**Bahria University,**

**Karachi Campus**



**LAB EXPERIMENT NO.**

**\_10**

**LIST OF TASKS**

|  |  |
| --- | --- |
| TASK NO | **OBJECTIVE** |
| 01 | Implement the microservice for user profile management, incorporating registration, login, and profile updates using ASP.NET Core Identity. After implementation, test the microservice endpoints using Postman to ensure functionality and seamless interaction. |
|  |  |

**Submitted On:**

**Date: 09/May/2024**

**TASK # 1:** Implement the microservice for user profile management, incorporating registration, login, and profile updates using ASP.NET Core Identity. After implementation, test the microservice endpoints using Postman to ensure functionality and seamless interaction.

using Microsoft.AspNetCore.Identity;

using Microsoft.EntityFrameworkCore;

using UserProfileManagement.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllers();

builder.Services.AddDbContext<UserProfileContext>(options =>

options.UseInMemoryDatabase("UserProfileDB"));

builder.Services.AddIdentity<ApplicationUser, IdentityRole>()

.AddEntityFrameworkStores<UserProfileContext>()

.AddDefaultTokenProviders();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace UserProfileManagement.Models

{

public class UserProfileContext : IdentityDbContext<ApplicationUser>

{

public UserProfileContext(DbContextOptions<UserProfileContext> options) : base(options) { }

} }

using Microsoft.AspNetCore.Identity;

namespace UserProfileManagement.Models

{

public class ApplicationUser : IdentityUser

{

public string Name { get; set; }

public string Designation { get; set; }

} }

**USERPROFILE CONTROLLER:**

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using UserProfileManagement.Models;

namespace UserProfileManagement.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UserProfileController : ControllerBase

{

private readonly UserManager<ApplicationUser> \_userManager;

private readonly SignInManager<ApplicationUser> \_signInManager;

public UserProfileController(UserManager<ApplicationUser> userManager, SignInManager<ApplicationUser> signInManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

}

[HttpPost("register")]

public async Task<IActionResult> Register([FromBody] RegisterModel model)

{

var user = new ApplicationUser

{

UserName = model.Email,

Email = model.Email,

Name = model.Name,

Designation = model.Designation

};

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

return Ok(new { Message = "User registered successfully" });

}

return BadRequest(result.Errors);

}

[HttpPost("login")]

public async Task<IActionResult> Login([FromBody] LoginModel model)

{

var result = await \_signInManager.PasswordSignInAsync(model.Email, model.Password, false, false);

if (result.Succeeded)

{

return Ok(new { Message = "User logged in successfully" });

}

return Unauthorized();

}

[HttpPut("{id}")]

public async Task<IActionResult> UpdateUserProfile(string id, [FromBody] UpdateProfileModel model)

{

var user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

user.Name = model.Name;

user.Designation = model.Designation;

user.Email = model.Email;

user.UserName = model.Email;

var result = await \_userManager.UpdateAsync(user);

if (result.Succeeded)

{

return Ok(new { Message = "User profile updated successfully" });

}

return BadRequest(result.Errors);

} }

public class RegisterModel

{

public string Name { get; set; }

public string Designation { get; set; }

public string Email { get; set; }

public string Password { get; set; }

}

public class LoginModel

{

public string Email { get; set; }

public string Password { get; set; }

}

public class UpdateProfileModel

{

public string Name { get; set; }

public string Designation { get; set; }

public string Email { get; set; }

} }

**A screenshot of a computer

Description automatically generated A screenshot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**Model:**

namespace CC\_LAB\_9\_Product.Model

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public double Price { get; set; }

} }

**Controller:**

using CC\_LAB\_9\_Product.Model;

using CC\_LAB\_9\_Product.Repository;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Transactions;

namespace CC\_LAB\_9\_Product.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductRepository \_productRepository;

public ProductController(IProductRepository productRepository)

{

\_productRepository = productRepository;

}

[HttpGet]

public IActionResult Get()

{

var products = \_productRepository.GetProducts();

return new OkObjectResult(products);

}

[HttpGet("{id}", Name = "Get")]

public IActionResult Get(int id)

{

var product = \_productRepository.GetProductByID(id);

return new OkObjectResult(product);

}

[HttpPost]

public IActionResult Post([FromBody] Product product)

{

using (var scope = new TransactionScope())

