**Employee Payroll Management**

|  |  |
| --- | --- |
| **Group :** | 02 |
| **Project Name:** | Employee payroll management |

**Table of Contents**

1. INTODUCTION
2. SYSTEM OVERVIEW
3. SUB-SYSTEM DETAILS
4. DATA ORGANIZATION
5. REST APIs to be Built
6. ASSUMPTIONS
7. EXPECTATIONS
8. ACCEPTANCE CRITERIA
9. OUTPUT SCREENSHOTS
10. CONCLUSION
11. **Introduction**

XYZ Ltd is a company which builds a software system which is responsible for Employee payroll management of employees.

XYZ Ltd plans to develop “Employee payroll management” – web application, where Admin can register, login, add employees, add employees for payment etc. and Employee can register, login, see their profile, payslip etc.

**Abstract:**

“Employee Payroll Management” is designed to make the existing manual system automatic with the help of computerised equipment and full-edged computer software, fulfilling their requirements, so that their valuable data and information can be stored for a longer period with easy access and manipulation of the same. The required software is easily available and easy to work with. This web application can maintain and view computerised records without getting redundant entries. The project describes how to manage user data for good performance and provide better services for the client.

**Background of the project:**

Employee payroll management is an internet-based Java application that automates the working of a company or work center that manage and maintain records of the employees in different department. The objective of the Employee payroll management project is to design a scheduling system for a work center. Scheduling is such a tool with which the process of intimating activities and notifications will be easy and even online in the organization where it is installed. But these tasks of scheduling the different activities if manually done whether they may be personal, or official is time consuming and may lead to confusion if not properly scheduled. The supervisor holds the various activities like sending notifications, mark attendance, and deleting the employees, and on the other hand employees view their details, view schedule or the notifications or any message from supervisor, and view attendance.

**Scope and Overview:**

The scope of the “Employee payroll management” will be provide the functionality as declared below. The system will be developed on a windows operating system using Java/J2EE, Hibernate, Angular, Spring Boot, MySQL.

**2 System overview**

The “Employee payroll management” should support basic functionalities for all below listed users.

* Admin
* Employee

**2.1 Authentication & Authorization**

**Authentication:**

Any employee should be authenticated using a unique emailed and password.

**Authorization**

The operations supported and allowed would be based on the user type. For example, Admin has the rights to add Employee and add Payment.

Whereas Employee has right to view their profile and payslip.

**Password Rules:**

When it comes to password safety, the longer and more complex the password is, the better. We think its good practice to enforce certain minimum requirements when asking users to create a new password. Of course, you have to find a balance between these requirements and user experience. If you make the sign-up process too tedious, you could be driving users away. To enforce password strength, you should define a set of rules that a password must satisfy and then enforce these with form validation. Example password strength rules:

• Minimum of 8 characters

• At least one uppercase letter

• At least one number

• At least one special character

**2.2 Requirement Specification:**

. To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

**HARDWARE REQUIREMENTS:**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

**HARDWARE REQUIREMENTS FOR PRESENT PROJECT**:

PROCESSOR: Intel dual Core, i3

RAM: 1 GB

HARD DISK: 80 GB

**SOFTWARE REQUIREMENTS:**

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

**SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:**

OPERATING SYSTEM: Windows 10

FRONT END: Html, CSS, java script. (Angular)

BACK END: STS (Spring Tool Suite)

SERVER-SIDE SCRIPT: Apache tomcat 8

DATABASE: MySQL

TESTING: Junit and Postman

**Assumptions and Dependencies:**

**Assumptions:**

• The code should be free with compilation errors/syntax errors.

• The product must have an interface which is simple enough to understand.

**Dependencies:**

• All necessary hardware and software are available for implementing and use of the tool.

• The proposed system would be designed, developed and implemented based on the software requirements specifications document.

• End users should have basic knowledge of computer and we also assure that the users will be given software training documentation and reference material.

• The system is not required to save generated reports.

**2.4 Schematic Diagram:**

**ADMIN**

**ADD EMP**

**ADD PAYMENT**

**ADD WORK**

**ADD ATTENDENCE**

**PROFILE**

**VIEW PAYSLIP**

**VIEW ATTENDENCE**

**ADD TIMESHEET**

**VIEW TIMESHEET**

**EMPLOYEE**

**LOGOUT**

**VIEW SCHEDULE**

**REGISTER**

**NO**

**LOGIN**

**LOGIN**

**NO**

**2.5 Employee payroll management Database**

ID

EMAIL\_ID

Password

USER\_NAME

USER\_NAME

PASSWORD

EMP\_ID

CITY

DESIGNATION

DOB

EMAIL\_ID

F\_NAME

GENDER

L\_NAME

LOCATION

MOB\_NO

PASSWORD

PINCODE

STATE

STREET

USER \_NAME

PASSWORD

**ADMIN REGISTER**

**ADMIN LOGIN**

**EMPLOYEE REGISTER**

**EMPLOYEE LOGIN**

**DURACTION**

**END\_DATE**

**LEAVE\_TYPE**

**START\_DATE**

**ADD LEAVE**

BREAK\_TIME

DATE

IN\_TIME

OUT\_TIME

OVERTIME\_HOURS

REGULAR\_TIME

TOTAL\_HOURS

**ADD TIMESHEET**

DATE

STATUS

**ADD ATTENDENCE**

**ADD PAYMENT**

BASIC\_PAY

DA

DEDUCTION

EMP\_ID

GROSS\_SALARY

HRS

NET\_PAY

PF\_AMOUNT

TAX

TRAN\_DATW

**Admin registration table:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Id | Int |  |
| Email\_id | Varchar | 255 |
| Password | Varchar | 255 |
| User\_name | varchar | 255 |

**Employee registration table:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Emp\_id | Int |  |
| City | Varchar | 255 |
| Designation | Varchar | 255 |
| Dob | Varchar | 255 |
| Email\_id | Varchar | 255 |
| F\_name | Varchar | 255 |
| Gender | Varchar | 255 |
| L\_name | Varchar | 255 |
| Location | Varchar | 255 |
| Mob\_no | Varchar | 255 |
| Password | Varchar | 255 |
| Pincode | Int |  |
| State | Varchar | 255 |
| Street | Varchar | 255 |

