**Angular Assessment**

**24-May-2024**

**Batch: EDS\_AMJ24\_Angular\_1**

Question 2: Create an angular web application Employee Management system. Get and post the data by angular service and inject this service at App component. Use array of data in service with hardcode values.

Solution:

I have used bootstrap cdn for my code. So please add the link to your index.html, so that the code will look exactly same according to the screenshots attached below:

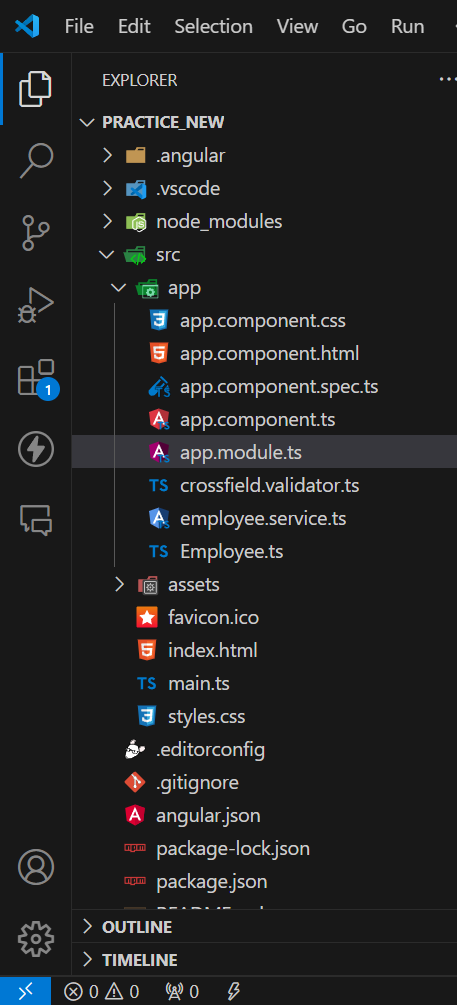
Boostrap CDN link:

  <link rel="stylesheet" href="

  https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"

    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

Please find the folder structure as per the code:



Please find the attached codes according to the files:

app.module.ts

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

import { FormsModule, ReactiveFormsModule } from '@angular/forms';

@NgModule({

  declarations: [

    AppComponent

  ],

  imports: [

    BrowserModule, FormsModule, ReactiveFormsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Employee Model (Employee.ts)

export class Employee{

    emp\_id?:number;

    emp\_name?:string;

    emp\_email?:string;

    password?:string;

    confirmPassword?:string;

    constructor(id:number,name:string,email:string,pwd:string,cnfpwd:string){

        this.emp\_id = id;

        this.emp\_name = name;

        this.emp\_email = email;

        this.password = pwd;

        this.confirmPassword = cnfpwd;

    }

}

employee.service.ts

import { Injectable } from "@angular/core";

import { Employee } from "./Employee";

@Injectable()

export class EmployeeService{

    employeeList: Employee[] = [

        {

            emp\_id : 141,

            emp\_name: 'Arihant',

            emp\_email: 'arihant.kapoor@hcltech.com',

            password: 'Yagi@#14',

            confirmPassword: 'Yagi@#14'

        },

        {

            emp\_id: 145,

            emp\_name: 'Aditi',

            emp\_email: 'aditi.shar@hcltech.com',

            password: 'Panda@#15',

            confirmPassword: 'Panda@#15'

        }

    ];

    addEmployee(emp:Employee):void{

        this.employeeList.push(emp);

    }

    getAllEmployees():Employee[]{

        return this.employeeList;

    }

}

crossfield.validators.ts

import { AbstractControl, ValidationErrors } from "@angular/forms";

export function crossFieldvalidators(control: AbstractControl) : ValidationErrors | null {

    const passwordVal = control.get('password');

    const confirmPasswordVal = control.get('confirmPassword');

   if(passwordVal && confirmPasswordVal && passwordVal.value!==confirmPasswordVal.value){

    return {notmatching: true};

   }

    return null;

}

app.component.ts

import { Component, OnInit } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { EmployeeService } from './employee.service';

import { Employee } from './Employee';

import { crossFieldvalidators } from './crossfield.validator';

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css'],

  providers:[EmployeeService]

})

export class AppComponent implements OnInit {

  empForm!: FormGroup;

  employeeList: Employee[] = [];

  constructor(private fb: FormBuilder, private empService: EmployeeService){}

  ngOnInit(): void {

    this.employeeList = this.empService.getAllEmployees();

    this.empForm = this.fb.group({

      employeeId: ['',[Validators.required,Validators.min(100), Validators.max(999)]],

      employeeName:['',[Validators.required,Validators.minLength(3),Validators.maxLength(30)]],

      employeeEmail:['',[Validators.required,Validators.email]],

      password:['',[Validators.required]],

      confirmPassword:['',[Validators.required]]

    },{validators: crossFieldvalidators});

  }

  //getters

  get Id(){

    return this.empForm.get('employeeId');

  }

  get Name(){

    return this.empForm.get('employeeName');

  }

  get Email(){

    return this.empForm.get('employeeEmail');

  }

  get Password(){

    return this.empForm.get('password');

  }

  get ConfirmPassword(){

    return this.empForm.get('confirmPassword');

  }

  onFormSubmit(){

    const empObj: Employee = new Employee(this.Id?.value,this.Name?.value,this.Email?.value,this.Password?.value,this.ConfirmPassword?.value);

    this.empService.addEmployee(empObj);

    this.employeeList = this.empService.getAllEmployees();

  }

}

app.component.html

<table class="table table-bordered">

  <tr>

    <th>Employee Id</th>

    <th>Employee Name</th>

    <th>Employee Email</th>

    <th>Password</th>

    <th>Confirm Password</th>

  </tr>

  <tr \*ngFor="let employees of employeeList">

    <td>{{employees.emp\_id}}</td>

    <td>{{employees.emp\_name}}</td>

    <td>{{employees.emp\_email}}</td>

    <td>{{employees.password}}</td>

    <td>{{employees.confirmPassword}}</td>

  </tr>

</table>

<div class="container">

  <form [formGroup]="empForm" (ngSubmit)="onFormSubmit()">

    <div class="form-group">

      <label>Employee Id</label>

      <input type="number" class="form-control" formControlName="employeeId">

      <small class="text-danger" \*ngIf="Id?.invalid && Id?.touched">Id should be of 3 digit number</small>

    </div>

    <div class="form-group">

      <label>Employee Name</label>

      <input type="text" class="form-control" formControlName="employeeName">

      <small class="text-danger" \*ngIf="Name?.invalid && Name?.touched">Name should be between 3 and 30 characters</small>

    </div>

    <div class="form-group">

      <label>Employee Email</label>

      <input type="email" class="form-control" formControlName="employeeEmail">

      <small class="text-danger" \*ngIf="Email?.invalid && Email?.touched">Email is required</small>

