, benefits, training, talent management, employee engagement, and employee attendance include some of the features our software will provide.

## Development Methodology

## We used agile methodology to complete this project. We started off with a prototype and then added on it in each sprint in a modular fashion. This approach allowed us to work independently on each module and then combine all the modules together. The project took a total of 4 sprints. In the last sprint, we integrated all the modules together to form an integrated system.

## Contributions

We made a list of everything that needed to be done and then assigned work based on our team members' preferences. The backend was largely done by Aamina and Yousuf, while the frontend was mostly done by Javeria and Adnan.

Our project streamlines the complex process of managing and recruiting employees in an organization. We provide a "one stop solution" for Small and medium-sized enterprises (SMEs) that caters to the HR needs of an organization.

Employees can update personal data without involving HR for simple tasks using our Project HRMS. This allows HR experts to focus on more important tasks. Storage is centralized because all of the data is in one location, reporting can be more efficient. HR professionals can streamline the recruitment process by filtering resumes based on role and other variables. Companies can improve employee engagement and reduce turnover by installing an HRMS. It also allows employees to participate effectively and productively in the overall direction of the company as well as the achievement of the organization's goals and objective.

# System Requirements

This chapter is briefly going to talk about our system actors and their jobs. For each actor their respective functional requirements will be mentioned in detail. And finally it will talk about the non functional requirements of our system/

## System Actors

|  |  |
| --- | --- |
| Actor Name | Description |
| HR manager | Will have access to information and statistics of all employees. |
| System Administrator | The manager of the website |
| Dept head | Will have access to the information of employees in their own department and will be able to enter information regarding each employee |
| Employee | Will have access to their information/analytics dashboard |

## 

## Functional Requirements

|  |  |
| --- | --- |
| **Requirements** | |
| **Sr#** | **Requirement** |
|  | **HR Manager** |
| **1** | I want to be able to view monthly/quarterly employee performance statistics (completion of task on time, task quality, behavior rating ) |
| **2** | I would like to send my emails straight through the system |
| **3** | I want to add new employees to the database |
| **4** | I want the system to be able to filter out potential candidates for a job and rank their resumes for me. |
| **5** | I want the system to use the CV scanner to parse an employee’s resume. |
| **6** | I want to be able to transfer/promote/terminate employee roles from the system |
| **7** | I want to see company statistics such as the number of active employees, latest news , new hires, open job vacancies and more. |
| **8** | I want to have access to a timeline to see when changes to compensation occurred during an employees lifetime |
| **9** | I want to be able to search employees through filtering (by department, skills, education etc) |
| **10** | I want to monitoremployees’ working hours and absences, vacation and sick leave accruals and job contract terminations. |
| **11** | I want to have access to the company organization hierarchy |
| **12** | I would like to be able to add news and announcements (e.g., about new job postings) |
|  | **Department Head** |
| **1** | I want to be able to search the content of database for the employees who are under my coverage |
| **2** | I want to be able to view monthly/quarterly employee performance statistics (completion of task on time, task quality, behavior rating ) - Employee KPI tracking (e.g., open/completed tasks, overtimes). |
| **3** | I would like to send my emails straight through the system |
| **4** | I would like to approve or reject employee requests |
| **5** | I want to have access to a timeline to see when changes to compensation occurred during an employees lifetime |
| **6** | I want to be able to add comments for each employee to describe their achievements and flaws |
| **7** | I want to be able to search employees through filtering (by department, skills, education etc) |
| **8** | I want individual employee performance reviews to be scheduled automatically and for employees to be notified in advance about them. |
| **9** | I want to monitoremployees’ working hours and absences, vacation and sick leave accruals and job contract terminations. |
|  | **Employee** |
| **1** | I want to be able to put in requests for sick days, personal days, vacation days, and volunteer hours |
| **2** | I want to be notified about the approval or rejection of my request |
| **3** | I want to view my own performance statistics through my dashboard |
| **4** | I want to have access to visualized reports of all my performance reviews |
| **5** | I want to have access to easy answers to employees’ common HR-related questions. |
|  | **System Administrator** |
| **1** | I want to have administrator level access to the system and its database. |
|  | **All Actors** |
| **1** | I want to login & from the system with my username and password. |
| **2** | I want to see the appropriate user interfaces after logging in (according to my role) |
| **3** | I want to have access to automated notifications (eg: contract expiration) |
| **4** | I would like to be able to read recent news and announcements (e.g., about new HR policies) |

## Non-functional Requirements

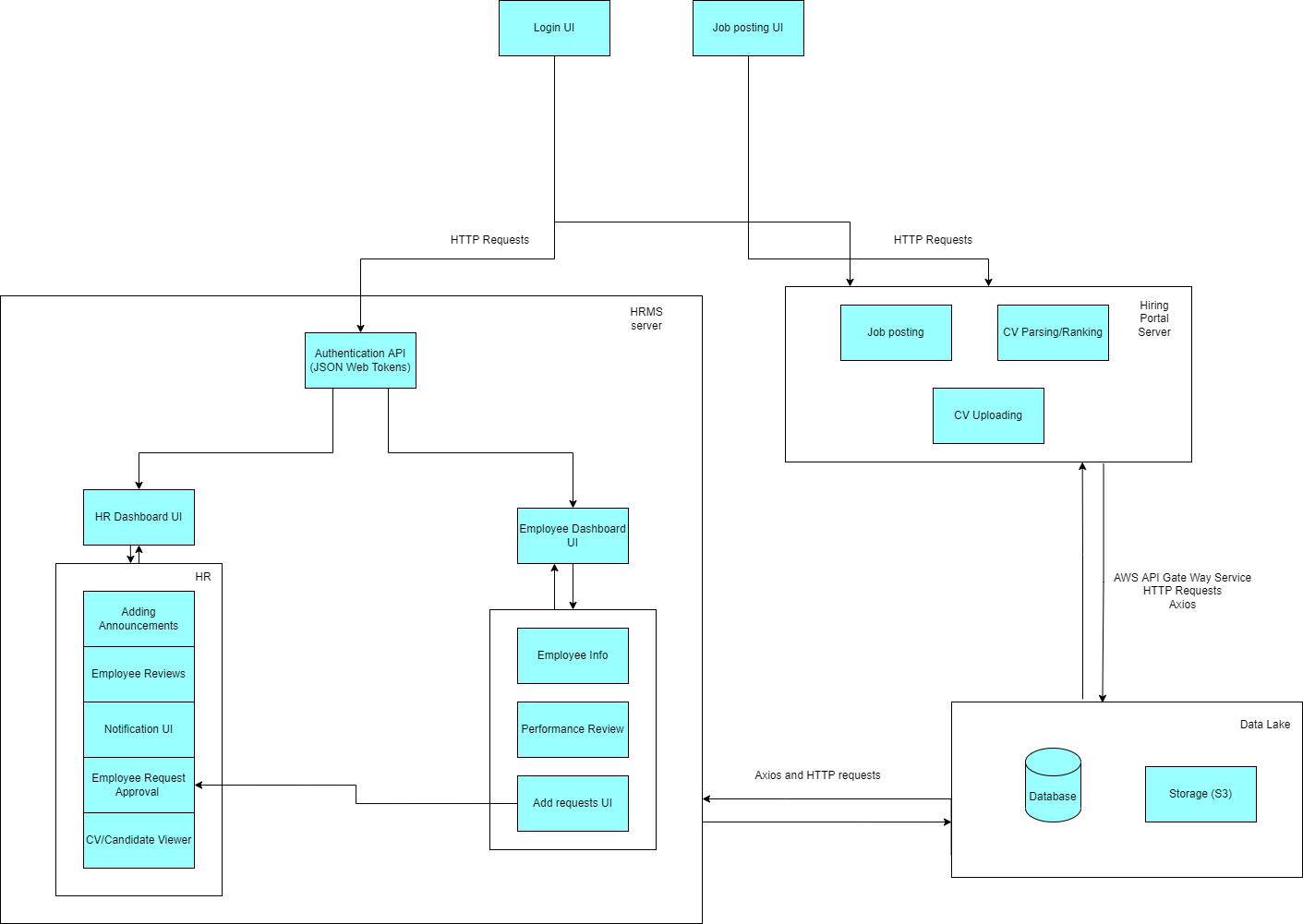
|  |  |
| --- | --- |
| **Sr#** | **Requirements** |
| 1 | The system shall not fail more than 3 times every 24 hours. In case of a failure, the system should restore to normal operations within 5 minutes of a failure. |
| 2 | Any interaction between the user and the system must have a maximum response time of 4 seconds. In cases where more time is required by the system, the system must display the progress |
| 3 | Personal data of employees shall only be available to the HR manager and System Manager. |
| 4 | Sensitive information of employees such as passwords and personal information shall be encrypted and must not be available to anyone except the system administrator. |
| 5 | The number of the simultaneous users of the system can accommodate shall be 50 |
| 6 | Only 5000 employees can be added to the database. |
| 7 | The system shall be reliable i.e information entered should be stored successfully. |
| 8 | System shall distinguish authorization levels based on the actor. Authorization levels will be hierarchical ( e.g Dept heads will be able to access information of their subordinates and the subordinates will be able to access data of the employees working under them respectively.) |
| 9 | The software shall be used on PCs and be functional via the internet using all major web browsers i.e Safari, Chrome and Firefox. |
| 10 | The website shall display errors/notifications in cases of failures  For unexpected failures the users should be notified and they should be able to go back to the previous page. |
| 11 | At least 20% of the processor and RAM capacity shall be unused at peak load periods. |

# 

# 3. System Architecture

This chapter will define the structure and behavior of our system. The diagram gives an overview of our system and the main components of our system are also explained with an in-detail justification of the architecture

## Architecture Diagram



## Architecture Description

**HRMS Server and Hiring Portal Server**

This subsystem processes the user’s requests and sends all of the required documents (based on the request) back to their browser. The required documents will usually consist of files such as JSON and XML. The way this subsystem works is that it calls the back-end infrastructures such as the database, etc. The server will be hosted on Amazon Web Services (AWS).

**Database**

The database subsystem contains information to be used for performing computations. The web application server can interact with the database to complete the information updates of the whole system such as adding, deleting, searching and organizing user data.

**Data Lake**

A data lake is a centralized repository that allows you to store all your structured and unstructured data. Since our system consists of data from two sources: employees and candidates, the data lake will act as a central repository for both of them. The application will interact with the data lake to provide analytics dashboards and for searching/querying data.

**Login (JSON Web Tokens)**

The Json web tokens will provide secure user authentication for the web application that will allow a user to login according to their role.

**Cloud Storage(S3)**

S3 is a storage container for storing the CVs and resumes of applicants. When a job applicant uploads their resume/CV it gets stored in the S3 container.

## Justification of the Architecture

List down pros and cons of the architecture you have defined. These pros and cons must be discussed in the context of your system. Moreover, give a justification of why this architecture is appropriate for your system. Make sure that you also discuss how this architecture helps in implementation of your system’s non-functional requirements.

We have chosen a hybrid between client server architecture and layered architecture. This architecture is specifically useful for our application since we require a separation of concerns between the client and the server along with the layered architecture separating the system functionality into layers. Furthermore, this also allows the candidate applying for a job to add their CV/resume without having to login to the internal HR portal of the company. This architecture is best suited for our application as it can cater to a large number of users without being overwhelmed and store and process data according to the requirements of the system. This also allows our system to scale both horizontally and vertically while ensuring the modularity of the system. Our architecture also ensures the security of sensitive user data like passwords by storing them after using bcrypt to hash the data. Json web tokens are being used for session management and authorization in our system.

Moreover, the architecture uses a combination of server and serverless protocols. This strategy optimizes performance and cost of operating the system while ensuring that there is no single point of failure. The use of serverless protocols for handling data intensive workloads (like AI chatbot and search queries) ensures efficient utilization of resources while providing fast and reliable performance. We have decided to use a DynamoDB as our database as it provides a serverless, key-value NoSQL database for high-performance applications at scale. DynamoDB provides an efficient and scalable solution for building ad-hoc and on-demand dashboards required by our system.

Lastly, we went with having an architecture with multiple servers and only one database. This model doesn’t store any data. When the client sends information to the web server, it is processed and written to the database, but managing this data takes place outside of the server. It’s called stateless architecture. It’s important to make our model reliable because if one server goes down, another one will take charge. So, in such a failure, all the requests will automatically go to the new server, without affecting the systems functioning. Thus, this model is more reliable than a single server. However, if something happens to the database, the app will crash.

The hybrid approach we have used also makes the system architecture more complex. This added complexity makes the system and its various functionalities difficult for a new developer to understand without interacting with the system.

## Tools and Technologies

● Backend: Django 3.8.2 or 4.0

● Frontend: React 17.0

● Databases: Amazon DynamoDB

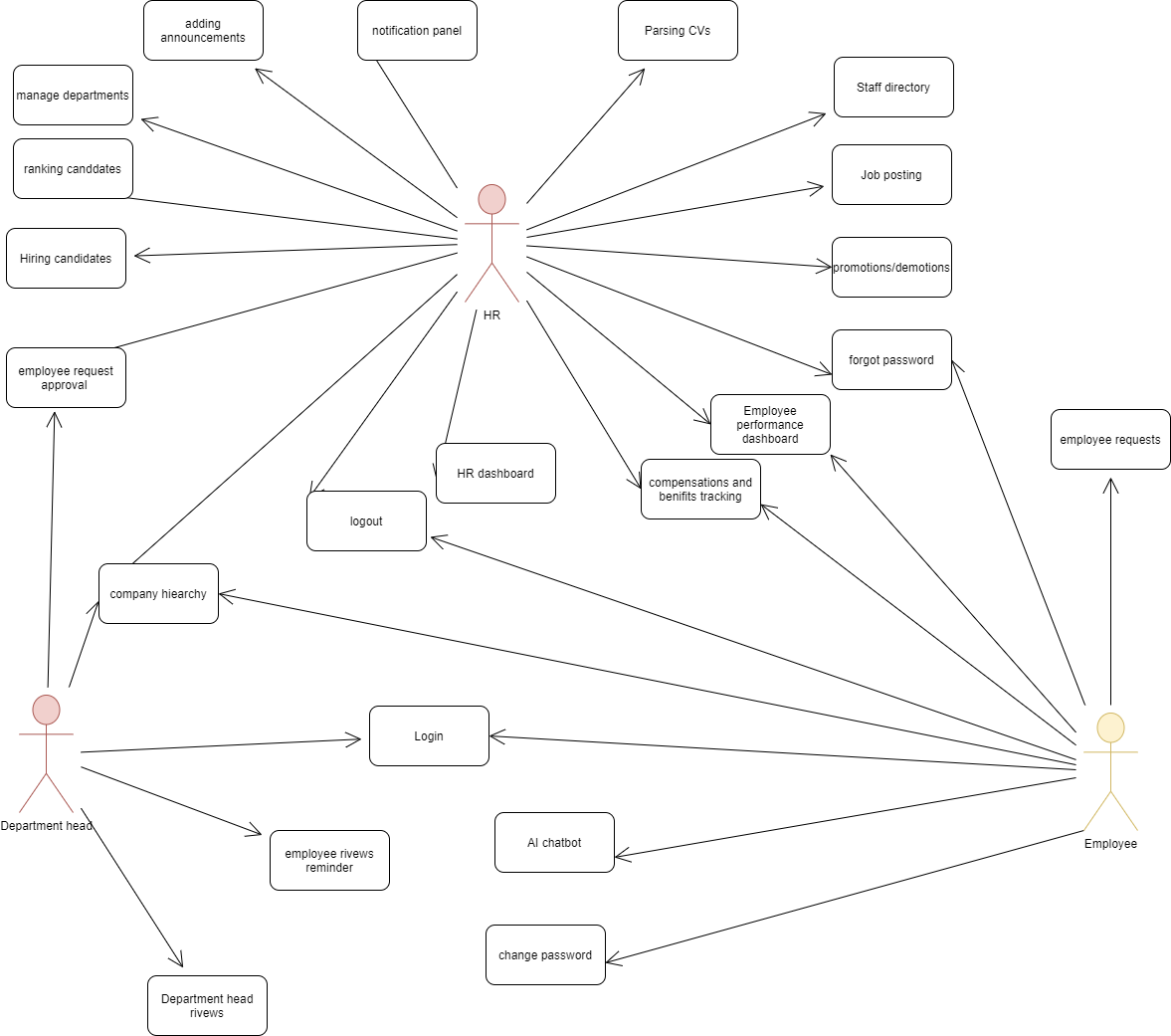
● Servers/Serverless: AWS EC2

# 

# Requirements Specifications

In this we will be discussing the major use cases of our system and then explaining them in detail in section 4.2.

## 4.1. Use Case Diagram



## 

## 

## 4.2 Use Cases

**4.2.1 Login**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-001 |
| **Purpose** | | A registered user login to system to access the functionality of system |
| **Pre-conditions** | | A user must have an account and the system must be connected to the network. |
| **Post-conditions** | | The user must be authorized to use the system. |
| **Step #** | **Typical Course of Action** | |
| **1.** | Users enter their username and password | |
| **2.** | The username and password are validated by the system. | |
| **3.** | User is allowed access to system | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | In step 1, if the username or password is invalid, the system shows an error message | |
| **Step #** | **Exception Paths** | |
|  | None | |

**4.2.2 Logout**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-002 |
| **Purpose** | | User log off from the system |
| **Pre-conditions** | | User must be logged in |
| **Post-conditions** | | The user is logged out of the system |
| **Step #** | **Typical Course of Action** | |
| **1.** | User clicks on the logout button and clicks confirm | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None. | |
| **Step #** | **Exception Paths** | |
| **1.** | None. | |

**4.2.3 Forgot Password**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-003 |
| **Purpose** | | A registered user trying to get a new password. |
| **Pre-conditions** | | A user must have an account. |
| **Post-conditions** | | Password will be changed` |
| **Step #** | **Typical Course of Action** | |
| **1.** | Enter Username | |
| **2.** | Enter password | |
| **3.** | Validate Username through email. | |
| **4.** | Reset password | |
| **5.** | System will let the user know that their password has been changed. | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | If an invalid username is entered, the system shows an error message | |

**4.2.4 Job Promotions/Demotions**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-004 |
| **Purpose** | | The HR manager wants to update the position/rank of an employee |
| **Pre-conditions** | | The HR manager manager must be logged in |
| **Post-conditions** | | The employee is promoted or demoted. |
| **Step #** | **Typical Course of Action** | |
| **1.** | HR manager clicks manage employees on the HR portal | |
| **2.** | A list of all employees will be presented to the HR manager | |
| **3.** | HR manager will select the employee | |
| **4.** | The profile of the employee will open up and the HR manager will be able to change/edit the information of the employee. | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | HR manager clicks manage employees on the HR dashboard | |
| **2.** | A list of all employees will be presented to the HR manager | |
| **3.** | HR manager will search for the name of the employee | |
| **4.** | HR manager will select the employee | |
| **5.** | The profile of the employee will open up and the HR manager will be able to change/edit the information of the employee. | |
| **Step #** | **Exception Paths** | |
|  | None | |

**4.2.5 HR Dashboard**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-005 |
| **Purpose** | | To provide an overview of the company and serve and give access to HR’s functionality |
| **Pre-conditions** | | The HR manager manager must have logged in |
| **Post-conditions** | | The overview is provided |
| **Step #** | **Typical Course of Action** | |
| **1.** | The HR staff goes to the website home page | |
| **2.** | The homepage requests their id and password | |
| **3.** | The user enters the information | |
| **4.** | The system loads the users home page | |
| **5.** | The user goes to their profile | |
| **6.** | The user selects the HR dashboard option | |
| **7.** | If the user has access, the Dashboard containing company information and HR functionality will show up | |
| **8.** | The use case ends. | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
|  |  | |
| **Step #** | **Exception Paths** | |
| **1.** | In step 7, if the user does not have access, then an error message is displayed | |

