

Documentation

[Xcode](#) / [Source Editor](#) / Writing code with intelligence in Xcode

Article

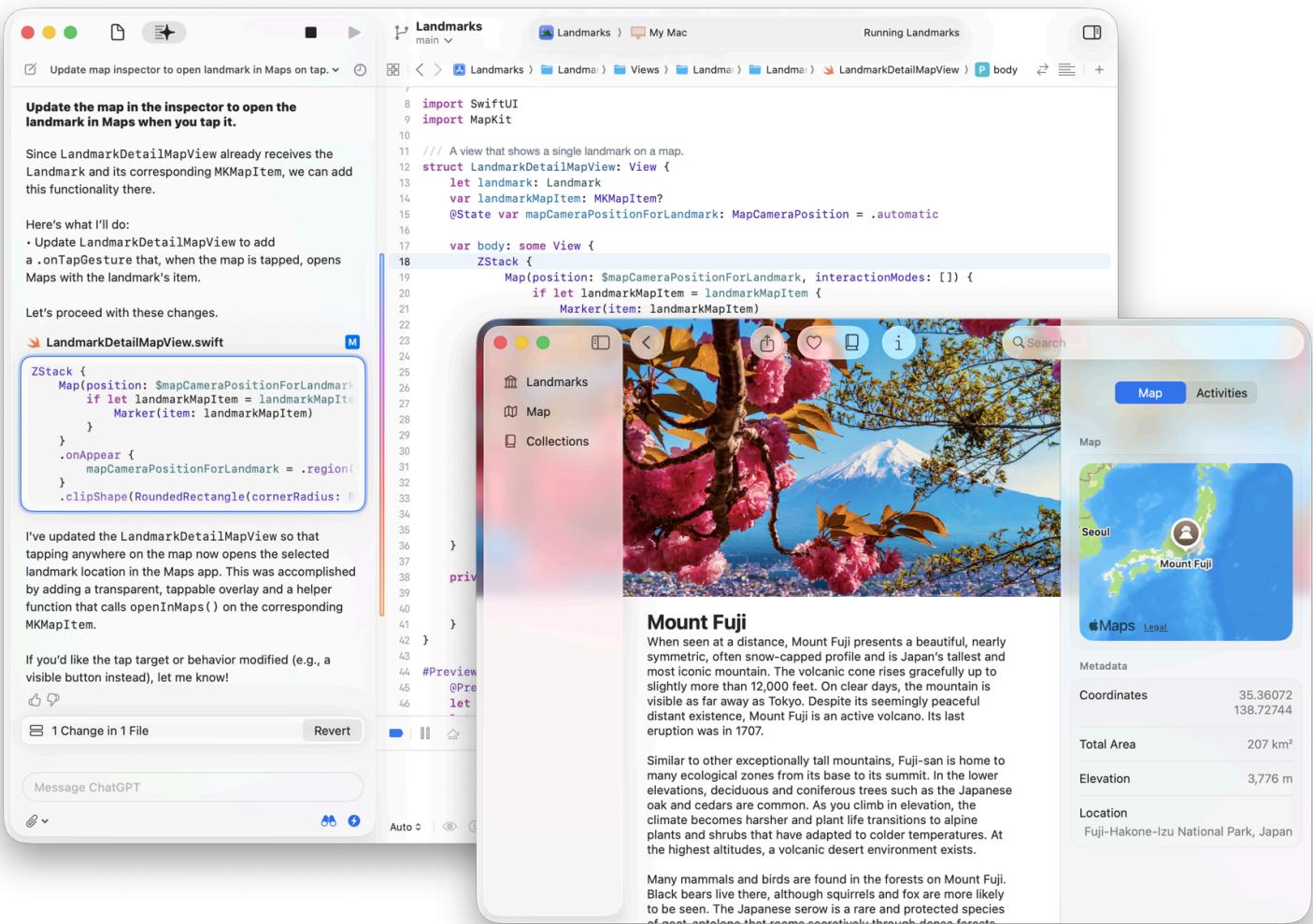
Writing code with intelligence in Xcode

Generate code, fix bugs fast, and learn as you go with intelligence built directly into Xcode.



Overview

The coding intelligence features in Xcode help you write code, navigate unfamiliar codebases, find opportunities for new features, fix or refactor existing code, and generate documentation along the way.



You interact with a large language model of your choice using natural language prompts to ask questions and give instructions. The model refines the responses to your prompts based on your previous interactions and project context. You stay in control over changes to your project by applying suggestions automatically or reviewing and applying them selectively yourself. Xcode maintains a history of your conversations with the model so that you can review past responses, track changes, and return to any previous state of your project.

