

---

# **PROJECT REPORT**

## **HR Management System**

Anish Thakkar  
CE082  
22CEUON124

Vaibhav Mackwana  
CE024  
22CEUBG138

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Overview . . . . .	5
1.2	Project Objectives . . . . .	5
1.3	System Overview . . . . .	5
1.4	Definitions, Acronyms, and Abbreviations . . . . .	6
1.5	Link . . . . .	6
<b>2</b>	<b>Requirement Specification and Analysis</b>	<b>7</b>
2.1	Functional Requirements . . . . .	7
2.1.1	Employee Management . . . . .	7
2.1.2	Payroll Management . . . . .	7
2.1.3	Attendance Management . . . . .	7
2.1.4	Role and Authorization Management . . . . .	7
2.1.5	Department Management . . . . .	7
2.2	Non-Functional Requirements . . . . .	8
2.2.1	Usage . . . . .	8
2.2.2	Performance . . . . .	8
2.2.3	Scalability . . . . .	8
<b>3</b>	<b>Design</b>	<b>9</b>
3.1	ER Diagram . . . . .	9
3.2	Class Diagram . . . . .	11
3.2.1	Attendance . . . . .	13
3.2.2	Payroll . . . . .	14
<b>4</b>	<b>Implementation details</b>	<b>15</b>
4.1	Technologies Used . . . . .	15
4.1.1	Frontend Technologies . . . . .	15
4.1.2	Backend Technologies . . . . .	15
4.1.3	Database . . . . .	15
4.1.4	Development Environment . . . . .	15
4.2	Screenshots . . . . .	15
<b>5</b>	<b>Testing</b>	<b>25</b>
5.1	Overview . . . . .	25
5.2	API Testing . . . . .	25
5.3	Frontend Testing . . . . .	25
5.4	Database Testing . . . . .	25

<b>6 Conclusion and Future Extensions</b>	<b>32</b>
6.1 Conclusion . . . . .	32
6.2 Future Extensions . . . . .	32
6.2.1 AI Chatbot for Employee Report Analysis . . . . .	32
6.2.2 Further Optimize the payroll calculations to include concurrency .	32

# List of Figures

Figure 3.1 Entity Relationship Diagram - 1 . . . . .	9
Figure 3.2 Entity Relationship Diagram - 2 . . . . .	10
Figure 3.3 Class Diagram - 1 . . . . .	11
Figure 3.4 Class Diagram - 2 . . . . .	12
Figure 3.5 Sequence Diagram - Attendance . . . . .	13
Figure 3.6 Sequence Diagram - Payroll . . . . .	14
Figure 4.1 login page . . . . .	16
Figure 4.2 navigation bar . . . . .	17
Figure 4.3 employee list . . . . .	18
Figure 4.4 departments . . . . .	18
Figure 4.5 holidays . . . . .	18
Figure 4.6 attendance sheet format . . . . .	19
Figure 4.7 bulk attendance addition form . . . . .	20
Figure 4.8 attendance list . . . . .	20
Figure 4.9 code for excel attendance addition . . . . .	21
Figure 4.10 salary report . . . . .	22
Figure 4.11 stored procedure code . . . . .	22
Figure 4.12 settings . . . . .	23
Figure 4.13 users . . . . .	23
Figure 4.14 roles . . . . .	24
Figure 4.15 azure online hosting . . . . .	24
Figure 5.1 database payroll client analysis using ssms query . . . . .	26
Figure 5.2 database payroll execution time - 19ms . . . . .	26
Figure 5.3 Backend Code Maintainability Score . . . . .	27
Figure 5.4 Frontend Lighthouse Performance score for initial page . . . . .	28
Figure 5.5 Frontend Lighthouse overall score for initial page . . . . .	28
Figure 5.6 swagger-1 . . . . .	29
Figure 5.7 Swagger-2 . . . . .	29
Figure 5.8 Swagger-3 . . . . .	30
Figure 5.9 Swagger-4 . . . . .	30
Figure 5.10 Swagger-5 . . . . .	31
Figure 5.11 Swagger-6 . . . . .	31

# 1 Introduction

## 1.1 Overview

This document details the requirements and specifications of the HR Management System, including its functional and non-functional requirements, constraints, and system design. The system aims to enhance productivity by automating repetitive HR tasks and providing a centralized platform for managing employee-related data.

## 1.2 Project Objectives

The primary objectives of the HR Management System are:

- Centralized employee information with secure access
- Automated attendance tracking via Excel
- Optimized payroll using stored procedures
- Role-specific functionalities
- Intuitive interfaces
- Use scalable and performance centric architecture

## 1.3 System Overview

The HR Management System is built using modern technologies including **.NET 8**, **Web API** for the backend, **Angular** for the frontend, and **Microsoft SQL Server** for database management. The system architecture follows **industry best practices** with a focus on performance optimization.

We have followed a layered architecture with organized folders, including **Controllers**, **DTOs**, **Repositories**, and **Models**. The concept of **Seeders** is used for faster development. Separate APIs for **pagination-based fetching** and a filter middleware for **authentication** are implemented.

**Stored procedures** and the **ExcelDataReader** NuGet package were major innovations in this project. The stored procedures significantly reduced processing time compared to traditional ORM-based approaches, while ExcelDataReader enabled the

bulk addition of employee attendance efficiently.

The system is deployed on **Azure**, with both backend, frontend and database deployed on azure for enhanced accessibility and reliability.

## 1.4 Definitions, Acronyms, and Abbreviations

- **HR:** Human Resources
- **employee:** Our product name
- **.NET Core:** A cross-platform framework for building modern, cloud-based applications
- **CRUD:** Create, Read, Update, Delete

## 1.5 Link

- Hosted Website: [hrms-client-eyejb3adcth0f7ae.centralindia-01.azurewebsites.net](https://hrms-client-eyejb3adcth0f7ae.centralindia-01.azurewebsites.net)
- Hosted APIs: [hrms-app-service-g0frh0djd3bug6gh.centralindia-01.azurewebsites.net/swagger](https://hrms-app-service-g0frh0djd3bug6gh.centralindia-01.azurewebsites.net/swagger)
- Github Repo: <https://github.com/Vaibhav31mak/HRMS>

# **2 Requirement Specification and Analysis**

## **2.1 Functional Requirements**

Based on our research these are the feature requirements a company's HR may have:

### **2.1.1 Employee Management**

- Complete CRUD (Create, Read, Update, Delete) operations for employee data
- Role-based access controls

### **2.1.2 Payroll Management**

- Bulk salary updates for efficiency
- Automated payroll calculations considering weekdays, overtime, and days off
- Integration with attendance data for accurate salary processing

### **2.1.3 Attendance Management**

- Excel-based attendance data upload mechanism
- Real-time integration with payroll systems

### **2.1.4 Role and Authorization Management**

- Role assignment to each user by Superadmin
- Default system roles (Superadmin and Admin)
- Fine-grained control over CRUD operations through claims
- Role-based dashboard views and accessibility

### **2.1.5 Department Management**

- CRUD of Departments

## **2.2 Non-Functional Requirements**

### **2.2.1 Usage**

The system is designed to ensure continuous availability, providing reliable access at all times.