**Admin Login:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Email\_id | Varchar | 255 |
| Password | Varchar | 255 |

**Employee Login:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Email\_id | Varchar | 255 |
| Password | Varchar | 255 |

**Salary table:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Sal\_id | Int |  |
| Basic\_pay | Double |  |
| Da | Double |  |
| Deduction | Double |  |
| Emp\_id | Int |  |
| Gross\_salary | Double |  |
| Hra | Double |  |
| Net\_pay | Double |  |
| Pf\_amount | Double |  |
| Tax | Double |  |
| Tran\_date | Varchar | 255 |

**Timesheet table:**

|  |  |  |
| --- | --- | --- |
| **column** | **Datatype** | **Length** |
| Emp\_id | Int |  |
| Break\_time | Varchar | 255 |
| Date | Varchar | 255 |
| In\_time | Varchar | 255 |
| Out\_time | Varchar | 255 |
| Overtime\_hours | Varchar | 255 |
| Regular\_time | Varchar | 255 |
| Total\_hours | Varchar | 255 |

**Attendance table:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Emp\_id | Int |  |
| Date | Varchar | 255 |
| Status | Varchar | 255 |

**Leaves table:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Emp\_id | Int |  |
| Duration | Varchar | 255 |
| End\_date | Varchar | 255 |
| Leave\_type | Varchar | 255 |
| Start\_date | Varchar | 255 |

**Work Schedule table:**

|  |  |  |
| --- | --- | --- |
| **Column** | **Datatype** | **Length** |
| Emp\_id | Int |  |
| Date | Varchar | 255 |
| Shift | Varchar | 255 |
| Strat\_time | Varchar | 255 |
| End\_time | Varchar | 255 |
| Duration | Varchar | 255 |

**case diagram:**

case diagram is a visual representation of how a user might interact with a program. A use case diagram depicts the system’s numerous use cases and different sorts of users. The circles or ellipses are used to depict the use cases.

**Employee**

**ADMIN**

**3 Sub-System Details**

The Employee payroll management is defined, where in all employees need to login successfully before performing any of their respective operations. Below tables that provides functionality descriptions for each type of user. Against each requirement, indicative data is listed in column ‘Data to include’.

**Admin**

The admin as a user is defined to perform below listed operations after once admin login.

|  |  |  |  |
| --- | --- | --- | --- |
| **Objects** | **Operations** | **Data to include** | **Remarks** |
| User | register | Id,Email\_id,User\_name,Password |  |
| Employee list | view | Emp\_id,Email\_id,Designation,F\_name,L\_name,Mob\_no,gender.. |  |
| Payment | generate | Sal\_id,Basicpay,Deduction,Emp\_id,Gross\_sal,Net\_pay |  |
| Work schedule | add | Emp\_id |  |

**Employee**

The Employee as a user is defined to perform below listed operations after once employee login.

|  |  |  |  |
| --- | --- | --- | --- |
| **Objects** | **Operations** | **Data to include** | **Remarks** |
| User | register | Id,Email\_id,User\_name,Password,Designtion  Mob\_no,gender.. |  |
| Profile | view | Emp\_id,Email\_id,Designation,F\_name,L\_name,Mob\_no,gender,password.. |  |
| Payslip | view | Sal\_id,Basicpay,Deduction,Emp\_id,Gross\_sal,Net\_pay,date\_Trans.. |  |
| Work schedule | view | Emp\_id |  |

**4 Data Organization**

This section explains the data storage requirements of the Employee payroll System and indicative data description along with suggested table (database) structure. The following section explains few of the tables (fields) with description. However, in similar approach need to be considered for all other tables.

**Admin Registration table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Admin\_id | Id is autogenerated after Registration |
| User\_name | Name of the admin |
| Email\_id | Email of the admin |
| Password | Admin unique password |

**Employee Registration table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Emp\_id | Id is autogenerated |
| Email\_id | Email of the employee |
| Password | Employee unique password |
| F\_name | Employee first name |
| L\_name | Employee last name |
| Mob\_no | 10 digit employee mobile number |
| Gender | Give gender weather male/female |
| Address | Address of the employee |
| Designation | Designation of employee |

**Payment table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Sal\_id | Id is autogenerated |
| Basic\_pay | The total basic salary |
| Deduction | Deduction in salary |
| Emp\_id | Employee id |
| Gross\_salary | Monthly salary |
| Hra | Hr allowances |
| Net\_pay | In hand Total salary |
| Tran\_date | Date of transaction |

**Time Sheet table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Emp\_id | Employee id |
| Date | Particular date |
| Regular\_time | Regular working hours |
| Overtime\_hours | Over time done by employee |
| Total\_hours | Total hours work done |

**Attendance table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Emp\_id | Employee id |
| Date | The attendance date |
| Status | Weather employee present/absent |

**Leaves table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Emp\_id | Employee id |
| Duration | How many leave days |
| Start\_date | Starting date of leave |
| End\_date | Ending date of leave |
| Leave\_type | Type of leave the employee taking |

**Work Schedule table**

|  |  |
| --- | --- |
| **Field name** | **Description** |
| Emp\_id | Employee id |
| Date | Date of work schedule |
| Shift | Shift of work |
| Start\_time | Start time of work |
| End\_time | End time of work |
| Duration | Duration of work |

**5 REST APIs to be Built**

Create following Rest resources which are required in the application

1. Creating Admin Entity: Create Spring Boot Application with Spring Data JPA

Technology stack:

* Spring Boot
* Spring REST
* Spring Data JPA

Here we have multiple layers into the application:

* Create an Entity: Admin
* Create a AdminRegistrationRepository interface and will make use of Spring data JPA

1. Will have findByEmailId method.
2. Will have findAdmByEmailIdandPassword
3. Add admin details

* Create AdminRegistrationService class and will expose all these services
* Finally, create Admin AdminRegistrationController will have the following Uri’s

POST

http://localhost:8085/api/v1/register

<http://localhost:8085/api/v1/login>

1. Create a AdminRepository interface and will make use of Spring data JPA
2. will have createEmployee methode
3. will have findById

* Finally, create Admin AdminController will have the following Uri’s

GET

<http://localhost:8085/admin>s

[http://localhost:8085/{empId}](http://localhost:8085/%7bempId%7d)

POST

[http://localhost:8085/{empId}](http://localhost:8085/%7bempId%7d)

DELETE

[http://localhost:8085/](http://localhost:8085/%7bempId%7d)admins

1. ­­­Create an Entity: Employee

* Create a EmpRepository interface and will make use of Spring data JPA

a) Will have findById methode.

