**Steps For ABP Commercial: -**

* Install abp suite – on cmd – **abp suite install -v 5.3.3**
* Run command – abp suite and enter
* Folder is created with two main folder angular and asp-net core
* Open aspnet-core folder and open the solution file in solution file there is src and test folder open src => and set webhost soln as start up project

1. In api host **appsetting.json** set connection string
2. In Dbmigrator also set connection string

**"ConnectionStrings": {**

**"Default": "Data Source=.;Initial Catalog=CustomerAppointmentScheduling;Integrated Security=True"**

**}**

* Using abp suite crud page generator =>  Give entity info and apoperties then click on save and generate button
* Set .DBMigrator as a startup project and run for the migration
* In nuget package manager console run the command for adding migration **add-migration “name” -context dbcontext classname                              update-database -context dbcontext classname**
* Run the web host soln with setting startup project
* In angular folder open terminal                                                                            run the command **yarn** and    **yarn start**
* Login to UI with **admin** and **1q2w3E\***

Appointment Scheduling Application

* Adding new appointments and deleting some of them. Auto-deletion of the appointment after the time has passed should also be implemented feature.

1. Adding new appointment is done with Automatic crud operation.
2. In the Customer Service at getlist() add this linq query to get one day before data and delete data using Id.
3. DeleteAsync is inbuild method.

public virtual async Task<PagedResultDto<CustomerDto>> GetListAsync(GetCustomersInput input)

{

var totalCount = await \_customerRepository.GetCountAsync(input.FilterText, input.Name, input.Address, input.MobileNumber, input.AppointmentDateMin, input.AppointmentDateMax, input.VerificationStatus);

var items = await \_customerRepository.GetListAsync(input.FilterText, input.Name, input.Address, input.MobileNumber, input.AppointmentDateMin, input.AppointmentDateMax, input.VerificationStatus, input.Sorting, input.MaxResultCount, input.SkipCount);

var autodelete = (from a in items

where a.AppointmentDate.Date == DateTime.Now.AddDays(-1).Date

select a).AsEnumerable().ToList();

if (autodelete.Count != 0)

{

foreach (var item in autodelete)

{

await \_customerRepository.DeleteAsync(item.Id);

}

}

return new PagedResultDto<CustomerDto>

{

TotalCount = totalCount,

Items = ObjectMapper.Map<List<Customer>, List<CustomerDto>>(items)

};

}

* Sending SMS to the user 1 day prior to the appointment time.

1. Register on Twilio Trial Account.
2. In pgm.cs add following code:

"TwilioSmsSettings": {

"AccountSId": "AC9f0daacc38a759115a289666039766fc",

"AuthToken": "a3f9b0e9c2418f636da42000ba382951",

"TwilioPhoneNumber": "+12056497825"

}

(U will get this id and number from Twilio Account)

3.Create interface named ITwilioSmsSender

Create method void SendSmsAsync(string to);

Add following code in TwilioSmsSender Service

In service create dependency injection like this and inherit ApplicationService for objectmapper

private readonly IConfiguration \_configuration;

// // private readonly IMapper \_mapper;

private readonly IDataFilter \_dataFilter;

IRepository<Customer, Guid> \_dbContextProvider;

ICustomerRepository \_icustomerRepository;

AppoinmentSchelulingProjectDbContext \_bd;

// //DbContextOptions<AppointmentSchedularProjectDbContext> options = new DbContextOptions<AppointmentSchedularProjectDbContext>();

public TwilioSmsSender(IConfiguration configuration, IDataFilter dataFilter, ICustomerRepository,

AppoinmentSchelulingProjectDbContext bd,

IRepository<Customer, Guid> dbContextProvider)

{

\_dataFilter = dataFilter;

\_configuration = configuration;

\_dbContextProvider = dbContextProvider;

\_icustomerRepository = icustomerRepository;

\_bd = bd;

// //\_mapper = Mapper;

And add builder.Services.AddScoped<ITwilioSmsSender, TwilioSmsSender>();

in pgm.cs in try block.

Then add following code:-

public void SendSmsAsync(string to)

{

var accountSid = \_configuration["TwilioSmsSettings:AccountSId"];

var authToken = \_configuration["TwilioSmsSettings:AuthToken"];

var phoneNumber = \_configuration["TwilioSmsSettings:TwilioPhoneNumber"];

TwilioClient.Init(accountSid, authToken);

try

{

MessageResource response = MessageResource.Create(

body: "Appointment Confirmed",

from: phoneNumber,

// to: "+918421721966"

to: to

);

}

catch (Exception ex)

{

throw;

}

}

5.Create Task<CustomerDto> getCustomerdataAsync();in interface

6.Add following code in the service

public async Task<CustomerDto> getCustomerdataAsync()

{

IQueryable<Customer> queryable = await \_dbContextProvider.GetQueryableAsync();

//

var customer = (from b in queryable

where b.AppointmentDate.Date == DateTime.Now.AddDays(+1).Date

select b).AsEnumerable().ToList();

// var CustDto = ObjectMapper.Map<Appointment, AppointmentDto>(customer);

foreach (var item in customer)