### **2.2.2 Performance**

In a busy office environment, the system offers significantly faster performance compared to traditional pen-and-paper methods. Manual tasks like attendance management are automated for completion within seconds, ensuring efficient operations across all functionalities.

### **2.2.3 Scalability**

The project is built with scalability in mind, capable of accommodating growth and supporting an iterative development approach for future enhancements.

# 3 Design

## 3.1 ER Diagram

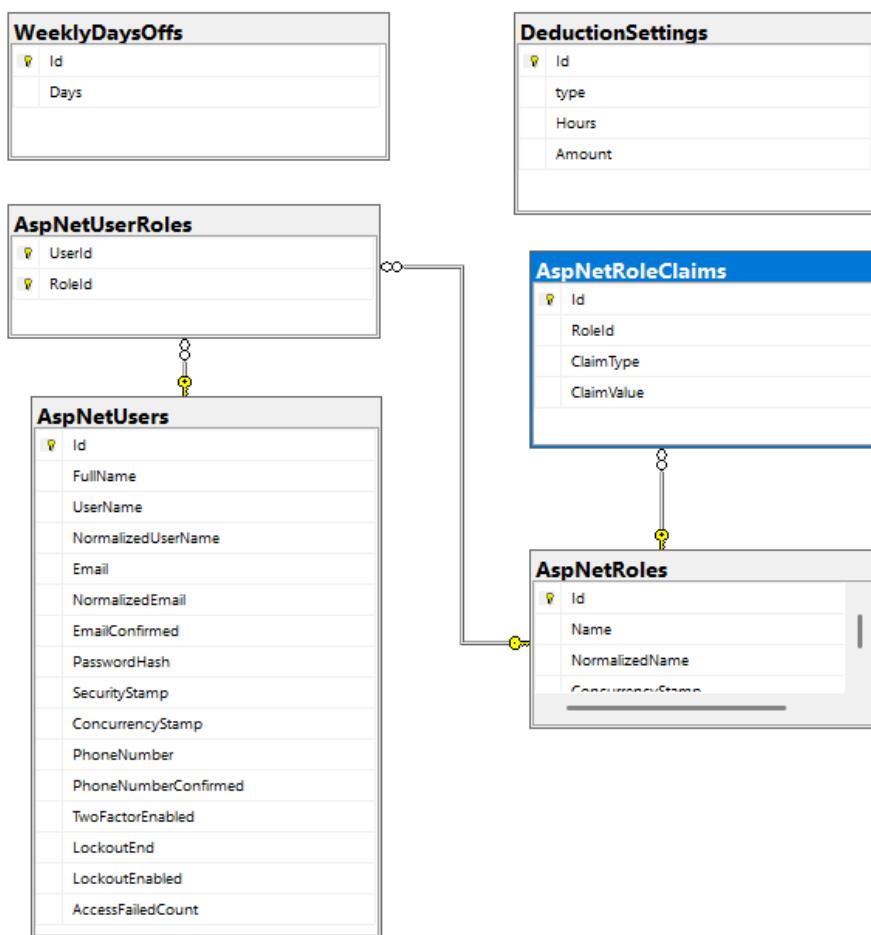


Figure 3.1: Entity Relationship Diagram - 1

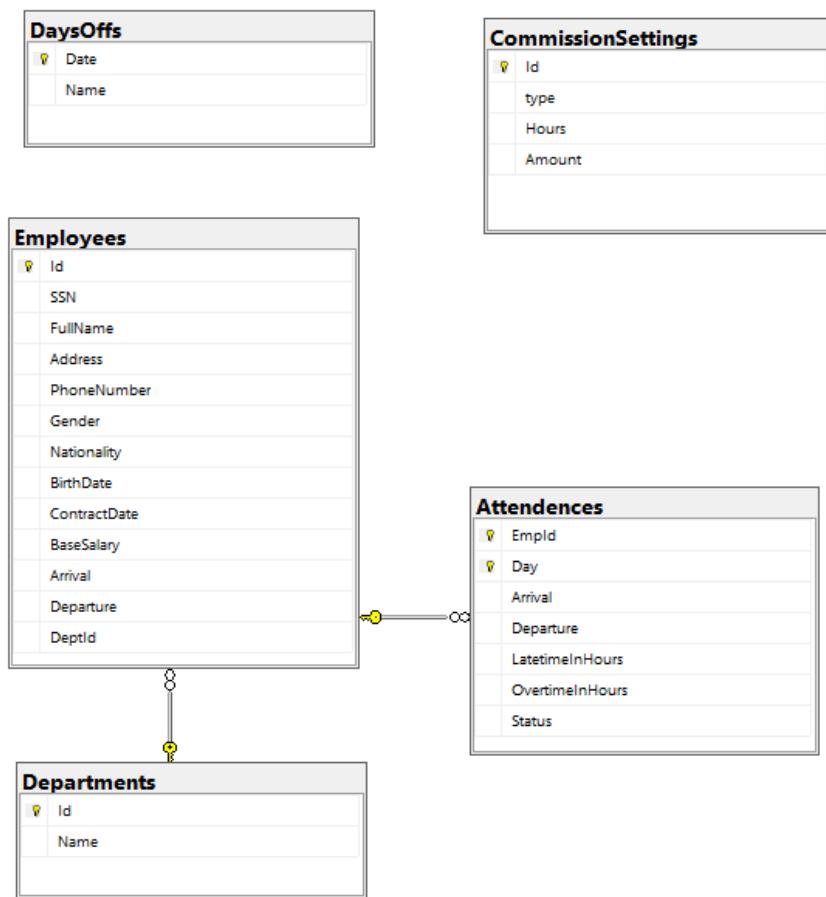