**4.2.6 Hiring Candidates**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-006 |
| **Purpose** | | Hiring candidates as employees and adding them to the system |
| **Pre-conditions** | | The HR Manager must be logged into the hiring portal. |
| **Post-conditions** | | The candidate is hired and added as an employee. |
| **Step #** | **Typical Course of Action** | |
| **1.** | The HR manager retrieves the CV and data of the candidate by searching for it in the job postings list. | |
| **2.** | The HR manager chooses the desired applicant by clicking Hire. | |
| **3.** | The candidate is added to the employee database | |
| **4.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

**4.2.7 Manage Departments**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-007 |
| **Purpose** | | To add or remove departments or employees from the database |
| **Pre-conditions** | | The user must be logged in from their HR account. |
| **Post-conditions** | | Departments or employees will be added/removed |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user goes to the manage departments option | |
| **2.** | The user clicks add or remove department | |
| **3.** | The system displays a drop down menu where they can either add a new department or add an employee to a specific department | |
| **4.** | The user clicks on the department that they want to add an employee to | |
| **5.** | The user enters the information of the employee | |
| **6.** | The user clicks add | |
| **7.** | Dept Head receives a notification after the database has been updated. | |
| **8.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | After step 2, the user can choose to add a new department. | |
| **2.** | The user clicks the ‘add button’ | |
| **3.** | A new department is added to the system | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

**4.2.8 Change Password**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-008 |
| **Purpose** | | The user goes to their profile to change their password |
| **Pre-conditions** | | The user has logged into their account successfully |
| **Post-conditions** | | The password is changed in the system |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user goes to their profile | |
| **2.** | The user selects the change password option | |
| **3.** | The user confirms their new password | |
| **4.** | If the user has successfully entered the new password twice, their new password is entered into the system and can be used to log-in in the future. | |
| **5.** | The use case ends. | |
| **Step #** | **Alternate Courses of Action** | |
|  | None | |
| **Step #** | **Exception Paths** | |
| **1.** | In step 7, if the passwords don’t match, then an error message is displayed and the user is asked to re-enter the passwords. | |

**4.2.9 Parsing CVs**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-009 |
| **Purpose** | | Parsing CV for shortlisting of suitable candidates |
| **Pre-conditions** | | The CV’s of candidates are in proper format |
| **Post-conditions** | | The parsed CV’s are then filtered |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The candidate uploads the cv in the job posting | |
| **2.** | The system will extract info from the CV | |
| **3.** | They are then parsed according to the relevant job descriptions. | |
| **4.** | The use case ends. | |
|  | | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

**4.2.10 Staff directory**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-011 |
| **Purpose** | | For the HR manager to search for employees through filtering (by name, department , skills , education and project names they have done) |
| **Pre-conditions** | | The HR manager has logged in. |
| **Post-conditions** | | Employee profile can be selected |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user will navigate to the search option in their dashboard. | |
| **2.** | The user can search by name & select any of the filters | |
| **3.** | The user will then select the employee whose profile they want to visit | |
| **4.** | Their profile is displayed | |
| **5.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | In step 2, the user can choose to not select any filters (and not enter a name). Instead they may scroll down the employee list (arranged alphabetically) | |
| **Step #** | **Exception Paths** | |
| **1.** | In step 3, if the user does not have access to the employee’s profile, an error message will be displayed | |

**4.2.11 Employee requests**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-012 |
| **Purpose** | | Employees can apply for sick days, personal days, vacation days, and volunteer hours |
| **Pre-conditions** | | The employee must be logged in and should be on the dashboard screen |
| **Post-conditions** | | A request is sent to the manager/head of the team for approval |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user goes to the website home page | |
| **2.** | The homepage requests their id and password | |
| **3.** | The user enters the information | |
| **4.** | The system loads the users home page | |
| **5.** | The user clicks on the ‘employee requests’ option | |
| **6.** | The user clicks on the apply for leave option | |
| **7.** | A calendar pops up and the employee selects a start date | |
| **8.** | The employee then selects an end date | |
| **9.** | The employee adds their reasoning in a text box | |
| **10.** | The employee clicks send and the application is sent to the department head | |
| **11.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | In step 6, the employee will have a variety of other options (sick days, personal days, vacation days, and volunteer hours) | |
| **Step #** | **Exception Paths** | |
| **1.** | If the starting/ending date is not selected or a date from the past is selected the application will not be sent and an error message will be shown | |

**4.2.12 Employee reviews**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-013 |
| **Purpose** | | For employee reviews to be added to their record . |
| **Pre-conditions** | | The Immediate Manager must be logged in with their account and must be authorized to give reviews. |
| **Post-conditions** | | Reviews for employees will be visible on the HR dashboard. |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user selects an employee from their department | |
| **2.** | The user clicks on the ‘employee review’ option | |
| **3.** | The user is taken to a new screen with the previous reviews. | |
| **4.** | The user can wish to edit old reviews or add new ones | |
| **5.** | The use case ends. | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | In step 3, if the user does not have the authority to edit the employee’s review, an error message will be shown | |

**4.2.13 Employee performance analytics dashboard**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-014 |
| **Purpose** | | For users to see the performance analytics of themselves or others. |
| **Pre-conditions** | | The user should be logged in and have access to the performance analytics of the individual whose dashboard they wish to see |
| **Post-conditions** | | The dashboard is displayed with all relevant statistics |
|  | | |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user clicks on the performance analytics option | |
| **2.** | The user is asked to enter the ID of the employee whose performance analytics they wish to see | |
| **3.** | The system determines if the user has access to the analytics of the particular individual | |
| **4.** | If the user has access, the dashboard is displayed. | |
| **5.** | The use case ends. | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | In step 6, the customer can cancel the transaction and go directly to step 9. | |
| **Step #** | **Exception Paths** | |
| **1.** | In step 7, if the user does not have access, then an error message is displayed and execution proceeds to step 9. | |

**4.2.14 Adding announcements**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-015 |
| **Purpose** | | For HR to add announcements to the notification panel of employees / department heads. |
| **Pre-conditions** | | The user should be logged in from their HR account |
| **Post-conditions** | | The announcement is posted |
| **Step #** | **Typical Course of Action** | |
| **1.** | The user goes to the announcements option | |
| **2.** | The user clicks on add announcement | |
| **3.** | The user enters the message to be posted | |
| **4.** | The use case ends. | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

**4.2.15 Notification panel**

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-016 | |
| **Purpose** | | For users to be able to read recent news and announcements (e.g., about new HR policies or request approval / rejection) | |
| **Pre-conditions** | | User must be logged in | |
| **Post-conditions** | | Recent news and announcements are displayed | |
| **Step #** | **Typical Course of Action** | |  |
| **1.** | User must click on the notification panel on the top right | |  |
| **2.** | The recent announcements are displayed | |  |
| **3.** | The user clicks on an announcement of their choice | |  |
| **4.** | The announcement is fully shown on a seperate screen | |  |
| **5.** | The use case ends. | |  |
| **Step #** | **Alternate Courses of Action** | |  |
| **1.** | After step 2, the user can decide to move straight to step 5.. | |  |
| **Step #** | **Exception Paths** | |  |
| **1.** | After step 1, if there are no new announcements, then the user moves to step 5 | |  |

**4.2.16 Employee request approval/denial**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-017 |
| **Purpose** | | For manager to approve or deny employee requests regarding sick days, personal days, vacation days, and volunteer hours |
| **Pre-conditions** | | User must be logged with their department head account |
| **Post-conditions** | | Request will be approved or denied |
| **Step #** | **Typical Course of Action** | |
| **1.** | Users must click on the notification panel. | |
| **2.** | The recent announcements are displayed. | |
| **3.** | The user clicks on an announcement regarding employee request | |
| **4.** | The user clicks either accept or reject | |
| **5.** | The result is forwarded to the employee’s notification panel | |
| **6.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | After step 4, the user can decide to ignore the request. | |
| **Step #** | **Exception Paths** | |
| **1.** | After step 1, if there are no new announcements, then the user moves to step 5 | |

**4.2.17 Employee reviews reminder**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-018 |
| **Purpose** | | To act as a reminder for managers/department heads to add a performance review for their employee |
| **Pre-conditions** | | User must be logged in with their department head account |
| **Post-conditions** | | Employee Review section will be updates |
| **Step #** | **Typical Course of Action** | |
| **1.** | Users must click on the notification panel. | |
| **2.** | The recent announcements are displayed. | |
| **3.** | The user clicks on an announcement regarding employee performance review reminder | |
| **4.** | The user is taken to the employee profile where they can update the performance of the employee using KPIs. | |
| **5.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | After step 2, the user can decide to ignore/delete the notification. | |
| **Step #** | **Exception Paths** | |
| **1.** | After step 1, if there are no new announcements, then the user moves to step 5 | |

**4.2.18 Hierarchy Tree**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-019 |
| **Purpose** | | To display information of the organization's hierarchy to the HR manager |
| **Pre-conditions** | | The HR manager needs to be logged in |
| **Post-conditions** | | The tree is displayed |
| **Step #** | **Typical Course of Action** | |
| **1.** | The HR manager clicks on the company hierarchy button | |
| **2.** | The organization's hierarchy is displayed. | |
| **3.** | The use case ends. | |
| **Step #** | **Alternate Courses of Action** | |
| **1**. | None | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

**4.2.19 Hiring portal/ Collecting CVs**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-020 |
| **Purpose** | | To collect CVs and other information relevant to available jobs |
| **Pre-conditions** | | User must be logged with their HR manager account |
| **Post-conditions** | | User will have collected the CVs and other information |
| **Step #** | **Typical Course of Action** | |
| **1.** | User must click on the hiring portal panel | |
| **2.** | The uploaded cvs are displayed | |
| **3.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

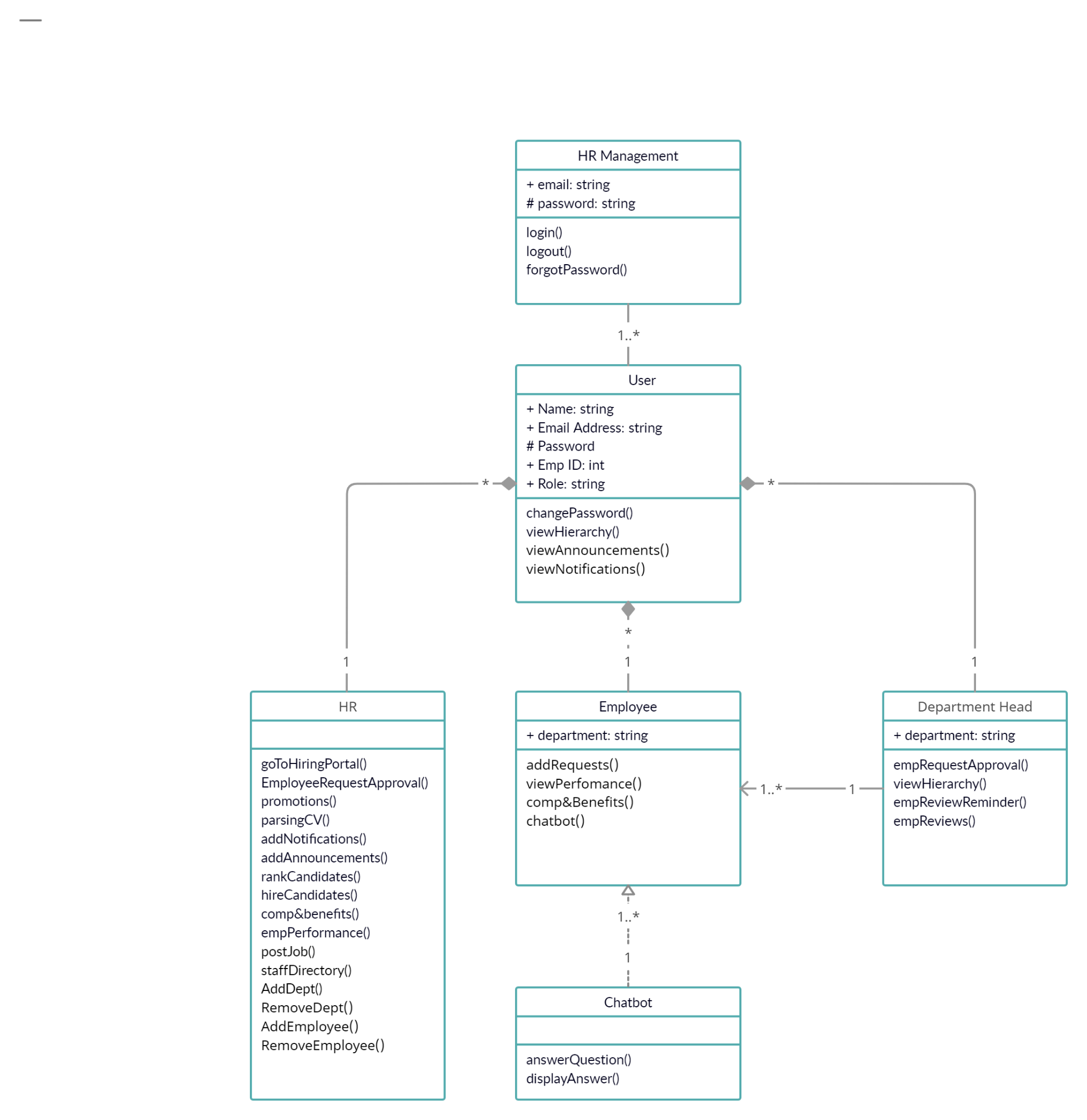
**4.2.20 Job posting**

|  |  |  |
| --- | --- | --- |
| **Identifier** | | UC-021 |
| **Purpose** | | To post available jobs on HR portal |
| **Pre-conditions** | | User must be logged with their HR manager account |
| **Post-conditions** | | Available jobs will be posted on HR portal |
| **Step #** | **Typical Course of Action** | |
| **1.** | User must click on the hiring portal panel | |
| **2.** | User must click on ‘add job’ | |
| **3.** | User must fill the information regarding requirements of job | |
| **4.** | User must click confirm | |
| **5.** | The use case ends | |
| **Step #** | **Alternate Courses of Action** | |
| **1.** | None | |
| **Step #** | **Exception Paths** | |
| **1.** | None | |

## 

## 4.3 Class Diagram

Use standard UML notation to draw the class diagram. Give a brief description/purpose of each class in the class diagram. Give readable names to classes, attributes and operations.



