

Create haikus about Google products with the PaLM API and Flutter

[Before you begin](#)

[Prerequisites](#)

[What you'll learn](#)

[What you'll need](#)

[Get set up](#)

[Download the starter code](#)

[Download the project dependencies](#)

[Run the starter app](#)

[Explore the starter app](#)

[Set up access to the PaLM API](#)

[Add a menu of Google products](#)

[product_repository_impl.dart](#)

[Send the request to the PaLM API and decode the response](#)

[poem_repository_impl.dart](#)

[Run the app on mobile platforms](#)

[Run the app on desktop platforms](#)

[Run the app on Linux](#)

[Run the app on macOS](#)

[DebugProfile.entitlements | Release.entitlements](#)

[Run the app on Windows](#)

[Run the app on the web platform](#)

[Congratulations](#)

[Learn more](#)

Before you begin

Duration: 3:00

One of the most exciting machine-learning (ML) breakthroughs is generative AI, which can produce amazing images, text, audio, and even video based on simple text—or prompts—from users. Specifically with the release of the [PaLM API](#), Google empowers developers to build categories of apps with delightful user experiences reimaged with PaLM technology.

In this codelab, you build an app that uses the PaLM API to generate haikus based on Google product names. You also use Flutter to create a cross-platform app that displays the haikus.

Prerequisites

- Basic knowledge of Large Language Models (LLMs), such as prompting

- Basic knowledge of Flutter development with Dart

What you'll learn

- How to use the PaLM API from Google.
- How to build a cross-platform Flutter app to display the results.

What you'll need

- The [Flutter SDK](#)
- A [text editor](#), such as Visual Studio Code (VS Code)
- [VS Code setup for Flutter and Dart](#)
- [Android](#) or [iOS](#) setup for Flutter (including emulators and simulators)
- [Desktop](#) setup for Flutter for Windows, Linux, or macOS
- [Web](#) setup for Flutter
- An API key for the PaLM API

Get set up

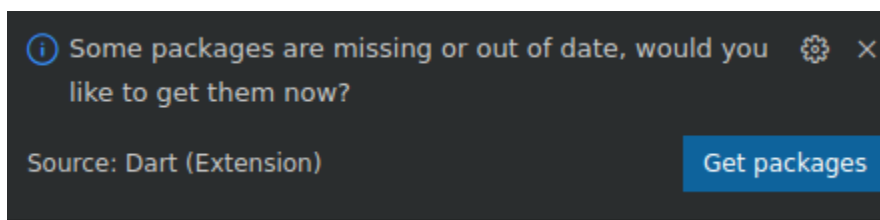
Duration: 10:00

Download the starter code

1. Navigate to [this GitHub repository](#).
2. Click **Code** > **Download zip** to download all the code for this codelab.
3. Unzip the downloaded zip file to unpack a `codelabs-main` root folder. You only need the `haiku-generator` subdirectory, which contains the following folders:
 - The `step0` to `step3` folders, which contain the starter code that you build upon for each step in this codelab.
 - The `finished` folder, which contains the completed code for the finished sample app.

Download the project dependencies

1. In VS Code, click **File** > **Open folder** > **codelabs-main** > **haiku_generator** > **step0** > **lib** > **main.dart**.
2. If you see a VS Code dialog that prompts you to download the required packages for the starter app, click **Get packages**.

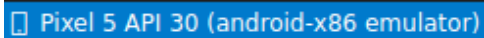


3. If you don't see a VS Code dialog that prompts you to download the required packages for the starter app, open your terminal, and then navigate to the `step0` folder and run the `flutter pub get` command.

Run the starter app


1. In VS Code, ensure that the Android Emulator or iOS Simulator is properly set up and appears in the status bar.

For example, here's what you see when you use Pixel 5 with the Android Emulator:

A screenshot of the VS Code status bar showing the text "Pixel 5 API 30 (android-x86 emulator)" in white on a blue background.

