TECHNOVATION
Girls

Generative AI Guide for Teachers

Table of Contents

Table of Contents
Syllabus: Generative AI for Teachers
Introduction to Generative AI
What is Generative AI?
1. Content generation
2. Creating custom materials
3. Automation of repetitive tasks
ChatGPT (OpenAI)
Gemini (Google)
Github Copilot (Github - OpenAI)
Poe
Canva AI
Why use this feature in Technovation Girls?
Important: Is it paid? Is it worth using?
Best practices for writing prompts
What is a prompt?
Elements of a good prompt
Key recommendations when writing prompts
Programming Basics for Teachers
Algorithm
Variable
Conditionals
Modify the example step by step:
Use ChatGPT to help you:
Create your own situation with conditionals!
Closing/reflection:
Loops
Try the following:
Part 2: while Loop
Try these modifications:
Reflect:
Part 1: for Loop
Part 2: while Loop
Part 3: Time to create!
Closing and reflection
Creating a Web Application with Generative AI
HTML
CSS
JavaScript
And how does it compare to a mobile app?

    - Python
- Creating a web app: “My goals for the month”
    - Option 1: Web app with HTML, CSS and JavaScript
        - Step 1: Prepare your workspace
        - Step 2: Request the code from ChatGPT using an effective prompt
        - Step 3: Test the code without installation
        - Step 4: Basic Debugging
            - Basic debugging techniques
            - Debugging in CodePen
            - Request corrections from genAI
            - Here are some useful phrases you can use to make adjustments or improvements to the generated code:
            - Example of debugging process
- Task: We want to make a mini app where the user types their name, clicks a button, and a message like this appears:
- “Hello, Mariana!”
- 1. Initial code
    - HTML (in the HTML tab):
    - JavaScript (in the JS tab):
- 2. What happens when you try it?
- 3. Guided activity
    - Step A. Observe
    - Step B. Hypothesis
    - Step C. Ask ChatGPT
- 4. What is the response from genAI?
- 5. Apply the correction
- Final reflection
        - Step 5. Publishing the my-goals web application
        - Publish with Tiiny.host
        - This isn't the only option for publishing your web app. We have two additional options that might interest you. To avoid cluttering this section, we invite you to check them out in the appendix if you wish to explore them.
        - Final recommendation
- Option with Visual Studio Code
            - Step 1. Install Visual Studio Code on your computer
            - Step 2. Open the folder in Visual Studio Code
            - Step 3. Create your code files
            - 1. HTML file
            - 2. CSS file
            - 3. JavaScript file
            - Step 4. Test your code
    - Option 2. Web app with Python and Streamlit
        - What is Streamlit?
            - Step 1. Ask ChatGPT for help generating the code
            - Go to https://chat.openai.com and type the following prompt:

- Step 2. Create your app in Streamlit
- Step 2. Test and debug your app
- Step 3. Commit your changes to Github and Streamlit
- How to continue learning with ChatGPT?
- Important Notes
- What if something goes wrong?
- OPTIONAL ACTIVITY
- Convert a web app to a mobile app
  - WebintoApp
    - Steps to convert your web app into a mobile app
    - 1. Register on WebIntoApp
    - 2. Click on the "Maker" button
    - 3. Select the "Online URL" option
    - 4. Customize your app
    - 5. Create the app
    - 6. Install the app on your cell phone (Android only)
  - Creating a Mobile App with AI + App Inventor
    - Step 1: Request support from GPT Chat
    - Step 2: Open App Inventor and follow the steps
  - Creating a Mobile App with AI + AppInventor (Chatbot Version)
- Ethical aspects and challenges of Generative AI
  - Biases in AI: Why do they occur?
  - Data Privacy: What to Share and What Not to Share
  - Responsible use in education
  - Conclusion
- Appendix
- What is Generative AI?
  - How does Generative AI work?
    - 1. Training (foundation model)
    - 2. Adjustment (model customization)
    - 3. Generation and continuous improvement
  - Neural networks and text prediction
- Other options for publishing your web app
  - Option 2: Publish with Netlify Drop
  - Option 3: Publish with GitHub Pages
- References

Syllabus: Generative AI for Teachers

Topic	Objective / Key Contents
Introduction to Generative AI	Understand what it is and how it is applied in education. ▪ Simple definition ▪ Content generation, personalization and task automation
AI Tools for Educators	Know accessible platforms: ▪ ChatGPT (OpenAI) ▪ Gemini (Google) ▪ Github Copilot ▪ Canva AI, Poe ▪ Claude
How to create good prompts	Learning to give effective instructions: ▪ What is a prompt? ▪ Key elements: context, role, action, format, audience
Programming Fundamentals	Simple introduction to: ▪ Algorithms ▪ Variables ▪ Conditionals ▪ Loops
Create a web app with AI	Using AI to build a web app: ▪ HTML, CSS, JS, Python básicos ▪ Request code, adapt it and test it ▪ Publish a GitHub Pages / Netlify / Tiinity
Create a simple mobile app	Create a mobile app from scratch: ▪ WebInToApp ▪ App Inventor
Ethics and challenges of AI	Reflect on its responsible use: ▪ Biases ▪ Privacy ▪ Critical evaluation

Introduction to Generative AI

What is Generative AI?

Before we begin, it is essential to understand what Artificial Intelligence (AI) is, especially **Generative AI**, which will be used throughout the manual. In general terms, **AI is the ability of computers to perform tasks that require human intelligence, such as learning, adapting, and making decisions.**

While **traditional AI follows instructions to solve specific tasks**, **Generative AI can create original content**—such as text, images, or music—based on what you've learned. This ability to generate new things sets it apart and makes it a powerful and creative tool.

If you want to know more about AI and how it works, we invite you to review the Appendix of this manual.

Educational uses of Generative AI

Generative artificial intelligence can be a powerful ally in teaching. Below, we share three ways you can leverage it in your teaching practice, along with small activities to begin exploring its potential without the need for technical knowledge.

1. Content generation

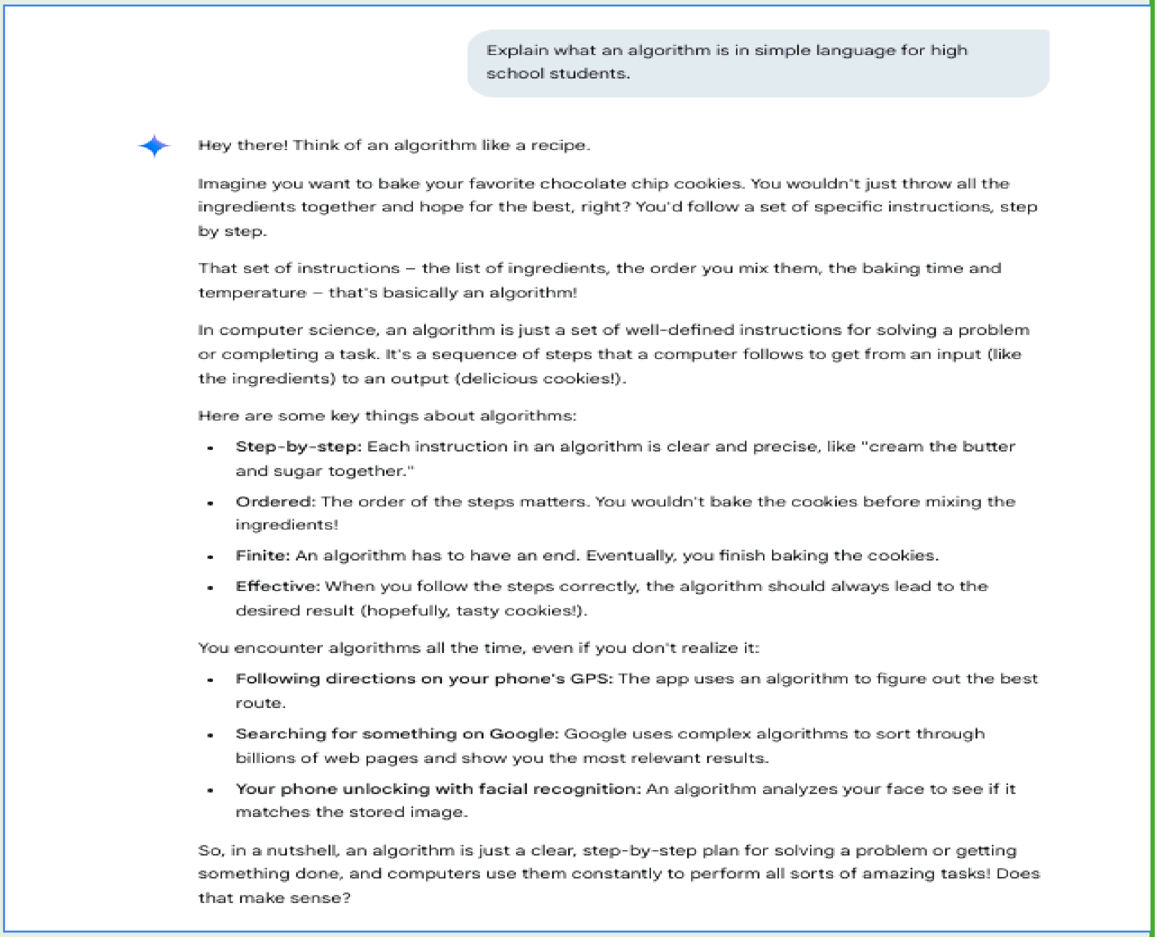
Generative AI can help you create learning materials like quizzes, summaries, step-by-step explanations, infographics, or study guides in just minutes.

Suggested mini activity:

- Enter a tool like Gemini or Copilot.
- Write a simple instruction like:

"Explain what an algorithm is in simple language for high school students."

- Look at the answer and evaluate whether you could use it in class as is or what adjustments you could make.



Explain what an algorithm is in simple language for high school students.

Hey there! Think of an algorithm like a recipe.

Imagine you want to bake your favorite chocolate chip cookies. You wouldn't just throw all the ingredients together and hope for the best, right? You'd follow a set of specific instructions, step by step.

That set of instructions – the list of ingredients, the order you mix them, the baking time and temperature – that's basically an algorithm!

In computer science, an algorithm is just a set of well-defined instructions for solving a problem or completing a task. It's a sequence of steps that a computer follows to get from an input (like the ingredients) to an output (delicious cookies!).

Here are some key things about algorithms:

- **Step-by-step:** Each instruction in an algorithm is clear and precise, like "cream the butter and sugar together."
- **Ordered:** The order of the steps matters. You wouldn't bake the cookies before mixing the ingredients!
- **Finite:** An algorithm has to have an end. Eventually, you finish baking the cookies.
- **Effective:** When you follow the steps correctly, the algorithm should always lead to the desired result (hopefully, tasty cookies!).

You encounter algorithms all the time, even if you don't realize it:

- **Following directions on your phone's GPS:** The app uses an algorithm to figure out the best route.
- **Searching for something on Google:** Google uses complex algorithms to sort through billions of web pages and show you the most relevant results.
- **Your phone unlocking with facial recognition:** An algorithm analyzes your face to see if it matches the stored image.

So, in a nutshell, an algorithm is just a clear, step-by-step plan for solving a problem or getting something done, and computers use them constantly to perform all sorts of amazing tasks! Does that make sense?

2.

3. Creating custom materials

You can use AI to adapt content based on your students' level, learning style, or specific needs. This supports more inclusive and personalized education.

Suggested mini activity:

- Now ask the AI:
  "Summarize the text you generated in three difficulty levels: basic, intermediate, and advanced."
- Review the results and decide if they are really suitable for different learning levels.

4. Automation of repetitive tasks

Generative AI can help you save time on administrative or repetitive tasks, such as writing emails, creating rubrics, or even creating lesson plans.

Suggested mini activity:

- In the same AI that you decided to test in the previous activities, write and send the following instruction:
  "Create a plan to teach an introductory class on Artificial Intelligence for high school students.".
- Adjust the proposed criteria to your needs and save the document, as it can help you in your role as a mentor.

Generative AI Tools

ChatGPT (OpenAI)

ChatGPT is a language model developed by **OpenAI** that simulates human conversations and can be a useful tool in educational contexts, such as Technovation Girls. This program allows girls to develop applications to solve real-life problems, and ChatGPT can support them in tasks such as devising solutions, understanding technical concepts, writing content, or developing the code they need to create their mobile or web application.

Recent studies indicate that generative artificial intelligence can improve learning by personalizing information and providing immediate feedback. However, its use must be accompanied by critical thinking, fact-checking, and ongoing ethical reflection, especially regarding gender bias, cultural representation, and limited understanding of languages other than the most widely spoken. For example, although this technology understands Spanish, it does not recognize languages such as Garifuna, a language spoken by some Indigenous people in Latin America..

Therefore, we recommend:

- **Always verify the information you generate.**
- **Use AI as a starting point, not a definitive source.**
- **Encourage conversations with students about the ethical uses of technology and the importance of including diverse voices in its development.**

Student Use:

For students **13 years or younger**, access to **Gemini, ChatGPT** and other **Generative AI tools** must be done **exclusively** through the **account of an adult and always under his or her direct supervision**.

Suggested mini activity:

- Create a ChatGPT account (we recommend creating a dedicated email address for your Technovation Girls mentor role). Go to :https://chatgpt.com/auth/login and click on “**Subscribe**”.
- You can also use a non-registration version.**(although the model was trained with information from October 2023 and earlier):** https://chatgpt.com/
- **Choose one of the following general topics** and review the specific problems:
- **Mental health in young people**
    - Lack of access to emotional support.
    - Difficulty talking to adults about what they feel.
    - Excessive use of social media that affects self-esteem.
- **Environmental care**
    - Excess garbage in public spaces.
    - Lack of knowledge about recycling.
    - Low participation of young people in ecological activities.
- **Gender violence**
    - Street harassment.
    - Lack of education in healthy relationships.
    - Little access to reliable information on support and reporting.

- **Choose one of the problems above** and copy this prompt into ChatGPT (you can change the text in brackets):
  "I'm participating in Technovation Girls. Help me create an idea for an app that solves a problem in my community. I'm interested in the topic of [gender violence / mental health / environmental care] Specifically, I want to work in [for example: street harassment / litter in public spaces / difficulty talking about emotions] "What kind of app could I create to help solve this problem?"
- **Read and analyze the answer** that ChatGPTgives you .
- **Check the information** or concepts you don't know using sources like

Google, Google Scholar and Scielo.

Reflect:

- Do you find the suggested ideas useful?
- Would you change anything?
- How could you support students to use ChatGPT to better understand the problems they are researching?

Gemini (Google)

Gemini, formerly known as Google Bard, is a generative artificial intelligence model developed by Google. It has the capability to generate text, analyze images, answer questions, and assist with productivity tasks. Gemini is free to use, and an advanced version with additional features is available for users over 18.

Gemini can be seamlessly integrated with tools that many educators already use, such as **Google Docs, Slides or Gmail**, thanks to **Gemini for Google Workspace.** This integration facilitates everyday use in the classroom and allows you to generate content, review texts, create presentations, or respond to emails more efficiently.

Like other generative AI tools, it's important to encourage critical use of Gemini. While it offers useful and creative insights, its responses may contain errors, oversimplifications, or biases derived from the data it was trained on.

Suggested mini activity:

- Go to https://gemini.google.com and sign in with a Gmail account.
- Type the following instruction in the conversation box:

  "I'm participating as a mentor for Technovation Girls. Help me create a schedule of activities to help my students complete their projects. Please note that I will support them in their learning and project development process from October to April, and that they must submit their projects by April 20 2026."

- Review your answer and edit it if necessary. Remember to check that it correctly addresses the topics covered in the Technovation Girls curriculum by visiting the following page: https://technovationchallenge.org/technovation-curriculum/
- Use it as a basis for your activity schedule with your students!

Github Copilot (Github - OpenAI)

GitHub Copilot is an artificial intelligence-based programming assistant developed by GitHub and OpenAI. It is designed to help you write code faster and with less effort by generating code snippets or complete functions from natural language comments. It is integrated with editors such as **Visual Studio Code**, making it especially useful for developing applications with custom code.

Within the framework of programs such as **Technovation Girls,** Github Copilot can be a strategic tool for educators who:

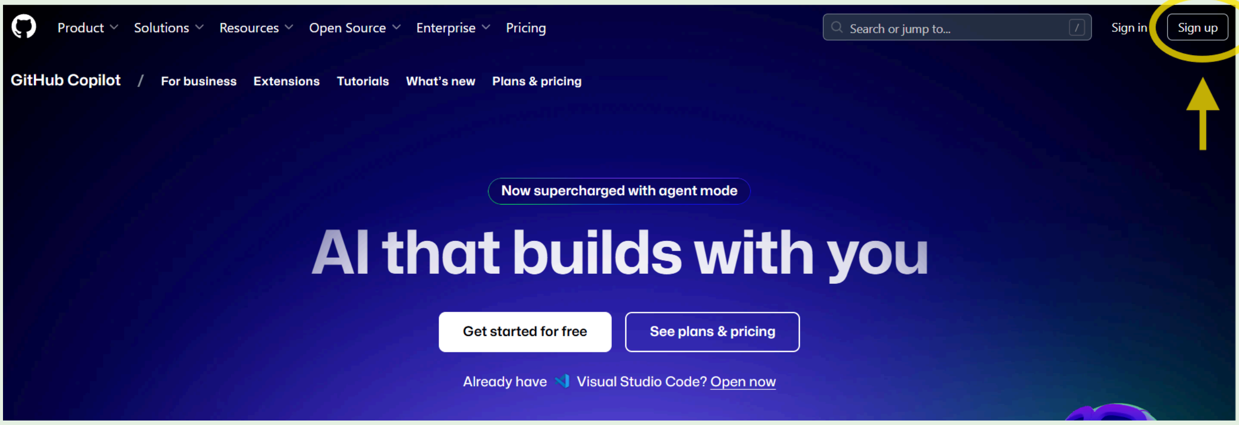
- assist students in writing or debugging code in HTML, CSS, JavaScript, Python, etc.
- want to become familiar with basic programming concepts without being technical experts.

- a tool that you can encourage students, especially older students, to use in their coding

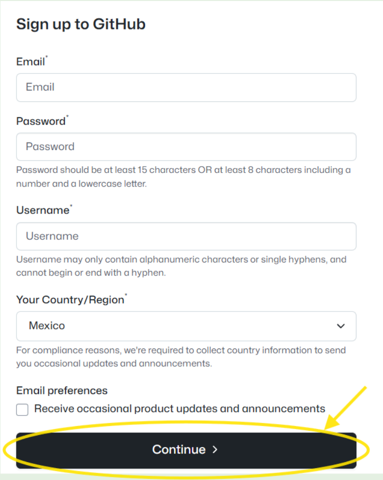
Additionally, Github Copilot can encourage you to **lose your fear of code**. You can develop greater autonomy and model active learning for your students.

Suggested mini activity:

1. Open https://github.com/features/copilot
2. Click on the button "**Sign up**" To create an account on GitHub, remember the recommendation to have a unique email for these activities.



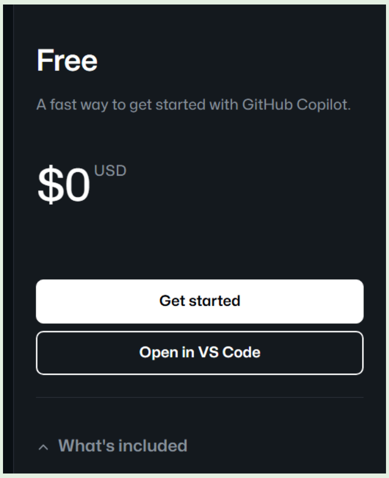
3. Provide the requested information to create an account. Once the requested information is completed, click on **Continue**.
4. Once you log in, choose the **Free Plan** and click on **Get started**.



Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

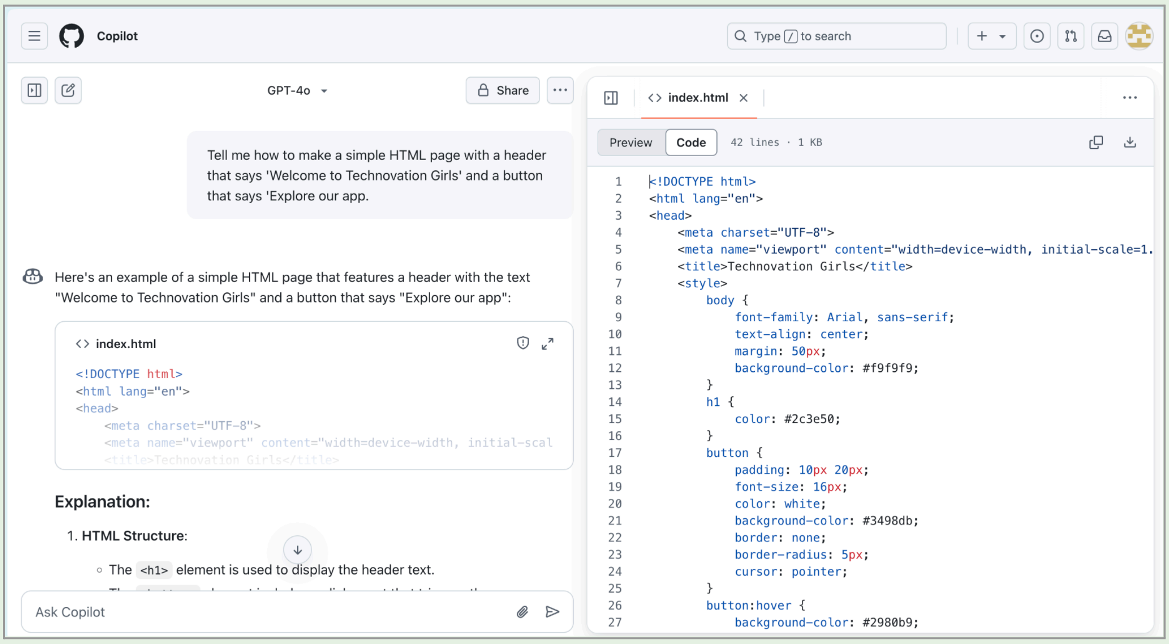
For compliance reasons, we're required to collect country information to send you occasional updates and announcements.



A fast way to get started with GitHub Copilot.

5. In the chat, type the following message:
   "Walk me through a simple HTML page with a header that says 'Welcome to Technovation Girls' and a button that says 'Explore our app.'"

6. Copilot will generate a code sample and explain each part to you:



Tell me how to make a simple HTML page with a header that says 'Welcome to Technovation Girls' and a button that says 'Explore our app.

Here's an example of a simple HTML page that features a header with the text "Welcome to Technovation Girls" and a button that says "Explore our app":

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scal
    <title>Technovation Girls</title>
```

Explanation:

1. **HTML Structure**:
   - The `<h1>` element is used to display the header text.

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>Technovation Girls</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            margin: 50px;
            background-color: #f9f9f9;
        }
        h1 {
            color: #2c3e50;
        }
        button {
            padding: 10px 20px;
            font-size: 16px;
            color: white;
            background-color: #3498db;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }
        button:hover {
            background-color: #2980b9;
```

7. It will likely ask you to save the file in Notepad (Windows) or Notes (Mac); it will also suggest using Visual Studio Code, which is a very useful piece of software, but we'll learn how to use it later. Remember to save the file as "index.html" so you can view it in a web browser.

Reflect: What do you understand about the generated code? What would you like to ask Github Copilot to better understand it?

Poe

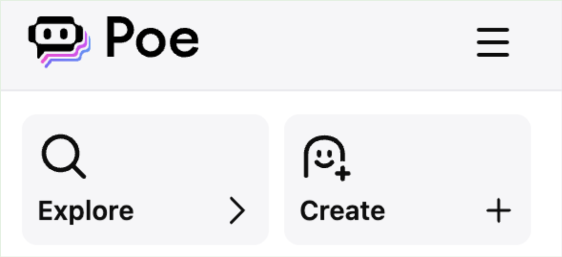
Poe is a platform that brings together the world's most advanced artificial intelligence models in a single interface. These include ChatGPT, Claude 3.5 Sonnet, and a wide variety of models specialized in generating text, images, audio, and video. It also offers access to millions of bots created by users around the world.

Basic access to Poe is free for most uses. For those who want advanced features, subscription plans are available starting at $4.99 USD per month.

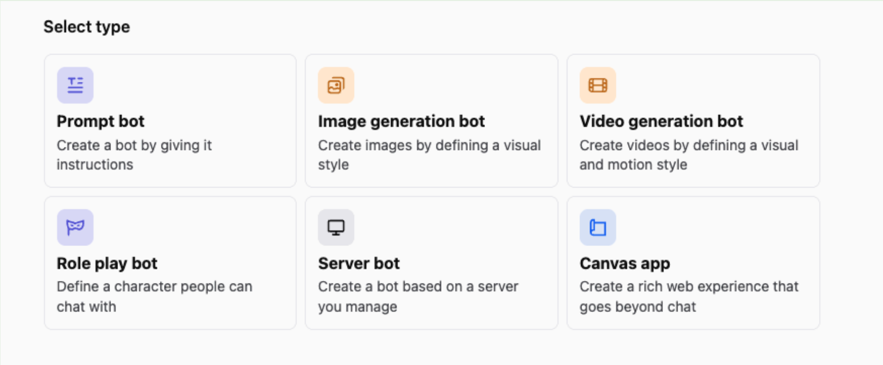
One of Poe's most powerful features is the ability to **create your own chatbot.** This tool is especially useful for student projects, as it allows you to **create a personalized artificial intelligence** quickly, easily, and without the need to program from scratch. Creating a bot in Poe can be the first step toward integrating generative AI into your apps or technology platforms within Technovation Girls.