## 4.4 Sequence Diagrams

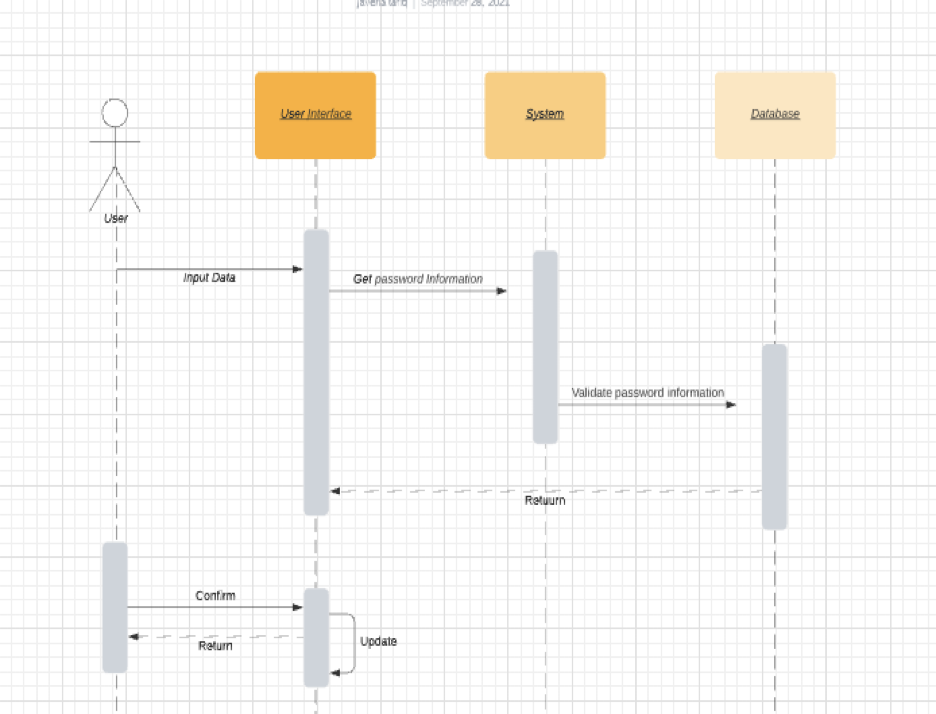
# 4.4.1 Login

# 

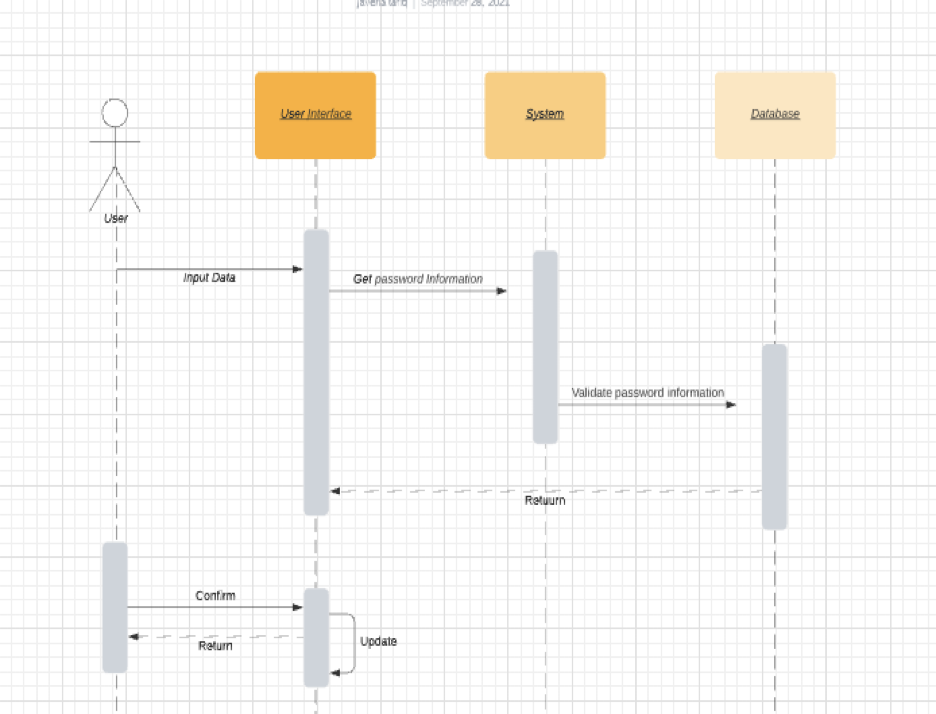
# 4.4.2 Logout

# 

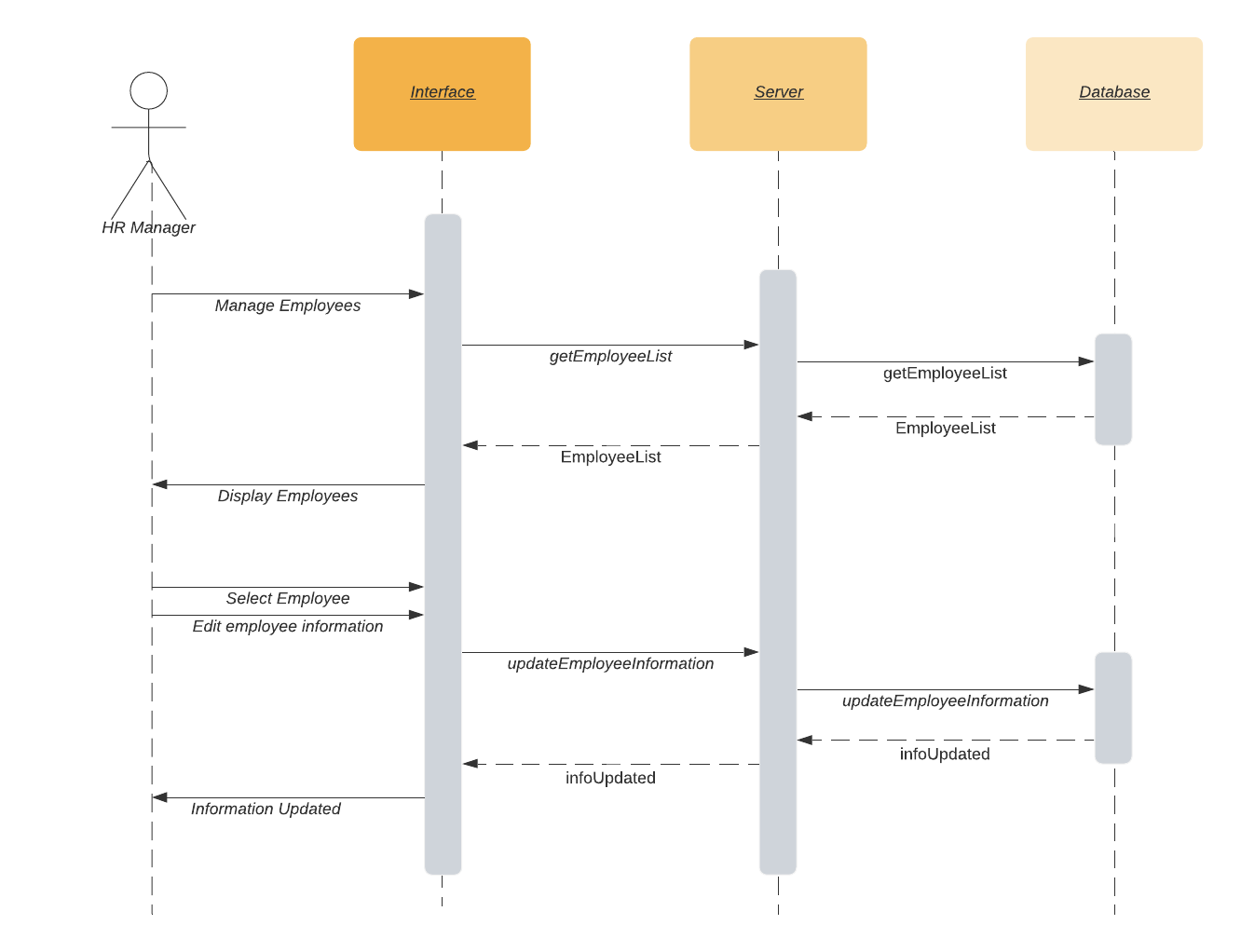
**4.4.3 Change Password**



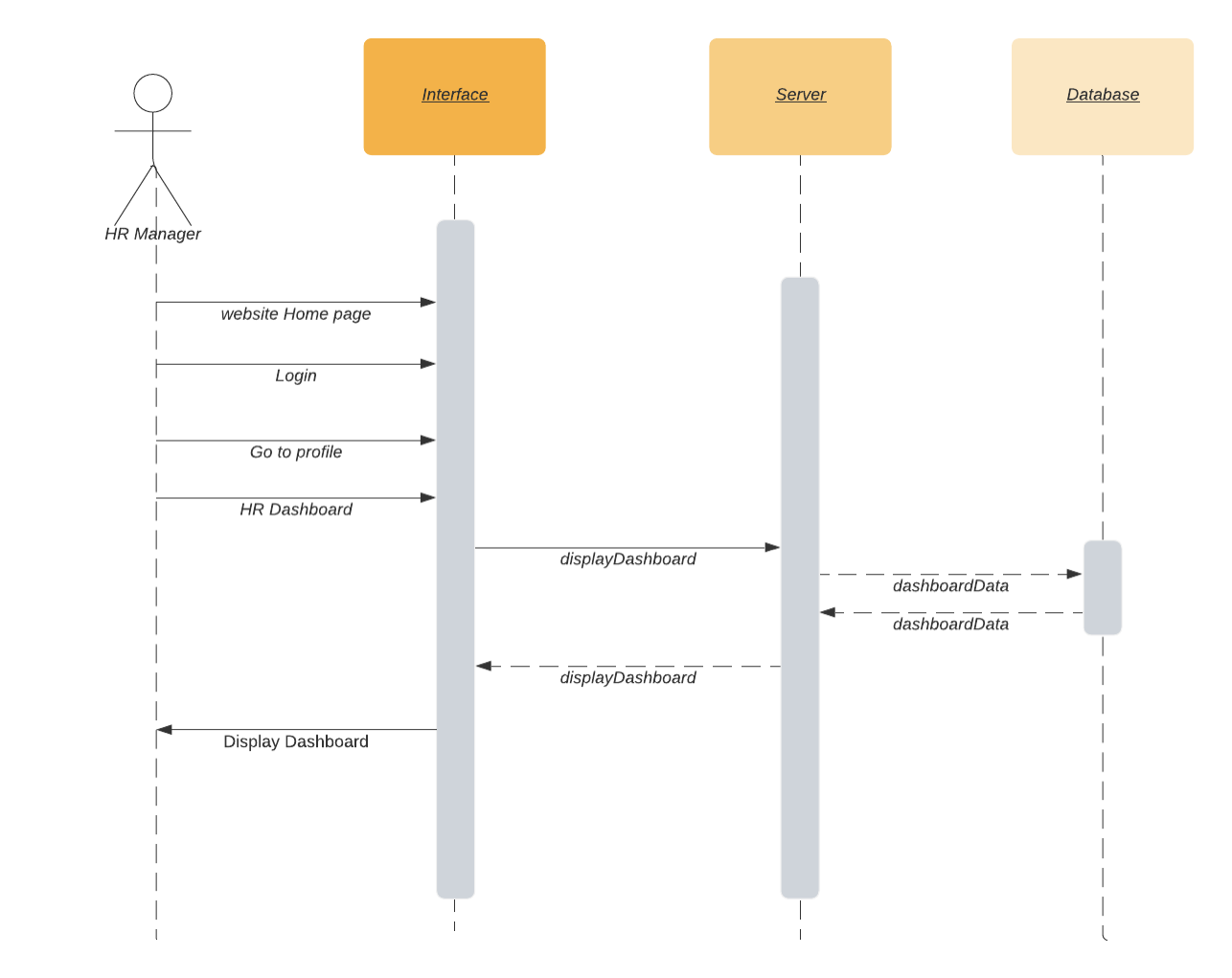
**4.4.4 Forgot Password**



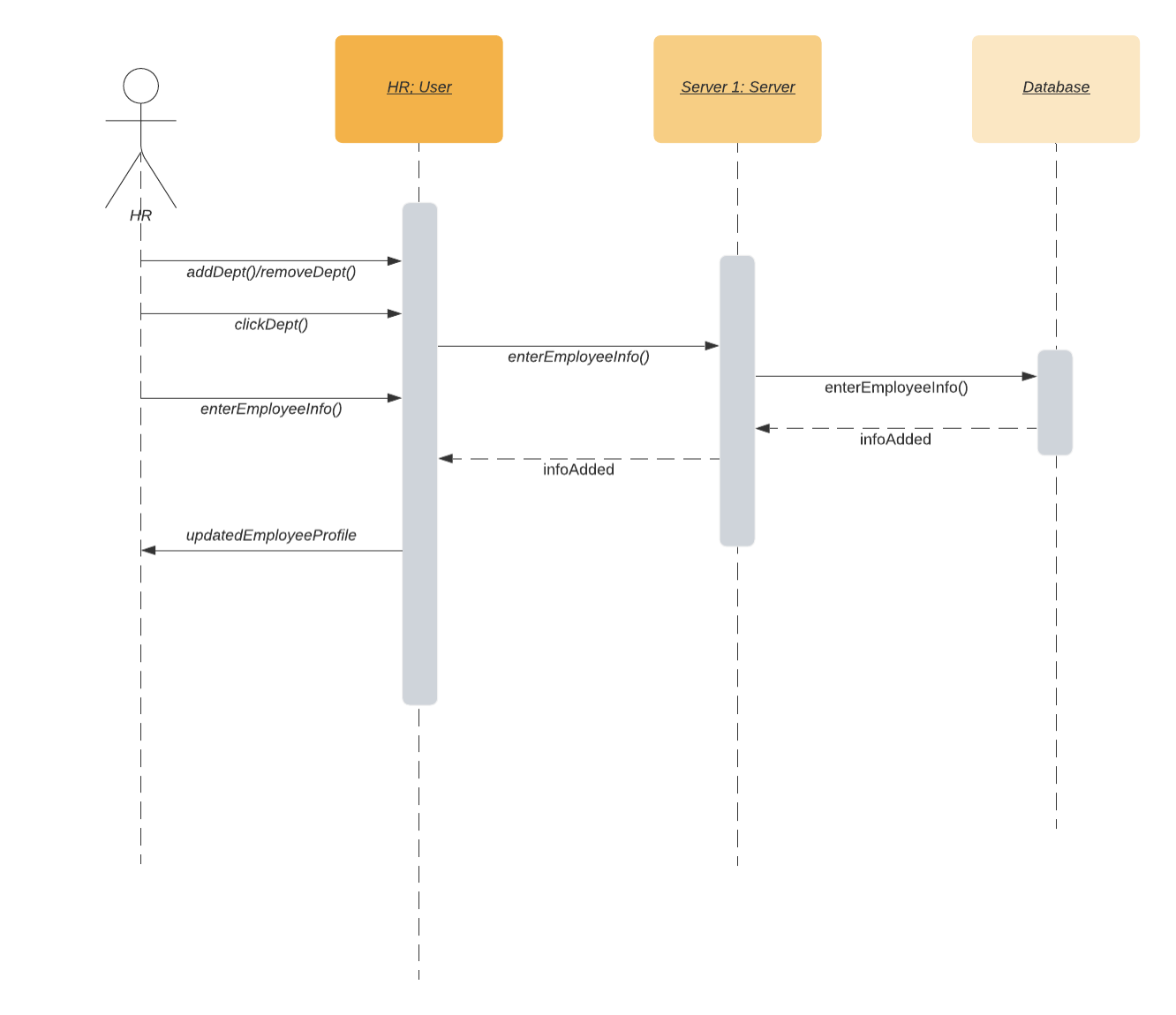
**4.4.5 Job promotions/demotions**



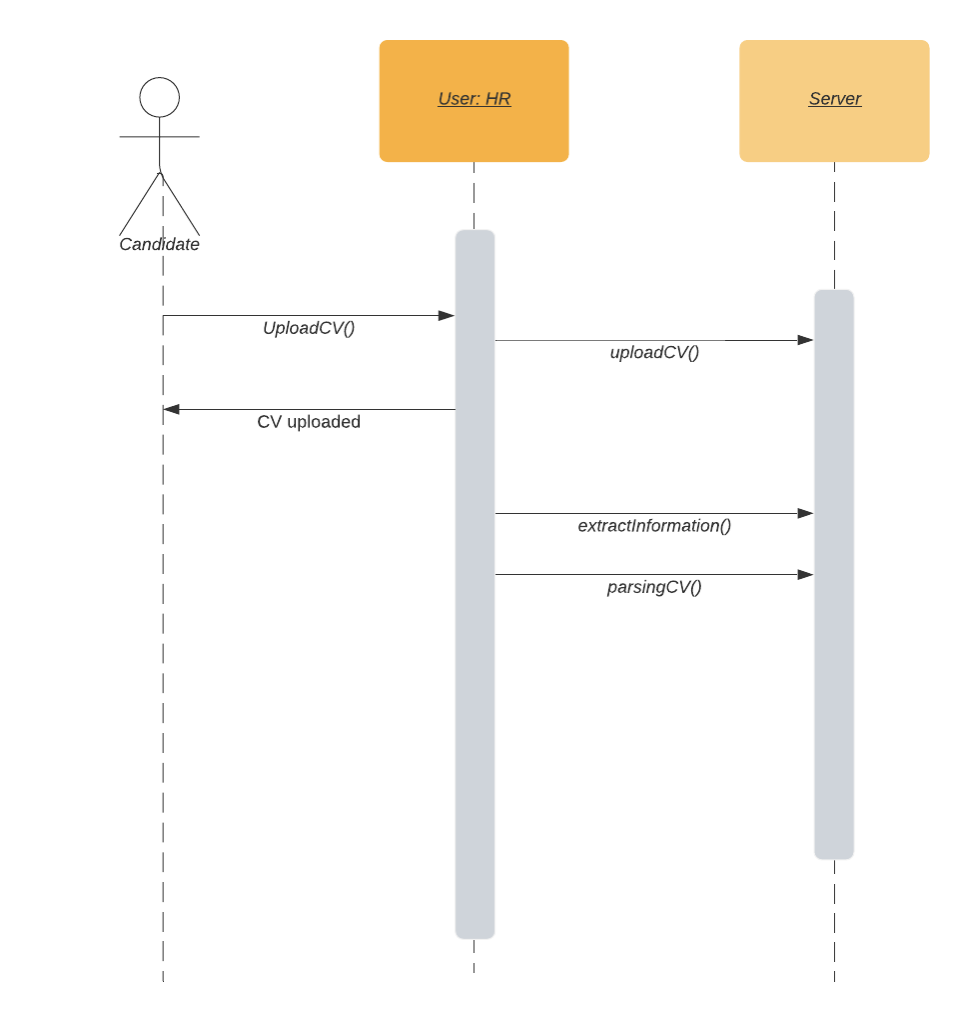
**4.4.6 HR Dashboard**



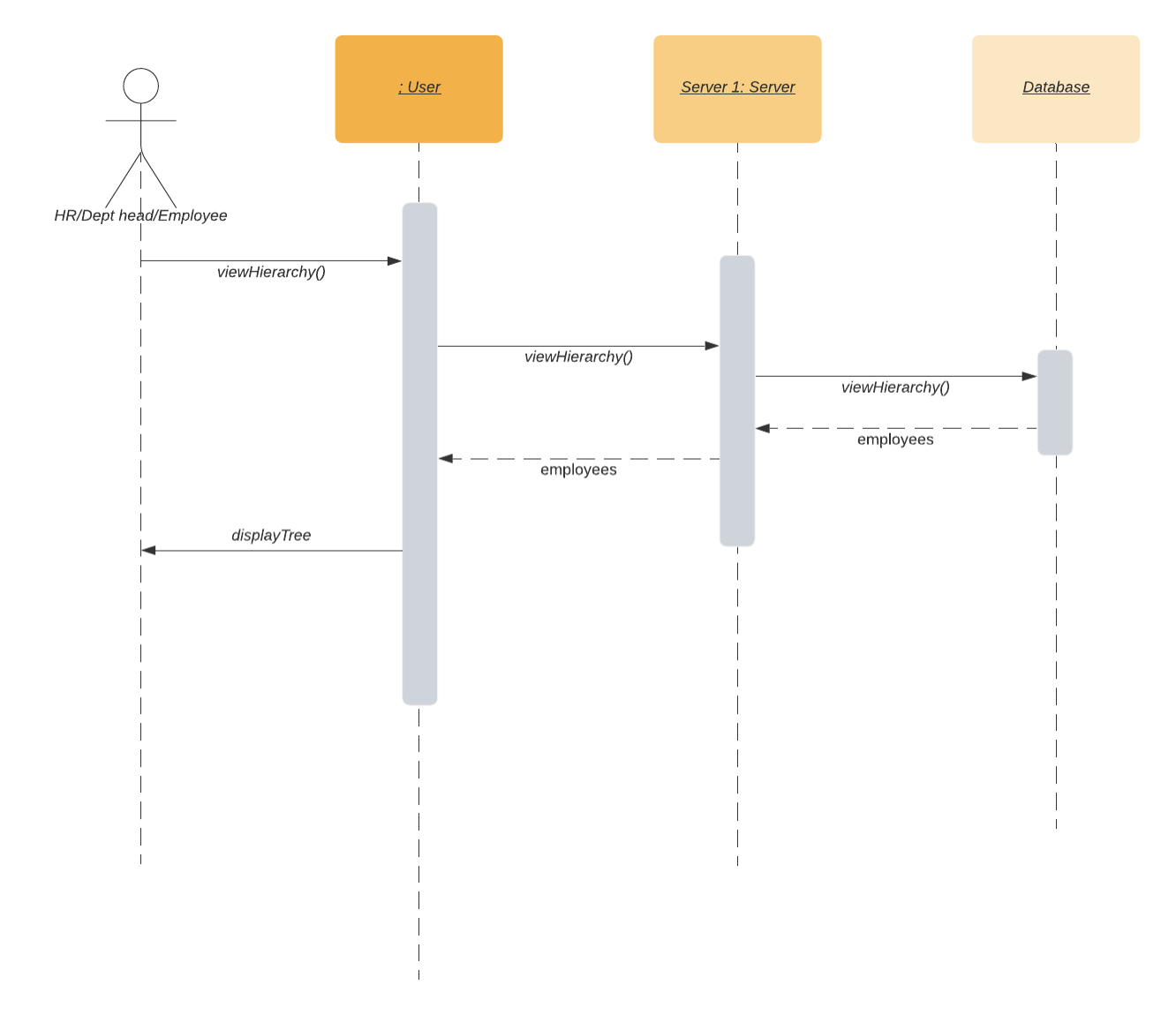
**4.4.7 Manage Departments**



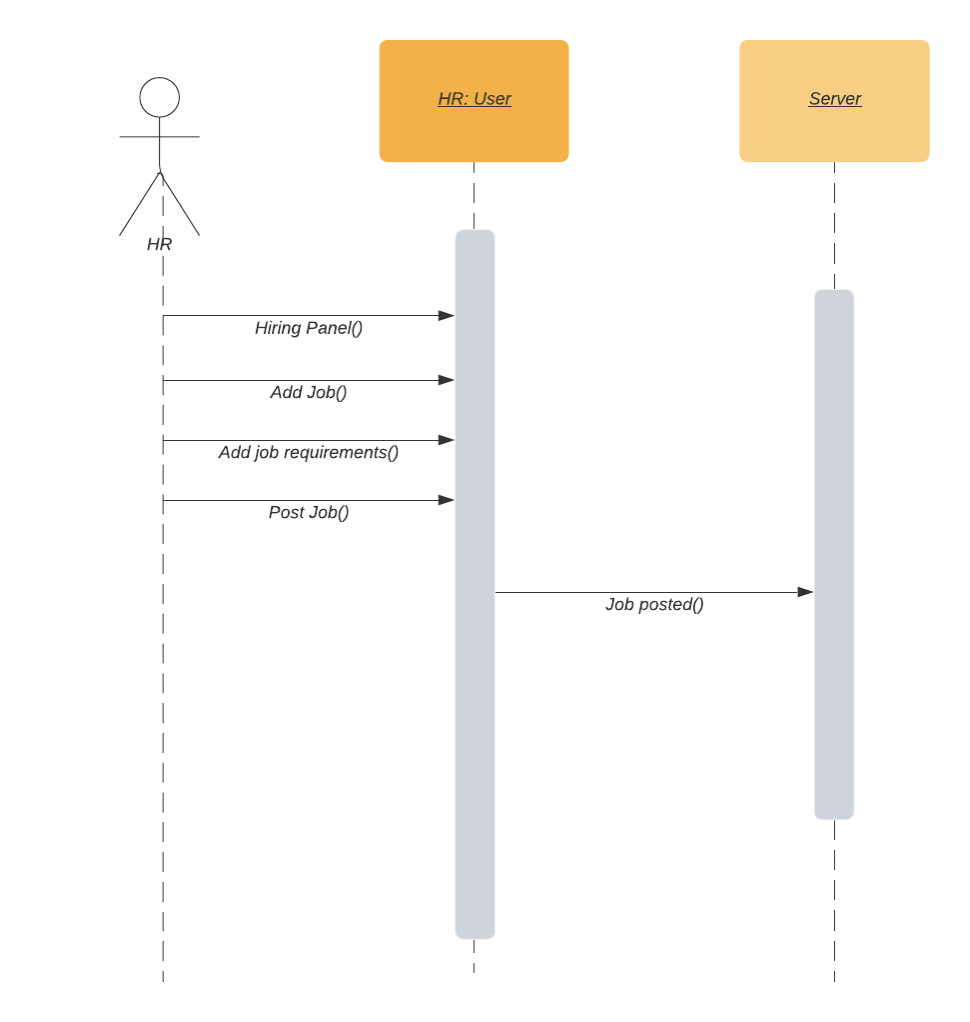
**4.4.8 Parsing CV’s**



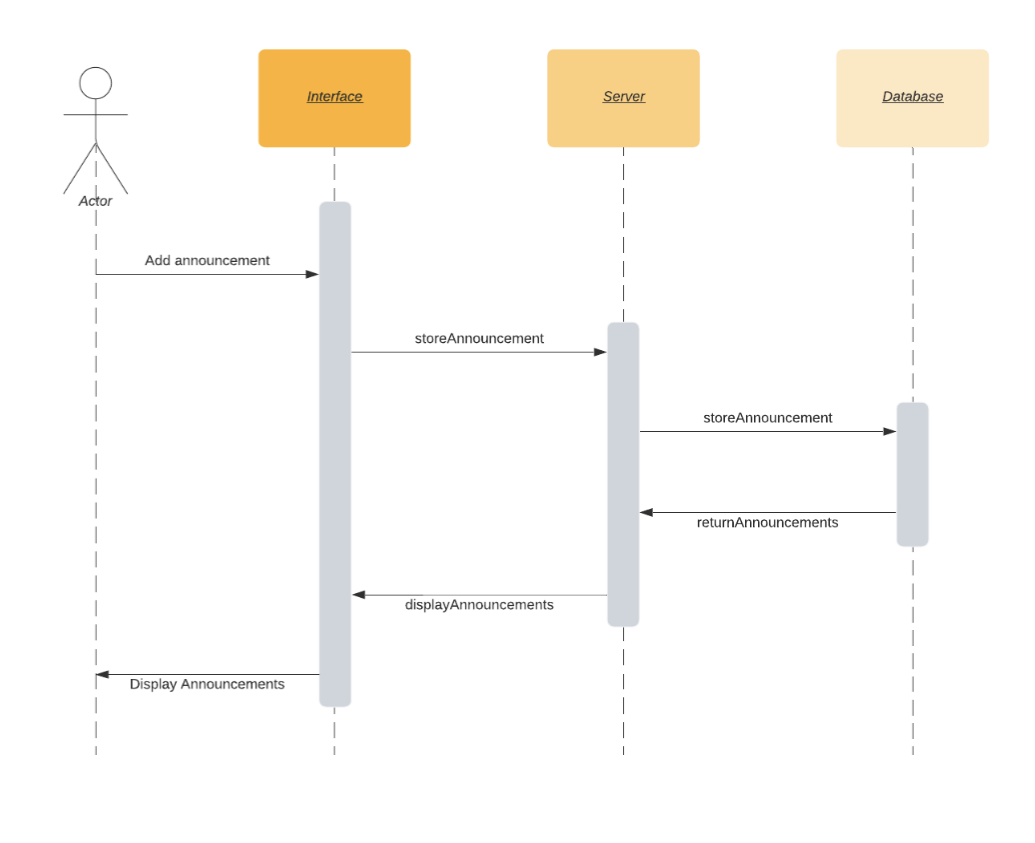
**4.4.9 Hierarchy Tree**



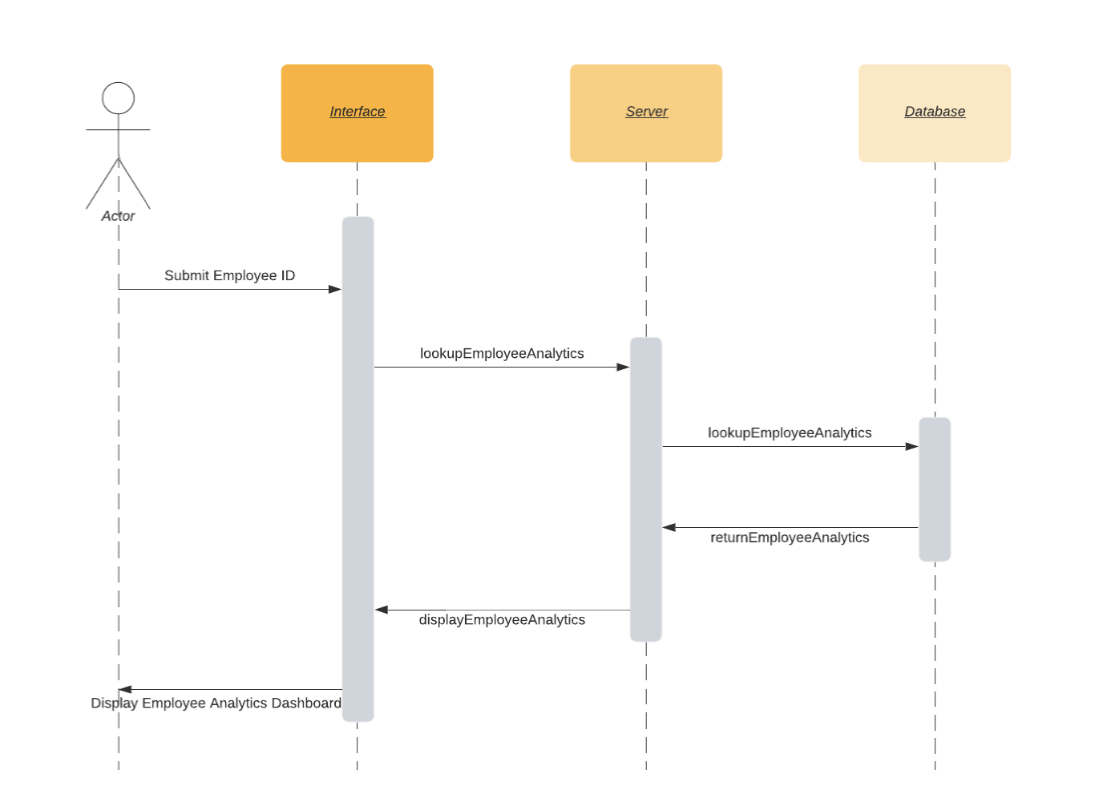
**4.4.10 Job Posting**

****

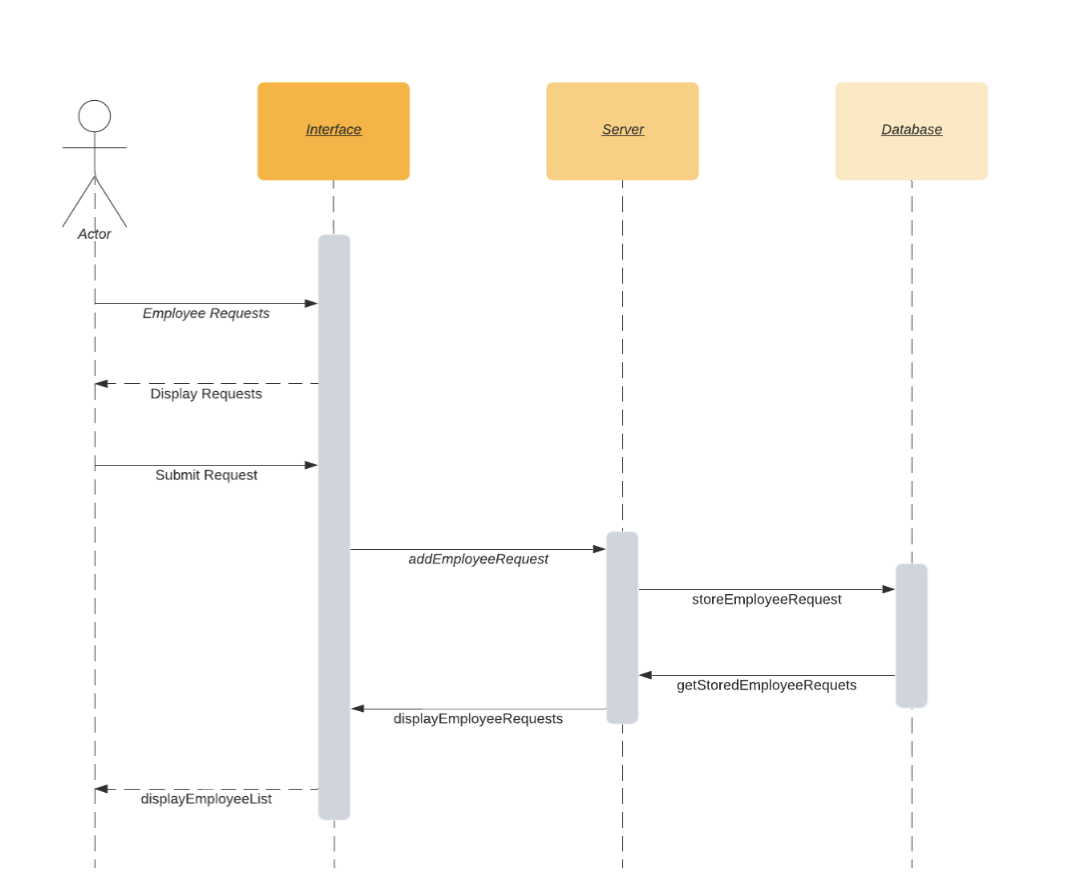
**4.4.11 Adding Announcements:**



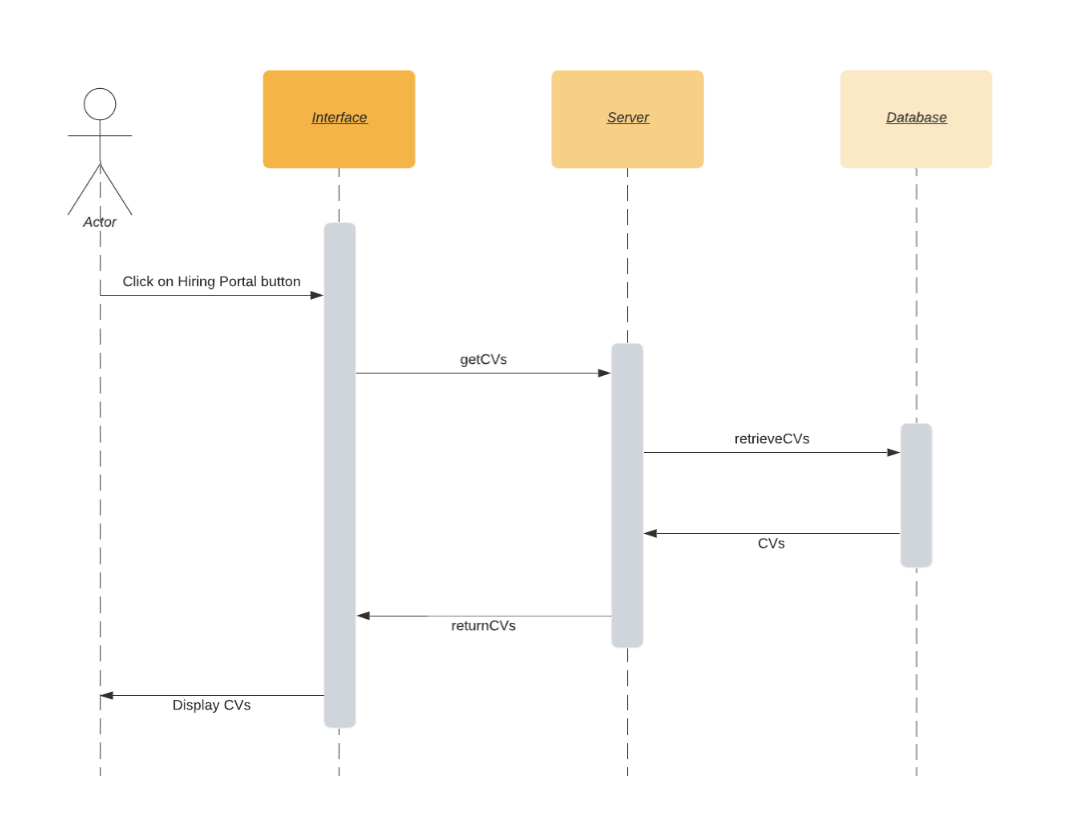
**4.4.12 Employee Data Analytics:**



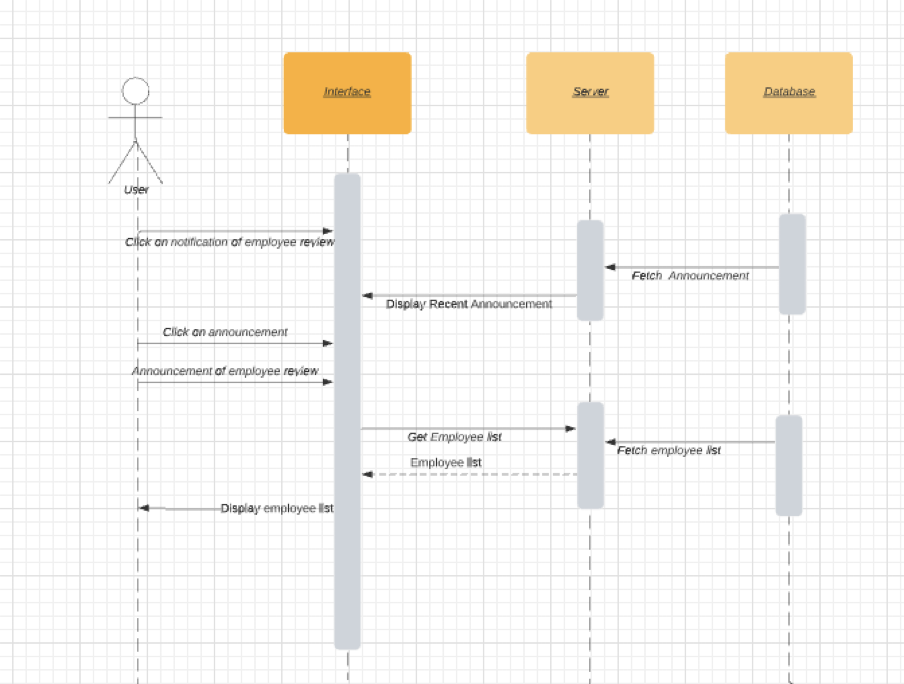
**4.4.13 Employee Requests:**

****

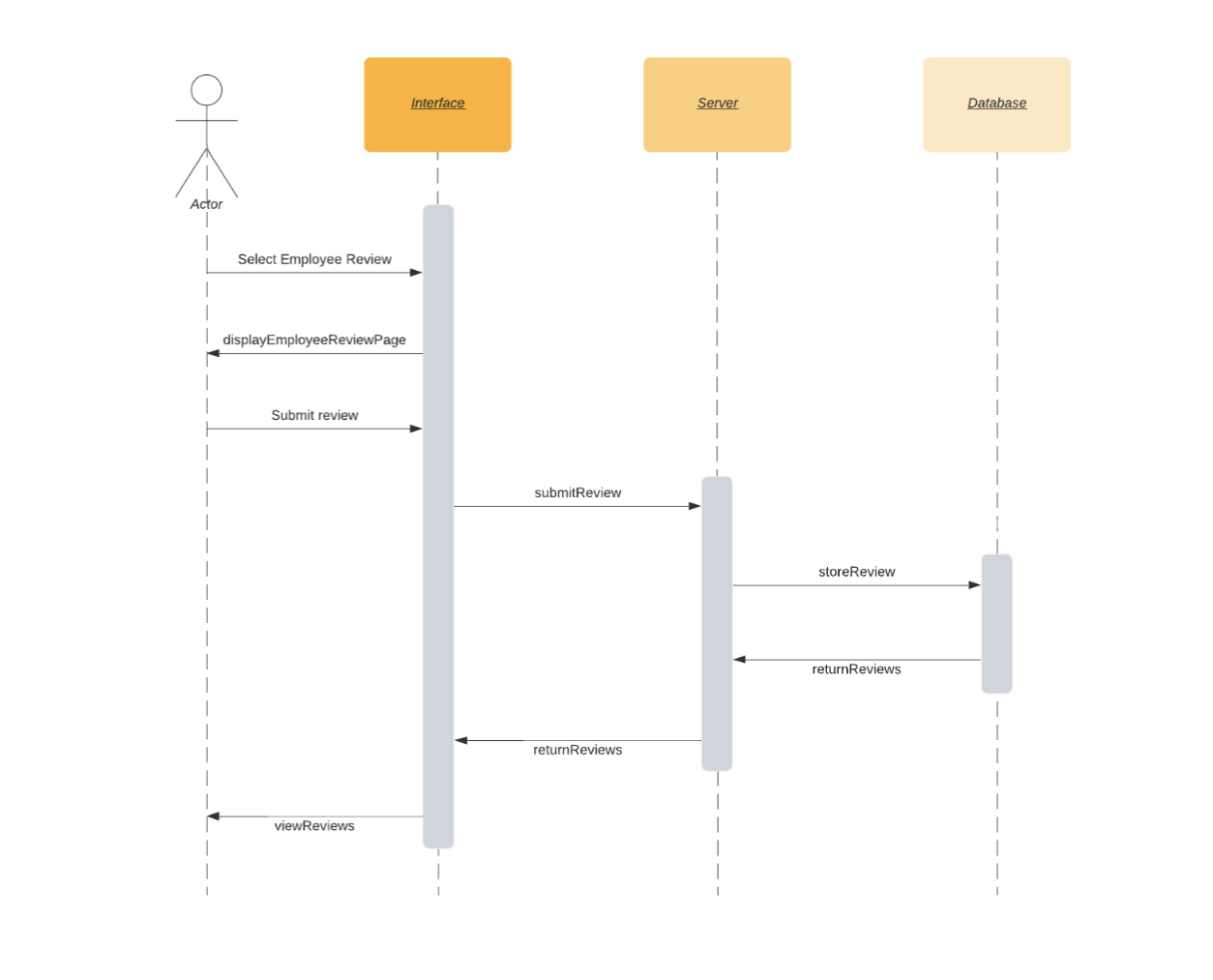
**4.4.14 Employee Review:**

****

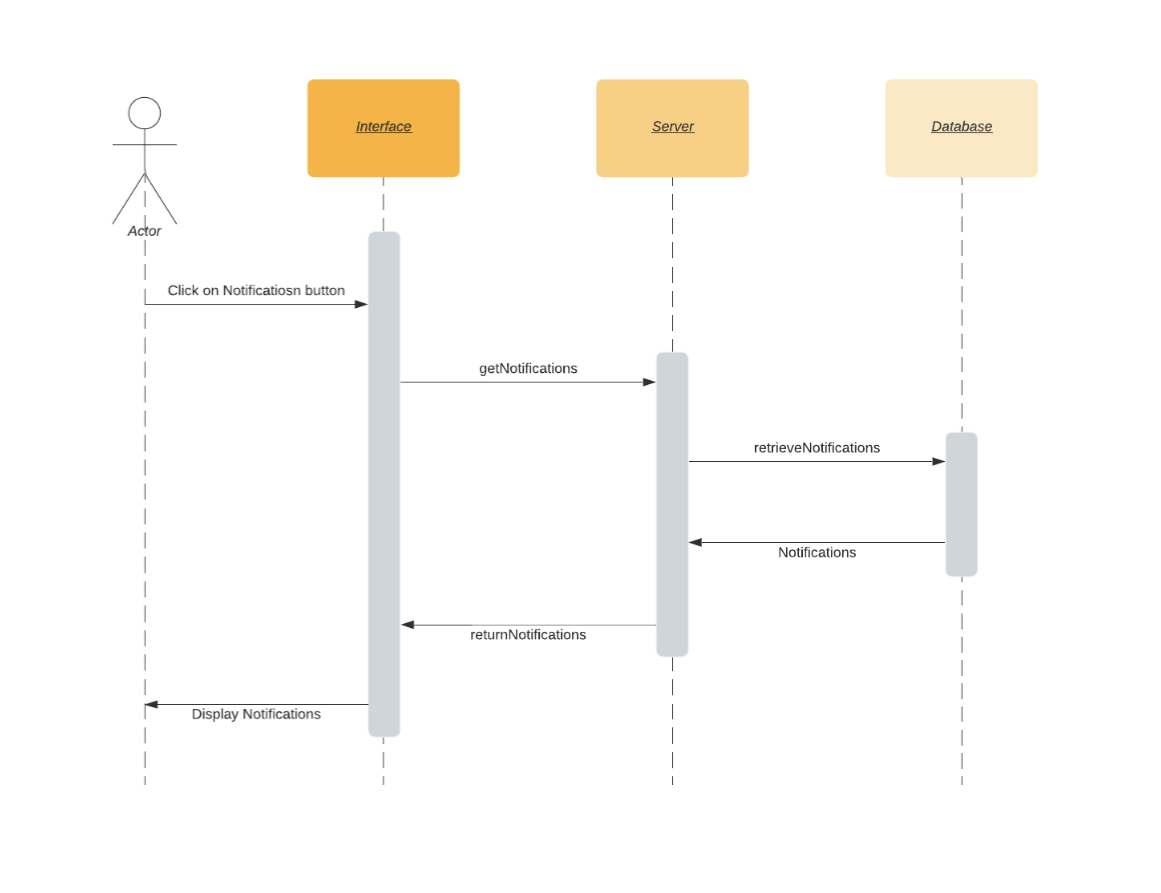
**4.4.15 Employee Reviews Reminder**

****

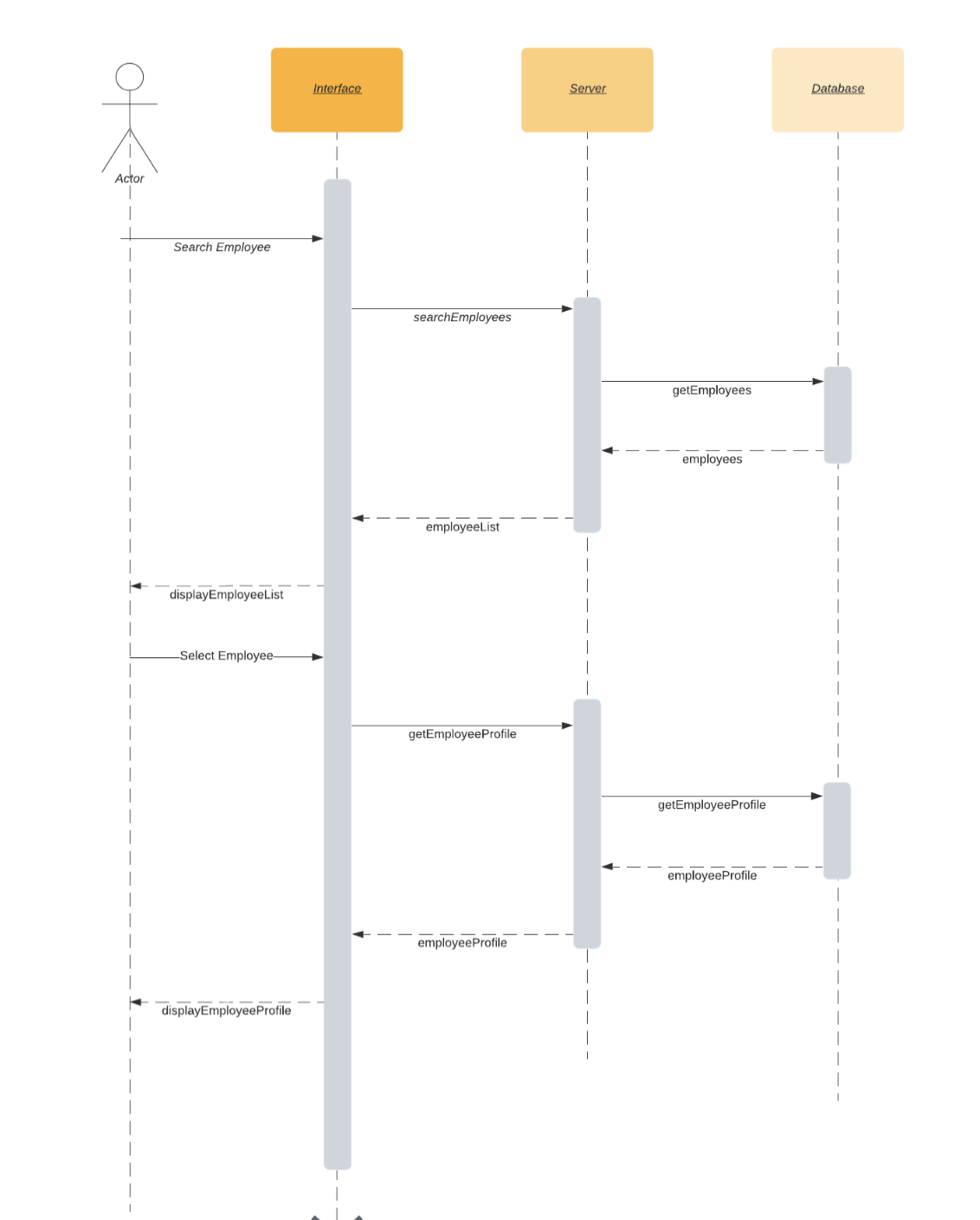
**4.4.16 Hiring Portal:**

****

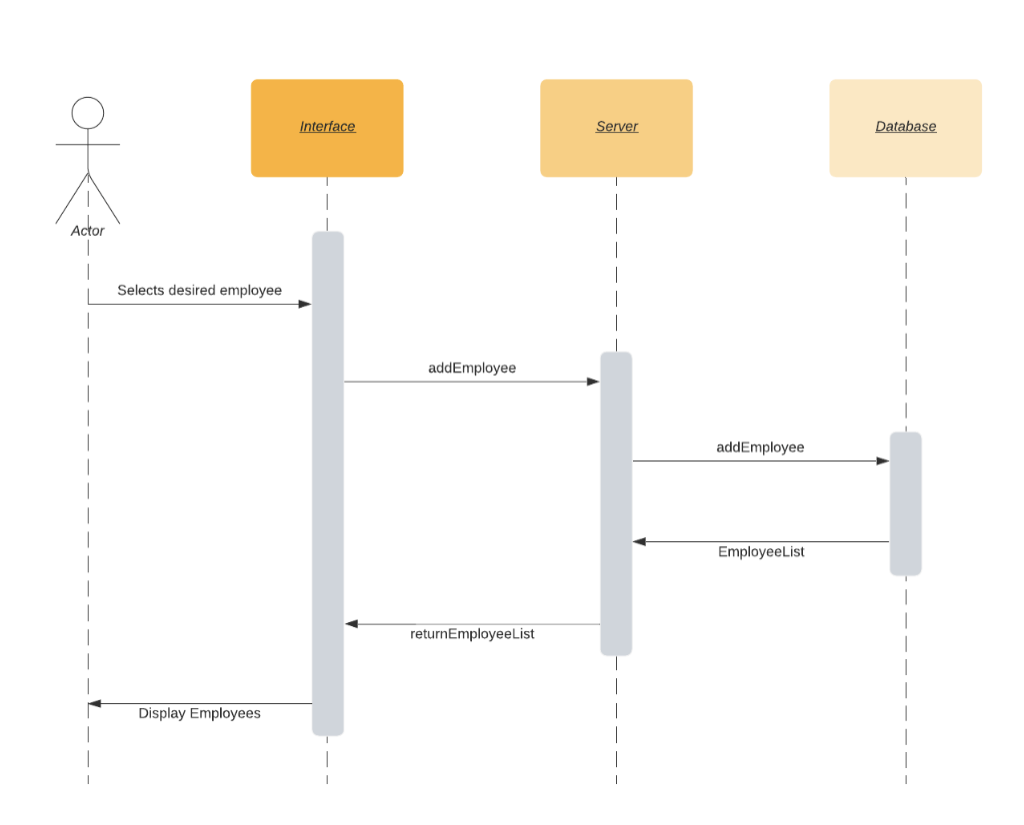
**4.4.17 Notifications:**



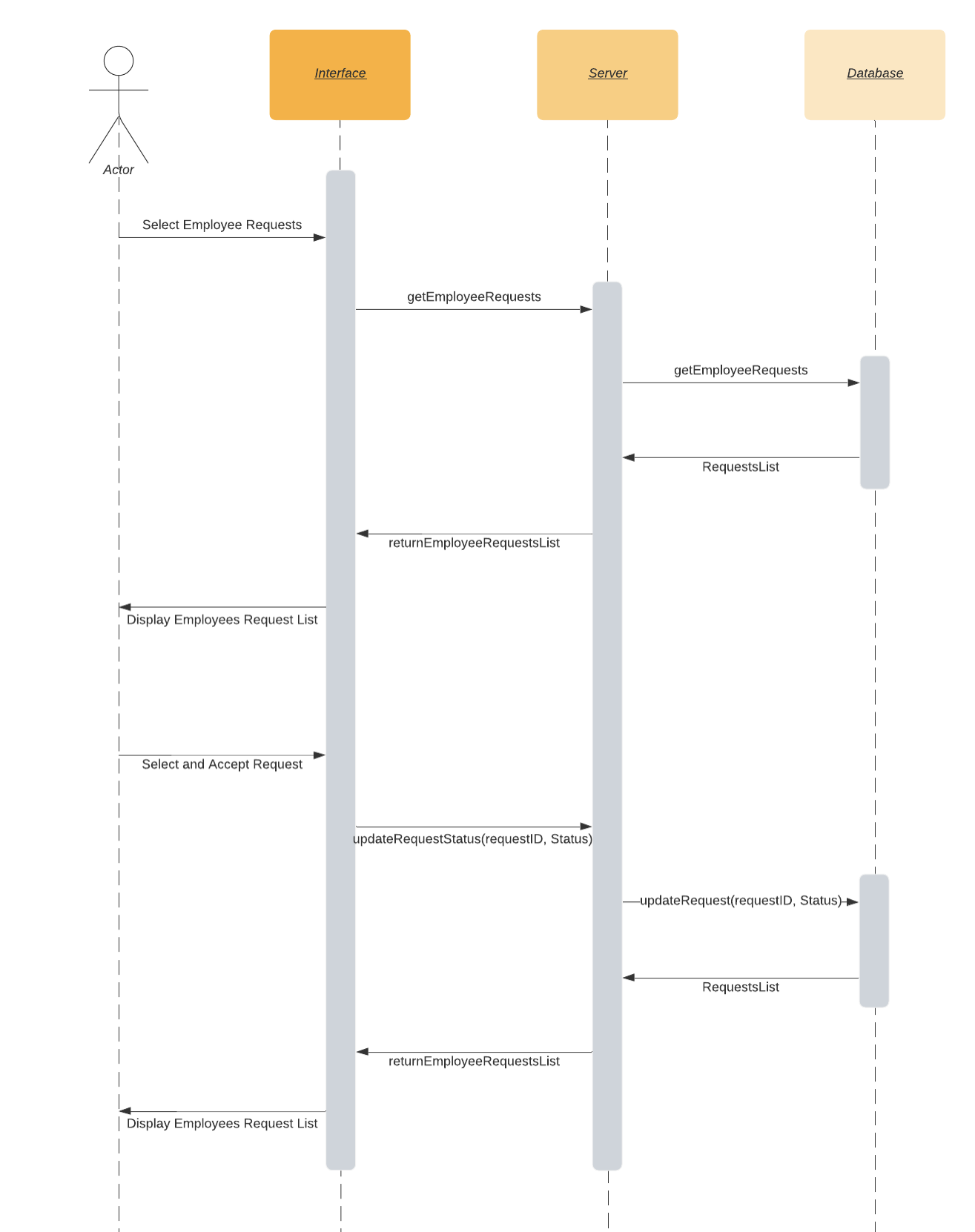
**4.4.18 Staff directory:**

****

**4.4.19 Hiring Candidates**

****

**4.4.20 Employee Request Accept/Denial**

****

# Software Development Methodology and Plan

This chapter discusses the two software processes we had in mind and how we decided on a process to use with complete justification of our choice.

