Employee Management System [Report]

Group Members:

20k-0385

20k-1067

20k-0488

ACKNOWLEDGEMENT

The successful presentation of the DBMS PROJECT would be incomplete without the mention of the people who made it possible and whose constant guidance crowned my effort with success.

I would like to extend my gratitude to the MANAGEMENT, Sir Ahmed Buddhi, Miss Eman Shahid for providing all the facilities to present this Project.

I would like to thank all the Teaching Staff and Non-Teaching Staff of the university for their co-operation.

Finally, I extend my heart-felt gratitude to my family for their encouragement and support without which I wouldn’t have come so far. Moreover, I thank all my friends for their invaluable support and cooperation.

ABSTRACT

Employees are the backbone of any company therefore their management plays a major role in deciding the success of the organization. We have prepared the web app for a company named **Cloudways** to manage the employees, department, attendance, projects, meetings scheduled etc. from one platform alone.

Our software will be accessed only by the employees or the admin of the software. All the forms in our application have validation so we do not get inconsistent data in our database and avoid anomalies.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

1.1 OVERVIEW

The employee management system is designed to make it easier for both admin and employee to handle all the operations at one place on one click of a button. The goal is to provide ease to employees in managing their schedule, attendance, training and other things more conveniently so they can concentrate more on their work.

1.2 PROBLEM STATEMENT

Many offices already have employee management systems but they are more intended to be helpful for administration but not the employees working in the office. They have to separately make their schedules for meetings etc. but with our software, they have one platform to handle all these at one place which makes them to not worry about other things but only on their job.

1.3 DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified, and the database schema, which defines the database's logical structure. These three foundational elements help to provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery.

1.4 SQL

SQL is a standard language for storing, manipulating and retrieving data in databases. Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language.

The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control.

1.5 HTML

HTML is a markup language used for structuring and presenting content on the web and the fifth and current major version of the HTML standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications.

Chapter 2

REQUIREMENTS SPECIFICATION

A computerized way of handling information about property and users details is efficient, organized and time saving, compared to a manual way of doing so. This is done through a database driven web application whose requirements are mentioned in this section.

2.1 OVERALL DESCRIPTION

A reliable and scalable database driven web application with security features that is easy to use and maintain is the requisite.

2.2 SPECIFIC REQUIREMENTS

The specific requirements of the Employee Management System are stated as follows:

2.2.1 SOFTWARE REQUIREMENTS

- DE – Visual Studio code
- Web Browser – Firefox 50 or later, Google Chrome – 60 or later
- Database support - MySQL (php myadmin)
- Operating system – Windows 10

2.2.2 HARDWARE REQUIREMENTS

- Processor – Pentium IV or above
- RAM – 2 GB or more
- Hard disk – 3 GB or more
- Monitor – VGA of 1024x768 screen resolution
- Keyboard and Mouse

2.2.3 TECHNOLOGY

- **HTML** is used for the front end design. It provides a means to structure text based information in a document. It allows users to produce web pages that include text, graphics and hyperlinks.

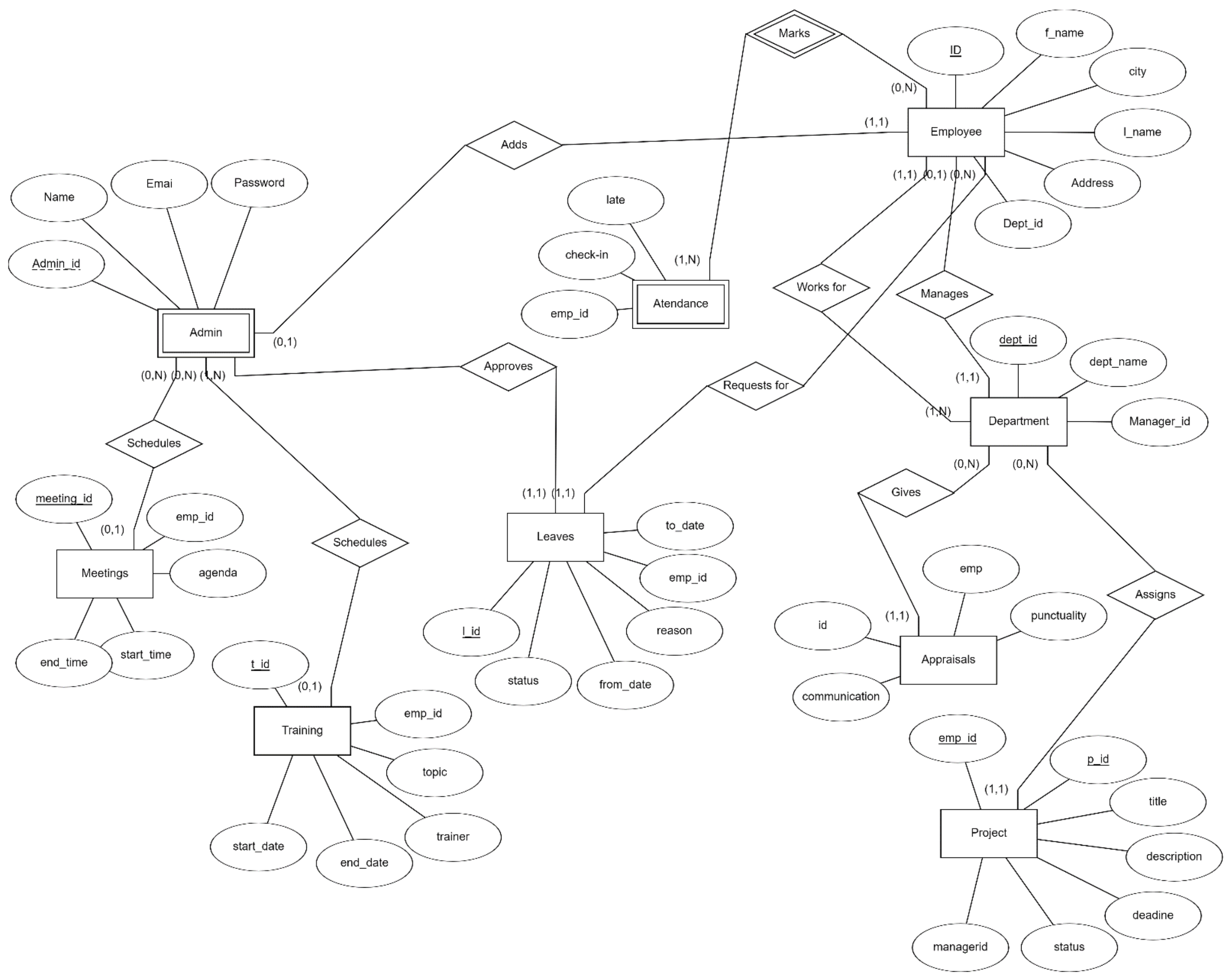
- **CSS** (Cascading Style Sheets) is a style sheet language used for describing the presentation of a document written in a markup language. Although most often used to set the visual style of web pages and user interfaces written in HTML and XHTML, the language can be applied to any XML document.
- **Bootstrap:** Bootstrap is the most popular CSS Framework for developing responsive and mobile-first websites.
- **SQL** is the language used to manipulate relational databases. It is tied closely with the relational model. It is issued for the purpose of data definition and data manipulation.
  Laravel: Laravel is a web application framework with expressive, elegant syntax. Laravel aims to make the development process a pleasing one for the developer without sacrificing application functionality. It is based on the **Model-View-Controller (MVC)** architectural design pattern.

3.2 ENTITY RELATIONSHIP DIAGRAM

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business.