Suggested mini activity:

1. Open https://poe.com/login
2. Create an account, you can sign up with your Gmail account.
3. Click on the Menu button and then on "**Create**".



4. Now click on the option "**Prompt Bot**".



Select type

Prompt bot
Create a bot by giving it instructions

Image generation bot
Create images by defining a visual style

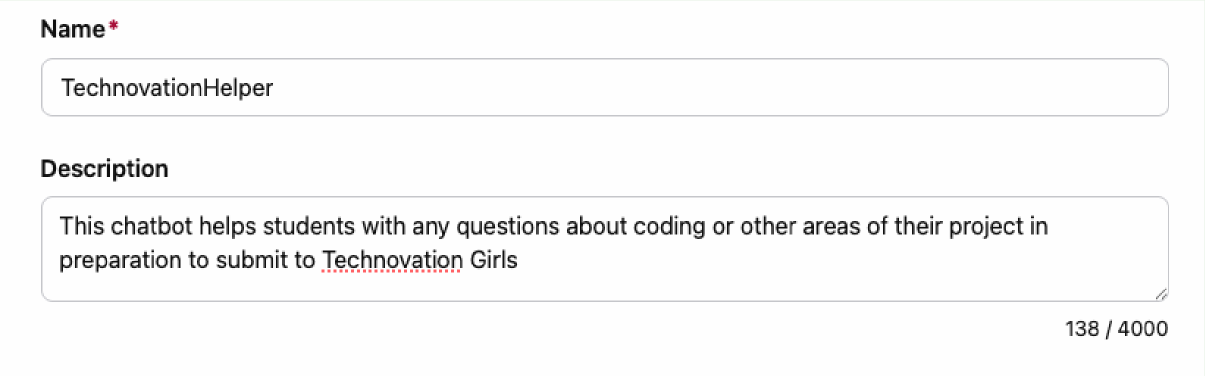
Video generation bot
Create videos by defining a visual and motion style

Role play bot
Define a character people can chat with

Server bot
Create a bot based on a server you manage

Canvas app
Create a rich web experience that goes beyond chat

5. Click on "**Edit Picture**" and add an image that identifies your chatbot. You can generate a custom image using an AI tool, for example, DALL-E.
6. Enter a name for your chatbot. You have 15 characters, and you can't repeat a name that's already been registered. I recommend that it refer to the task, for example: *Educate*.
7. In the description, write a short paragraph that will be visible to your target audience. For example: ***This chatbot answers any programming doubts and questions you may have as you prepare to participate in Technovation Girls.***



Name*

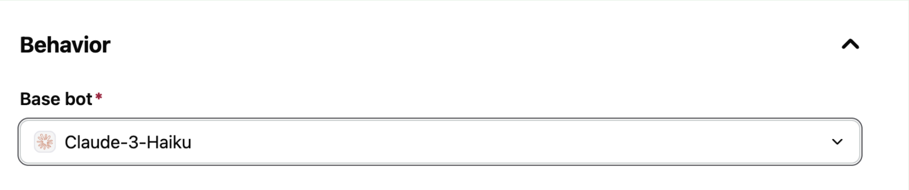
TechnovationHelper

Description

This chatbot helps students with any questions about coding or other areas of their project in preparation to submit to Technovation Girls

138 / 4000

8. In the Base Bot section, we choose the AI engine our chatbot will run on. There are several options, but we recommend choosing a ChatGPT model, such as GPT-4o, GPT-4o-mini, GPT-4.1, GPT-4.1-mini, GPT-4.1-nano, etc. In this case, we chose **GPT-4o-mini.** Another good option is **Claude-3.7-Sonnet**.



Behavior

Base bot*

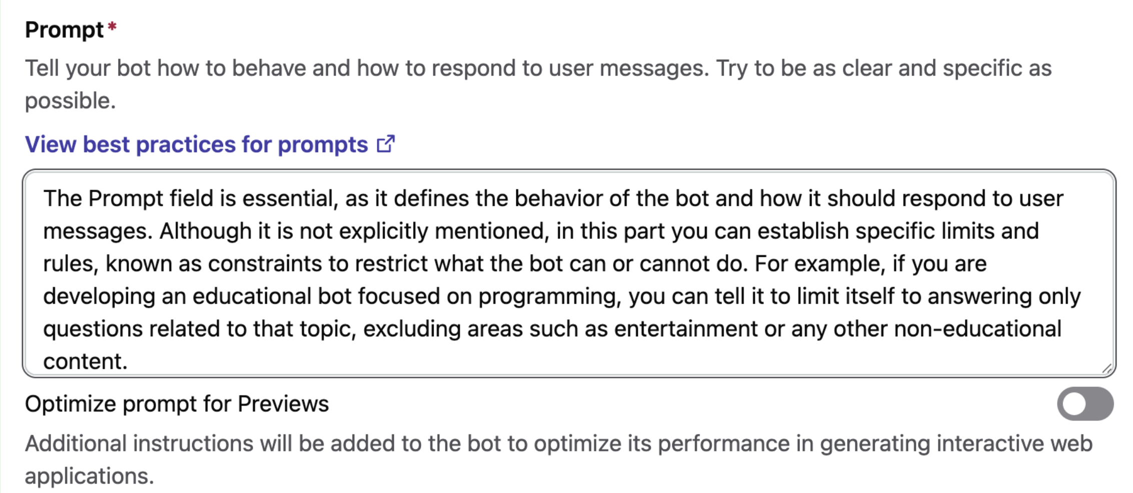
Claude-3-Haiku

9. The **Prompt** field is essential, as it defines the behavior of the bot and how it should respond to user messages. Although it is not explicitly mentioned, in this part you can establish **specific limits and rules**, known as **constraints** to restrict what the bot can or cannot do. For example, if you are developing an educational bot focused on programming, you can tell it to limit itself to

answering only questions related to that topic, excluding areas such as entertainment or any other non-educational content.

Below is an example instruction for setting up a bot for this purpose:

You are an educational bot specializing in programming. Your mission is to help girls and teens learn and understand programming concepts so they can successfully complete their Technovation Girls program projects. You should not provide entertainment recommendations or offer unverified information. Whenever you provide facts or explanations, please include the source. When advising on the development of their projects, please also refer them to the official submission guidelines: https://technovationchallenge.org/submission-guidelines/



Prompt *

Tell your bot how to behave and how to respond to user messages. Try to be as clear and specific as possible.

View best practices for prompts

The Prompt field is essential, as it defines the behavior of the bot and how it should respond to user messages. Although it is not explicitly mentioned, in this part you can establish specific limits and rules, known as constraints to restrict what the bot can or cannot do. For example, if you are developing an educational bot focused on programming, you can tell it to limit itself to answering only questions related to that topic, excluding areas such as entertainment or any other non-educational content.

Optimize prompt for Previews

Additional instructions will be added to the bot to optimize its performance in generating interactive web applications.

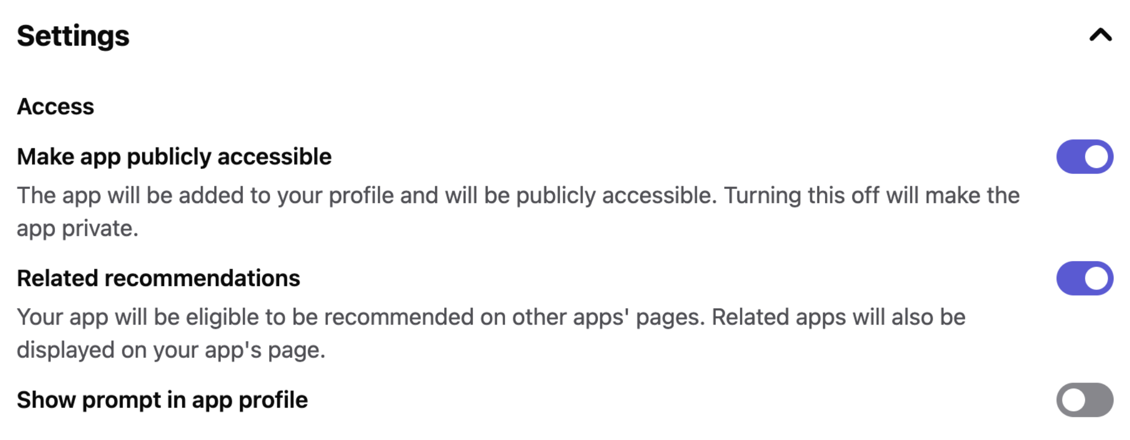
10. Poe allows you to add documents (in PDF format) on a topic to "train" the chatbot and make it specialized in that topic. In the case of this example, Technovation rubrics could be added so the bot can provide feedback based on them. **It is not mandatory to add sources of knowledge.**
11. The greeting message is also optional, but highly recommended. It's the first message users will read when interacting with the chatbot. It should be direct, focused on the context in which we want to have the conversation, and encourage interaction. Example: ***Hello! I'm TechnovationHelper, and I'll help you complete your project at Technovation Girls. Ask me anything you want about programming, and I'll give you the answer because I***

know almost everything. You can test me, for example, with these questions: What is an algorithm? Which programming language is best for creating a mobile app?

12. In the **Settings** field, we recommend that you keep the option activated **“Make the application publicly accessible”.** This will generate a unique web address (URL) that you can share with anyone interested in learning about your chatbot. Remember that this URL won't be visible to the general public unless you choose to publicize it.

    You can also activate the option **“Related Recommendations”**, so that the chatbot includes links to the sources used and suggests additional questions that help continue the conversation in a more enriching way.

    Finally, we suggest you **keep disabled** the option **“Show instructions in the app profile”.** This section contains the internal guidelines or "locks" we previously defined to guide the chatbot's behavior, and is not relevant or necessary information for the end user.



Settings

Access

Make app publicly accessible ●

The app will be added to your profile and will be publicly accessible. Turning this off will make the app private.

Related recommendations ●

Your app will be eligible to be recommended on other apps' pages. Related apps will also be displayed on your app's page.

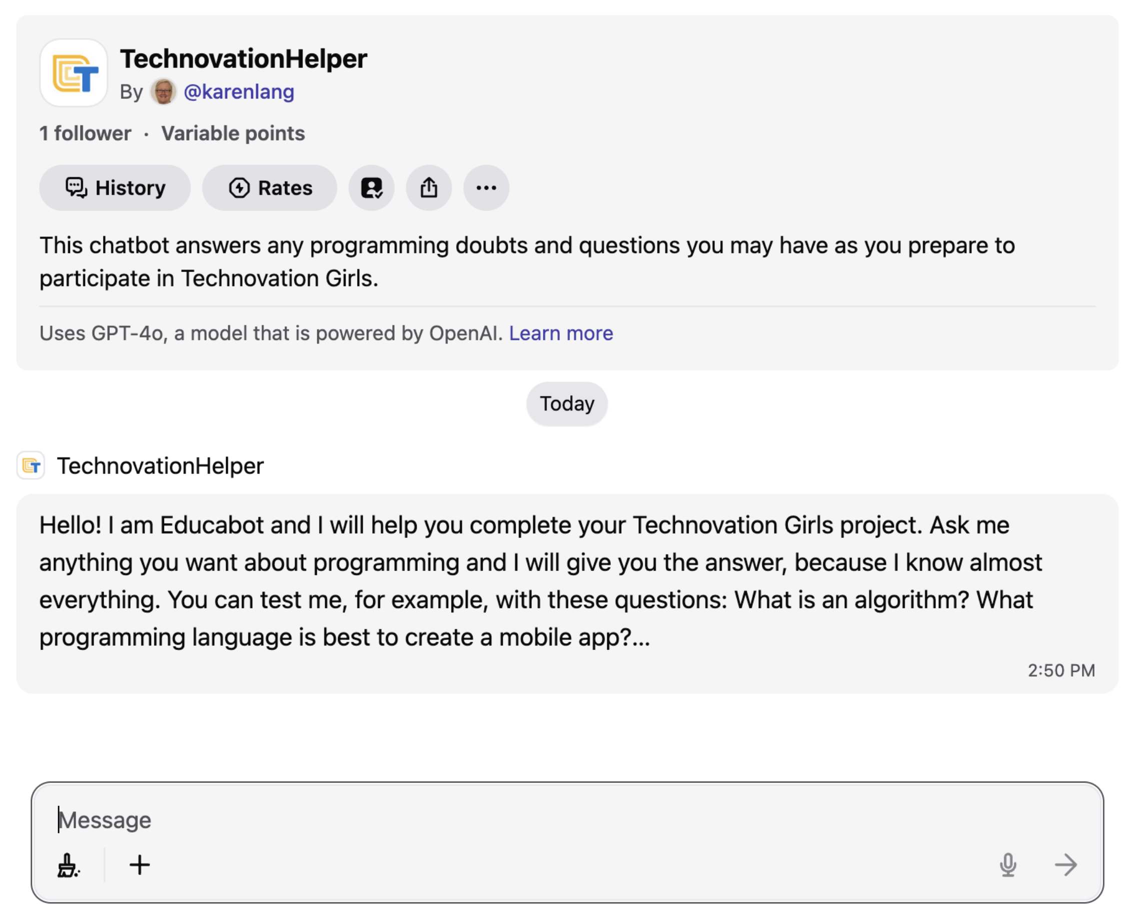
Show prompt in app profile ○

You can also activate the option **“Related Recommendations”**, so that the chatbot includes links to the sources used and suggests additional questions that help continue the conversation in a more enriching way.

Finally, we suggest you **keep disabled** the option **“Show instructions in the app profile”.** This section contains the internal guidelines or "locks" we

previously defined to guide the chatbot's behavior, and is not relevant or necessary information for the end user.

13. Finally click on **Publish** and you can test your chatbot.



This chatbot answers any programming doubts and questions you may have as you prepare to participate in Technovation Girls.

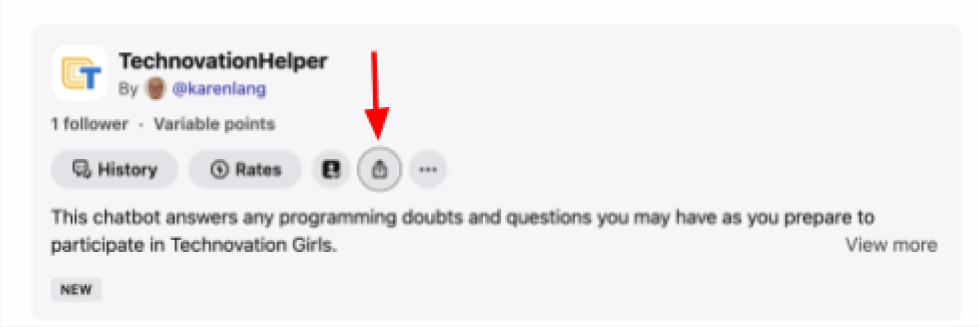
Uses GPT-4o, a model that is powered by OpenAI. Learn more

Hello! I am Educabot and I will help you complete your Technovation Girls project. Ask me anything you want about programming and I will give you the answer, because I know almost everything. You can test me, for example, with these questions: What is an algorithm? What programming language is best to create a mobile app?...

2:50 PM

How could your students apply this activity to their project? Remember the previous exercise with GitHub Copilot? Now let's use it to integrate the chatbot you created in Poe.

First, copy the **URL of your chatbot** from Poe's platform:

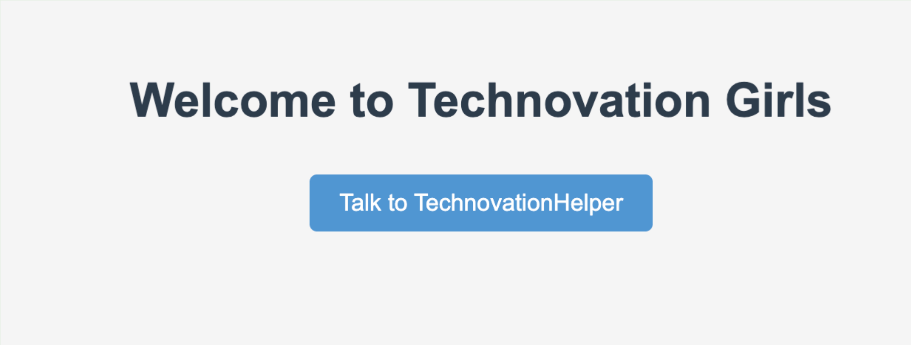

TechnovationHelper
By @karenlang
1 follower · Variable points

This chatbot answers any programming doubts and questions you may have as you prepare to participate in Technovation Girls.

Then, in a conversation with GitHub Copilot, you can write an instruction like this:

> ***"Create a chatbot in Poe. I want you to add a button to my page that says 'Talk to [bot name]' and, when clicked, it will redirect to this link: [chatbot URL]."***

Follow the instructions and view the chatbot on your HTML page.


Welcome to Technovation Girls

This type of integration can enrich the students' proposals, as it allows them to offer an interactive and real-life experience with artificial intelligence within their project.

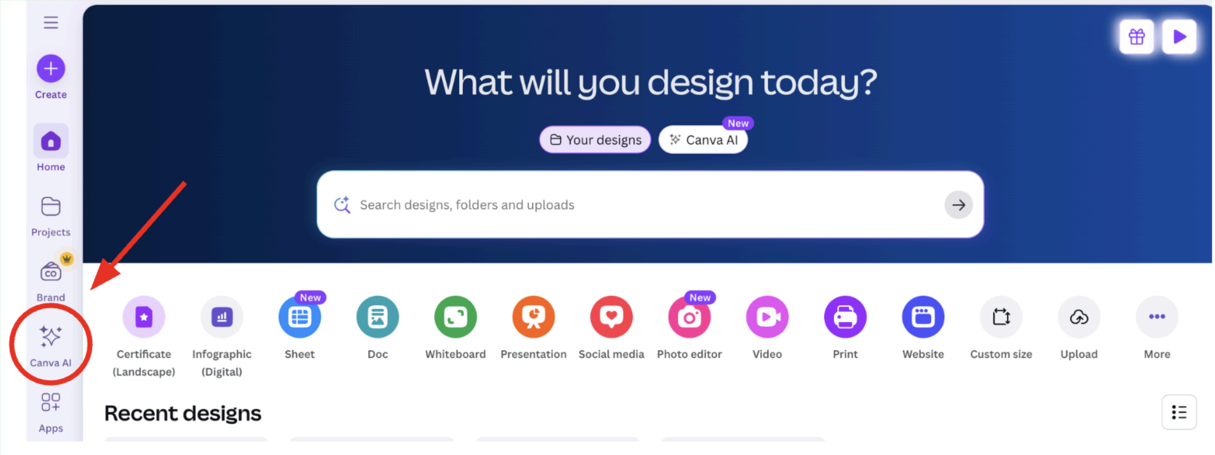
Canva AI

Canva AI is a set of artificial intelligence tools to help users create designs, generate content, and improve their projects. These tools are integrated into

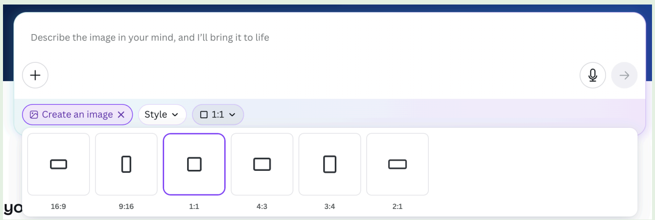
Canva's design platform and offer a variety of functions, from creating images from text descriptions to generating presentation slides, all with the help of AI. These tools can be useful as you support students in *Technovation Girls*, as they allow you to generate materials to guide, present or communicate your students' projects with visual impact.

Suggested mini activity:

1. Sign in to [Canva]
2. If you don't have an account yet, you can easily create one by clicking the button **"Register"**. Canva allows you to sign up with your **Gmail** account, **Facebook** or with any **email**
3. Once you're logged in, click on the Canva AI icon.

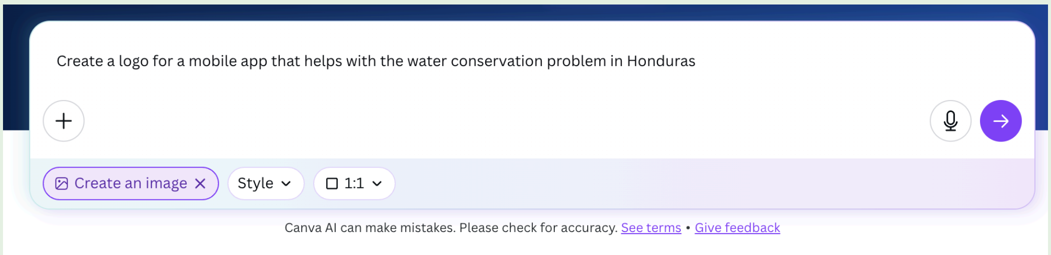


4. In the text field, select the "Create an image" option.
5. Select size 1:1



6. Type the following prompt in the text box:

   "Create a logo for a mobile app that helps with the water conservation problem in Honduras"

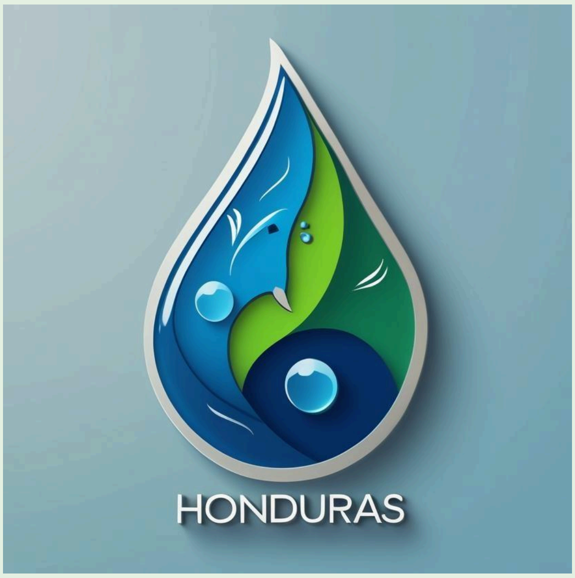


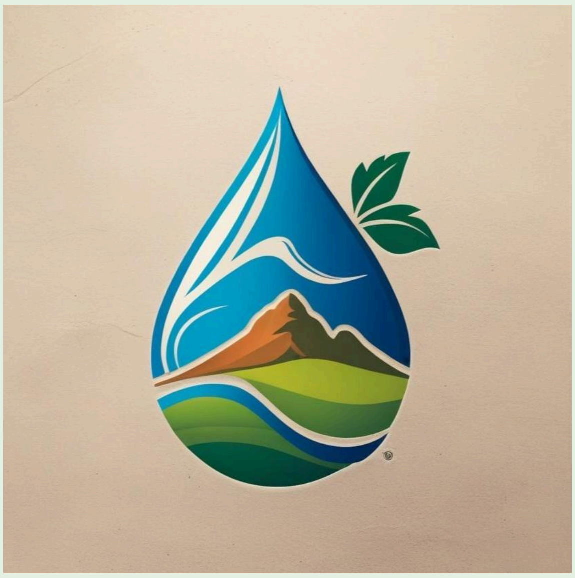
Create a logo for a mobile app that helps with the water conservation problem in Honduras

+
Create an image ×
Style
1:1
Canva AI can make mistakes. Please check for accuracy. See terms • Give feedback

(Remember that this is a suggestion and that, once you are working with the students on the development of their app, they can request a logo with a more specific prompt according to the characteristics of their project)

7. Select and download the best result that Canva AI provided you, these were some results of my request:





Why use this feature in Technovation Girls?

- To create **visual sketches** of your app design (screens, scenarios, logos).

- To strengthen the **visuals for the pitch** for your project (using eye-catching images that explain your idea).

Important: Is it paid? Is it worth using?

- Some images or styles may be blocked with a golden crown 🟡 (Pro), but:
    - **Most basic styles are free**.
    - **Each account, in its free version, has 20 credits to use Canva AI. Each request is equivalent to 1 credit, and each team can use them to create their logos, powerful images for their pitch, or prototypes for their app.**
    - **You don't have to pay to take advantage of it creatively.**

Best practices for writing prompts

What is a prompt?

A ***prompt*** is an instruction or question given to a language model, such as ChatGPT, to generate a response. It's how we communicate with AI so it understands what we want to accomplish and helps us with a specific task (Radford et al., 2018).

The quality of your *prompt* is key: a *prompt* that is clear, well-defined and contextualized can make the difference between a generic response and a truly useful solution. On the contrary, if the *prompt* is ambiguous or imprecise, AI could generate confusing or irrelevant answers (Sun et al., 2019).

As Technovation Girls mentors or other technologically focused educational projects, understanding how to write good *prompts* will allow you to:

- Better guide your students when working with AI tools.
- Get more useful results when generating educational content, design ideas, scripts, data analysis, or support materials.
- Teach your students the critical thinking behind good technology instruction.

