
Smart Components

Smart Components lets you **add genuinely useful AI-powered features to your .NET apps quickly, easily, and without risking wasted effort.**

You don't have to spend weeks of dev time redesigning your UX or researching machine learning and prompt engineering. Smart Components are prebuilt end-to-end AI features that you can drop into your existing UIs to upgrade them, truly making your app more productive for your end users.

This is an experiment from the .NET team, and is initially available for **ASP.NET Core 6.0 and later** with either:

- **Blazor** (see: Getting started with Smart Controls and Blazor)
- **MVC / Razor Pages** (see: Getting started with Smart Controls and MVC/Razor Pages)

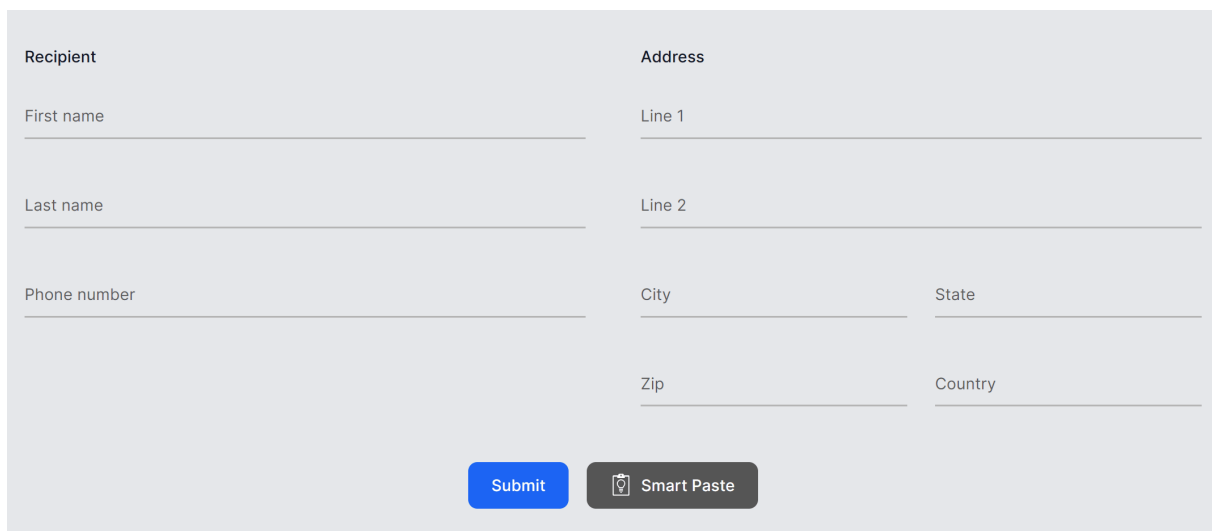
We may add support for other UI tech (e.g., native apps) later, depending on feedback.

What's included

The set of components and features may expand over time. Currently, Smart Components includes:

Smart Paste

A button that fills out forms automatically using data from the user's clipboard. You can use this with any existing form in your web app. This helps users add data from external sources without re-typing.

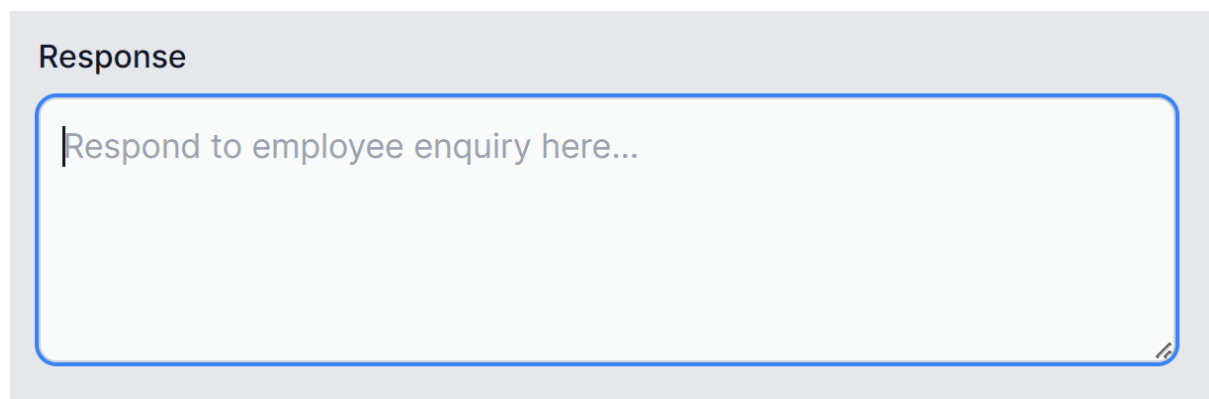


Recipient	Address	
First name	Line 1	
Last name	Line 2	
Phone number	City	State
	Zip	Country
<div><button>Submit</button><button>Smart Paste</button></div>		

Learn more: [Smart Paste docs](#)