An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities.

Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models.

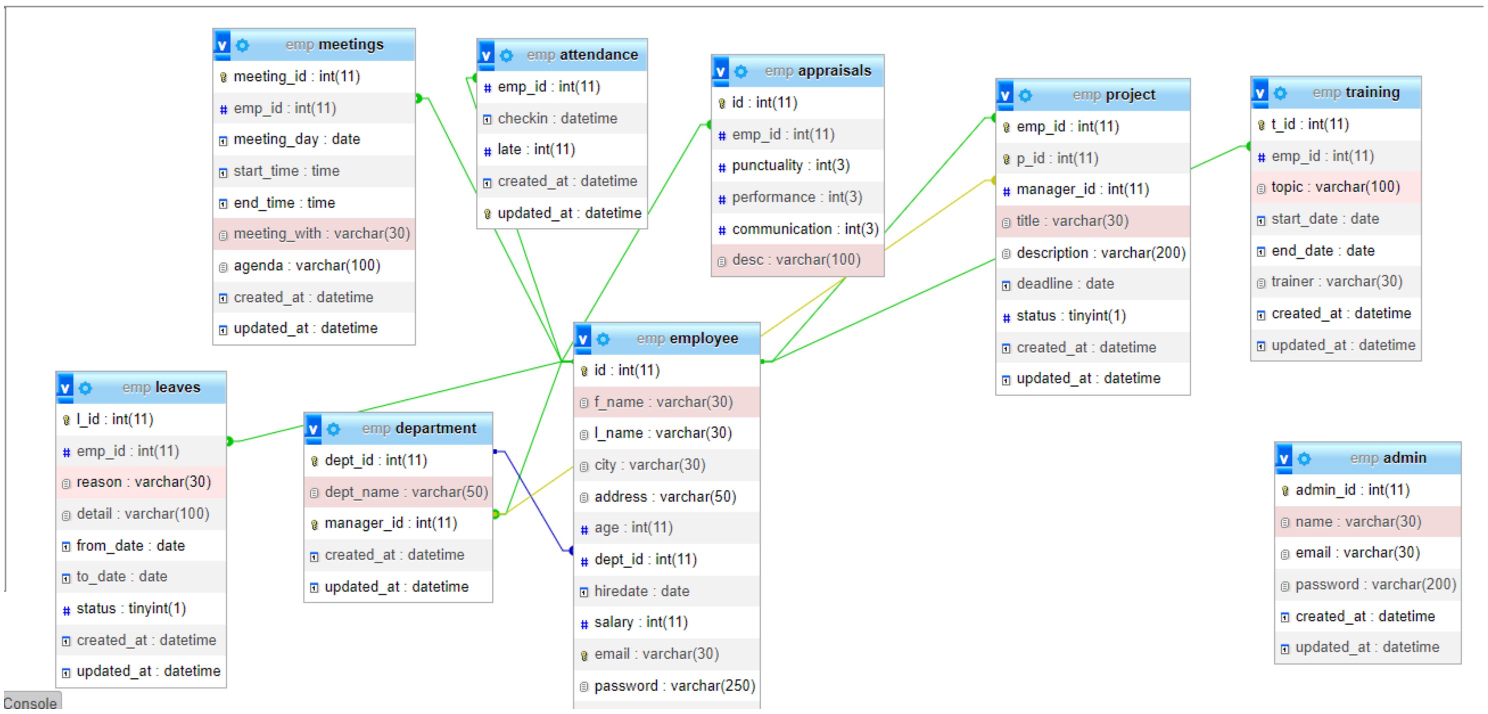

Marks
ID
f_name
city
l_name
Address
Dept_id
Employee
Adds
Name
Emai
Password
Admin_id
Admin
late
check-in
emp_id
Atendance
Works for
Manages
dept_id
dept_name
Department
Manager_id
Requests for
Approves
Schedules
meeting_id
emp_id
Meetings
agenda
end_time
start_time
Schedules
t_id
Training
emp_id
topic
trainer
start_date
end_date
Leaves
to_date
emp_id
reason
l_id
status
from_date
Gives
Assigns
emp
id
punctuality
Appraisals
communication
emp_id
p_id
title
Project
description
deadine
managerid
status
(0,N)
(1,1)
(1,1)
(0,1)
(0,N)
(1,N)
(0,1)
(0,N)
(0,N)
(1,N)
(1,1)
(1,N)
(1,1)
(1,1)
(0,1)
(0,1)
(0,N)
(0,N)
(1,1)
(1,1)

3.3 RELATIONAL SCHEMA

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the

database. When a primary key is referenced in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute.

A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.



3.4 DESCRIPTION OF TABLES

The database consists of 9 tables:

1. Employee It stores the employee details.

- id: Unique user id done by auto increment.
- f_name
- l_name
- Email
- Password
- Address
- City
- Age
- Dept_id
- salary

2. Admin: It stores the details of the Admin.

- Admin_id: Unique admin id done by auto increment.
- Name
- Email
- Password

3. Appraisals : Stores the reviews given by manager to its employees

- Id
- Emp_id
- Punctuality
- Communication
- Desc
- Performance

4. Attendance: It stores attendance of employees on daily basis

- Emp_id
- Checkin
- Late

5. Department:departments in the office

- Dept_id
- Dept_name
- Manager_id

6. Leaves: Details of leaves applied by empoyees and approved by admin

- L_id
- Emp_id
- Reason
- Details
- From_date
- To_date
- status

7. Meetings: meetings scheduled by admin

- Meeting_id
- Emp_id
- Meetingday
- Starttime
- Endtime
- Meetingwith
- agenda

8. Projects: Projects assigned by manager

- Em_id
- P_id (composite primary key with emp_id)
- Manager_id
- Title
- Description
- Deadline
- Status

9. Training: trainings scheduled by admin

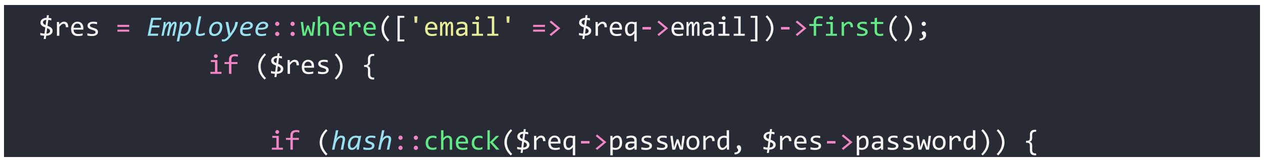
- T_id
- Emp_id
- Topic
- Start_date
- End_date
- Trainer

10.Deleted_employees: With trigger, the deleted employee is stored in this table

IMPLEMENTATION

4.1 MODULES AND THEIR ROLES

4.1.1 Login: Login for the employee.



```
 $res = Employee::where(['email' => $req->email])->first();
            if ($res) {

                if (hash::check($req->password, $res->password)) {
                     $req->session()->put('Emp_Login', True);
                     $req->session()->put('Emp_Id', $res->id);
                     $req->session()->put('Emp_Name', $res->f_name);
                     $check_man=Employee::join('department','department.manager_i
d','=','employee.id')
                     ->where('department.manager_id',$res->id)->first();

                     if($check_man){
                        //the person is manager
                        $req->session()->put('Manager', True);
                        $req->session()->put('DeptId', $check_man->dept_id);
                     }
                      return redirect()->route('emp_home');
                 }
                 else {
                     return redirect('/emp_login')->with('myerror','Invalid login
details');
                 }
             }
             else{
                return redirect('/emp_login')->with('myerror','Invalid login
details');
             }
```