Elements of a good prompt

  - **Context:** *What's the conversation about? What does the AI need to know?*
  - **Role:** *From what perspective should the AI respond? (For example: "You act as an education expert," "You are a programmer.")*
  - **Action:** *What do you want me to do? (analyze, create, propose, summarize, explain, etc.)*
  - **Format:** *In what form do you want the answer? (a list, a summary, an email, a table).*
  - **Target audience:** *Who is this response directed to? Is it for you as a teacher, or for your students?*

Example: *"You are an expert in STEM education for teenagers. Suggest three creative activities to teach the concept of artificial intelligence to high school students, including the learning objective and the necessary materials".*

Key recommendations when writing prompts

- **Define a clear objective**: Helps focus the response (McTear et al., 2016).
- **Be concise and specific**: Long or vague sentences confuse AI (Jurafsky & Martin, 2020).
- **Use natural language**: There is no need to sound technical, but it must be clear (Vossen et al., 2018).
- **Provide sufficient context**: Give background, avoid assuming that "you already know."
- **Avoid overly broad open questions**: Narrow down what you are looking for.
- **Take care of the tone and style**: Especially in educational contexts, use a respectful, approachable, and empathetic tone (Serban et al., 2017).
- **Use precise verbs**: The choice of verb guides the action that the AI will perform (Kumar et al., 2022).

For example, instead of *"Do this better"*, it is more useful to say: *"Reword this paragraph to make it clearer and more professional."*

Suggested mini activity:

1. Select one of these scenarios related to your role as a mentor:
    - Create an activity to explain artificial intelligence or programming to teenagers.
    - Write a guide for parents on how to support their daughters in Technovation Girls.
    - Propose ideas for an app that solves a local problem.
2. Write a *prompt* following this formula:

    "Act like ***[role]****. Help me to* ***[action]****, considering that the target is* ***[target audience]****. The format I need is* ***[format]****. The context is* ***[brief explanation of the purpose]****".*

    Example: *"Act as an expert in teen education. Help me create a fun activity to explain how algorithms work. The audience is a group of middle school girls. I want the answer in a step-by-step list format. The context is that I'm designing a workshop as a mentor for Technovation Girls."*

3. Type your *prompt* in ChatGPT or a similar tool and analyze the response. Did it provide what you requested? Was it clear? Was anything missing?
4. Edit your *prompt* to fine-tune it. You can:
    - Add more context.
    - Better specify the format.
5. Try it again in AI and compare the results.

Programming Basics for Teachers

Algorithm

The word "algorithm" derives from the name of the 9th-century Arab mathematician Al-Khuwarizmi, who described several methods for solving arithmetic problems. Algorithm design is a fundamental resource for solving problems in almost all disciplines.

Algorithm definition: An algorithm is an unambiguous, finite, and ordered sequence of steps designed to solve a problem. Below are the three essential components of an algorithm:

- **Unambiguous:** Each step of the algorithm must be able to be interpreted in a unique way, without any confusion.
- **Finite:** The number of steps that make up the algorithm is limited. The algorithm has a defined start and end.
- **Ordered:** The steps of the algorithm must follow a specific sequence to reach the solution to the problem.

An algorithm can be compared to a cooking recipe: it is a set of instructions or specifications about a process to achieve a goal..

Example: Instruct how to sweeten a cup of coffee.

- **Algorithm A:** Add a little sugar to the cup and stir.
- **Algorithm B:** Add a teaspoon of sugar to the cup, stir, and enjoy. Repeat the process until the coffee is sweet enough.

Algorithm A presents an ambiguous solution to the problem posed: How much is "a little"?

Algorithm B, on the other hand, presents a clear and precise solution to the problem.

In programming, algorithms help computers perform similar tasks by following clear and orderly instructions.

Suggested mini activity:

1. Think of a daily activity you do regularly. It could be:
    a. Prepare a sandwich
    b. Wash your hands
    c. Tie your shoelaces
    d. Prepare a cup of coffee or tea
    e. Turn on your computer and open your web browser.

2. Write the algorithm for performing that activity, step by step, as if you were explaining it to a robot or a child who has never done that task before. Remember to include:
    a. Clear **start and end**
    b. **Ordered** instructions
    c. **Specific** and **unambiguous** steps

For example, avoid phrases like “add a little water” or “wait a moment.” Be precise: “pour 200 ml of water” or “wait 1 minute.”

3. **Check your algorithm** using this checklist:
    a. Is it clear?
    b. Is it in logical order?
    c. Does it have a limited number of steps?
    d. Could anyone else replicate it without getting confused?

Optional: Ask someone close to you to read it and try to follow it exactly. See if they can complete the task without asking questions or improvising.

Suggested student activity:

You can replicate this same activity in class with your students by asking them to:

- Write your own algorithm for a simple activity.
- Exchange algorithms with another classmate and try to execute them as written.
- Then reflect on which instructions worked well and which caused confusion.

More suggested activities here.

Variable

In programming, **a variable is a storage space that allows you to save, modify, and retrieve data** during the execution of a program. It is a fundamental tool for solving problems, as it allows us to work with information that can change over time, such as a user's name, the score in a game, or the number of attempts at a task. In short, a **variable** stores information that can change. You can think of it as a "tag" that is given to a piece of data so that it can be used later.

A variable consists of two key elements:

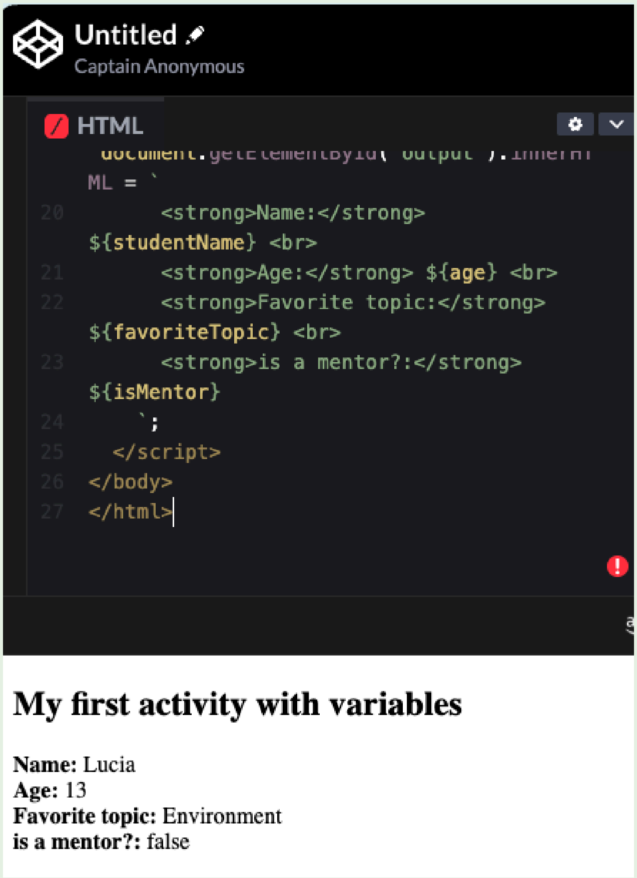
1. **Name or identifier:** the name we give to the variable to refer to it. For example, age, username, finalscore.
2. **Value:** the information stored in the variable. It can be of different types:
    - A number: **42**
    - A text (string of characters): **"Ana"**
    - A logical (boolean) value: **true** (True) or **false** (False)
    - A more complex object or data structure (depending on the programming language)

Suggested mini activity:

Programming language: JavaScript

1. Open https://codepen.io
2. Copy and paste this code into the HTML section:

```
<!DOCTYPE html>
<html>
<head>
 <title>Variables Activity</title>
</head>
<body>
 <h2>My first activity with variables</h2>
 <p id="output"></p>


 <script>
// declare variables
  let studentName ="Lucia";
  let age =13;
  let favoriteTopic = "Environment";
  let isMentor = false;

// show the values on the page
  document.getElementById("output").innerHTML = `
   <strong>Name:</strong> ${studentName} <br>
   <strong>Age:</strong> ${age} <br>
   <strong>Favorite topic:</strong> ${favoriteTopic} <br>
   <strong>is an adult?:</strong> ${isMentor}
  `;
 </script>
</body>
</html>
```



```
document.getElementById('output').innerHT
ML = `
          <strong>Name:</strong>
${studentName} <br>
          <strong>Age:</strong> ${age} <br>
          <strong>Favorite topic:</strong>
${favoriteTopic} <br>
          <strong>is a mentor?:</strong>
${isMentor}
      `;
   </script>
</body>
</html>
```

```
My first activity with variables

Name: Lucia
Age: 13
Favorite topic: Environment
is a mentor?: false
```

3. **Reflect:**
   a. What happens if you change the value of `age` or `StudentName`?
   b. What happens if you misspell a variable name?
   c. What is the difference between `true` and `false`?
4. Try changing the values of the variables to see what happens.

Suggested student activity:

1. Ask your students to modify the variables with their own data:
    - `student_name` = your name
    - `age` = your age
    - `favorite_subject` = one that you like
    - `isMentor` = True or False

2. Then, let them add a new variable: `city = "Your city"` and print it too.

Example:

```
<!DOCTYPE html>
<html>
<head>
 <title>Variables Activity</title>
</head>
<body>
 <h2>My first activity with variables</h2>
 <p id="output"></p>


 <script>
// declare variables
  let studentName ="Lucia";
  let age =13;
  let favoriteTopic = "Environment";
  let isMentor = false;
  let city = "New York City"

// show the values on the page
```

```
    document.getElementById("output").innerHTML = `
      <strong>Name:</strong> ${studentName} <br>
      <strong>Age:</strong> ${age} <br>
      <strong>Favorite topic:</strong> ${favoriteTopic} <br>
      <strong>is a mentor?:</strong> ${isMentor}<br>
      <strong>City:</strong> ${city}
    `;
  </script>
</body>
</html>
```

3. **Ask each team to identify one or two situations where variables are used in the mobile apps they typically use.**
   Example: Tiktok would need variables for username, followers, following, and much much more!

Conditionals

In programming, **conditionals** are structures that allow a program to **make decisions** based on specific conditions. That is, they allow the program to **choose between different paths** depending on whether something is true or false.

Conditionals follow a structure of the type:

```
IF (condition is true) THEN
  execute this action
ELSE
  perform a different action
```

For example:
If the weather is rainy, **then** carry an umbrella;
Else, go out without an umbrella.

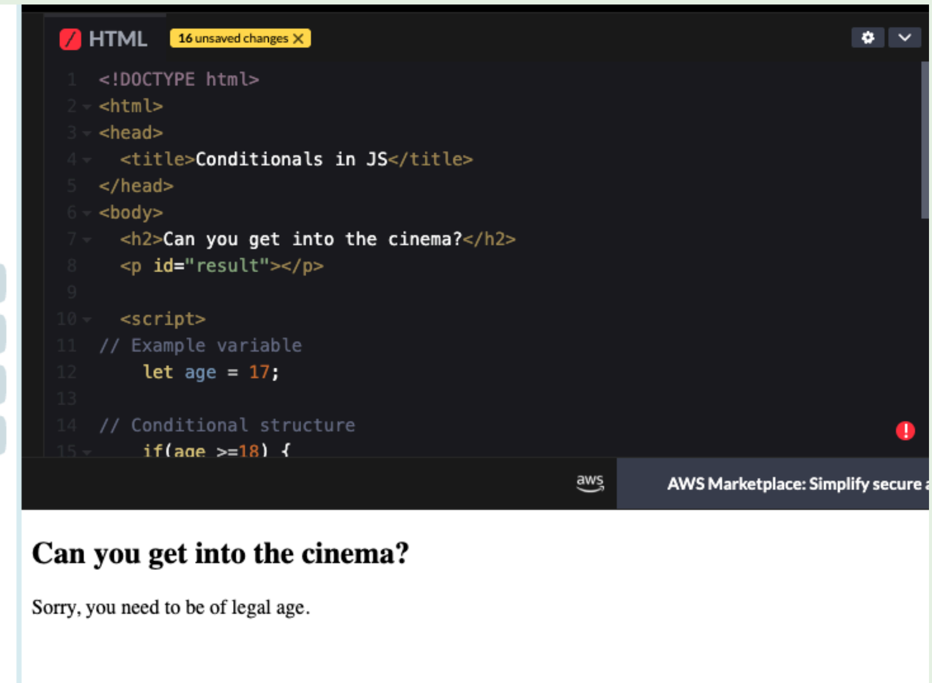
This type of structure is known as **if - else** and is one of the most common ways to use **conditionals**. It allows a program to make decisions based on whether **a condition is met or not**.

Suggested mini activity:

1. Open the site: https://codepen.io/
2. Copy and paste the following code into the HTML section:

```
<!DOCTYPE html>
<html>
<head>
  <title>Conditionals in JS</title>
</head>
<body>
  <h2>Can you get into the cinema?</h2>
  <p id="result"></p>

  <script>
// Example variable
    let age = 17;

// Conditional structure
    if(age >=18) {
      document.getElementById("result").innerText = "You can come in and see the movie!";
    } else {
      document.getElementById("result").innerText = "Sorry, you need to be of legal age.";
    }
  </script>
</body>
</html>
```

3. Look at the message that appears below the title.

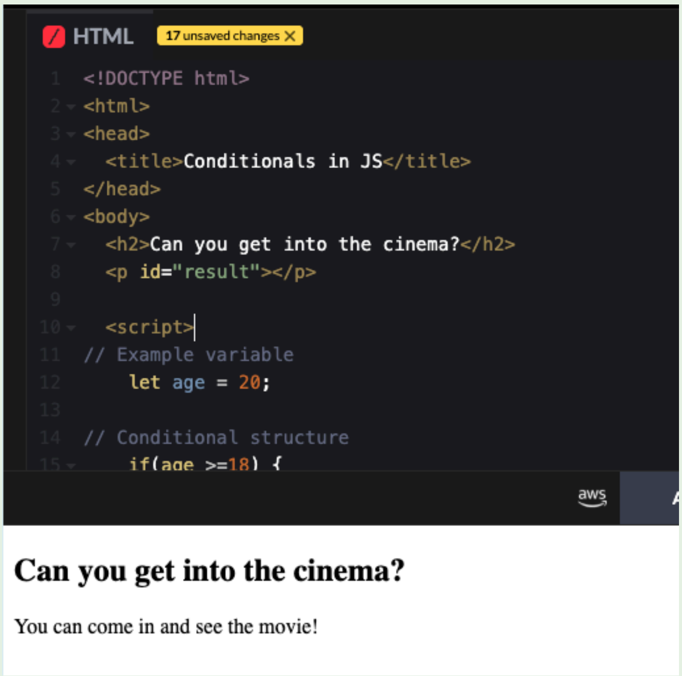


```
<!DOCTYPE html>
<html>
<head>
  <title>Conditionals in JS</title>
</head>
<body>
  <h2>Can you get into the cinema?</h2>
  <p id="result"></p>

  <script>
// Example variable
    let age = 17;

// Conditional structure
    if(age >=18) {
```

Can you get into the cinema?

Sorry, you need to be of legal age.

4. Change the value of the variable `age` to something over 18 and watch how the message changes.



```
<!DOCTYPE html>
<html>
<head>
  <title>Conditionals in JS</title>
</head>
<body>
  <h2>Can you get into the cinema?</h2>
  <p id="result"></p>

  <script>
// Example variable
    let age = 20;

// Conditional structure
    if(age >=18) {
```

Can you get into the cinema?

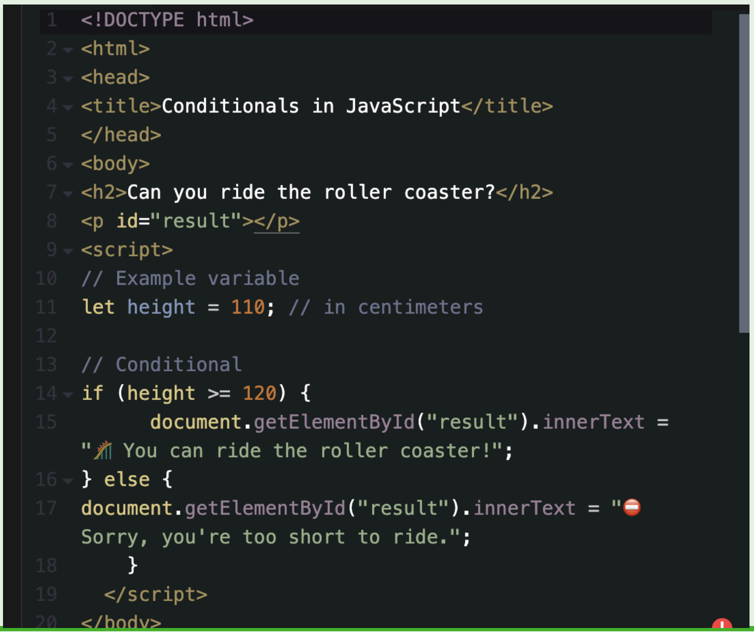
You can come in and see the movie!

Conditionals in action: Can you ride the roller coaster?

1. Let's use a similar example to the previous one, but we'll change the age variable to height.
2. Copy and paste the following code into the **HTML** section:

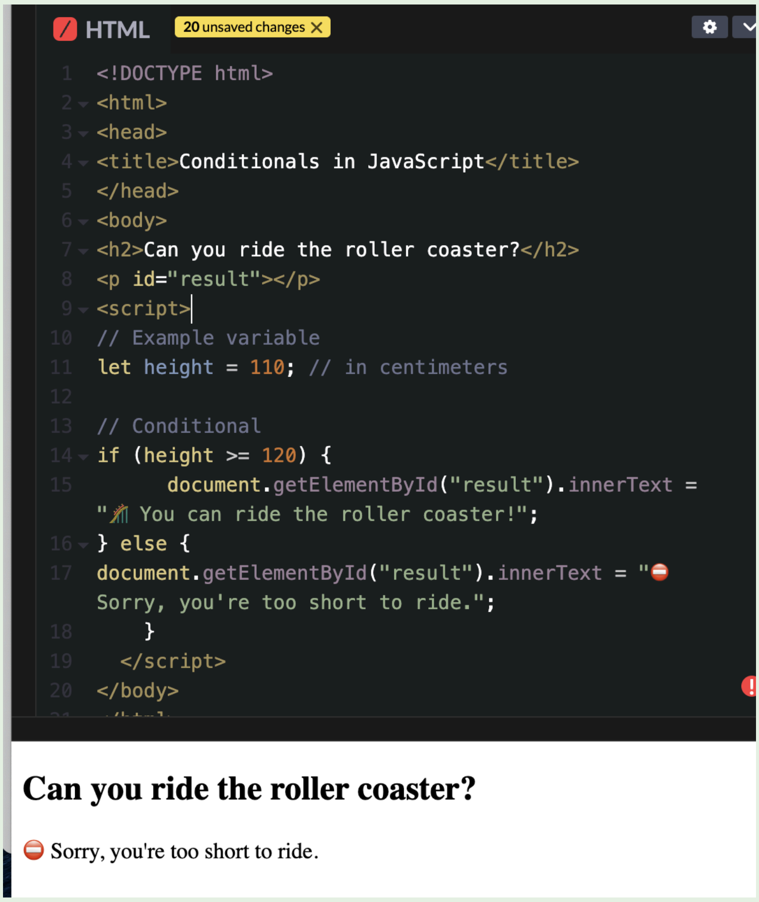
```
<!DOCTYPE html>
<html>
<head>
<title>Conditionals in JavaScript</title>
</head>
<body>
<h2>Can you ride the roller coaster?</h2>
  <p id="result"></p>

  <script>
// Example variable
let height = 140; // in centimeters

// Conditional
if (height >= 120) {
      document.getElementById("result").innerText = "🎢 You can ride the roller coaster!";
} else {
document.getElementById("result").innerText = "⛔ Sorry, you're too short to ride.";
    }
  </script>
</body>
</html>
```

3. Observe the result.



```
<!DOCTYPE html>
<html>
<head>
<title>Conditionals in JavaScript</title>
</head>
<body>
<h2>Can you ride the roller coaster?</h2>
<p id="result"></p>
<script>
// Example variable
let height = 110; // in centimeters

// Conditional
if (height >= 120) {
      document.getElementById("result").innerText =
"🎢 You can ride the roller coaster!";
} else {
document.getElementById("result").innerText = "⛔
Sorry, you're too short to ride.";
    }
  </script>
</body>
```

4. Change the value of the variable `height` to different numbers (for example, 110, 125, 160.) It should automatically run the code again. See how the message changes.



```
<!DOCTYPE html>
<html>
<head>
<title>Conditionals in JavaScript</title>
</head>
<body>
<h2>Can you ride the roller coaster?</h2>
<p id="result"></p>
<script>
// Example variable
let height = 110; // in centimeters

// Conditional
if (height >= 120) {
      document.getElementById("result").innerText =
"🎢 You can ride the roller coaster!";
} else {
document.getElementById("result").innerText = "⛔
Sorry, you're too short to ride.";
    }
  </script>
</body>
```

Can you ride the roller coaster?

⛔ Sorry, you're too short to ride.

Modify the example step by step:

This part will help you to **understand the logic** of the code.

1. **Identify**:
    - Where is the data being evaluated? (`let height = ...`)
    - What is the condition that decides which message to display? `if (height >= 120)`
    - What text is displayed in each case?

2. **Edit**:

Change these parts of the code to create another scenario. Some ideas:

  - Change the variable `height` to `age`, and decide whether the user can enter a concert.
  - Change the condition to `height >= 150`, and adapt the messages.
  - Changes the text displayed if the condition is met or not.

Use ChatGPT to help you:

Not sure how to change something? You can ask for help with a prompt like this:

> *"I'm practicing conditional structures in JavaScript. I have some code that tells me whether a person can ride a roller coaster based on their height. I want to change it to tell me whether they can enter a concert based on their age. Can you help me modify the code step by step?"*

Or if the code gives you an error or doesn't work:

> *"This code isn't displaying the message it should. Can you review it with me? Here's the code: [paste the code]."*

Reflect:

- What happens if you change the condition to `>` instead of `>=`?
- What is the difference if you write `height == 120`?
- Why do you think these conditions are used in real-life apps?

Suggested student activity:

Create your own situation with conditionals!

1. Divide the students into small teams.
2. Each team must present a different situation where a decision needs to be made based on a numerical value such as age, height, etc.
3. They will use ChatGPT to generate the code in JavaScript with `if - else.` Help them write the prompt for their request, using the mini-activity suggested above as a reference.
4. They will use **[CodePen](https://codepen.io)** to write your code, starting from this base example:

```
<!DOCTYPE html>
<html>
<head>
<title>Conditionals in action</title>
</head>
<body>
<h2>Can I use this social network?</h2>
<p id="message"></p>

<script>
let age = 12;

if (age >= 13) {
document.getElementById("message").innerText = "✅ You can create an account.";
} else {
document.getElementById("message").innerText = "🚫 Sorry, you must be at least 13 years old.";
}
</script>
</body>
</html>
```

4. Each team must then modify the example to adapt it to their invented situation.
5. Teams may use ChatGPT if they need help with:
    - writing the code from scratch
    - correcting errors
    - changing messages
    - understanding what each line means

6. Each team will share their final version with the group and explain what decision their conditional represents.

Closing/reflection:

- What kind of apps do you think these decisions are used in?
- Was it easy or difficult for you to make the modifications?
- What did you learn about how code works?

Loops

In programming, a **Loop** is a structure that allows a block of instructions to be executed **multiple times**, automatically, until a certain condition is met. You can use a loop to display messages, count, check if a condition is met multiple times, etc.

Common types of loops

- `while`: when **you don't know how many times** it will be repeated.
- `for`: when **you know how many times** it will be repeated.
- `do-while`: is executed **at least once and then continues if the condition is met**(not available in all programming languages).

Suggested mini activity:

Part 1: Loop `for`

1. Open the site: https://codepen.io/
2. Copy and paste the following code into the HTML section:

```
<!DOCTYPE html>
<html>
<head>
<title>Loops Activity</title>
</head>
<body>
<h2>My first activity with loops</h2>
  <p id="output"></p>

  <script>
    // We use a for loop to repeat an action
let message = "";
for (let counter = 1; counter <= 5; counter++) {
message += `Hello, I'm number ${counter} <br>`;
    }

document.getElementById("output").innerHTML = message;
  </script>
</body>
</html>
```

3. Run it and observe the result.

Try the following:

- Change the initial counter value to `0`. Run it and see what happens.
- Change the condition to counter `<= 10`. Run it and see what happens.
- Use counter `--` instead of `++`. Run it and see what happens.

4. **Reflect:**
    - What happened when the initial value of `counter` was less than the condition value?
    - What happened when you used `counter--` ?
    - What did you have to change to get something to display?

Part 2: `while` Loop

Now we are going to do the same, but using another type of loop: `while`.

1. Replace the previous `<script>` block with this:

```
<script>
let message = "";
let counter = 1;

while (counter <= 5) {
message += `Hello from while loop number ${counter} <br>`;
counter++;
}

document.getElementById("output").innerHTML = message;
</script>
```

2. Run it again and see what comes up.

Try these modifications:

- Change `counter = 1` to `counter = 3.` Run it and see what happens.
- Change the condition `counter <= 5` to `counter < 3.` Run it and see what happens.
- Change `counter = 3` back to `counter = 1.` What happens if you remove `counter++`? (Caution! This may cause an infinite loop, your browser may freeze)

Reflect:

- What differences and similarities are there between `for` and `while`?
- In what cases would you use each one?

