SIEMENS Healthineers

Tancuj s káčerom

Tiež čakáš na začiatok prezentácie?

Jan Jakubcik, 2023

SIEMENS Healthineers

Minimal Code Paradigm

How to keep Kiss and Dry

DotnetMinimalApi project example

https://github.com/Johnz86/DotnetMinimalApi

What we will cover:

Architecture and usage

Minimal API boilerplate

Project Templates

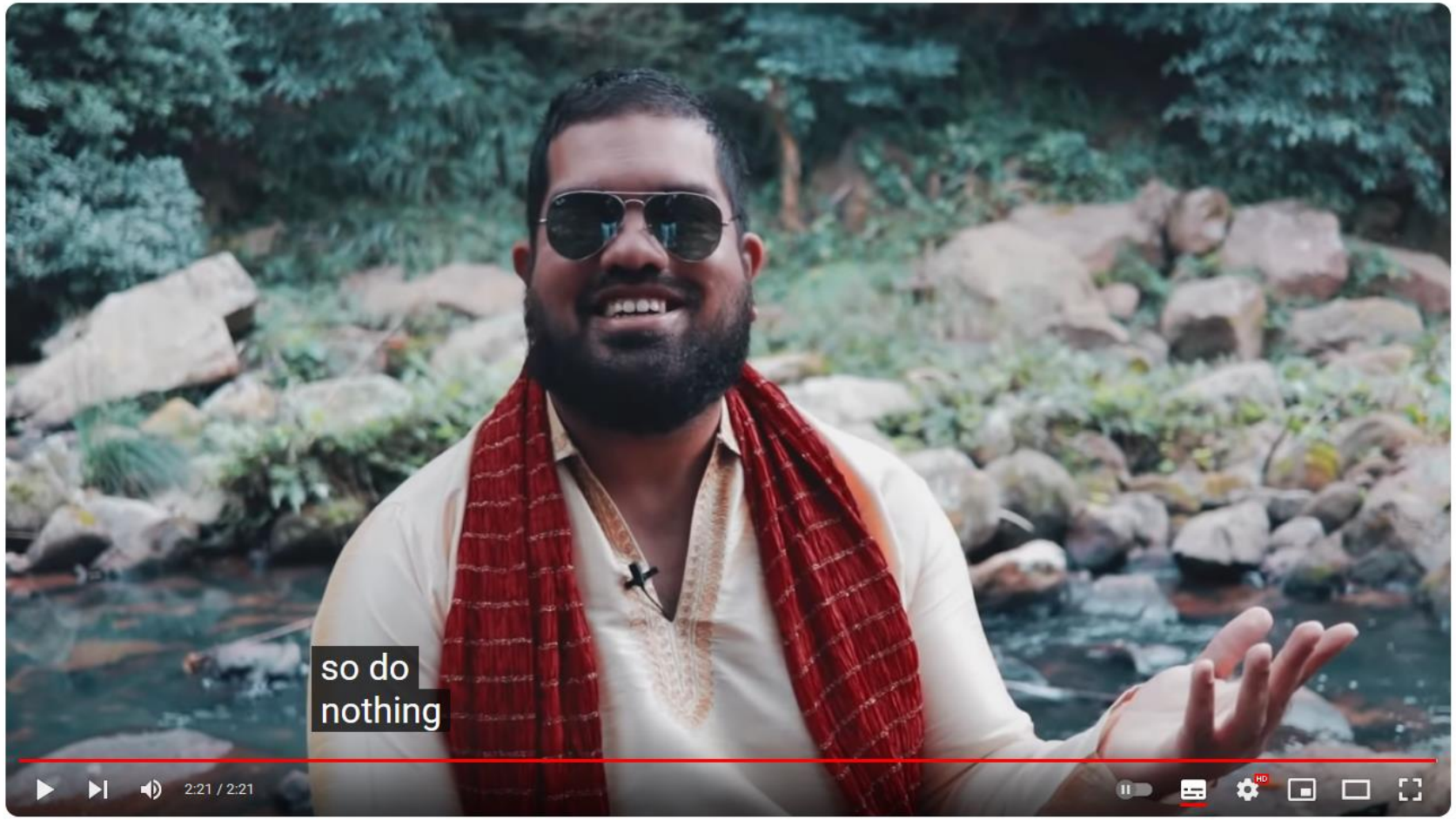
BuildContainers

API Specs like OpenAPI

Craftsman and Wrapt.dev

Continue with LLMs

Masood Boomgaard – Do nothing



Do Nothing- a message of motivation from Self-help Singh- (un) motivational speaker and life coach

Do nothing at every possible moment,
but if you have to do something,
do absolute minimum,
but it is most preferable to do fuck all

Minimal Code Paradigm Rules

SIEMENS Healthineers

KISS
Keep it simple stupid

Dry
Dont repeat yourself

Ace
Avoid ceremonies

How to for Minimal Code Paradigm Rules

SIEMENS Healthineers

1. Writing as little code as possible:

This can reduce the likelihood of bugs and make the codebase easier to understand.

2. **Reuse and leveraging existing solutions:**

Instead of reinventing the wheel, leverage libraries, tools that solve the problem.