Figure 3.2: Entity Relationship Diagram - 2

## 3.2 Class Diagram

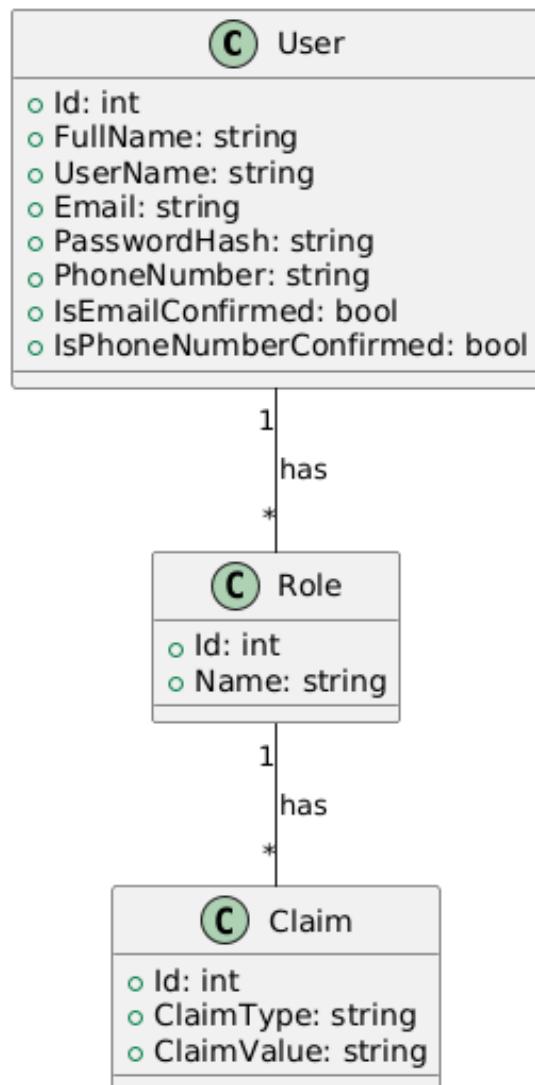


Figure 3.3: Class Diagram - 1

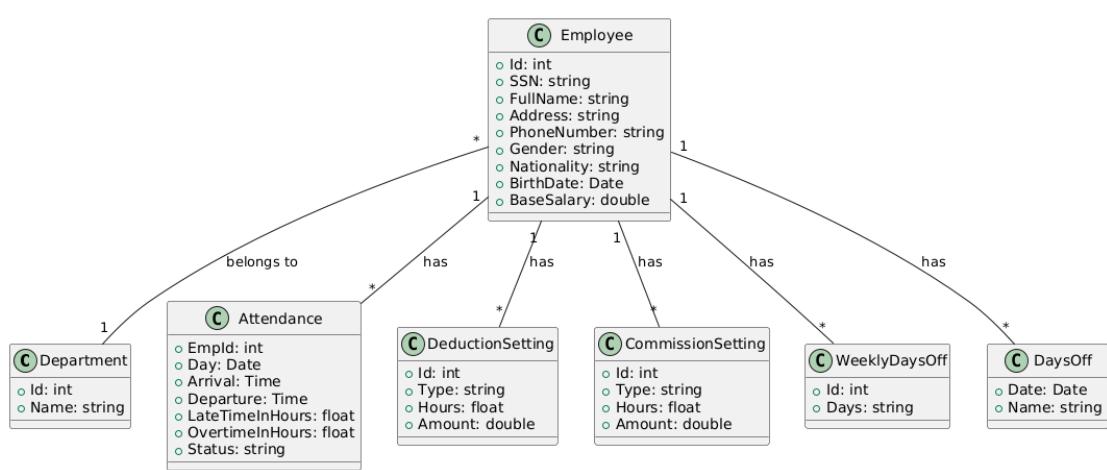


Figure 3.4: Class Diagram - 2

### 3.2.1 Attendance

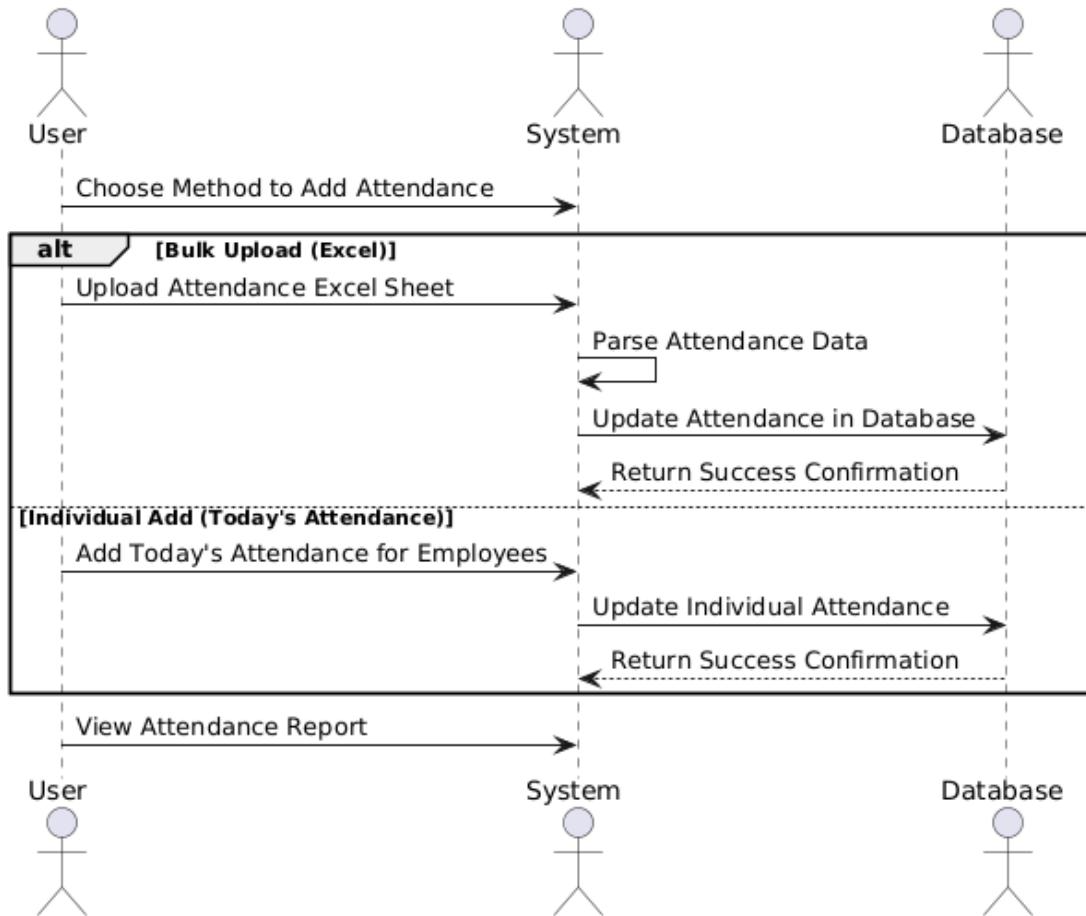


Figure 3.5: Sequence Diagram - Attendance

### 3.2.2 Payroll

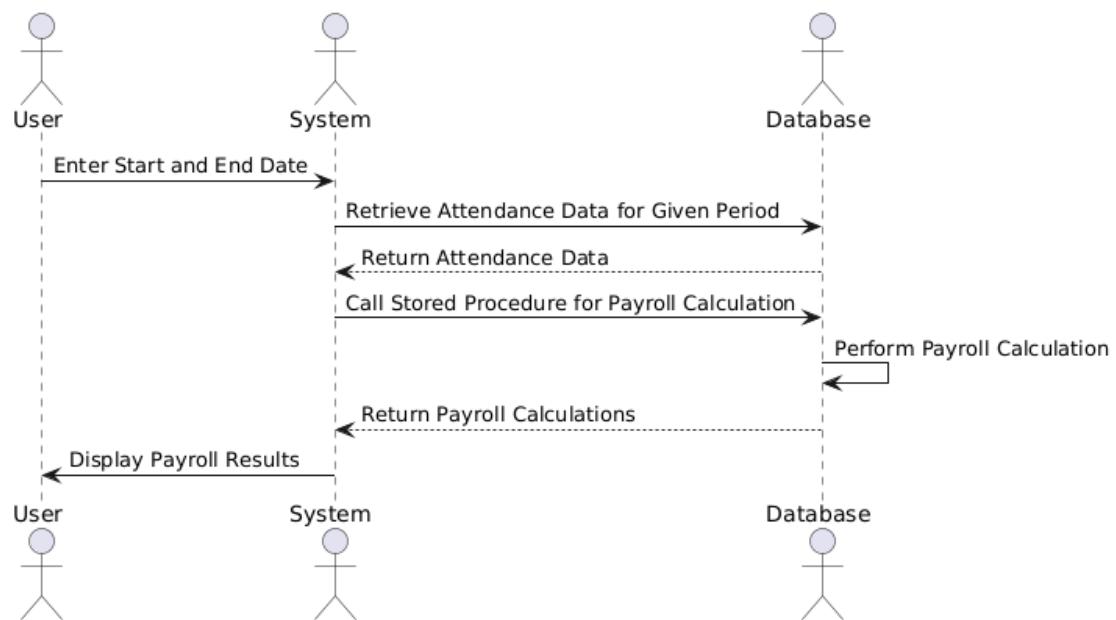


Figure 3.6: Sequence Diagram - Payroll

## **4 Implementation details**

### **4.1 Technologies Used**

#### **4.1.1 Frontend Technologies**

- Angular
- Special feature of type checking came very useful
- deployed on Azure