Extra tip! If you don't understand why a change isn't working, use ChatGPT! You can type:

> *"I'm using a while loop in JavaScript, but it's not displaying anything. Here's my code: [paste your code]. What could be wrong?"*

Suggested Student Activity:

Part 1: `for` Loop

1. Ask the students to copy and edit the following code in the HTML section of https://codepen.io/:

```
<!DOCTYPE html>
<head>
<title>Loops in action</title>
</head>
<body>
<h2>Activity with for loop</h2>
  <p id="output"></p>

  <script>
let message = "";
    for (let i = 1; i <= 3; i++) {
message += `Clap number ${i}! 👏<br>`;
    }

document.getElementById("output").innerHTML = message;
  </script>
</body>
</html>
```

2. Now, make small changes to the code to see how the results change. You can try:

- Change the final number (for example, from 3 to 5).
- Change the initial number (for example, from 1 to 0).
- Change the message (for example: `"Hello ${i} times!"` 🎉).

Reflect with the students:

- What changed in the result?
- What happens if the starting number is greater than the ending number?
- What happens if I use `i--` instead of `i++`?

Part 2: `while` Loop

Now let's do the same thing... but with a different loop!

1. Replace just the `<script>` block with this new block of code:

```
<script>
let message = "";
 let i = 1;
 while (i <= 3) {
message += `Hello from while loop number ${i}! 👋<br>`;
   i++;
 }

document.getElementById("output").innerHTML = message;
</script>
```

Now, make the following suggested changes with the students:

- Change the initial value of `i`.
- Change the condition `i <= 3` with another number.
- Change the increment `i++` to `i += 2` to see what happens.

Reflect:

- What similarities and differences did you notice between the `for` loop and `while`?
- What would happen if you forget to increase `i?` Why might that be a problem?

Part 3: Time to create!

Now invite your students to create their own loop using what they have learned.

Some ideas:

- Repeating phrases like “You can do it!” 💪
- Count stars: “Star ⭐ number 1”, “Star ⭐ number 2”, etc.
- Make a countdown timer.

You can also use ChatGPT to ask for help, for example:

> *“I want a loop that counts down from 10 to 1. Can you give me the code and explain how it works?”*

Closing and reflection

Lead a small talk with questions like:

- What did we learn about loops?
- Where do you think loops are used in a real app?
- What was the most fun or challenging part of this activity?

Creating a Web Application with Generative AI

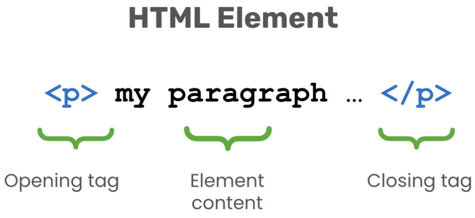
HTML

To display a web page, a computer needs more than just knowing what words or images should appear on the screen. It must understand how to organize content, determine its size, select appropriate fonts and colors, and handle many other details that transform text and images into a well-designed web page.

To address this challenge, computer scientists have developed languages that convey all this additional information. One of the most important is HTML, which stands for *Hypertext Markup Language*. **HTML allows programmers to use a tag system to define the purpose of each part of a web page**. These tags provide the computer with the instructions necessary to correctly display the different elements on the screen.

HTML tags

HTML indicates the purpose of different parts of a web page by surrounding them with pairs of opening and closing tags, as in the following example:



```
<p> my paragraph … </p>
```

Most common tags in HTML

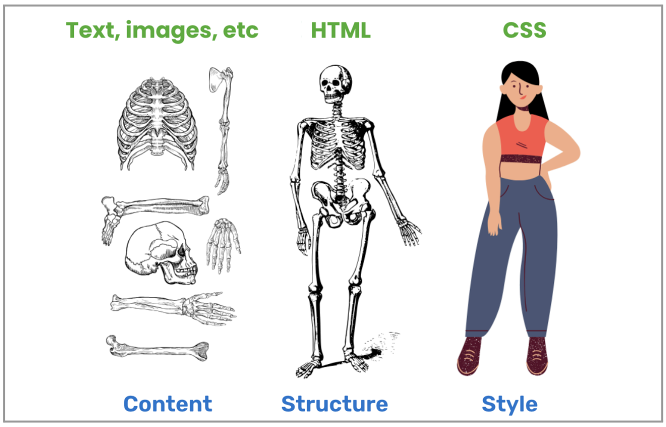
HTML tag	**What are you doing?**
`<!DOCTYPE html>`	Tells the computer that it is a document written in HTML.
`<html>`	Indicates the beginning of the code written in HTML.
`<head>`	Contains information (sometimes called "metadata") about a web page.
`<body>`	Contains all the main content of a web page.
`<h1>, <h2>, <h3>, <h4>, <h5>, <h6>`	These are page headings, ranging from H1 to H6. These tags allow you to prioritize and rank information, making it easier for search engines. `<h1>` specifies the most important heading, while `<h6>`specifies the least important heading.
`<p>`	Beginning of a paragraph. Paragraphs are simply blocks of text on your web page.
`</p>`	Marks the end of a paragraph. An end tag is always just the opening tag with a slash (/) at the beginning.

There are many great resources for learning about HTML tags online, but one of the most used is w3schools.com This website includes a lot of useful information on how all the tags work.

CSS

In web development, intentionally separating content and style is critical to creating code that is easier to maintain and update. The content of a web page is organized using HTML, which defines the page's structure. **CSS (*Cascading Style Sheets*) is the language used to style and give appearance** to that content.

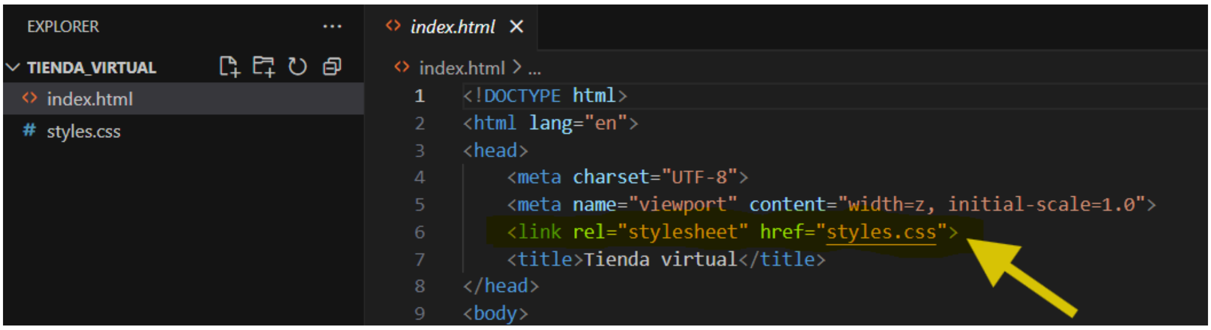
When developing a web page, it's crucial to understand the difference between content, structure, and style. Developers use HTML to structure different types of content, such as headings, lists, tables, and more. This ensures that each piece of content has a clear role on the page. However, without CSS, the browser would apply default styles to this content, based solely on the HTML structure.



Using CSS style rules, developers gain much greater control over how these elements are presented, allowing for complete customization of page layout.

Adding a style sheet

To apply custom styles to an HTML page, you need to create a separate style sheet (CSS) file. This file is linked to the HTML document within the <head> tag as follows:



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=z, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Tienda virtual</title>
</head>
<body>
```

Image taken from my own code to create a virtual store using Visual Studio Code

CSS consists of two main components: the **selector** and the **rules**.

- **Selectors:** Selectors identify the parts of the web page to which a specific style will be applied. A common way to select elements is by the HTML element type name, such as **h1**, **p**, and **body**. When you select an element type, all elements of that type on the page will have the style defined by the corresponding rules.

- **Rules:** Rules determine how the selected elements will change. Each rule includes a **property** (such as color or font-size) and a value associated with that property. For example, the following set of rules would apply to all **h1** elements a blue text color and underline:

```
h1 {

color: blue;

text-decoration: underline;

}
```

It's important to pay attention to punctuation in CSS rule sets, as this is how the computer understands where each rule begins and ends.

This modular approach to web development allows developers to maintain precise control over the design and functionality of their pages, making it easy to update and modify them over time.

JavaScript

JavaScript, or **JS**, is a scripting language. This means that your code **runs in real time**, line by line, while the program is running. It's a similar idea to a translator translating a book. The translator interprets each line in the book as she reads. Every time you open a javascript web page, it's as if the entire book is re-translated.

JavaScript is usually combined with **HTML** (page structure) and **CSS** (visual styles) to create **interactive websites**.

And how does it compare to a mobile app?

A **native mobile app**, such as those made in languages like Java (for Android) or Swift (for iOS), works differently. Your code must **compile** before use: that is, **it is fully translated into a language that the machine understands before being executed**.

To continue the translator analogy:

- **JavaScript**: Reading a French book with a translator sitting next to you. The translator reads each sentence and speaks the translated words in English for you. You don't have to wait for the translated version of the book, but the actual reading is slower.
- **Compiled language (as in mobile apps)**: Before you can read the book, it must be translated from French to English by a professional translator. It takes more time up front, but then you have a fully translated book.

Both approaches have their advantages. JavaScript is ideal for the web because it allows for quick changes and responsiveness to site visitors' actions. Mobile apps, being precompiled, are often faster and more efficient for more complex tasks or those that require access to the device's hardware.

As you probably noticed if you did the Github Copilot activity suggested above, HTML and CSS create static websites. These websites may present information but remain unchanged. JavaScript adds **interactivity** and the ability for the website to change and update based on external factors. A dynamic, interactive website is essentially a web application.

Python

Python is a programming language created by Guido van Rossum in the early 1990s, inspired by the English comedy group "Monty Python." It's a similar language with a very clean syntax that promotes readable code, making it a simple, versatile, and accessible language for programmers.

It's used in many different aspects of software development, such as web development and machine learning. Therefore, Python is a great choice for learning and developing more advanced AI web applications.

Creating a web app: “My goals for the month”

As part of your training as a mentor in **Technovation Girls**, you will learn how to create two versions of the same web app:

- One with **HTML, CSS and JavaScript**
- Another with **Python and Streamlit**

This will allow you to have different technological tools that you can share with your students, depending on their interests or level of knowledge.

Option 1: Web app with HTML, CSS and JavaScript

In this section, you will develop a simple web application that will allow users to **write down their goals for the month**, add them to a list and **mark as completed** those that have been achieved.

In addition to learning how to structure a basic app, you will become familiar with the use of **generative artificial intelligence development tools**, such as ChatGPT, to request and adapt code for educational purposes.

What do you need?

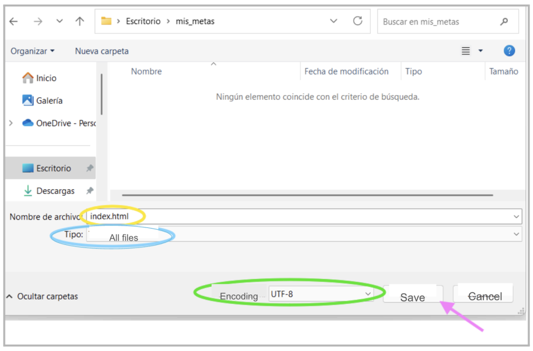
For this activity, we offer you two options to write and save your code:

1. **Notepad on Windows** or **TextEdit on Mac** (no additional installation required).
2. **Visual Studio Code (VS Code)**: a free tool recommended for more advanced projects. It is an **IDE (Integrated Development Environment)** which provides a complete environment for writing, organizing, and testing your code.

Step 1: Prepare your workspace

1. Create a folder on your Desktop named: my-goals.
   - Right click on the Desktop → select New → Folder → name it **my-goals.**

2. Open Notepad or TextEdit
3. Click "Save As" and assign the name **index.html.** Make sure you are in the my-goals folder.
4. Under "Save as type," select "All Files."**Very important!**
5. Use UTF-8 encoding (it should be the default) and click “Save”.
6. For Mac, be sure to uncheck the option If no extension is provided, use “txt”.



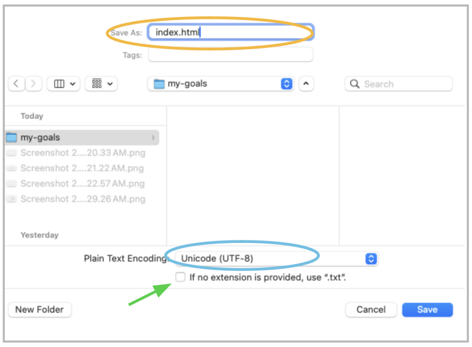
Ningún elemento coincide con el criterio de búsqueda.

Nombre de archivo: index.html

Tipo: All files

Encoding: UTF-8

Windows



my-goals
Search
Today
my-goals
Screenshot 2....20.33 AM.png
Screenshot 2....21.22 AM.png
Screenshot 2....22.57 AM.png
Screenshot 2....29.26 AM.png
Yesterday
New Folder
Cancel
Save

Save As: index.html

Tags:

Plain Text Encoding: Unicode (UTF-8)

☐ If no extension is provided, use ".txt".

Mac

7. Repeat this procedure to create two more files, but named:
    - **styles.css**
    - **script.js**

Step 2: Request the code from ChatGPT using an effective prompt

- Login to chat.openai.com with your account.
- Start a new conversation.
- Copy and paste the following **prompt**:

> *I'm a high school teacher and I want to create a simple web app in HTML, CSS, and JavaScript where users (my students) can enter their monthly goals. I want it to have a text field, a button to add the goal to a list, and the ability to check off completed goals. Generate the code in separate files:* `index.html`, `styles.css` *and* `script.js`, *and explain the programming step by step.*

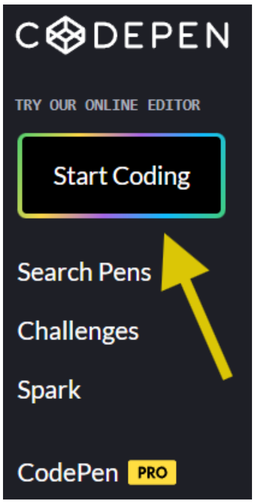
This **prompt works well** because it:

- Is clear and specific about what is needed.
- Specifies the programming languages that the AI should use.
- Provides educational context to adjust the level and focus of the code.
- Requests a step-by-step explanation, ideal for you and your students to understand how it works.

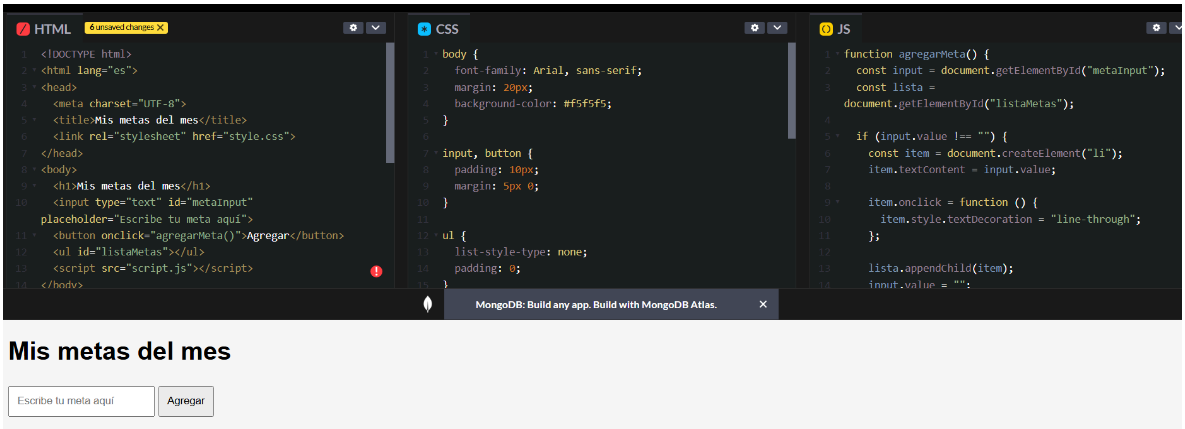
Step 3: Test the code without installation

You can see how your app works directly in the browser using **CodePen**, a free platform for programmers:

1. Open https://codepen.io.
2. Click the button **"Start Coding"** (or "Create Pen").
3. Paste the code given to you by ChatGPT into the corresponding fields:
    - ○ HTML code → in the "HTML" box
    - ○ CSS code → in the "CSS" box
    - ○ JavaScript code → in the "JS" box



Done! You can now interact with your app directly in the browser.



```html
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mis metas del mes</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Mis metas del mes</h1>
  <input type="text" id="metaInput"
placeholder="Escribe tu meta aquí">
  <button onclick="agregarMeta()">Agregar</button>
  <ul id="listaMetas"></ul>
  <script src="script.js"></script>
</body>
```

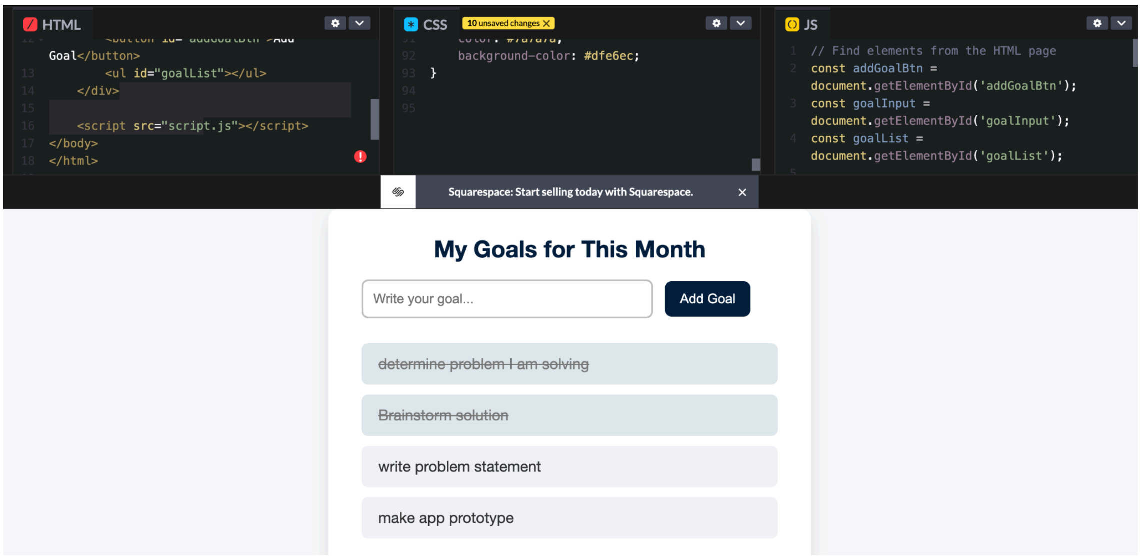
```css
body {
  font-family: Arial, sans-serif;
  margin: 20px;
  background-color: #f5f5f5;
}

input, button {
  padding: 10px;
  margin: 5px 0;
}

ul {
  list-style-type: none;
  padding: 0;
}
```

```javascript
function agregarMeta() {
  const input = document.getElementById("metaInput");
  const lista =
document.getElementById("listaMetas");

  if (input.value !== "") {
    const item = document.createElement("li");
    item.textContent = input.value;

    item.onclick = function () {
      item.style.textDecoration = "line-through";
    };

    lista.appendChild(item);
    input.value = "";
```

If the app is working correctly, but you'd like to improve its appearance, you can ask ChatGPT for a more modern version. For example:

> *The code works fine, however, I'd like you to modify the CSS code. I want the web app to have a modern style, suited to a professional website. I like the Navy Blue, White, and Silver combinations.*

The AI will update the file `styles.css` with a more attractive design that you can continue to adapt.



```
Goal</button>
        <ul id="goalList"></ul>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

```
    background-color: #dfe6ec;
}
```

```
// Find elements from the HTML page
const addGoalBtn =
document.getElementById('addGoalBtn');
const goalInput =
document.getElementById('goalInput');
const goalList =
document.getElementById('goalList');
```

My Goals for This Month

~~determine problem I am solving~~

~~Brainstorm solution~~

write problem statement

make app prototype

Step 4: Basic Debugging

What to do if the code doesn't work?

Debugging is the process of identifying and correcting errors in an application's source code. It is a key step in improving software quality, performance, and stability.

Even when we formulate an appropriate prompt, genAI may make mistakes or generate incomplete code. Don't worry! This is a natural part of the learning and development process.

Basic debugging techniques

Below are some strategies you can use when something doesn't work as expected:

- **Incremental testing**: Run the code every time you make a change. This makes it easier to detect errors.
- **Use the Browser Console**: When you open the developer tools in the browser (F12 or right-click → Inspect → Console), you will see error messages that will give you troubleshooting tips.

Debugging in CodePen

CodePen is a great platform for debugging, allowing you to see results immediately without having to install additional software.

1. If the app isn't responding as it should, observe what's happening and make a note of the problem.
2. Then, return to ChatGPT and type a clear prompt explaining the error. For example:

> *I want goals to be saved even when I close the app. Can you modify the code to use local storage (localStorage)?*

In this case, you will be using **incremental testing**, as you will apply the suggested change and check if it fixes the problem before moving forward.

Helpful tip: Sometimes ChatGPT may modify sections of code that were already working correctly. To avoid this, specifically tell it to **just correct the part with the error** and don't touch the rest of the code.

Example of how to make the request:

"My code displays a list of goals and allows me to add new ones, but they aren't saved when I close the app.

Please only modify the part that saves goals. Don't change the other functions.

Request corrections from genAI

Here are some useful phrases you can use to make adjustments or improvements to the generated code:

- **"The code works, but I want..."**
  Example: "The code works, but I want the goals to be removable if they are no longer needed."

- **"I don't understand this part of the code, can you explain it?"**
  Ideal for you and your students to better understand how it works.
- **"Can you break the code down into smaller parts and explain it step by step?"**
- **"I'd like the app to have a more teen-friendly design. Can you adapt the visual style?"**

Remember: AI responds best to clear and specific instructions. The more detailed your message, the more useful the response will be.

Example of debugging process

Imagine you've already copied the code generated by ChatGPT into CodePen, but when you test it, the following happens:

Problem: You type in a goal and click the “Add” button, but nothing appears in the list.

Step 1: Observe the error

- The text field and button appear correctly.
- You can type in the field, but when you click, nothing happens.
- No error message appears, but the goal is not added either.

Step 2: Review the JavaScript code

Since this is a feature that is not responding, the error is probably in the file `script.js`. Check this part of the code:

```
addButton.addEventListener('click', function() {
 const taskText = inputField.valu;
 if (taskText !== "") {
  const listItem = document.createElement('li');
  listItem.innerText = taskText;
  taskList.appendChild(listItem);
 }
});
```

Notice anything strange? Look closely:

```
 const taskText = inputField.valu;
```

The problem is in the word `valu`, which is **missing a letter. It should be value.** This is a very common typing error and the reason why what you type in the field isn't being read.

Step 3: Apply correction

Correct that line so it looks like this:

```
const taskText = inputField.value;
```

Now run the app again in CodePen. You should be able to type a goal and see it appear in the list. These types of errors are known as **syntax errors**, and are common in early stages of programming.

What if I can't find the error?

We know that debugging code can be challenging, especially if you're just starting out. But with observation, you can identify if the error is related to:

- **HTML**: App structure (for example, buttons or fields that are not displayed).
- **CSS**: visual design (e.g. colors or styles not applied).
- **JavaScript**: functionality (e.g. buttons that do nothing).

Once you've identified the type of error, you can ask the AI for help with a prompt like this:

> *The JavaScript code isn't working. When I click the button, the goal doesn't appear. Can you check for errors?*

Tip: If you don't know what language the error is in, simply describe the behavior you expected.

Apply the fixes suggested by ChatGPT, rerun the code, and repeat the observation and tweaking process until the application works as expected.

Code debugging activity:

Goal: Explore a simple piece of code, see what's not working, and ask ChatGPT for help finding and fixing the error.

Task: We want to make a mini app where the user types their name, clicks a button, and a message like this appears:
"Hello, Mariana!"

1. Initial code

Copy and paste into CodePen

HTML (in the HTML tab):

```
<h2>👋 Personalized greeting</h2>
<input type="text" id="name" placeholder="Enter your name">
<button id="HelloButton">Greet</button>
<p id="greetingMessage"></p>
```

JavaScript (in the JS tab):

```
const fieldName = document.getElementById("name");
const button = document.getElementById("HelloButton");
const message = document.getElementById("greetingMessage");

button.addEventListener("click", function() {
  const username = fieldName.innerText;
  message.textContent = "Hello, " + username + "! 👋";
});
```

2. What happens when you try it?

- You can type your name ✅
- The button works ✅
- But... **The greeting does not appear** or shows something wrong

3. Guided activity

Step A. Observe

- Can you write in the field?
- Is any message displayed?
- Is this the correct name?

Step B. Hypothesis

Which of these hypotheses do you think could be the cause?