Set up coding intelligence

To use coding intelligence in Xcode, you need to choose a model. You can enable ChatGPT in Xcode or Claude in Xcode, where these models are available, or use another model that you prefer. First, choose Xcode > Settings and select Intelligence in the sidebar.

To use ChatGPT in Xcode (with or without an account):

1. In the ChatGPT in Xcode row, click Turn On.
2. In the dialogs that appear, click Next and then click Turn On ChatGPT.

To sign in to a free ChatGPT account, or a paid account with higher limits:

1. In Intelligence settings, click the ChatGPT in Xcode row.

2. With the ChatGPT in Xcode switch on, click Sign In, and in the next dialog, click Sign In again.

3. In the browser window that appears, follow the instructions to enter your credentials.

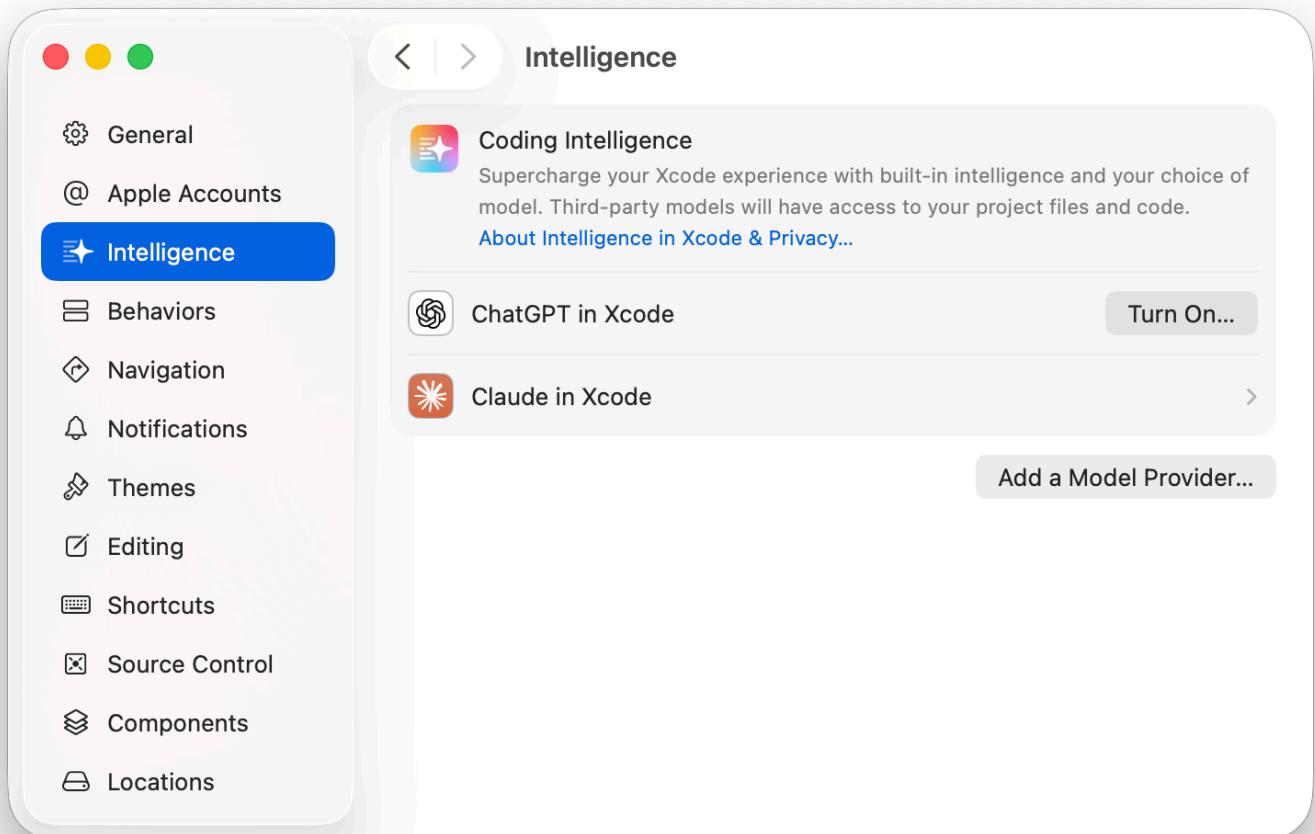
To upgrade your free ChatGPT account to a paid account, click Upgrade to ChatGPT Plus on the bottom of the ChatGPT in Xcode settings.

