**Step 1: Create a New ASP.NET Core Web API Project**

1. **Open Visual Studio** (or your preferred IDE).
2. **Create a New Project**:
   * Choose **ASP.NET Core Web API**.
   * Click **Next**.
3. **Configure Your Project**:
   * Name your project (e.g., Scheduler).
   * Choose a location for your project.
   * Click **Next**.
4. **Configure the New ASP.NET Core Web API**:
   * Choose **.NET 6.0 (or later)** as the target framework.
   * Make sure to check the option **Enable OpenAPI Support** (this will add Swagger).
   * Click **Create**.

**Step 2: Define Your Service Interface and Implementation**

1. **Create the Service Interface**:
   * Right-click on the Services folder (create it if it doesn’t exist).
   * Add a new class named IService.cs:

csharp

Copy code

namespace Scheduler.Services.Interfaces

{

public interface IService

{

string GetMessage();

void UpdateMessage(string message); // Method to update the message

}

}

1. **Create the Service Implementation**:
   * Add a new class named Service.cs in the same Services folder:

csharp

Copy code

using Scheduler.Services.Interfaces;

namespace Scheduler.Services

{

public class Service : IService

{

private string \_message = "Initial Message";

public string GetMessage()

{

return \_message;

}

public void UpdateMessage(string message)

{

\_message = message; // Update the message

}

}

}

**Step 3: Create the Controller**

1. **Create the Controller**:
   * Right-click on the Controllers folder.
   * Add a new class named HomeController.cs:

csharp

Copy code

using Microsoft.AspNetCore.Mvc;

using Scheduler.Services.Interfaces;

namespace Scheduler.Controllers

{

[ApiController]

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

public class HomeController : ControllerBase

{

private readonly IService \_service;

public HomeController(IService service)

{

\_service = service;

}

[HttpGet]

public IActionResult GetHello()

{

return Ok(\_service.GetMessage()); // Return the latest message

}

}

}

**Step 4: Implement the Background Service**

1. **Create the Background Service**:
   * Right-click on the Services folder.
   * Add a new class named ScheduledService.cs:

csharp

Copy code

using Microsoft.Extensions.Hosting;

using System;

using System.Threading;

using System.Threading.Tasks;

using Scheduler.Services.Interfaces;

namespace Scheduler.Services

{

public class ScheduledService : BackgroundService

{

private readonly IService \_service;

public ScheduledService(IService service)

{

\_service = service;

}

protected override async Task ExecuteAsync(CancellationToken stoppingToken)

{

while (!stoppingToken.IsCancellationRequested)

{

// Update the message every minute

\_service.UpdateMessage($"Updated Message at {DateTime.Now}");

await Task.Delay(TimeSpan.FromMinutes(1), stoppingToken); // Wait for 1 minute

}

}

}

}

**Step 5: Update the Program.cs File**

1. **Open Program.cs**:
   * Register your services and background service.

csharp

Copy code

using Scheduler.Services;

using Scheduler.Services.Interfaces;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Register your services

builder.Services.AddScoped<IService, Service>();

builder.Services.AddHostedService<ScheduledService>(); // Register the background service

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**Step 6: Run the Application**

1. **Start the Application**:
   * Press F5 or click on the green play button to run the application.
   * You should see the Swagger UI open in your browser.

**Step 7: Test the API**

1. **Check the API Endpoint**:
   * In the Swagger UI, expand the Home controller section.
   * Click on the GET method for /api/home.
   * Click **Try it out** and then **Execute**.
   * You should see the initial message.
2. **Wait for Updates**:
   * After about a minute, refresh the Swagger UI and try the GET method again.
   * You should see an updated message reflecting the time it was updated.

**Conclusion**

You now have a fully functional ASP.NET Core application that includes:

* A background service updating a message every minute.
* An API controller that exposes this message via an endpoint.
* Swagger support for easy testing and documentation of your API.