Smart TextArea

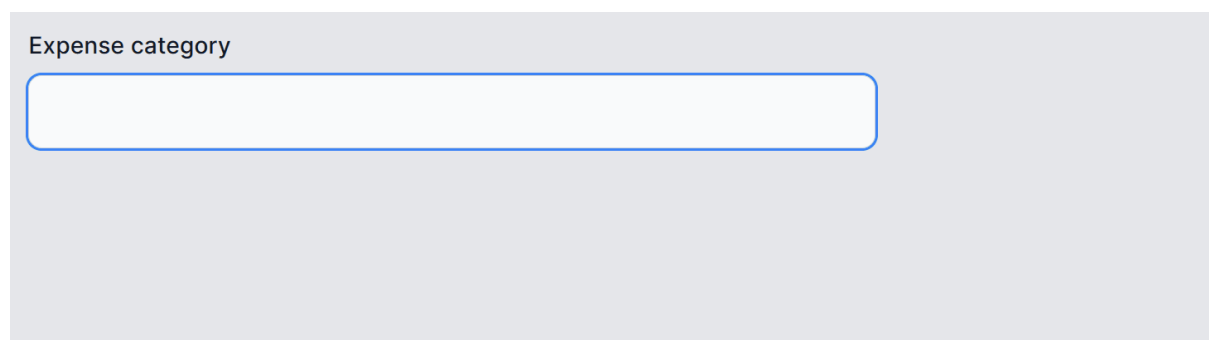
An intelligent upgrade to the traditional textarea. You can configure how it should autocomplete whole sentences using your own preferred tone, policies, URLs, and so on. This helps users type faster and not have to remember URLs etc.

A UI mockup of a Smart TextArea. It features a light gray background with the word "Response" in a bold, dark font at the top left. Below it is a large, rounded rectangular text input field with a blue border. Inside the field, the placeholder text "Respond to employee enquiry here..." is displayed in a light gray font. A small cursor is visible at the start of the text. In the bottom right corner of the input field, there is a small, faint icon of a pencil.

Learn more: [Smart TextArea docs](#)

Smart ComboBox

Upgrades the traditional combobox by making suggestions based on semantic matching. This helps users find what they're looking for.

A UI mockup of a Smart ComboBox. It features a light gray background with the text "Expense category" in a bold, dark font at the top left. Below it is a horizontal, rounded rectangular text input field with a blue border. The field is currently empty.

Learn more: [Smart ComboBox docs](#)

Local Embeddings

Computes the level of semantic similarity between two natural language strings, or finds the closest match from a set of candidates. **This runs entirely locally on your server's CPU**, so doesn't need any external AI service.

Example: evaluating the semantic similarity between two strings

```
1 var article1 = embedder.Embed("Vacation allowance policy");
2 var article2 = embedder.Embed("Returning a company vehicle");
3 var article3 = embedder.Embed("How to get your boss fired");
4
5 var searchTerm = embedder.Embed("car");
6 Console.WriteLine(searchTerm.Similarity(article1)); // Outputs: 0.41
7 Console.WriteLine(searchTerm.Similarity(article2)); // Outputs: 0.70
8 Console.WriteLine(searchTerm.Similarity(article3)); // Outputs: 0.38
```

Example: finding closest matches

```
1 // Find closest matches to "ball game"
2 var candidates = embedder.EmbedRange(["Soccer", "Tennis", "Swimming", "
   Horse riding", "Golf", "Gymnastics"]);
3
4 var closest = LocalEmbedder.FindClosest(
5     embedder.Embed("ball game"), candidates, maxResults: 3);
6
7 Console.WriteLine(string.Join(", ", closest)); // "Soccer, Golf, Tennis
   "
```

Unlike the others, this isn't a prebuilt end-to-end UI feature; it's a general capability you can use to power your own features, such as search or retrieval-augmented generation (RAG).

Learn more: [Local Embeddings docs](#)

Running the samples

1. If you don't already have it, install a current .NET SDK for Windows, Linux, or Mac.
2. Clone this repo

```
1 git clone https://github.com/dotnet-smartcomponents/
   smartcomponents.git
2 cd smartcomponents
```

3. If you want to run the Smart Paste or Smart TextArea samples, edit the [RepoSharedConfig.json](#) file at the root of the solution to add your API key. See [How to configure an OpenAI backend](#).

You can skip this if you only want to run the Smart ComboBox or Local Embeddings samples, since they run entirely locally.

4. Run it

```
1 cd samples/ExampleBlazorApp
2 dotnet run
```

Once you're ready to add Smart Components to your own app, see:

- [Getting started with Smart Controls and Blazor](#)
- [Getting started with Smart Controls and MVC/Razor Pages](#)

Feedback and support

The purpose of this experiment is to assess whether/how the .NET community would want to use prebuilt UI components for AI features.

Please take a moment to share your thoughts and feedback with us by filling out our short .NET Smart Components survey.

You can also report issues and suggest improvements by creating an issue on GitHub.

Smart Components isn't yet an officially supported part of .NET. Whether or not it graduates to full support depends on community feedback and usage levels.

Contributing

This project has adopted the Microsoft Open Source Code of Conduct. For more information see the Code of Conduct FAQ or contact opencode@microsoft.com with any additional questions or comments.