To use Claude in Xcode:

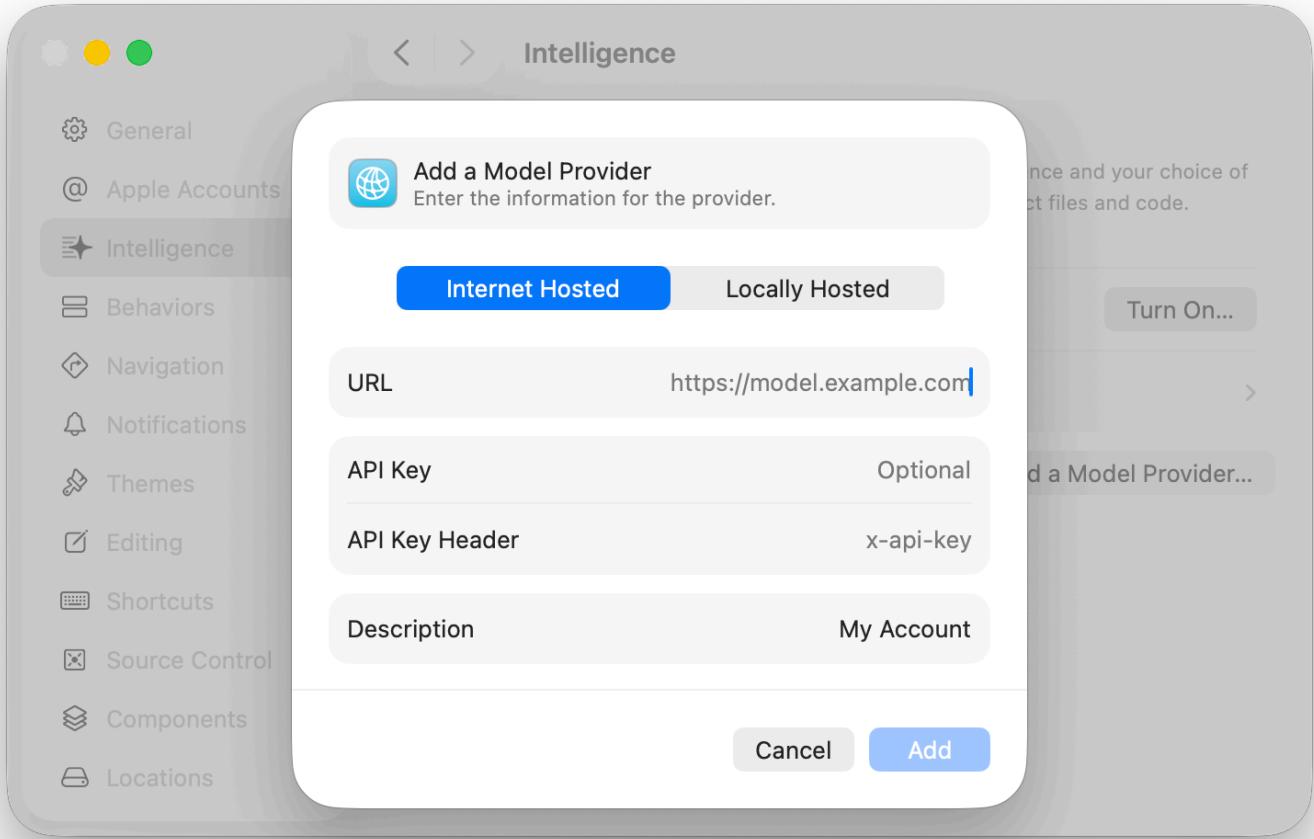
1. In Intelligence settings, click the Claude in Xcode row.

2. In Claude in Xcode settings, click Sign In.

3. In the browser window that appears, follow the instructions to enter your credentials.



To use another model provider, click the Add a Model Provider button. To add a model that's hosted on the internet, select Internet Hosted, enter the URL and other details, and click Add in the dialog that appears. To add a model that's hosted locally on your Mac, select Locally Hosted and enter a port and optional description instead.