## Software Process Selection

The Waterfall Model methodology is followed in a sequential order. The project development team only moves to the next phase of development after the previous step has been completed. The thorough upfront preparation of Waterfall yields precise project plans. Clients will know exactly what will be delivered because the project scope, cost, and deadline are all well-defined. With a clear framework, even if a team encounters turnover, a new member may step in and contribute without sidetracking the timetable. However in the Waterfall model deviation from the original plan is difficult with a strict layout, testing is completed at the end of the Waterfall project, and the final quality assurance process takes a long time. If a client's needs change while a Waterfall project is in progress, it cannot be handled.

Agile methodology encourages flexible, rapid progress using iterative development, delivering pieces of the project along the way to ensure customer needs are met. Frequent delivery allows the customer to provide constant feedback, resulting in a higher-quality product. The methodology breaks down large tasks to be completed in specific time frames. The development and testing activities are being done at the same time. The Agile methodology is known for its flexibility and Agile allows changes in project development requirements. Agile also performs testing concurrently with software development. Agile, on the other hand, does not comply with strict timeline, which might be tough to manage under pressure. Project requirements that change may generate issues in other sections of the company. Agile requires a dependable team. Wasted time and money could occur from a weak link in the Agile team or management.

**Our Selection:**

Our group will go with the agile model. The waterfall model is not ideal for projects where the requirements are not completely decided at the beginning. That is the case in our project since we will be updating the requirements in the later phases. Our project does not have strict regulatory requirements and there is a lot of room to make changes, this pushed us away from the Waterfall model.

**Justification of Selection:**

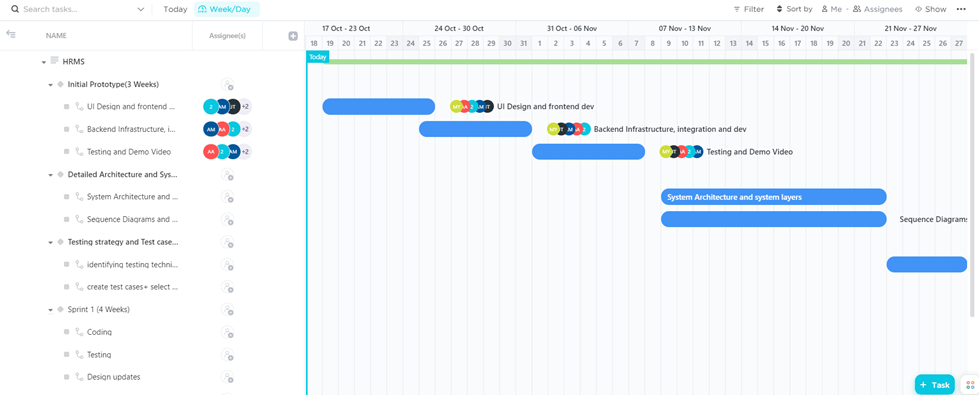
Since the water model has defined requirements, it leaves less room for creativity which our team would not prefer. Agile however, allows us to adjust requirements and priorities along the way. Our team is trying to implement some innovative features that do not fully exist in any form today so the agile model will be ideal for that.

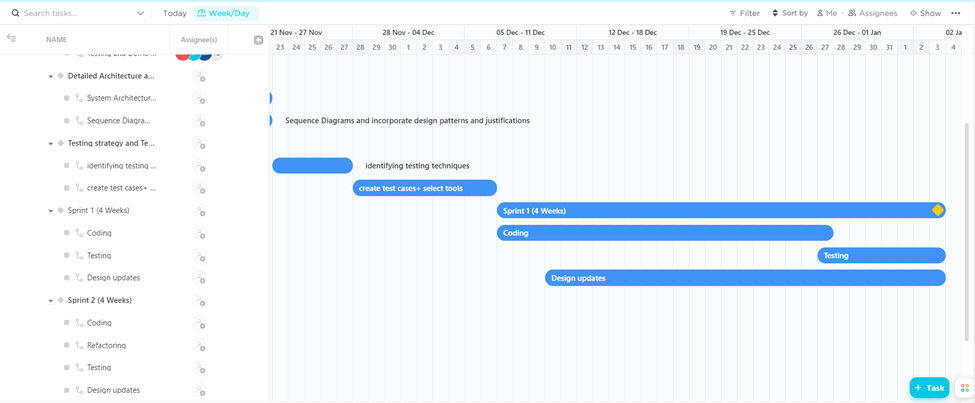
The major drawback will be the timeline. If the project timeline is fixed and can not be moved, the Waterfall model will offer a more predictable outcome. We however believe that this won’t be that much of an issue due to the great teamwork in our group.

**Project Context Analysis**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1  (Low) | 2 | 3 | 4 | 5  (High) |
| Potential loss due to defect/bugs |  |  | - |  |  |
| Developers’ experience/skills |  |  |  | - |  |
| Rate of requirements change |  |  |  | - |  |
| Team Size (5,10,25,50,100+) | - |  |  |  |  |
| Organization culture (adaptive to change) |  |  | - |  |  |
| Pressure to develop early releases |  |  | - |  |  |
| Business staff’s commitment to work extensively with development team |  |  |  |  | - |
| Developers experience with similar systems |  | - |  |  |  |
| Availability of reusable components |  |  |  | - |  |

## Gantt Chart



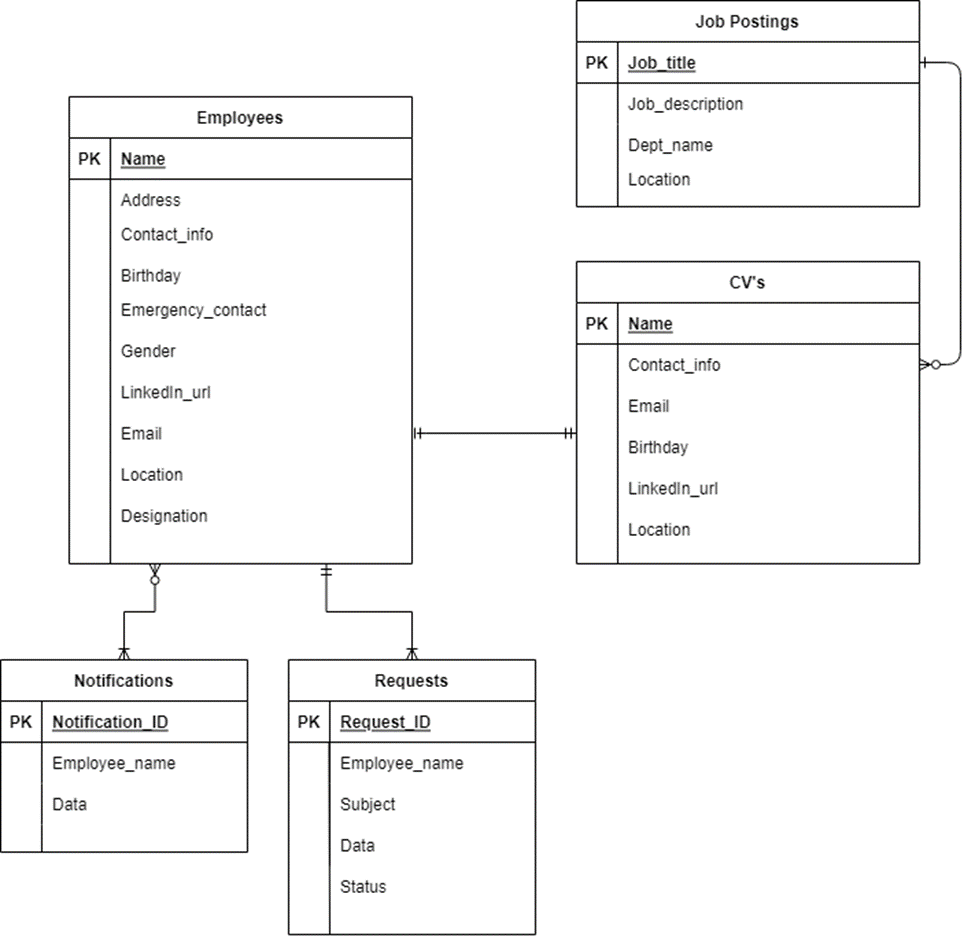


# Database Design and Web Services

This chapter describes our database in detail

1. **Database Design**

We have chosen a NoSQL database, DynamoDB.