- You are not reading the name field correctly
- The message is not updating
- The button is not connected

Step C. Ask ChatGPT

Type this prompt (or a similar one):

I'm trying this code, but it's not working as I expect.
When I click the button, it should display the message "Hello, [name]!", but it doesn't. What's wrong?
Paste the JS code, because it is a functionality issue.

4. What is the response from genAI?

ChatGPT should tell you something like this:

The error is with innerText
This attribute is used to read text that is already written *within* the element (such as a paragraph), but **not to read what the user types in a text field**.

In fieldName, `.value` should be used.

5. Apply the correction

Change this line:

```
const username = fieldName.innerText;
```

To this:

```
const username = fieldName.value;
```

And that's it! Now when you click the button, the greeting should appear correctly.

Final reflection

- What did you learn about code debugging?
- What role did ChatGPT play in this process?
- Did you notice how writing a specific question helps resolve the problem?

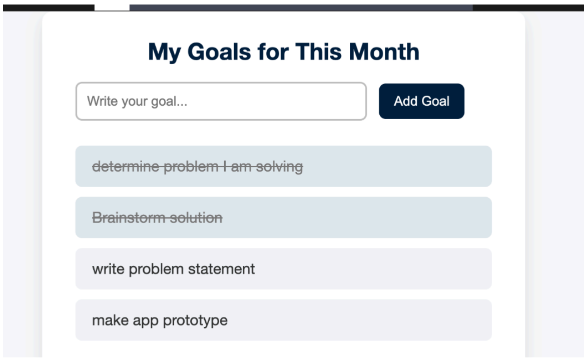
Step 5. Publishing the my-goals web application

Go back to your **my-goals** project. Once the app code is ready, copy and paste it into the corresponding files inside the “my-goals” folder you created at the beginning:

index.html → **HTML code**
styles.css → **CSS Code**
script.js → **JavaScript Code**

IMPORTANT: Make sure the file names are the same as shown in this manual, otherwise you may have difficulty creating the web or mobile app.

Finally, double-click on the file `index.html`. This will open it in your browser, and you'll be able to see your web app running locally.



My Goals for This Month

~~determine problem I am solving~~

~~Brainstorm solution~~

write problem statement

make app prototype

Is my application online yet?

Although the file **index.html** can be opened in any browser, this doesn't mean the app is published online. The link you see when you open it corresponds to the file's location on your computer, not an address accessible to others.

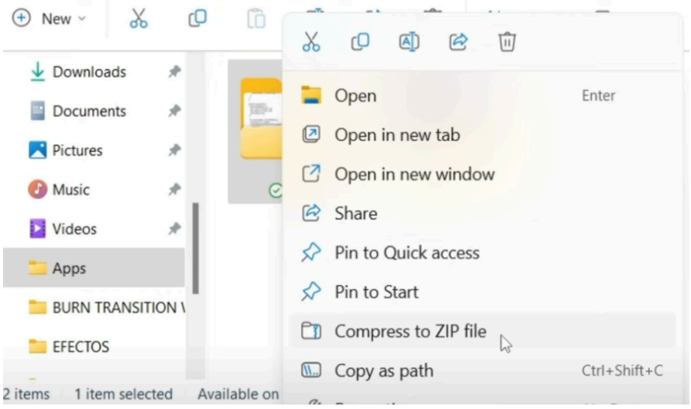
In order for your application to be accessible from anywhere in the world, you need to **publish it online**; that is, upload it to a cloud server using a **hosting** cloud service. This process creates a web link for your app that you can easily share.

In general terms, hosting consists of **renting space on a server** where your application files are stored so they can be accessed from the internet. Many hosting services have a fee, but for educational purposes and for Technovation Girls projects, we'll be using **free services** that allow you to upload web applications with certain limitations.

Prepare your folder in .ZIP format

Before uploading your app, make sure to zip the folder containing your files (`index.html`, `styles.css`, `script.js`):

1. Locate the folder where you saved the files.
2. Right click on it.
3. Select the option**"Compress into ZIP file"** or **“Compress to ZIP file”**.



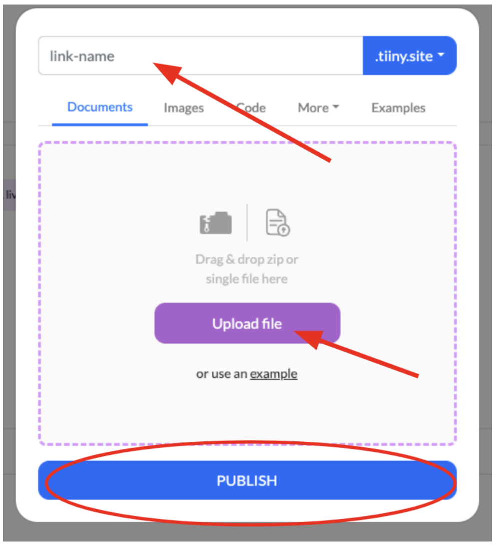
This .ZIP file will be the one you upload to the hosting platforms we'll look at below.

Publish with Tiiny.host

Tiiny.host is a simple and fast platform for publishing static websites. Ideal for beginners.

Steps to publish:

1. Create a free account. You can sign up with your email address, or log in with Google, Facebook, and more.
2. Once logged in, choose a name for your site.
3. Drag the `.ZIP` file with your web application to the indicated area.
4. Click on **“Publish”**.
5. Done! You'll get a link you can share with others.



Important considerations:

1. On the free plan, **files must be a maximum of 3 MB.**.
2. The website will remain online indefinitely, **as long as you log in at least once every three months**.
3. It is perfect for prototypes, demos and small projects

This isn't the only option for publishing your web app. We have two additional options that might interest you. To avoid cluttering this section, we invite you to check them out in the appendix if you wish to explore them.

Final recommendation

Either option you choose will allow you to have your app available online. Remember:

4. Always compress your folder as a `.ZIP` file before uploading it.
5. Test each generated link to make sure everything works correctly.
6. Save the links in a backup document.

Option with Visual Studio Code

If you find it difficult to use Notepad, here is another way to create your web app using a specialized program for writing code called **Visual Studio Code**.

Step 1. Install Visual Studio Code on your computer

1. Go to the official website: https://code.visualstudio.com
2. Click the download button and select the operating system your computer uses (Windows, Mac, or Linux).
3. Follow the steps in the installer until the program is ready to use.

Step 2. Create a folder for your project

Just as with the previous option, create a folder where you will save all your app files:

1. Go to your Desktop.
2. Right click on an empty space.

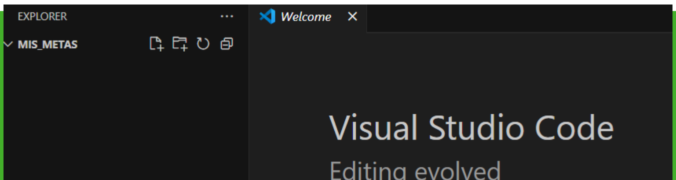
3. Select New → Folder.
4. Name your folder as: my-goals-new.

Follow the step 2 from the previous section to request the code from genAI.

Step 2. Open the folder in Visual Studio Code

1. Open Visual Studio Code.
2. Click on **"Open Folder..."**(or "Open folder").
3. Find and select the folder my-goals-new that you created.

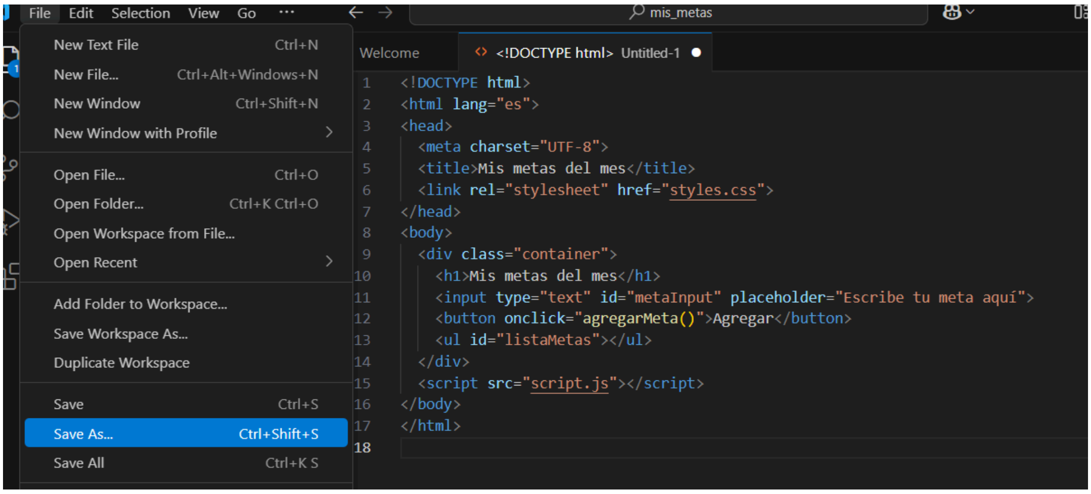
This will open your working folder within the program, ready for you to add files.



Step 3. Create your code files

1. HTML file

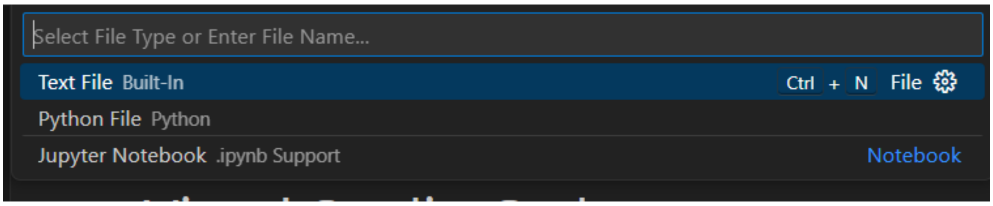
- Click on **"File"** → **"New File"** → **"New Text File"**
- A blank file will open. Paste the HTML code.
- Save the file as: **index.** using **"File"** → **"Save As..."** (you don't need to write `.html`, the program detects it automatically).



```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mis metas del mes</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Mis metas del mes</h1>
    <input type="text" id="metaInput" placeholder="Escribe tu meta aquí">
    <button onclick="agregarMeta()">Agregar</button>
    <ul id="listaMetas"></ul>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

2. CSS file

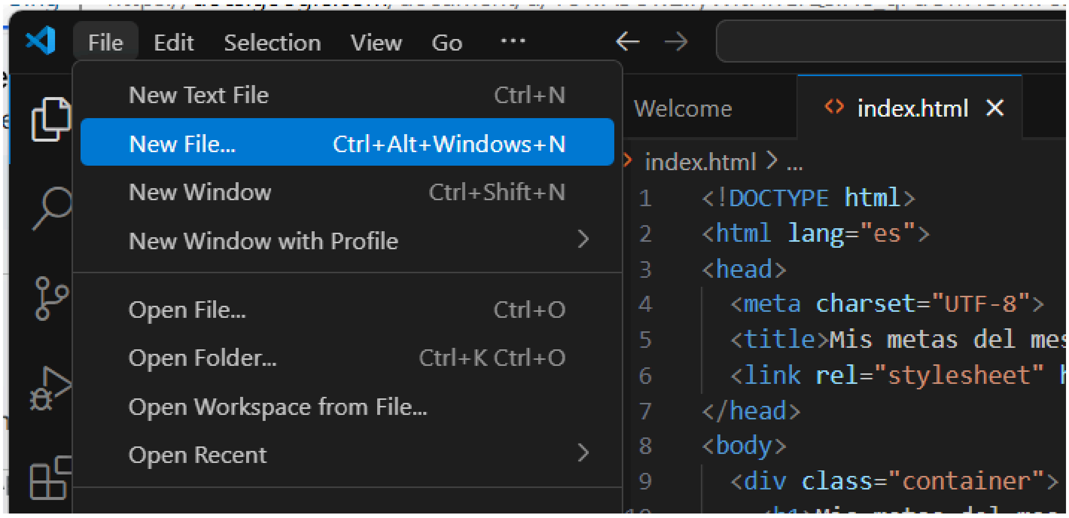
- Repeat the process:**"File"** → **"New Text File"**.


Select File Type or Enter File Name...
Text File Built-In
Ctrl + N
File
Python File Python
Jupyter Notebook .ipynb Support
Notebook

- Paste the CSS code.
- Save the file as: `styles`.

3. JavaScript file

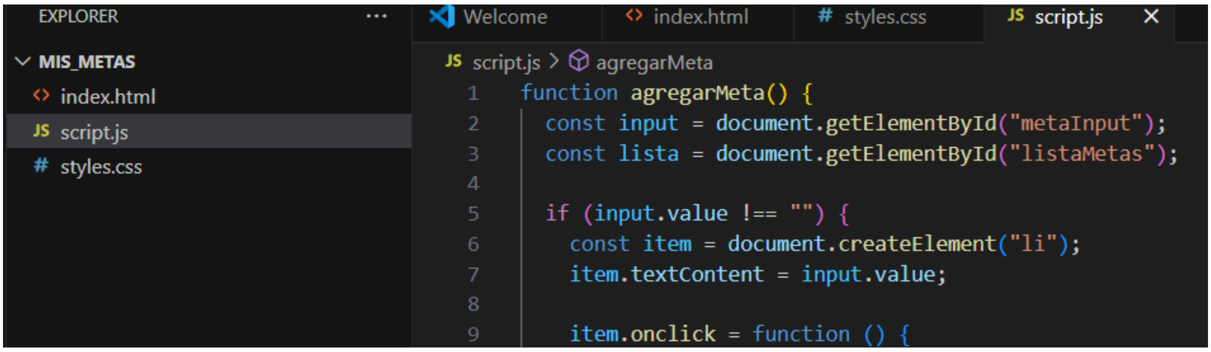
- One more time: **"File"** → **"New Text File"**.



```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Mis metas del me
  <link rel="stylesheet"
</head>
<body>
  <div class="container">
```

- Paste the JavaScript code.
- Save the file as: `script`.

After saving the three files, you'll see them listed in the left pane of Visual Studio Code, and also within the folder on your desktop.

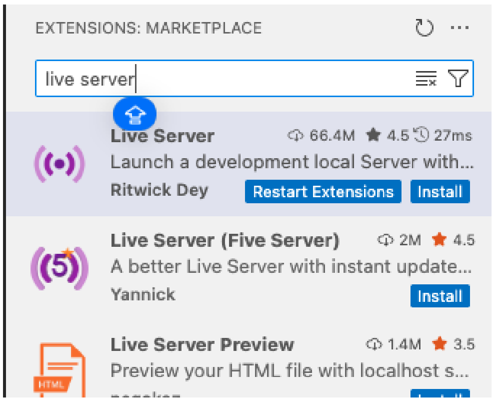


```
function agregarMeta() {
  const input = document.getElementById("metaInput");
  const lista = document.getElementById("listaMetas");

  if (input.value !== "") {
    const item = document.createElement("li");
    item.textContent = input.value;

    item.onclick = function () {
```

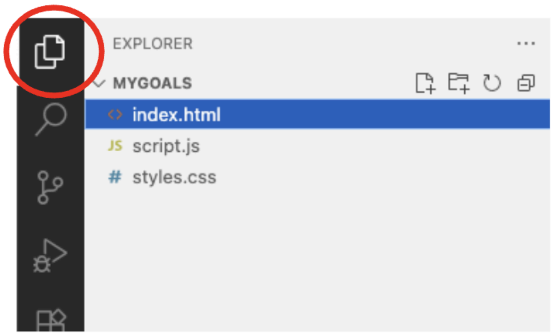
Step 4. Test your code

To test your code, you need to install an extension in Visual Studio Code called Live Server.

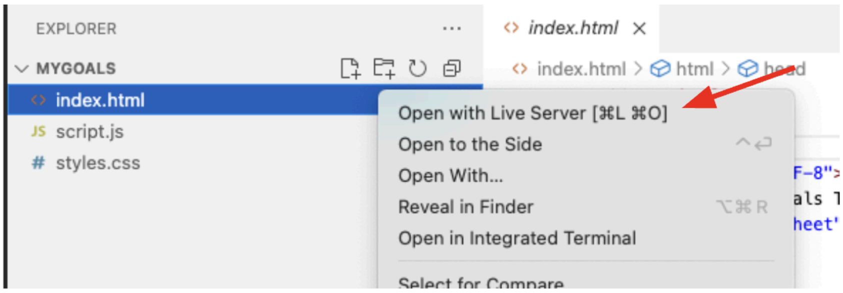
1. Go to View -> Extensions.
2. Type "live server" into the search field.
3. Locate the version by Ritwick Dey and click **Install**.


EXTENSIONS: MARKETPLACE
live server
Live Server 66.4M 4.5 27ms
Launch a development local Server with...
Ritwick Dey
Restart Extensions
Install
Live Server (Five Server) 2M 4.5
A better Live Server with instant update...
Yannick
Install
Live Server Preview 1.4M 3.5
Preview your HTML file with localhost s...
HTML

4. Now return to your file list by clicking on the Explorer icon on the left icon menu.



5. Right-click on index.html and select Open with Live Server.



This will open your web page in your browser. You can then follow steps 3 and 4 from the previous section to test and debug your code.

Once your app is ready, you can follow any of the options in **Step 5 from the previous option** to publish it on the internet and share it with the world.

Option 2. Web app with Python and Streamlit

Now we are going to create the same monthly goals app, but using **Python**.
Don't worry if you've never used Python before! We'll be using an online tool called **Streamlit Community Cloud**, which allows you to create apps without installing anything.

What is Streamlit?

Streamlit is a tool that converts Python code into an interactive web app. It runs in the cloud, so all you need is a GitHub account and a copy of the code.

Step 1. Ask ChatGPT for help generating the code

Go to https://chat.openai.com and type the following prompt:

I want to create a Python app with Streamlit that allows me to write my monthly goals, add them to a list, and mark them as completed. I have no prior experience, so please walk me through it step by step and tell me what each line of code does.

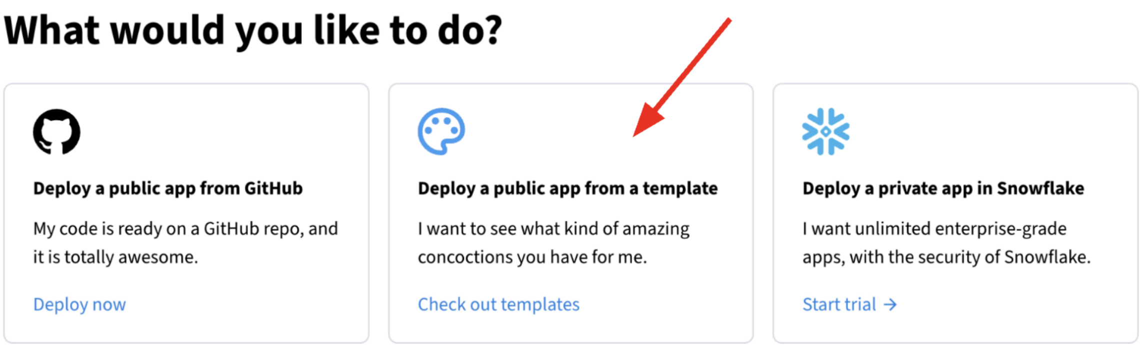
ChatGPT will explain the code to you line by line. You can copy the code it gives you, but first **read the explanations** to understand how each part works.

For example, the code could look like this (it may vary slightly depending on the AI):

```
import streamlit as st

st.title("🎯 My goals for the month")

goals = st.session_state.get("goals", [])

new_goal = st.text_input("Write a new goal:")

if st.button("Add goal"):

    if new_goal:

        goals.append({"text": new_goal, "completed": False})

        st.session_state.goals = goals
st.write("### Goal List:")

for i, goal in enumerate(goals):

    completed = st.checkbox(goal["text"], value=goal["completed"], key=i)

    goals[i]["completed"] = completed
```

Step 2. Create your app in Streamlit

1. Go to https://share.streamlit.io and log in with your Github account. If you do not have a Github account, you can create one.
2. Click on the Create app button in the top right.
3. Choose the middle button, “Deploy a public app from a template”



What would you like to do?

Deploy a public app from GitHub

My code is ready on a GitHub repo, and it is totally awesome.

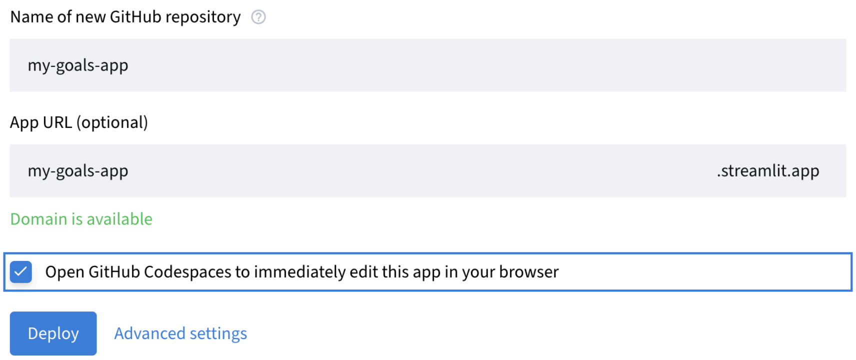
Deploy a public app from a template

I want to see what kind of amazing concoctions you have for me.

Deploy a private app in Snowflake

I want unlimited enterprise-grade apps, with the security of Snowflake.

4. In **Deploy from a Template**, choose **Blank App**
5. On the next screen, scroll down and give your Github repository and your Streamlit app a name. A Github repository is the most basic element of GitHub, and is essentially a project saved under your account. It's a place where you can store your code, your files, and each file's revision history. I chose **my-goals-app** but you can choose your own name. If a name is already taken, you can choose something else.



Name of new GitHub repository: my-goals-app

App URL (optional): my-goals-app .streamlit.app

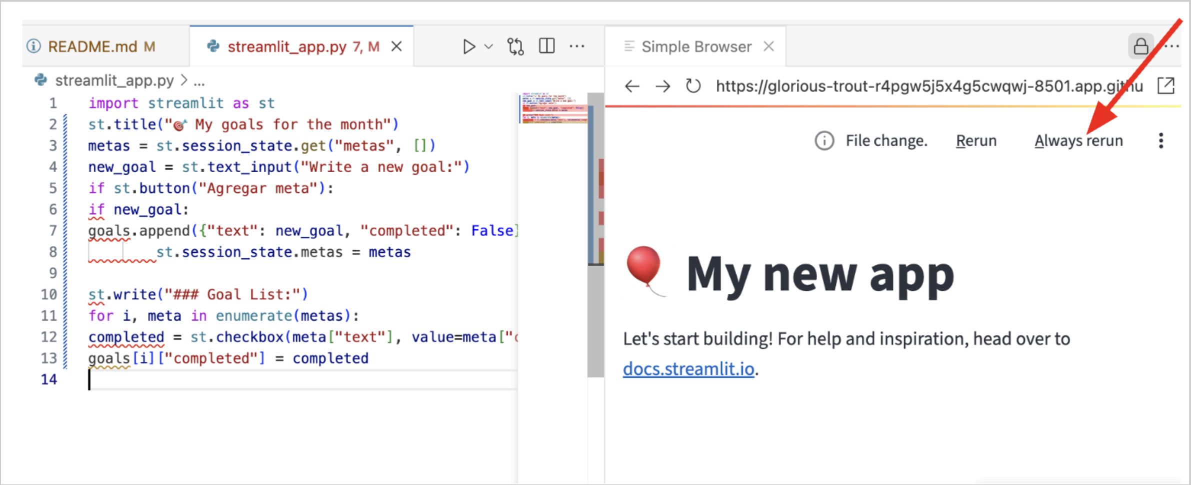
Domain is available

☑ Open GitHub Codespaces to immediately edit this app in your browser

6. Check the box next to Open Github Codespaces and click the Deploy button.
7. It will take several seconds but eventually your app code will appear in the browser, in an IDE that looks very similar to Visual Studio Code.
8. There are 3 files automatically added. README.md, requirements.txt, and streamlit_app.py.
    a. Edit the README.md file by deleting what is there and adding a description of your app.
    b. Click on streamlit_app.py and add the code that ChatGPT provided for your streamlit my-goals app.

Step 2. Test and debug your app

1. Note that your app is automatically running in the window on the right side of the screen. It detects any changes you make, so click the "Always rerun" option for it to automatically update and rerun your code when you make changes.