If you choose another model, it needs to support the Chat Completions API. In addition, Xcode expects the model to support these endpoints that list models and perform completions:

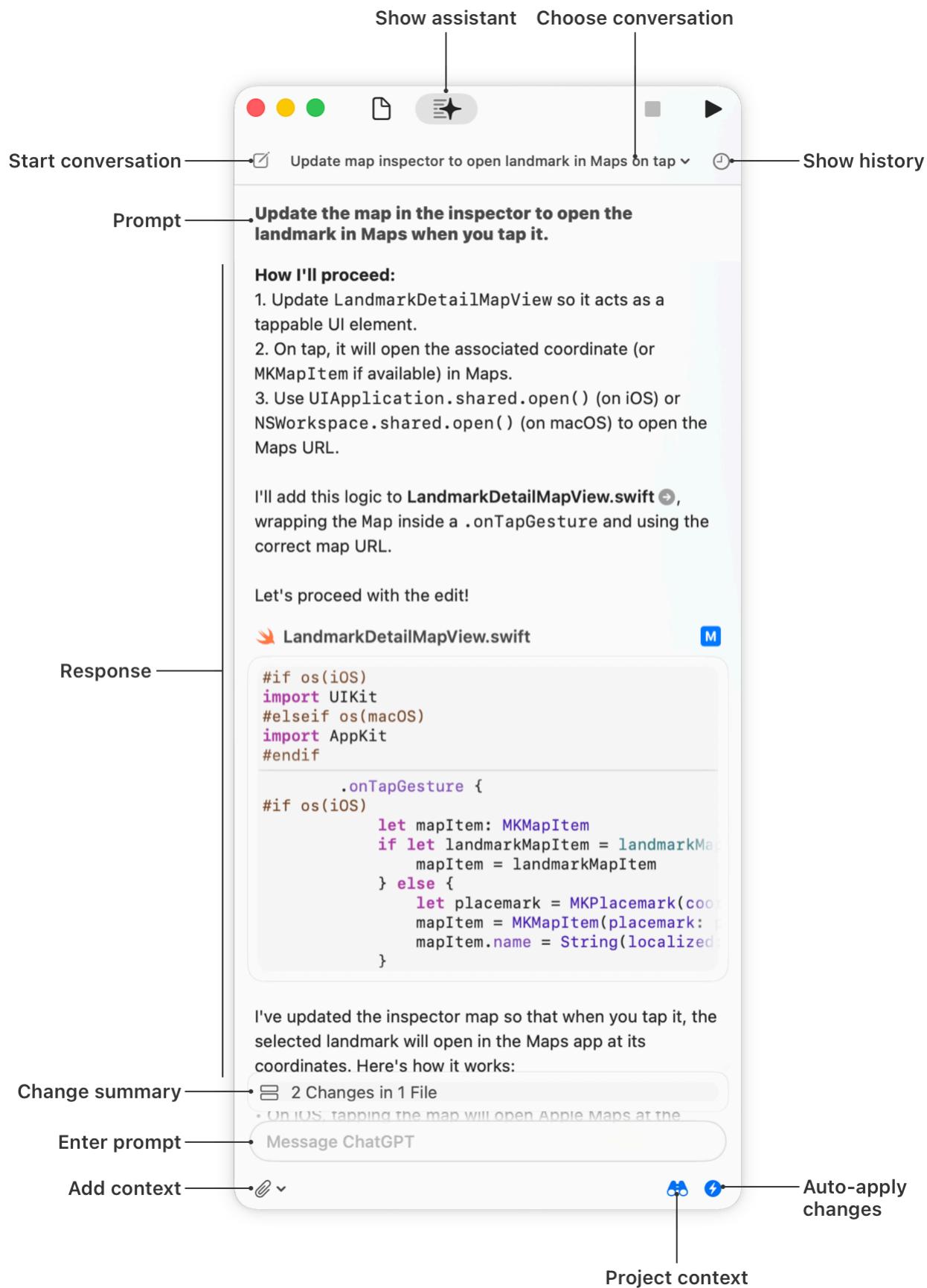
- {Model provider URL}/v1/models
- {Model provider URL}/v1/chat/completions

Note

If you need to disable the coding assistant for managed devices, set the `CodingAssistant AllowExternalIntegrations` key to `false` in a mobile device management (MDM) profile. For more information, see [Device management restrictions for Mac computers](#).

Display the coding assistant

In the upper-left corner of the toolbar, click the button to the right of the navigator button to open the coding assistant area in the sidebar or press Command-0. Here, you can enter prompts, see responses, start and navigate between conversations, rollback changes, and more.