3. **Clear and concise code:**

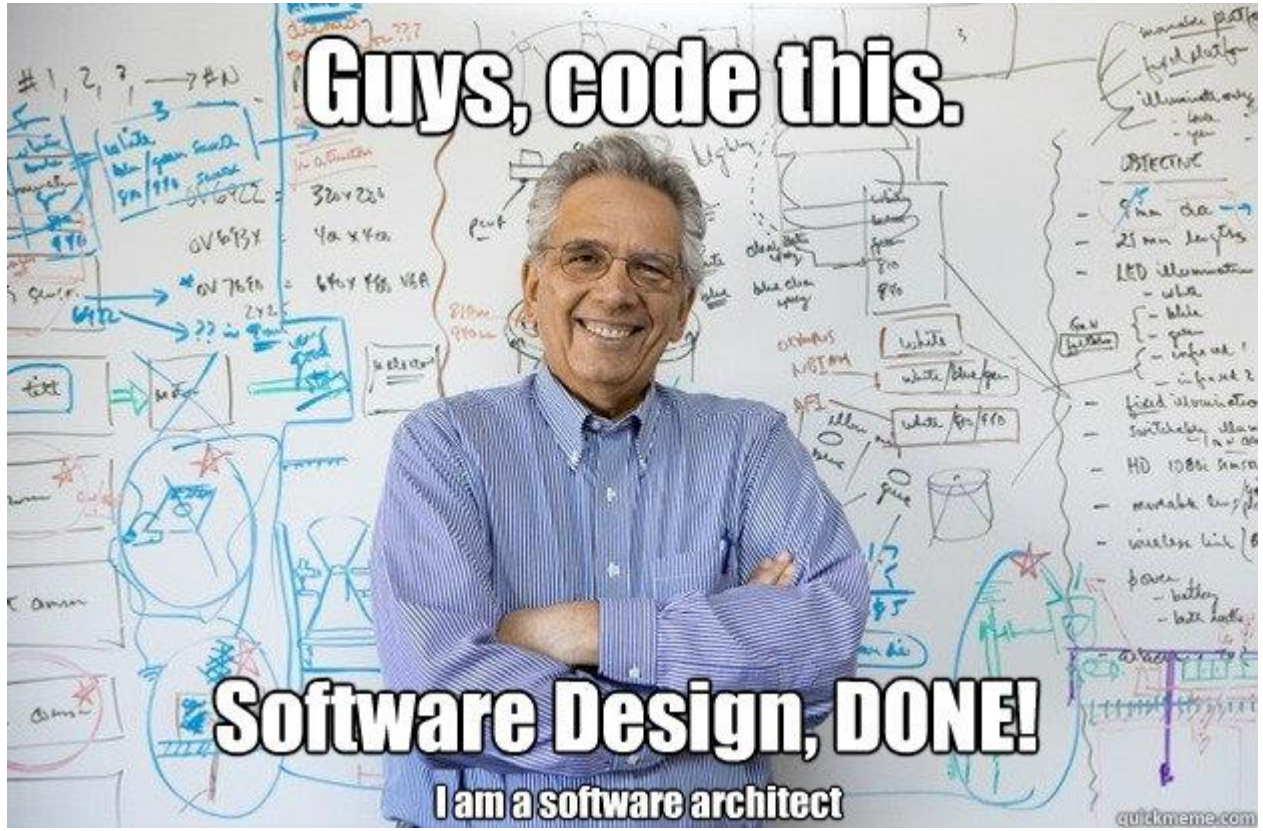
Aim for clarity over cleverness. If a function or method can be simplified, it should be.

4. **Abstraction**

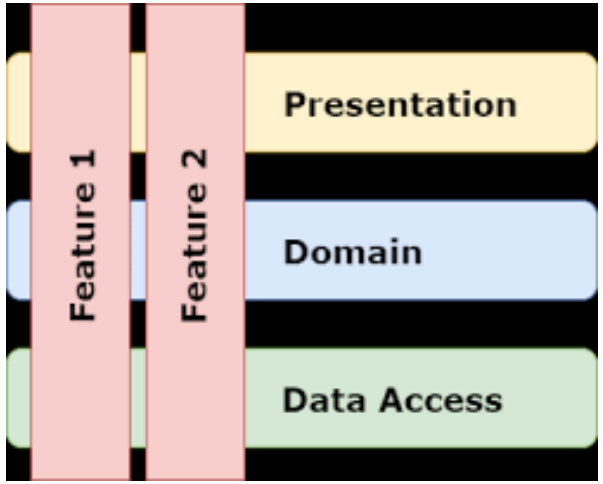
Hide unnecessary details, creating interfaces that expose only what's needed.