#### **4.1.2 Backend Technologies**

- .NET Framework
- WEB API architecture
- Deployment using Azure

#### **4.1.3 Database**

- SQL server management Studio
- Special feature of database diagram creation is used
- Deployed using Azure

#### **4.1.4 Development Environment**

- Visual Studio 2022
- Visual Studio Code
- SQL Server Management tool
- Azure site

### **4.2 Screenshots**

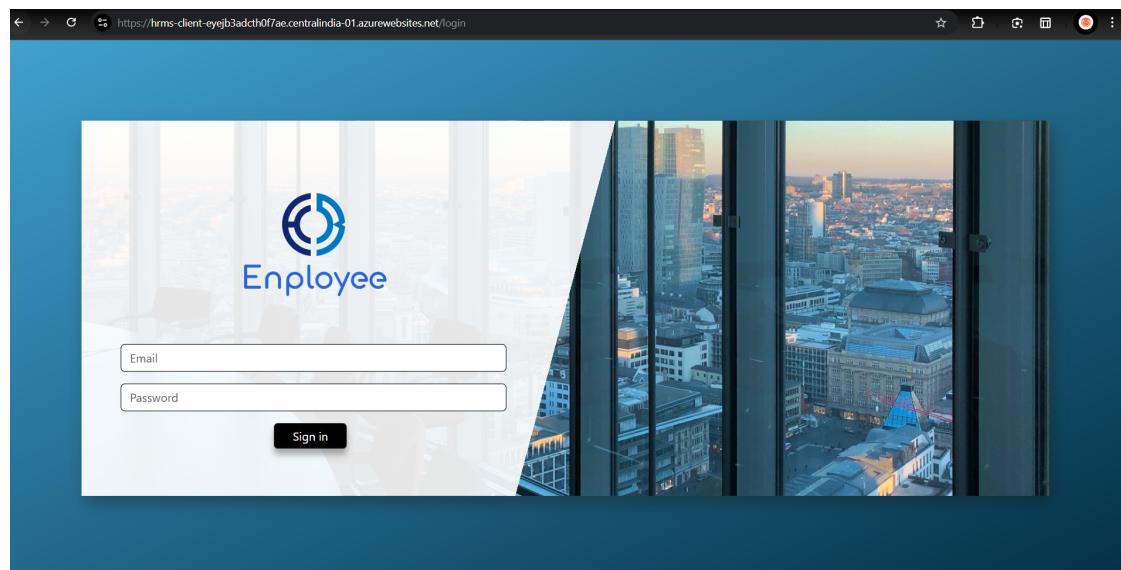


Figure 4.1: login page

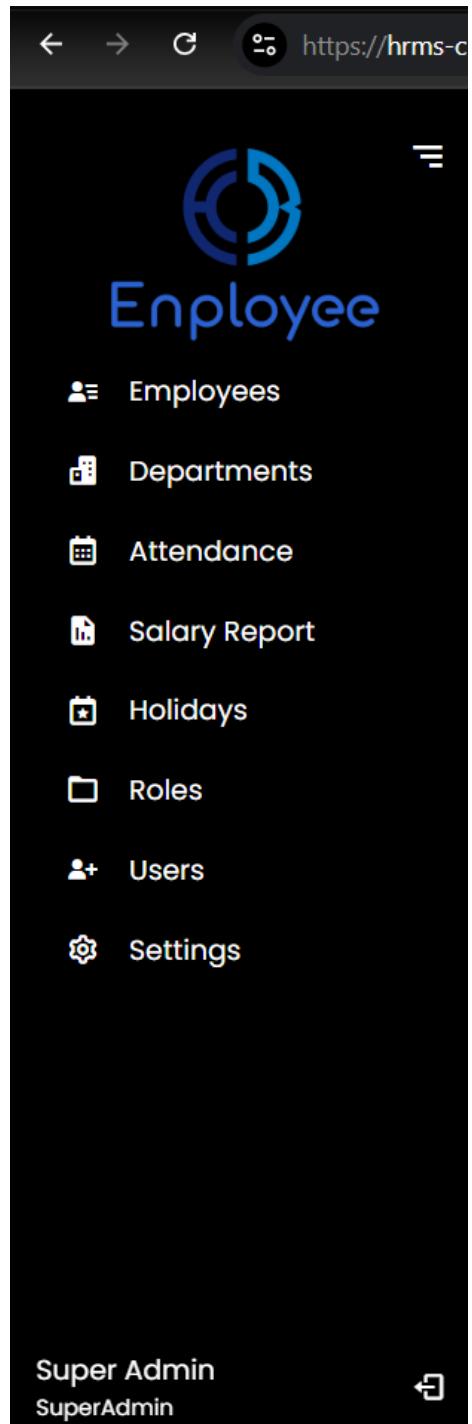


Figure 4.2: navigation bar

<https://hrms-client-eyejb3adcth0f7ae.centralindia-01.azurewebsites.net/employee/display>