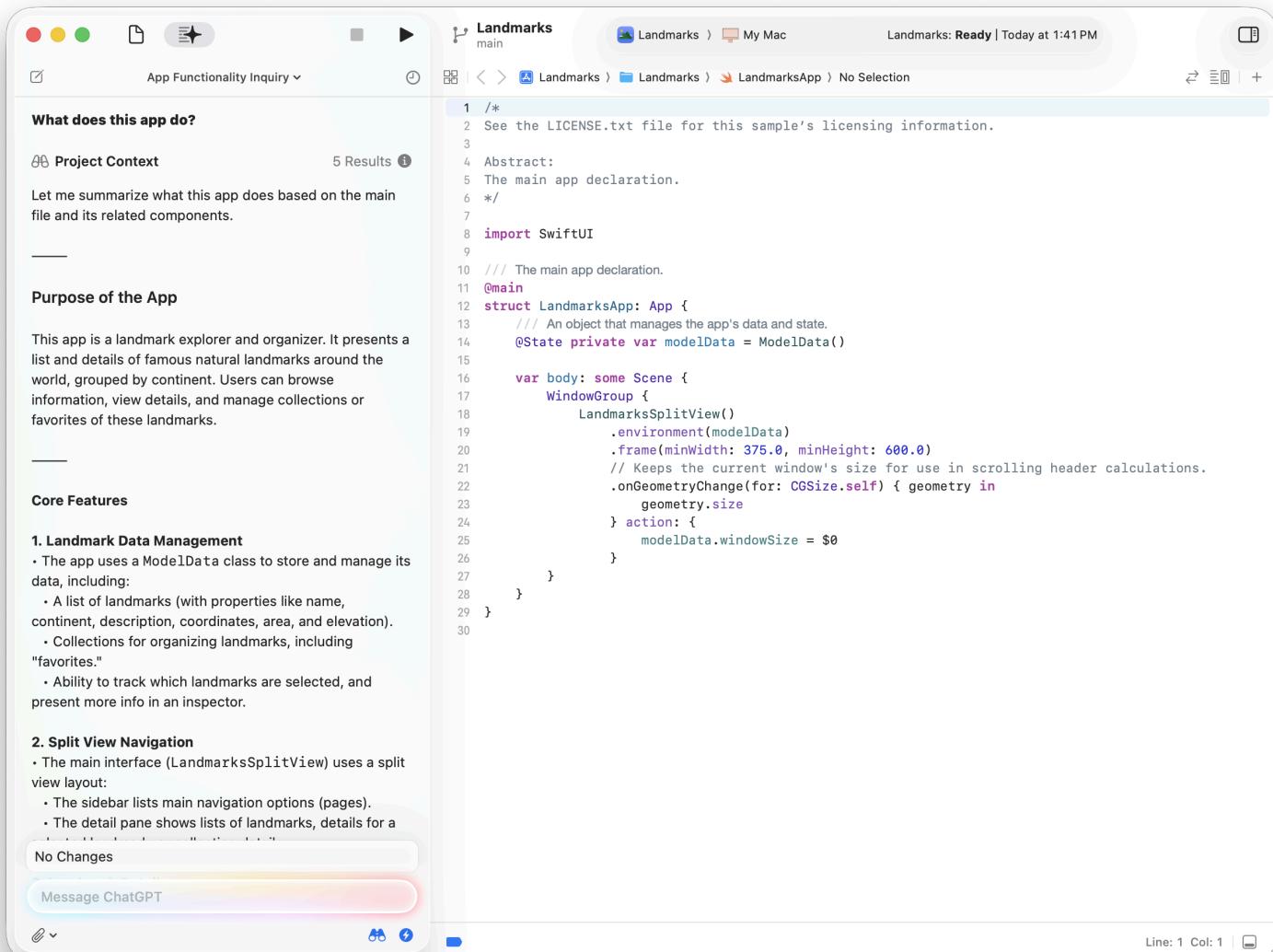
If a Set Up Intelligence button appears in the sidebar instead, click the button and enable a coding intelligence model in Xcode > Settings > Intelligence.

Explore unfamiliar code

At any time, you can ask Xcode to explain code and find files to implement a new feature or just to familiarize yourself with some code. For example, if you download the [Landmarks](#) sample app, you

can select code and ask questions, such as:

- *What does this app do?*



Xcode responds under your prompt in the conversation area of the coding assistant. The response may contain content that you can interact with. For example, if the response references a filename, click the arrow button next to the filename to open it in the source editor. To continue the conversation with the coding assistant, enter follow-up prompts, like:

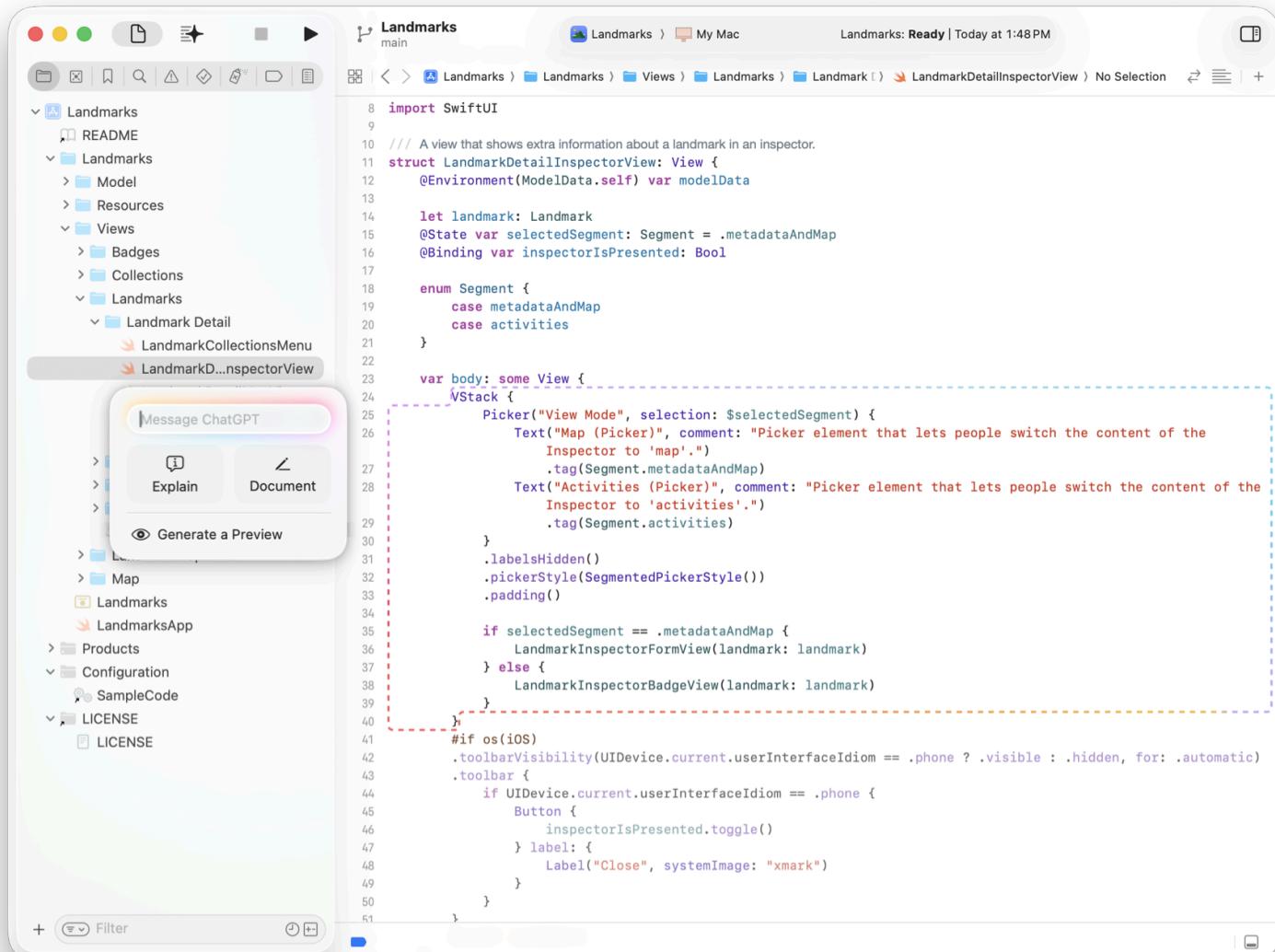
- *Tell me more about the views that display this object*

When you enter another prompt, the coding assistant appends your prompt and response to the conversation. Xcode maintains an entire transcript of your interactions with the model so you can refer back to them.

Learn about symbols and code

In the source editor, Control-click a symbol or code selection and press Command-Option-0 or choose Show Coding Tools > Show Coding Tools from the contextual menu. Then click Explain, or

enter a more specific prompt in the coding tools popover. The coding assistant displays the prompt and its response in the conversation area.



Alternatively, click the coding assistant button in the source editor gutter to display the coding tools popover.

Generate or modify code

Give Xcode specific instructions on how to generate or modify code. If you aren't getting the result you expected, try breaking down your question or adding more detailed instruction. Between each prompt, review and validate the code changes with a preview or playground, and continue iterating on your app by adjusting your prompts to get the behavior you're looking for.