All the three tables shown in this ER Diagram are part of AWS Dynamodb. The Employees table is part of the HRMS node server. The Job postings and CV’s tables are part of the Job Application node server.

The Employees table will have complete information about the current employees of the company. All the attributes will be stored as strings.

The Job postings table will store the information about all the active job postings.

The CV’s Table will contain the data of the application form such as the applicant’s name, their contact information, their location and their LinkedIn profile url. The CV documents will be stored in amazon s3 storage.

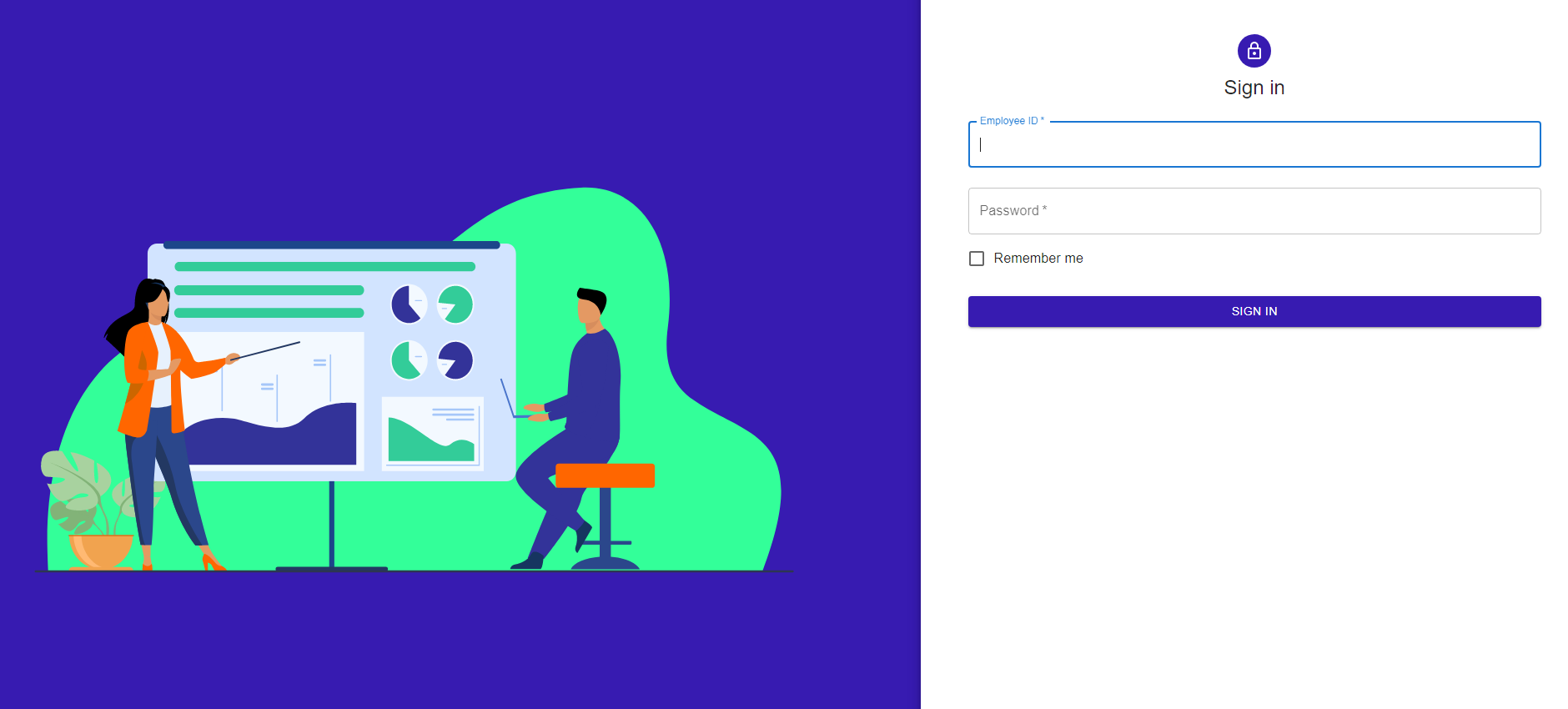
## API Specification

We are not using any external API’s. All of the API’s being used are made by us.

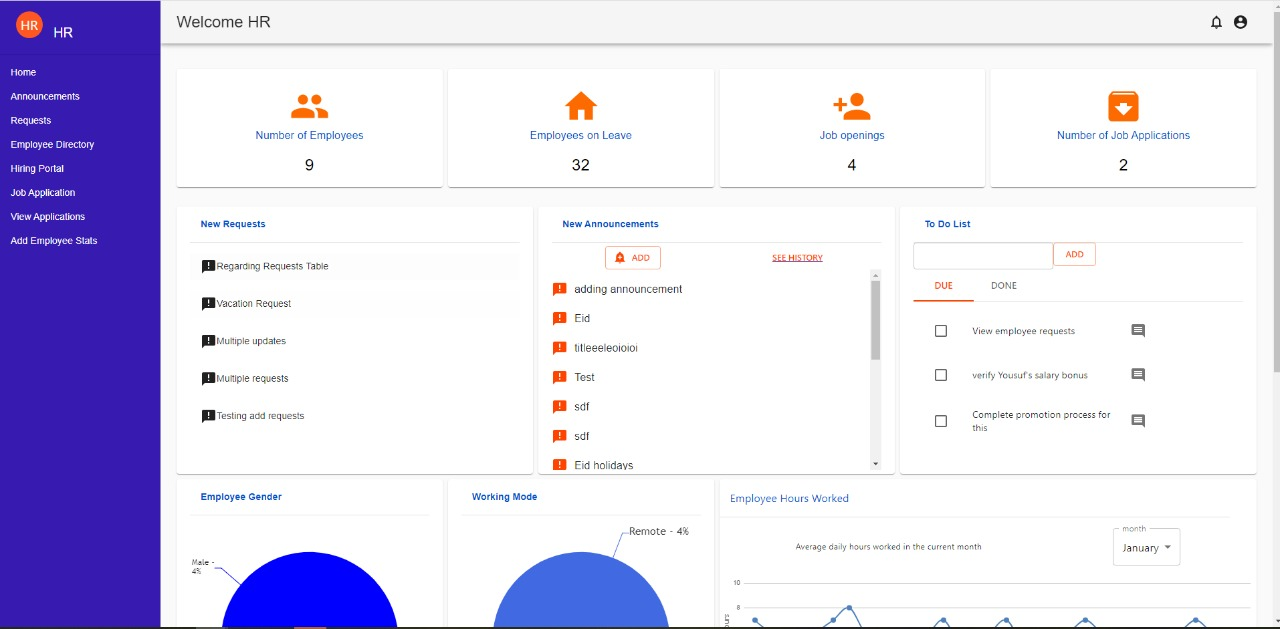
# System User Interface

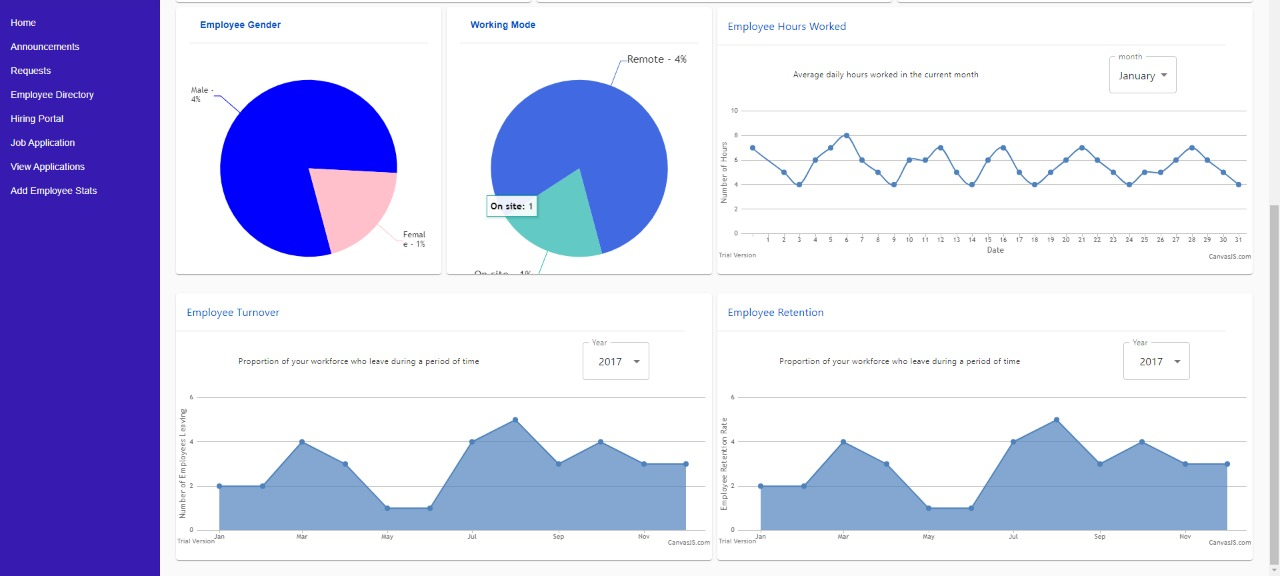
Our application consists of 3 parts: an HR portal, Employee and a jobs portal for candidates.

Login screen:



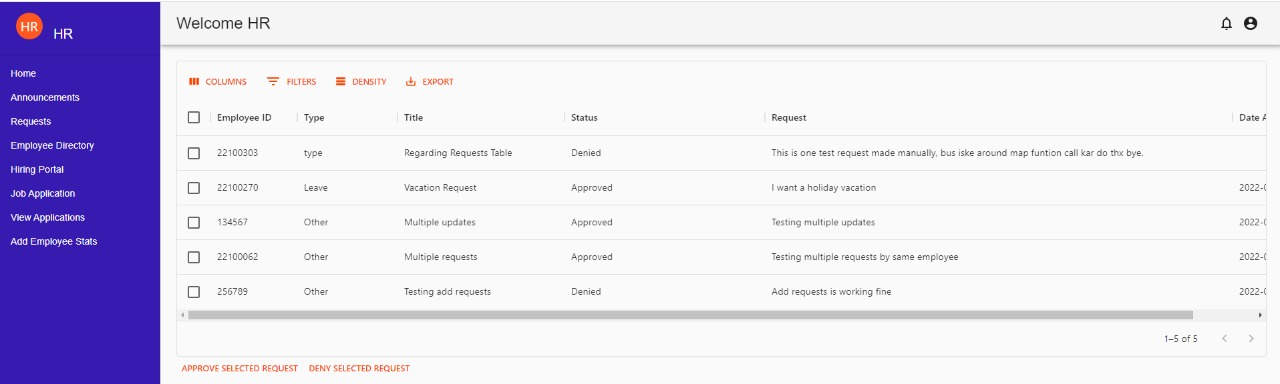
HR portal when user logs in as HR:





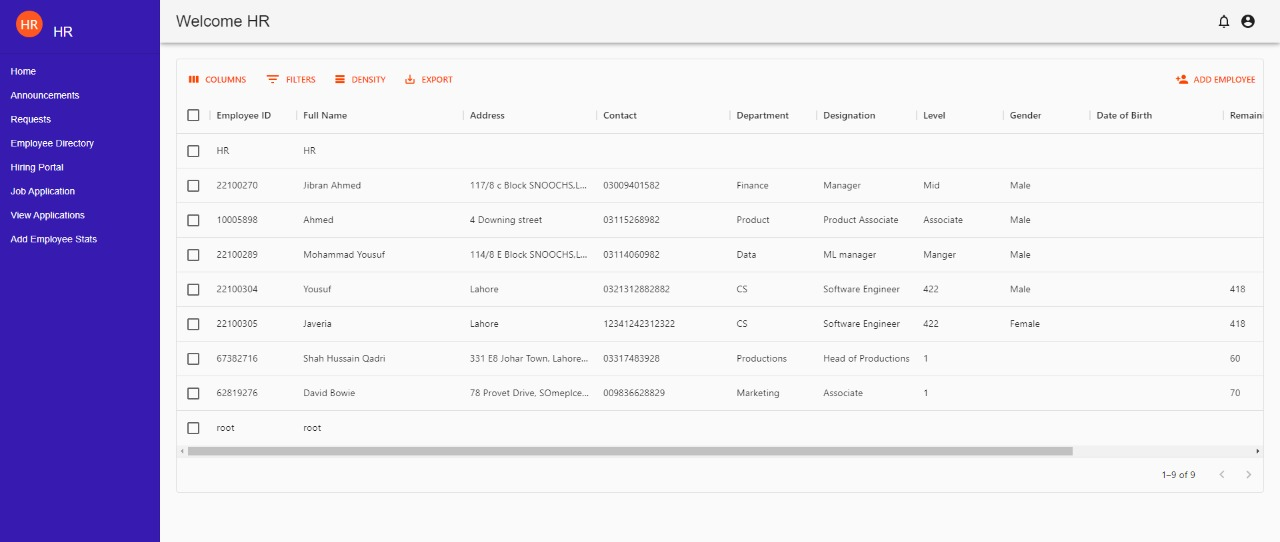
This allows HR to monitor real-time analytics, view requests, add announcements,requests, job postings,job applications and number of employees. It also has a navigation bar and a header bar to logout and change passwords.

Employee Requests:

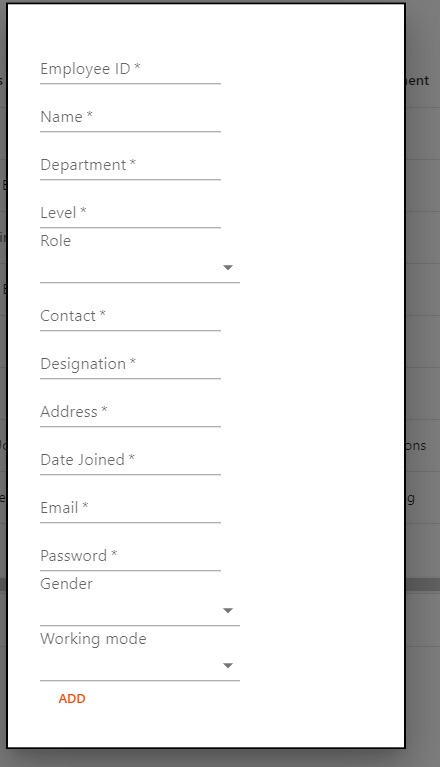


A list containing all the requests by employees with an option to accept or deny these requests by the HR.

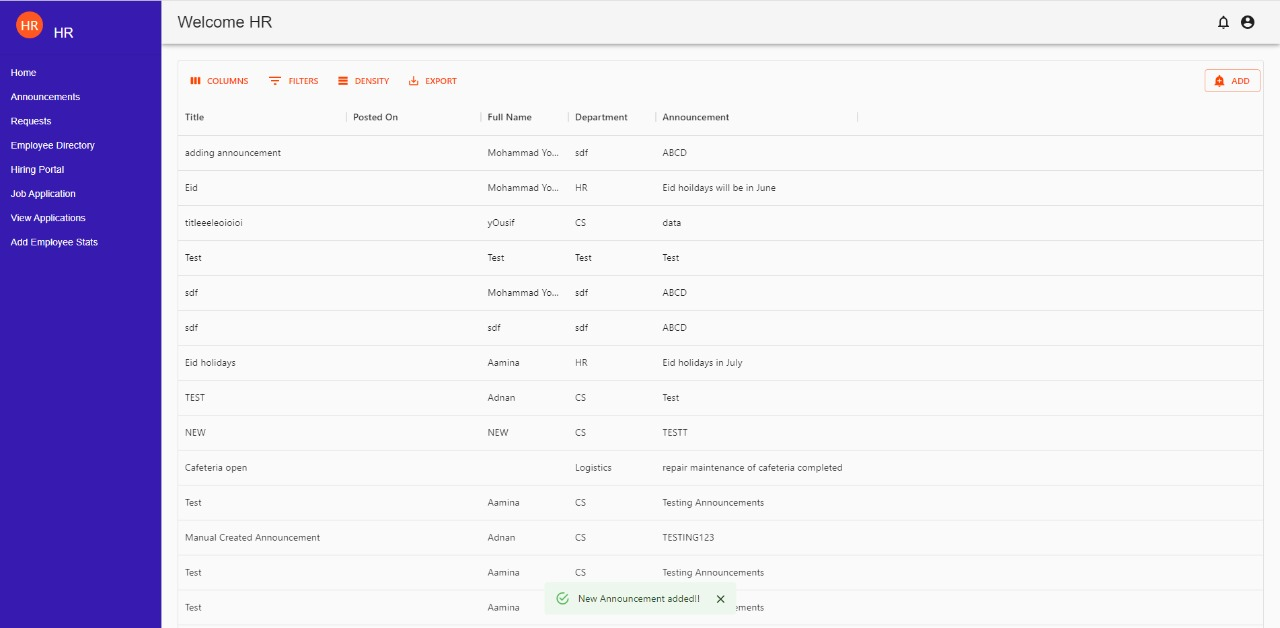
Employee directory:



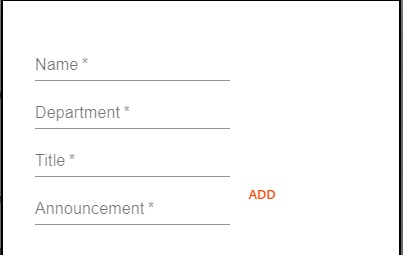
A list containing all the data of employees eg name and date of birth etc with a button to add employees and their details.



