# Introduction

# What are we doing?

Windows folks, use this: Prototyping_AI_Tools_101_Windows.docx

## Main objective:

- Building a very simple application which uses an AI in the back end and has some sort of front-end.

## Nested objectives:

- Learning to use command line tools
- Setting up your local coding environment
- Learning to set up and call an API
- Use command line tools
- Use node.js to setup a basic application
- Setting up a server and running a local application
- Learn to remix code examples

**We will work in pairs (pls sit close to your pair programmer)**

# Why are we doing this?

There is some concern around not knowing how to use code, and some desire from students to do more programming from the ground up. So we will do some live coding with going over all the setup instructions.

Note that this class is not an intro to programming class, but it should set you up with enough skill to learn how to remix code to create tools that ***you*** want to build.

This is also just using one library, in one type of AI. But you should be able to extend this skill to other types of AI. That's on you to practice other models, and other libraries!

# Prior Concepts

- What's an API key used for?
- Which AI service's API did we use so far?
- Which AI models did we use?
- What are these models in general called?
- Which programming language did we use in the homework?

# Installing Stuff

# Installing Stuff

## Installing Tools:

A. **Installing Python:** [https://www.python.org/downloads/](https://www.python.org/downloads/)
(so that you can write python code on your computer)
Make sure to follow the instructions for your computer type.

B. **Installing Node:** [https://nodejs.org/en/download](https://nodejs.org/en/download)
(so that you can install relevant packages)

C. **Installing Sublime Text:** [https://www.sublimetext.com/](https://www.sublimetext.com/)
(make sure to add the application to the application folder)

## Terminal:

Terminal (Command Prompt in Windows) is an application where you can talk to your computer using commands (computer speak). Every computer has one. Just search and open it.

Use this command in your **Terminal** to add command line tools for your Sublime text.

```
echo 'export PATH="/Applications/Sublime Text.app/Contents/SharedSupport/bin:$PATH"' >> ~/.zprofile
```

Also add this line

```
ln -s "/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl" /usr/local/bin/subl
```

(If your MacOS is less that 10.15, use this command):

```
echo 'export PATH="/Applications/Sublime Text.app/Contents/SharedSupport/bin:$PATH"' >> ~/.bash_profile
```

Also add this line

```
ln -s "/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl" /usr/local/bin/subl
```

## Programming on your machine.

We will first use Python. Start the program Terminal (Command Prompt for windows).

Some basic terminal commands:

| | |
|---|---|
| `cd` | Return to home directory |
| `cd ..` | Return one step back in folders |
| `mkdir` | Make new directory (use mkdir projects to make a projects folder) |
| `ls` | List everything in this directory |
| `cd [path to folder]` | Takes you to that folder (use cd projects to go to the projects folder) |
| `open .` | Open this folder |
| `python` | To run a python file (e.g. python [filename]) |
| `python3` | To run a python file (python3 [file name]) |
| `Ctr + C` | Exit out from current running command (e.g. if you're running a server) |
| `subl` | To open something in sublime text |

Use your command line to create a new folder called "**projects**" in your home directory.

Gemini Developers

# Getting Set Up with Gemini

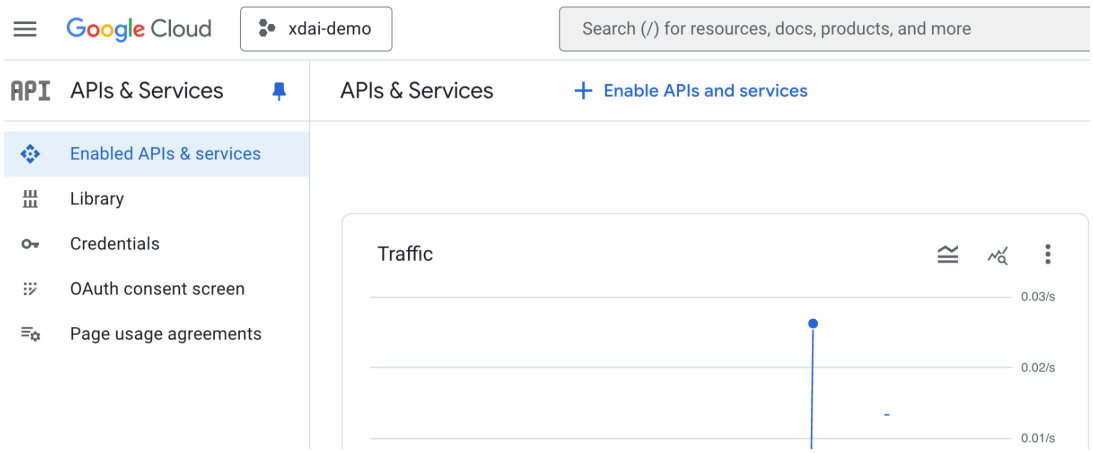
We will use Gemini for this example, but you can use any API service. They all have their own documentation that you can follow along from their developer site.

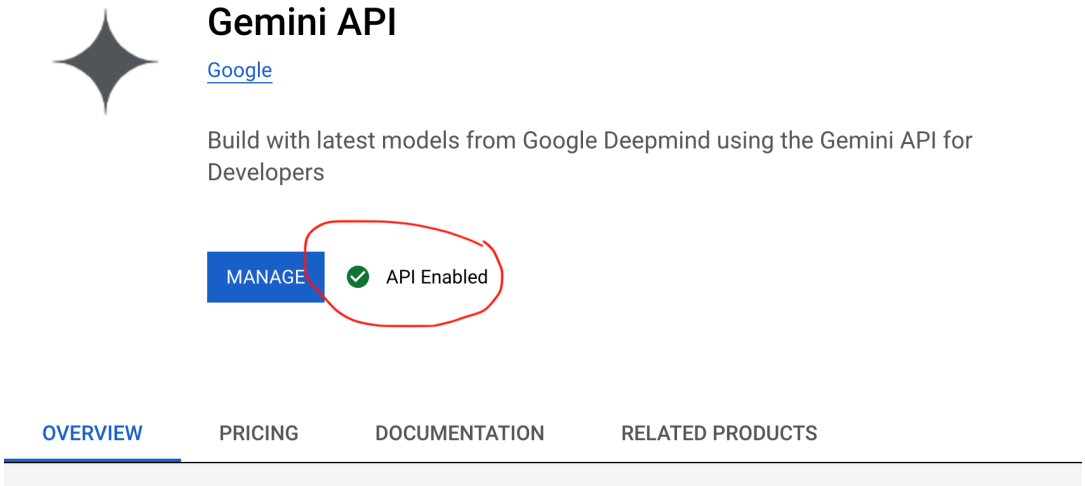
- You actually have built apps with Gemini before. **What was that practice called?**

**A. Create a Gemini API key: [https://console.cloud.google.com/apis](https://console.cloud.google.com/apis)**

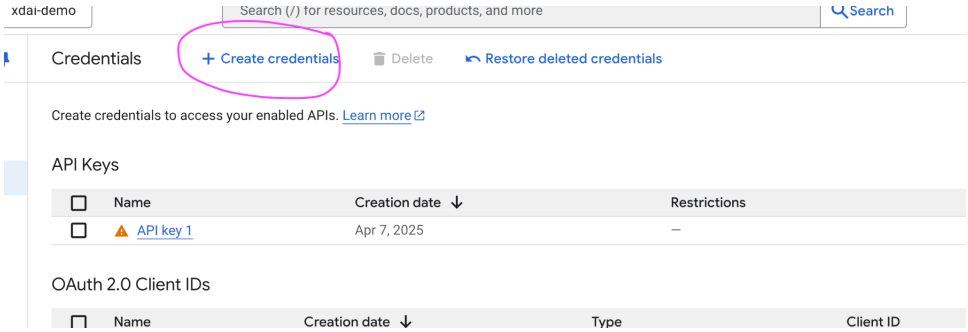
a. You will need to create some credentials and accept terms and services.

b. You will need to enable API services (use the +Enable APIs and services button) → Enable Gemini API. It should show API enabled when you go to Gemini-API in your APIs and services.




c. Then go to Create Credentials -> API Keys, to get your key.




B. **Install the Gemini API on your computer:**
(Install both the Python and the Javascript versions)

Go to your Terminal App. Use these commands:

(for Python)
`pip3 install -q -U google-genai`

(if this gives you an error, use **pip3** instead of pip)

(for Javascript)
`npm install @google/genai`

C. **Then follow this step to put your key in your dev environment:**
https://ai.google.dev/gemini-api/docs/api-key (make sure to select mac for macs, and windows for windows)

This is how your computer will know this password forever!

You do need to add your credit card info, but there are a bunch of free credits available to begin with, so it should not cost you anything. DO NOT push your API keys anywhere public (e.g. github)

**D. Follow the Gemini quickstart steps to write some basic Python scripts to call the Gemini API:**

We will start with **Python** for ease of understanding. We will later move to **node.js**, because Javascript will help you make front-end tools.

Create a new python file (ideally in your projects folder). You can navigate to your project folder on your computer using the terminal.

```
cd projects
```

Then create a new python file in there using the following command:

```
subl basicGemini.py
```

This should open up your python file in sublime text. Now add the following code in there:

```
from google import genai

# The client gets the API key from the environment variable
`GEMINI_API_KEY`.
client = genai.Client(api_key="YOUR_API_KEY")

response = client.models.generate_content(
    model="gemini-2.5-flash", contents="Explain how AI works in a few
words"
)
print(response.text)
```

If your environment variable isn't working, use this command for client:

```
client = genai.Client(api_key="YOUR_API_KEY")
```

Try to understand this code line by line.

Let's try a *thinking model* next.

## Try on your own:

Try some more complex text generation examples here:
https://ai.google.dev/gemini-api/docs/text-generation

Try to make some changes to the scripts to make them relevant to your project.

When you forget stuff, documentation is available here:
https://ai.google.dev/gemini-api/docs/libraries

# Making Apps

# Making Web Apps with Gemini

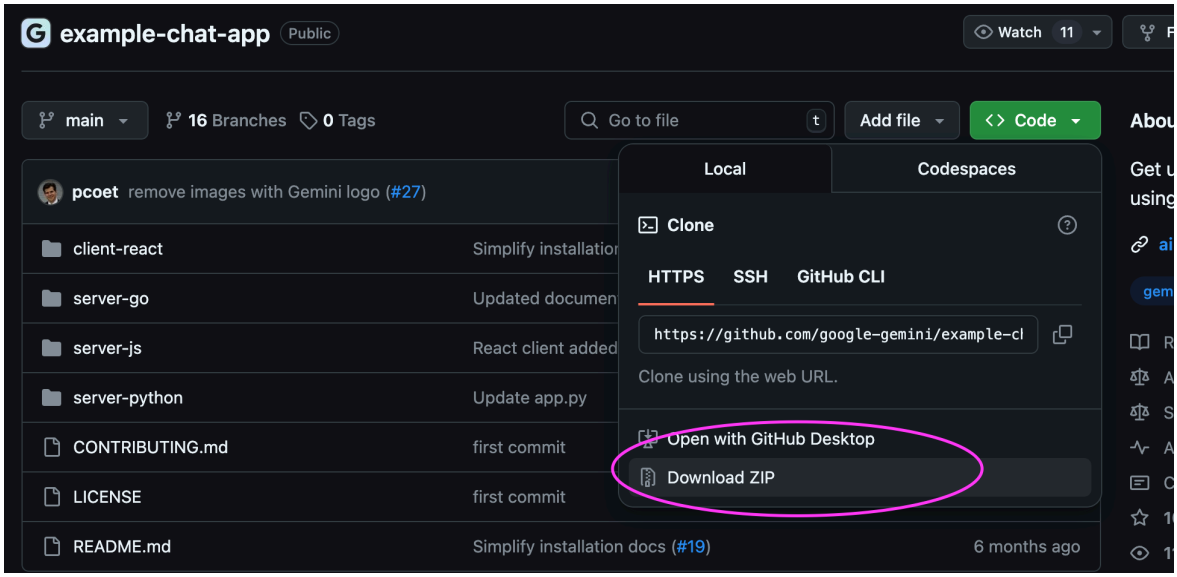
Front-end is not the easiest stuff to prototype for first timers. If you already feel confident in making front end in React.js, or p5.js, or other javascript libraries, go for it! If not, use code templates by other creators that you can easily modify.

Here is a great set of example codes to use:
https://ai.google.dev/gemini-api/tutorials/web-app?lang=node

For just trying the AI out, and building a back-end: Python is just fine!
For front end: Make sure to select the node.js option for code, and NOT python.

NOTE: You may see some commands like “git ..” these are basically using github in your Desktop to download stuff. You may not have github installed in your command line, and will get an error. You can solve this by:

1. Download the code directly for here:
   https://github.com/google-gemini/example-chat-app




2. Unzip and move this folder to your projects folder.
3. Go to the folder from your terminal ( `cd projects/example-chat-app` )
4. Now basically follow the configuration steps below (they are also available here):

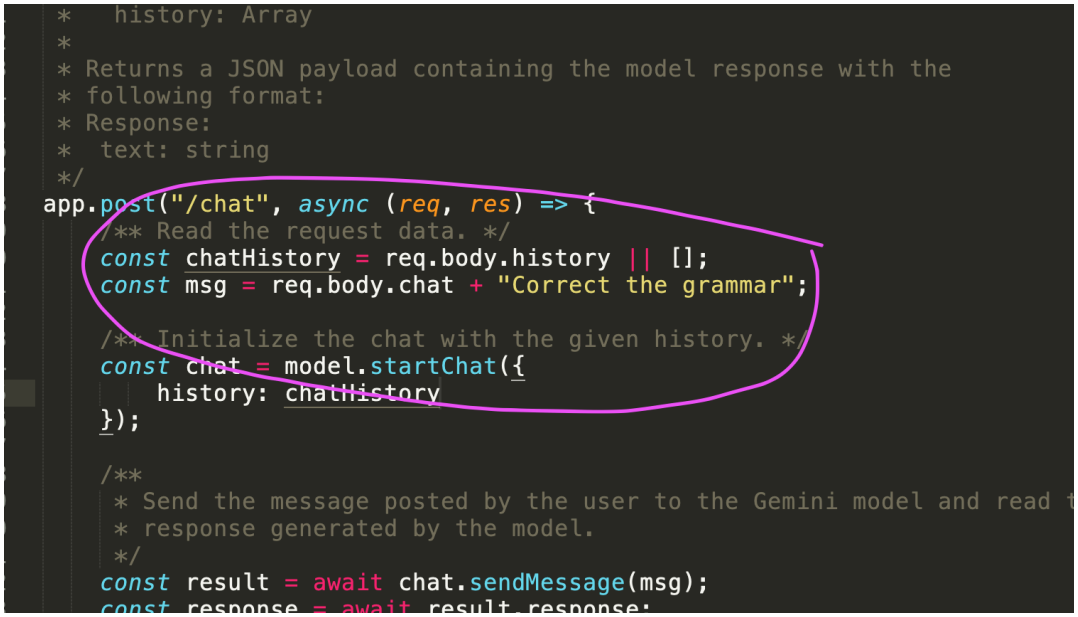
   Install the app:
   a. Navigate to the app directory, (`cd server-js`) (i.e. where package.json is located).

b. Run **`npm install`**
c. Run **`cp .env.example .env`**
d. Open the .env file using any text editor to add your Gemini API key. (example, **`subl .env`**).
e. Tell your code what the environment file is, where your secret key is: **`node --env-file=.env app.js`**
f. Leave this running. Start a new tab in your terminal.

5. In a NEW TERMINAL TAB!! Run the app:
    a. Navigate to the app directory, **`cd client-react`**
    **b.** Run **`npm install`**
    c. Run the application with the following command: **`npm run start`**
    d. The client will start on **`localhost:3000`**
    Go to any browser (e.g. Chrome, and type localhost:3000 to view your app running). You basically started a local server and are running the app.

    **Note for 2025 folks:** This app will first give you an error. This is because the AI model they are using is deprecated. We will change it to gemini-2.5 in class.

6. Now make changes to the code (you can open them in your sublime text, to make the chatbot do something more interesting - e.g. give you grammar feedback). T
    a. The place to make quick changes in the back end (what the app does) would be **server-js > app.js** file. Open it in sublime text or another code editor. See below, the change I made (just said “correct the grammar” and now my chatbot is doing grammar correction)
    b. The place to make quick changes in the front end (what the app looks like) would be client-react > src>components. Example, I changed the **conversationDisplayArea.js** file to say **“I am here to correct the grammar of your text”** instead of **“How can I help you today”**




**For help: Slack Safinah if you need help, or come to office hours.**

## Try on your own:

- If you feel confident with the steps so far, you can try more advanced stuff, like try to use a fine-tuned model API (create it here), and use that model's API instead of this one.
- You don't have to use Gemini. See this documentation to do the same thing with OpenAI: https://github.com/openai/openai-responses-starter-app
- You don't have to build a chatbot. See these for example of other cool stuff you can do: https://ai.google.dev/gemini-api/tutorials/web-app?lang=node

# Some general resources for making with AI

**Tools:**

1. [Exploring Art](https://explore-art-204c2f2aab0f.herokuapp.com/)
2. [SketchRNN](https://magenta.tensorflow.org/assets/sketch_rnn_demo/index.html)
3. [Magic sketches](https://magic-sketchpad.glitch.me/)
4. [Visual anagrams](https://dangeng.github.io/visual_anagrams/)
5. [Illusion Diffusion](https://huggingface.co/spaces/AP123/IllusionDiffusion)
6. [Segment anything](https://segment-anything.com/demo#)

**Toolkits:**

1. [Tensorflow](https://www.tensorflow.org/)
2. [Tensorflow.js](https://www.tensorflow.org/js)
3. [Pytorch](https://pytorch.org/)
4. [P5.js](https://p5js.org/)
5. [ML5.js](https://ml5js.org/)
6. [Huggingface](https://huggingface.co/)

**Notebooks:**

1. [Vision notebook](https://colab.research.google.com/drive/1PSrNYNf4iemAhoffa9_pWjVrlzrEXjFX?usp=sharing)
2. [Tensorflow notebook](https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb)

**Cool Datasets:**

1. [Kaggle](https://www.kaggle.com/datasets)
2. [HuggingFace Datasets](https://huggingface.co/datasets)

# [ignore] Old doc with all the things

This is a very basic guide for end-to-end prototyping of AI tools including front and back-end.

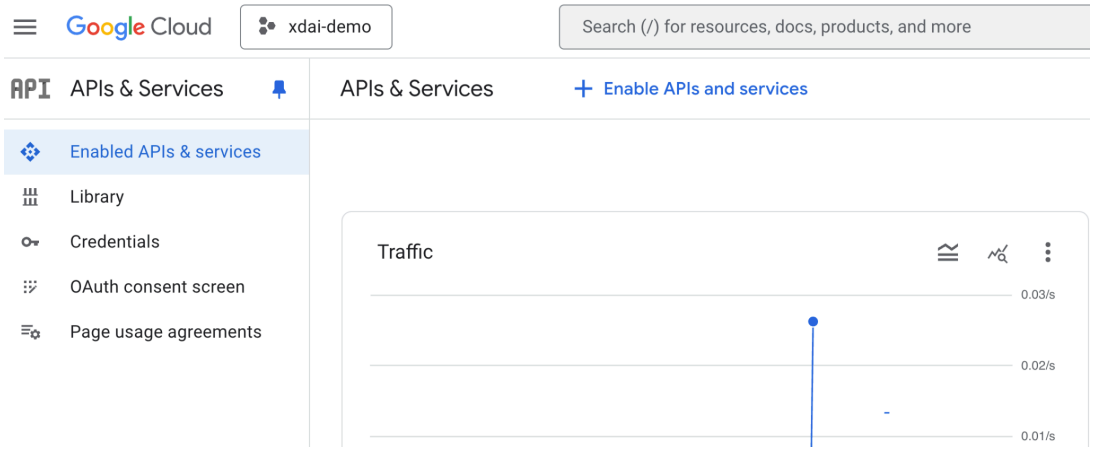
A. **Installing Python:** [https://www.python.org/downloads/](https://www.python.org/downloads/)
(so that you can write python code on your computer)

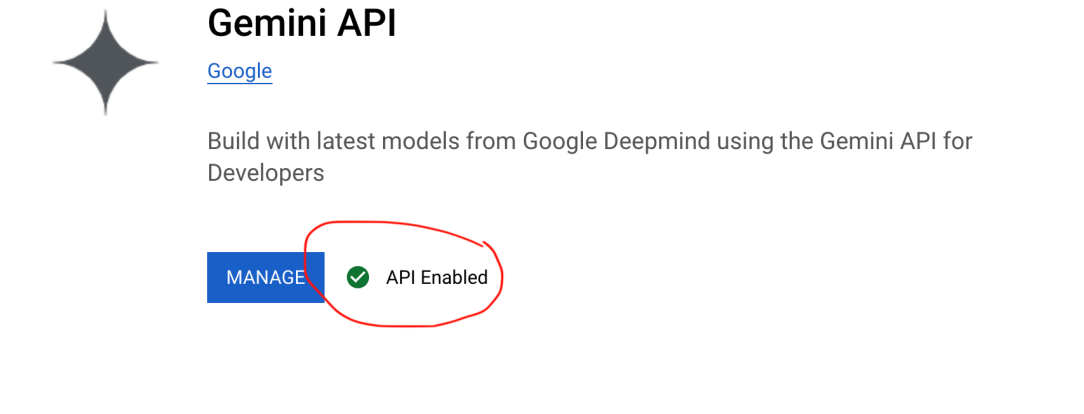
B. **Installing Node:** [https://nodejs.org/en/download](https://nodejs.org/en/download)
(so that you can install relevant packages)

We will use Gemini for this example, but you can use any API service. They all have their own documentation that you can follow along from their developer site.

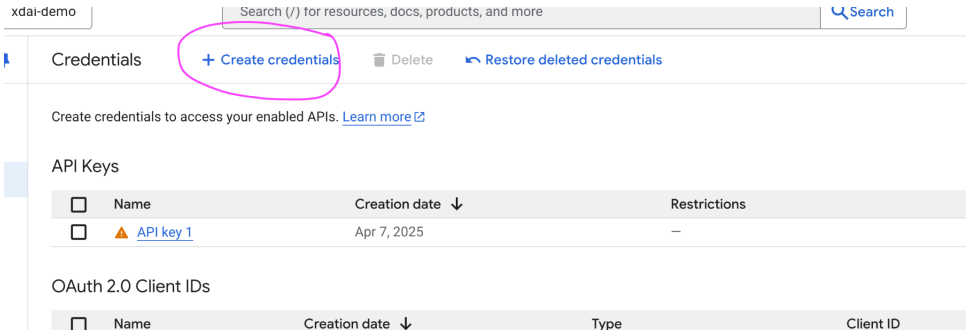
You do need to add your credit card info, but there are a bunch of free credits available to begin with, so it should not cost you anything. DO NOT push your API keys anywhere public (e.g. github)

C. **Create a Gemini API key:** [https://console.cloud.google.com/apis](https://console.cloud.google.com/apis)
   a. You will need to create some credentials and accept terms and services.
   b. You will need to enable API services (use the +Enable APIs and services button) → Enable Gemini API. It should show API enabled when you go to Gemini-API in your APIs and services.






c. Then go to Create Credentials -> API Keys, to get your key.




d. Install the Gemini API on your computer:
https://ai.google.dev/gemini-api/docs/libraries
(Install both the Python and the Javascript versions)

e. Then follow this step to put your key in your dev environment:
https://ai.google.dev/gemini-api/docs/api-key (make sure to select mac for macs, and windows for windows)

## D. Programming on your machine.

We will first use Python. Start the program **Terminal (Command Prompt** for windows).

Some basic terminal commands:

| | |
|---|---|
| `cd` | Return to home directory |
| `cd ..` | Return one step back in folders |

| | |
|---|---|
| `mkdir` | Make new directory (use `mkdir projects` to make a projects folder) |
| `ls` | List everything in this directory |
| `cd [path to folder]` | Takes you to that folder (use `cd projects` to go to the projects folder) |
| `open .` | Open this folder |
| `python` | To run a python file (e.g. python [filename]) |
| `python3` | To run a python file (python3 [file name]) |
| `Ctr + C` | Exit out from current running command (e.g. if you're running a server) |

Also install Sublime text (or any other text editor) to write code.
https://www.sublimetext.com/download

Follow the Gemini quickstart steps to write some basic Python scripts to call the Gemini API:
https://ai.google.dev/gemini-api/docs/quickstart?lang=python

You can start with Python for ease of understanding. We will later move to node.js, because Javascript will help you make front-end tools.

Create a new python file (ideally in your projects folder). You can navigate to your project folder on your computer (cd projects)

Try some more complex text generation examples here:
https://ai.google.dev/gemini-api/docs/text-generation

Try to make some changes to the scripts to make them relevant to your project.

### E. Let's start making some web apps.

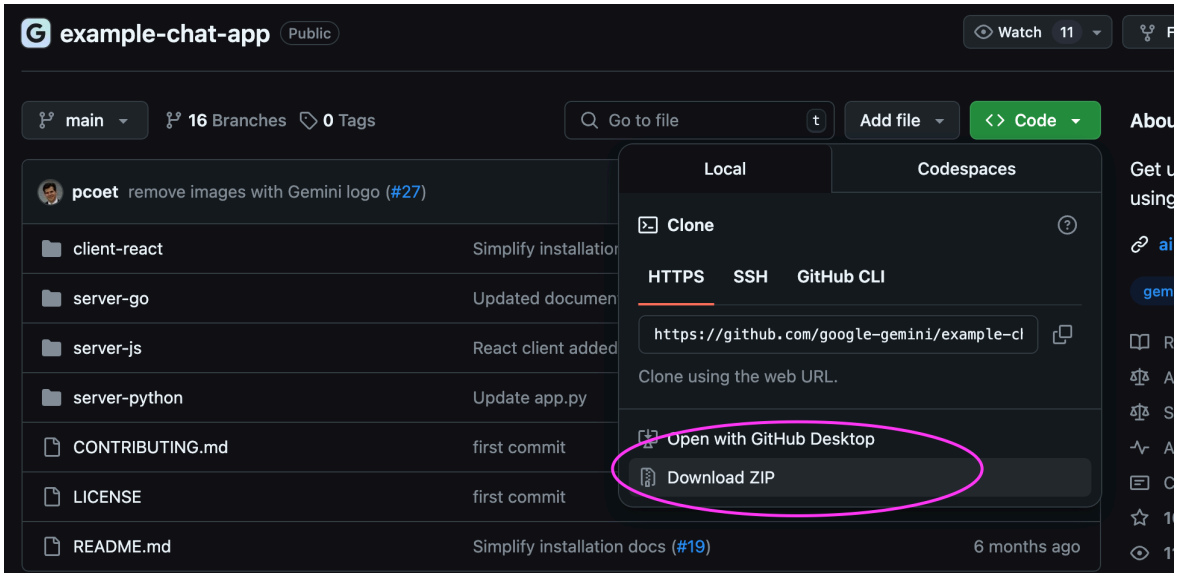
Front-end is not the easiest stuff to prototype for first timers. If you already feel confident in making front end in React.js, or p5.js, or other javascript libraries, go for it! If not, use code templates by other creators that you can easily modify.

Here is a great set of example codes to use:
https://ai.google.dev/gemini-api/tutorials/web-app?lang=node

For just trying the AI out, and building a back-end: Python is just fine!
For front end: Make sure to select the node.js option for code, and NOT python.

NOTE: You may see some commands like “git ..” these are basically using github in your Desktop to download stuff. You may not have github installed in your command line, and will get an error. You can solve this by:

1. Download the code directly for here: https://github.com/google-gemini/example-chat-app




2. Unzip and move this folder to your projects folder.
3. Go to the folder from your terminal ( **`cd projects/example-chat-app`** )
4. Now basically follow the configuration steps here: https://github.com/google-gemini/example-chat-app/tree/main/server-js

   Install the app:
   a. Navigate to the app directory, (**`cd server-js`**) (i.e. where package.json is located).
   b. Run **`npm install`**
   c. Run **`cp .env.example .env`**
   d. Open the .env file using any text editor to add your Gemini API key. (example, **`vi .env`**). See here how to edit vi files here: https://docs.oracle.com/cd/E19683-01/806-7612/editorvi-43/index.html
   e. Tell your code what the environment file is, where your secret key is: **`node --env-file=.env app.js`**
   f. Leave this running. Start a new tab in your terminal.
5. In a NEW TERMINAL TAB!! Run the app:
   a. Navigate to the app directory, **`cd client-react`**
   b. Run **`npm install`**
   c. Run the application with the following command: **`npm run start`**

d. The client will start on `localhost:3000`
   Go to any browser (e.g. Chrome, and type localhost:3000 to view your app running). You basically started a local server and are running the app.

6. Now make changes to the code (you can open them in your sublime text, to make the chatbot do something more interesting - e.g. give you grammar feedback). T
   a. The place to make quick changes in the back end (what the app does) would be **server-js > app.js** file. Open it in sublime text or another code editor. See below, the change I made (just said “correct the grammar” and now my chatbot is doing grammar correction)
   b. The place to make quick changes in the front end (what the app looks like) would be client-react > src>components. Example, I changed the **conversationDisplayArea.js** file to say **“I am here to correct the grammar of your text”** instead of **“How can I help you today”**

```
 *   history: Array
 *
 * Returns a JSON payload containing the model response with the
 * following format:
 * Response:
 *  text: string
 */
app.post("/chat", async (req, res) => {
    /** Read the request data. */
    const chatHistory = req.body.history || [];
    const msg = req.body.chat + "Correct the grammar";

    /** Initialize the chat with the given history. */
    const chat = model.startChat({
        history: chatHistory
    });

    /**
     * Send the message posted by the user to the Gemini model and read t
     * response generated by the model.
     */
    const result = await chat.sendMessage(msg);
    const response = await result.response;
```

**Slack Safinah if you need help, or come to office hours.**

If you feel confident with the steps so far, you can try more advanced stuff, like try to use a fine-tuned model API (create it here - I showed this in a previous class), and use that model’s API instead of this one.

—

Other AI services:

You don’t have to use Gemini. See this documentation to do the same thing with OpenAI: https://github.com/openai/openai-responses-starter-app

You don’t have to build a chatbot. See these for example of other cool stuff you can do:
https://ai.google.dev/gemini-api/tutorials/web-app?lang=node