{

SendSmsAsync(item.MobileNumber);

}

return null;

}

* Call the method in controller like this

[HttpGet("messagesend")]

public virtual async Task SmsSender()

{

try

{

await \_ismssender.getCustomerdataAsync();

}

catch (Exception ex)

{

//return ex.Message.ToString;

}

}

* To set the timer to send msg in every 24 hrs

public void StartTimerAsync()

{

Timer newTimer = new Timer();

newTimer.Interval = 86400000;

newTimer.Elapsed += DisplayTimeEvent;

newTimer.AutoReset = true;

newTimer.Enabled = true;

newTimer.Start();

}

* In DisplayTimeEvent call getCustomerdataAsync()

public void DisplayTimeEvent(object source, ElapsedEventArgs s)

{

getCustomerdataAsync();

}

**In pgm.cs,in try block define a scope..so that after execute the pgm timer will start**

using (var scope = app.Services.CreateScope())

{

var smssender = scope.ServiceProvider.GetRequiredService<ITwilioSmsSender>();

smssender.StartTimerAsync();

}

await app.RunAsync();

return 0;

The list of all the upcoming appointments

* Create method in Icustomer

Task<PagedResultDto<CustomerDto>> UpcomingAppointment(GetCustomersInput input);

* In customer Service create method UpcomingAppointments and add following query

public virtual async Task<PagedResultDto<CustomerDto>> UpcomingAppointment(GetCustomersInput input)

{

IQueryable<Customer> queryable = await \_customerRepository.GetQueryableAsync();

var customer = (from b in queryable

where b.AppointmentDate.Date == DateTime.Now.AddDays(+1).Date

select b).ToList();

return new PagedResultDto<CustomerDto>

{

Items = ObjectMapper.Map<List<Customer>, List<CustomerDto>>(customer)

};

}

* Call this method in controller

[HttpGet("Upcoming")]

public virtual async Task<PagedResultDto<CustomerDto>> UpcomingAppointment(GetCustomersInput input)

{

var abc= await \_customersAppService.UpcomingAppointment(input);

return abc;

}

The list of all the Cancelled appointments

* Add method in ITwilioSmsSender

Task<PagedResultDto<CustomerDto>> DeletedDataAsync();

* Add following code in Service

public async Task<PagedResultDto<CustomerDto>> DeletedDataAsync()

{

using (\_dataFilter.Disable<ISoftDelete>())

{

IQueryable<Customer> queryable = await \_dbContextProvider.GetQueryableAsync();

//var queryable= \_bd.Customers.ToList();

var customer = (from b in queryable

where b.IsDeleted == true

select b).AsEnumerable().ToList();

return new PagedResultDto<CustomerDto>

{

Items = ObjectMapper.Map<List<Customer>, List<CustomerDto>>(customer)

};

}

}

* Call in controller

[HttpGet("Deletedata")]

public virtual async Task<ActionResult<PagedResultDto<CustomerDto>>> deletedata()

{

try

{

var delete = await \_ismssender.DeletedDataAsync();

return delete;

}

catch (Exception ex)

{

return BadRequest(ex.Message);

}

}

Angular:-

Add the following code in Service

getUpcomingAppointments = () =>

    this.restService.request<any, PagedResultDto<CustomerDto>>({

      method: 'GET',

      url: '/api/app/customers/Upcoming',

      //params: { filterText: input.filterText, name: input.name, address: input.address, mobileNumber: input.mobileNumber, appointmentDateMin: input.appointmentDateMin, appointmentDateMax: input.appointmentDateMax, verificationStatus: input.verificationStatus, sorting: input.sorting, skipCount: input.skipCount, maxResultCount: input.maxResultCount },

    },

    { apiName: this.apiName });

   DeletedAppoinments = () =>

    this.restService.request<any, PagedResultDto<CustomerDto>>({

      method: 'GET',

      url: '/api/app/customers/Deletedata',

      //params: { filterText: input.filterText, name: input.name, address: input.address, mobileNumber: input.mobileNumber, appointmentDateMin: input.appointmentDateMin, appointmentDateMax: input.appointmentDateMax, verificationStatus: input.verificationStatus, sorting: input.sorting, skipCount: input.skipCount, maxResultCount: input.maxResultCount },

    },

    { apiName: this.apiName });

Add following code in ts file to list upcoming and cancelled appointment

customers!:any[]

  getUpcomingappointment()

 {

  this.\_apiservice.getUpcomingAppointments().subscribe({next:(data)=>{

    console.log(data)

    this.customers=data.items}}

  )

  console.log(this.customers)

 }

 deleteappointments!:any[]

 getDeletedData()

 {

  this.\_apiservice.DeletedAppoinments().subscribe({next:(data)=>{

    console.log(data)

    this.deleteappointments=data.items}}

  )

  console.log(this.customers)

 }

Add following code in html

<div class="text-lg-end pt-2">

      <div ngbDropdown container="body" class="d-inline-block">

        <button class="btn btn-primary btn-sm " data-toggle="dropdown" aria-haspopup="true"

        ngbDropdownToggle >

          <!-- <i class="bi bi-calendar-heart" aria-hidden="true"></i> -->

          <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="white" class="bi bi-calendar-heart" viewBox="0 0 16 16">