* Create RegistrationService class and will expose all these services
* Finally, create EmpController will have the following Uri’s

GET

[http://localhost:8085/api/v2/ {empId}](http://localhost:8085/api/v2/%20%7bempId%7d)

1. ­­­Create an Entity: Employee

* Create a EmpRepository interface and will make use of Spring data JPA

a) Will have findById methode.

* Finally, create RegistrationController will have the following Uri’s

POST

http://localhost:8085/emlployees

<http://localhost:8085/employees/login>

PUT

[http://localhost:8085/forgotpassword](http://localhost:8085/forgotpasswordemp)

1. ­­­Create an Entity: Attendance

* Create a AttendanceRepository interface and will make use of Spring data JPA

a) Will have findById methode.

* Finally, create EmpController will have the following Uri’s

GET

[http://localhost:8085/api/v3/{empId}](http://localhost:8085/api/v3/%7bempId%7d)

[http://localhost:8085/api/v3/attendances](http://localhost:8085/api/v3/employees)

POST

<http://localhost:8085/api/v3/attendances>

DELETE

[http://localhost:8085/api/v3/attendances/ {empId}](http://localhost:8085/api/v3/attendances/%20%7bempId%7d)

1. Create an Entity: Leave

* Create a LeaveRepositoryinterface and will make use of Spring data JPA

1. Will have findById methode.
2. Will have findAll methode

* Finally, create EmpController will have the following Uri’s

GET

[http://localhost:8085/api/v4/ {empId}](http://localhost:8085/api/v4/%20%7bempId%7d)

<http://localhost:8085/api/v4/leaves>

POST

<http://localhost:8085/api/v4/leave>s

DELETE

[http://localhost:8085/api/v4/ leaves/{empId}](http://localhost:8085/api/v4/%20leaves/%7bempId%7d)

1. Create an Entity: Salary

* Create a SalaryRepository and will make use of Spring data JPA

1. Will have findById methode.
2. Will have fetchSalaryList methode
3. Will have saveSalary methode
4. Will have fetchSalaryById methode
5. Will have deleteSalById methode

* Create SalaryServices service
* Finally, create SalaryController will have the following Uri’s

GET

[http://localhost:8085/api/v7/empid/{empId}](http://localhost:8085/api/v7/empid/%7bempId%7d)

<http://localhost:8085/api/v7/salaries>

http://localhost:8085/api/v7/salaries/{id}

POST

<http://localhost:8085/api/v7/salaries>

[http://localhost:8085/api/v7/](http://localhost:8085/api/v7/%20)editsalaries

DELETE

[http://localhost:8085/api/v7/salaries/{empId}](http://localhost:8085/api/v7/salaries/%7bempId%7d)

1. Create an Entity: Schedule

* Create a ScheduleRepository and will make use of Spring data JPA

1. Will have findById methode.
2. Will have findAll methode

* Finally, create ScheduleController will have the following Uri’s

GET

[http://localhost:8085/ /api/v6/](http://localhost:8085/%20/api/v6/)schedules

[http://localhost:8085/](http://localhost:8085/%20) api/v6/{empId}

POST

[http://localhost:8085/](http://localhost:8085/%20) api/v6/schedules

PUT

[http://localhost:8085/](http://localhost:8085/%20) api/v6/schedules/{empId}

DELETE

[http://localhost:8085/api/v6// schedules/{empId}](http://localhost:8085/api/v6//%20schedules/%7bempId%7d)

1. Create an Entity: Timesheet

* Create a TimesheetRepository and will make use of Spring data JPA

1. Will have findById methode.
2. Will have findAll methode

* Finally, create TimesheetController will have the following Uri’s

GET

<http://localhost:8085/>api/v5/{empId}

[http://localhost:8085/](http://localhost:8085/%20) api/v5/timesheets

[http://localhost:8085/](http://localhost:8085/%20) api/v5/timesheets/{emplId}

POST

[http://localhost:8085/](http://localhost:8085/%20) api/v5/timesheets

PUT

[http://localhost:8085/](http://localhost:8085/%20) api/v5/timesheets/{empId}

DELETE

[http://localhost:8085/api/v5/timesheets/{empId}](http://localhost:8085/api/v5/timesheets/%7bempId%7d)

**6 Assumptions**

●Each user (Admin, Employee) must have a valid user id and password

●Server must be running for the system to function

●Users must log in to the system to access any record.

●Only the Administrator can delete records

**7 General Expectations**

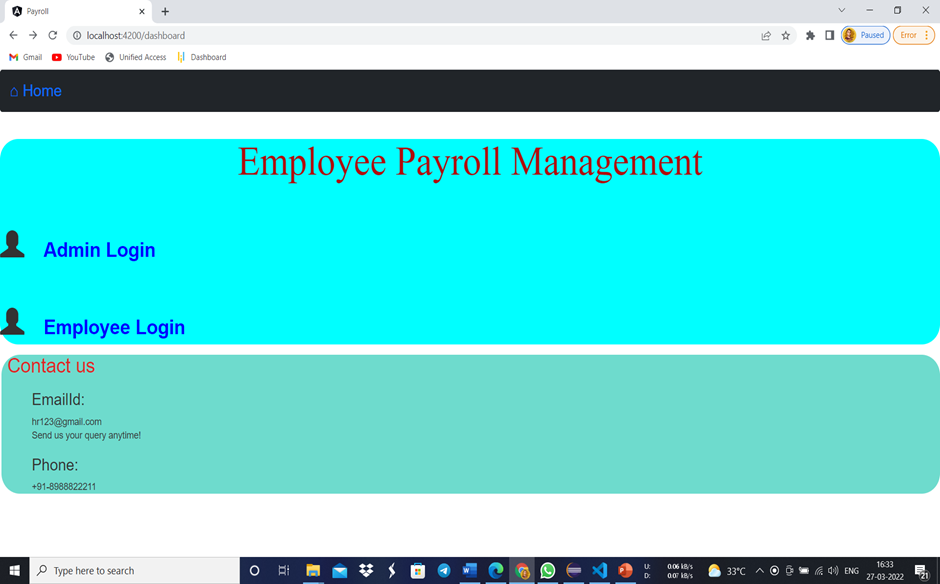
* Participants must create the Class Diagram, Sequence Diagram and ER Diagram.
* Participants must do Unit testing and Functional Testing using POSTMAN tool.
* Integration of Angular and Spring Boot with Microservices should be done, referring Project 2 -Frond End Development Project.
* The server should be a concurrent server servicing multiple clients.
* Database can be implemented using MySQL.
* To begin with, the application should support at least 1 admin.
* Compilation and Build should be done using Eclipse IDE or STS
* Source-code and all documents must be maintained (checked-in) in configuration management system (subversion)
* Coding standards (for Java) should be followed.

**8 Acceptance Criteria**

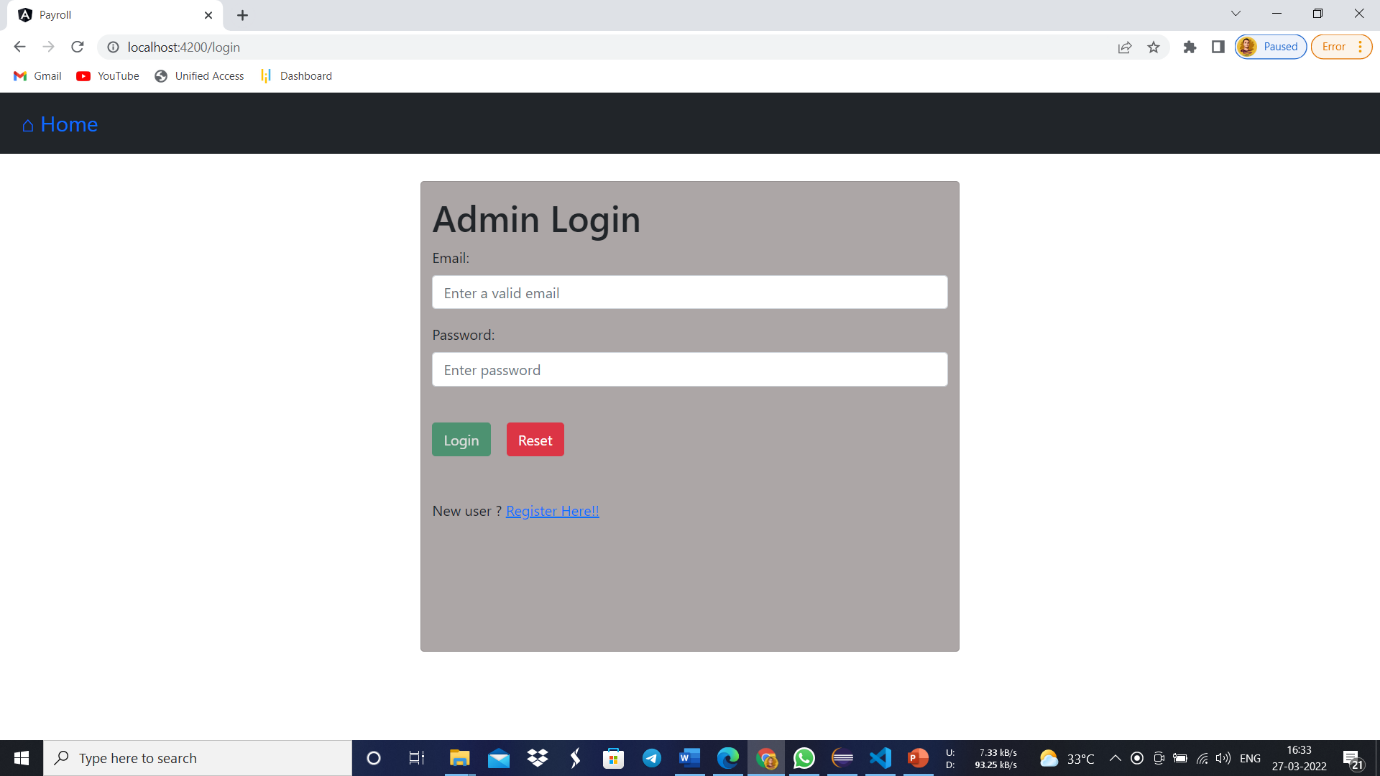
All P1 requirements must be mandatorily implemented.