```
import streamlit as st
st.title("🎯 My goals for the month")
metas = st.session_state.get("metas", [])
new_goal = st.text_input("Write a new goal:")
if st.button("Agregar meta"):
if new_goal:
goals.append({"text": new_goal, "completed": False
            st.session_state.metas = metas

st.write("### Goal List:")
for i, meta in enumerate(metas):
completed = st.checkbox(meta["text"], value=meta["
goals[i]["completed"] = completed

```

My new app

Let's start building! For help and inspiration, head over to docs.streamlit.io.

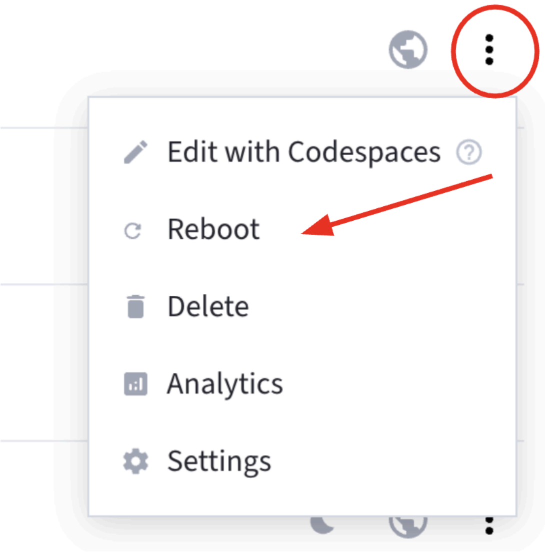
2. Test out your app and use ChatGPT to help you debug if you run into any issues.

Step 3. Commit your changes to Github and Streamlit

Once you are ready to make it public, you need to push your changes to your Github repository. This means the files you have just created and edited will get pushed up to your Github account and stored there. Github tracks all changes, just like Google drive does, but you have to "commit" the changes to your online account, saying - yes, I made my edits and want to save them now.

1. Make sure to save both your README.md and streamlit_app.py files.
2. Click on the Source Control icon in the left sidebar menu. This allows you to commit your changes, which will upload them to Github for you.
3. Type a comment under Changes, something like "initial commit to repository"
4. Click Commit.
5. Click Yes when asked if you want to stage your changes and commit directly.
6. Click the Sync Changes button to synchronize your github repository with your Streamlit project and do the upload.

You can now go back to https://share.streamlit.io, find your my-goals app and click on it to run it in the cloud. If the original template appears, you might have to reboot the project to get your updated app. Click on the 3 dots on the right side of the screen and click on Reboot.



Your app is now live! You can share your Streamlit app URL with friends and family!

How to continue learning with ChatGPT?

Here are some prompt ideas you can use if you want to improve your app:

- *How can I make my goals stay saved even when I close the app?*
- *I want the goals to be displayed in colors according to their status. Can you help me?*
- *How do I add a button to delete completed goals?*

Important Notes

- If you make changes to your code, remember to **commit changes to GitHub** for Streamlit to detect them.
- **If you use ChatGPT code, try to read and understand every part. That's what will make you a creator of technology, not just a user. Break it down and ask ChatGPT questions if you don't understand the code.**

What if something goes wrong?

Here are some tips for overcoming common obstacles:

Obstacle	What you can do with ChatGPT
Code error	Copy the entire message and tell the genAI tool: "This is the error I'm seeing: ... How can I fix it?"
App does not load in Streamlit	Verify that the file name is streamlit_app.py or app.py and is in the root repository. Also try a reboot.
I don't understand what a line of code does	Prompt: "Explain this line to me: session_state.get(...) as if I were 12 years old"
I don't see the same menus or buttons.	Describe what you see and ask: "This is what appears on my screen... what should I do?"

Coding Error

Copy the error message and ask for correction suggestions in chatpt.

Problem with Application Loading

Make sure the file is called app.py or streamlit_app.py and is in the root repository.

Understanding the Code

Ask ChatGPT to explain the specific lines of code in a simple way

OPTIONAL ACTIVITY

Convert a web app to a mobile app

In this chapter we are going to create a mobile application. You might be wondering: ***What is the difference between a mobile app and a web app?*** That's a very good question! To answer it simply, we're sharing a comparison table showing the main differences between the two.

Mobile application	Web application
• A program that runs natively on the phone. • Downloaded and installed on the device. • You can access phone functions such as GPS and camera. • Platform-specific (iOS or Android).	• It looks a lot like a mobile app. • It runs in an Internet browser. • It is not native to a specific device (iOS or Android) • It is usually coded with HTML, CSS, Javascript and Python. • Cannot run when offline.

• Coded with particular languages to adapt to the operating system.	
A web application is different from a website. A website is static, coded with HTML and CSS. Web applications are dynamic and change based on user input and other external interactions. For girls to participate in a Technovation project, a website is not acceptable.	

WebintoApp

This step is completely optional and not all girls need to do it. In fact, it's best to decide from the start whether their project will be a **web app** or a **mobile app**, based on the characteristics and needs of the community they seek to benefit. This decision is explored in depth in the [Technovation curriculum](#).

However, we decided to include this step in case a girl or team starts developing a web app (with HTML, CSS, and JavaScript) and later realizes it would be better to have it as a mobile app. There's no need to throw away your work and start from scratch! There are tools that allow you to easily convert your web app into a mobile app, and one of them is **WebIntoApp**.

When to use this step?

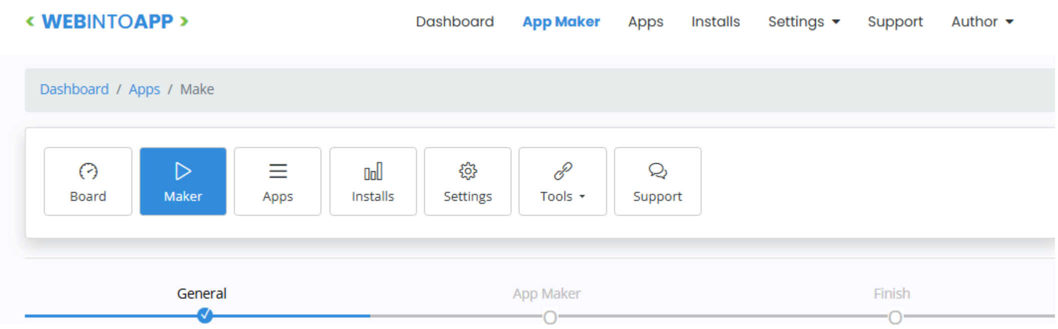
- If a web app has already been created.
- If the team believes its users would benefit more from an app that they can install directly on their phone.
- Only if it is **really necessary** to perform the conversion. Don't do it just out of curiosity, as this step can be technical and isn't required.

Steps to convert your web app into a mobile app

1. Register on WebIntoApp

- Enter: https://webintoapp.com
- Create a free account or log in.
- To register, click on the option "**Author"**and then "**register**".

2. Click on the "Maker" button



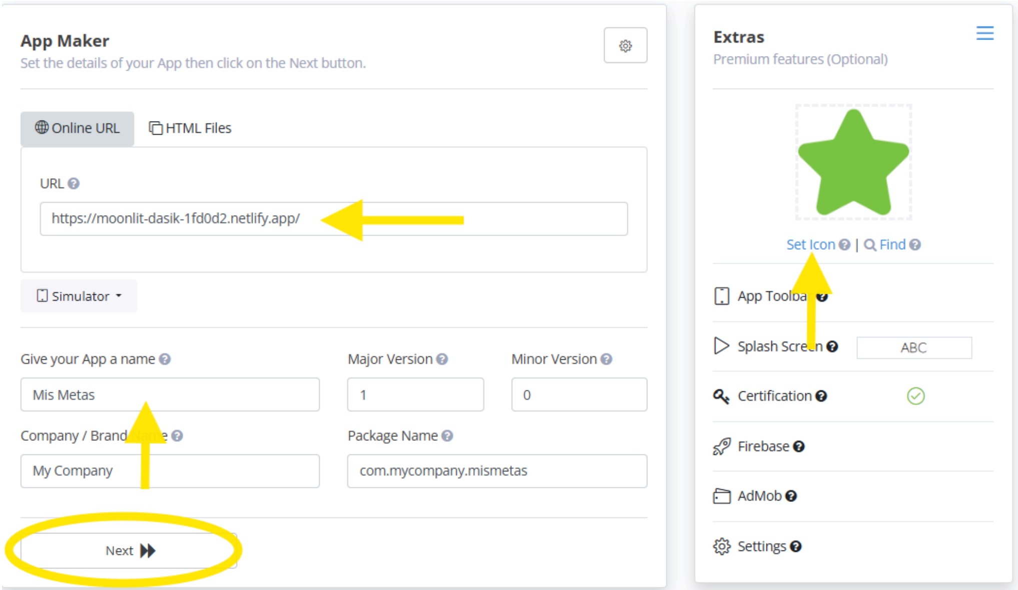
3. Select the "Online URL" option

Here you'll need to paste the link to your published web app. This can be a URL from:

- Tiiny.host
- Netlify Drop
- GitHub Pages

4. Customize your app

- Type a **name for your app**.
- Click on **"Set Icon"** to add a custom icon.
  You can create one with tools like **Canva** (using DreamLab) or search for one on **Flaticon**.
- Then, click on **"Next"**.



App Maker

Set the details of your App then click on the Next button.

URL: https://moonlit-dasik-1fd0d2.netlify.app/

Give your App a name: Mis Metas

Major Version: 1

Minor Version: 0

Company / Brand Name: My Company

Package Name: com.mycompany.mismetas

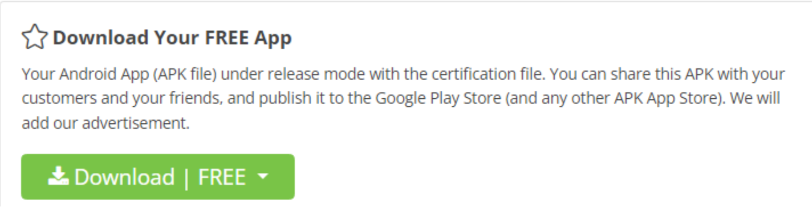
Extras

Premium features (Optional)

5. Create the app

Click on **"Make App"**.
Important note: In the free version you can only generate apps for Android devices.

Once the app is generated, you'll see a button to download it.



Download Your FREE App

Your Android App (APK file) under release mode with the certification file. You can share this APK with your customers and your friends, and publish it to the Google Play Store (and any other APK App Store). We will add our advertisement.

- A compressed file (.ZIP) will be downloaded.
- Right click on the file and select **"Extract everything"**.
- Inside the extracted folder, open the folder called **"android"**.
- Search for a file with the extension **.apk**. This file is the app ready to install on an Android device. This is the installation package for apps on Android, similar to the `.exe` en Windows or `.dmg` on Mac. It contains everything needed for the app to work.

6. Install the app on your cell phone (Android only)

1. **Open Google Drive** and access the folder from your cell phone to download the .apk file.
   or:
2. **Send it to yourself via WhatsApp Web,** opening a chat with yourself and attaching the file.

When you click on the file from your cell phone:

- Your phone will ask you for permission to install apps from "unknown sources."
- Temporarily enable those permissions to allow installation.

And that's it! Your app will be installed like any other on your phone.

Important: Even if you convert your web app into a mobile app with WebIntoApp, students **must submit the source code of their app** (HTML, CSS, JS) as part of their project to participate in Technovation Girls. To do so, you will compress all your files into a ZIP file and upload it to the Technovation platform .

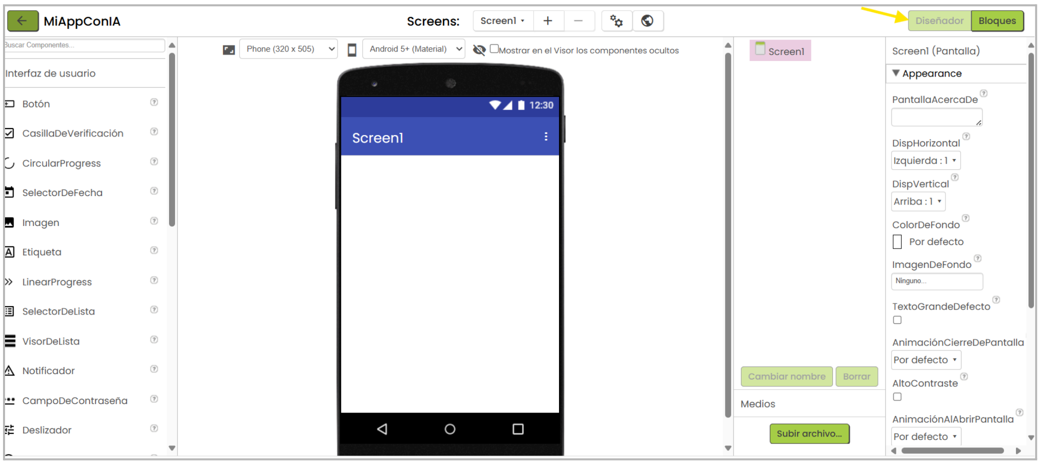
Remember: This step is **totally optional** and is only recommended if **you have already created a web app** that you invested time and effort into, but after researching your users' needs, you decided it would be better to offer it as a mobile app. This would allow you to send the apk file to your target users for them to download and install on their phones.

Creating a Mobile App with AI + App Inventor

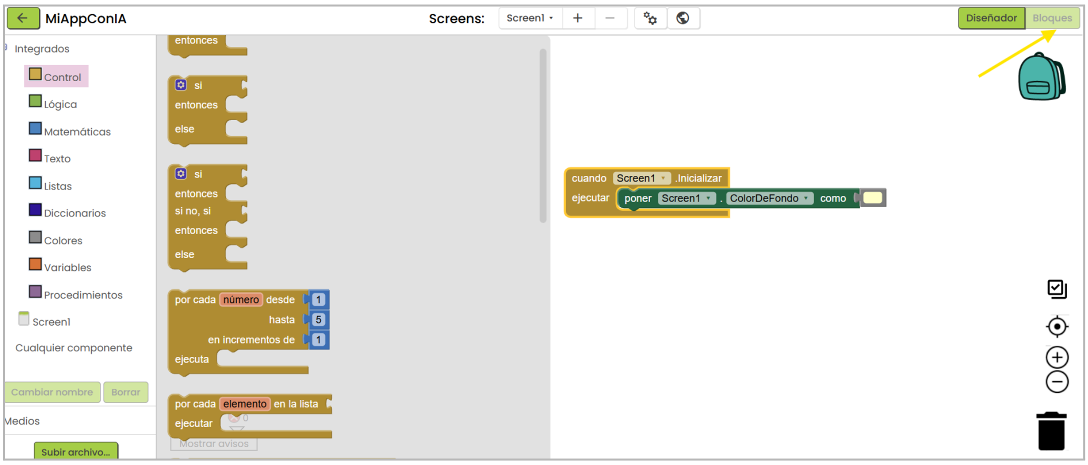
We are going to develop a mobile app and for that we will use a tool called **MIT App Inventor**, a free platform for creating mobile apps **without the need to program with code**, using a block system. Everything is done from the browser. It's ideal for those just starting out in technology creation. We'll go through it here, but if you want more information, check out our Technovation App Inventor lesson.

It consists of 2 parts:

1. **Designer:** where you drag and drop elements of your app, such as buttons, text, images, etc.



2. **Blocks Editor:** where you decide what each button or element does. Here you connect "blocks" like LEGO pieces.



It's important to note that Chat GPT can't create block-based programming, but it can guide you step-by-step through creating a mobile app in App Inventor.

Step 1: Request support from GPT Chat

- Log in to [Chat GPT]
- **Type this prompt**:

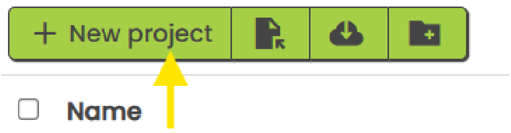
> *Help me create a mobile app in App Inventor.*
> *I want it do the following:*
> - *The user types a question.*
> - *When you press a button, the app gives you a random response like a fun artificial intelligence.*
> - *Responses should be motivational, thoughtful, and contain emojis.*
>
> *Give me step-by-step instructions and programming blocks. I've never used App Inventor before, so I need you to explain each visual part of the tool to me so I can learn how to use it.*

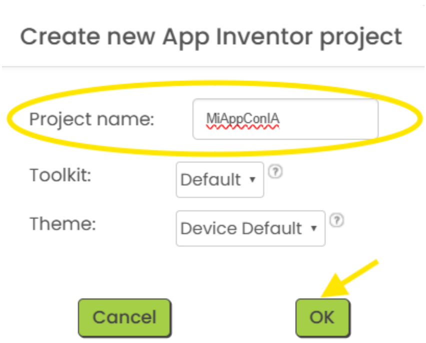
- **Watch how the AI responds.** It will give you steps, suggest blocks, and even sentence ideas.

Step 2: Open App Inventor and follow the steps

1. Open https://ai2.appinventor.mit.edu
2. Sign in with your Gmail account.

3. Click on"**New Project**"and name your project: `MyAppWithAI` and click “OK”




Create new App Inventor project

Project name: MiAppConIA

Toolkit: Default

Theme: Device Default

4. Follow the instructions given to you by ChatGPT.

For example, these were the results that Chat GPT gave me when I typed the suggested prompt:

STEP BY STEP TO CREATE YOUR APP

1. Enter App Inventor

- And a: 👉 https://ai2.appinventor.mit.edu
- Click on**“Create Apps!”**
- Sign in with your Google account
- Click on the button **“New project”**
- Write the name of the project: MyAppWithAI and click on**OK**

APP DESIGN (Designer)

In the **Designer** window, you will see:

- 🧩 **Palette:** list of elements you can use (such as buttons and text)
- 📱 **Viewer:** the screen where you see your app and arrange the elements
- 🧰 **Components:** list of the items you have already added
- ⚙️ **Properties:** where you change the text, colors, or size of selected element

ADDING ELEMENTS TO THE SCREEN

On the left side (Palette), open the categories and **drag** these elements to the viewer:

A) 🟦 Label

- Drag it to the center of the screen
- In **Properties**, change:
  - **Text:** Ask your fun AI a question! 🤖
  - **FontSize:** 20
  - **TextAlignment (Text Position):** Center

B) 🟧 TextBox (Text field)

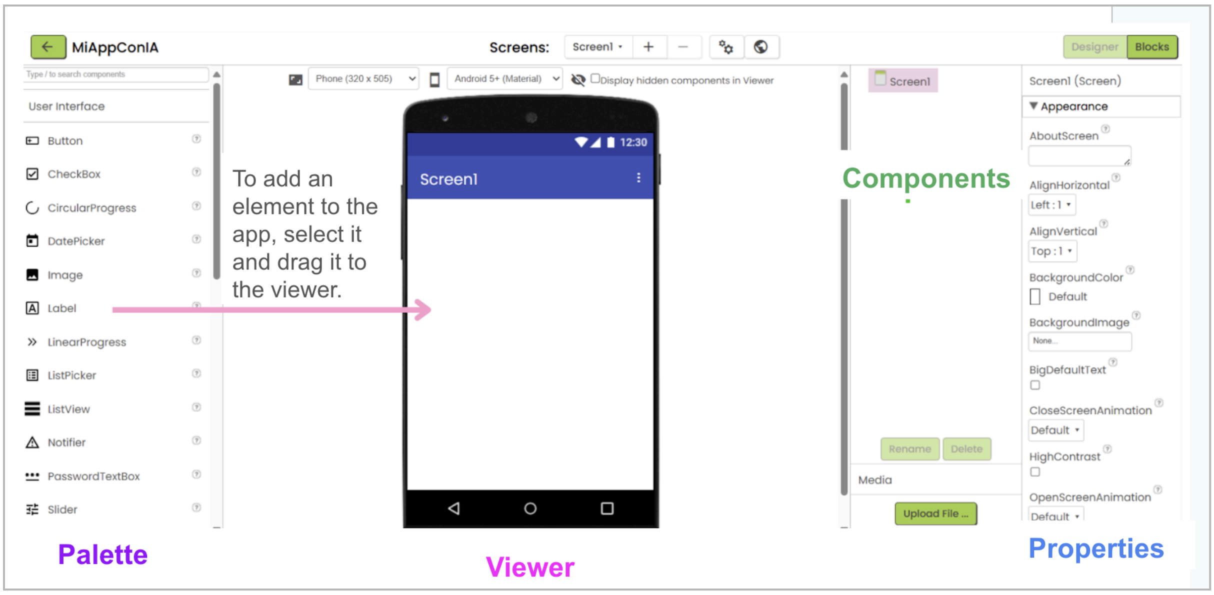
- Drag it under the label
- In **Properties**, change:
  - **Hint:** Write your question here
  - **Width:** :Select Fill parent (fit to container) so that it occupies the entire width

C) 🟩 Button

- Drag it below the text box
- In **Properties**, change:
  - **Text:** Get an answer 🧠

D) 🟪 Label (Another label)

- This will be the space where the answer will appear
- In **Properties**, change:
  - **Text:**(delete the text, leave it blank)
  - **FontSize:** 18
  - **TextAlignment (Text Position):** Center
  - **Width:** Fill parent (fit to container)



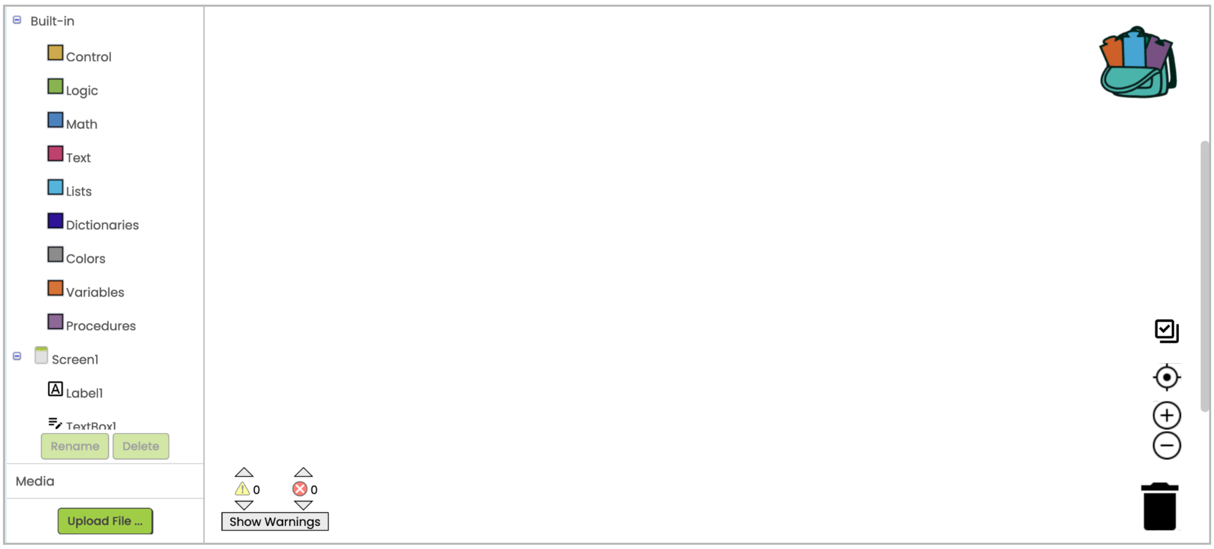
To add an element to the app, select it and drag it to the viewer.

PROGRAM THE APP'S LOGIC (Blocks)

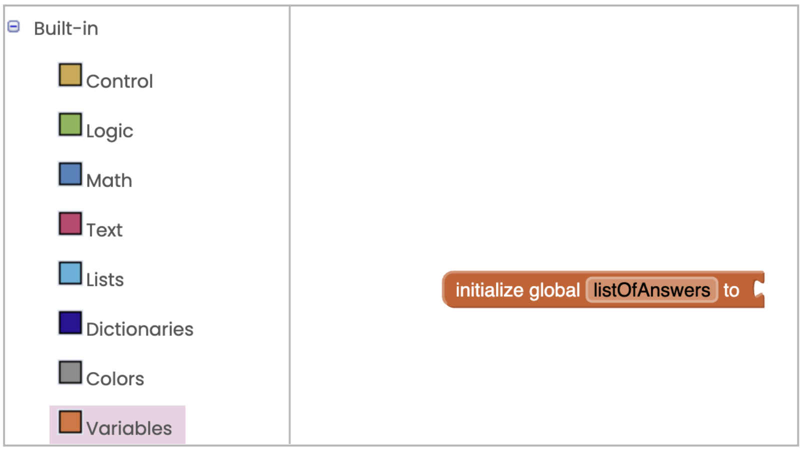
From here, we are including additional context and screenshots to the instructions, as ChatGPT will not include screenshots.

Click on the **"Blocks"** button (top right).

On the left side of the screen, you will see a list of built-in blocks followed by the components you added in the Designer. The built-in blocks contain blocks that can be used throughout your app. Click on Control to see the available Control blocks. In addition, each component has its own set of blocks. Click on Label1 and TextBox1 to see the blocks available for those components.



A) Create a list of answers

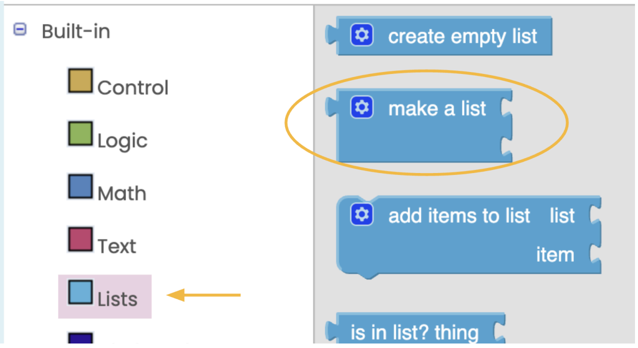


1. Click on **Variables** (left column).
2. Drag the block: `initialize global name to` Into the central workspace.
3. Click on "name" and change it to: `listOfAnswers`
4. From the **Lists** drawer, drag out the block:

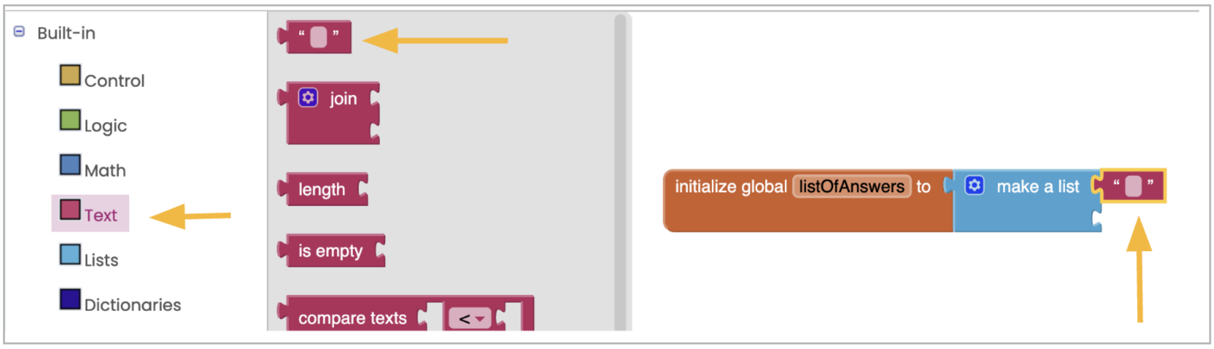
   `make a list`

   And snap it to

   `initialize global listOfAnswers to`



5. Now from the **Text** drawer, drag out blocks of text (those with quotes "") and snap them to the block `make a list`.



5. Write sentences like these in the pink text blocks:

   `"Trust yourself ✨ Your intuition is wiser than you think."`

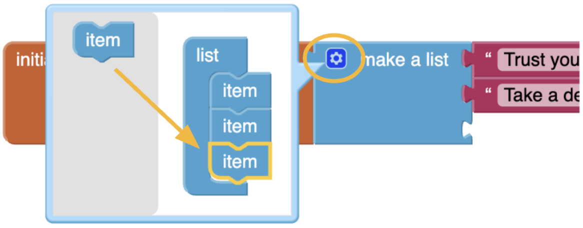
   `"There are no mistakes, only lessons learned 📚💡"`

   `"Take a deep breath and keep going 🌬️💪"`

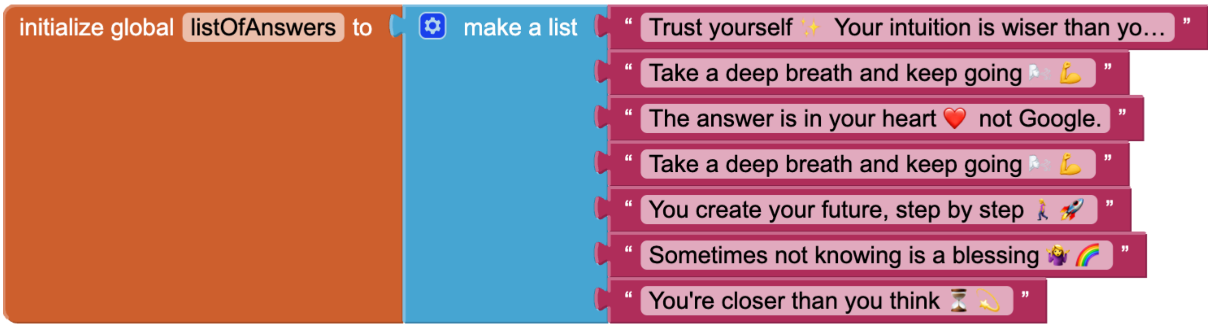
   `"The answer is in your heart ❤️ not Google."`