Here's what you see when you use iPhone 13 with the iOS Simulator:

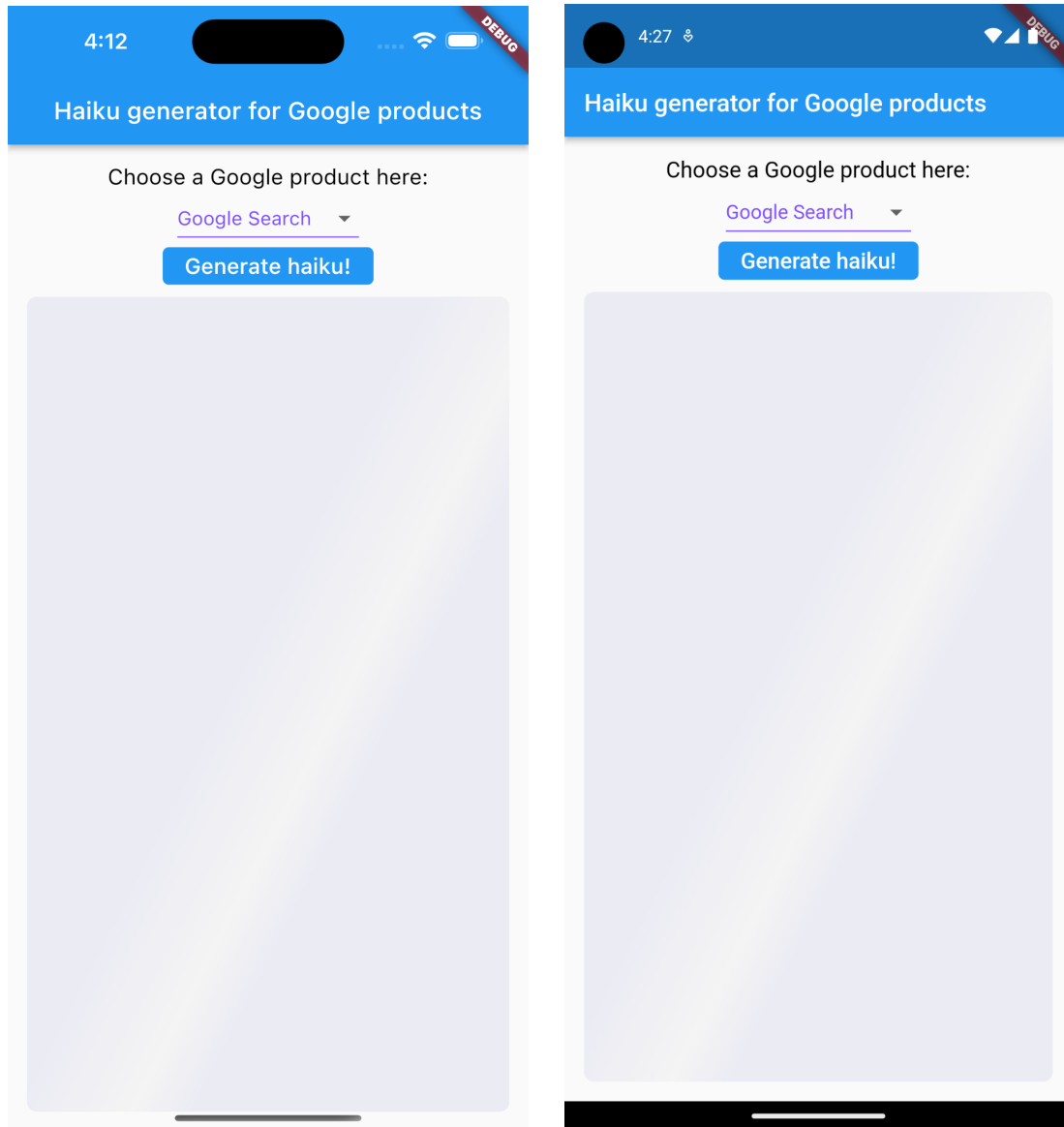
A screenshot of the VS Code status bar showing the text "iPhone 13 (ios simulator)" in white on a blue background.

2. Click  **Start debugging**. The app launches on your Android Emulator or iOS Simulator.

Explore the starter app

In the starter app, notice the following:

- The UI is pretty straightforward.
- There's a drop-down menu that lets users choose a specific Google product.
- After users select the **Generate haiku!** button, the Flutter app sends the built-in prompt to the PaLM API endpoint, which generates haikus.
- The app displays the generated haikus in the text widget after it receives a response. However, if you select **Generate haiku!**, nothing happens because the app can't communicate with the PaLM API yet.