            <path fill-rule="evenodd" d="M4 .5a.5.5 0 0 0-1 0V1H2a2 2 0 0 0-2 2v11a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V3a2 2 0 0 0-2-2h-1V.5a.5.5 0 0 0-1 0V1H4V.5ZM1 14V4h14v10a1 1 0 0 1-1 1H2a1 1 0 0 1-1-1Zm7-6.507c1.664-1.711 5.825 1.283 0 5.132-5.825-3.85-1.664-6.843 0-5.132Z"/>

          </svg>&nbsp;&nbsp;

          {{ 'AbpUi::Appoinments' | abpLocalization }}

        </button>

        <div ngbDropdownMenu>

          <button type="button" data-bs-toggle="modal" data-bs-target="#staticBackdrop" ngbDropdownItem

            (click)="getUpcomingappointment()">

            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-calendar2-check-fill" viewBox="0 0 16 16">

              <path d="M3.5 0a.5.5 0 0 1 .5.5V1h8V.5a.5.5 0 0 1 1 0V1h1a2 2 0 0 1 2 2v11a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2V3a2 2 0 0 1 2-2h1V.5a.5.5 0 0 1 .5-.5zm9.954 3H2.545c-.3 0-.545.224-.545.5v1c0 .276.244.5.545.5h10.91c.3 0 .545-.224.545-.5v-1c0-.276-.244-.5-.546-.5zm-2.6 5.854a.5.5 0 0 0-.708-.708L7.5 10.793 6.354 9.646a.5.5 0 1 0-.708.708l1.5 1.5a.5.5 0 0 0 .708 0l3-3z"/>

            </svg>&nbsp;&nbsp;

            {{ 'AbpUi::Upcoming Appoinments' | abpLocalization }}

          </button>

      <button type="button"  data-bs-toggle="modal" data-bs-target="#exampleModal" ngbDropdownItem (click)="getDeletedData()">

            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-calendar-x-fill" viewBox="0 0 16 16">

              <path d="M4 .5a.5.5 0 0 0-1 0V1H2a2 2 0 0 0-2 2v1h16V3a2 2 0 0 0-2-2h-1V.5a.5.5 0 0 0-1 0V1H4V.5zM16 14V5H0v9a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2zM6.854 8.146 8 9.293l1.146-1.147a.5.5 0 1 1 .708.708L8.707 10l1.147 1.146a.5.5 0 0 1-.708.708L8 10.707l-1.146 1.147a.5.5 0 0 1-.708-.708L7.293 10 6.146 8.854a.5.5 0 1 1 .708-.708z"/>

            </svg>&nbsp;&nbsp;

            {{ 'AbpUi::Cancelled Appoinments' | abpLocalization }}

          </button>

</div>

      </div>

      <button \*abpPermission="'AppoinmentSchelulingProject.Customers.Create'" class="btn btn-primary btn-sm mx-3"

        type="button" (click)="create()">

        <i class="fa fa-plus me-1" aria-hidden="true"></i>

        {{ '::NewCustomer' | abpLocalization }}

      </button>

    </div>

<!-- Modal for upcoming-->

<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"

  aria-labelledby="staticBackdropLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content " style="width: 650px" >

      <div class="modal-header">

        <h5 class="modal-title" id="staticBackdropLabel">Upcoming Appoinments</h5>

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

      </div>

      <div class="modal-body">

        <table class="table">

          <thead style="background-color: purple; color: white;">

            <tr>

              <th scope="col">Name</th>

              <th scope="col">Address</th>

              <th scope="col">MobileNumber</th>

              <th scope="col">AppointmentDate</th>

            </tr>

          </thead>

          <tbody>

            <tr \*ngFor="let c of customers">

             <td>{{c.name}}</td>

            <td>{{c['address']}}</td>

            <td>{{c['mobileNumber']}}</td>

            <td>{{c['appointmentDate']}}</td>

            </tr>

          </tbody>

        </table>

        </div>

    </div>

    </div>

</div>

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content" style="width: 650px" >

      <div class="modal-header">

        <h5 class="modal-title" id="exampleModalLabel">Cancelled Appoinments</h5>

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

      </div>

      <div class="modal-body">

        <table class="table">

          <thead style="background-color: purple; color: white;">

            <tr>

              <th scope="col">Name</th>

              <th scope="col">Address</th>

              <th scope="col">MobileNumber</th>

              <th scope="col">AppointmentDate</th>

            </tr>

          </thead>

          <tbody>

            <tr \*ngFor="let c of deleteappointments">

             <td>{{c.name}}</td>

            <td>{{c['address']}}</td>

            <td>{{c['mobileNumber']}}</td>

            <td>{{c['appointmentDate']}}</td>

            </tr>

          </tbody>

        </table>

      </div>

      <div class="modal-footer">

        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

        <button type="button" class="btn btn-primary">Save changes</button>

      </div>

    </div>

  </div>

</div>

Add Calender using DevExpress:-

* 1. In VS Code open new terminal and type command to add devextream

npm i -g devextreme-cli

npx -p devextreme-cli devextreme add devextreme-angular

Add the import in app.module.ts =

import { DxSchedulerModule } from 'devextreme-angular';

imports: [

DxSchedulerModule,

* 1. Add following code in calender.service.ts
* import { FullAuditedEntityDto, ListResultDto, RestService } from "@abp/ng.core";
* import { Injectable } from "@angular/core";
* export class Appointment {
* text: string;
* startDate: Date
* endDate: Date;
* allDay?: boolean;
* }
* const appointments: Appointment[] = [
* {
* text: '',
* startDate: new Date(''),
* endDate: new Date(''),
* },
* ];
* export interface GetCustomersInput  {
* filterText?: string;
* name?: string;
* address?: string;
* mobileNumber?: string;
* appointmentDateMin?: string;
* appointmentDateMax?: string;
* verificationStatus?: boolean;
* }
* export interface CustomerDto extends FullAuditedEntityDto<string> {
* name: string;
* text:string;
* address?: string;
* mobileNumber: string;
* appointmentDate?: Date;
* verificationStatus?: boolean;
* }

* @Injectable({
* providedIn: 'root'
* })
* export class CalenderService {
* apiName = 'Default';
* getcustincalender= () =>
* {
* return this.restService.request<any, ListResultDto<CustomerDto>>({
* method: 'GET',
* url: '/api/app/customers/Calender',
* },
* { apiName: this.apiName });
* }
* constructor(private restService: RestService) {}
* getAppointments(): Appointment[] {
* return appointments;
* }
* }

3)Then add following code in calender.component.ts

* import { Component, NgModule, OnInit } from '@angular/core';
* import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
* import { DxSchedulerModule } from 'devextreme-angular';
* import { DatePipe } from '@angular/common';
* import { Appointment, CalenderService, CustomerDto } from './calender.service';
* import { getRandomBackgroundColor } from '@abp/ng.components/chart.js';
* import { style } from '@angular/animations';
* @Component({
* selector: 'app-calender',
* templateUrl: './calender.component.html',
* styleUrls: ['./calender.component.scss'],
* providers: [CalenderService],
* })
* export class CalenderComponent {
* time = {hour: 13, minute: 30};
* appointmentsData: Appointment[] = [];
* appointmentsData1: CustomerDto;
* currentDate: Date = new Date(2022, 9, 27);
* constructor(public readonly service: CalenderService,
* private \_apiservice: CalenderService,) {
* //this.appointmentsData1.name="something";
* // this.appointmentsData = service.getAppointments();
* this.getCustomerinCalender();
* //console.log("const",this.appointmentsData1);
* }
* // ngOnInit(): void {
* //  this.getCustomerinCalender()
* // }
* customers!: any[]
* getCustomerinCalender() {
* this.\_apiservice.getcustincalender().subscribe({
* next: (data) => {
* console.log(data)
* if (data.items.length > 0) {
* data.items.forEach(item => {
* let event = new Appointment();
* event.text = item.name;
* event.startDate = new Date(item.appointmentDate);
* event.endDate = new Date(new Date(item.appointmentDate).setMinutes(new Date(item.appointmentDate).getMinutes() + 30));
* this.appointmentsData.push(event);
* });
* this.appointmentsData = [...this.appointmentsData];
* console.log(this.appointmentsData);
* }
* // console.log(data)
* // this.appointmentsData1 = data.items[0];
* // this.appointmentsData1.appointmentDate = new Date('2022-09-27T21:00:00.000Z')
* // console.log("const", this.appointmentsData1);
* }
* }
* )
* //console.log(this.customers ,"customer");
* }
* }
* function getUpcomingappointment() {
* throw new Error('Function not implemented.');
* }

4)Add the following code in HTML

<!-- <abp-page [title]="'::Customers' | abpLocalization"> -->

  <abp-page-toolbar-container class="col">

    <div class="text-lg-end pt-2">

      <div ngbDropdown container="body" class="d-inline-block">

        <button class="btn btn-primary btn-sm " data-toggle="dropdown" aria-haspopup="true"

        ngbDropdownToggle >

          <!-- <i class="bi bi-calendar-heart" aria-hidden="true"></i> -->

          <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="white" class="bi bi-calendar-heart" viewBox="0 0 16 16">

            <path fill-rule="evenodd" d="M4 .5a.5.5 0 0 0-1 0V1H2a2 2 0 0 0-2 2v11a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V3a2 2 0 0 0-2-2h-1V.5a.5.5 0 0 0-1 0V1H4V.5ZM1 14V4h14v10a1 1 0 0 1-1 1H2a1 1 0 0 1-1-1Zm7-6.507c1.664-1.711 5.825 1.283 0 5.132-5.825-3.85-1.664-6.843 0-5.132Z"/>

          </svg>&nbsp;&nbsp;

          {{ 'AbpUi::Appoinments' | abpLocalization }}

        </button>

        <div ngbDropdownMenu>

          <button type="button" data-bs-toggle="modal" data-bs-target="#staticBackdrop" ngbDropdownItem

            (click)="getUpcomingappointment()">

            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-calendar2-check-fill" viewBox="0 0 16 16">

              <path d="M3.5 0a.5.5 0 0 1 .5.5V1h8V.5a.5.5 0 0 1 1 0V1h1a2 2 0 0 1 2 2v11a2 2 0 0 1-2 2H2a2 2 0 0 1-2-2V3a2 2 0 0 1 2-2h1V.5a.5.5 0 0 1 .5-.5zm9.954 3H2.545c-.3 0-.545.224-.545.5v1c0 .276.244.5.545.5h10.91c.3 0 .545-.224.545-.5v-1c0-.276-.244-.5-.546-.5zm-2.6 5.854a.5.5 0 0 0-.708-.708L7.5 10.793 6.354 9.646a.5.5 0 1 0-.708.708l1.5 1.5a.5.5 0 0 0 .708 0l3-3z"/>

            </svg>&nbsp;&nbsp;

            {{ 'AbpUi::Upcoming Appoinments' | abpLocalization }}

          </button>

          <!-- <button ngbDropdownItem \*abpPermission="'AppoinmentSchelulingProject.Customers.Delete'" (click)="getDeletedData()">

            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-calendar-x-fill" viewBox="0 0 16 16">

              <path d="M4 .5a.5.5 0 0 0-1 0V1H2a2 2 0 0 0-2 2v1h16V3a2 2 0 0 0-2-2h-1V.5a.5.5 0 0 0-1 0V1H4V.5zM16 14V5H0v9a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2zM6.854 8.146 8 9.293l1.146-1.147a.5.5 0 1 1 .708.708L8.707 10l1.147 1.146a.5.5 0 0 1-.708.708L8 10.707l-1.146 1.147a.5.5 0 0 1-.708-.708L7.293 10 6.146 8.854a.5.5 0 1 1 .708-.708z"/>

            </svg>&nbsp;&nbsp;

            {{ 'AbpUi::Cancelled Appoinments' | abpLocalization }}

          </button> -->

          <button type="button"  data-bs-toggle="modal" data-bs-target="#exampleModal" ngbDropdownItem (click)="getDeletedData()">

            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-calendar-x-fill" viewBox="0 0 16 16">

              <path d="M4 .5a.5.5 0 0 0-1 0V1H2a2 2 0 0 0-2 2v1h16V3a2 2 0 0 0-2-2h-1V.5a.5.5 0 0 0-1 0V1H4V.5zM16 14V5H0v9a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2zM6.854 8.146 8 9.293l1.146-1.147a.5.5 0 1 1 .708.708L8.707 10l1.147 1.146a.5.5 0 0 1-.708.708L8 10.707l-1.146 1.147a.5.5 0 0 1-.708-.708L7.293 10 6.146 8.854a.5.5 0 1 1 .708-.708z"/>

            </svg>&nbsp;&nbsp;

            {{ 'AbpUi::Cancelled Appoinments' | abpLocalization }}

          </button>

      </div>

      </div>

      <button \*abpPermission="'AppoinmentSchelulingProject.Customers.Create'" class="btn btn-primary btn-sm mx-3"

        type="button" (click)="create()">

        <i class="fa fa-plus me-1" aria-hidden="true"></i>

        {{ '::NewCustomer' | abpLocalization }}

      </button>

    </div>

  </abp-page-toolbar-container>

<!-- Button trigger modal -->

  <div class="card">

    <div class="card-body">

      <abp-page [title]="'::Customers' | abpLocalization"></abp-page>

<!------------- Search and Filter ----------------->

      <div class="w-30" style="margin-left: 55%; margin-top: -6%;">

      <abp-advanced-entity-filters localizationSourceName="" [list]="list"  >

        <!-- <abp-advanced-entity-filters-form>

          <form #filterForm (keyup.enter)="list.get()" >

            <div class="row" >

              <div class="col-12 col-sm-auto" >

                <div class="mb-3">

                  <label class="form-label" for="nameFilter">

                    {{ '::Name' | abpLocalization }}

                  </label>

                  <input id="nameFilter" class="form-control" [(ngModel)]="filters.name"

                    [ngModelOptions]="{ standalone: true }" />

                </div>

              </div>

              <div class="col-12 col-sm-auto">

                <div class="mb-3">

                  <label class="form-label" for="addressFilter">

                    {{ '::Address' | abpLocalization }}

                  </label>

                  <input id="addressFilter" class="form-control" [(ngModel)]="filters.address"

                    [ngModelOptions]="{ standalone: true }" />

                </div>

              </div>

              <div class="col-12 col-sm-auto">

                <div class="mb-3">

                  <label class="form-label" for="mobileNumberFilter">

                    {{ '::MobileNumber' | abpLocalization }}

                  </label>

                  <input id="mobileNumberFilter" class="form-control" [(ngModel)]="filters.mobileNumber"

                    [ngModelOptions]="{ standalone: true }" />

                </div>

              </div>

              <div class="col-12 col-sm-auto">

                <div class="mb-3">

                  <label class="form-label" for="minAppointmentDate">

                    {{ '::MinAppointmentDate' | abpLocalization }}

                  </label>

                  <div class="input-group" validationTarget>

                    <input id="minAppointmentDate" class="form-control" #minAppointmentDateInput

                      #minAppointmentDateDatepicker="ngbDatepicker" name="minAppointmentDate" ngbDatepicker container="body"

                      readonly (click)="minAppointmentDateDatepicker.toggle()"

                      (keyup.space)="minAppointmentDateDatepicker.toggle()" [(ngModel)]="filters.appointmentDateMin"

                      [ngModelOptions]="{ standalone: true }" />

                    <div \*ngIf="minAppointmentDateInput.value" class="input-group-append">

                      <button class="btn btn-link px-2" type="button" [title]="'AbpUi::Clear' | abpLocalization" (click)="

                          minAppointmentDateDatepicker.writeValue(null);

                          minAppointmentDateDatepicker.manualDateChange(null, true)

                        ">

                        <i class="fa fa-times" aria-hidden="true"></i>

                      </button>

                    </div>

                  </div>

                </div>

              </div>

              <div class="col-12 col-sm-auto">

                <div class="mb-3">

                  <label class="form-label" for="maxAppointmentDate">

                    {{ '::MaxAppointmentDate' | abpLocalization }}

                  </label>

                  <div class="input-group" validationTarget>

                    <input id="maxAppointmentDate" class="form-control" #maxAppointmentDateInput

                      #maxAppointmentDateDatepicker="ngbDatepicker" name="maxAppointmentDate" ngbDatepicker readonly

                      (click)="maxAppointmentDateDatepicker.toggle()" (keyup.space)="maxAppointmentDateDatepicker.toggle()"

                      [(ngModel)]="filters.appointmentDateMax" [ngModelOptions]="{ standalone: true }" />

                    <div \*ngIf="maxAppointmentDateInput.value" class="input-group-append">

                      <button class="btn btn-link px-2" type="button" [title]="'AbpUi::Clear' | abpLocalization" (click)="

                          maxAppointmentDateDatepicker.writeValue(null);

                          maxAppointmentDateDatepicker.manualDateChange(null, true)

                        ">

                        <i class="fa fa-times" aria-hidden="true"></i>

                      </button>

                    </div>

                  </div>

                </div>

              </div>

              <div class="col-12 col-sm-auto">

                <div class="mb-3">

                  <label class="form-label" for="verificationStatusFilter">

                    {{ '::VerificationStatus' | abpLocalization }}

                  </label>

                  <select name="verificationStatus" id="verificationStatusFilter" class="form-select form-control"

                    [(ngModel)]="filters.verificationStatus" [ngModelOptions]="{ standalone: true }" (change)="list.get()">

                    <option [ngValue]=""></option>

                    <option [ngValue]="true">{{ 'AbpUi::Yes' | abpLocalization }}</option>

                    <option [ngValue]="false">{{ 'AbpUi::No' | abpLocalization }}</option>

                  </select>

                </div>

              </div>

              <div class="col-12 col-sm-auto align-self-end mb-3">

                <div class="row">

                  <div class="col-6 col-sm-auto d-grid">

                    <button type="button" class="btn btn-outline-primary" (click)="clearFilters()">

                      <span>{{ 'AbpUi::Clear' | abpLocalization }}</span>

                    </button>

                  </div>

                  <div class="col-6 col-sm-auto d-grid">

                    <button type="button" class="btn btn-primary" (click)="list.get()">

                      <span>{{ 'AbpUi::Refresh' | abpLocalization }}</span>

                    </button>

                  </div>

                </div>

              </div>

            </div>

          </form>

        </abp-advanced-entity-filters-form> -->

      </abp-advanced-entity-filters>

    </div>

<!-- ----------------------------------------------------- -->

      <ngx-datatable [rows]="data.items" [count]="data.totalCount" [list]="list" default>

        <ngx-datatable-column name="{{ '::Name' | abpLocalization }}" prop="name">

          <ng-template let-row="row" ngx-datatable-cell-template>

            {{ row.name }}

          </ng-template>

        </ngx-datatable-column>

        <ngx-datatable-column name="{{ '::Address' | abpLocalization }}" prop="address">

          <ng-template let-row="row" ngx-datatable-cell-template>

            {{ row.address }}

          </ng-template>

        </ngx-datatable-column>

        <ngx-datatable-column name="{{ '::MobileNumber' | abpLocalization }}" prop="mobileNumber">

          <ng-template let-row="row" ngx-datatable-cell-template>

            {{ row.mobileNumber }}

          </ng-template>

        </ngx-datatable-column>

        <ngx-datatable-column name="{{ '::AppointmentDate' | abpLocalization }}" prop="appointmentDate">

          <ng-template let-row="row" ngx-datatable-cell-template>

            {{ row.appointmentDate | date}}

          </ng-template>

        </ngx-datatable-column>

        <ngx-datatable-column name="{{ '::VerificationStatus' | abpLocalization }}" prop="verificationStatus">

          <ng-template let-row="row" ngx-datatable-cell-template>

            <ng-template [ngIf]="row.verificationStatus" [ngIfThen]="yes" [ngIfElse]="no"></ng-template>