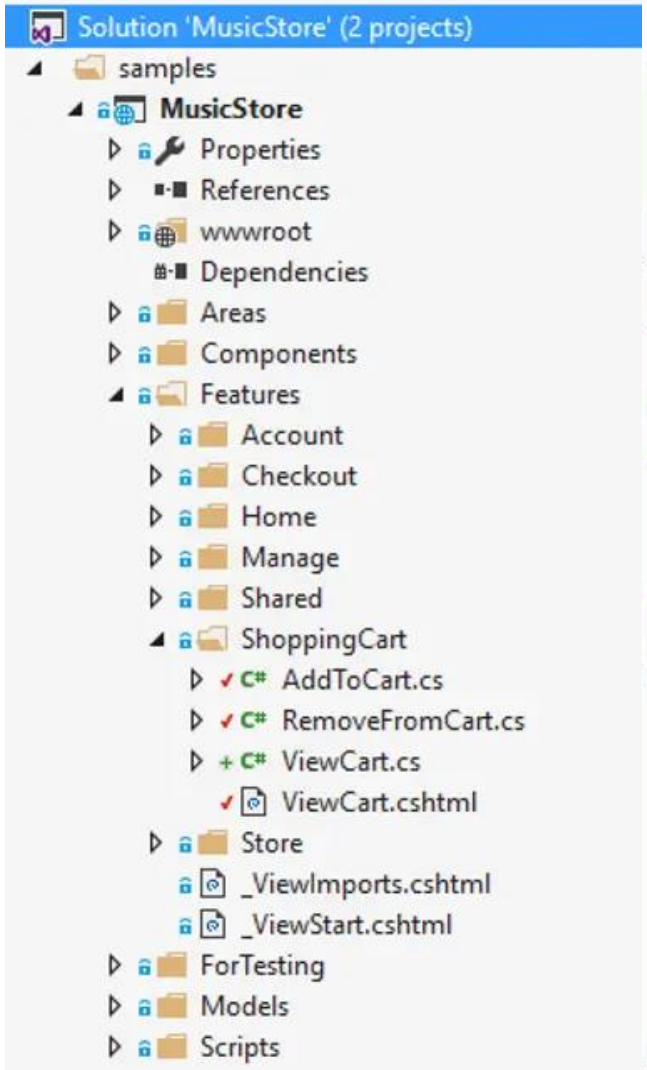
Minimal Web API are commonly used on

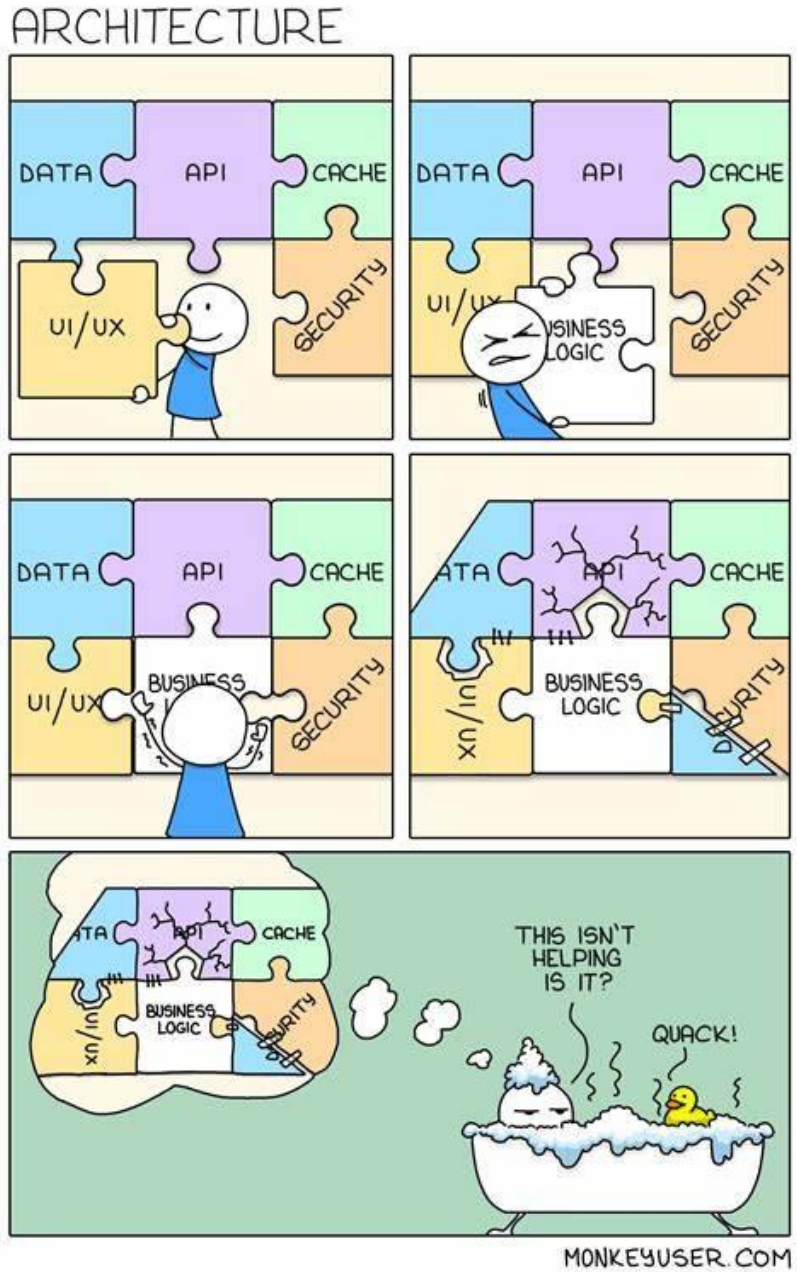
- Vertical Slice Architecture
- Microservice Architecture
- Clean Architecture,
- IoT Edge devices
- Small APIs infrastructure components.



Minimal sliced API?







Minimal APIs

SIEMENS Healthineers

Traditional ASP.NET Core Web API

csharp

```csharp
[ApiController]
[Route("[controller]")]
public class HelloController : ControllerBase
{
    [HttpGet]
    public string Get()
    {
        return "Hello, world!";
    }
}
```

Minimal API (with .NET 6)

csharp

```csharp
using Microsoft.AspNetCore.Builder;
using Microsoft.Extensions.Hosting;

var builder = WebApplication.CreateBuilder(args);
var app = builder.Build();

app.MapGet("/", () => "Hello, world!");

app.Run();
```

Avoid large ceremonies for simple stuff

SIEMENS Healthineers

EXPLORER

OPEN EDITORS

Program.cs

ASPCOREWEBAPI

- Controllers
  - WeatherForecastController.cs
- obj
- Properties
  - launchSettings.json
- appsettings.Development.json
- appsettings.json
- AspCoreWebApi.csproj
- Program.cs
- WeatherForecast.cs

Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

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();
```

Start from scratch with Minimal API

SIEMENS Healthineers

Following commands will get you started from scratch

```
dotnet new sln
dotnet new webapi --use-minimal-apis -o Weatherforecast.Api
dotnet new xunit -o Weatherforecast.Tests
dotnet sln add .\Weatherforecast.Api\ .\Weatherforecast.Tests\
```

In case you get certificate warning execute:

```
dotnet dev-certs https --trust
```

Add dependencies if you want persistance layer

```
dotnet add package Microsoft.EntityFrameworkCore.InMemory
dotnet add package Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore
```

Use Custom Templates for Minimal API in .Net

SIEMENS Healthineers

Templates which scaffold projects with Minimal API and Unit tests in .Net

```
dotnet new --install .\templates\
```

Then you can create new projects with:

```
dotnet new miniapihello -o HelloFeature
dotnet new miniapitodo -o TodoFeature
dotnet new miniapiauth -o EmployeeFeature
```

If you want to remove these project examples do:

```
dotnet new --uninstall .\templates\
```

How do I design my own template?

https://learn.microsoft.com/en-us/dotnet/core/tools/custom-templates

Just write a config file:

.template.config/template.json

Install the template:

```
dotnet new –install
```

.template.config/template.json

DOTNETMINIMALAPI

- templates
  - Minimal.HelloWorld.Api
  - Minimal.JwtAuth.Api
    - .template.config
      - template.json
    - Minimal.Api
    - Minimal.Api.Tests
    - .gitignore
    - Minimal.Api.sln
  - Minimal.ToDo.Api
  - README.md
- .gitignore
- README.md

```json
{
  "$schema": "http://json.schemastore.org/template",
  "author": "Jan Jakubcik <jan.jakubcik@siemens-healthineers.com>",
  "classifications": [ "Minimal", "Web", "Api", "Authentication", "JWT"],
  "identity": "Minimal.JwtAuth.Api",
  "name": "Minimal web api wit JWT Authentication in ASP.NET Core",
  "shortName": "miniapiauth",
  "tags": {
    "language": "C#",
    "type": "project"
  },
  "sourceName": "Minimal.Api",
  "preferNameDirectory": true,
  "symbols":{
    "includetest": {
        "type": "parameter",
        "datatype": "bool",
        "defaultValue": "true"
    }
  },
  "sources":[{
    "modifiers": [{
      "condition": "(!includetest)",
      "exclude": [ "Minimal.Api.Tests/**/*"]
      }
    ]
  }]
}
```

How to deploy Minimal API?

https://github.com/Johnz86/DotNETBuildContainers

1. Create a new minimal API project with authorization:

```
dotnet new miniapiauth -o BuildContainersFeature
```

2. Add the Microsoft.NET.Build.Containers package:

```
dotnet add package Microsoft.NET.Build.Containers
```

3. Publish your project for linux-x64:

```
dotnet publish --os linux --arch x64 -c Release -
p:PublishProfile=DefaultContainer
```

4. Run your app using the newly created container:

```
docker run -it --rm -p 5010:80 buildcontainersfeature:1.0.0
```

Working with Secrets and Certificates in .NET

SIEMENS Healthineers

Note replace $CREDENTIAL_PLACEHOLDER with your password:

```
dotnet dev-certs https -ep "$env:USERPROFILE\.aspnet\https\buildcontainersfeature.pfx" -p
$CREDENTIAL_PLACEHOLDER
dotnet dev-certs https --trust
dotnet user-secrets init
dotnet user-secrets set "Kestrel:Certificates:Development:Password" $CREDENTIAL_PLACEHOLDER --project
.\BuildContainersFeature.csproj
$secretValue = dotnet user-secrets list | Select-String 'Kestrel:Certificates:Development:Password' |
ForEach-Object { $_.ToString().Split(' ')[2] }
```

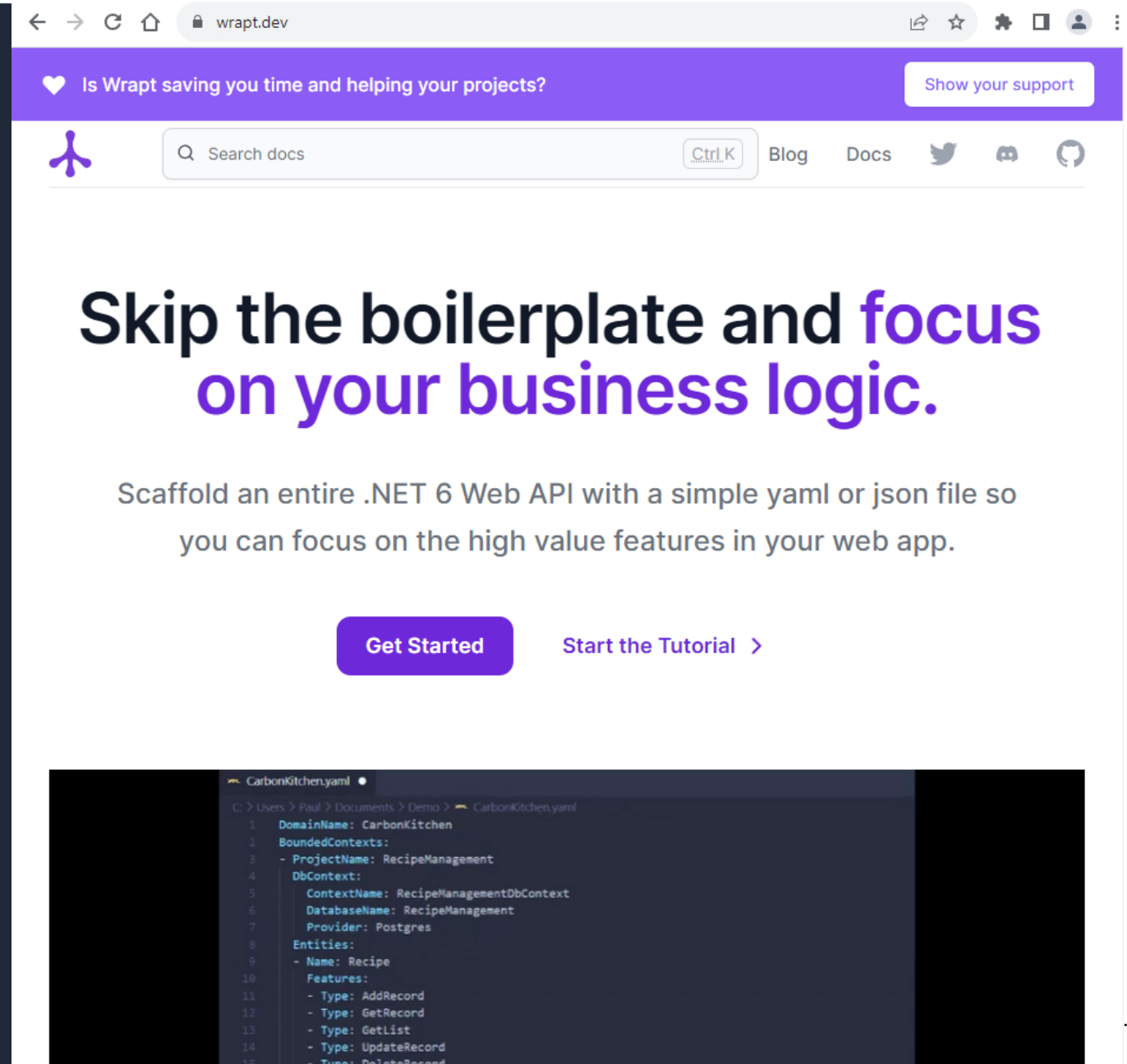
Then, run your app in a container with HTTPS:

```
docker run --rm -it -p 8000:80 -p 8001:443 -v $env:USERPROFILE\.aspnet\https\:/https/ -e
ASPNETCORE_URLS="https://+:443;http://+:80" -e ASPNETCORE_HTTPS_PORT=8001 -e
ASPNETCORE_ENVIRONMENT=Development -e ASPNETCORE_Kestrel__Certificates__Default__Password=$secretValue -e
ASPNETCORE_Kestrel__Certificates__Default__Path=/https/buildcontainersfeature.pfx
buildcontainersfeature:1.0.0
```

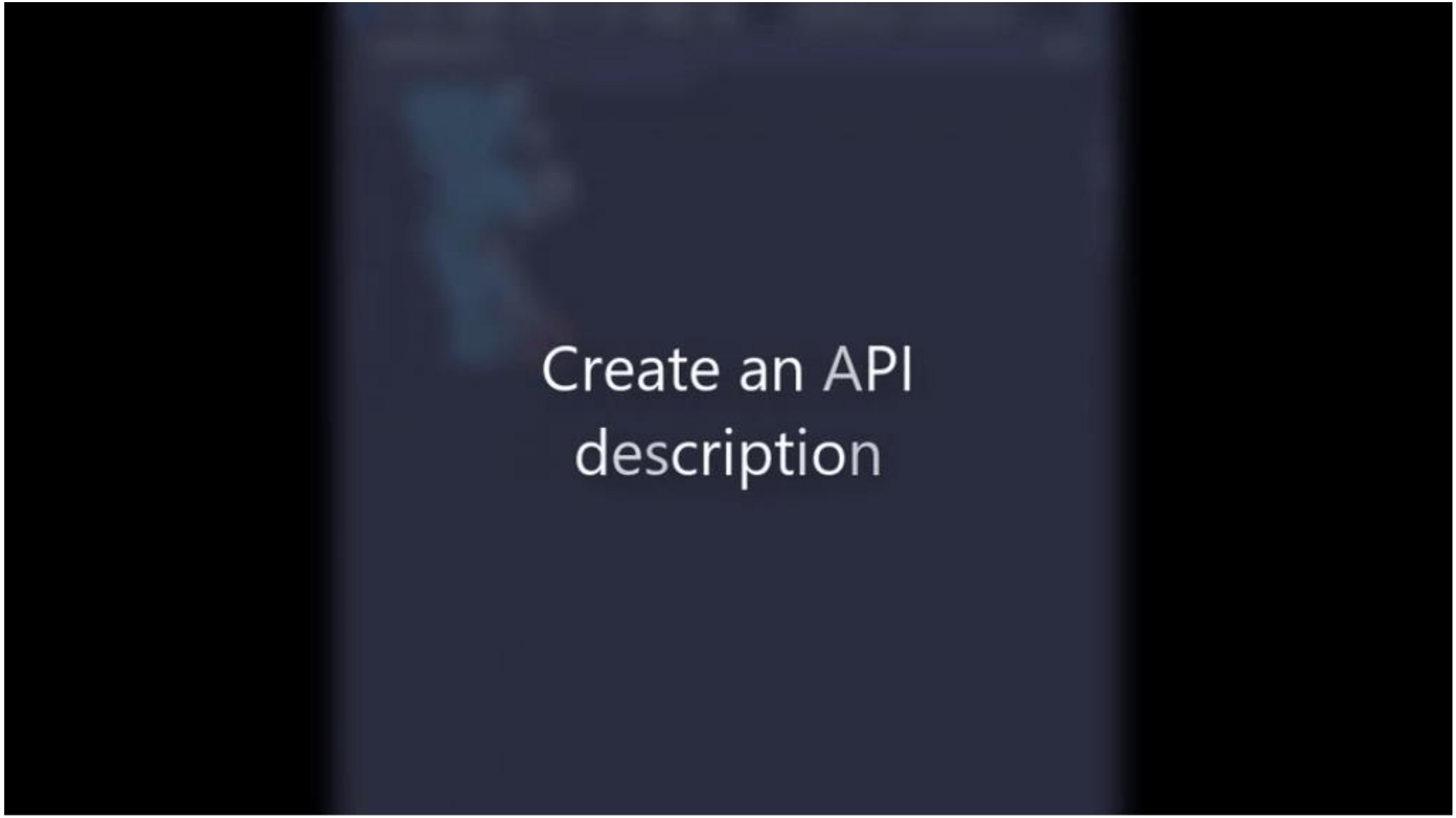
Wrapt.dev

SIEMENS Healthineers