Set up access to the PaLM API

Duration: 15:00

You need an API key to use the PaLM API. At the time of this codelab's publication, the PaLM API is still in private preview.

- To set up your access to the PaLM API, follow the [documentation](#) to create an API key, and then note the key for use later in this codelab.

Add a menu of Google products

Duration: 5:00

Your goal is to generate haikus for Google products. At runtime, the app user can dynamically choose a product from a prepopulated list of product names.

To add a list of Google products to the app, follow these steps:

1. In VS Code, navigate to the `step1/lib/data/repositories/product_repository_impl.dart` file.
2. In the `getAllProducts()` function's body, add the following variable that stores an array of Google product names:

[product_repository_impl.dart](#)

```
var productData = [
  {'productName': 'Google Search'},
  {'productName': 'YouTube'},
  {'productName': 'Android'},
  {'productName': 'Google Maps'},
  {'productName': 'Gmail'}
];
```

Send the request to the PaLM API and decode the response

Duration: 5:00

The product name chosen by the user is combined with the following prompt template:

```
Context: You are an awesome haiku writer.
Message content: Write a cool haiku about {product name}.
```

To send this request to the PaLM API endpoint to generate a haiku, follow these steps:

1. In VS Code, navigate to the `step2/lib/data/repositories/poem_repository_impl.dart` file.
2. In the `getPoems()` function's body, add the following code:

[poem_repository_impl.dart](#)

```
// TODO: Replace YOUR_API_KEY with your API key.
var apiKey = 'YOUR_API_KEY';
const haikuCount = 5;

final url = Uri.parse(

'https://generativelanguage.googleapis.com/v1beta2/models/chat-bis
```

```

on-001:generateMessage?key=$apiKey');
final headers = {'Content-Type': 'application/json'};
final body = jsonEncode({
  "prompt": {
    "context": "You are an awesome haiku writer.",
    "examples": [
      {
        "input": {"content": "Write a haiku about Google Photos."},
        "output": {
          "content":
            "Google Photos, my friend\nA journey of a
lifetime\nCaptured in pixels"
        }
      }
    ],
    "messages": [
      {"content": "Write a cool haiku for $productName"}
    ]
  },
  "candidate_count": haikuCount,
  "temperature": 1,
});

try {
  final response = await http.post(url, headers: headers, body:
body);
  if (response.statusCode == 200) {
    final decodedResponse = json.decode(response.body);
    String haikus = 'Here are $haikuCount haikus about
$productName:\n\n';
    for (var i = 0; i < haikuCount; i++) {
      haikus += '${i + 1}.\n';
      haikus += decodedResponse['candidates'][i]['content'] +
'\n\n';
    }
  }
}

```

```
        return haikus;
    } else {
        return 'Request failed with status:
${response.statusCode}.\n\n${response.body}';
    }
} catch (error) {
    throw Exception('Error sending POST request: $error');
}
```

Replace the `YOUR_API_KEY` string with your API key from earlier..


After the response is received and decoded successfully, the text widget in the UI renders the generated haikus.