4.2 Update Employee details

```
  $validate = $req->validate([
        'f_name' => 'required|max:30',
         'l_name' => 'required|max:30',
         'city' => 'required|max:30',
         'address'=>'required|max:50',
        'age'=>'required|numeric|min:18',
        ]);
   $updated=Employee::find($id);
   $updated->f_name=$req->f_name;
   $updated->l_name=$req->l_name;
   $updated->city=$req->city;
   $updated->address=$req->address;
   $updated->age=$req->age;
   $updated->save();
   return redirect('emp_home');
```

4.2 Update Password

```
$validate = $req->validate([
        'currentpass' => 'required|max:30',
         'updatedpass' => 'required|max:30'
        ]);
   $updated=Employee::find($id);
   if (hash::check($req->currentpass, $updated->password) )
   {
    $pass=Hash::make($req->updatedpass);
    $updated->password=$pass;
    $updated->save();
```

4.3 Add new employee (by admin)

```
  $validate = $req->validate([
       'f_name' => 'required|max:30',
        'l_name' => 'required|max:30',
        'city' => 'required|max:30',
       'age'=>'required|numeric|min:18',
        'email'=>'required|max:30|unique:employee',
        'password' =>'required|max:20',
        'dept_id'=> 'required|numeric',
        'hiredate'=>'required',
        'salary'=>'required|numeric|min:0'
       ]);
         $depart= Department::where('dept_id',$req->dept_id)->first();


        $emp=new Employee;
        $emp->f_name=$req->f_name;
        $emp->l_name=$req->l_name;
        $emp->city=$req->city;
        $emp->age=$req->age;
        $emp->email=$req->email;
        $emp->dept_id=$req->dept_id;
        $emp->hiredate=$req->hiredate;
        $emp->salary=$req->salary;
        $pass=Hash::make($req->password);
        $emp->password=$pass;
         $emp->save();
```

4.4 Add new department (by admin)

```
 $validate = $req->validate([
     'dept_name' => 'required|max:30|unique:department|regex:/^[a-zA-Z ]+$/',
     'manager_id'=>'unique:department',
     ]);

   $new=new Department;
   $new->dept_name=$req->dept_name;
   $new->manager_id=$req->manager_id;
```

4.5 Delete department

```
  public function delete_dept($id){
   $dept=Department::where('dept_id',$id)->delete();
   return redirect('/admin_home')->with('pass2','Department deleted');
  }
```

4.6 Approve Leaves (by admin)

```
  public function approve_leaves(){

  $emps=Employee::join('leaves','leaves.emp_id','employee.id')-
>where('leaves.status',0)->orderby('leaves.from_date')->get();
 return view('admin_approve_leaves',['emps'=>$emps]);
  }
  public function approve_leave($l_id){
   $user = DB::table('leaves')->where('l_id',$l_id)->limit(1)-
>update(['status'=>1] );
   return redirect('/admin_home')->with('pass2','Leave Approved');

  }
```

4.7 Filter salaries of employee by department (by admin)

```
  public function filter_sal_dept(Request $req){
   $validate = $req->validate([
     'dept_id' => 'required',
     ]);
     $emps=Employee::join('department','department.dept_id','employee.dept_id')
     ->where('employee.dept_id',$req->dept_id)->get();
     return view('admin_filter_sal',['emps'=>$emps]);
```

4.8 Filter number of employees in each department (by admin)

```
 public function filter_emp_by_dept(Request $req){
    $validate = $req->validate([
      'dept_id' => 'required|numeric',
      ]);
      $check=Department::where('dept_id',$req->dept_id)->first();
      if(!$check){
        return redirect()->back()->with('myerror','Pease enter correct Department 
ID');
      }
      $emps=DB::table('employee')
      ->select('employee.dept_id','department.dept_name', DB::raw('count(*) as 
total'))->join('department','department.dept_id','employee.dept_id')
      ->groupBy(['employee.dept_id','department.dept_name'])-
>having('employee.dept_id',$req->dept_id)->first();
      if(!$emps){
        return redirect()->back()->with('myerror','No Employees are currently in 
this Department');
      }
      return view('filter_emp_by_dept',['emps'=>$emps]);
     }
```

4.9 Apply for Leave(by employee)

```
public function apply_leave_logic(Request $req){

    $validate = $req->validate([
        'reason' => 'required|max:30',
         'detail' => 'required|max:100',
         'from_date'=>'required',
         'to_date'=>'required',
        ]);

       $leave=new Leaves;
       $leave->emp_id=session('Emp_Id');
       $leave->reason=$req->reason;
       $leave->detail=$req->detail;
       $leave->from_date=$req->from_date;
       $leave->to_date=$req->to_date;
       $leave->status=0;
       $leave->save();
       return redirect('/emp_home')->with('pass2','Applied for leave
successfully');
   }
```

4.10 Give appraisal (by employee)

```
    public function emp_app(Request $req){
        $validate = $req->validate([
            'punctuality' => 'required|numeric|min:0|max:10',
          'performance' => 'required|numeric|min:0|max:10',
          'communication'=>'required|numeric|min:0|max:10',
           'comments' => 'required|max:100',
             ]);

        DB::table('appraisals')->insert([
        'emp_id' => $req->emp_id,
        'punctuality' => $req->punctuality,
        'performance' => $req->performance,
        'communication' => $req->communication,
        'desc' => $req->comments
    ]);
    return redirect('/emp_home')->with('pass2','Appraisal added succesfully');

    }
```

4.11 Delete appraisal(by admin)

```
  public function delete_review($id){
        DB::table('appraisals')->where('ID',$id)->delete();
        return redirect('/admin_home')->with('pass2','Review Removed');
      }
```

4.12 Schedule Meetings (by admin)

```
 public function meet_schedule_form(Request $req){
        $validate = $req->validate([
            'emp_id' => 'required',
             'meeting_day' => 'required',
             'start_time' => 'required',
            'end_time'=>'required',
             'meeting_with'=>'required|max:30',
             'agenda' =>'required|max:100',
            ]);
            $meeting=new Meeting;
            $meeting->emp_id=$req->emp_id;
            $meeting->meeting_day=$req->meeting_day;
            $meeting->start_time=$req->start_time;
            $meeting->end_time=$req->end_time;
            if($meeting->end_time<$meeting->start_time){
                return redirect('/admin_home')->with('myerror','Could not
scheduled Meeting');
            }
            $meeting->meeting_with=$req->meeting_with;
            $meeting->agenda=$req->agenda;
            $meeting->save();
            return redirect('/admin_home')->with('pass2','Meeting Sceduled');

    }
```

4.13 Check meetings scheduled

```
    public function meetings_scheduled(){
        //only show meetins sceduled in future and not the nes already done
         $emps=Meeting::join('employee','employee.id','meetings.emp_id')
         ->where('meetings.meeting_day','>=',date('Y-m-d'))-
>orderby('meetings.meeting_day')->get();
        return view('meetings_scheduled',['emps'=>$emps]);
    }
```

4.14 Assign Project

```php
 public function assign_logic(Request $req){
       $validate = $req->validate([
           'emp_id' => 'required',
         'p_id' => 'required',
         'title'=>'required|regex:/^[a-zA-Z ]+$/|max:30',
          'description'=>'required|max:200',
          'deadline'=>'required',
            ]);
       try{
           $pro=new Project;
           $pro->emp_id=$req->emp_id;
           $pro->p_id=$req->p_id;
           $pro->manager_id=session('Emp_Id');
           $pro->title=$req->title;
           $pro->description=$req->description;
           $pro->deadline=$req->deadline;
           $pro->status=0;
           $pro->save();
       }catch(\Illuminate\Database\QueryException $e){
           if($e->getCode() == "23000"){ //23000 is sql code for integrity
constraint violation

               return redirect('/emp_home')->with('myerror','Couldnt Assigned
Project.Please try again');
           }

          }
```