Full Name	Social Security Number	Address	Gender	Phone Number	Base Salary	Birth Date	Contract Date	Clock In	Clock Out	Department	Actions
Employee 1	526555007.24012816	Address 1	Female	9.37417e+009	34656.09	2001-06-06	2017-07-26	18:00:00	18:00:00	Intern	
Employee 2	703509158.0722036	Address 2	Female	9.95921e+009	90833.16	2008-11-10	2017-09-30	09:00:00	01:00:00	SDE	
Employee 3	169389189.6469565	Address 3	Female	9.21306e+009	14818.13	1998-06-09	2019-12-27	20:00:00	04:00:00	HR	
Employee 4	987693136.0049014	Address 4	Female	9.57949e+009	74085.83	2017-03-28	2022-04-25	06:00:00	09:00:00	Intern	
Employee 5	806186721.2822202	Address 5	Female	9.01518e+009	34174.85	2011-12-01	2024-11-26	18:00:00	01:00:00	SDE	
Employee 6	140052511.448811	Address 6	Female	9.90686e+009	29874.15	2020-12-11	2019-04-06	08:00:00	16:00:00	HR	
Employee 7	295370572.8467754	Address 7	Female	9.45917e+009	81859.47	2007-08-10	2024-09-15	00:00:00	01:00:00	Intern	

« Previous 1 Next »

Figure 4.3: employee list

<https://hrms-client-eyejb3adcth0f7ae.centralindia-01.azurewebsites.net/department/display>

Department ID	Department Name	Actions
3	HR	
2	Intern	
1	SDE	

« Previous 1 Next »

Figure 4.4: departments

<https://hrms-client-eyejb3adcth0f7ae.centralindia-01.azurewebsites.net/dayoff>

#	Name	Date	Handles
1	Holi	2025-03-21	

« Previous 1 Next »

Figure 4.5: holidays

	A	B	C	D	E
1	Empld	Day	Arrival	Departure	Status
2	1	2025-02-01	09:00:00	17:00:00	Present
3	1	2025-02-02			Absent
4	1	2025-02-03	09:00:00	17:00:00	Present
5	1	2025-02-04			Absent
6	1	2025-02-05	09:00:00	17:00:00	Present
7	1	2025-02-06			Absent
8	1	2025-02-07	09:00:00	17:00:00	Present
9	2	2025-02-01	09:00:00	17:00:00	Present
10	2	2025-02-02			Absent
11	2	2025-02-03	09:00:00	17:00:00	Present
12	2	2025-02-04			Absent
13	2	2025-02-05	09:00:00	17:00:00	Present
14	2	2025-02-06			Absent
15	2	2025-02-07	09:00:00	17:00:00	Present
16	3	2025-02-01	09:00:00	17:00:00	Present
17	3	2025-02-02			Absent
18	3	2025-02-03	09:00:00	17:00:00	Present
19	3	2025-02-04			Absent
20	3	2025-02-05	09:00:00	17:00:00	Present
21	3	2025-02-06			Absent
22	3	2025-02-07	09:00:00	17:00:00	Present

Figure 4.6: attendance sheet format

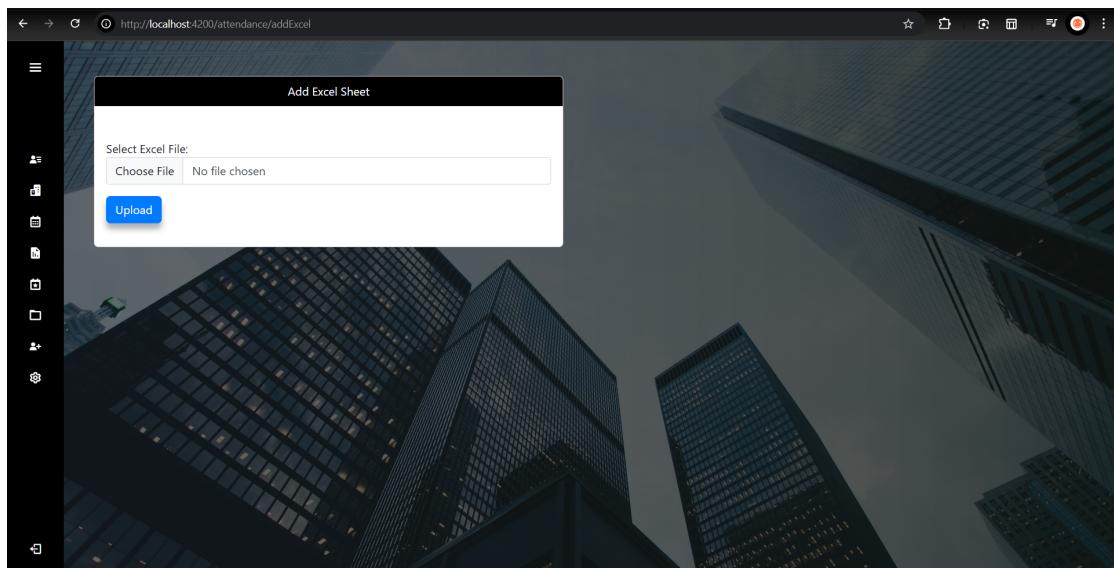


Figure 4.7: bulk attendance addition form

Department	Employee name	Clock in	Clock out	Date	Status	Handler
Intern	Employee 1	09:00:00	17:00:00	2025-02-01	Attended	
Intern	Employee 1	-----	-----	2025-02-02	Absent	
Intern	Employee 1	-----	-----	2025-02-03	Absent	
Intern	Employee 1	09:00:00	17:00:00	2025-02-04	Attended	
Intern	Employee 1	09:00:00	17:00:00	2025-02-05	Attended	
Intern	Employee 1	09:00:00	17:00:00	2025-02-06	Attended	
Intern	Employee 1	09:00:00	17:00:00	2025-02-07	Attended	
Intern	Employee 1	09:00:00	17:00:00	2025-02-08	Attended	

Figure 4.8: attendance list

```
using (var stream = file.OpenReadStream())
{
    var excelData = new List<List<object>>();
    using (var reader = ExcelReaderFactory.CreateReader(stream))
    {
        var resultDataset = reader.AsDataSet(new ExcelDataSetConfiguration
        {
            FilterSheet = (tableReader, sheetIndex) => true,
            UseColumnDataType = true,

            ConfigureDataTable = _ => new ExcelDataTableConfiguration
            {
                UseHeaderRow = true,
                //filter columns
                FilterColumn = (rowReader, columnIndex) =>
                {
                    return columnIndex == 0 || columnIndex == 1 || columnIndex == 2 || columnIndex == 3 || columnIndex == 4;
                }
            }
        });
        dataTable = resultDataset.Tables[0];
    }
}
```

Figure 4.9: code for excel attendance addition

Employee Name	Department	Base Salary	Attendance Days	Absence Days	Overtime (hours)	Deduction (hours)	Overall Overtime	Overall Deduction	Net Salary	Handler
Employee 1	Intern	34656.09	28	2	0	0	0	2475.44	32180.66	
Employee 2	SDE	90833.16	28	3	0	0	0	9732.12	81101.04	
Employee 3	HR	14818.13	28	3	0	0	0	1587.66	13230.47	
Employee 4	Intern	74085.83	28	2	0	0	0	5291.85	68793.99	
Employee 5	SDE	34174.85	28	2	0	0	0	2441.06	31733.79	
Employee 6	HR	29874.15	28	3	0	0	0	3200.8	26673.35	
Employee 7	Intern	81859.47	28	2	0	0	0	5847.11	76012.37	

« Previous 1 Next »