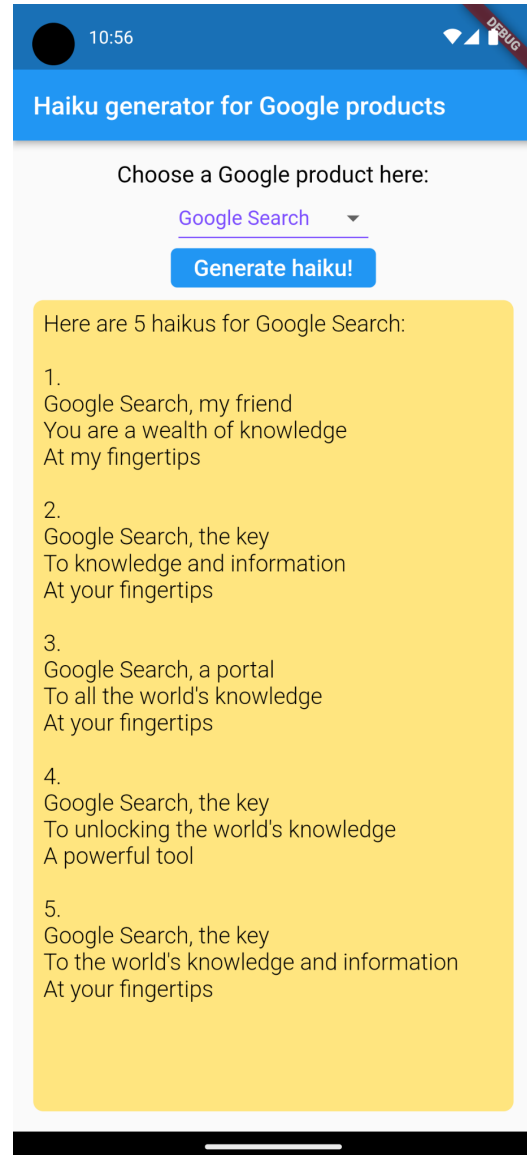
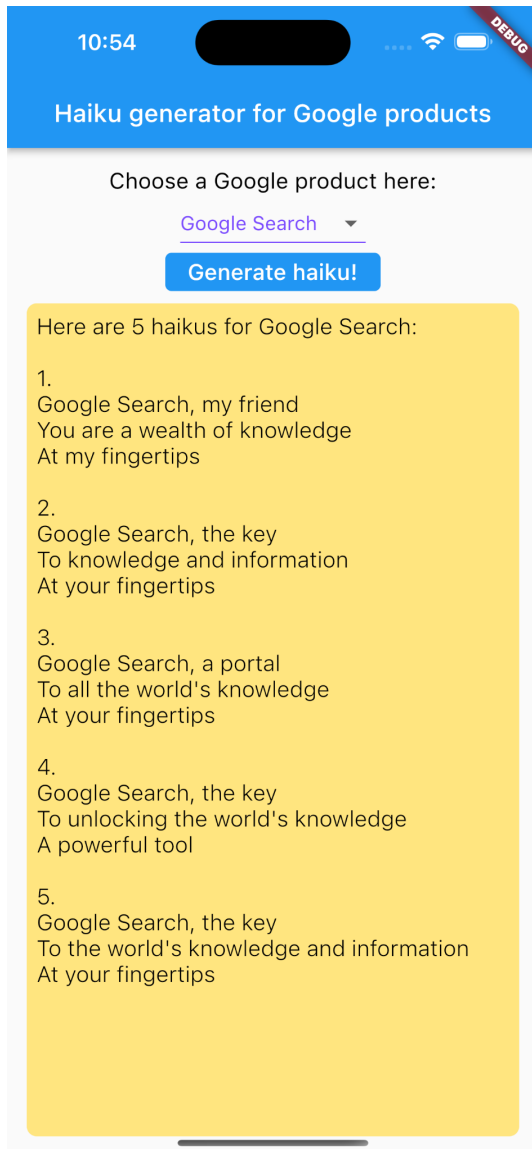
Note: The `examples` field in the payload serves as a reference and tells the API how to generate similar text. This is often called *in-context learning*. You can provide multiple examples.

This codelab doesn't go into detail about how to tune the prompt or think of prompt ideas, but there's an entire field of [prompt engineering](#) that studies how to best write prompts for LLMs. For more information, play with other prompts and adjust the UI accordingly, and see this [Prompt guide](#) for this emergent domain.

Run the app on mobile platforms

Duration: 5:00

1. In VS Code, set the target device to an Android or iOS device.
2. Click  **Start debugging**, and then wait for the app to load.
3. Select a product from the drop-down menu, and then select **Generate haiku!**. The app displays a haiku about the selected product.