4.15 Check Pending Projects

```php
   public function my_pending_projects($id){
       $projects= Project::where('emp_id',$id)->where('project.status',0)-
>where('deadline','>=',date('Y-m-d'))
       ->get();
       return view('my_pending_projects',['projects'=>$projects]);
    }
```

4.16 Submit Project

```
    public function submit_project($pid){

        $user = DB::table('project')->where('emp_id',session('Emp_Id'))-
>where('p_id',$pid)->limit(1)->update(['status'=>1] );
        return redirect('/emp_home')->with('pass2','Project Submited');

    }
```

4.17 Schedule training

```
        public function train_schedule_form(Request $req){
    $validate = $req->validate([
                'emp_id' => 'required',
                 'topic' => 'required|max:100',
                 'start_date' => 'required',
                'end_date'=>'required',
                 'trainer'=>'required|max:30',
                ]);
                $training=new Training;
                $training->emp_id=$req->emp_id;
                $training->topic=$req->topic;
                $training->start_date=$req->start_date;
                $training->end_date=$req->end_date;
                if($training->end_date<$training->start_date){
                    return redirect('/admin_home')->with('myerror','Could not
scheduled Training');
                }

                $training->trainer=$req->trainer;

                $training->save();
                return redirect('/admin_home')->with('pass2','Training
Sceduled');
        }
```

4.18 Mark Attendance

```
   public function mark_attendance(Request $req, $id){
   $flag=0;
           $time= date("Y-m-d H:i:s");
           $att=new Attendance;
           $att->emp_id=$id;
           $att->checkin=$time;
           $marked = new DateTime($att->checkin);
         $checkin_time = $marked->format('H:i:s');
         if($checkin_time>'09:00:00'){
           $att->late=1;
           $flag=1;
         }
         else{
           $att->late=0;
         }
           $att->save();
           if($flag==1){
               return redirect('emp_home')->with('marked','Atendance maked. you 
are late today');;
           }
          else{
               return redirect('emp_home')->with('marked','Atendance maked.');;
           }

   }
```

4.19 Filter Attendance

```
public function admin_filter_attendance( Request $req){
    $validate = $req->validate([
        'from' => 'required',
         'to' => 'required',
         'emp_id' => 'required|max:30|min:2',
        ]);
        $emp_fname=Employee::where('f_name','like','%'.$req->emp_id.'%')->get();
         if($emp_fname->count()==0){
            return redirect()->back()->with('myerror','No Employee found');
         }
        if($req->to < $req->from){
            return redirect()->back()->with('myerror','please enter correct range 
of dates');
        }

      $emps=Attendance::join('employee','employee.id','attendance.emp_id')->
      where('employee.f_name','like','%'.$req->emp_id.'%')-
>wherebetween('attendance.checkin',[$req->from.' 00:00":00',$req->to.' 
23:59:59'])->paginate(20);
      return view('admin_filter_attendance',['emps'=>$emps]);
```

4.20 Check if attendance is already marked for today

```
public function attendance($id){
    $status=Attendance::where('emp_id','=',$id)->get();
   if($status){
              foreach($status as $sta){
           $marked = new DateTime($sta->checkin);
          $date = $marked->format('Y-m-d');
            if($date==date('Y-m-d')){
              return redirect('/emp_home')->with('myerror','you have already
checked in today');
              }
              }
              $emp=Employee::find($id);
              return view('emp_attendance',['emp'=>$emp]);
        }
        else{
       $emp=Employee::find($id);
       return view('emp_attendance',['emp'=>$emp]);
   }

   }
```

RESULT

The resulting system is able to:

Authenticate employee/Admin credentials during login.

Allows admin to add, update, delete new employee

Allows admin to add new department

Allows admin to schedule training and meetings

Allows admin to check attendance specific employees from a given range of dates

Allows employee to update password or personal details

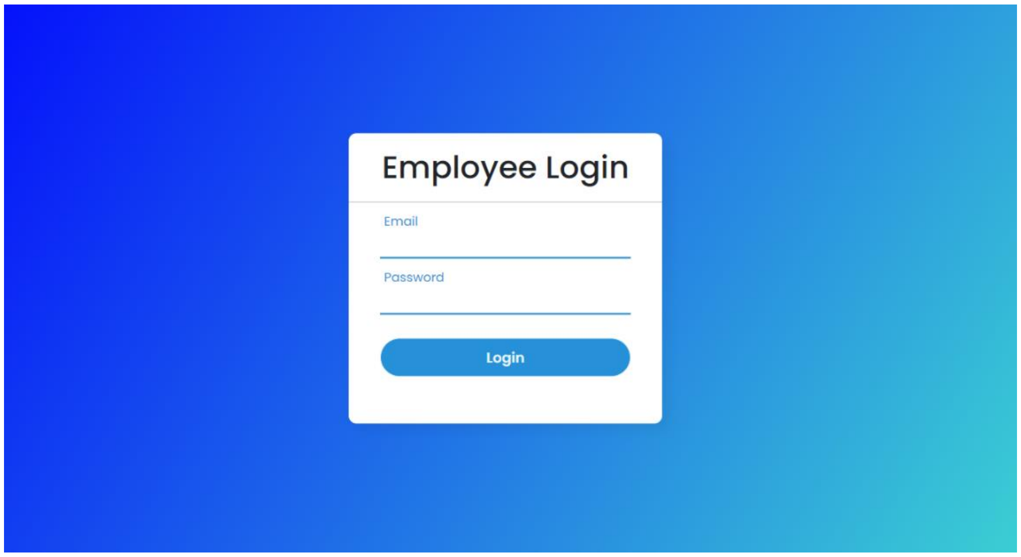
Allows to employee to check-in once daily

Allows manager to assign project

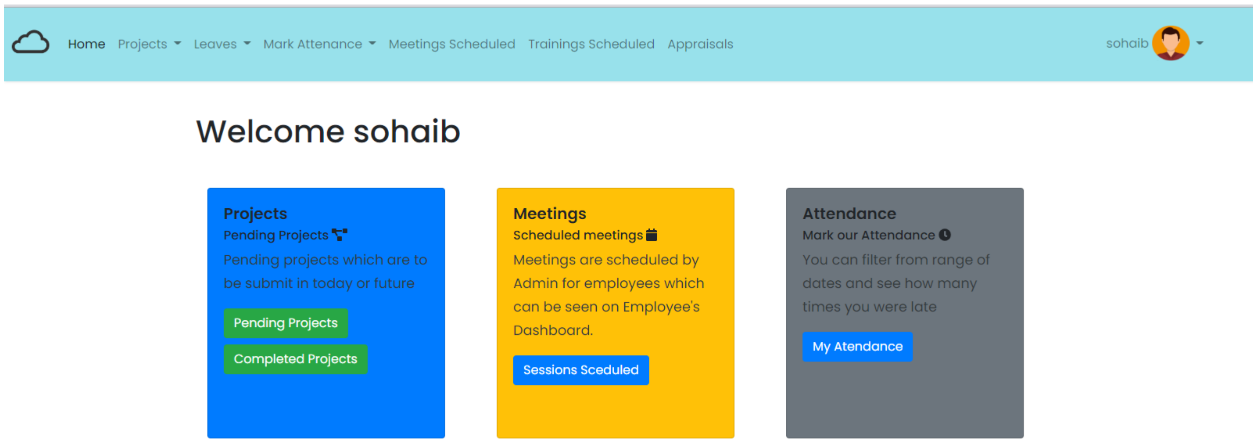
Allows employees to see ow many times they have been late.