**9 Output Screenshots**

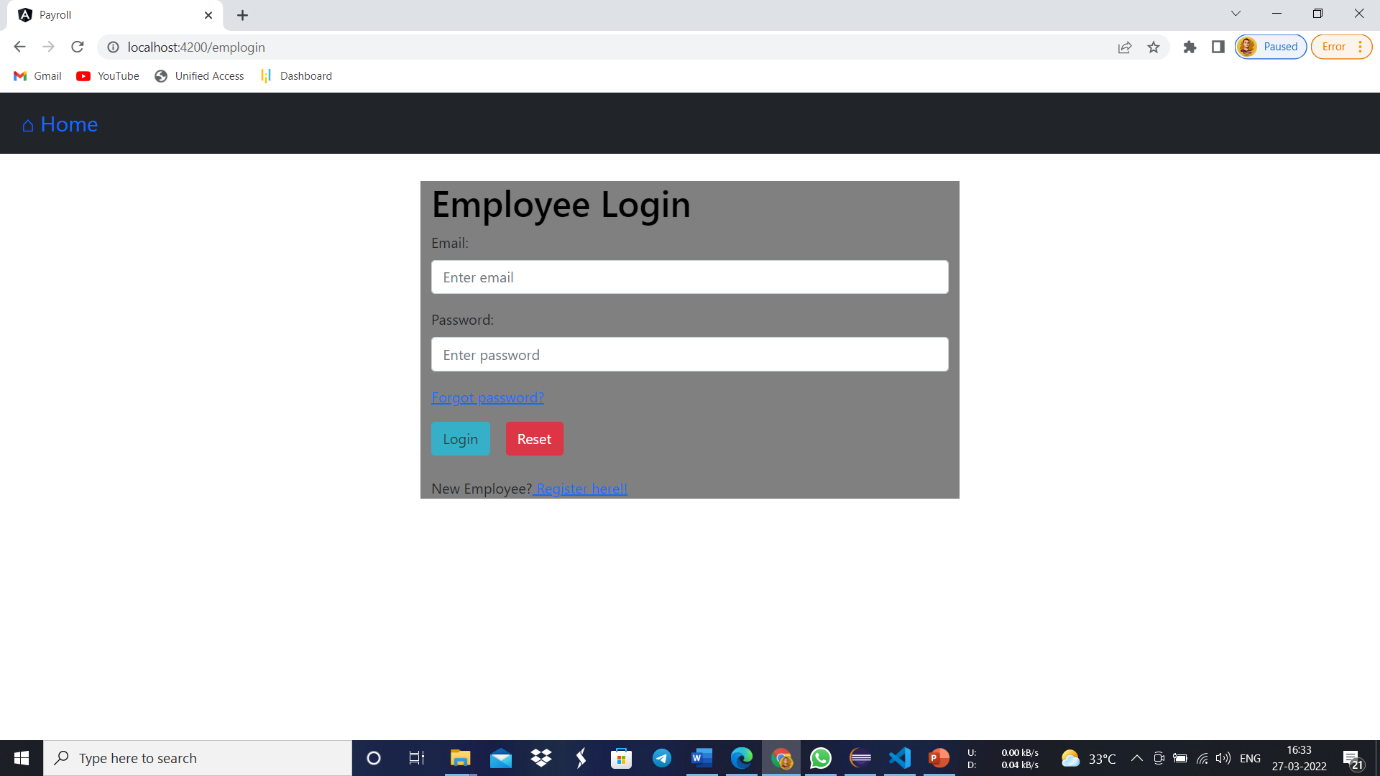
**Homepage:**

****

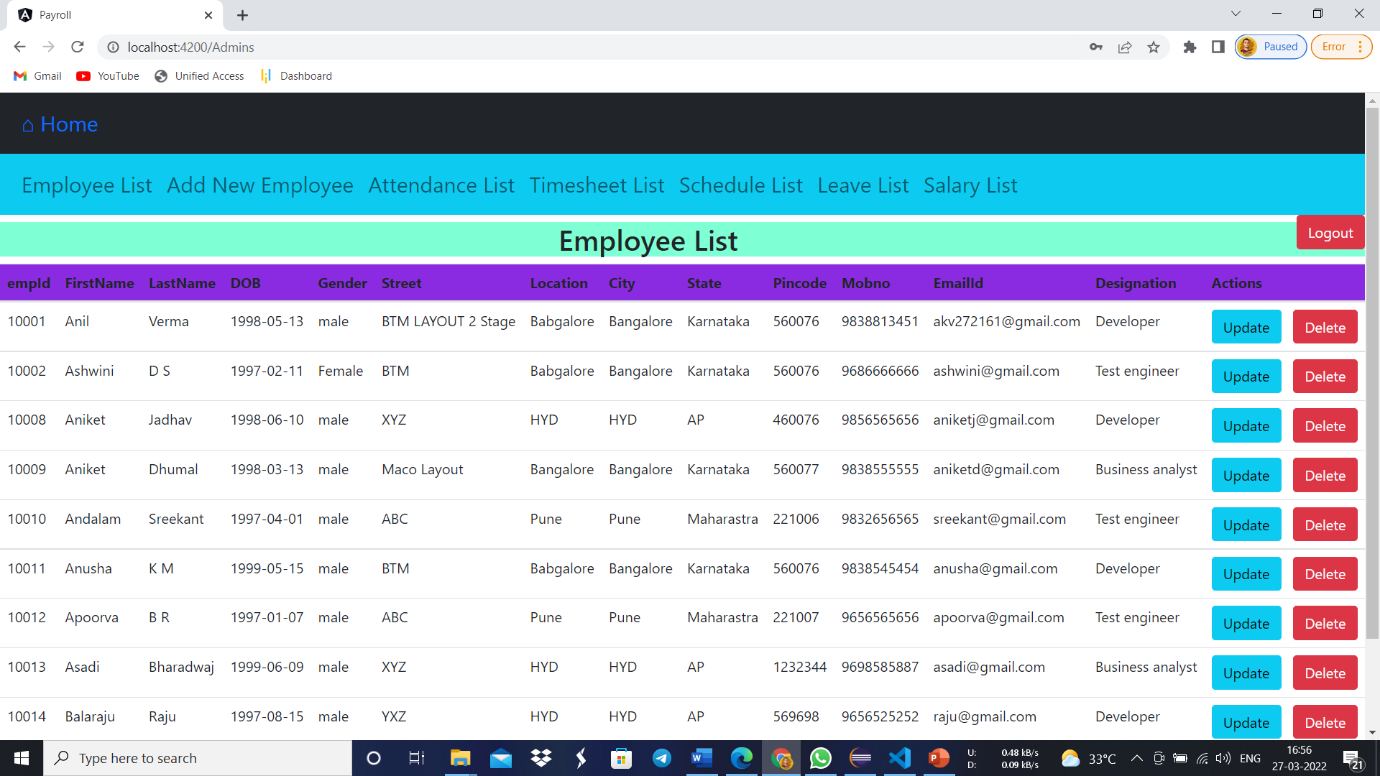
**Admin Login:**

****

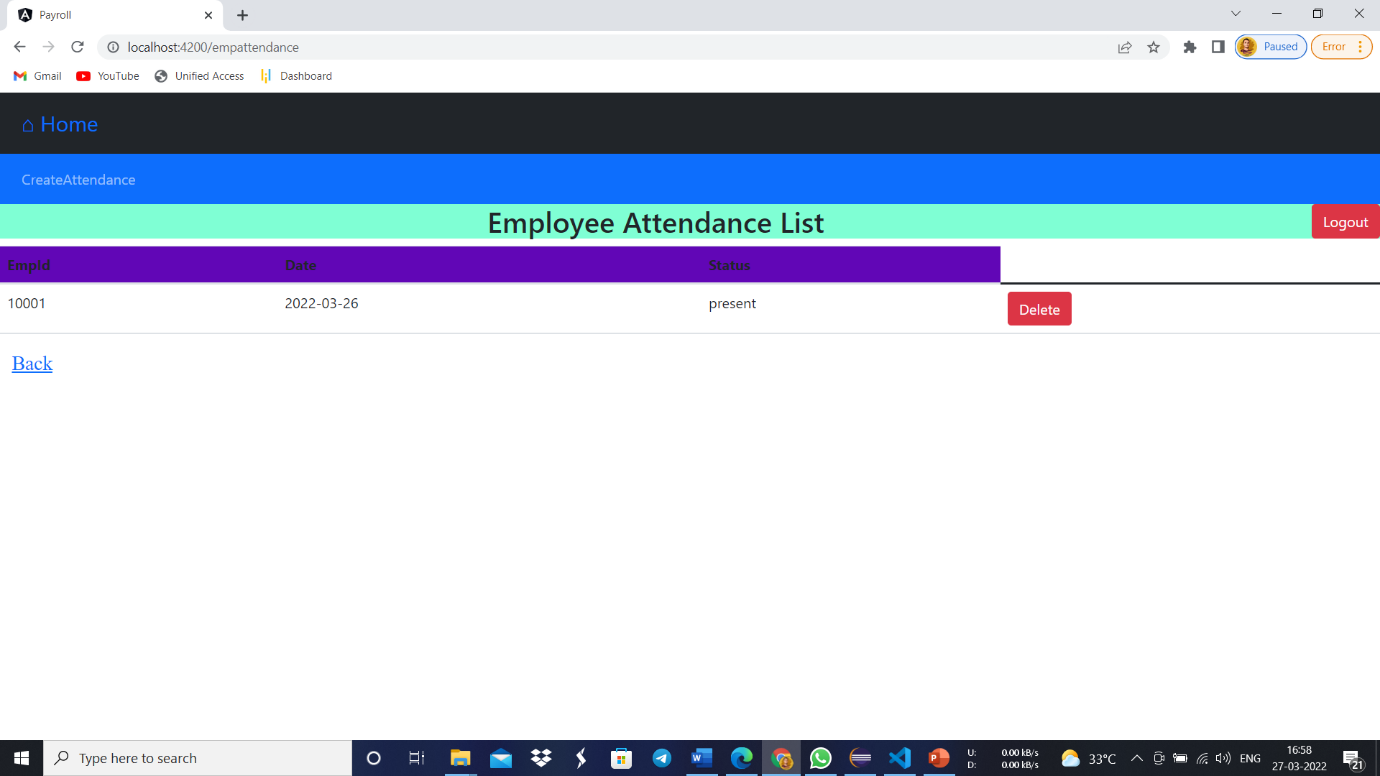
**Employee login:**

****

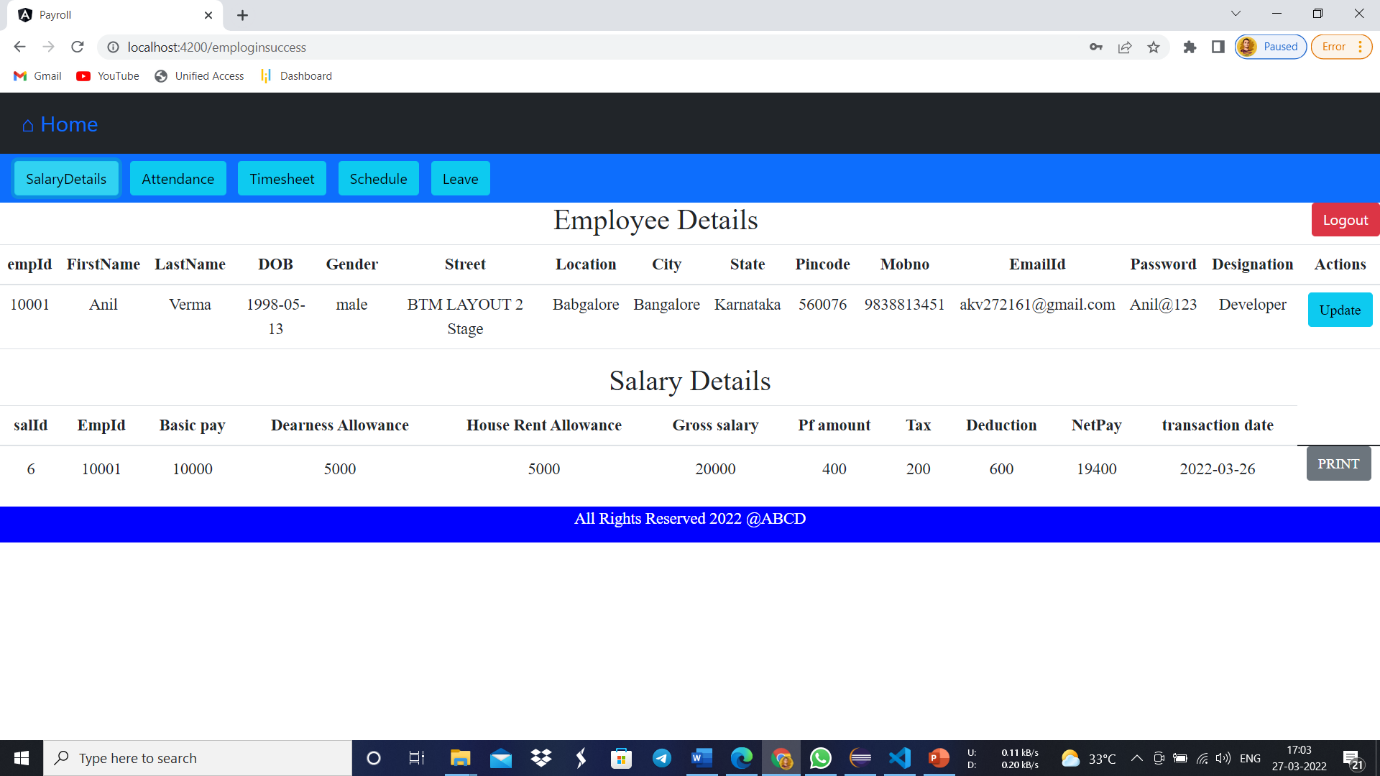
**Employee list:**

****

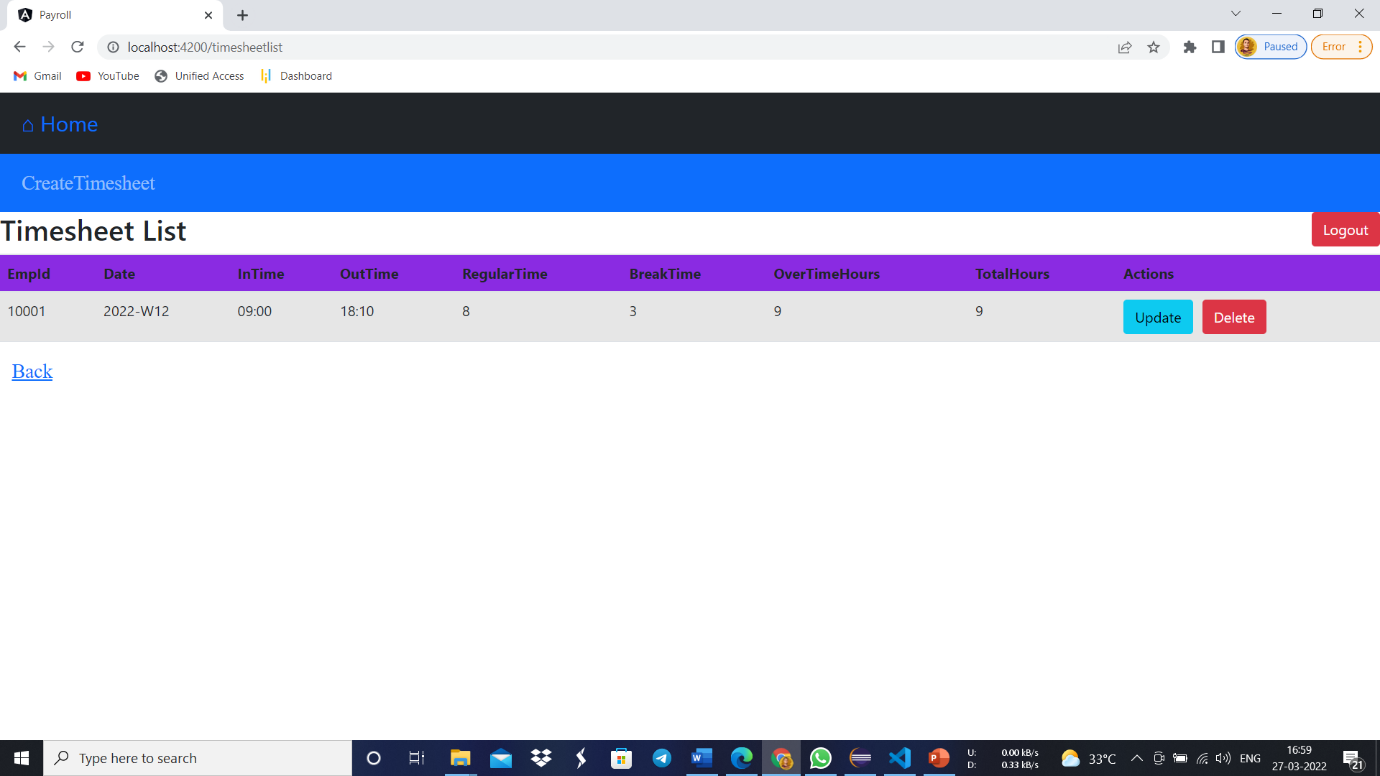