Figure 4.10: salary report

```

SELECT
    EmpId,
    FullName,
    DepartmentName,
    BaseSalary,
    AttendanceDays,
    AbsentDays,
    TotalOvertimeHours,
    TotalLateHours,
    ShiftDuration,

    -- Pre-calculate the hourly rate and reuse it
    ROUND(BaseSalary / (AttendanceDays * ShiftDuration), 2) AS HourlyRate,

    -- Salary Calculation
    ROUND(BaseSalary+((BaseSalary / (AttendanceDays * ShiftDuration)) * TotalOvertimeHours)-(
        (BaseSalary / (AttendanceDays * ShiftDuration)) * TotalLateHours
    )
    +
    ((BaseSalary / AttendanceDays) * AbsentDays
    )),2)
    AS CalculatedSalary,

    -- Additional Pay (Overtime)
    ROUND(
        (BaseSalary / (AttendanceDays * ShiftDuration)) * TotalOvertimeHours,
        2
    ) AS AdditionalPay,

    -- Deduction Pay (Lateness & Absence)
    ROUND(
        (
            (BaseSalary / (AttendanceDays * ShiftDuration)) * TotalLateHours
        )
        +
        ((BaseSalary / AttendanceDays) * AbsentDays
        ),
        2
    ) AS DeductionPay

FROM
    AttendanceData
    order by EmpId;

```

Figure 4.11: stored procedure code

## Attendance Settings

### Overtime Settings

Method

Hours

Hours  
2

### Weekly days off

- Saturday
- Sunday
- Monday
- Tuesday
- Wednesday
- Thursday
- Friday

### Deduction Settings

Method

Money

Amount  
200

**Submit**

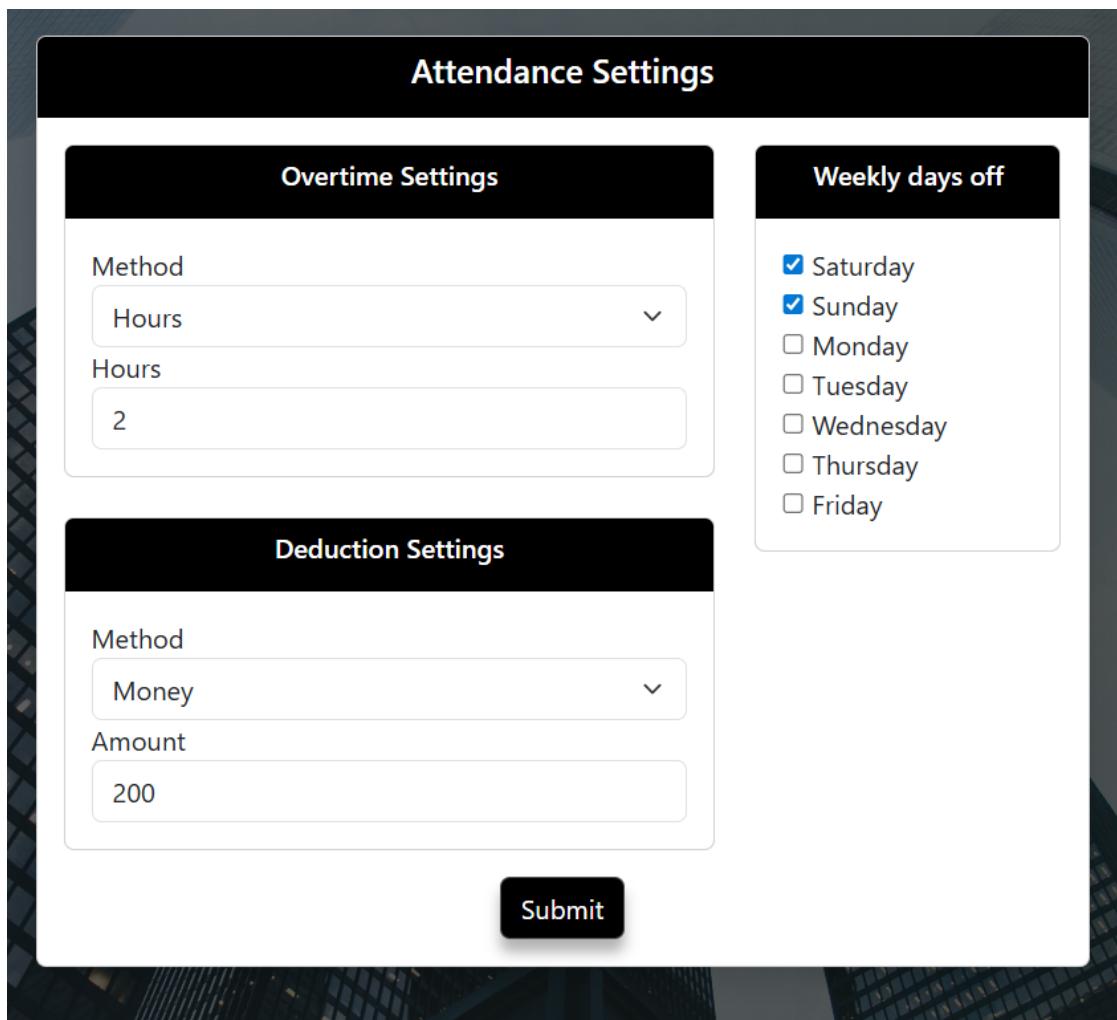


Figure 4.12: settings

Add User		Search <input type="text"/>	
Full Name	Email	Role	Actions
Vaibhav	vaibhav@gmail.com	Admin	
Admin	admin@gmail.com	Admin	
Super Admin	super@gmail.com	SuperAdmin,Admin	
Madhav	madhav@gmail.com	HR	

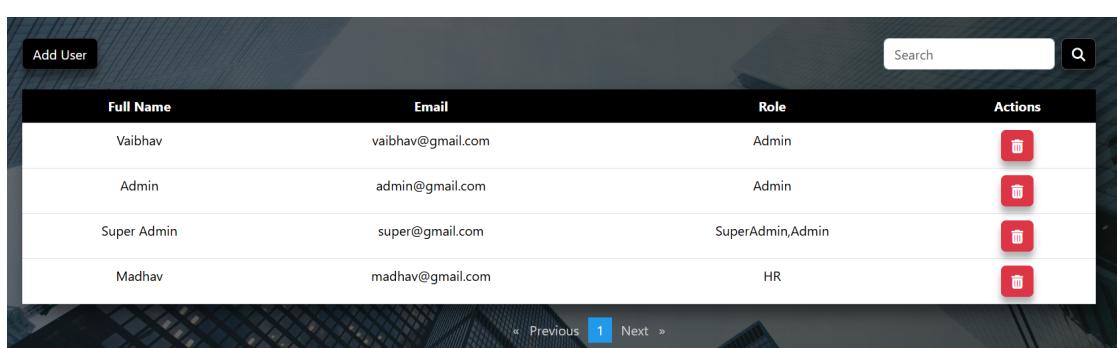


Figure 4.13: users

Role Name		Operations
SuperAdmin		
HR		
Admin		

Figure 4.14: roles

Name	Type	Location
Application Insights Smart Detection	Action group	Global
HRMS	App Service plan	Central India
hrms-app-service	App Service	Central India
hrms-client	Application Insights	Central India
hrms-client	App Service	Central India
hrms-db (hrms-server/hrms-db)	SQL database	Central India
hrms-server	SQL server	Central India

Figure 4.15: azure online hosting

# 5 Testing

## 5.1 Overview

The testing phase encompassed unit and stress testing, focusing on key metrics such as database query performance and maintainability scores from Visual Studio 2022. Integration testing was performed manually, ensuring seamless functionality by verifying the application after each module integration.