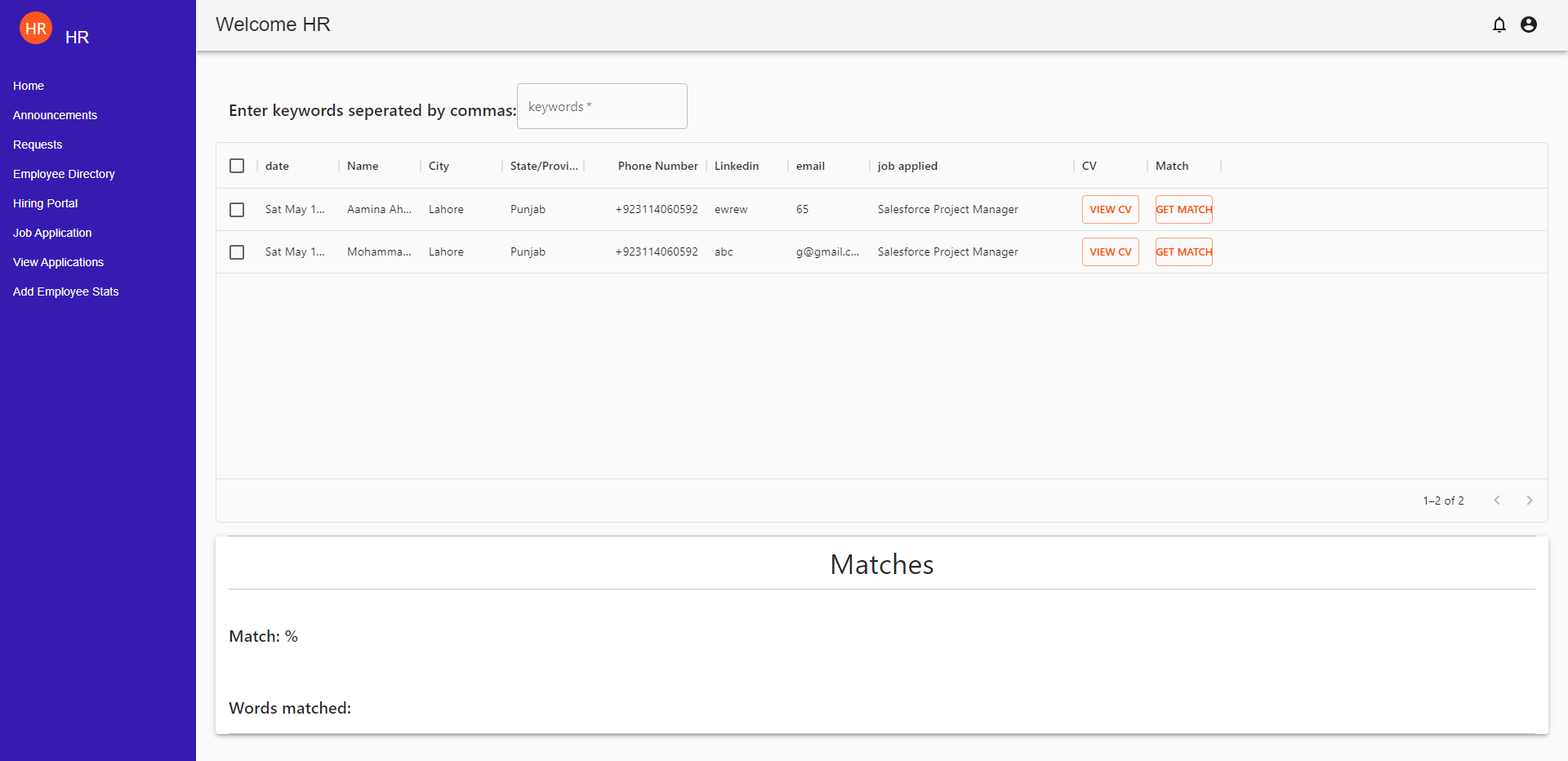
Announcements:



Announcements page with the details of the announcements along with options to add announcements as well.

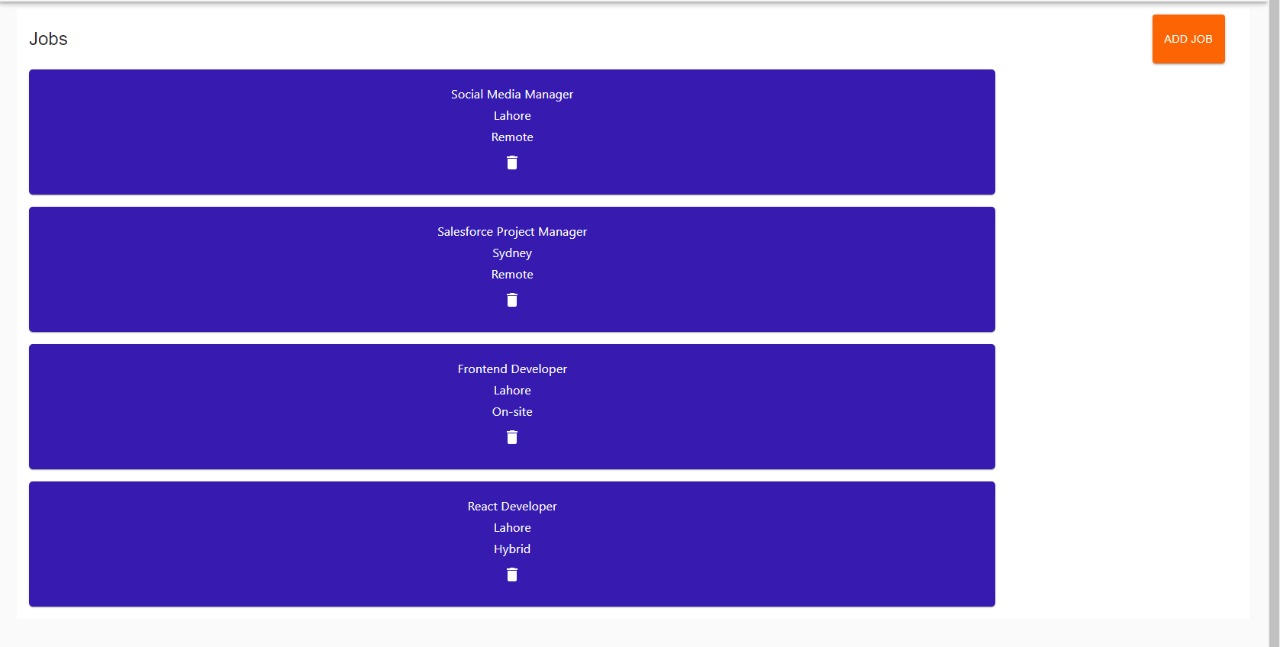


View Applications:

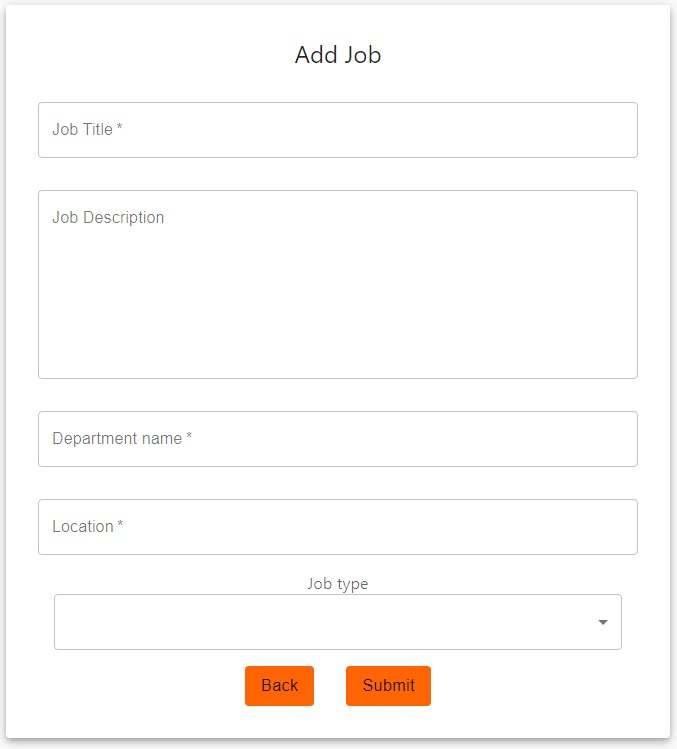


A view application page where the HR can view applications for jobs. The HR also has the option to view the CV by clicking view CV. The HR can also scan a candidate’s CV with words like Javascript, python etc without looking at the CV by adding keywords and clicking get match. This feature can save time for HR/hiring managers as they will not have to read through each and every CV thus saving time.

Hiring Portal:

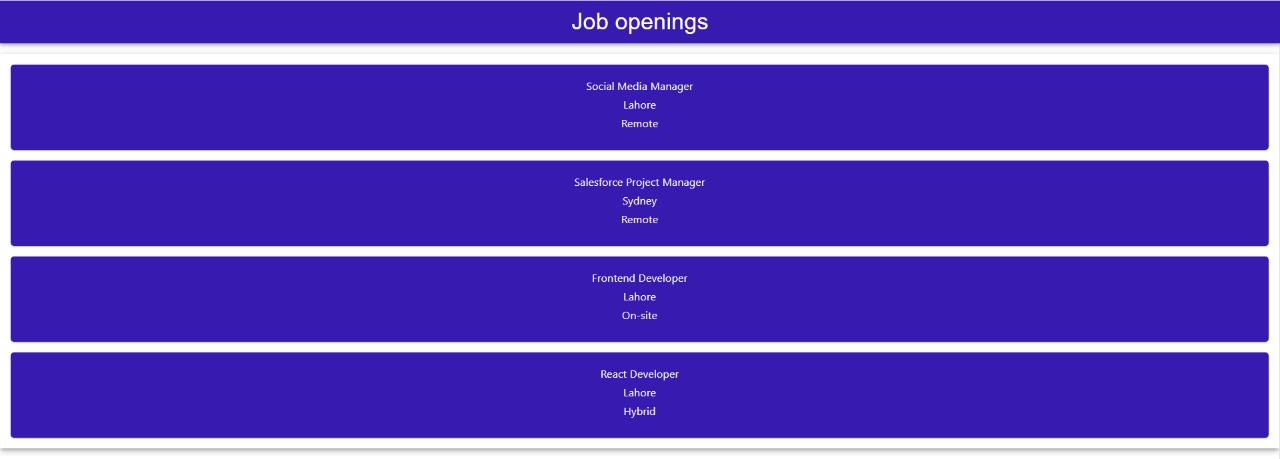


A hiring portal where all jobs posted can be viewed by the HR. The HR can also add jobs by clicking the add job button or delete jobs by clicking the trash icon.

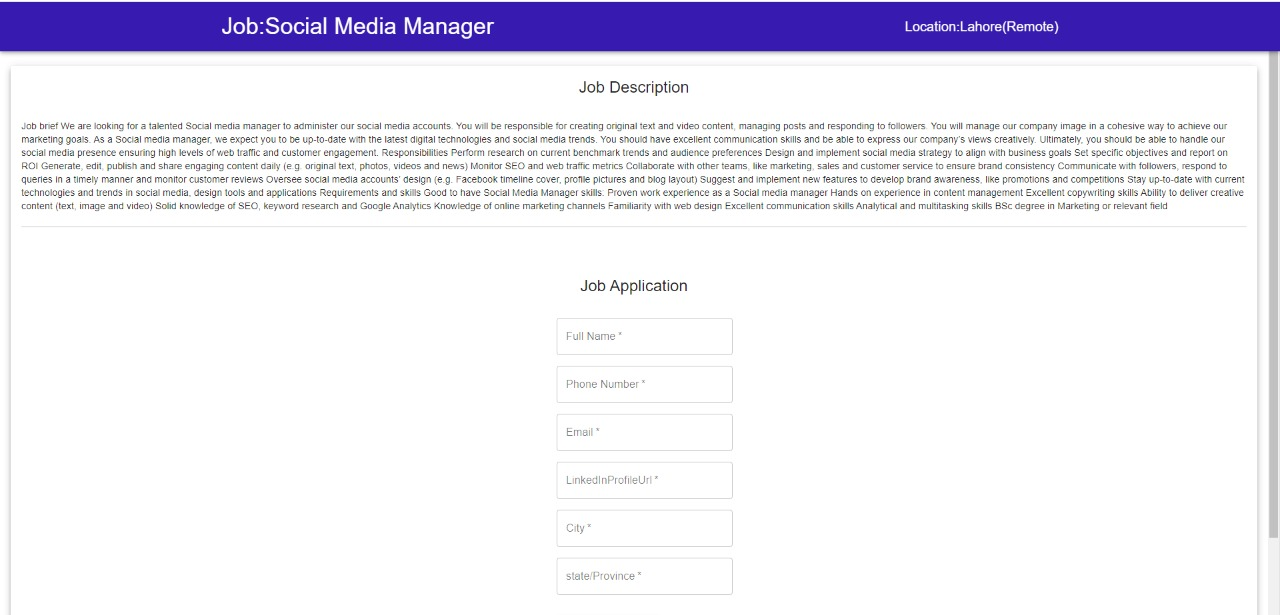


Add a job portal where a job can be added along with details of the job.

Jobs portal:

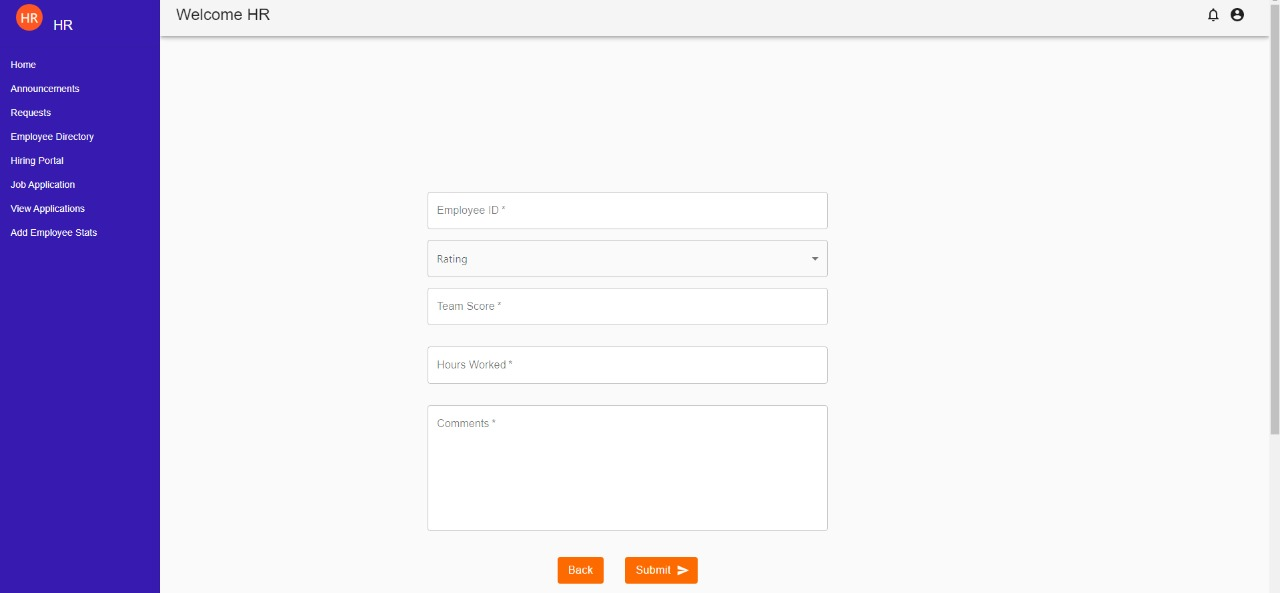


A jobs portal where all the jobs can be seen by candidates. A candidate can apply for a job by clicking on a job posting.



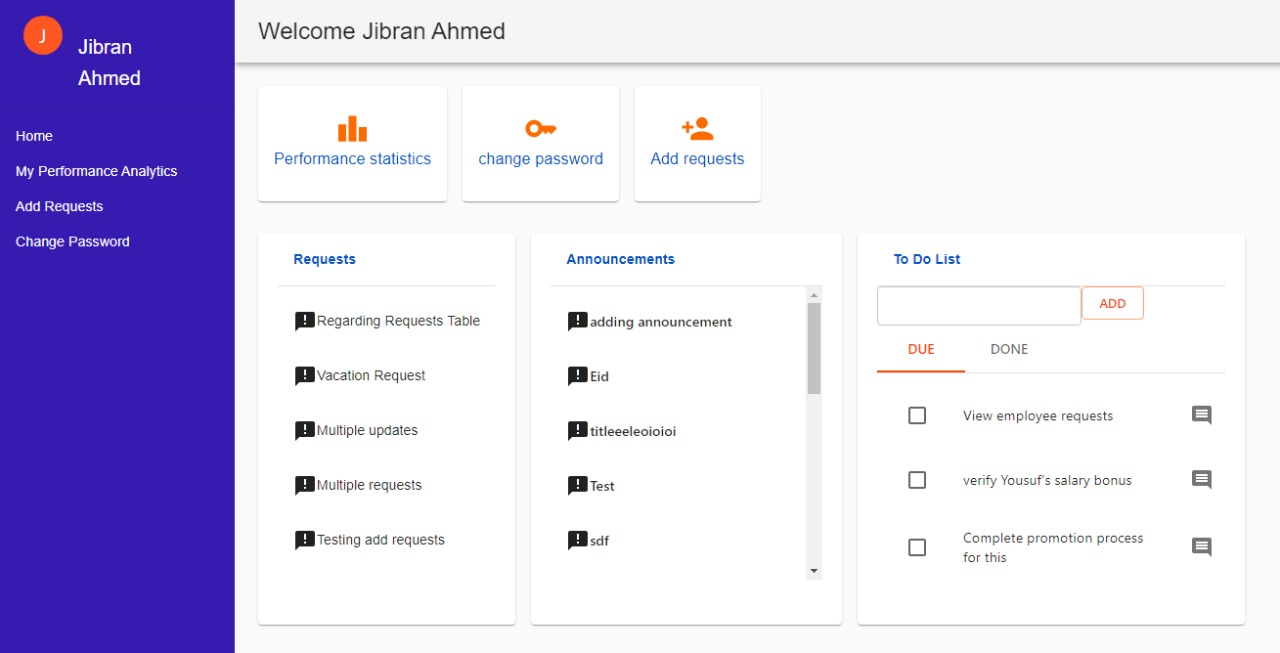
The candidate will be redirected to a form with the job application details and description. The candidate can then enter their details and submit the form.

Add stats form:



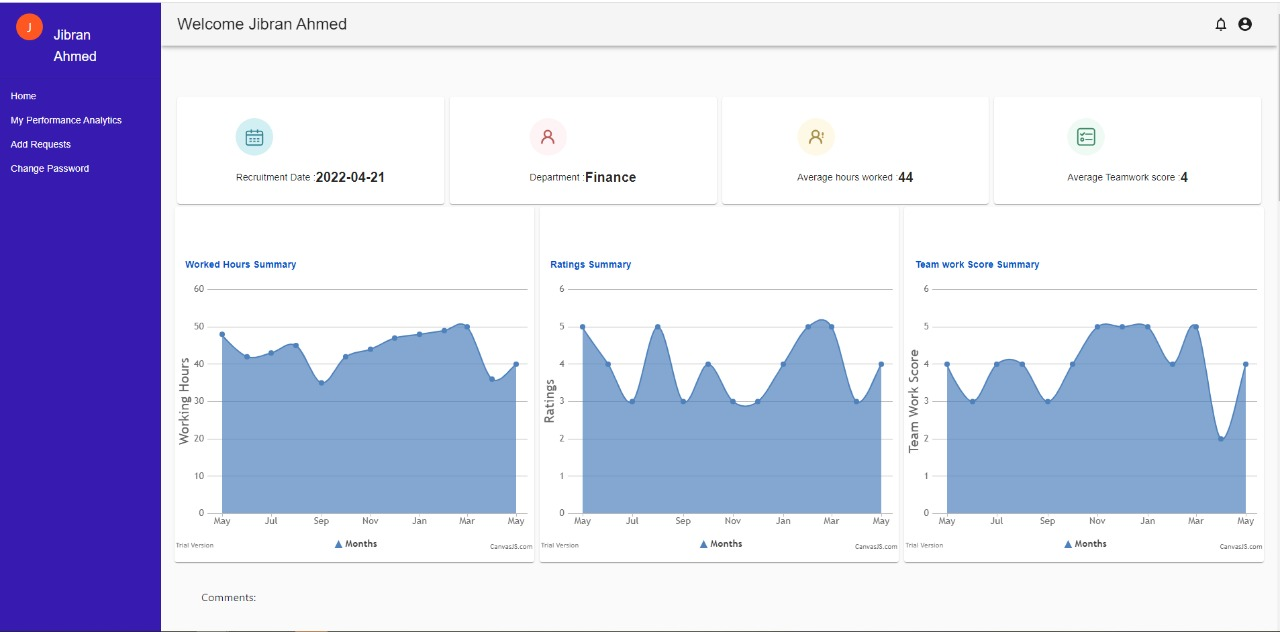
A form to add stats where an HR manager can add stats and ratings for an employee and they will show up on the employee’s analytics dashboard.