            <ng-template #yes>

              <div class="text-success">

                <span [title]="'AbpUi::Yes' | abpLocalization">

                  <i class="fa fa-check" aria-hidden="true"></i>

                </span>

              </div>

            </ng-template>

            <ng-template #no>

              <div class="text-danger">

                <span [title]="'AbpUi::No' | abpLocalization">

                  <i class="fa fa-times" aria-hidden="true"></i>

                </span>

              </div>

            </ng-template>

          </ng-template>

        </ngx-datatable-column>

        <ngx-datatable-column [name]="'AbpUi::Actions' | abpLocalization" [maxWidth]="150" [width]="150"

        [sortable]="false">

        <ng-template let-row="row" let-i="rowIndex" ngx-datatable-cell-template>

          <div ngbDropdown container="body" class="d-inline-block">

           <button class="bg-white border-0 " data-toggle="dropdown" aria-haspopup="true"

              ngbDropdownToggle>

              <!-- <i class="fa fa-cog me-1" aria-hidden="true"></i>

              {{ 'AbpUi::Actions' | abpLocalization }} -->

              <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-three-dots" viewBox="0 0 16 16">

                <path d="M3 9.5a1.5 1.5 0 1 1 0-3 1.5 1.5 0 0 1 0 3zm5 0a1.5 1.5 0 1 1 0-3 1.5 1.5 0 0 1 0 3zm5 0a1.5 1.5 0 1 1 0-3 1.5 1.5 0 0 1 0 3z"/>

              </svg>

            </button>

            <!-- <i class="bi bi-three-dots"></i>

            <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-three-dots" viewBox="0 0 16 16">

              <path d="M3 9.5a1.5 1.5 0 1 1 0-3 1.5 1.5 0 0 1 0 3zm5 0a1.5 1.5 0 1 1 0-3 1.5 1.5 0 0 1 0 3zm5 0a1.5 1.5 0 1 1 0-3 1.5 1.5 0 0 1 0 3z"/>

            </svg> -->

            <div ngbDropdownMenu style="width:10px">

              <button ngbDropdownItem \*abpPermission="'AppoinmentSchelulingProject.Customers.Edit'"

                (click)="update(row)">

                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-pencil-fill" viewBox="0 0 16 16">

                  <path d="M12.854.146a.5.5 0 0 0-.707 0L10.5 1.793 14.207 5.5l1.647-1.646a.5.5 0 0 0 0-.708l-3-3zm.646 6.061L9.793 2.5 3.293 9H3.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.5h.5a.5.5 0 0 1 .5.5v.207l6.5-6.5zm-7.468 7.468A.5.5 0 0 1 6 13.5V13h-.5a.5.5 0 0 1-.5-.5V12h-.5a.5.5 0 0 1-.5-.5V11h-.5a.5.5 0 0 1-.5-.5V10h-.5a.499.499 0 0 1-.175-.032l-.179.178a.5.5 0 0 0-.11.168l-2 5a.5.5 0 0 0 .65.65l5-2a.5.5 0 0 0 .168-.11l.178-.178z"/>

                </svg>&nbsp;&nbsp;

                {{ 'AbpUi::Edit' | abpLocalization }}

              </button>

              <button ngbDropdownItem \*abpPermission="'AppoinmentSchelulingProject.Customers.Delete'"

                (click)="delete(row)">

                <svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="indigo" class="bi bi-person-x-fill" viewBox="0 0 16 16">

                  <path fill-rule="evenodd" d="M1 14s-1 0-1-1 1-4 6-4 6 3 6 4-1 1-1 1H1zm5-6a3 3 0 1 0 0-6 3 3 0 0 0 0 6zm6.146-2.854a.5.5 0 0 1 .708 0L14 6.293l1.146-1.147a.5.5 0 0 1 .708.708L14.707 7l1.147 1.146a.5.5 0 0 1-.708.708L14 7.707l-1.146 1.147a.5.5 0 0 1-.708-.708L13.293 7l-1.147-1.146a.5.5 0 0 1 0-.708z"/>

                </svg>&nbsp;&nbsp;

                {{ 'AbpUi::Delete' | abpLocalization }}

              </button>

            </div>

          </div>

        </ng-template>

      </ngx-datatable-column>

      </ngx-datatable>

    </div>

  </div>

<!-- </abp-page> -->

<abp-modal [busy]="isModalBusy" [(visible)]="isModalOpen">

  <ng-template #abpHeader>

    <h3>

      {{ (selected ? 'AbpUi::Edit' : '::NewCustomer') | abpLocalization }}

    </h3>

  </ng-template>

  <ng-template #abpBody>

    <form [formGroup]="form" id="customer" (ngSubmit)="submitForm()" alre validateOnSubmit>

      <div class="mt-2 fade-in-top">

        <div class="mb-3">

          <label class="form-label" for="customer-name"> {{ '::Name' | abpLocalization }} \* </label>

          <input id="customer-name" formControlName="name" class="form-control" autofocus />