## 5.2 API Testing

The system's APIs were thoroughly tested using **Swagger**, an API documentation and testing tool. Swagger provided an interactive interface to validate API endpoints, ensuring that each endpoint responded correctly to different request types, including GET, POST, PUT, and DELETE. It also facilitated the verification of request payloads, response codes, and data consistency.

## 5.3 Frontend Testing

Once the APIs were successfully tested, the backend was integrated with the **Angular** frontend. The frontend underwent rigorous manual testing to assess the accuracy of data representation, navigation flows, and overall user experience. **Lighthouse** was used to generate performance and seo score for the initial page.

## 5.4 Database Testing

To validate database performance and scalability, stress testing was performed using synthetic data. ChatGPT was utilized to generate large datasets resembling real-world scenarios. Queries were executed to measure database response times and identify potential bottlenecks.

A significant observation was made during the payroll calculation testing. Initially, payroll was processed using a backend loop-based approach, which took approximately three minutes to handle 10,000 employee records. After optimization using SQL stored procedures, the same payroll calculations were completed in less than 10 seconds. This remarkable performance improvement was a key success indicator of the testing phase.

**Figure 5.1 displays the execution of a stored procedure for a test case.**

The screenshot shows a SQL Server Management Studio (SSMS) window with three tabs: Results, Messages, and Client Statistics. The Results tab displays a table of query profile statistics across four trials (Trial 3, Trial 2, Trial 1, Average). The table includes sections for Query Profile Statistics, Network Statistics, and Time Statistics. In the Time Statistics section, the 'Total execution time' row is highlighted in blue, showing values of 45, 52, 46, and 47.6667 respectively. The 'Wait time on server replies' row is also visible.

	Trial 3	Trial 2	Trial 1	Average
<b>Query Profile Statistics</b>				
Number of INSERT, DELETE and UPDATE statements	0	→ 0	→ 0	→ 0.0000
Rows affected by INSERT, DELETE, or UPDATE statements	0	→ 0	→ 0	→ 0.0000
Number of SELECT statements	4	→ 4	↑ 3	→ 3.6667
Rows returned by SELECT statements	5	→ 5	→ 5	→ 5.0000
Number of transactions	0	→ 0	→ 0	→ 0.0000
<b>Network Statistics</b>				
Number of server roundtrips	5	→ 5	→ 5	→ 5.0000
TDS packets sent from client	6	→ 6	→ 6	→ 6.0000
TDS packets received from server	5	→ 5	→ 5	→ 5.0000
Bytes sent from client	6354	→ 6354	↑ 6248	→ 6318.6670
Bytes received from server	2753	↓ 2764	↑ 1438	→ 2318.3330
<b>Time Statistics</b>				
Client processing time	9	↓ 13	↑ 4	→ 8.6667
<b>Total execution time</b>	<b>45</b>	<b>↓ 52</b>	<b>↑ 46</b>	<b>→ 47.6667</b>
Wait time on server replies	36	↓ 39	↓ 42	→ 39.0000

Figure 5.1: database payroll client analysis using ssms query

The screenshot shows a SQL Server Management Studio (SSMS) window with three tabs: Results, Messages, and ClientStatistics. The Results tab displays the following output:

```

SQL Server parse and compile time:
CPU time = 15 ms, elapsed time = 17 ms.

(3 rows affected)

SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 1 ms.

SQL Server Execution Times:
CPU time = 15 ms, elapsed time = 19 ms.

SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 0 ms.
Stored procedure is created and tested successfully.

Completion time: 2025-03-25T12:31:41.1081441+05:30

```

Figure 5.2: database payroll execution time - 19ms

Code Metrics Results		Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Source code	Lines of Executable code	
Hierarchy	Filter: None	Min:	Max:					
WebAPI (Debug)		83	626	6	281	7,392	2,112	
__ WebAPI		38	1	1	72	130	54	
__ WebAPI.Constants		94	8	1	5	102	15	
__ WebAPI.Controllers		67	166	2	137	1,193	443	
__ WebAPIDTOs		99	100	1	11	156	0	
__ WebAPI.Extensions		76	1	1	8	18	3	
__ WebAPI.Filters		80	21	2	25	94	27	
__ WebAPI.Helpers		79	41	2	38	199	63	
__ WebAPI.Interfaces		100	39	0	13	88	0	
__ WebAPI.Migrations		35	22	2	41	4,543	1,336	
__ WebAPI.Models		99	127	6	33	196	10	
__ WebAPI.ProjectProcessing		74	26	1	28	223	50	
__ WebAPI.Repositories		88	54	1	35	292	68	
__ WebAPI.Seeds		68	20	1	35	158	43	

Figure 5.3: Backend Code Maintainability Score

- **Maintainability Index:** Higher is better (ranges from 0 to 100). - 83
- **Depth of Inheritance:** Measures class hierarchy depth. - 6