```
"You create your future, step by step 🚶‍♀️🚀"

"Sometimes not knowing is a blessing 🤷‍♀️🌈"

"You're closer than you think ⏳💫"
```

To add more elements to your block `make a list`, select the settings icon and add all the elements you require:



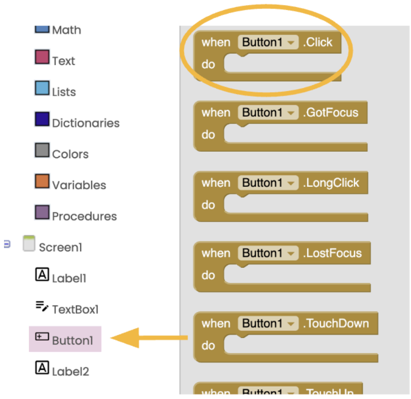
In the end the result should look like this:



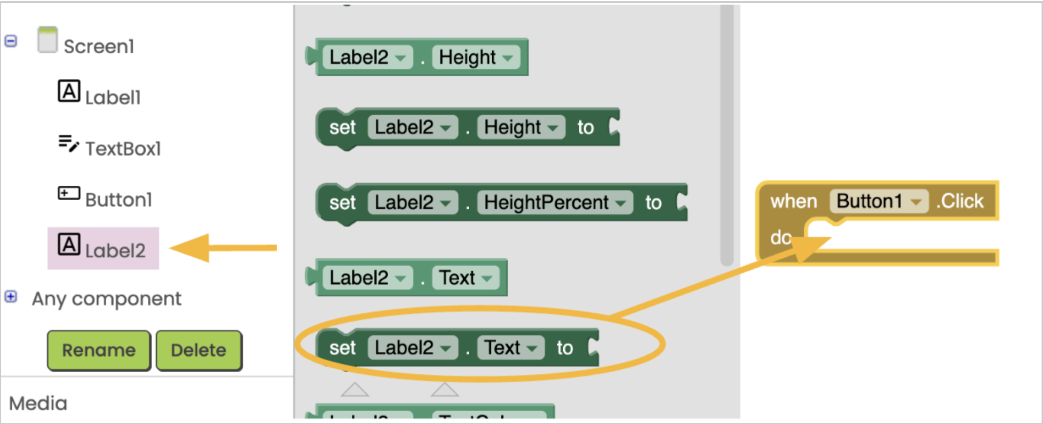
initialize global listOfAnswers to make a list
“ Trust yourself ✨ Your intuition is wiser than yo… ”
“ Take a deep breath and keep going 🌬️ 💪 ”
“ The answer is in your heart ❤️ not Google. ”
“ Take a deep breath and keep going 🌬️ 💪 ”
“ You create your future, step by step 🚶‍♀️ 🚀 ”
“ Sometimes not knowing is a blessing 🤷‍♀️ 🌈 ”
“ You're closer than you think ⏳ 💫 ”

B) Program the button to display a random response

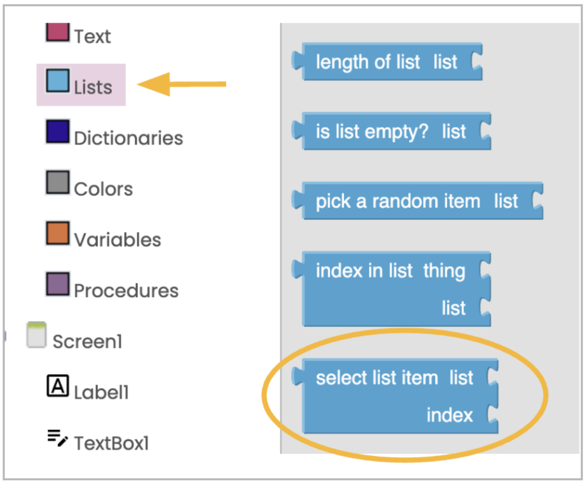
1. In the left panel, click on `Button1`
2. Drag out the block:
   `when Button1.Click do`



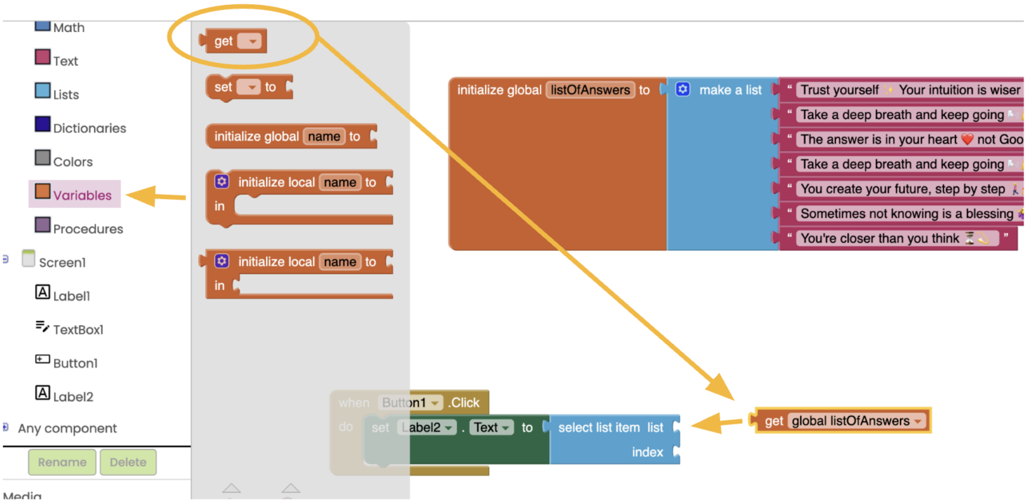
3. Now go to the response tag (for example `Label2`) and drag out :
   `set Label2.Text to`
   and snap it into the `when Button1.Click do` block.



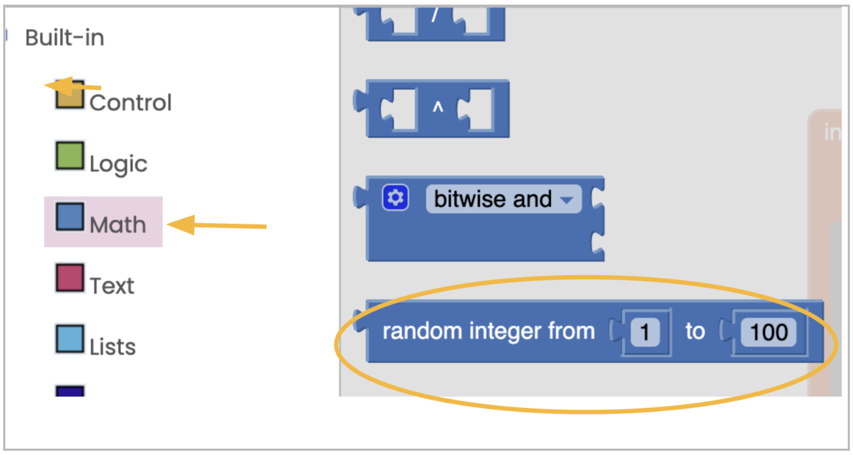
3. In the **Lists** drawer, find and drag out:
   `select list item`
   Snap it to `set Label2.Text to`



4. To that block:
    - Connect to the list slot: `get global listOfAnswers` (from the **Variables** drawer)

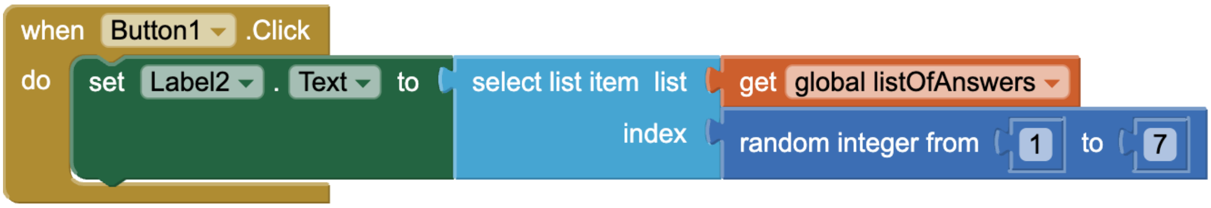

Math
Text
Lists
Dictionaries
Colors
Variables
Procedures
Screen1
Label1
TextBox1
Button1
Label2
Any component
Rename
Delete
Media
get
set to
initialize global name to
initialize local name to
in
initialize local name to
in
initialize global listOfAnswers to
make a list
Trust yourself Your intuition is wiser
Take a deep breath and keep going
The answer is in your heart not Goo
Take a deep breath and keep going
You create your future, step by step
Sometimes not knowing is a blessing
You're closer than you think
when Button1 .Click
do set Label2 . Text to select list item list
index
get global listOfAnswers

And for an index: go to the **Math** drawer and drag out:
`random integer from 1 to 100`



Change the number **100** to **7** (or however many phrases you added to your phrase list)

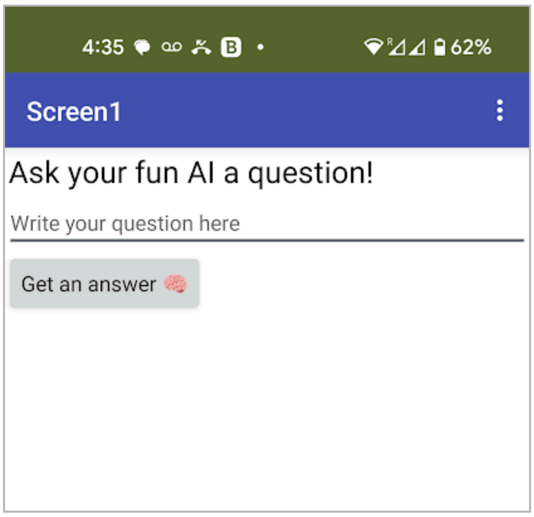
Connect everything! The block should look like this:


when Button1 .Click
do set Label2 . Text to select list item list get global listOfAnswers
index random integer from 1 to 7

TRY THE APP

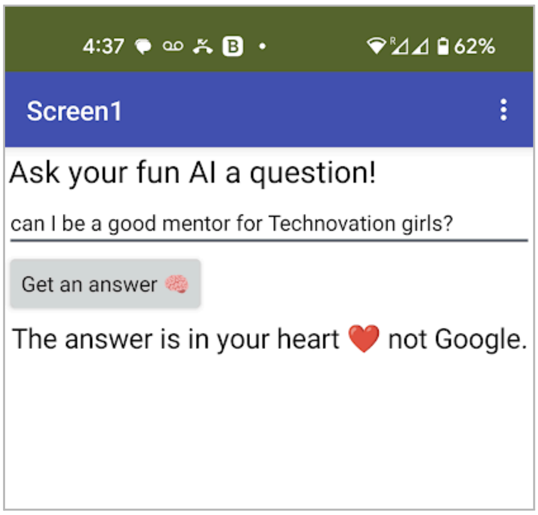
1. Download the app **MIT AI2 Companion** from the App Store or Play Store and install it on your phone.
2. In App Inventor, click **Connect → AI Companion**
3. Scan the QR code that appears from the MIT AI2 Companion app.

You'll see your app working in real time!



Ask your fun AI a question!

Write your question here



Ask your fun AI a question!

can I be a good mentor for Technovation girls?

The answer is in your heart ❤️ not Google.

If you have any questions or do not understand some of the instructions, **prompt ChatGPT to get answers**. For example:

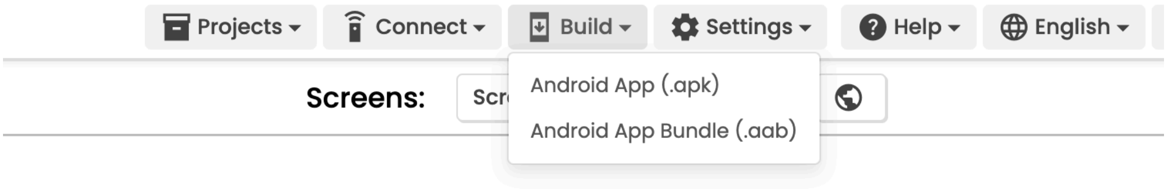
> *I didn't understand how to add a list of phrases to blocks. Can you explain it to me step by step?*

****Find more tips in this section.***

IMPORTANT: As mentioned above, ChatGPT does not directly generate code in a block-based programming language. The images accompanying this step-by-step guide were taken from the App Inventor platform while following the instructions provided by ChatGPT.

If at any time you find it difficult to understand the App Inventor interface, you can take screenshots of the process and send them along with your query in the *prompts. You can also right-click on a block and download it as a .png.* This will make it easier for the AI tool to guide you better.

Finally, if you manage to finish a working app and want to download it onto your phone as a complete app, simply click on **Build → Android App (.apk)**.



A **QR code** will be generated that you must scan with **your cell phone camera** (not with the MIT AI2 Companion app). Make sure you grant the necessary permissions for the download, and you'll be able to install and view your app directly on your mobile device in just a few moments.

Creating a Mobile App with AI + AppInventor (Chatbot Version)

Suggested mini activity:

Use ChatGPT to help you create a chatbot app that embeds ChatGPT into a mobile app.

1. *Open ChatGPT.*
2. *Type the following prompt:*
   Hi, I'm a teacher and I've never used App Inventor before. I want to create a mobile app that simulates an AI chatbot. Please guide me step by step without using technical menu or component names. Explain in simple terms how to:
   1. Add a text field for the user to type a question.
   2. Add a button that initiates sending that question to the AI.
   3. Place a space where the answer appears.

4. Configure the block that calls the ChatGPT service and displays the response.
I want each instruction to be clear ("click here," "drag this"), as if you were showing me the screen.

3. Follow the instructions the ChatGPT gives you to make the mobile app.
4. If you have questions or problems, prompt the AI tool to help you through the process. You can also check out this tutorial from App Inventor.

Ethical aspects and challenges of Generative AI

Generative AI is a powerful tool that has transformed the way we work, learn, and create. However, its use also involves **ethical challenges and responsibilities**, especially in educational contexts. In this final chapter, we'll address some of the main challenges in a simple and practical way.

Biases in AI: Why do they occur?

AI models, such as ChatGPT or DALL·E, **learn from large volumes of data** available on the Internet. This means they can reflect (and amplify) **existing biases** in those data.

For example:

- If a model has been trained primarily on English text, it may not understand other languages well.
- If a model is trained with more images of men in science than women, it might generate responses that reinforce gender stereotypes.

Why does this happen?

Because AI **does not reason like humans**: It simply repeats patterns. If the data is biased, the AI will be too. This can affect our decisions, especially when used in education, justice, or healthcare.

What can we do?

- Use AI as **a support tool, not the sole source of truth.**
- Teach your students to **detect and challenge biases.**
- Promote the development of AI with more diverse, fair, and representative data.

Data Privacy: What to Share and What Not to Share

Many AI models collect and process data to improve their performance. However, as educators, we must be protective of the **privacy and security** of our students and of ourselves.

Do not share:

- Full names of students.
- Sensitive information (such as address, ID, school).
- Photographs without explicit consent.

Practical recommendations:

- If you use an image generator, or are building an image classification model, **never upload photos of people without their permission.**
- When working with AI in class, remember to explain to the students **what information is safe to share** and what isn't.
- Read the privacy policies of the tools before using them.

Responsible use in education

AI can be a great ally in the classroom, but its use must be accompanied by **critical thinking, ethics and transparency.**

Transparency with students

It is important to tell our students **what is AI**, how it works, and what limitations it has. If we use AI-generated text in class, we can say:

> "This explanation was created with the help of artificial intelligence. Let's review it and see if it's clear or if anything is missing."

This helps students to **not assume that everything generated is correct**, and to develop a critical outlook.

Critical evaluation of what was generated

Just as not everything on the Internet is reliable, **what AI generates is not automatically accurate either.**

Invite your students to ask themselves:

- Does this make sense?
- Are there any errors?
- Is something important missing?
- From what perspective is it written?

These types of questions strengthen critical thinking and prevent AI from becoming an unquestionable "voice of authority."

Conclusion

Generative artificial intelligence is here to stay. It has transformed the way we create, learn, and communicate. However, with its power comes great responsibility, especially when working with girls and young women who are developing not only technical skills but also critical, ethical, and creative thinking.

Throughout this manual, we have been going over the most important aspects so that any teacher, regardless of their previous training in technology, can support students to participate in **Technovation Girls**, accompanying girls and adolescents on the exciting path of **creating technology with purpose**.

We began by understanding **What is generative AI?**, how it works, and why it's different from other forms of artificial intelligence. We learned that it's not magic, but a prediction system trained on vast amounts of data. Then, we explored **accessible tools like ChatGPT, Gemini and Canva AI**, which can be great allies in the classroom and in the development of technological projects.

We also learned how to talk to AI effectively, **using well-structured prompts** that allow us to obtain more useful, specific, and creative results. We understood that programming is a skill that any teacher can understand and teach. We became familiar with **the basics of programming**, and we took our first steps creating web and mobile applications with the help of artificial intelligence.

But perhaps the most important thing has been to recognize that technology, by itself, **does not solve the world's problems**. It requires conscious, empathetic, and responsible people who use it with a clear purpose. Therefore, we close this book by reflecting on the **ethical aspects of generative AI**: biases, data privacy, the need for transparency and critical evaluation of what is generated.

As leaders and mentors of *Technovation Girls*, you are not only going to teach how to use technological tools, but also how to **encourage conscious digital citizens**, capable of imagining and building solutions for their communities, from an ethical, inclusive and transformative perspective.

This manual is just the beginning. What comes next—the ideas, the projects, the conversations, the discoveries—**It will depend on you and the girls you work with.** May this journey be a space for mutual growth, inspiration, and confidence in the power of education and technology to change the world.

Appendix

What is Generative AI?

Before we begin, it's important to understand what Artificial Intelligence (AI) is, specifically Generative AI, which you'll use throughout this manual. There are several definitions of AI:

- “The ability of a system to correctly interpret external data, learn from that data, and use that learning to accomplish specific goals and tasks through flexible adaptation” (Kaplan and Haenlein, 2019:15).
- “Any technique that enables computers to mimic human behavior” (Jiang et al., 2022:5).

We can summarize the concept of AI as **the ability of computers to perform tasks that normally require human intelligence.** This includes skills such as **learning from experience, recognizing patterns, adapting to new situations, and making decisions independently.** It's as if a machine can observe, think, and act on its own, even if it doesn't do so exactly like a person. AI learns through examples, similar to how humans learn when we practice something repeatedly.

And what do we mean by Generative AI? Well, traditional machine learning AI focuses on learning patterns to solve specific tasks. For example, a virtual assistant that tells you the weather or an app that recognizes whether an image shows a cat or a dog. This type of machine learning AI learns to recognize patterns and make decisions (generates an outcome) from data, but it doesn't create new things.

Instead, **generative AI** can **create original content**, such as *text, images, music* or even *videos*, based on what it has learned. For example, if we ask a generative AI tool to write a story about a boy who travels to space, it can invent it entirely from scratch. This is possible because it has been trained with vast amounts of information and is able to combine it creatively.

Feature	Machine Learning AI	Generative AI
What are you doing?	Recognize patterns, answer simple questions, classify, predict.	Create new content: text, images, music, code, etc.
How does it work?	Use trained models built with algorithms based on existing data.	Use trained models built with algorithms based on existing data.
Tool example	**Siri**, **Alexa**, spam filters in emails.	**ChatGPT**, **Copilot, Claude Gemini, Canva AI, DeepSeek**.
Example in education	Siri answers what a word means.	ChatGPT writes a lesson plan or generates customized exercises.
Example outside the classroom	A map app like Google Maps or Apple Maps that determines the best route for you to travel.	An AI that generates an artistic image from a description.
Level of creativity	**Low**: performs defined and repetitive tasks.	**High**: can combine ideas, invent and produce original content.

How does Generative AI work?