For example, if you're new to Swift and SwiftUI, you can code along with the modifications that Xcode makes. Start with a Swift app that you create from a template and instruct Xcode to make incremental changes, such as:

- Add properties and methods to a class

- Create a list view and wrap it in a NavigationStack
- Add the ability to edit the properties of items in the list view
- Change the list view to a table view showing all the properties

While working on a response, Xcode displays progress messages in the message text field before posting its response in the conversation area. The response may contain a description of the changes, including some steps or code changes.

The screenshot shows the Xcode interface with the following details:

- Title Bar:** Landmarks main
- Project Navigator:** Shows the project structure: Landmarks > Landmarks > Views > LandmarkDetailMapView.swift
- Message Text Field:** Displays a series of steps and code changes.
- Code Editor:** Shows the `LandmarkDetailMapView.swift` file with Swift code.

```

Update the map in the inspector to open landmark in Maps on tap.

Since LandmarkDetailMapView already receives the Landmark and its corresponding MKMapItem, we can add this functionality there.

Here's what I'll do:
- Update LandmarkDetailMapView to add a .onTapGesture that, when the map is tapped, opens Maps with the landmark's item.

Let's proceed with these changes.

LandmarkDetailMapView.swift

ZStack {
    Map(position: $mapCameraPositionForLandmark
        if let landmarkMapItem = landmarkMapItem
            Marker(item: landmarkMapItem)
    )
    .onAppear {
        mapCameraPositionForLandmark = .region(
    }
    .clipShape(RoundedRectangle(cornerRadius: 8
}

I've updated the LandmarkDetailMapView so that tapping anywhere on the map now opens the selected landmark location in the Maps app. This was accomplished by adding a transparent, tappable overlay and a helper function that calls openInMaps() on the corresponding MKMapItem.

If you'd like the tap target or behavior modified (e.g., a visible button instead), let me know!

```

Message ChatGPT

```

8 import SwiftUI
9 import MapKit
10
11 /// A view that shows a single landmark on a map.
12 struct LandmarkDetailView: View {
13     let landmark: Landmark
14     var landmarkMapItem: MKMapItem?
15     @State var mapCameraPositionForLandmark: MapCameraPosition = .automatic
16
17     var body: some View {
18         ZStack {
19             Map(position: $mapCameraPositionForLandmark, interactionModes: [])
20                 if let landmarkMapItem = landmarkMapItem {
21                     Marker(item: landmarkMapItem)
22                 }
23         }
24         .onAppear {
25             mapCameraPositionForLandmark = .region(landmark.coordinateRegion)
26         }
27         .clipShape(RoundedRectangle(cornerRadius: 8))
28
29         // Transparent tappable layer
30         Color.clear
31             .contentShape(Rectangle())
32             .onTapGesture {
33                 openInMaps()
34             }
35     }
36
37     private func openInMaps() {
38         guard let mapItem = landmarkMapItem else { return }
39         mapItem.openInMaps(launchOptions: nil)
40     }
41 }
42
43 #Preview {
44     @Previewable @State var modelData = ModelData()
45     let previewLandmark = modelData.landmarksById[1012] ?? modelData.landmarks.first
46     let previewMapItem = modelData.mapItemsByLandmarkId[1012]
47
48     LandmarkDetailView(landmark: previewLandmark, landmarkMapItem: previewMapItem)
49
50 }
51

```

Line: 18 Col: 1

The changes may go beyond what you asked for or build on your previous prompts. The response may contain next steps and ask you a follow-up question. Xcode puts you in control, where you lead and redirect the conversation as needed to steer the model in the direction you'd like to go.

You can either answer the question (continue the conversation with the assistant) or enter a new prompt.

To generate code about specific symbols while in the source editor, you can use the coding tools popover. Control-click a symbol and choose Show Coding Tools > Show Coding Tools from the contextual menu, then enter a prompt in the coding tools popover.

Apply changes to your code

You can see code changes that Xcode automatically applies to your files as file *snippets* in the conversation. These snippets show the modified or added file, and clicking on them takes you to that specific change in the source editor.

Tip

Use snippets to navigate through all of the changes that Xcode applies and review them in the source editor.

If you want Xcode to automatically edit your code, turn on Automatically Apply Changes by clicking the button at the lower-right corner of the sidebar. The response may contain content that you can interact with. For example, click a code listing to open changes in the source editor. Xcode uses multicolor change bars to highlight the lines of code that it changes using coding intelligence.