SNAPSHOTS

Employee Login page



Employee Dashboard

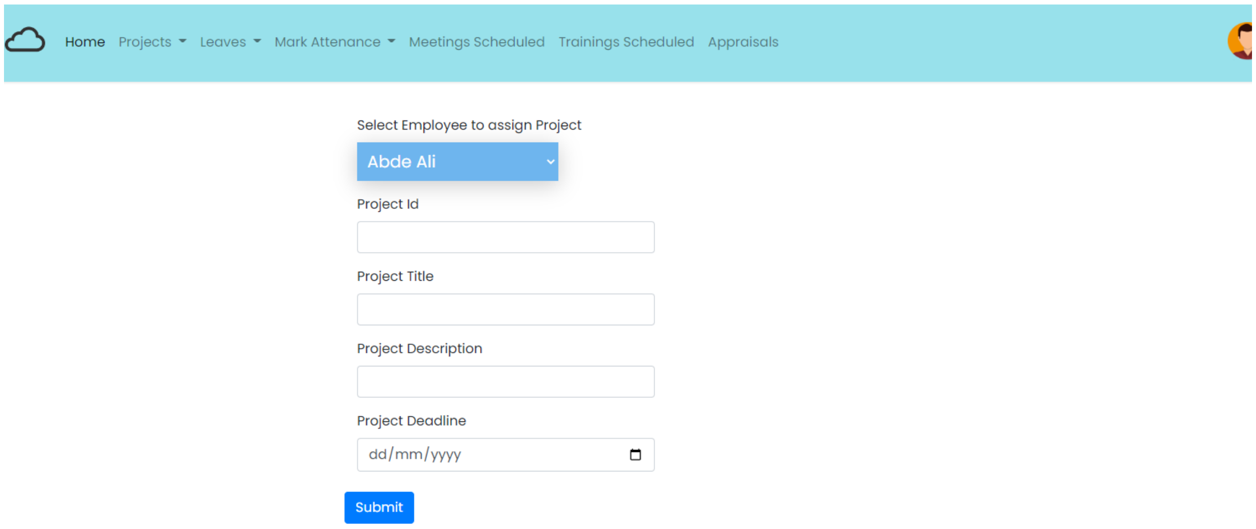


Pending projects which are to be submit in today or future

Meetings are scheduled by Admin for employees which can be seen on Employee's Dashboard.