        </div>

        <div class="mb-3">

          <label class="form-label" for="customer-address">

            {{ '::Address' | abpLocalization }}

          </label>

          <input id="customer-address" formControlName="address" class="form-control" />

        </div>

        <div class="mb-3">

          <label class="form-label" for="customer-mobile-number">

            {{ '::MobileNumber' | abpLocalization }} \*

          </label>

          <input id="customer-mobile-number" formControlName="mobileNumber" class="form-control" />

        </div>

        <div class="mb-3">

          <label class="form-label" for="customer-appointment-date">

            {{ '::AppointmentDate' | abpLocalization }}

          </label>

          <div class="input-group" validationTarget>

            <input type="datetime-local" id="customer-appointment-date" formControlName="appointmentDate" class="form-control"

            min="{{currentdate|date:'yyyy-MM-ddThh:mm'}}"/>

          </div>

        </div>

        <div class="mb-3">

          <label class="form-label" for="customer-appointment-date">

            {{ '::AppointmentDate' | abpLocalization }}

          </label>

          <div class="input-group" >

            <input type="date" id="customer-appointment-date"

              />

          </div>

        </div>

        <div class="form-check mb-3">

          <input type="checkbox" id="customer-verification-status" formControlName="verificationStatus"

            class="form-check-input" />

          <label for="customer-verification-status" class="form-check-label">

            {{ '::VerificationStatus' | abpLocalization }}

          </label>

        </div>

      </div>

    </form>

  </ng-template>

  <ng-template #abpFooter>

    <button type="button" class="btn btn-secondary" abpClose>

      {{ 'AbpUi::Cancel' | abpLocalization }}

    </button>

    <abp-button iconClass="fa fa-check" buttonType="submit"  formName="customer">

      {{ 'AbpUi::Save' | abpLocalization }}

    </abp-button>

  </ng-template>

</abp-modal>

<!-- Modal for upcoming-->

<div class="modal fade" id="staticBackdrop" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"

  aria-labelledby="staticBackdropLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content " style="width: 650px" >

      <div class="modal-header">

        <h5 class="modal-title" id="staticBackdropLabel">Upcoming Appoinments</h5>

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

      </div>

      <div class="modal-body">

        <table class="table">

          <thead style="background-color: purple; color: white;">

            <tr>

              <th scope="col">Name</th>

              <th scope="col">Address</th>

              <th scope="col">MobileNumber</th>

              <th scope="col">AppointmentDate</th>

            </tr>

          </thead>

          <tbody>

            <tr \*ngFor="let c of customers">

             <td>{{c.name}}</td>

            <td>{{c['address']}}</td>

            <td>{{c['mobileNumber']}}</td>

            <td>{{c.appointmentDate | date }}</td>

            </tr>

          </tbody>

        </table>

        </div>

    </div>

    </div>

</div>

<!-- Modal -->

<div class="modal fade" id="exampleModal" tabindex="-1" aria-labelledby="exampleModalLabel" aria-hidden="true">

  <div class="modal-dialog">

    <div class="modal-content" style="width: 650px" >

      <div class="modal-header">

        <h5 class="modal-title" id="exampleModalLabel">Cancelled Appoinments</h5>

        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

      </div>

      <div class="modal-body">

        <table class="table">

          <thead style="background-color: purple; color: white;">

            <tr>

              <th scope="col">Name</th>

              <th scope="col">Address</th>

              <th scope="col">MobileNumber</th>

              <th scope="col">AppointmentDate</th>

            </tr>

          </thead>

          <tbody>

            <tr \*ngFor="let c of deleteappointments">

             <td>{{c.name}}</td>

            <td>{{c['address']}}</td>

            <td>{{c['mobileNumber']}}</td>

            <td>{{c.appointmentDate | date }}</td>

            </tr>

          </tbody>

        </table>

      </div>

      <div class="modal-footer">

        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>

        <button type="button" class="btn btn-primary">Save changes</button>

      </div>

    </div>

  </div>

</div>

<router-outlet></router-outlet>

Routing steps for New Menu

1. Create a new component using ng g c component-name
2. Add 3 files

* App-route.provider.ts
* App-routing.module.ts
* App.module.ts

3.In app.module.ts Copy code from existing file and add AppRountingModule in imports.

4.In app-routing.module.ts Copy code from existing file and give the path for routing.

const routes: Routes = [

  {

    path: '',

    component: CalenderComponent,

  },

];

5.In app-route.provider.ts copy code from existing file and give the const name

export const CALENDER\_ROUTE\_PROVIDER = [

  { provide: APP\_INITIALIZER, useFactory: configureRoutes, deps: [RoutesService], multi: true },

];

6.In function configureroutes add the following code

function configureRoutes(routes: RoutesService) {

  return () => {

    routes.add([

      {

        path: '/calender',

        iconClass: 'fas fa-file-alt',

        name: '::Calender',

        layout: eLayoutType.application,

        //requiredPolicy: 'AppoinmentSchelulingProject.MyCalenderss',

      },

    ]);

  };

}

It is for displaying main menu..

7.Add the const in provider of app.module.ts file

 providers: [

    APP\_ROUTE\_PROVIDER,

    CUSTOMERS\_CUSTOMER\_ROUTE\_PROVIDER,

    CALENDER\_ROUTE\_PROVIDER,

  ],