**Create a new Web API project**

dotnet new webapi -n EventManagementAPI

cd EventManagementAPI

**2️⃣ Create the Event Model**

Create a folder Models and add Event.cs:

namespace EventManagementAPI.Models

{

public class Event

{

public int EventId { get; set; }

public string Name { get; set; }

public DateTime Date { get; set; }

public string Location { get; set; }

}

}

**3️⃣ Create EventService for in-memory CRUD operations**

Create a folder Services and add EventService.cs:

using EventManagementAPI.Models;

using System.Collections.Generic;

using System.Linq;

namespace EventManagementAPI.Services

{

public class EventService

{

private readonly List<Event> events;

public EventService()

{

events = new List<Event>

{

new Event { EventId = 1, Name = "Event 1", Date = DateTime.Now.AddDays(7), Location = "Location 1" },

new Event { EventId = 2, Name = "Event 2", Date = DateTime.Now.AddDays(14), Location = "Location 2" },

new Event { EventId = 3, Name = "Event 3", Date = DateTime.Now.AddDays(21), Location = "Location 3" }

};

}

public List<Event> GetAllEvents() => events;

public Event GetEventById(int eventId) => events.FirstOrDefault(e => e.EventId == eventId);

public void CreateEvent(Event newEvent)

{

int nextId = events.Any() ? events.Max(e => e.EventId) + 1 : 1;

newEvent.EventId = nextId;

events.Add(newEvent);

}

public bool UpdateEvent(int eventId, Event updatedEvent)

{

var existingEvent = GetEventById(eventId);

if (existingEvent == null) return false;

existingEvent.Name = updatedEvent.Name;

existingEvent.Date = updatedEvent.Date;

existingEvent.Location = updatedEvent.Location;

return true;

}

public bool DeleteEvent(int eventId)

{

var existingEvent = GetEventById(eventId);

if (existingEvent == null) return false;

events.Remove(existingEvent);

return true;

}

}

}

**4️⃣ Register the service in Program.cs**

using EventManagementAPI.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddSingleton<EventService>();

builder.Services.AddControllers();

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure middleware

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

**5️⃣ Create EventController**

Create a folder Controllers and add EventController.cs:

using EventManagementAPI.Models;

using EventManagementAPI.Services;

using Microsoft.AspNetCore.Mvc;

namespace EventManagementAPI.Controllers

{

[ApiController]

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

public class EventController : ControllerBase

{

private readonly EventService \_eventService;

public EventController(EventService eventService)

{

\_eventService = eventService;

}

// GET: /api/event

[HttpGet]

public IActionResult GetAllEvents()

{

var events = \_eventService.GetAllEvents();

if (events.Count == 0)

return NoContent(); // 204 No Content

return Ok(events); // 200 OK

}

// GET: /api/event/{id}

[HttpGet("{id}")]

public IActionResult GetEventById(int id)

{

var evt = \_eventService.GetEventById(id);

if (evt == null) return NotFound(); // 404 Not Found

return Ok(evt); // 200 OK

}

// POST: /api/event

[HttpPost]

public IActionResult CreateEvent([FromBody] Event newEvent)

{

if (newEvent == null || string.IsNullOrWhiteSpace(newEvent.Name) || string.IsNullOrWhiteSpace(newEvent.Location))

return BadRequest("Invalid event data."); // 400 Bad Request

\_eventService.CreateEvent(newEvent);

return CreatedAtAction(nameof(GetEventById), new { id = newEvent.EventId }, newEvent); // 201 Created

}

// PUT: /api/event/{id}

[HttpPut("{id}")]

public IActionResult UpdateEvent(int id, [FromBody] Event updatedEvent)

{

if (updatedEvent == null || string.IsNullOrWhiteSpace(updatedEvent.Name) || string.IsNullOrWhiteSpace(updatedEvent.Location))

return BadRequest("Invalid event data."); // 400 Bad Request

var success = \_eventService.UpdateEvent(id, updatedEvent);

if (!success) return NotFound(); // 404 Not Found

return NoContent(); // 204 No Content

}

// DELETE: /api/event/{id}

[HttpDelete("{id}")]

public IActionResult DeleteEvent(int id)

{

var success = \_eventService.DeleteEvent(id);

if (!success) return NotFound(); // 404 Not Found

return NoContent(); // 204 No Content

}

}

}

**6️⃣ Test the API**

* Run the project:

dotnet restore

dotnet run

* Open Swagger at https://localhost:5001/swagger or http://localhost:8080/swagger/index.html.
* Test all endpoints (GET, POST, PUT, DELETE) with different scenarios to check status codes.