You can filter from range of dates and see how many times you were late

Assign Project



Select Employee to assign Project

Abde Ali

Project Id

Project Title

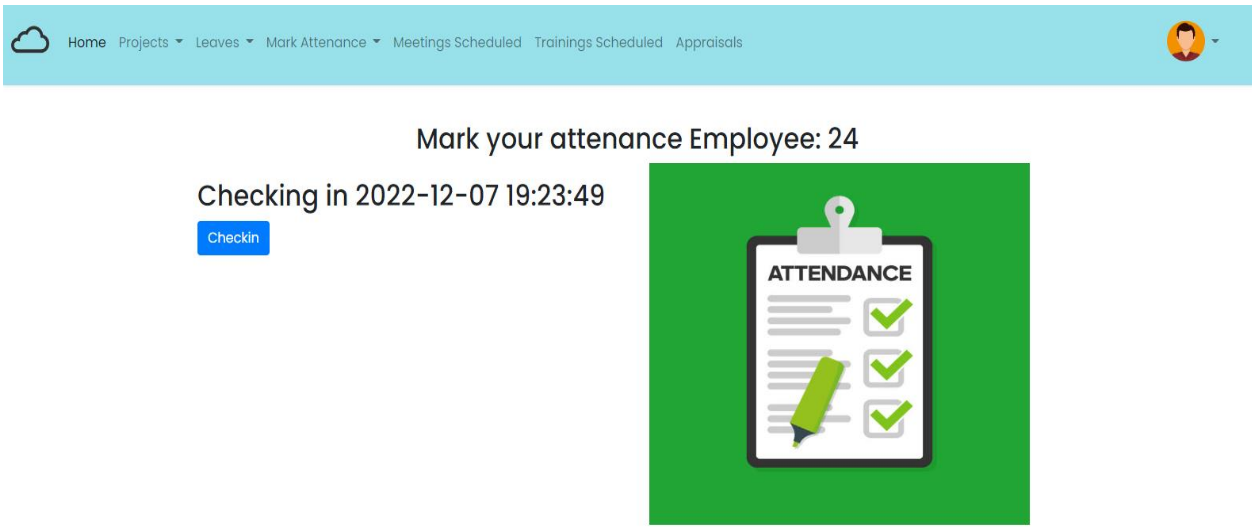
Project Description

Project Deadline

dd/mm/yyyy

Submit

Mark Attendance

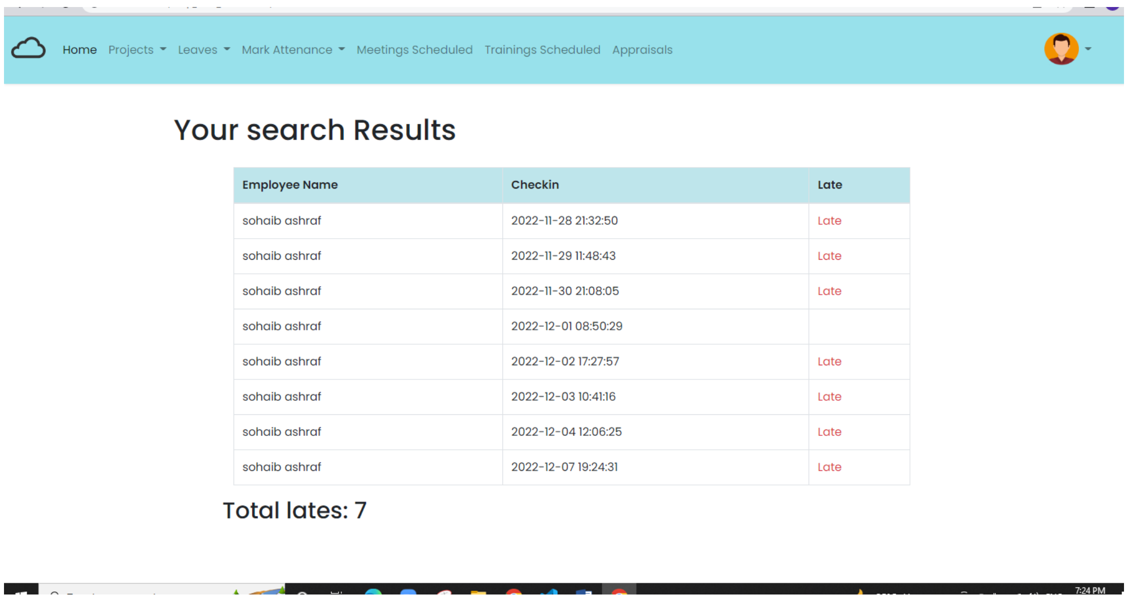


Mark your attenance Employee: 24

Checking in 2022-12-07 19:23:49

Checkin

Check/filter Attendance

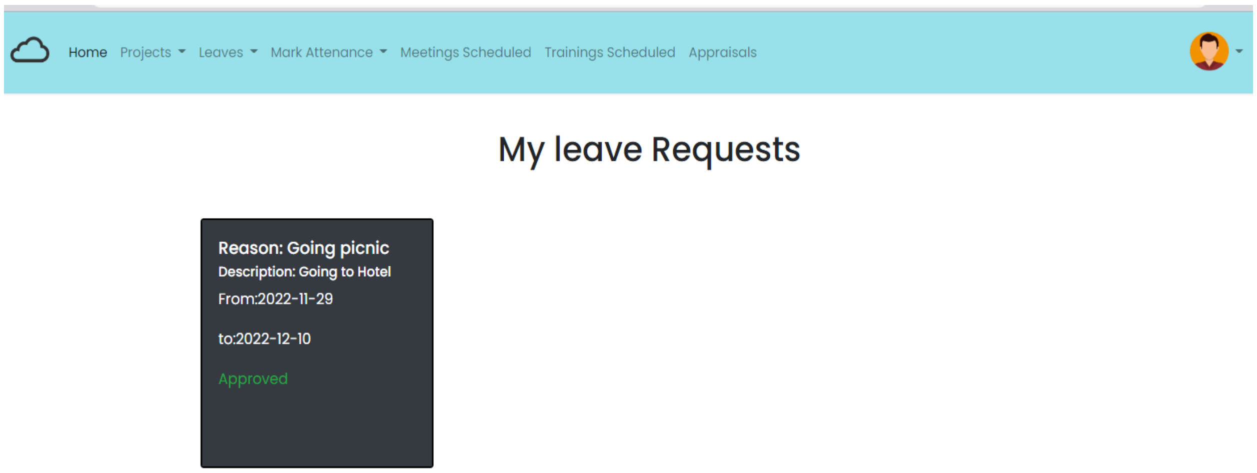


Your search Results

Employee Name	Checkin	Late
sohaib ashraf	2022-11-28 21:32:50	Late
sohaib ashraf	2022-11-29 11:48:43	Late
sohaib ashraf	2022-11-30 21:08:05	Late
sohaib ashraf	2022-12-01 08:50:29	
sohaib ashraf	2022-12-02 17:27:57	Late
sohaib ashraf	2022-12-03 10:41:16	Late
sohaib ashraf	2022-12-04 12:06:25	Late
sohaib ashraf	2022-12-07 19:24:31	Late

Total lates: 7

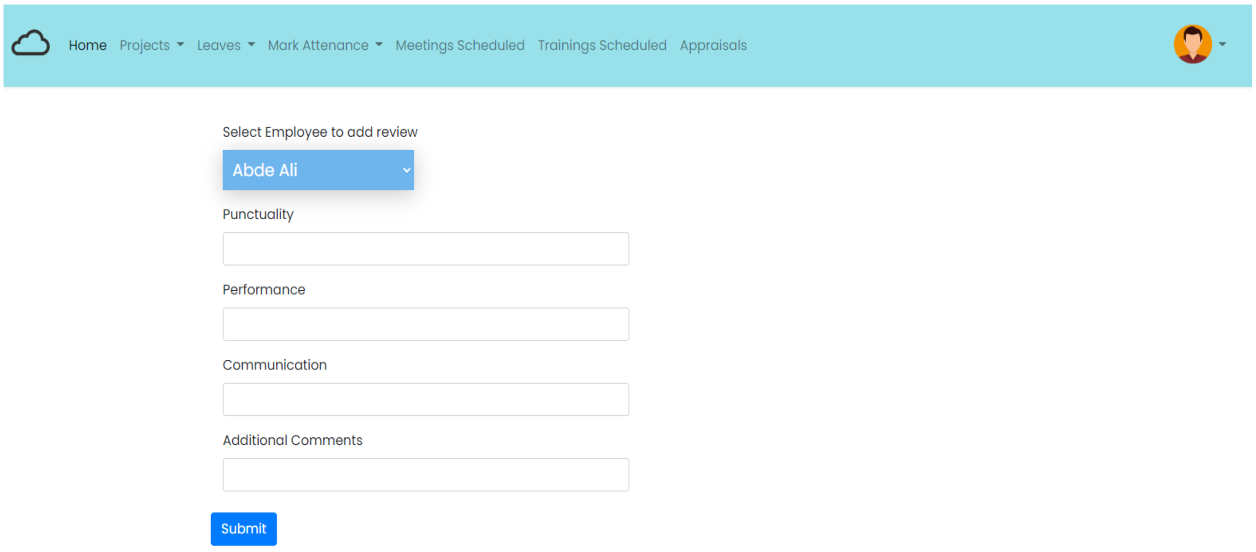
Check leave Status



My leave Requests

Reason: Going picnic
Description: Going to Hotel
From:2022-11-29
to:2022-12-10
Approved