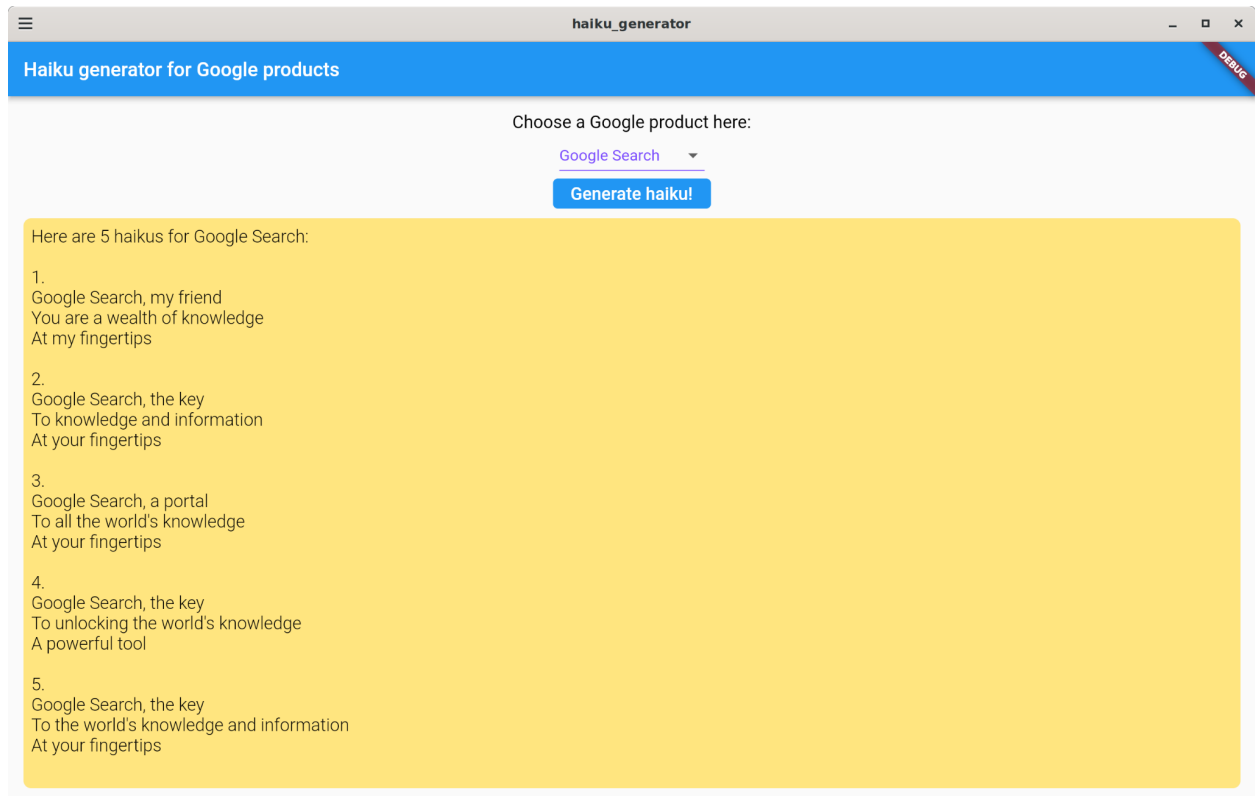
Run the app on desktop platforms

Duration: 10:00

In addition to Android and iOS, Flutter also supports desktop platforms, including Linux, macOS, and Windows.

Run the app on Linux

1. In VS Code, set the target device to **Linux (linux-x64)**.
2. Click  **Start debugging**, and then wait for the app to load.
3. Choose a product from the drop-down menu, and then select **Generate haiku!**.



Run the app on macOS


For macOS, you need to set up appropriate entitlements because the app sends HTTP requests to the backend. For more information, see [Entitlements and the App Sandbox](#).

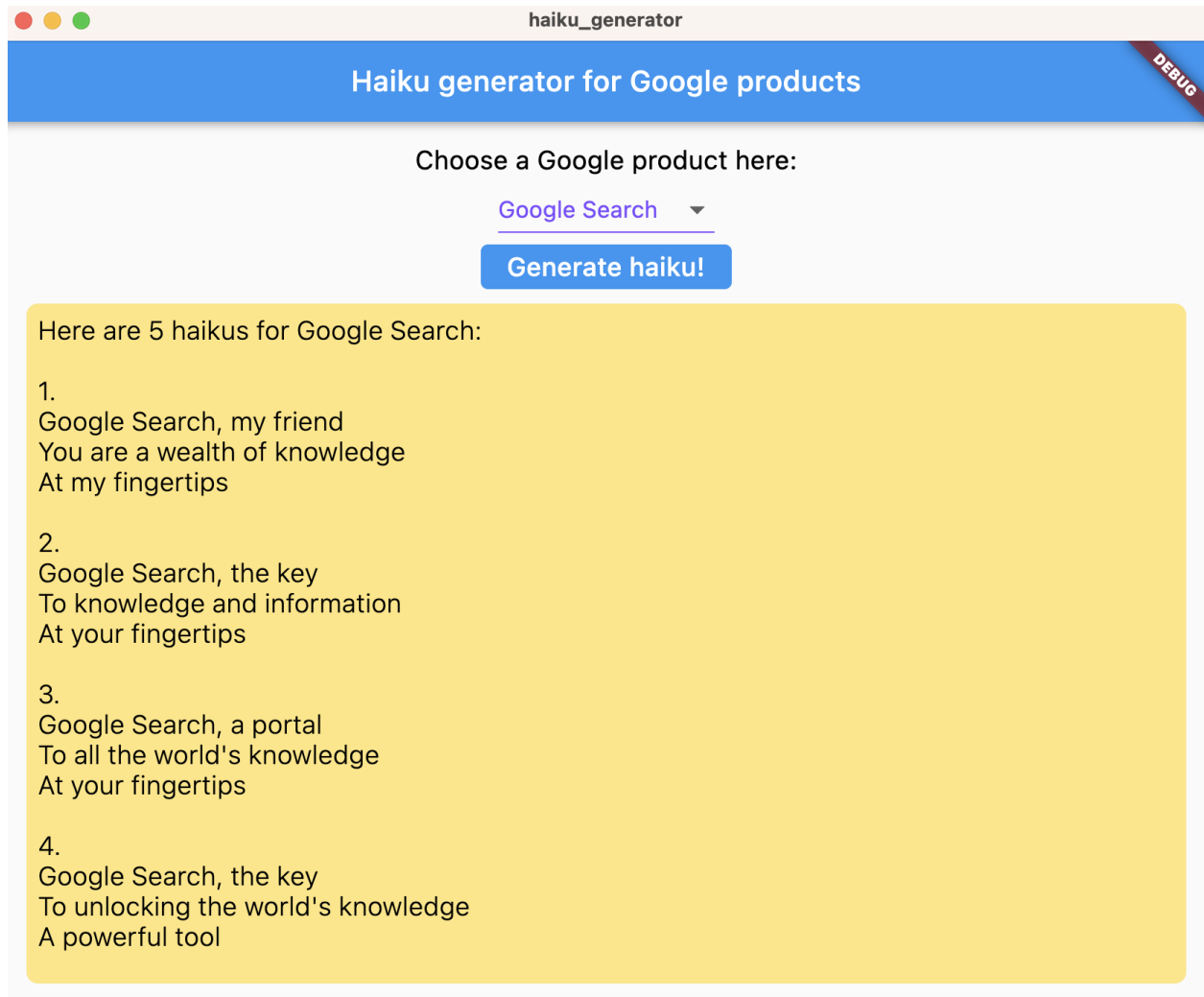
To run the app on macOS, follow these steps:

1. In the `step3/macos/Runner/DebugProfile.entitlements` and `step3/macos/Runner/Release.entitlements` files, add the following code:


[DebugProfile.entitlements](#) | [Release.entitlements](#)