    </div>

    <div class="form-group">

      <label>Password</label>

      <input type="text" class="form-control" formControlName="password">

      <small class="text-danger" \*ngIf="Password?.invalid && Password?.touched">Password is required</small>

    </div>

    <div class="form-group">

      <label>Confirm Password</label>

      <input type="text" class="form-control" formControlName="confirmPassword">

      <small class="text-danger" \*ngIf="empForm?.errors?.['notmatching'] && ConfirmPassword?.touched">Confirm password and password do not matched</small>

    </div>

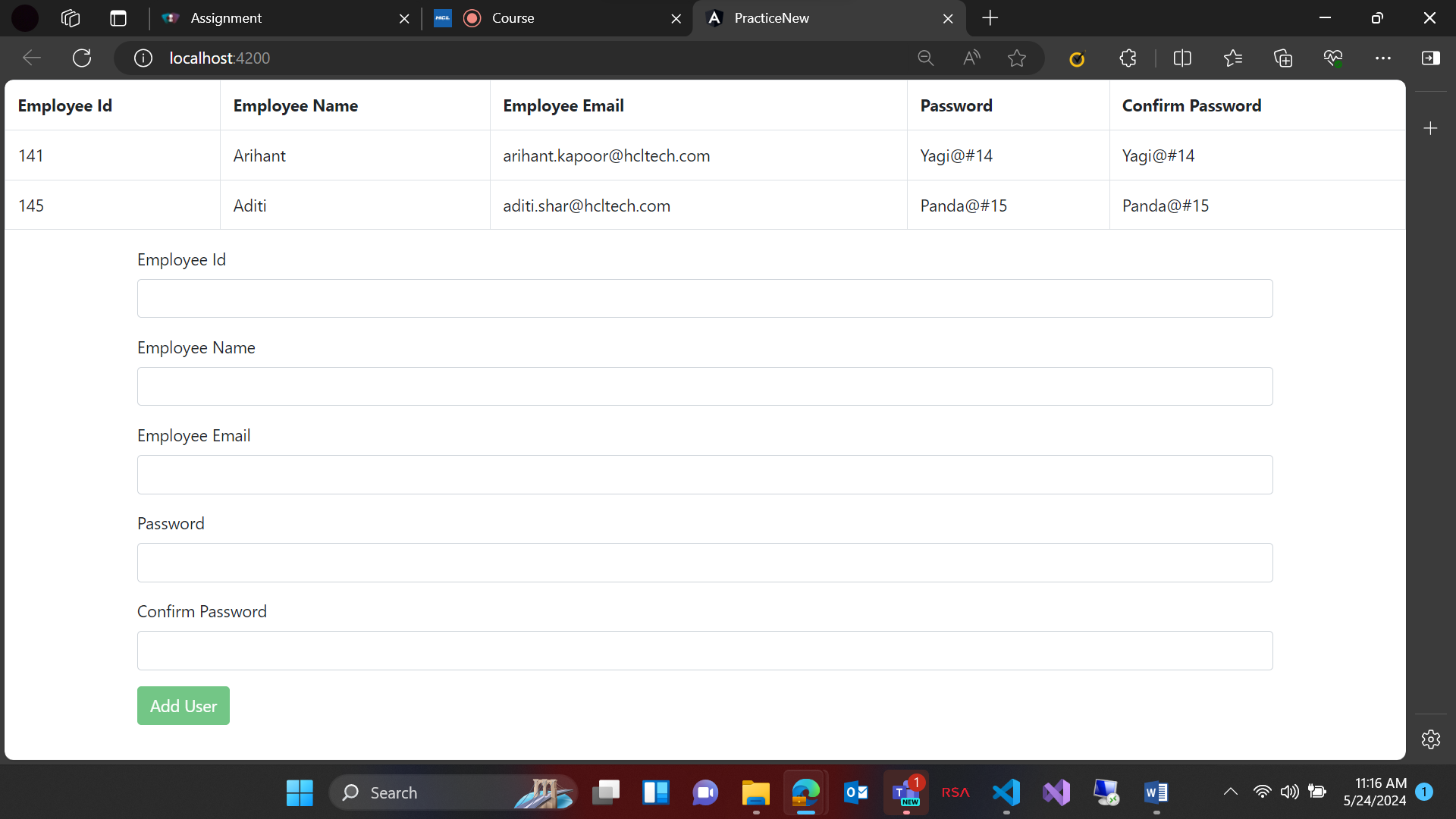
    <button class="btn btn-success" [disabled]="empForm.invalid">Add User</button>