Give Appraisal



Select Employee to add review

Abde Ali

Punctuality

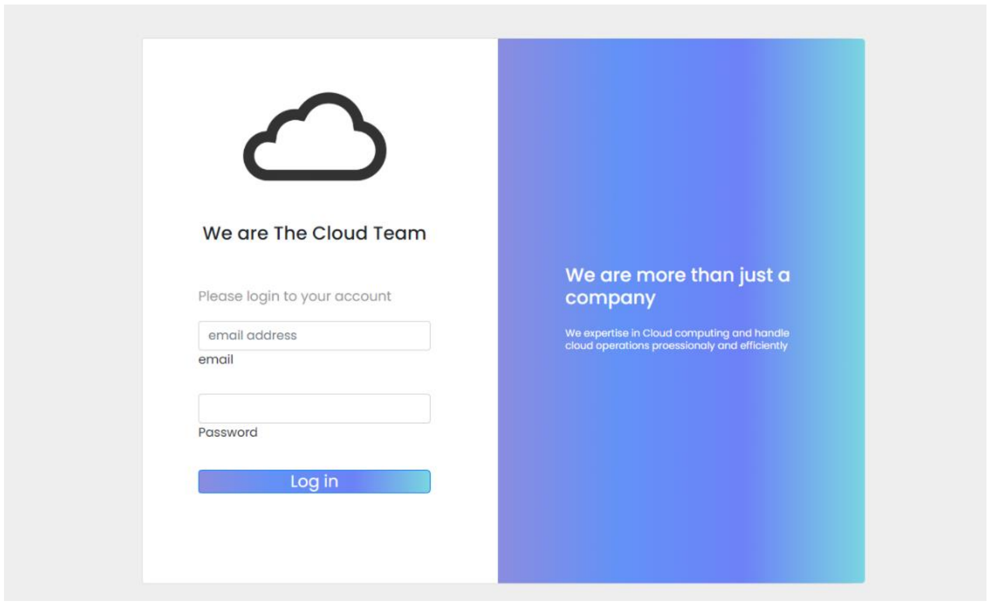
Performance

Communication

Additional Comments

Submit

Admin Login



We are The Cloud Team

Please login to your account

email address

email

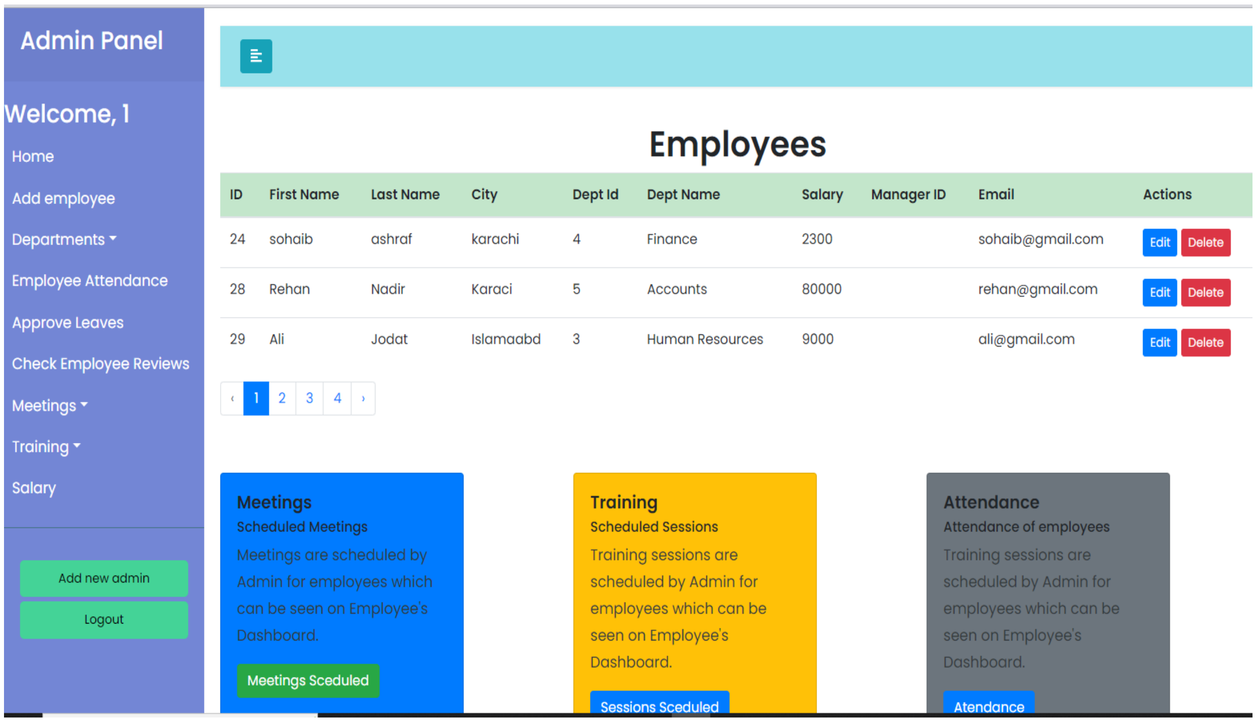
Password

Log in

We are more than just a company

We expertise in Cloud computing and handle cloud operations proessionaly and efficiently

Admin Dashboard



Admin Panel

Welcome, 1

- Home
- Add employee
- Departments
- Employee Attendance
- Approve Leaves
- Check Employee Reviews
- Meetings
- Training
- Salary

Add new admin

Logout

Employees

ID	First Name	Last Name	City	Dept Id	Dept Name	Salary	Manager ID	Email	Actions
24	sohaib	ashraf	karachi	4	Finance	2300		sohaib@gmail.com	Edit Delete
28	Rehan	Nadir	Karaci	5	Accounts	80000		rehan@gmail.com	Edit Delete
29	Ali	Jodat	Islamaabd	3	Human Resources	9000		ali@gmail.com	Edit Delete

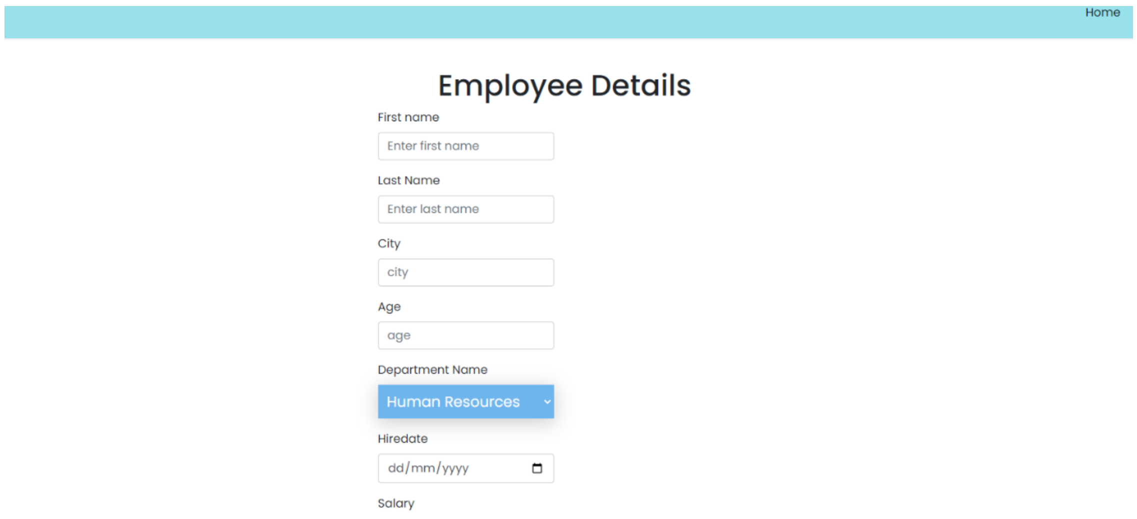
‹ 1 2 3 4 ›

Meetings
Scheduled Meetings
Meetings are scheduled by Admin for employees which can be seen on Employee's Dashboard.
Meetings Sceduled

Training
Scheduled Sessions
Training sessions are scheduled by Admin for employees which can be seen on Employee's Dashboard.
Sessions Sceduled

Attendance
Attendance of employees
Training sessions are scheduled by Admin for employees which can be seen on Employee's Dashboard.
Atendance

Add new employee

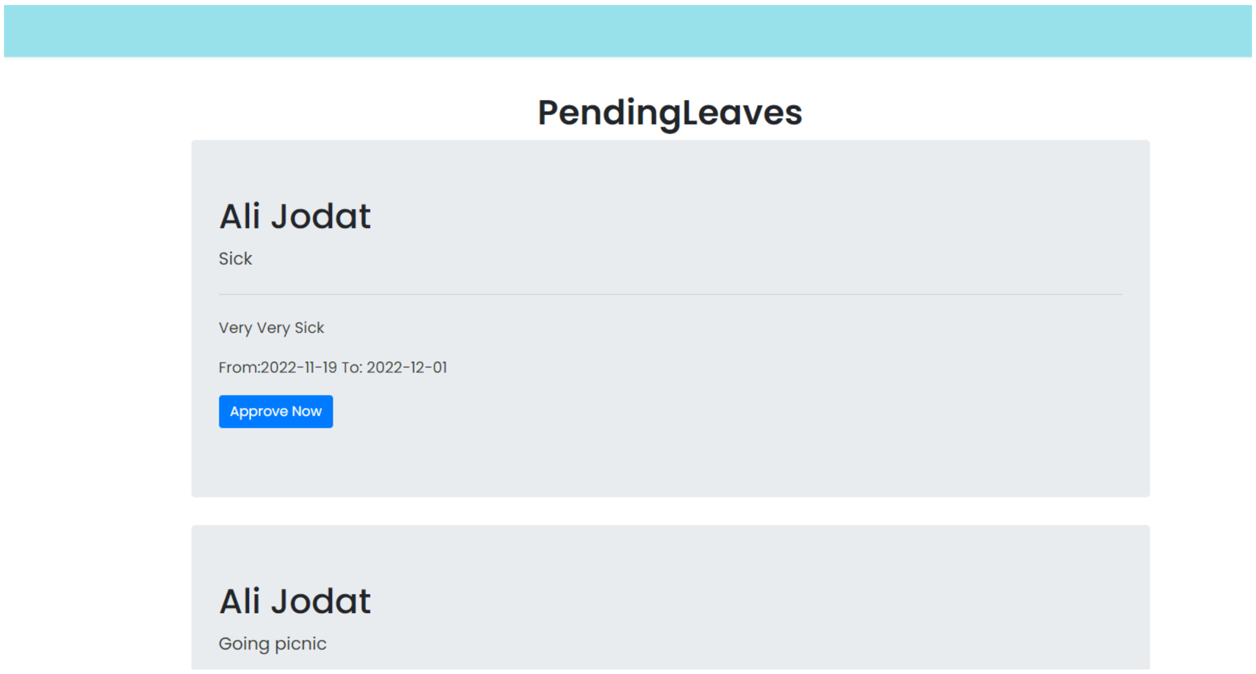


Home

Employee Details

First name	Enter first name
Last Name	Enter last name
City	city
Age	age
Department Name	Human Resources
Hiredate	dd/mm/yyyy
Salary	