{ \_productRepository.InsertProduct(product); scope.Complete();

return CreatedAtAction(nameof(Get), new { id = product.Id },

product);

} }

[HttpPut]

public IActionResult Put([FromBody] Product product)

{

if (product != null)

{

using (var scope = new TransactionScope())

{

\_productRepository.UpdateProduct(product); scope.Complete();

return new OkResult();

} }

return new NoContentResult();

}

[HttpDelete("{id}")]

public IActionResult Delete(int id)

{

\_productRepository.DeleteProduct(id); return new OkResult();

}} }

**REPOSITORIES:**

using CC\_LAB\_9\_Product.DbContexts;

using CC\_LAB\_9\_Product.Model;

using Microsoft.EntityFrameworkCore;

namespace CC\_LAB\_9\_Product.Repository

{

public class ProductRepository : IProductRepository

{

private readonly ProductContext \_dbContext;

public ProductRepository(ProductContext dbContext)

{

\_dbContext = dbContext;

}

public void DeleteProduct(int productId)

{

var product = \_dbContext.Products.Find(productId);

\_dbContext.Products.Remove(product);

Save();

}

public Product GetProductByID(int productId)

{

return \_dbContext.Products.Find(productId);

}

public IEnumerable<Product> GetProducts()

{

return \_dbContext.Products.ToList();

}

public void InsertProduct(Product product)

{

\_dbContext.Add(product);

Save();

}

public void UpdateProduct(Product product)

{

\_dbContext.Entry(product).State = EntityState.Modified;

Save();

}

public void Save()

{

\_dbContext.SaveChanges();

}

public void save()

{

throw new NotImplementedException();

} } }

using CC\_LAB\_9\_Product.Model;

namespace CC\_LAB\_9\_Product.Repository

{

public interface IProductRepository

{

Product GetProductByID(int productid);

IEnumerable<Product> GetProducts();

void DeleteProduct(int productid);

void InsertProduct(Product product);

void UpdateProduct(Product product);

void save();

} }

**CONTROLLER:**

using CC\_LAB\_9\_Product.Model;

using CC\_LAB\_9\_Product.Repository;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Transactions;

namespace CC\_LAB\_9\_Product.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class ProductController : ControllerBase

{

private readonly IProductRepository \_productRepository;

public ProductController(IProductRepository productRepository)

{

\_productRepository = productRepository;

}

[HttpGet]

public IActionResult Get()

{

var products = \_productRepository.GetProducts();

return new OkObjectResult(products);

}

[HttpGet("{id}", Name = "Get")]

public IActionResult Get(int id)

{

var product = \_productRepository.GetProductByID(id);

return new OkObjectResult(product);

}

[HttpPost]

public IActionResult Post([FromBody] Product product)

{

using (var scope = new TransactionScope())

{

\_productRepository.InsertProduct(product); scope.Complete();

return CreatedAtAction(nameof(Get), new { id = product.Id },

product);

} }

[HttpPut]

public IActionResult Put([FromBody] Product product)

{

if (product != null)

{

using (var scope = new TransactionScope())

{ \_productRepository.UpdateProduct(product); scope.Complete();

return new OkResult();

}}

return new NoContentResult();