Generative artificial intelligence uses very advanced machine learning models called **deep learning models.** These models are inspired by the way the human brain works: they learn to identify patterns and relationships within large amounts of information, and then use that knowledge to respond to instructions or questions creatively.

Generative AI Process

Although everything happens in seconds when we use a tool like **ChatGPT** or **DALL-E**, behind it there is a complex process that can be divided into three major stages:

1. Training (foundation model)

It all starts with the creation of a **foundation model**, which is like a vast knowledge base for generative AI. This model is trained on vast amounts of data—for example, books, articles, web pages, or images—that aren't sorted or categorized. Through many tasks (such as guessing the next word in a sentence or completing an image), the AI learns to understand and generate coherent content.

These models are so large that training them requires weeks of work on thousands of powerful computers, which entails costs in the millions, as well as lots and lots of energy. Examples of foundational models are **GPT (used in ChatGPT)**, **FLAME (from Meta)** and **Stable Diffusion** (for images).

2. Adjustment (model customization)

Once you have the foundation model, you can **adjust** for specific tasks. For example, if you want to create a virtual assistant for a school, it's trained with frequently asked questions and answers from the educational field.

This process can be done in two ways:

- **Fine tuning:** The AI is fed with labeled examples (question-answer) related to a specific task.
- **Learning with Human Feedback (LHF):** People review what the AI generates and give it feedback, helping it improve its responses.

3. Generation and continuous improvement

Once AI is in use, its results are constantly evaluated. If it generates errors or unhelpful content, developers fine-tune it again so it learns from those mistakes. This is key to ensuring applications continue to improve over time.

In addition, methods like **RAG (Recovery Augmented Generation)** are used, which allows AI to consult updated external sources, like databases or trusted websites, to provide more accurate and up-to-date answers.

Although all this happens "behind the scenes", knowing these phases helps to understand why some generative AI tools, such as **Copilot**, **ChatGPT, Claude** and **Gemini** may seem so smart: they are not improvising, but using everything they learned to generate something new and useful in seconds.

Neural networks and text prediction

So far, we've learned that generative AI models learn by observing massive amounts of data and looking for patterns. To achieve this, they use a key technology called and **artificial neural network**, which mimics (in a very simplified way) the way the human brain processes information.

What is a neural network?

A neural network is a mathematical model inspired by the central nervous system. It is composed of **"artificial neurons"**, which are not real cells, but small processing units connected to each other. These neurons are organized in **layers**:

- **Input layer**: receives the information (for example, the initial words of a sentence).
- **Hidden layers**: They process information through millions of internal connections and adjustments.
- **Output layer**: generates the final result (e.g., the next probable word).

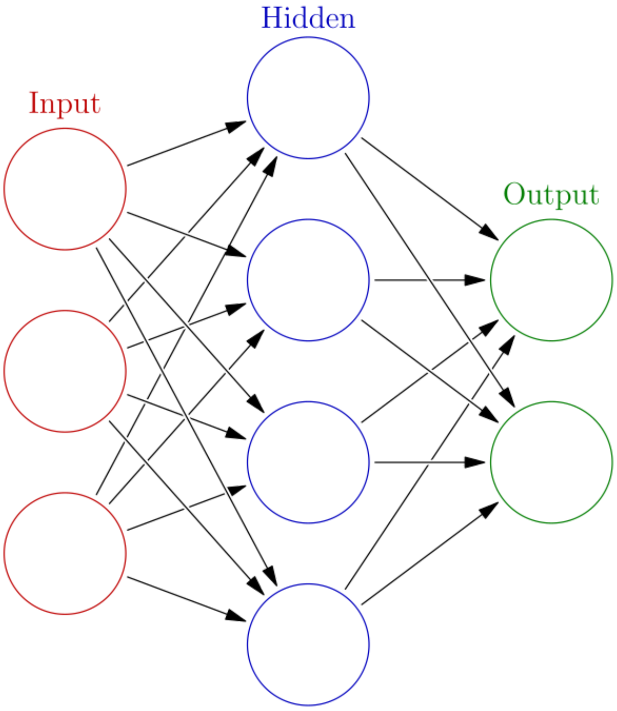


Image 1. Artificial neural network. Courtesy of https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

This process is what allows tools like **ChatGPT** to complete sentences coherently, answer questions, or generate text from scratch.

Imagine you're writing a text and your phone suggests the next word. That's text prediction. In generative AI models, this process happens on a massive scale thanks to neural networks.

At first, all connections between neurons have random values, so the results are incorrect or meaningless. But the network **learns** over time. You show them examples (real sentences), compare what you predicted with what actually came next, and **errors are corrected,** making small adjustments to the connections. This is called **training**.

This training occurs millions of times. With each repetition, the network becomes more accurate and learns to better predict which word (or image, or sound) should come next.

> *For example, if the entry is: "The sun sets behind…", the AI will search through millions of similar patterns and predict with high probability that the next word may be "the mountains" or "the horizon".*

You can imagine this process as a giant game of **"fill in the blanks".** The AI is given sentence fragments with a missing word, and it tries to guess the correct one. If it fails, its "internal connections" are adjusted and it tries again. In this way, the network gradually improves its ability to predict, understand, and generate language.

Once the network has been trained and tuned, it can be used in tools such as:

- **ChatGPT** or **Gemini**, to generate text.
- **Github Copilot** or **Claude**, to write code.
- **Siri** or **Alexa**, to answer spoken questions.
- **Canva Dream lab** or **DALL-E**, to create visual content.

Can an AI recognize your drawings? Try out Quick Draw, a neural network that uses a machine learning algorithm to analyze the strokes and shapes of drawings made by players. As more people play, the AI collects more data and becomes more accurate at identifying drawings.

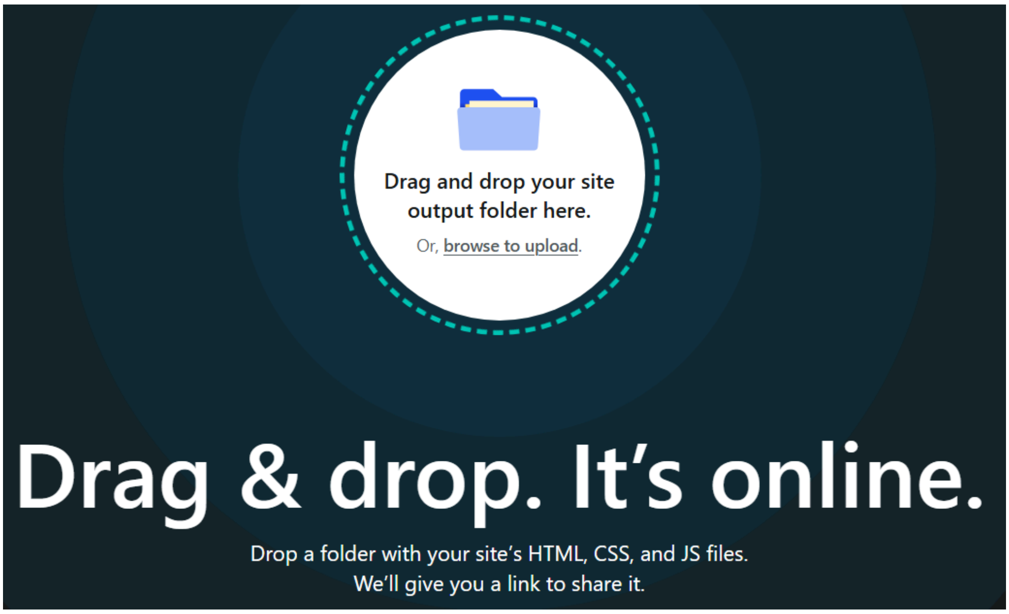
Other options for publishing your web app

Option 2: Publish with Netlify Drop

Netlify Drop It is another very useful tool to easily publish your website for free.

Steps to publish:

1. Go to Netlify Drop.
2. Drag your .ZIP folder to the area that says:
   "Drag and drop your site output folder here."



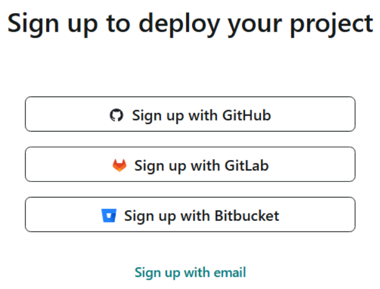
Drag and drop your site output folder here.

Or, browse to upload.

Drag & drop. It's online.

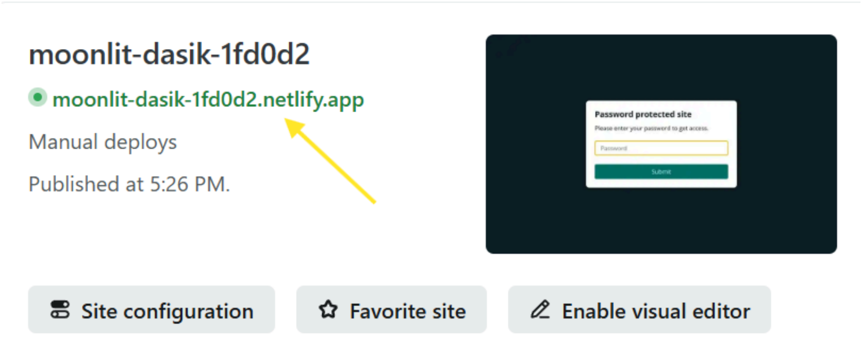
Drop a folder with your site's HTML, CSS, and JS files.
We'll give you a link to share it.

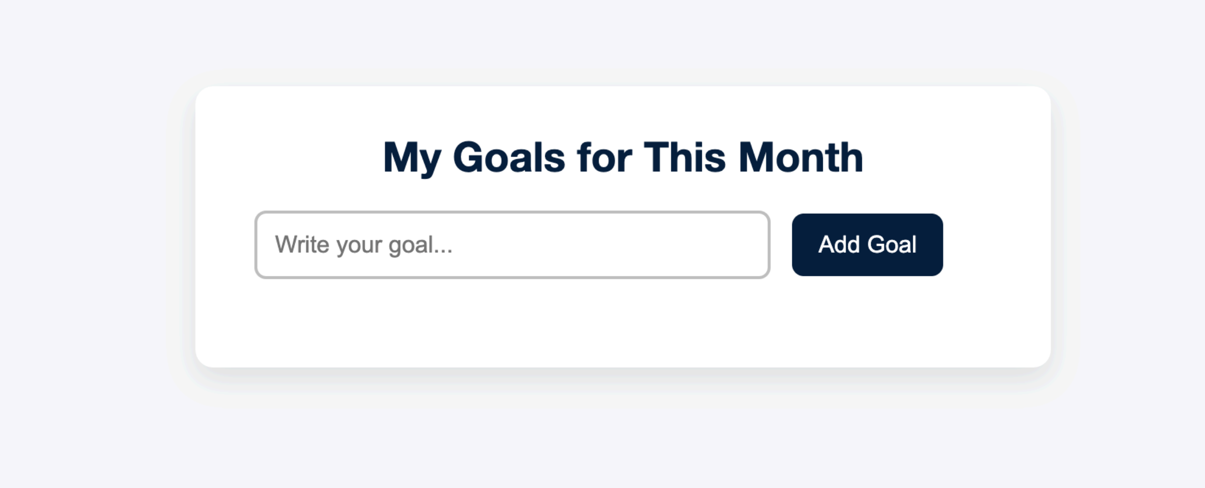
3. Finally, **Sign up with your Gmail or GitHub account**.
   If you followed the GitHub Copilot activity in this course, you should already have a GitHub account. We recommend using it for this step. If you have repositories created on GitHub, you may be prompted for permission to update all your repositories when linking to GitHub. Please keep this in mind.



Sign up to deploy your project

4. Once your app is published, you'll see a link.
   Click to view your website online.



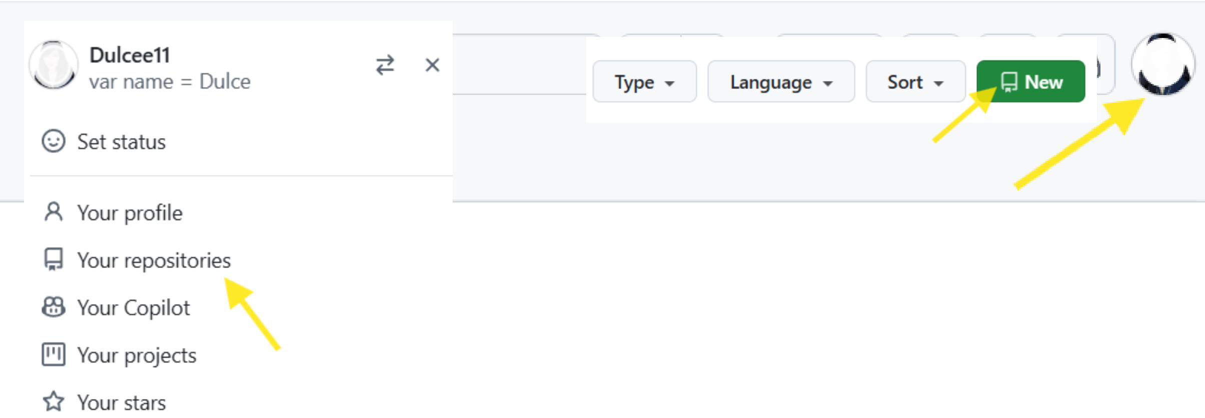


Option 3: Publish with GitHub Pages

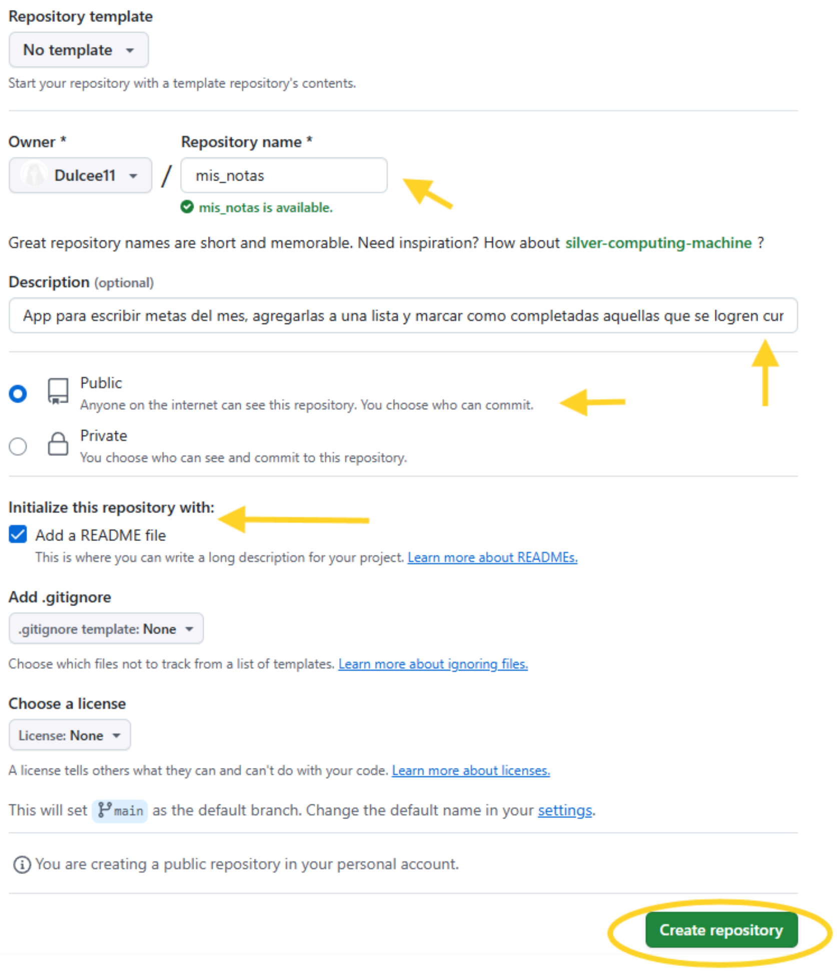
GitHub Pages is a free service for publishing static web pages. It's ideal if you already have an account and want to become more familiar with this platform, which is also widely used in professional development.

Steps to publish:

1. **Sign in** to GitHub.
2. **Create a new repository**:
    - Go to your profile > **Your repositories** > **New**.



- Assign a descriptive name (for example, my-goals-of-the-month).
- Check the following options:
  - Visibility: **Public**
  - Add README file :**Yes (Add a README file)**



Repository template

Start your repository with a template repository's contents.

Owner * Dulcee11 / Repository name * mis_notas

mis_notas is available.

Great repository names are short and memorable. Need inspiration? How about silver-computing-machine ?

Description (optional)

App para escribir metas del mes, agregarlas a una lista y marcar como completadas aquellas que se logren cur

● Public
Anyone on the internet can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

Initialize this repository with:

☑ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.

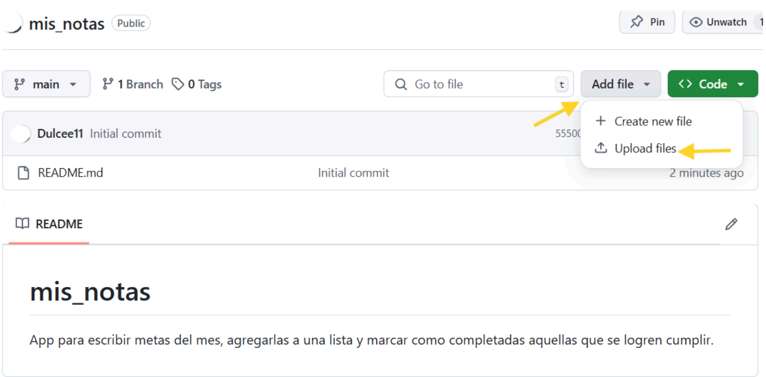
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.

This will set `main` as the default branch. Change the default name in your settings.

You are creating a public repository in your personal account.

- Click on**"Create repository"**.

3. **Upload your web files**
   - Click on**"Add file"** > **"Upload files"**.
   - Drag the individual files (not the ZIP file) from your folder my_goals: **index.html, style.css, script.js.**
   - Click on**"Commit changes"**to save them.



Dulcee11 Initial commit

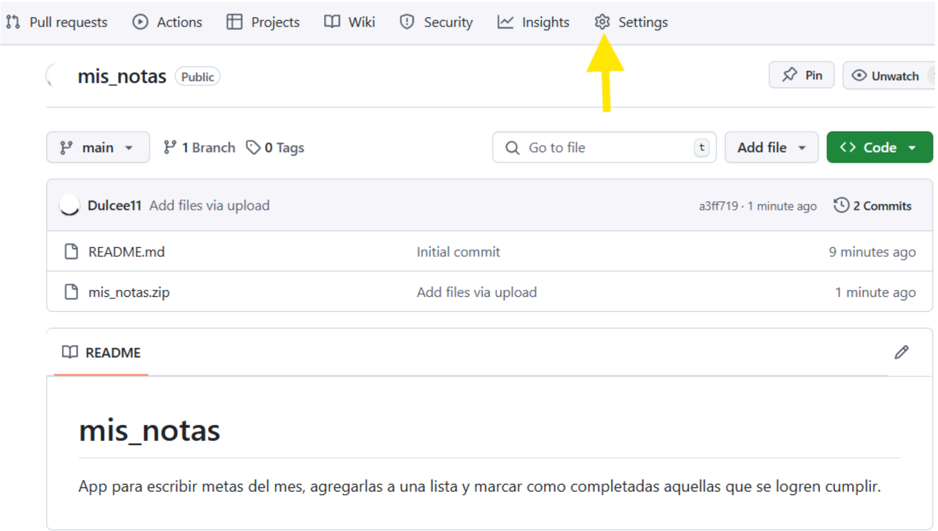
README.md Initial commit 2 minutes ago

mis_notas

App para escribir metas del mes, agregarlas a una lista y marcar como completadas aquellas que se logren cumplir.

4. **Configura GitHub Pages**:

   - Go to the **Settings** tab from the repository.



Add files via upload

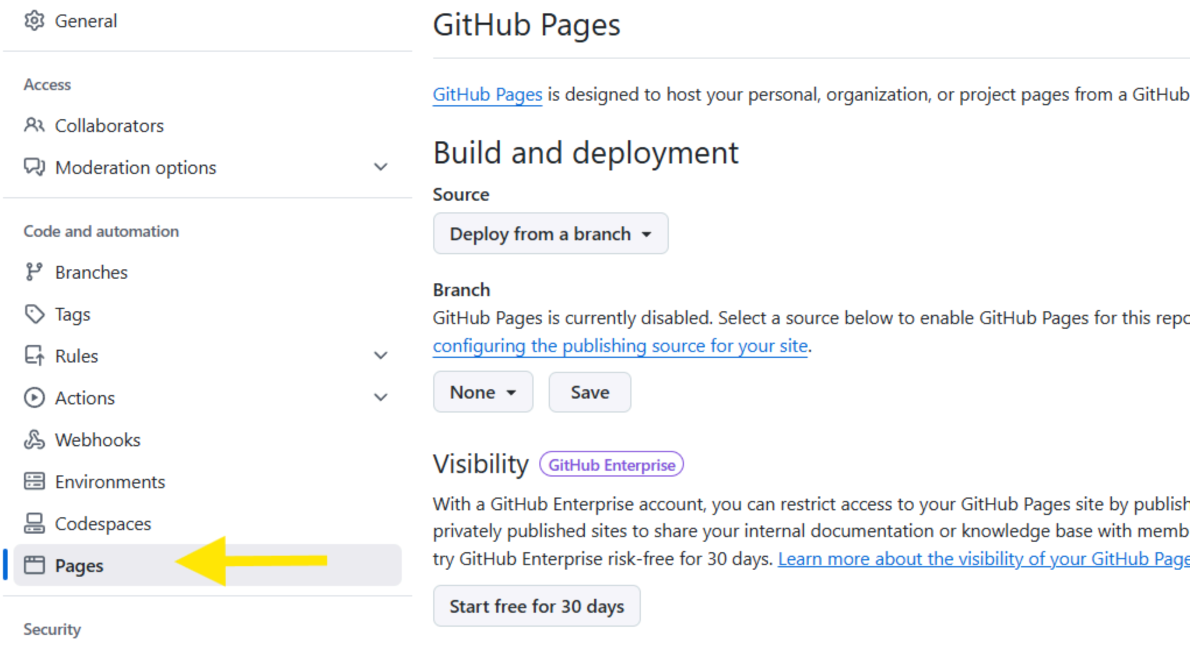
Initial commit

Add files via upload

mis_notas

App para escribir metas del mes, agregarlas a una lista y marcar como completadas aquellas que se logren cumplir.

   - Search for the section **"Pages"** on the left side menu.

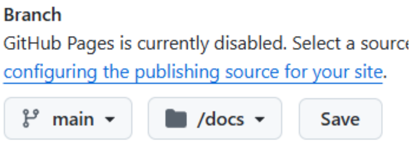


GitHub Pages is designed to host your personal, organization, or project pages from a GitHub

GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repc
configuring the publishing source for your site.

With a GitHub Enterprise account, you can restrict access to your GitHub Pages site by publish
privately published sites to share your internal documentation or knowledge base with memb
try GitHub Enterprise risk-free for 30 days. Learn more about the visibility of your GitHub Page

   - Under **"Branch"**, select:
     - Branch: **main**
     - Folder: **/docs**
   - Click on**"Save"**.



GitHub Pages is currently disabled. Select a sourc
configuring the publishing source for your site.

For GitHub Pages to work, files must be inside a folder called docs. You can create it directly from GitHub, move the files there, and then apply these settings.

Your website will be ready in a few minutes. You'll see a message at the top of the page letting you know when it's ready:***"Your site is active on..."*** GitHub will generate a link with this structure:

https://**your_username**.github.io/**repository_name**/

You can share this link with others so they can view your app from anywhere.

References

Binns, R. (2018). *Fairness in Machine Learning: Lessons from Political Philosophy*. Proceedings of the 2018 Conference on Fairness, Accountability and Transparency. https://doi.org/10.1145/3287560.3287583

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv https://arxiv.org/abs/2005.14165

Celik, I., Dindar, M., Muukkonen, H. et al. The Promises and Challenges of Artificial Intelligence for Teachers: a Systematic Review of Research. TechTrends 66, 616–630 (2022). https://doi.org/10.1007/s11528-022-00715-y

devCamp (2020) *What is a variable in programming?*https://devcamp.es/que-es-variable-programacion/

Educaopen tech school (2022) *curls* https://www.educaopen.com/digital-lab/metaterminos/b/bucle#:~:text=Metat%C3%A9rminos-,Bucle,escribiendo%20el%20mismo%20c%C3%B3digo%20continuamente

Floridi, L., & Cowls, J. (2019). *A Unified Framework of Five Principles for AI in Society*. Harvard Data Science Review. https://doi.org/10.1162/99608f92.8cd550d1

Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2021). *Datasheets for Datasets*. Communications of the ACM, 64(12), 86-92. https://doi.org/10.1145/3458723

González Duque, R. (2011). Python for everyone.

IBM (March 22, 2024).*What is Generative AI?*https://www.ibm.com/mx-es/think/topics/generative-ai

Jiang, Yuchen, Xiang Li, Hao Luo, Shen Yin y Okyay Kaynak (2022), "Quo Vadis Artificial Intelligence?", Discover Artificial Intelligence, vol. 2, no. 4. DOI:https://doi.org/10.1007/s44163-022-00022-8

Kaplan, Andreas y Michael Haenlein (2019), "Siri, Siri, in my Hand: Who's the fairest in the land? On the interpretations, illustrations, and implications of artificial intelligence", Business Horizons, vol. 62, núm. 1, pp. 15-25.

Kumar, H., Musabirov, I., Shi, J., Lauzon, A., Choy, K. K., Gross, O., Kul zhabayeva, D., & Williams, J. J. (2022). Exploring The Design of Prompts for Applying GPT-3 based Chatbots: A Mental Wellbeing Case Study on Mechanical Turk. University of Toronto.

McTear, M., Callejas, Z., & Griol, D. (2016). The Conversational Interface: Talking to Smart Devices. Springer.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2018). Language models are unsupervised multitask learners. OpenAI blog, 1(8), 9.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., & Pineau, J. (2017). A hierarchical latent variable encoder-decoder model for generating dialogues. In Thirty-First AAAI Conference on Artificial Intelligence.

Sun, S., Zhang, C., Huang, L., & Li, J. (2019). Context-aware response generation for multiturn conversation with deep reinforcement learning. IEEE Access, 7, 49918-49927.

Tapalova, O., & Zhiyenbayeva, N. (2020). Artificial Intelligence in Education: AIEd for Personalised Learning Pathways. European Journal of E Learning, 18(5), 361-373. https://doi.org/10.34190/ejel.20.5.2597

UNICEF. (2021). *Policy Guidance on AI for Children*. https://www.unicef.org/globalinsight/reports/policy-guidance-ai-children

UNESCO. (2021). *Artificial Intelligence and Education*: *Guidance for Policy-makers*. https://unesdoc.unesco.org/ark:/48223/pf0000376709

UNESCO. (2023). *Guide to the use of artificial intelligence in education*. https://unesdoc.unesco.org/ark:/48223/pf0000385008

Vossen, P., Caselli, T., Lapesa, G., & Saggion, H. (2018). A survey of current datasets for natural language processing and understanding. arXiv preprint arXiv:1806.05667.