Approve Leaves



PendingLeaves

Ali Jodat

Sick

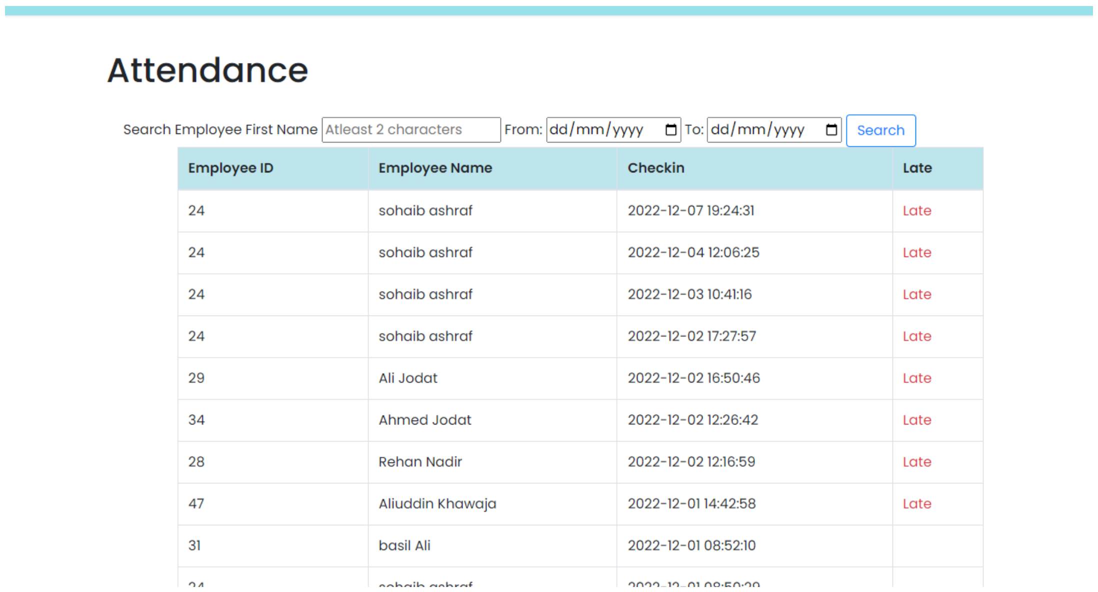
Very Very Sick

From:2022-11-19 To: 2022-12-01

Ali Jodat

Going picnic

Filter Attendance by employee name

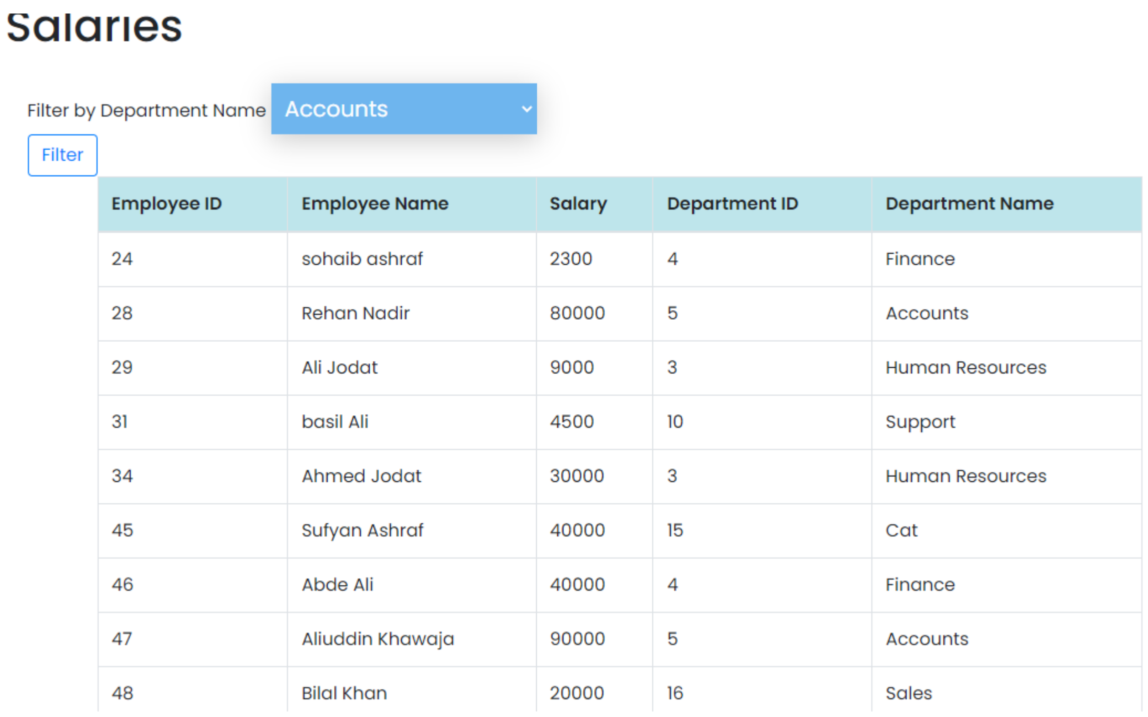


Attendance

Search Employee First Name Atleast 2 characters From: dd/mm/yyyy To: dd/mm/yyyy

Employee ID	Employee Name	Checkin	Late
24	sohaib ashraf	2022-12-07 19:24:31	Late
24	sohaib ashraf	2022-12-04 12:06:25	Late
24	sohaib ashraf	2022-12-03 10:41:16	Late
24	sohaib ashraf	2022-12-02 17:27:57	Late
29	Ali Jodat	2022-12-02 16:50:46	Late
34	Ahmed Jodat	2022-12-02 12:26:42	Late
28	Rehan Nadir	2022-12-02 12:16:59	Late
47	Aliuddin Khawaja	2022-12-01 14:42:58	Late
31	basil Ali	2022-12-01 08:52:10	

Filter salary of employees by department name



Salaries

Filter by Department Name: Accounts

Filter

Employee ID	Employee Name	Salary	Department ID	Department Name
24	sohaib ashraf	2300	4	Finance
28	Rehan Nadir	80000	5	Accounts
29	Ali Jodat	9000	3	Human Resources
31	basil Ali	4500	10	Support
34	Ahmed Jodat	30000	3	Human Resources
45	Sufyan Ashraf	40000	15	Cat
46	Abde Ali	40000	4	Finance
47	Aliuddin Khawaja	90000	5	Accounts
48	Bilal Khan	20000	16	Sales

FUTURE ENHANCEMENTS

Future upgrades to this project will implement:

- Give notification/warning to user once he got late 3 times in a month
- Add more triggers like deleted_emps trigger to our database

References

https://laravel.com/docs/9.x/queries

Ramakrishnan, R., & Gehrke, J. (2011). Database management systems by Boston: McGraw-Hill.

https://github.com/KhawajaAbdullah2000/emp