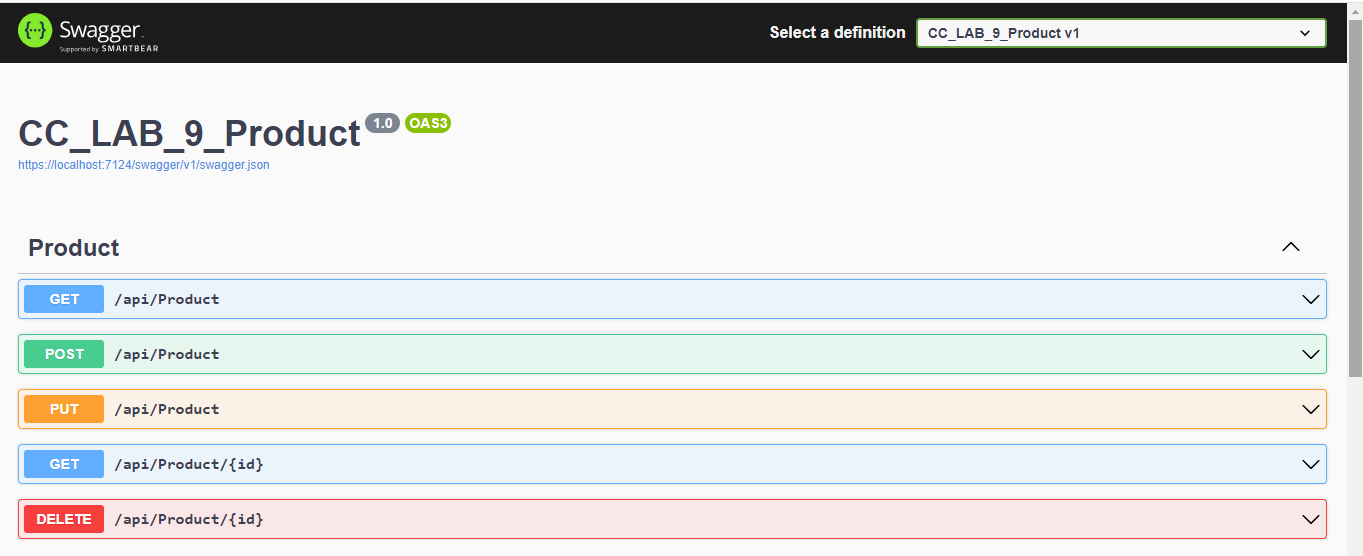
}

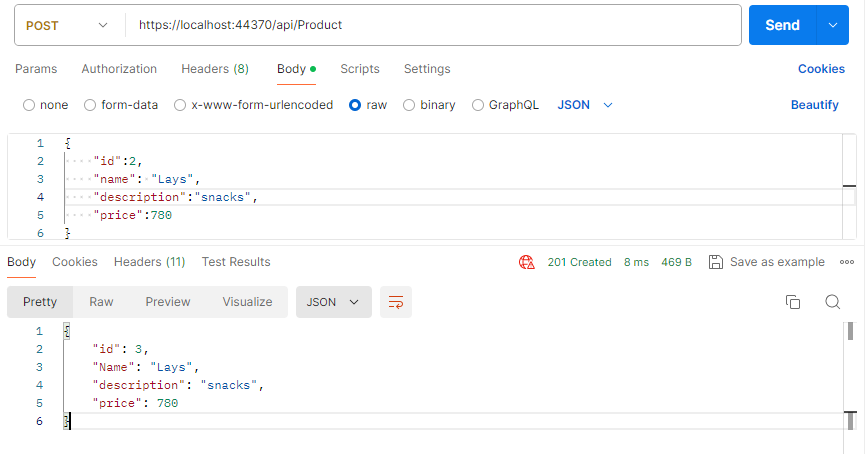
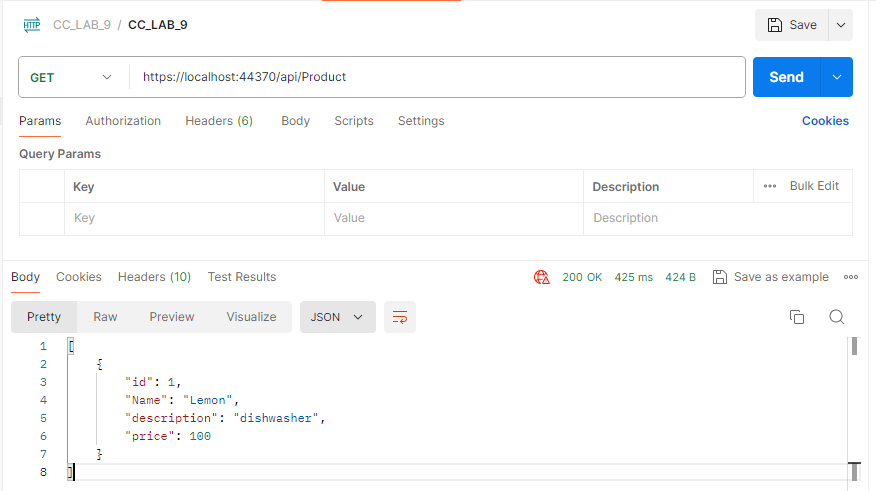
[HttpDelete("{id}")]

public IActionResult Delete(int id)

{ \_productRepository.DeleteProduct(id); return new OkResult();

} } }



A screenshot of a computer

Description automatically generated