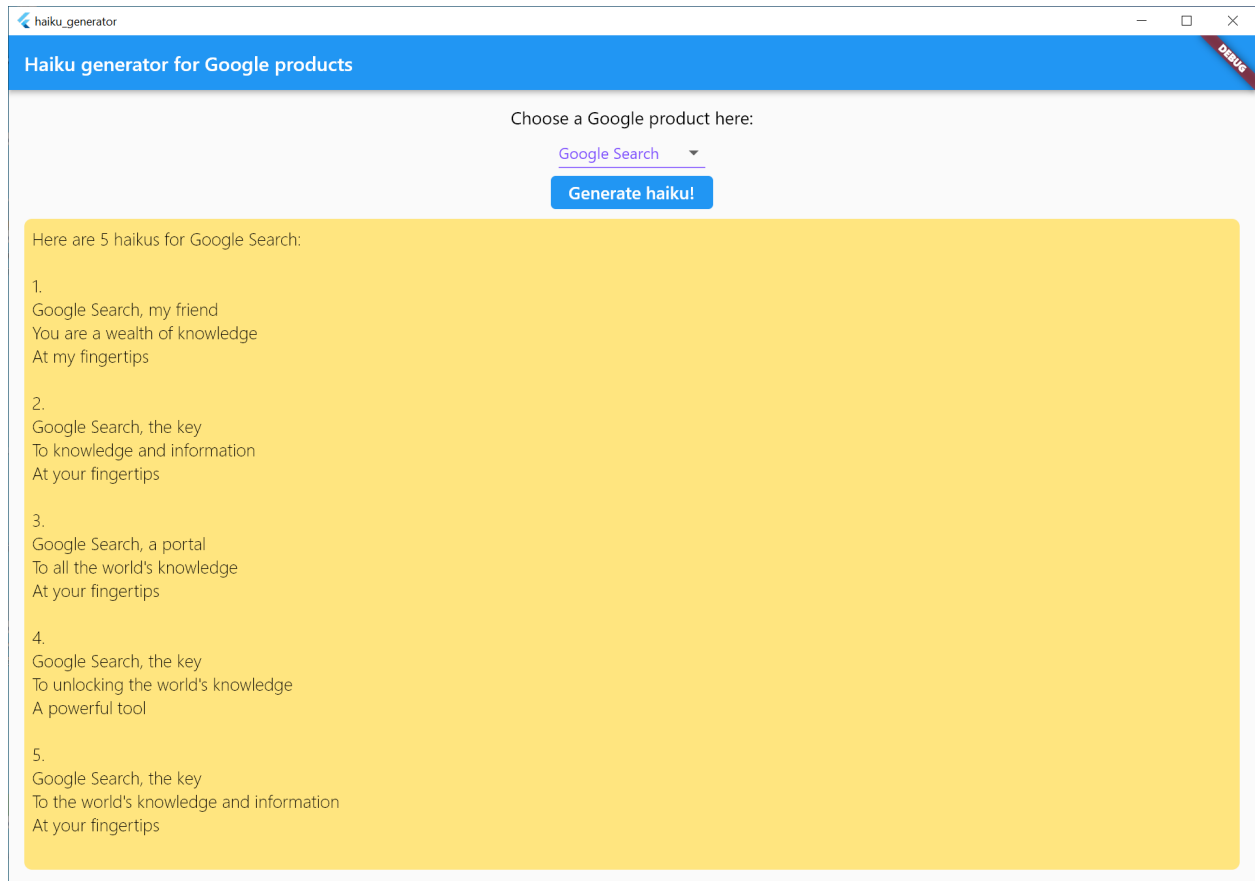
```
<key>com.apple.security.network.client</key>
<true/>
```

2. In VS Code, set the target device to **macOS (darwin)**.
3. Click  **Start debugging**, and then wait for the app to load.
4. Choose a product from the drop-down menu, and then select **Generate haiku!**.



Run the app on Windows

1. In VS Code, set the target device to **Windows (windows-x64)**.
2. Click  **Start debugging** and then wait for the app to load.
3. Choose a product from the drop-down menu, and then select **Generate haiku!**.