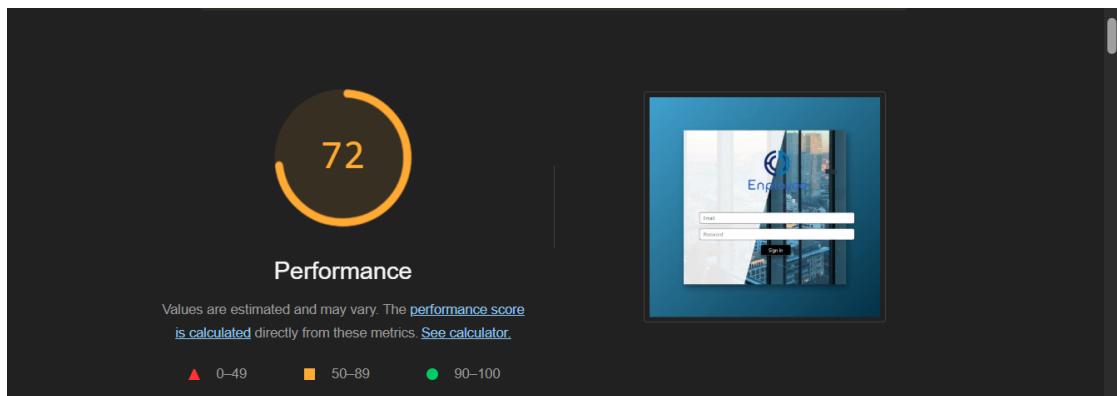


Figure 5.4: Frontend Lighthouse Performance score for initial page

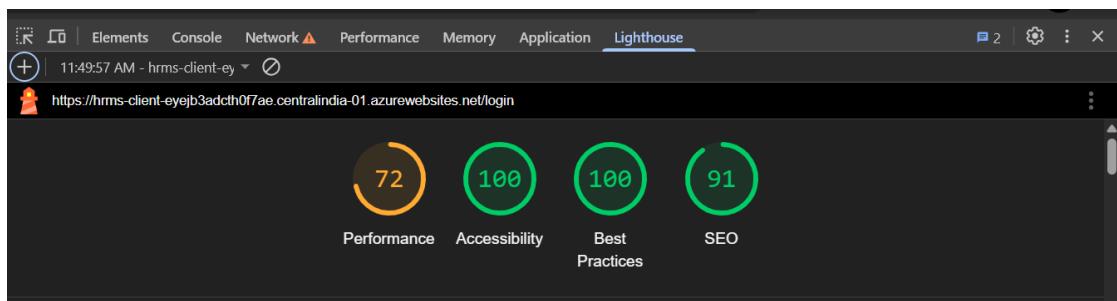


Figure 5.5: Frontend Lighthouse overall score for initial page

The screenshot shows the Swagger UI interface for a WebAPI 1.0 OAS3 definition named 'HRMS API V1'. At the top, there's a navigation bar with the 'Swagger' logo and a dropdown menu labeled 'Select a definition' set to 'HRMS API V1'. Below the header, the title 'WebAPI 1.0 OAS3' is displayed, along with the URL '/swagger/v1/swagger.json'. On the right side of the main content area, there's a green 'Authorize' button with a lock icon. The main content is organized into sections: 'ApplicationUser' and 'Attendance'. The 'ApplicationUser' section contains two POST methods: '/api/ApplicationUser/register' and '/api/ApplicationUser/login'. The 'Attendance' section contains three methods: a GET method for '/api/Attendance', a POST method for '/api/Attendance', and a POST method for '/api/Attendance/AddExcel'. Each method entry includes a small lock icon.

Figure 5.6: swagger-1

This screenshot shows another instance of the Swagger UI. It features a single POST method for '/api/ApplicationUser/login'. Below it, the 'Attendance' section lists several methods: a GET method for '/api/Attendance', a POST method for '/api/Attendance', a POST method for '/api/Attendance/AddExcel', a PUT method for '/api/Attendance/{empId}', a DELETE method for '/api/Attendance/{empId}', a GET method for '/api/Attendance/{empId}', a GET method for '/api/Attendance/GetEmployeeDay/{empId}', and a GET method for '/api/Attendance/GetByPeriod'. The 'DaysOff' section contains a GET method for '/api/DaysOff' and a POST method for '/api/DaysOff'. Similar to Figure 5.6, each method entry includes a lock icon.

Figure 5.7: Swagger-2

DaysOff	
GET	/api/DaysOff
POST	/api/DaysOff
GET	/api/DaysOff/{day}
PUT	/api/DaysOff/{day}
DELETE	/api/DaysOff/{day}

Department	
GET	/api/Department
POST	/api/Department
GET	/api/Department/{name}
PUT	/api/Department/{id}
DELETE	/api/Department/{id}

Employee	
GET	/api/Employee
POST	/api/Employee
GET	/api/Employee/{id}
PUT	/api/Employee/{id}
DELETE	/api/Employee/{id}

Figure 5.8: Swagger-3

Employee	
GET	/api/Employee
POST	/api/Employee
GET	/api/Employee/{id}
PUT	/api/Employee/{id}
DELETE	/api/Employee/{id}

Organization	
POST	/api/Organization
GET	/api/Organization

Salary	
POST	/api/Salary

Seeders	
GET	/api/Seeders/run-seeders

Figure 5.9: Swagger-4

The screenshot shows the Swagger UI interface for the SuperAdmin API. At the top, there is a header with the title "SuperAdmin". Below the header, there is a list of API endpoints categorized by method and path. Each endpoint is represented by a row with a color-coded background and a lock icon. The endpoints are:

- GET /api/SuperAdmin/GetUsers
- DELETE /api/SuperAdmin/DeleteUser
- GET /api/SuperAdmin/UserRoles
- POST /api/SuperAdmin/UpdateRoles
- GET /api/SuperAdmin/AllRoles
- POST /api/SuperAdmin/AddRole
- GET /api/SuperAdmin/GetRoleIdBy
- DELETE /api/SuperAdmin/DeleteRole
- GET /api/SuperAdmin/AllPermissions
- POST /api/SuperAdmin/AddPermission

Below the endpoint list, there is a section titled "Schemas" which contains a list of schema definitions:

- AttendanceDTO >
- CheckBoxDTO >
- CommissionDTO >
- DaysOffDTO >
- DeductionDTO >
- DepartmentDTO >
- EmployeeDTO >
- LoginDTO >
- OrganizationSettings >

Figure 5.10: Swagger-5

The screenshot shows the Swagger UI interface focusing on the "Schemas" section. The title "Schemas" is at the top of the list. Below it, there is a list of schema definitions, each preceded by a right-pointing arrow:

- AttendanceDTO >
- CheckBoxDTO >
- CommissionDTO >
- DaysOffDTO >
- DeductionDTO >
- DepartmentDTO >
- EmployeeDTO >
- LoginDTO >
- OrganizationSettings >

Figure 5.11: Swagger-6

# 6 Conclusion and Future Extensions

## 6.1 Conclusion

The HR Management System has successfully achieved its objectives of automating and streamlining key HR processes. From attendance tracking using Excel sheet uploads to optimized payroll processing through SQL stored procedures, the system has demonstrated significant efficiency improvements. Role-based access management ensures secure and restricted access to sensitive information, while the intuitive user interface simplifies HR operations.

Furthermore, the reduction in payroll processing time from three minutes to under ten seconds for large datasets highlights the effectiveness of the system's design and implementation. The successful integration of .NET Core, Angular, and SQL Server has provided a scalable and robust solution suitable for organizations of all sizes.

## 6.2 Future Extensions

### 6.2.1 AI Chatbot for Employee Report Analysis

An AI chatbot can serve as a virtual HR assistant, capable of answering employee-related queries by analyzing reports and contract data. The chatbot would be particularly beneficial for HR managers, as it could provide instant insights without the need to manually search through records.

#### Implementation Approach

**Gemini API with Vector Database.** Convert your document to vector embeddings and provide them as input to the Gemini API. Then you can generate accurate answers to your questions, with proper reference to the original phrase in the employee policy.

### 6.2.2 Further Optimize the payroll calculations to include concurrency

To further optimize payroll processing, concurrency can be introduced to handle large datasets efficiently. By dividing the payroll calculation task into smaller, independent operations and executing them in parallel, significant time reductions can be achieved. Implementing multi-threading or parallel processing within SQL Server or using application-level concurrency mechanisms will ensure faster and more efficient payroll generation, especially for organizations with thousands of employees.