Employee Portal:



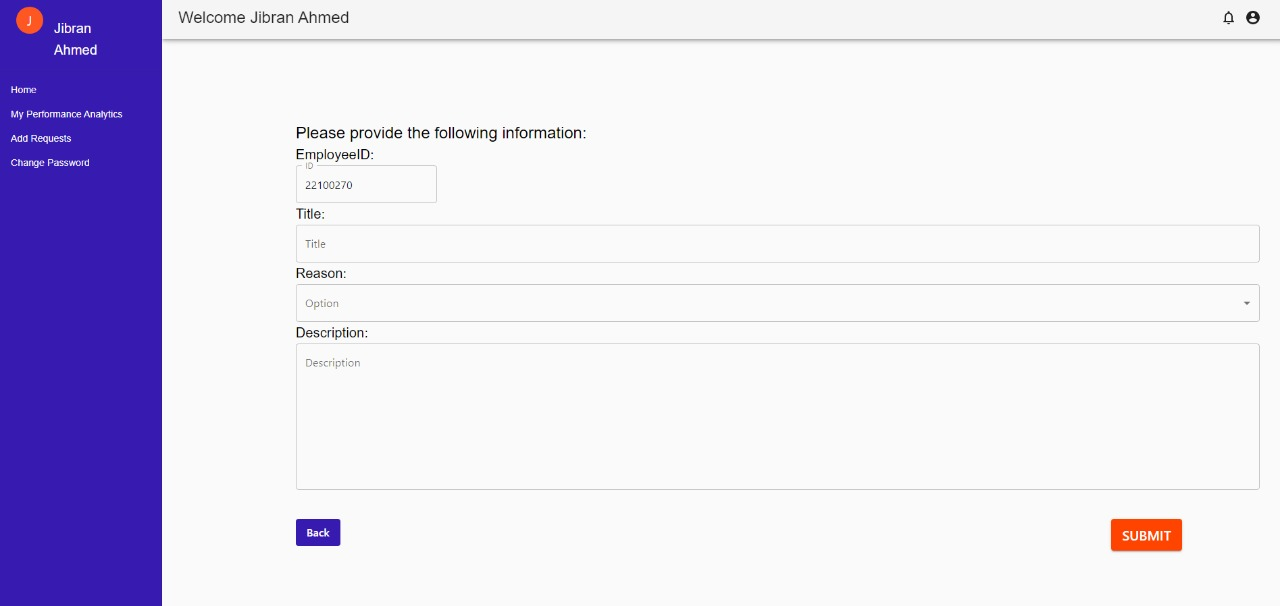
The employee can view announcements, requests, make a to do list and view performance stats added by the HR.

Employee analytics:



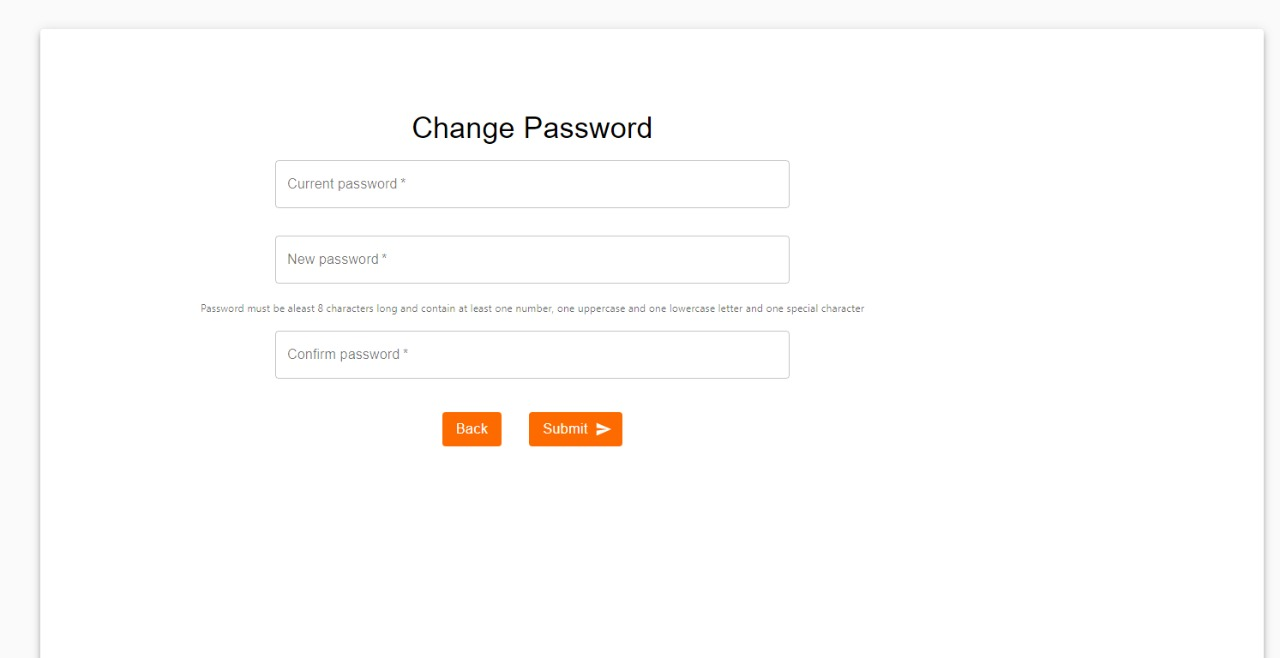
an employee analytics page where an employee can view their teamwork scores, hours worked and comments by the HR.

Employee Requests:



A requests portal where an employee can add like maternity leave and vacations leave etc. These requests are then sent to the HR for approval.

Change password:



A change password page where an employee can change their password.

# Project Security

In this we are discussing the top 3 security risks and their controls for our system. Also, the potential losses for our system if we do not implement the controls. Plus, we will also be discussing some testing tools suitable for our system.

## Project Threats

**1. Broken Access Control**

1. Violation of the principle of least privilege or deny by default
2. Accessing API with missing access controls for POST, PUT and DELETE.
3. Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user.

**2. Cryptographic failures**

All employee passwords are encrypted and then stored. If the cryptographic algorithms are weak or old it can lead to attackers gaining access to our system.

**3. Identification and Authentication Failures**

The attacker has a list of valid usernames and passwords they can access our system without raising any red flags

## Potential Losses

**1. Broken Access Control**

1. An attacker might be able to change or delete content
2. Perform unauthorized functions
3. Attackers can take over site administration.

**2. Cryptographic failures**

As with broken access control the attacker can change or delete information or perform unauthorized actions

**3. Identification and Authentication Failures**

If the attacker can access our system without raising any red flags and have access to all of the company information it can be dangerous for the security and privacy of employees.

## Security Controls

For Broken Access Control we can:

1. Minimize Cross-Origin Resource Sharing (CORS) usage.
2. Rate limit API and controller access to minimize the harm from automated attack tooling.

For Cryptographic Failures we can:

1. Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as scrypt or bcrypt
2. Always use authenticated encryption instead of just encryption.

For Identification and Authentication failures we can:

1. Implement multi-factor authentication
2. Enforce a password policy to prevent users from setting weak passwords
3. Use session tokens

All of these are protective controls

## Static and Dynamic Security Scanning Tools

For Static Testing we chose Coverity because it is open source and it is compatible with JavaScript and NodeJs, 2 of the major technologies used in our system. Its fast and accurate incremental analysis runs in the background to minimize disruption. It can comprehensively track and manage compliance through a wide range of security, quality, data protection, and safety standards.

And for Dynamic testing we chose APIsec. It automatically creates and runs thousands of test cases tailored to your unique API architecture - integrated directly into your workflow. Unlike traditional security solutions that look for common security issues, APIsec pressure-tests the entire API, including hidden business logic flaws responsible for most breaches.

# Risk Management

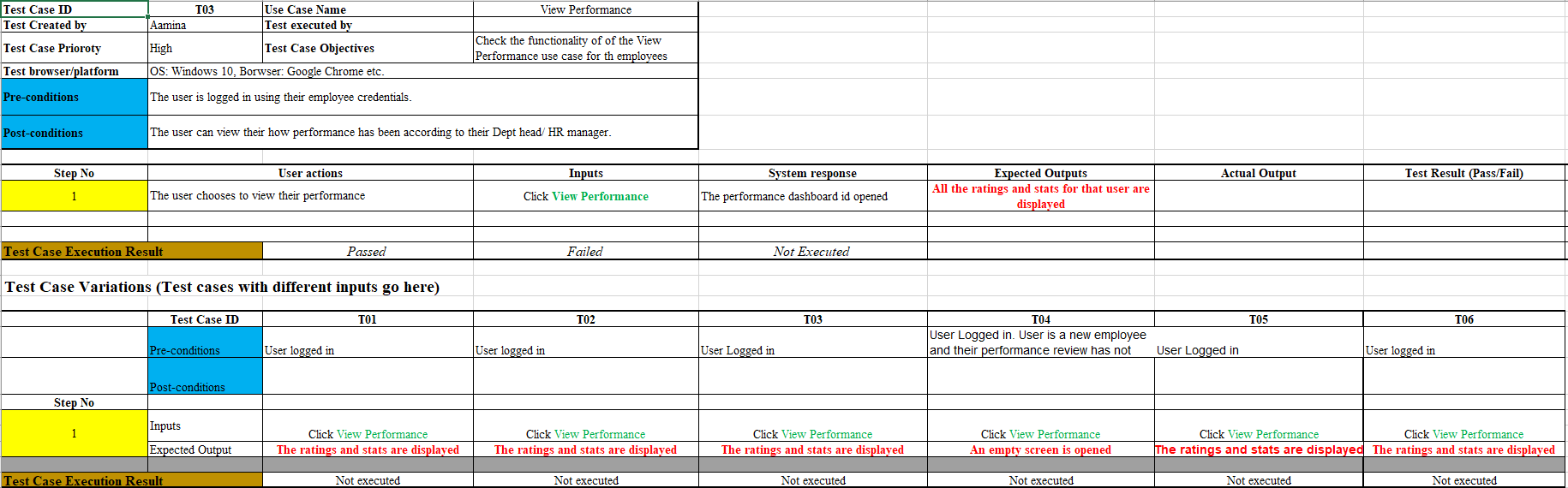
## Potential Risks and Mitigation Strategies

|  |  |  |
| --- | --- | --- |
| **Sr.** | **Risk Description** | **Mitigation Strategy** |
| 1. | Server failure | There are two ways to fix this issue: to store data in all the accessible databases or distribute it evenly among them |
| 2. | Broken Authentication | Configure multi-factor authentication whenever possible. The more hoops an attacker has to jump, the harder it is to get into your system. Also paying attention to session management and setting application timeouts correctly. When a user closes a browser, log them out of the system. Any time a user leaves a session while still logged in, the entire system remains vulnerable. |
| 3. | Sensitive data exposure | Sensitive data exposure can be prevented by  ● Encrypting data in transit and at rest.  ● Using the latest encryption algorithms.  ● Disabling auto-complete on forms that collect data.  ● Disabling caching on forms that collect data. |
| 4. | Broken Access Control | It is important to control who has access to which portion of the website. Access control is only effective if it is enforced in trusted server-side code, where the attacker cannot modify the access control authentication.  ● Deny access to standard functions.  ● Use access control lists and role-based authentication mechanisms.  ● Don't just hide features. |
| 5. | Security Misconfiguration | Security misconfiguration can be mitigated by not allowing default configuration in our website. It can be prevented by  ● Disabling administration interfaces  ● Disabling use of default accounts/passwords.  ● Configuring the server to prevent unauthorized access, directory listing, etc.  ● Considering periodically performing scans and audits to help detect future configuration errors or missing fixes. |
| 6. | Cross Site Scripting | In general, effective prevention of cross site scripting vulnerabilities can involve a combination of the following measures:  ● **Sanitize User Input**  1. Validate to detect potentially malicious input from users.  2. Encrypt the output to prevent potentially malicious user-supplied data from triggering the autoload and execute behavior by a browser  ● **Limit use of user provided data**  1. Only use when it’s necessary  2. Utilize content security policy  By providing additional layers of protection against cross site scripting attacks. |
| 7. | Insufficient logging and monitoring | Depending on the risk of the data being stored or processed by the application:  ● Ensure that all server-side login, access control, and access validation errors can be logged with sufficient user context to identify suspicious or malicious accounts and for a sufficient period of time to allow for deferred forensic analysis.  ● Ensure that high-value transactions have an audit trail with integrity checks to prevent tampering or deletion, such as add-only database tables and the like.  ● Provide effective monitoring and alerts to detect and address suspicious activity in a timely manner. |
| 8. | Insecure Direct Object References | Perform proper and consistent user authorization and whitelist the choices. More often than not, however, the whole problem can be avoided by storing the data internally. |
| 9. | Unvalidated Redirects and Forwards | This can be prevented by  ● Avoiding redirects  ● Providing a static list of valid locations to redirect to. |
| 10. | Unrestricted File Upload | ● Never accept a filename and its extension directly without an allow list filter.  ● The application should filter and check content for all files uploaded to the server. Files must be carefully analyzed and validated before being made available to other users. When in doubt, the file should be discarded.  ● It is necessary to have a list of the only allowed extensions on the web application. And the file extension can be selected from the list. |

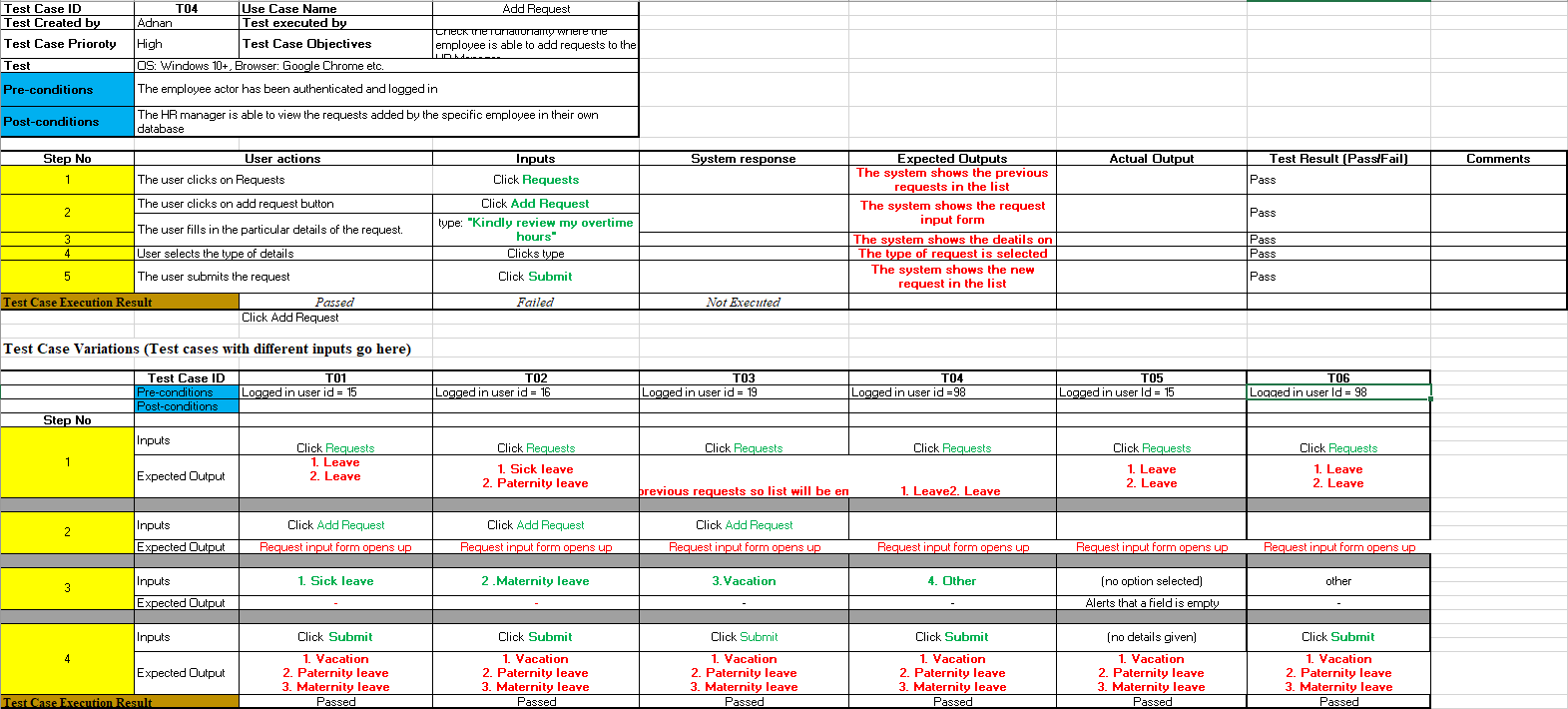
# Testing and Evaluation

Throughout the course of the project, we have manually tested our system. We mostly used Postman to test our API’s. Two of the test cases that we created are mentioned below. Since our application was very heavy and our machines could not handle the load of automated testing, we did not carry out any automated tests.

**View Performance:**



**Add Request:**



# Deployment Guidelines

Prerequisites:

* Node js should be installed on PC
* env file
* pdf services api credentials
* git

Deployment:

1. open terminal and type: git clone <https://github.com/aaminamariam/P05.git>
2. then type: git checkout nevdev
3. Navigate to P05>P05-HR Management System>Development>Sprint-1
4. on the terminal type: npm i
5. then type:npm run devStart
6. open another terminal or on a separate server and open the same folder and type : npm run server2
7. on another terminal Navigate to P05>P05-HR Management System>Prototype
8. then type: npm i
9. then type: npm run build
10. then type: serve -s build (port number)

**We were unable to deploy since our free trial ended on AWS**

# Conclusion

## Summary

This project aims to simplify the complex HR process of recruitment and managing employees for HR. It simplifies recruitment by making it easier for HR to filter employees. It also makes it easier for HR to manage current employees by giving HR real-time statistics and details of each employee.

## Challenges

1. Parsing CVs and filtering them was a huge challenge since it was difficult to make an automated system for parsing and filtering through pdfs.
2. Working with multiple servers was also a challenge as it was difficult to figure out the correct routing mechanism for each of them
3. Analytics and graphs were tricky to manage since they required a lot of data and made the application heavier.
4. The UX was also complex to understand as HR use-cases for each organization vary.

## Future

The system is built in a modular format so it can be edited and added on easily without major changes needed to the databases or the system.

# Review checklist

Before submission of this report, the team must perform an internal review. Each team member will review one or more sections of the deliverable.

|  |  |
| --- | --- |
| **Chapter/Section Name** | **Reviewer Name(s)** |
| System Requirements, Requirement Specifications | Ali Adnan Arif |
| Introduction, System User Interface, Deployment Guidelines, Conclusion | Mohammad Yousuf |
| Introduction, Software Development Methodology and Plan, Risk Management | Javeria Tariq |
| Introduction, System Architecture, Database Design and Web services, Project Security, Testing and Evaluation | Aamina Mariam |

# References