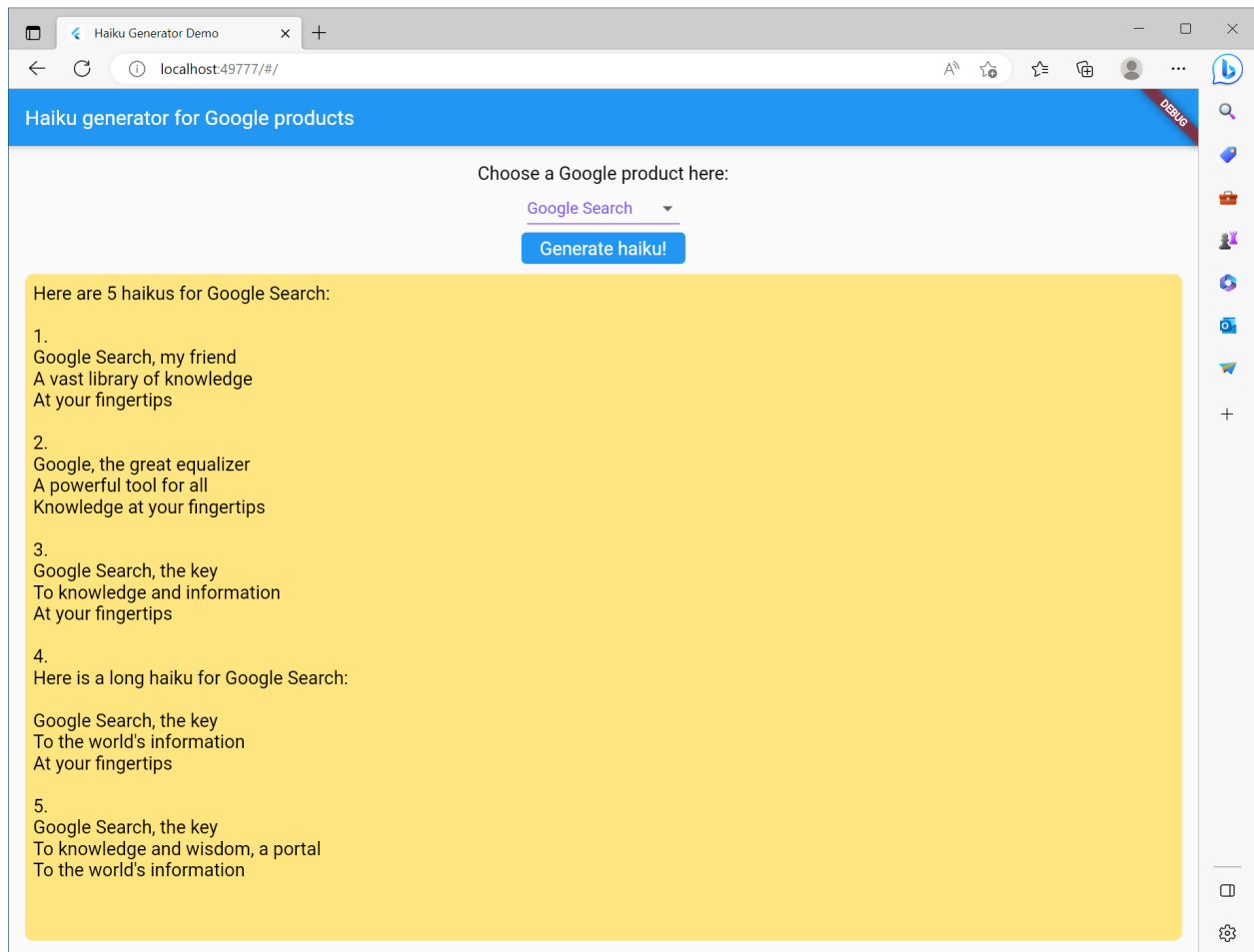
Run the app on the web platform

Duration: 2:00

You can also add web support to the Flutter app. By default, the web platform is automatically enabled for Flutter apps, so all you need to do is to launch it.

To run the app on the web platform, follow these steps:

1. In VS Code, set the target device to **Chrome (web-javascript)**.
2. Click  **Start debugging**, and then wait for the app to load in Google Chrome.
3. Choose a product from the drop-down menu, and then select **Generate haiku!**.



Congratulations

Duration: 1:00

You built a full-stack app that generates haikus about Google products! Although the app only generates haikus for select Google products, you can easily change the prompt and generate your desired text. Now that you know how to use the PaLM API, you can build amazing apps with the incredible power of LLMs!

Learn more

- [PaLM API & MakerSuite: An approachable way to start prototyping and building generative AI apps](#)
- [Google Generative AI](#)
- [Flutter](#)