```yaml
DomainName: CarbonKitchen
BoundedContexts:
  - ProjectName: RecipeManagement
    Port: 5005
    DbContext:
      ContextName: RecipesDbContext
      DatabaseName: RecipeManagement
      Provider: postgres
    Entities:
      - Name: Recipe
        Features:
          - Type: AddRecord
          - Type: GetRecord
          - Type: GetList
          - Type: UpdateRecord
          - Type: DeleteRecord
        Properties:
          - Name: Title
            Type: string
          - Name: Directions
            Type: string
          - Name: RecipeSourceLink
            Type: string
          - Name: Description
            Type: string
          - Name: ImageLink
            Type: string
      - Name: Ingredient
```



Craftsman CLI and Wrapt.dev



Am I more productive or just plain stupid?

SIEMENS Healthineers

Write less text

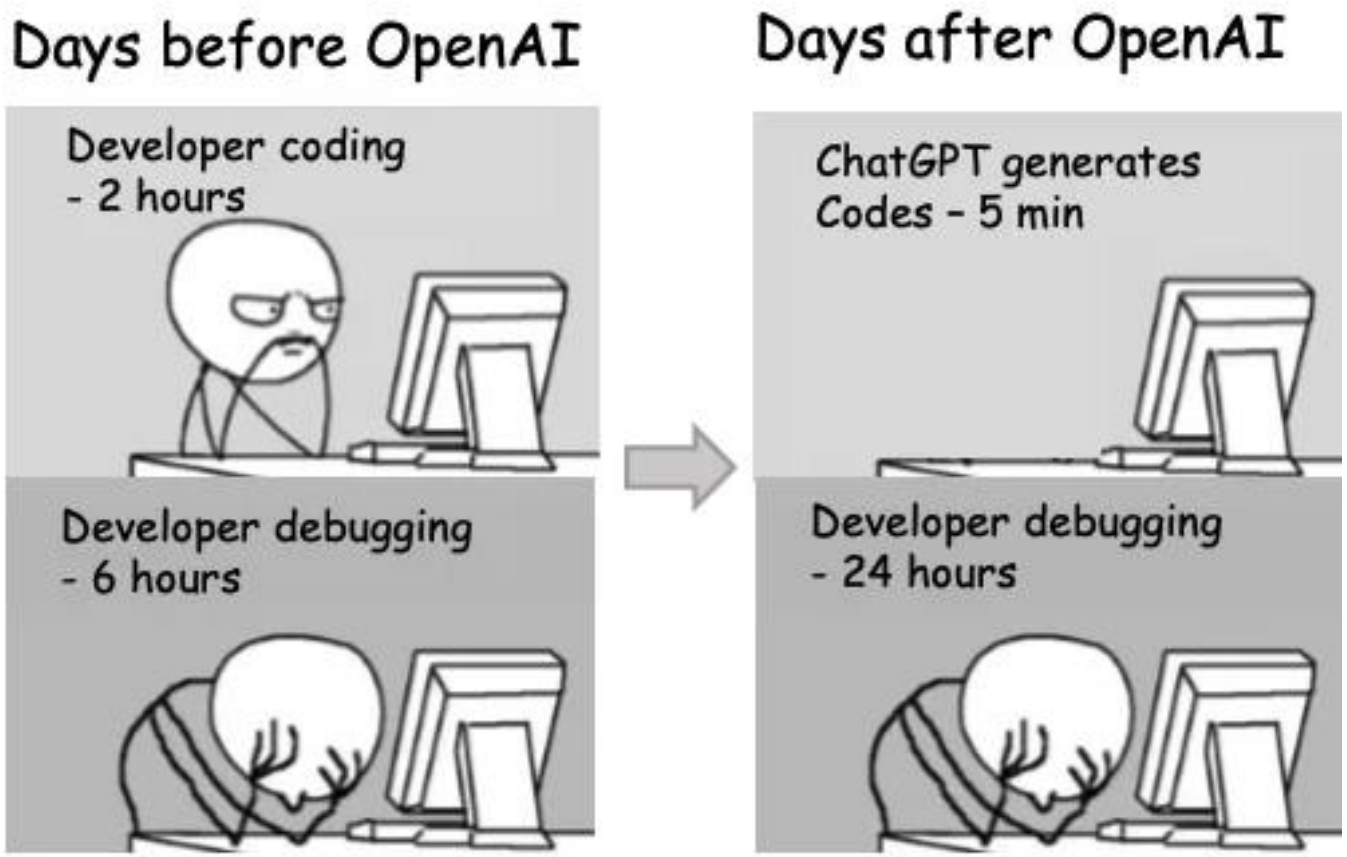
Read less code

Make less mistakes

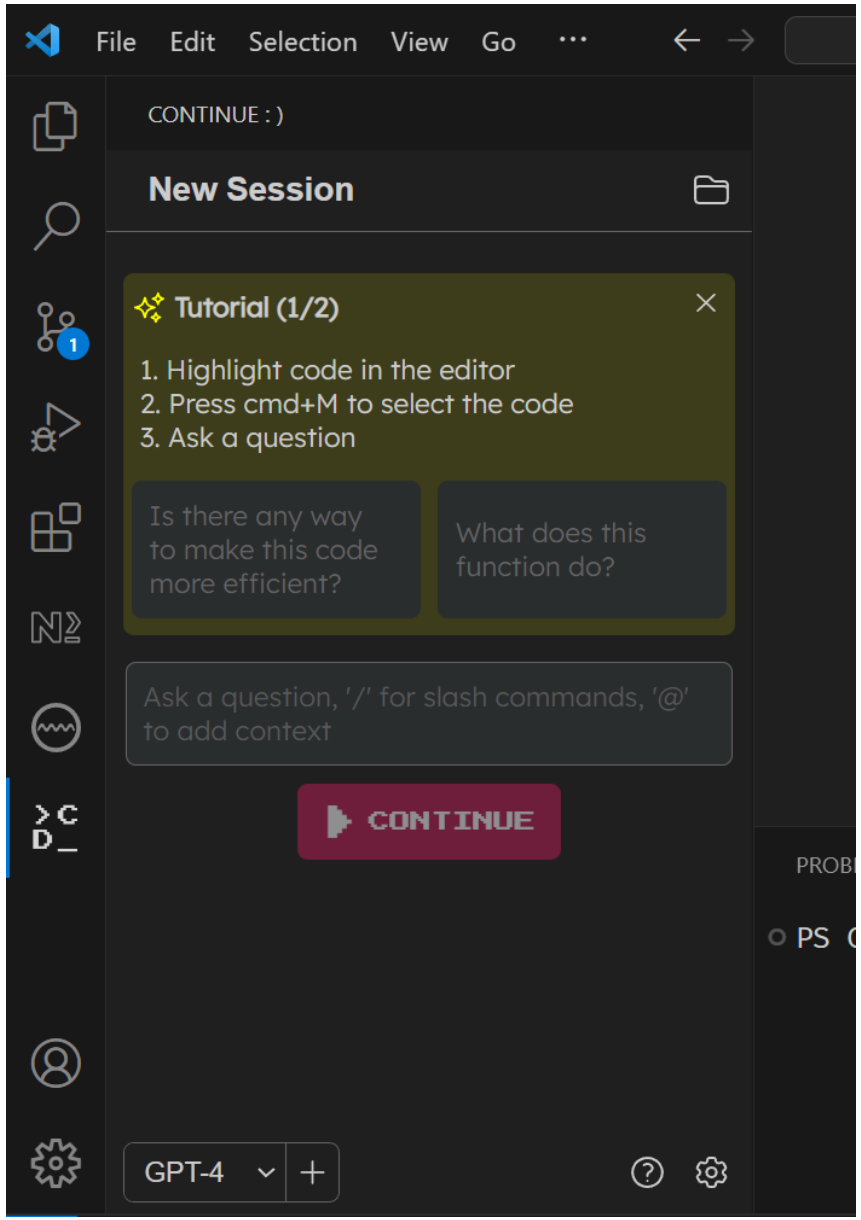
Iterate on your ideas

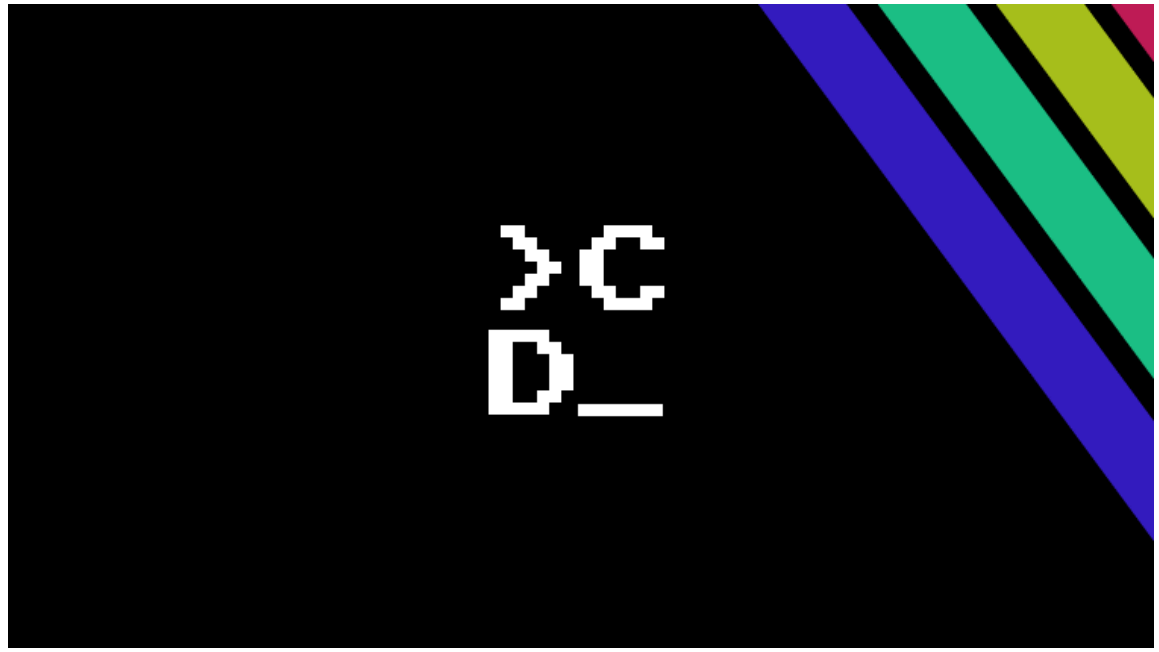
Commit more

Deploy more



Continue with LLMs





Customize and automate with LLMs

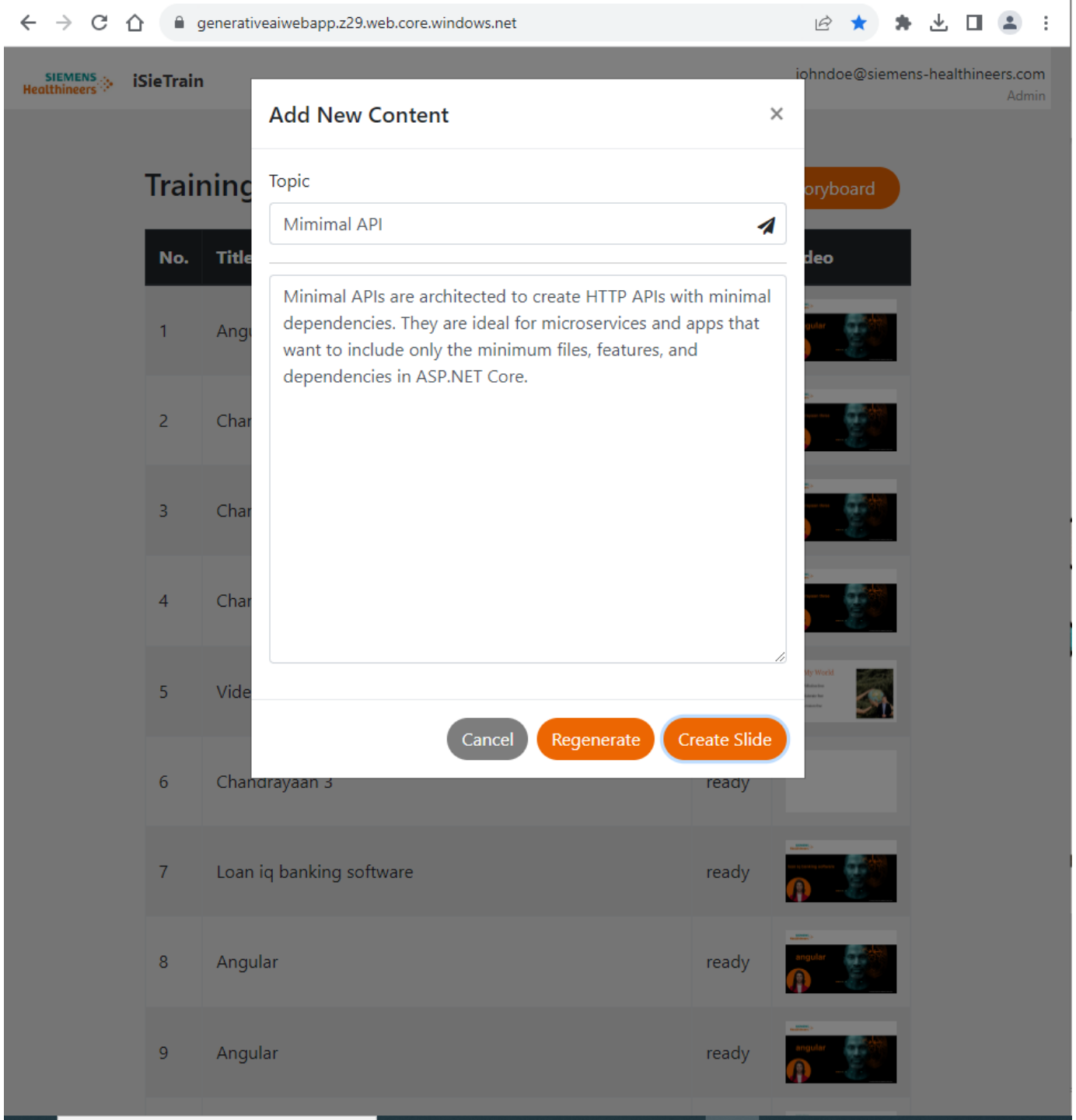
Custom commands are great when you are frequently reusing a prompt. For example, if you've crafted a great prompt and frequently ask the LLM to check for mistakes in your code, you could add a command like this:

```
config = ContinueConfig(
    ...
    custom_commands=[
        CustomCommand(
            name="check",
            description="Check for mistakes in my code",
            prompt=dedent("""\
            Please read the highlighted code and check for any mistakes. You should look for the fo
            - Syntax errors
            - Logic errors
            - Security vulnerabilities
            - Performance issues
            - Anything else that looks wrong

            Once you find an error, please explain it as clearly as possible, but without using ext
        )
    ]
)
```

Custom Slash Commands

SIEMENS Healthineers

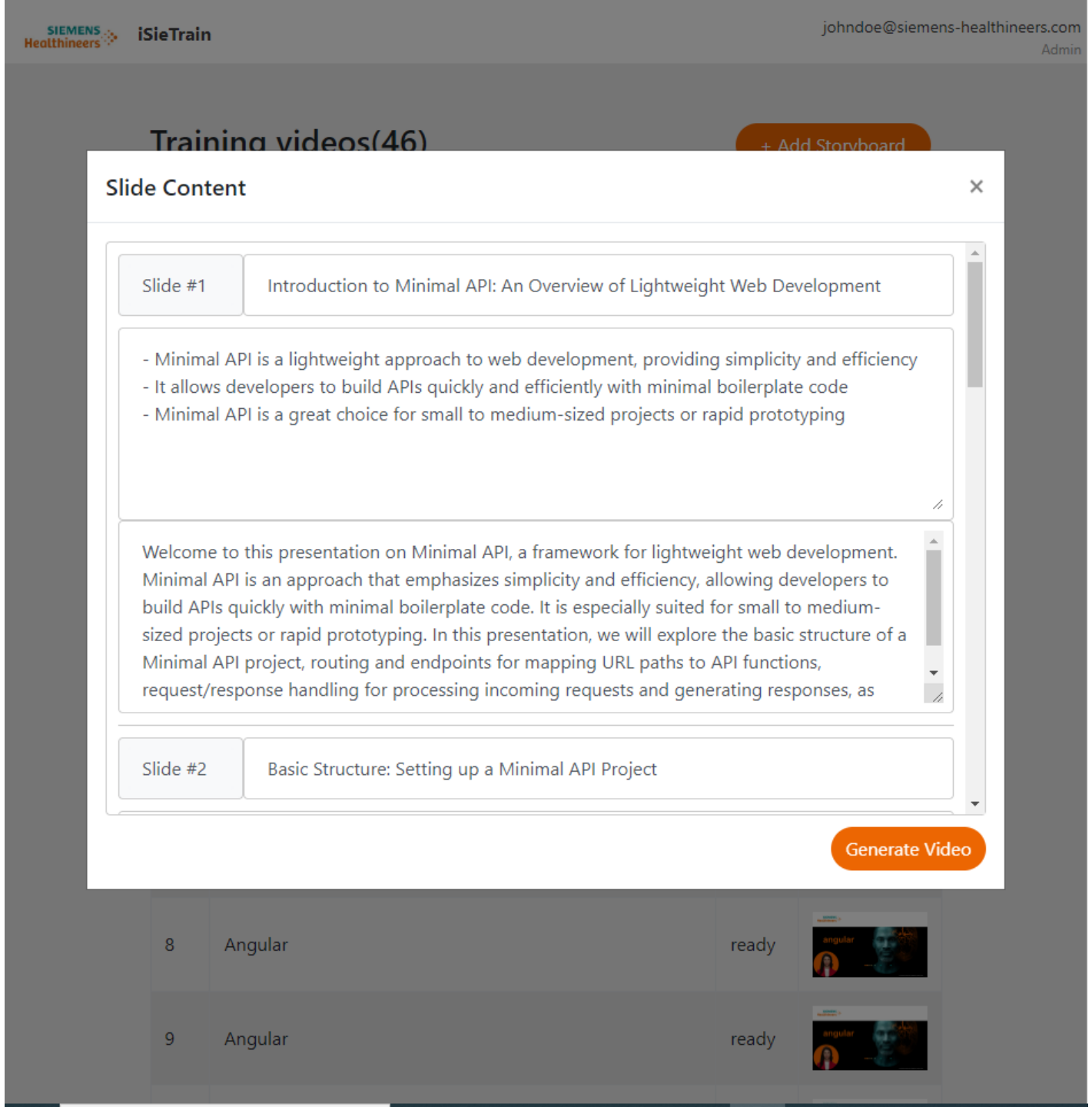
```python
class CommitMessageStep(Step):
    async def run(self, sdk: ContinueSDK):

        # Get the root directory of the workspace
        dir = sdk.ide.workspace_directory

        # Run git diff in that directory
        diff = subprocess.check_output(
            ["git", "diff"], cwd=dir).decode("utf-8")

        # Ask the LLM to write a commit message,
        # and set it as the description of this step
        self.description = await sdk.models.default.complete(
            f"{diff}\n\nWrite a short, specific (less than 50 chars) commit message about the above changes:")

config=ContinueConfig(
    ...
    slash_commands=[
        ...
        SlashCommand(
            name="commit",
            description="Generate a commit message for the current changes",
            step=CommitMessageStep,
        )
    ]
)
```



SIEMENS Healthineers

Use new tools

While creating cool things



SIEMENS Healthineers

Be creative

Create a lot with minimal effort

Future is here, use it to your advantage

SIEMENS Healthineers

SIEMENS Healthineers iSieTrain

smith@siemens-healthineers.com
Trainee

SIEMENS Healthineers

Type/Ask question

Welcome to the chat! Ask a question to get started.

Q - What version of dotnet are supporting minimal apis?

A - The .NET Core framework supports minimal APIs, which includes .NET Core 5.0 or higher.

Mimimal API

Published on October 5, 2023 • 1000 views • 500 likes

Minimal API is a concept in software development that aims to simplify the process of creating lightweight and efficient APIs (Application Programming Interfaces). The idea behind Minimal APIs is to reduce the complexity and verbosity associated with traditional API development frameworks. It focuses on providing a streamlined approach to building APIs with minimal code and configuration. With Minimal API, developers can create APIs with just a few lines of code, eliminating the need for a lot of boilerplate code that was traditionally required. It often involves leveraging the capabilities of modern programming languages and frameworks to achieve a more concise and intuitive API development experience. Minimal APIs are usually designed to be highly modular and customizable, allowing developers to choose the specific functionalities they need and discard unnecessary dependencies. This approach can lead to faster development cycles, improved code readability, and easier maintenance. Overall, the concept of Minimal API emphasizes simplicity, efficiency, and flexibility, enabling developers to quickly create lightweight and focused APIs without sacrificing functionality or scalability.

SIEMENS Healthineers

SIEMENS Healthineers

You can make coffee and clean your desk

If you are full stack developer

SIEMENS Healthineers

That is all

Thanks for your attention