**List of employee attendance:**

****

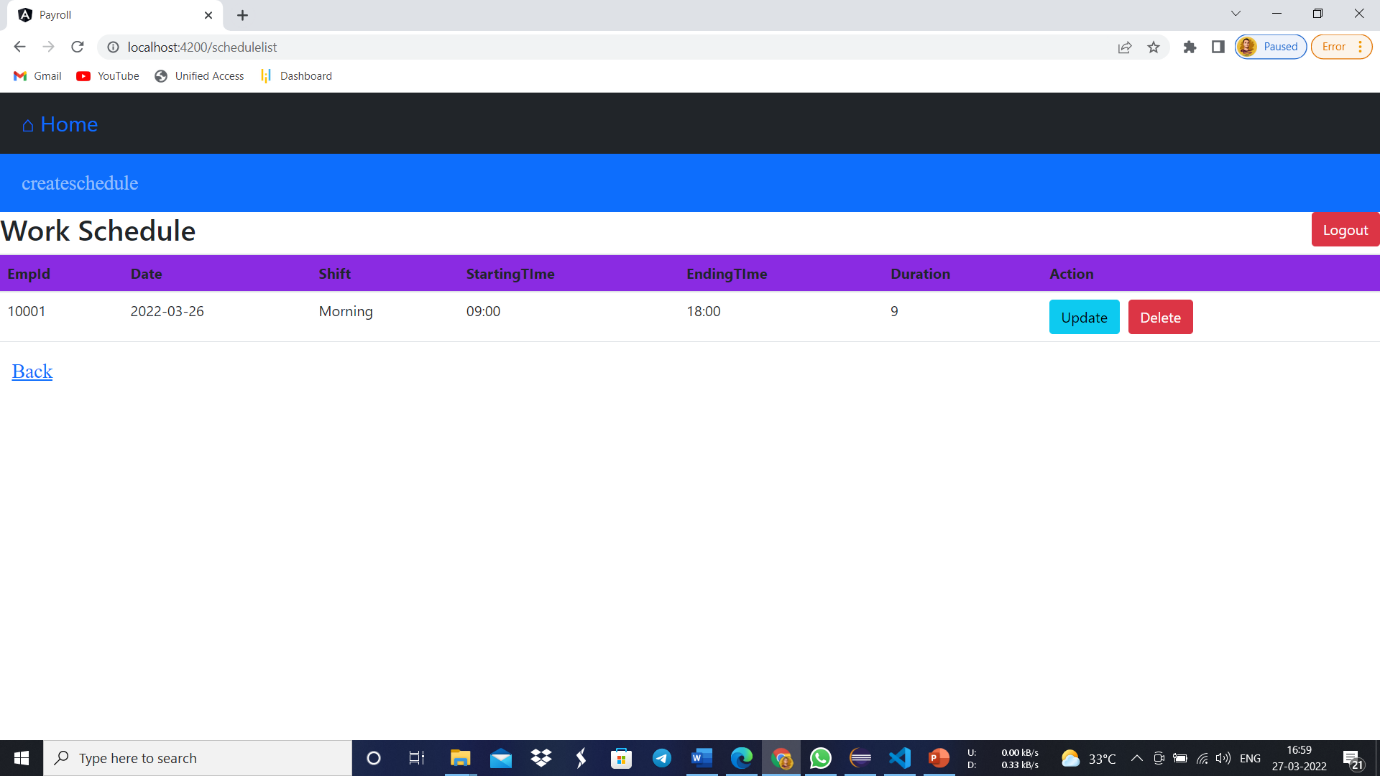
**Employee profile & Pay Slip:**

****

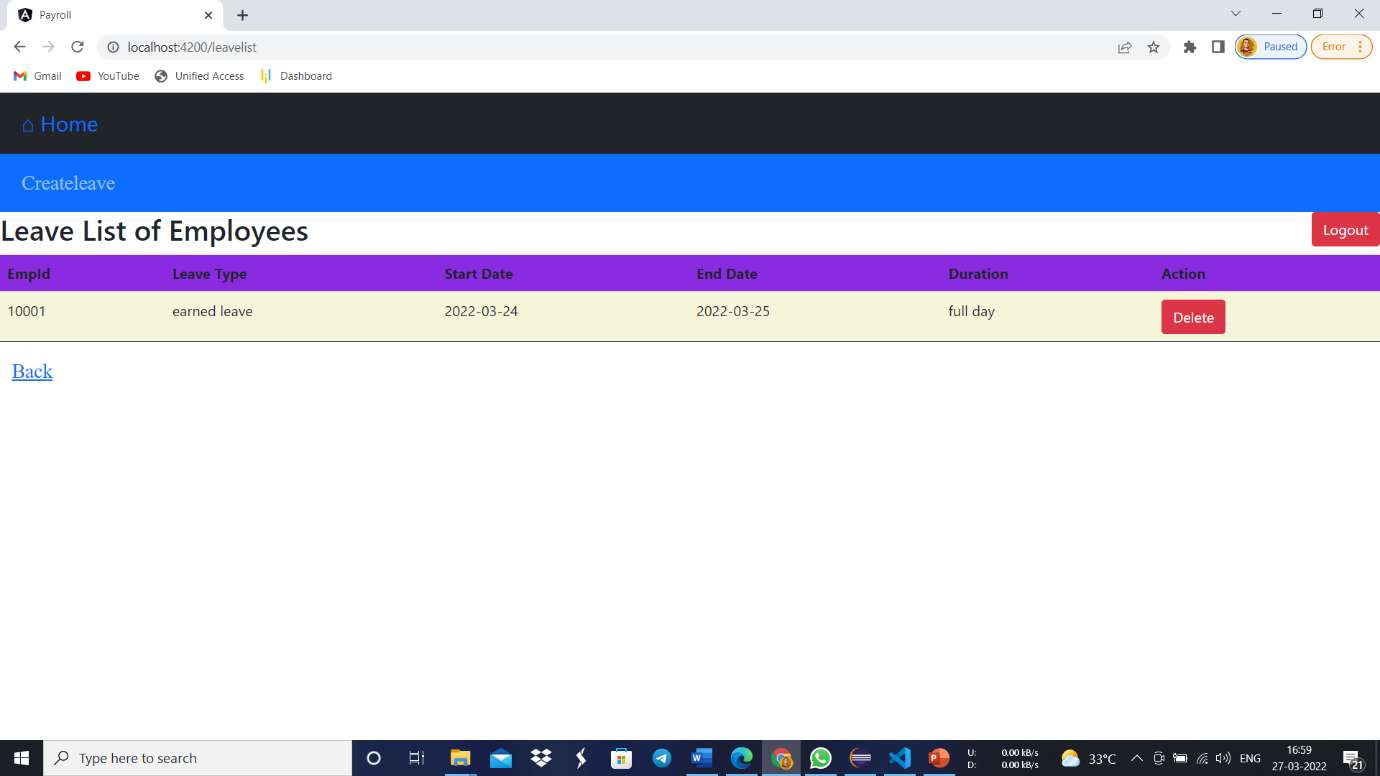
**Timesheet List:**

****

**Work Schedule:**

****

**Leave List:**

****

**Conclusion:**

This project is built keeping in mind that it is to use by admin and the employees in an organization. It is built for use in small scale organization where the number of employees is limited. According to the requested requirement the admin can add, manipulate, update and delete all employee data in his organization. The admin can add new Employee and delete them. The admin can also add predefined pay grades for the employees. The required records can be easily viewed by the admin anytime time he wants in an instant. The payment of the employee is based on monthly basis. Numerous validations implemented would enable the admin to enter accurate data. The main objective of this framework is to save time, make the system cost effective and management records efficiently.