To undo changes, click Revert above the message text field. To reapply the changes, click Reapply. Alternatively, click the History button in the upper-right corner of the sidebar, to see a history of changes and roll them back by prompt (see [Rollback changes using the conversation history](#)).

If you turn the automatically apply button off, Xcode proposes changes to your code instead of applying them and labels them as “Proposed Changes” in the conversation. The response may describe the code changes that the assistant suggests and contain proposed code that you can selectively apply or paste into your files. To apply a proposed change, click the code in the response and click Apply. To apply all the suggested changes, click the Apply button above the message text field.

Customize the context of your prompts

By default, Xcode automatically gathers relevant context to send to the model, based on your prompt and the conversation history. In addition to the automatic context, you can reference specific symbols and files, upload attachments, or refer to a selection in the source editor by mentioning it in your prompt.

You can add specific references to symbols and files by typing the @ character and choosing a symbol or file:

To add additional files from outside your project, choose “Upload files” from the Attachments pop-up menu in the lower-left corner, under the message text field, and select the files to upload from the dialog.

The Project Context button in the lower-right corner of the sidebar is on by default. This allows Xcode to share relevant code and other context from your project with the model. To see the files and search terms that Xcode used, click the information icon next to Project Context if it appears in the response. To narrow the scope of the project files, turn off the automatic search feature and add explicit references to files and symbols in your prompt.

Generate playgrounds and previews

Playgrounds and previews are a great way to experiment with new code without modifying your app. Use playgrounds to run and display code snippets in the canvas, and use previews to validate UI code across platforms. Xcode generates playground and preview code that may contain sample data to help you better understand and visualize the code in the canvas.

To add a playground macro to your project, open coding tools and choose Generate a Playground:

Xcode shows the results of the playground, and for SwiftUI files, the previews, in the canvas area. If the canvas isn't open, choose Editor > Canvas to show it, then click Resume.

To learn more about the playground macro, see [Running code snippets using the playground macro](#). For previews, see [Previewing your app's interface in Xcode](#).

Fix your code

If you encounter a compilation warning or error while building your app, Xcode may be able to generate a fix for you.

The source editor highlights any issues with a red underline and presents an issue summary and icon. Click the icon to show more information about the issue, then click Generate next to "Generate Fix for Issue". Xcode applies the model-generated change and shows the fix in the coding assistant.

Generate documentation

Let Xcode draft your API documentation for you. Select a symbol in the source editor and click the coding intelligence icon that appears. In the dialog, click Document.

Xcode can add DocC-style comments to the source file above the symbol. For example, select a class and Xcode adds documentation for the class, its properties and methods, including the method parameters.

Xcode displays coding intelligence controls at the bottom of the source editor that summarizes the change. To see the response in the conversation area of the coding assistant, click the coding assistant button. To undo changes, click the Revert button.

Browse previous conversations

At any time you can review the conversations you have with the coding assistant. A conversation is a thread of prompts and responses that appears in the conversation area. For example, you can ask the model to make a series of changes for a feature you're working on in the same conversation. Then create a new conversation for another feature that's in a different part of your code.

You can manage your conversations in the conversation area or using the Conversation pop-up menu in the toolbar to:

- Review prompts and responses in the same conversation by scrolling up or down in the list of prompts.
- Show or hide a response by clicking the disclosure triangle to the right of the prompt.
- Jump to a recent or previous conversation by choosing the conversation from the menu.
- Remove previous conversations by choosing Clear All from the menu.
- Start a new conversation by clicking the New Conversation button on the left in the toolbar, then enter a prompt in the message text field below.

Rollback changes using the conversation history

Use the conversation history that Xcode maintains to rollback changes to a known state of your project, or to review changes across multiple files in your project.

To rollback changes in a conversation by prompt, choose the conversation from the Conversation pop-up menu and click the History button. Xcode shows a chronological list of your prompts with a slider on the right. Move the slider from the bottom to the top to unwind changes in the order that you made them. Move the slider up to remove changes, that Xcode shows on the right, and move the slider down to restore changes in the next prompt.

Xcode retains all the edits it made after that state in case you decide to roll forward any changes later. After scrubbing back to the point you would like to restore to, click the Restore button to update your project files to the state from this point on. To keep all the changes in a conversation regardless of the current slider position, click Cancel.

Note

To use the History feature, your project must have a Git repository. If you don't have a repository, click the Create Repository button that appears when you click the History button. Alternatively, choose Integrate > New Git Repository to create a local repository. Xcode doesn't make changes to your repository, but its history relies on your project's Git history for reference purposes.