  </form>

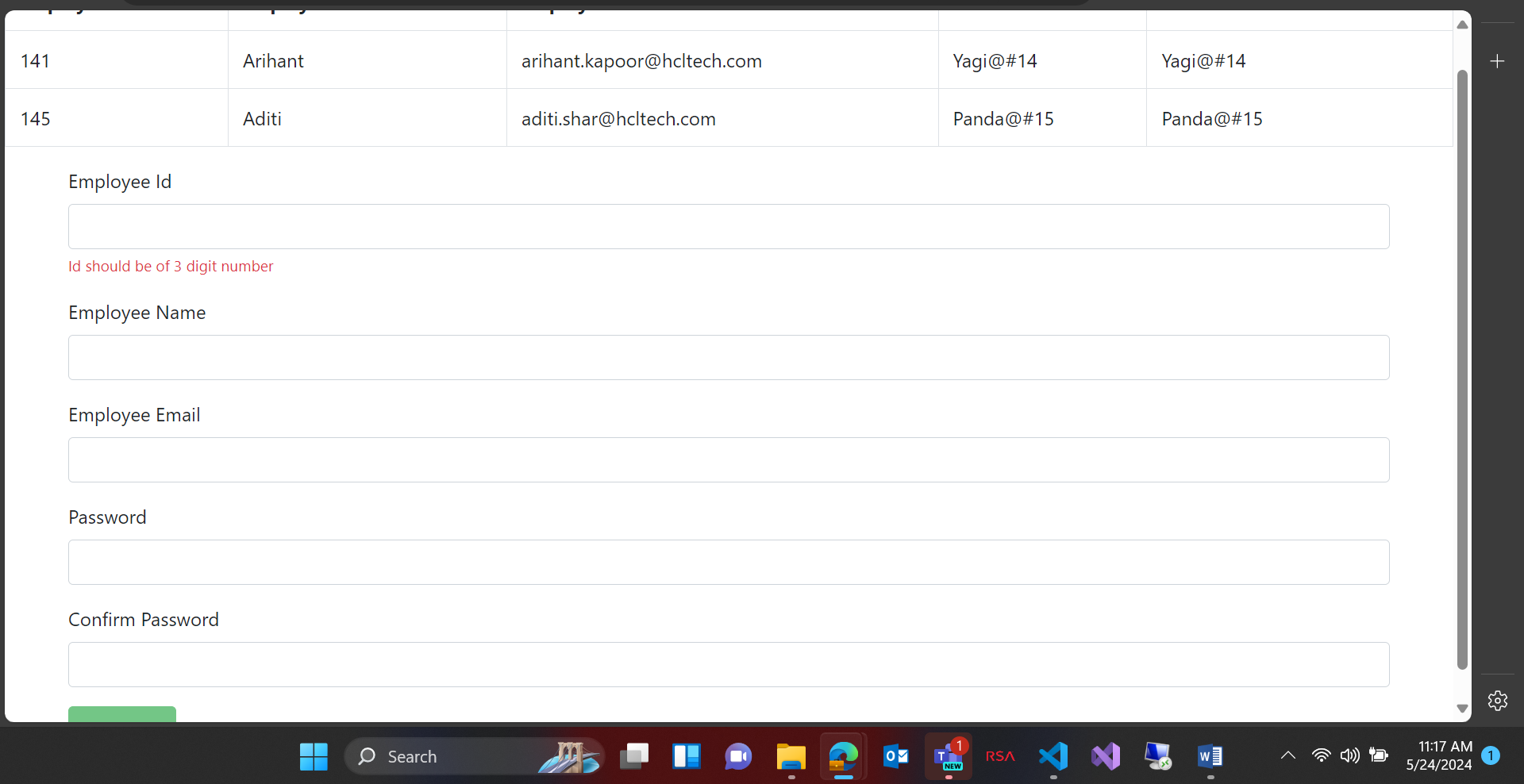
</div>

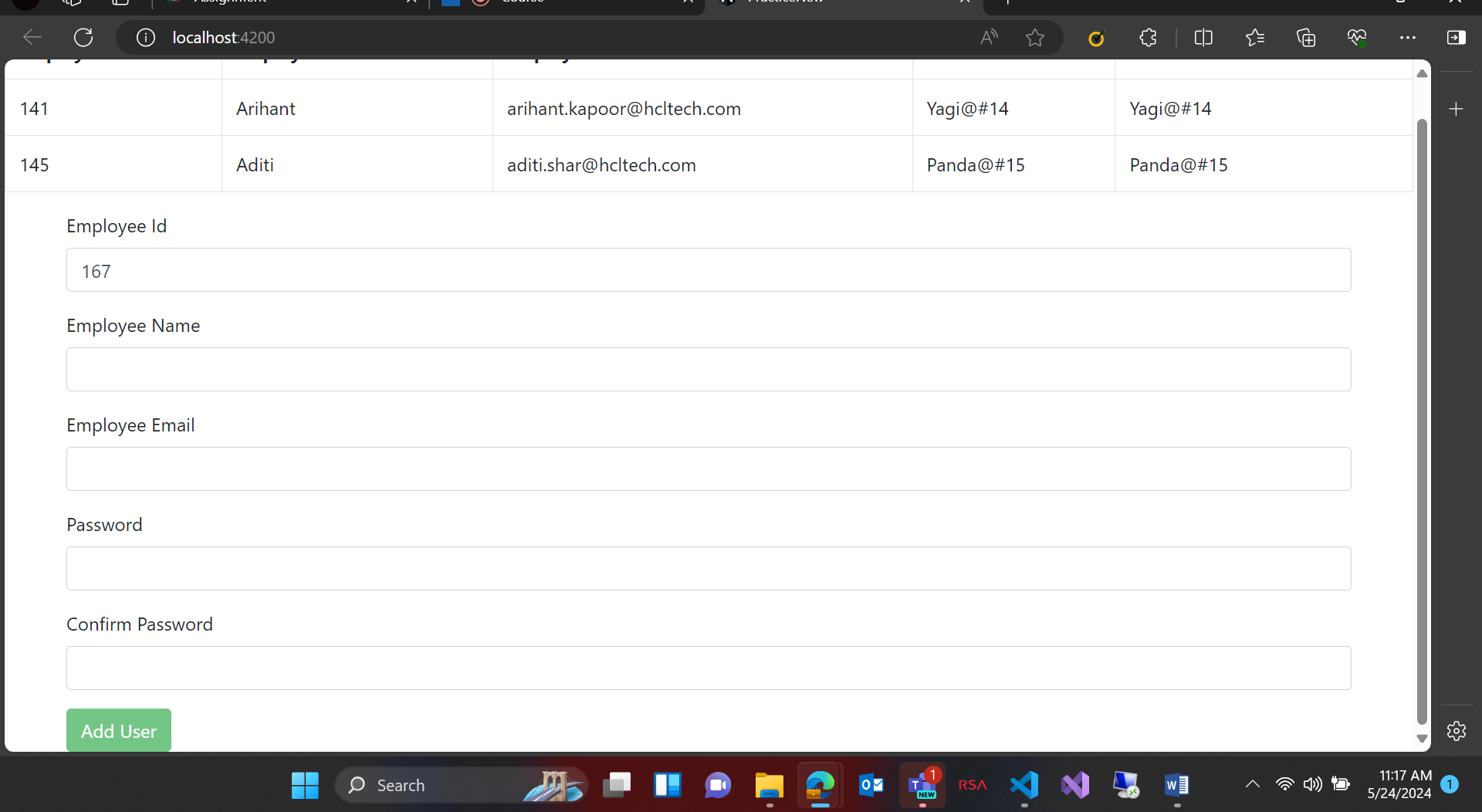
Please find the attached Output screenshots for each scenario:

Output(View) of code:

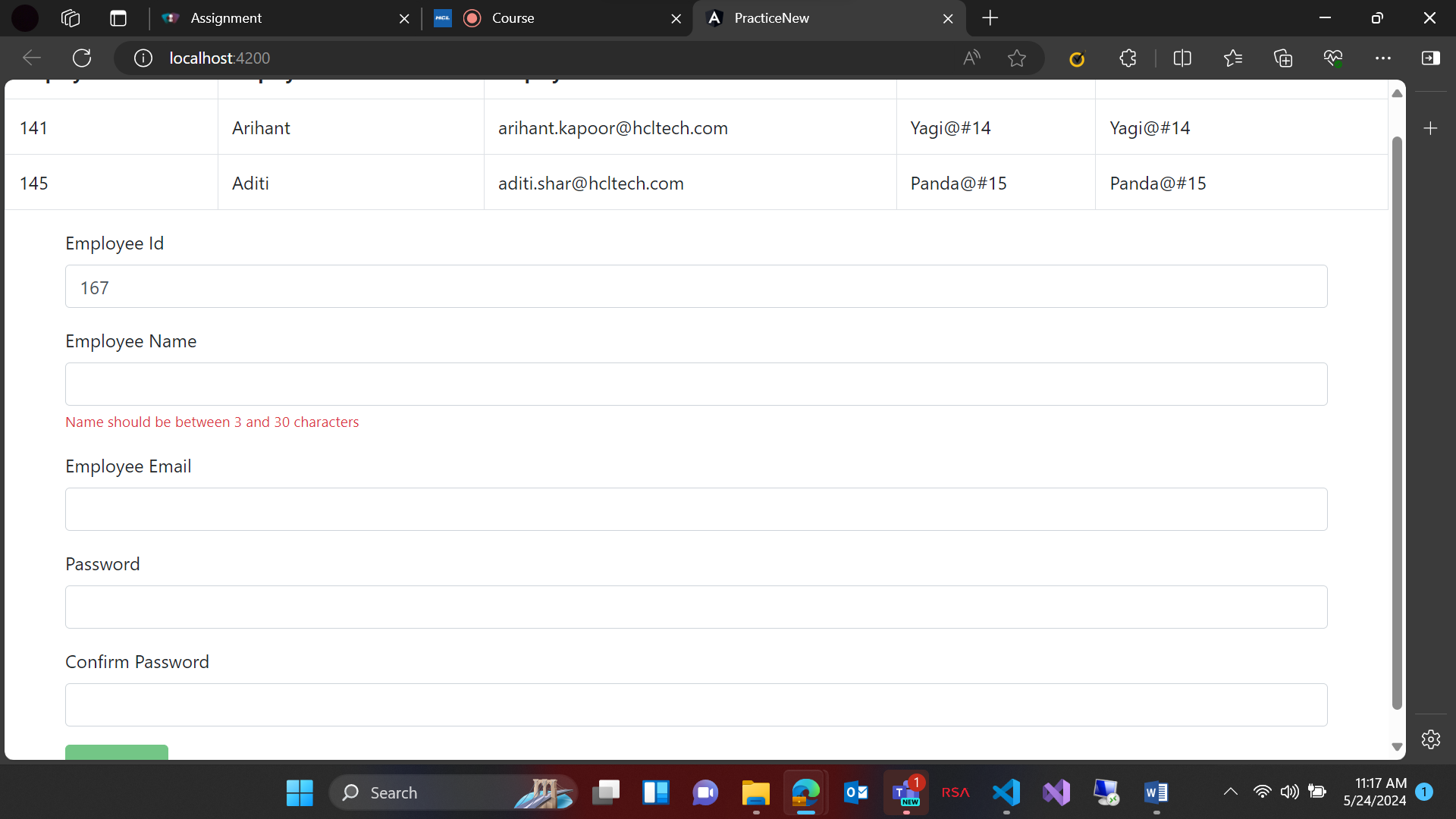


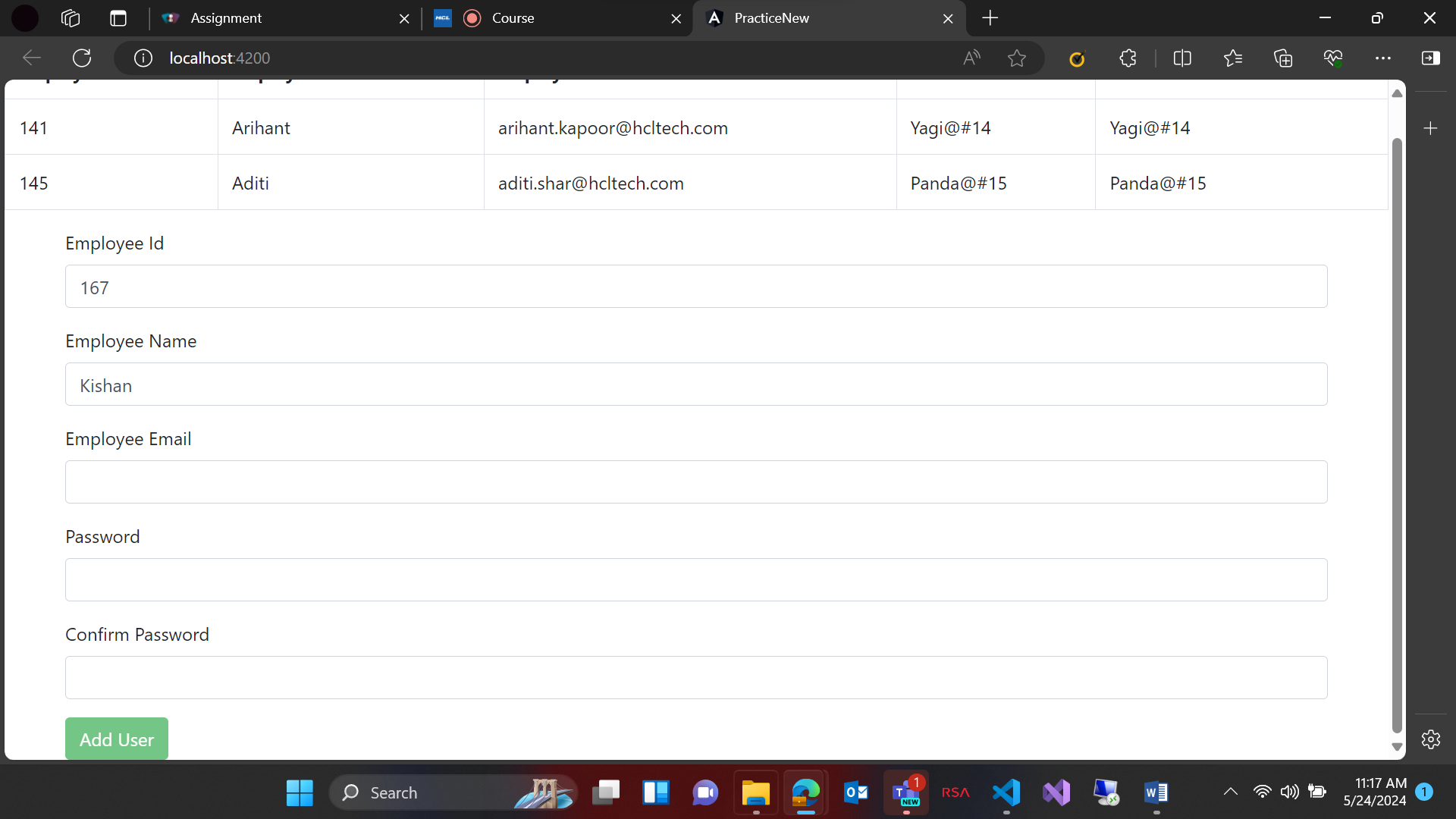
Id validation:



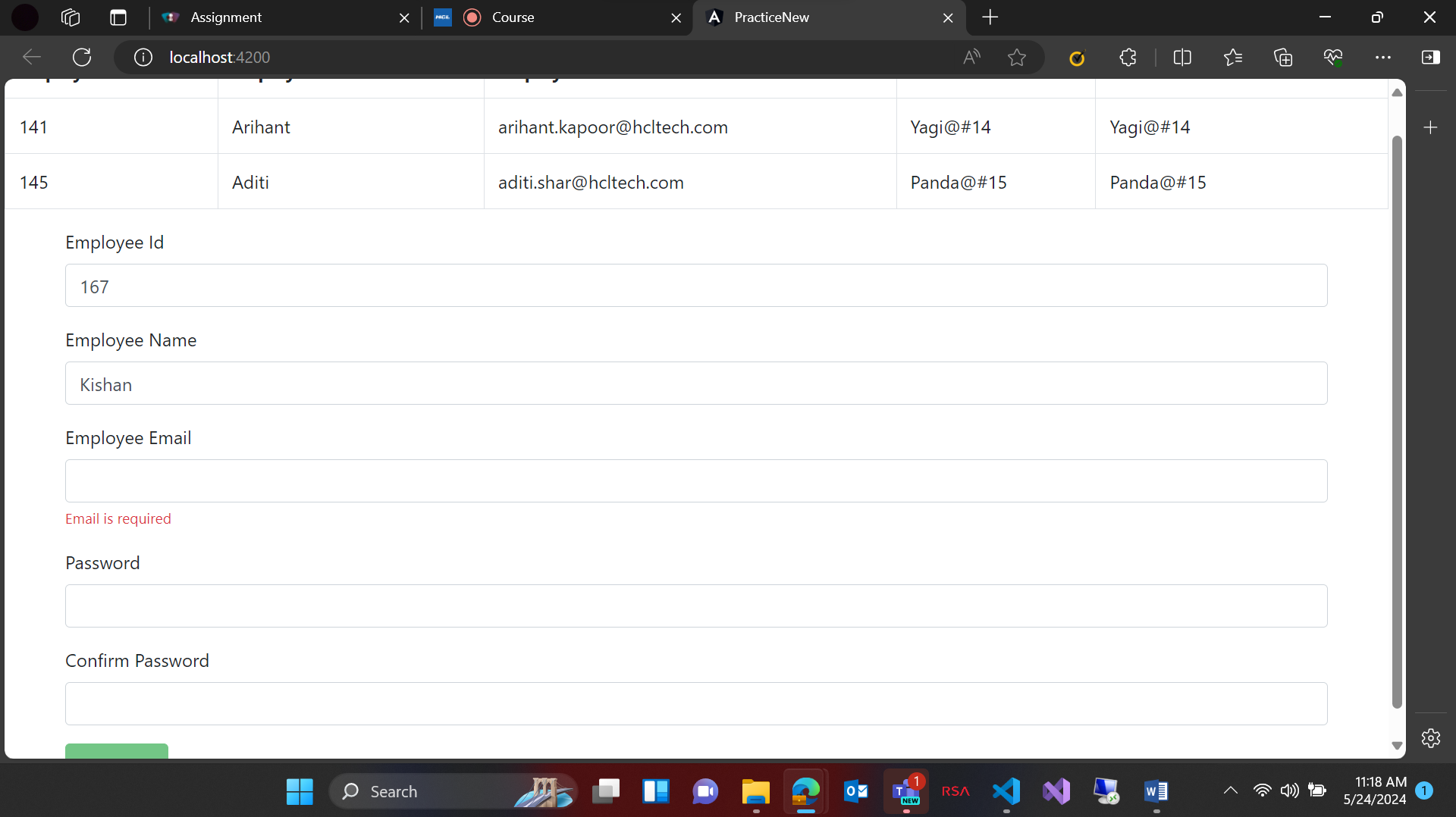


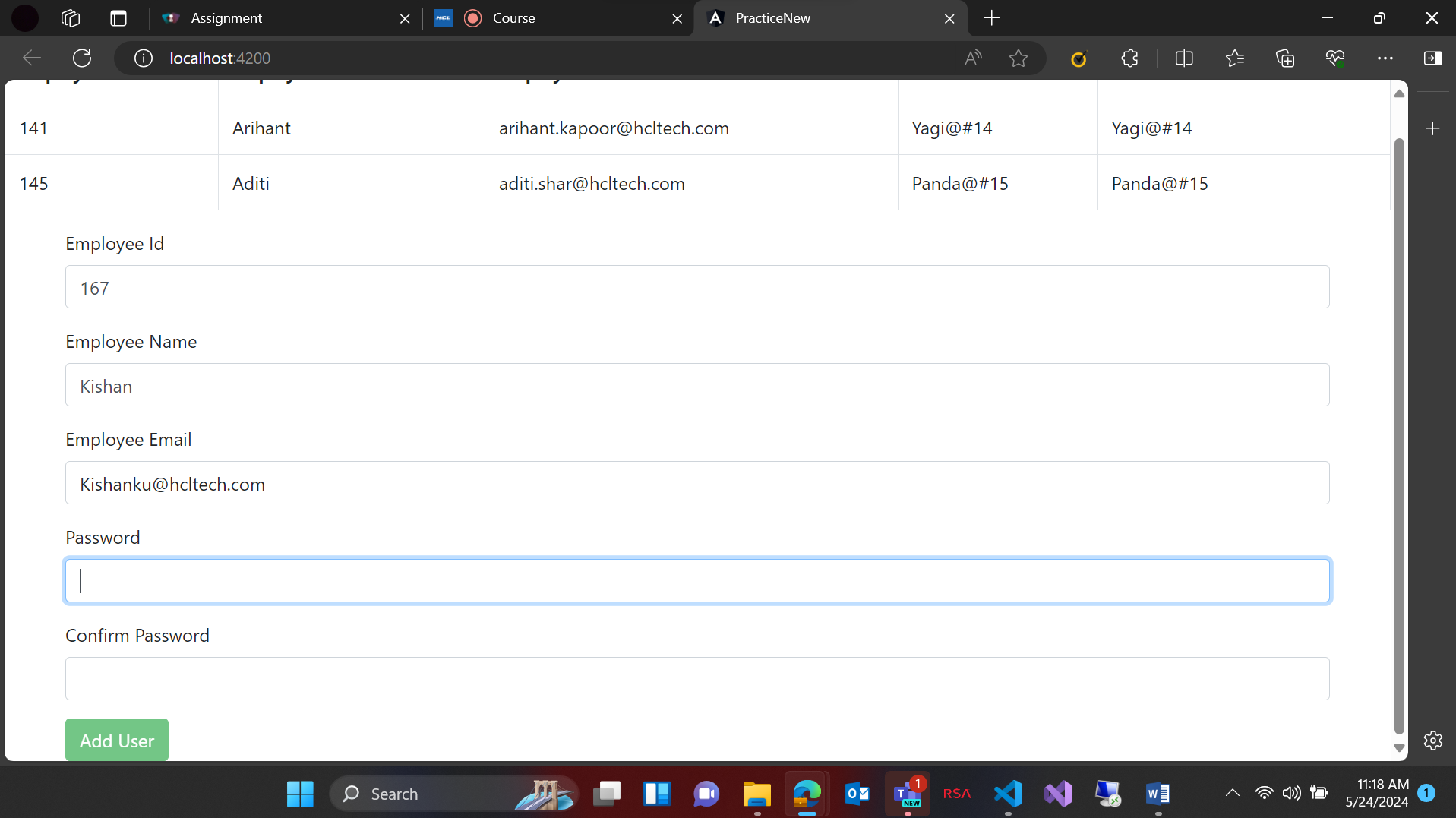
Name Validation:



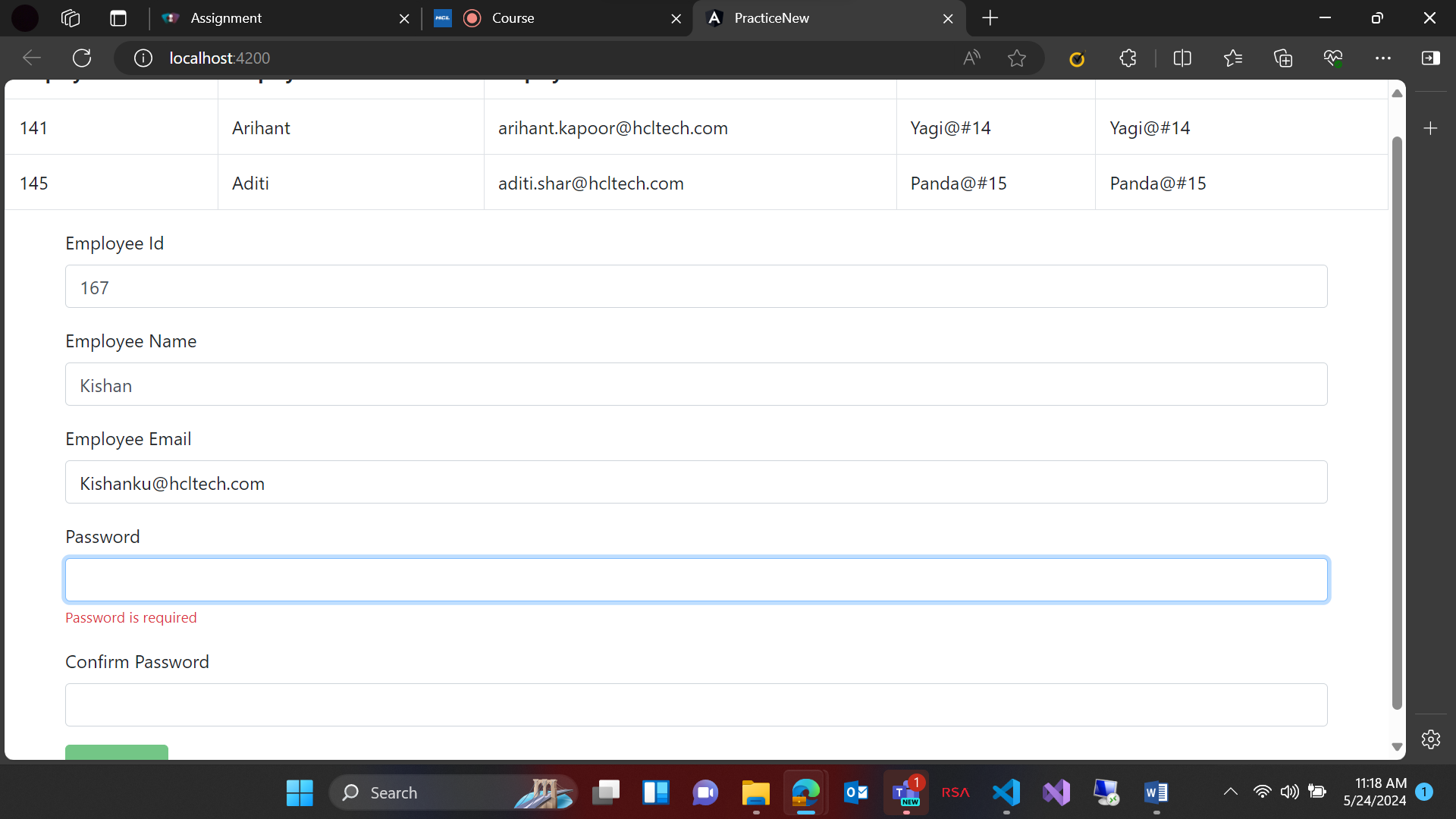


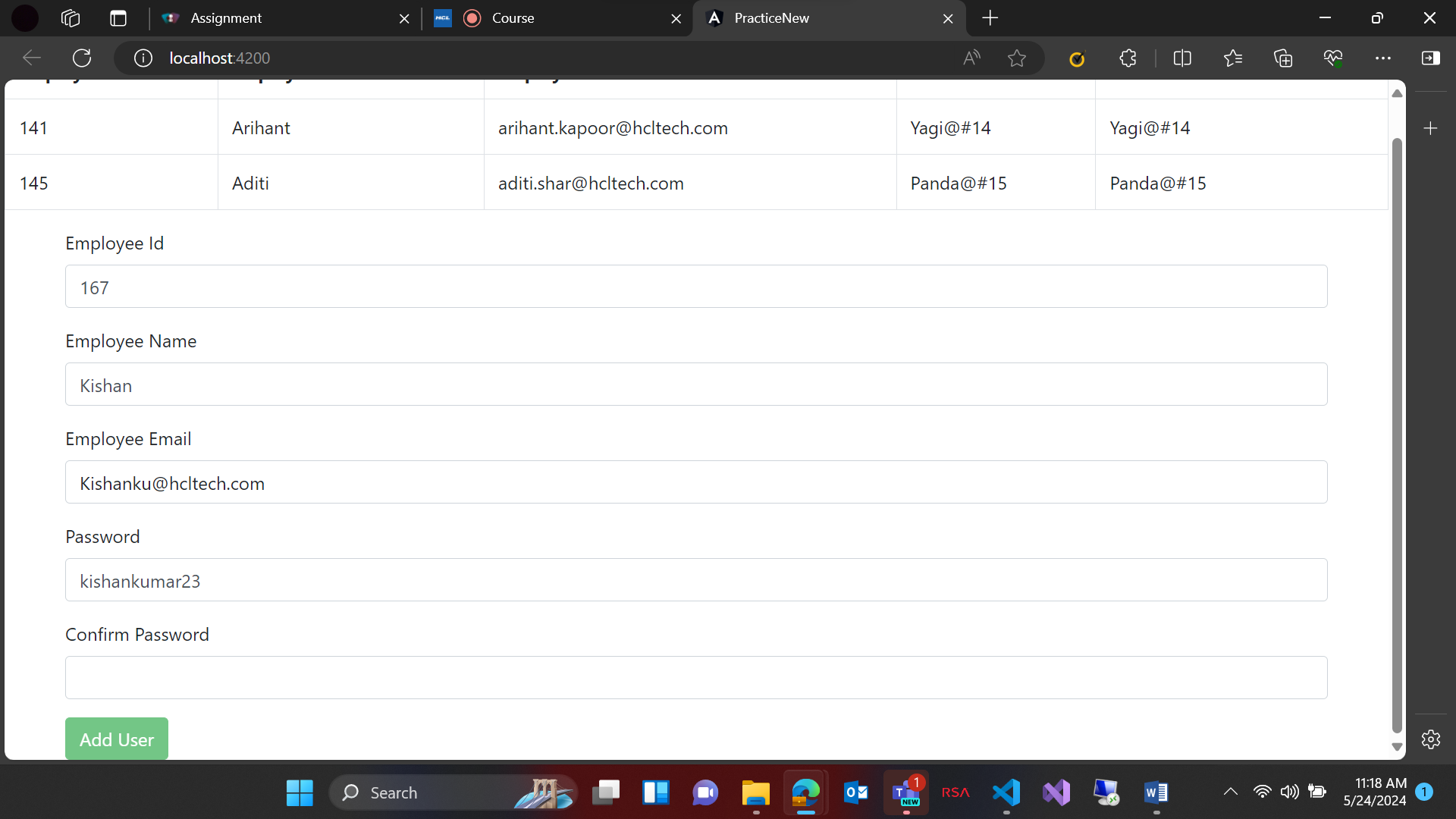
Email validation:



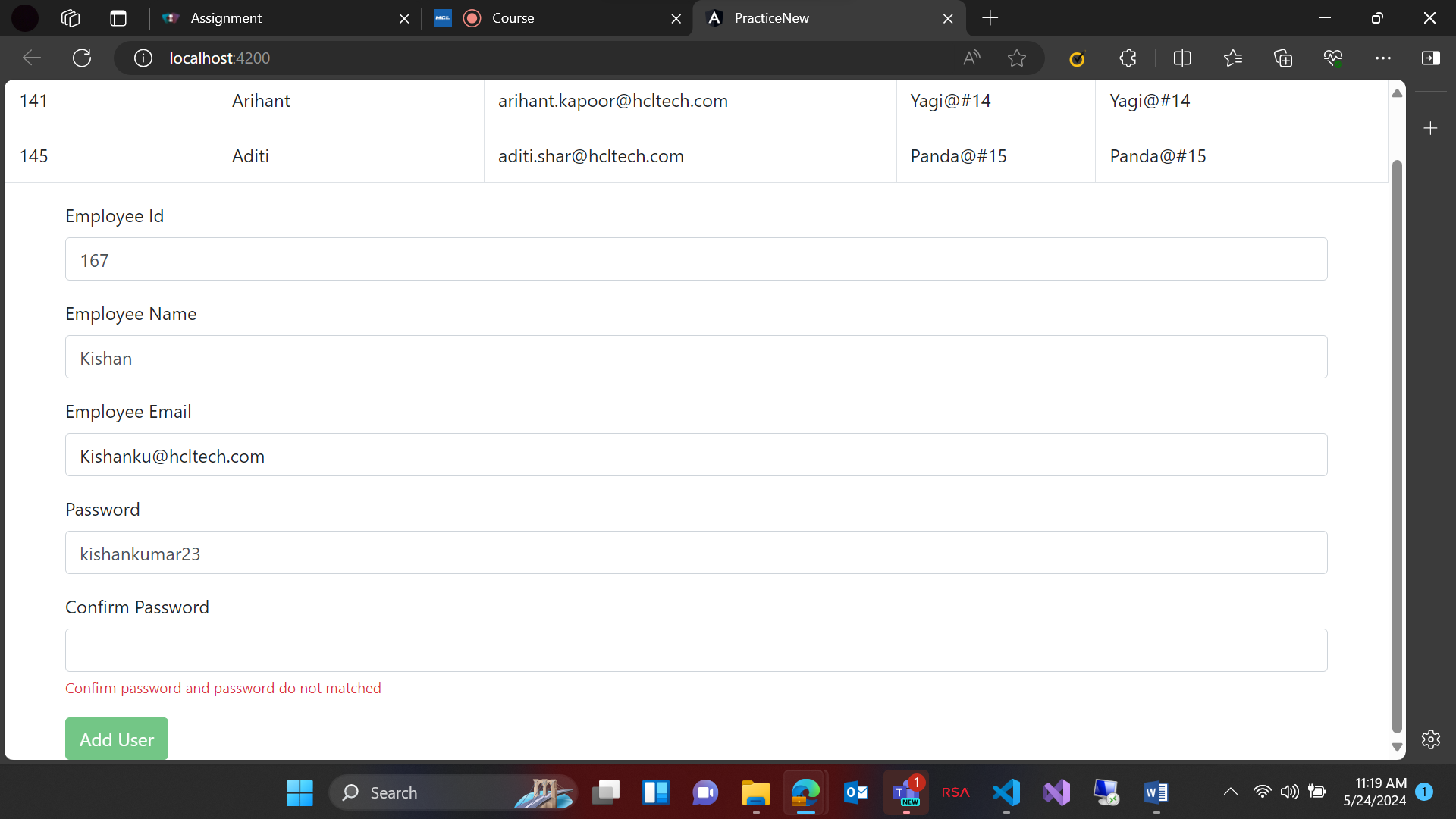


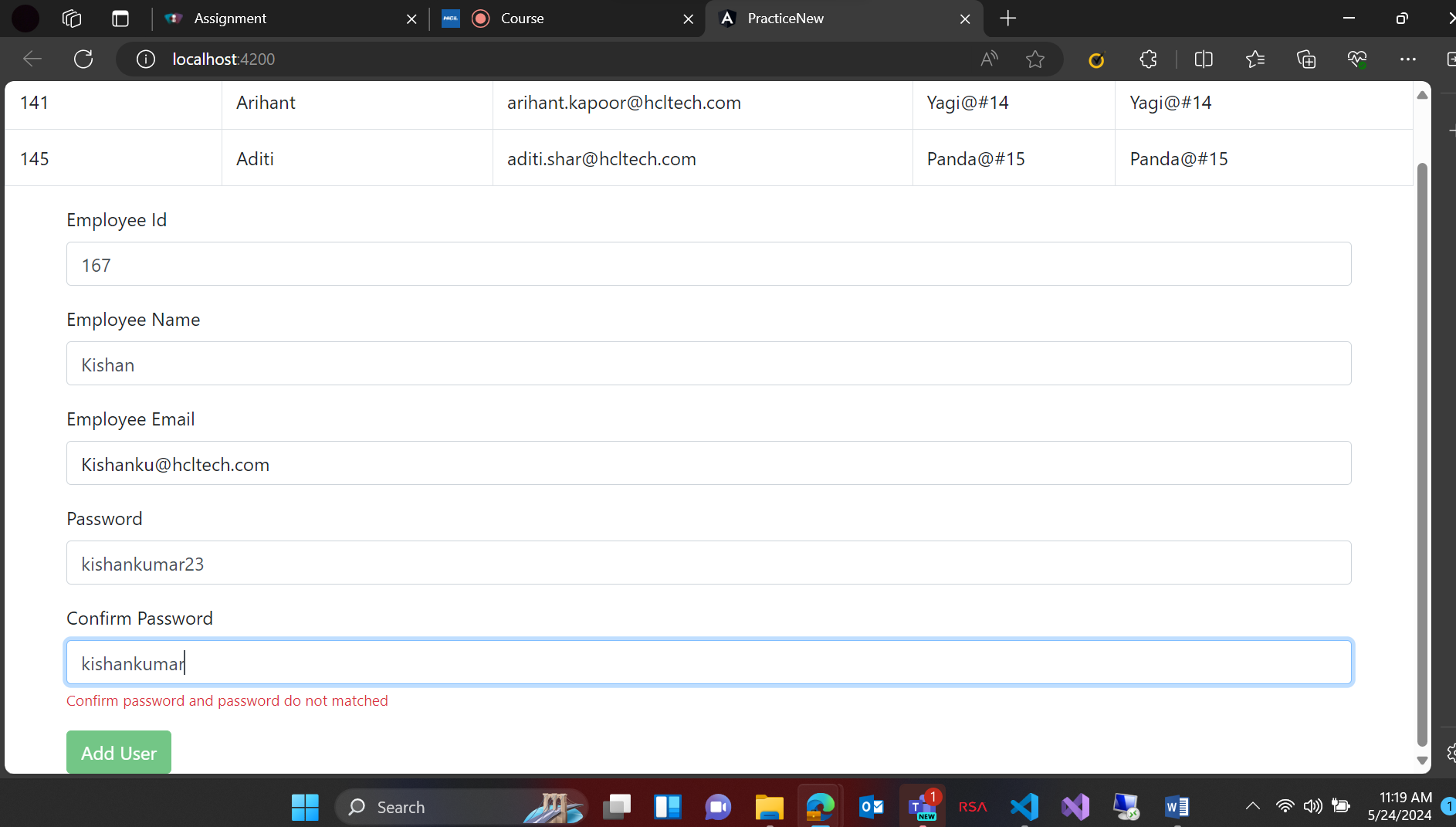
Password validation:

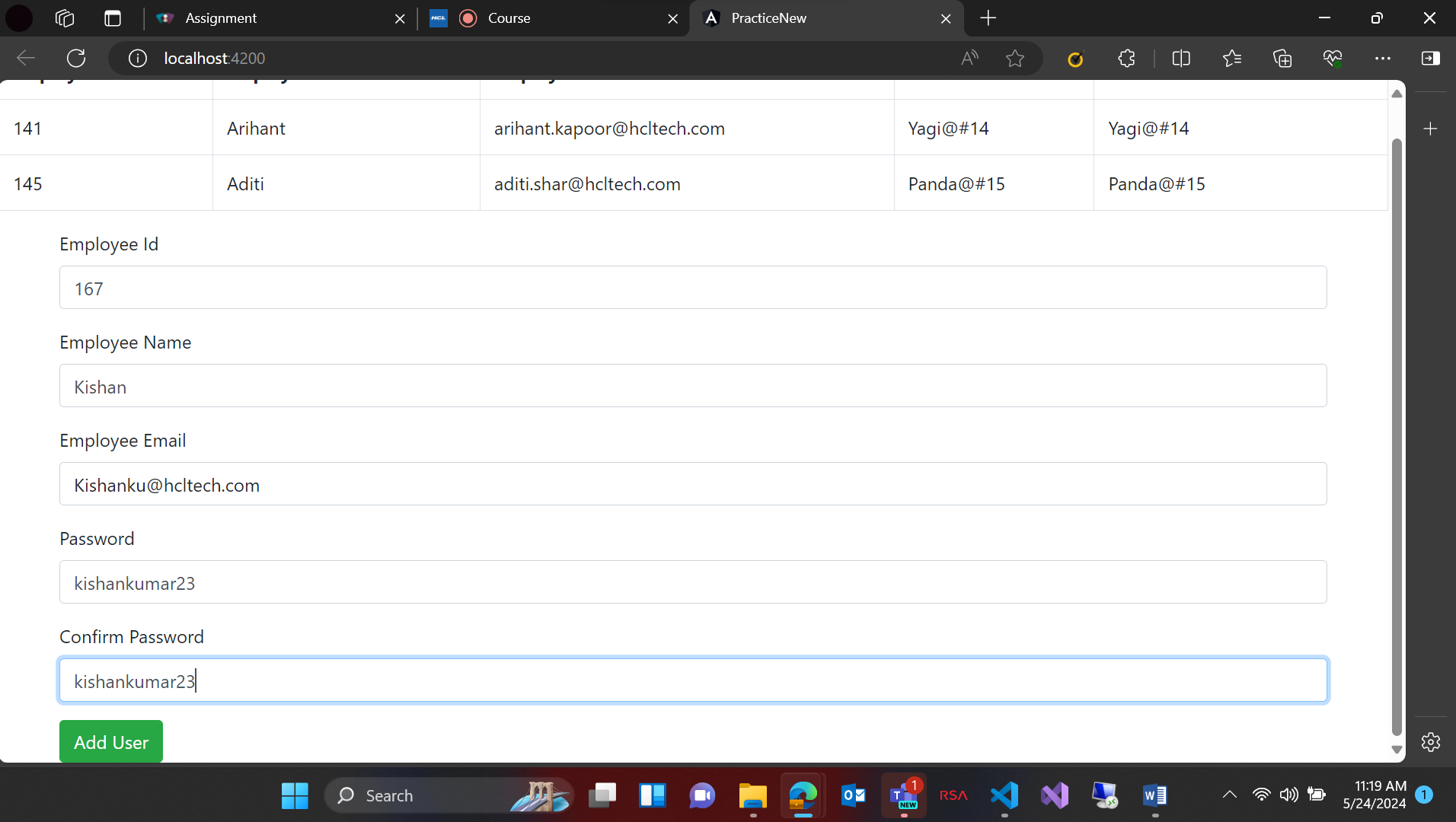




Confirm Password validation:







Adding the User data in the list:

