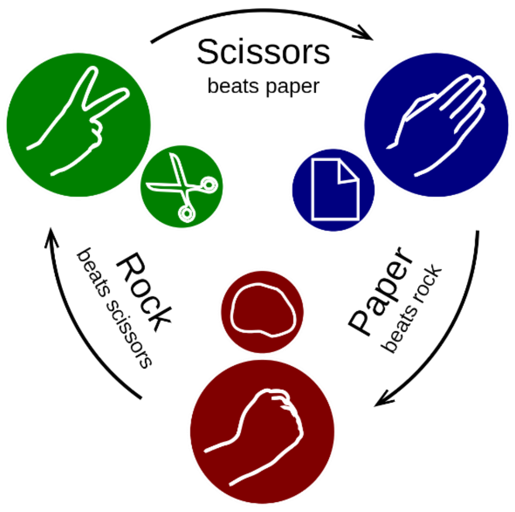
Practical Project: Rock–Paper–Scissors

This is additional practical project and it **is not mandatory and it is not included in the final score**. The main purpose is to use gained knowledge in different type of problems and to improve your portfolio and GitHub skills.

Today we will make the console game "`Rock` – `Paper` – `Scissors`":



Rock-Paper-Scissors is a simple **two-player game** where you and your opponent (the computer) simultaneously choose one of the following three options: "**rock**", "**paper**" or "**scissors**". The rules are as follows:

- **Rock beats scissors** (the scissors get broken by the rock)
- **Scissors beats paper** (the paper gets cut by the scissors)
- **Paper beats rock** (the paper covers the rock)

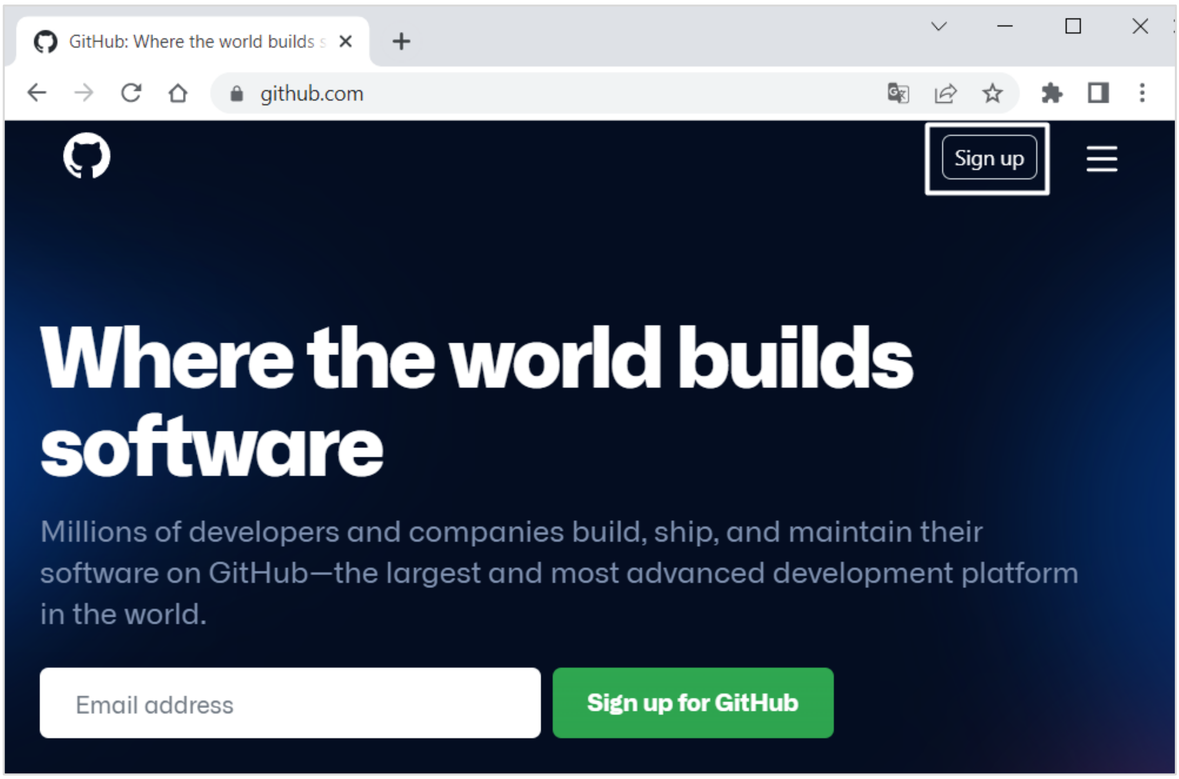
The **winner** is the player whose choice beats the choice of his opponent. If both players choose the same option (e.g., "paper"), the game outcome is "**draw**":

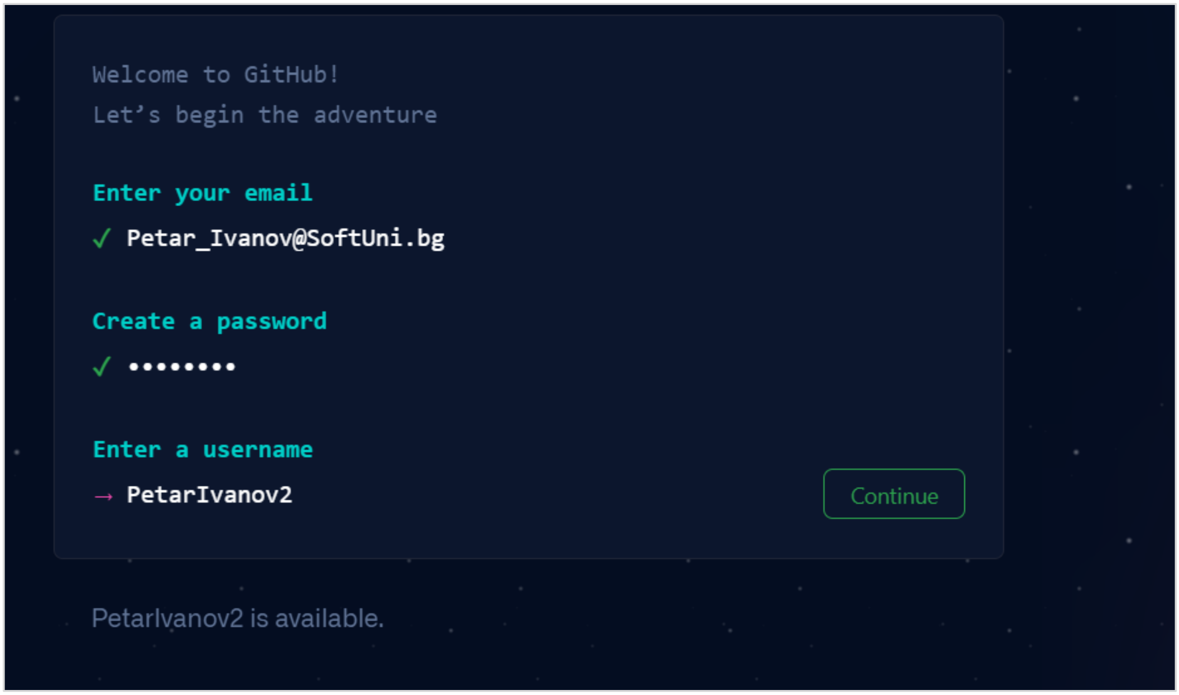
1. Create a GitHub Profile and Repo

Everyone should have a GitHub developer profile. First, we should **create our profile on GitHub**.

Register a GitHub Profile

Register for a free **developer account at GitHub** here: http://github.com. With an email and a username:


GitHub: Where the world builds s
github.com
Sign up
Where the world builds software
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.
Email address
Sign up for GitHub



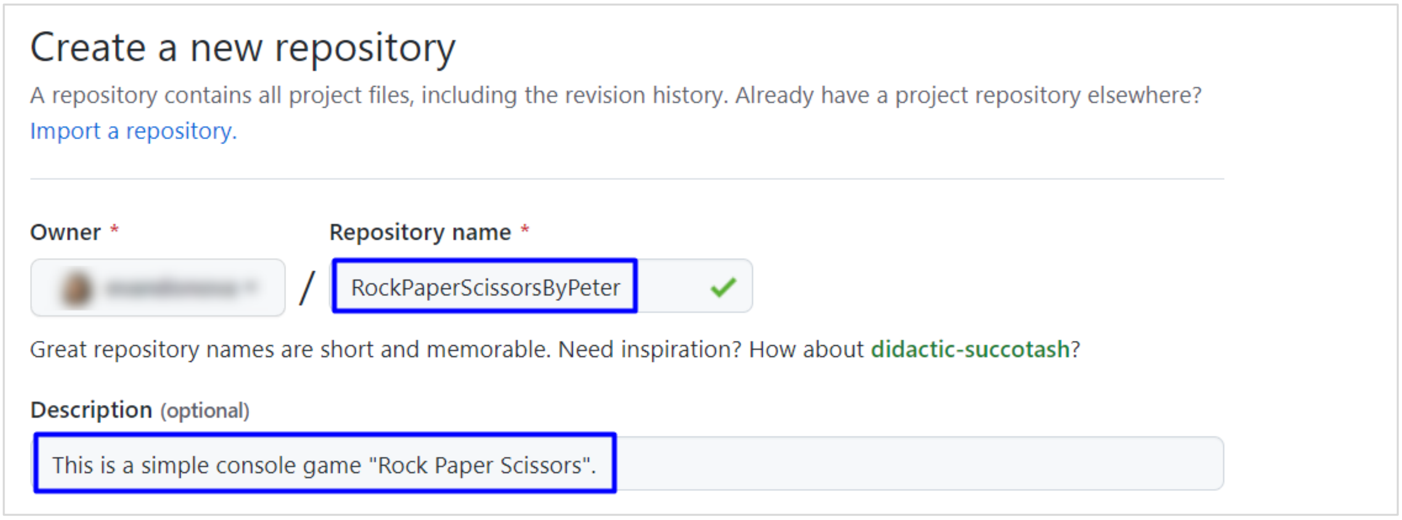
```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ Petar_Ivanov@SoftUni.bg

Create a password
✓ ••••••••

Enter a username
→ PetarIvanov2
```

PetarIvanov2 is available.

When you are ready, it is time to **create your first repository**. A **repository** contains **all of your project's files** and each file's revision history. You can discuss and manage your project's work within the repository.

Create a GitHub Repo

Create a **new repository** from: https://github.com/new. Choose a **meaningful name**, e. g. "**`RockPaperScissorsByUsername`**" add a **short description** and make your repo **public**:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * / Repository name *

RockPaperScissorsByPeter

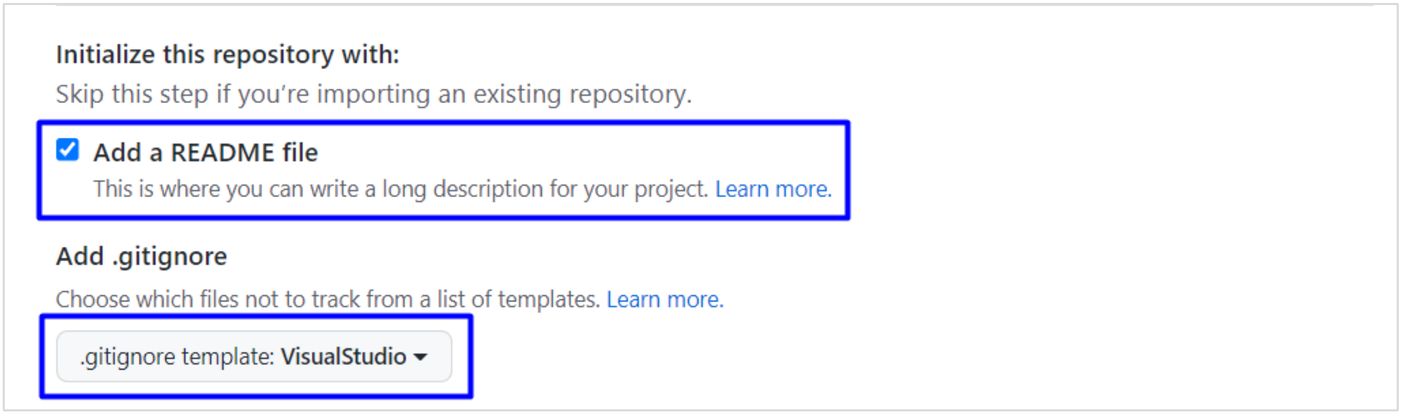
Great repository names are short and memorable. Need inspiration? How about didactic-succotash?

Description (optional)

This is a simple console game "Rock Paper Scissors".

⚠	Please choose **your original and unique name** for your project! Your GitHub profile should be **unique**, not the same as your classmates. You can follow this tutorial, but you can also **make changes** and **implement your project differ** from your classmates.

Also, **add a `README.md`** file and **`.gitignore` for Visual Studio**, as shown below:



Initialize this repository with:

Skip this step if you're importing an existing repository.

✓ Add a README file

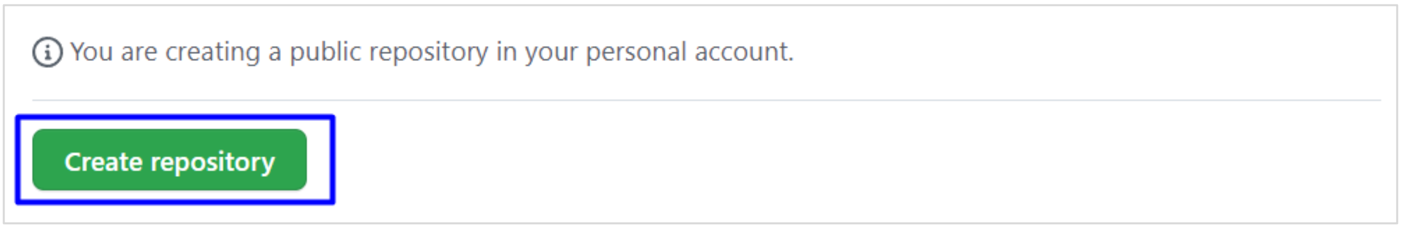
This is where you can write a long description for your project. Learn more.

Add .gitignore

Choose which files not to track from a list of templates. Learn more.

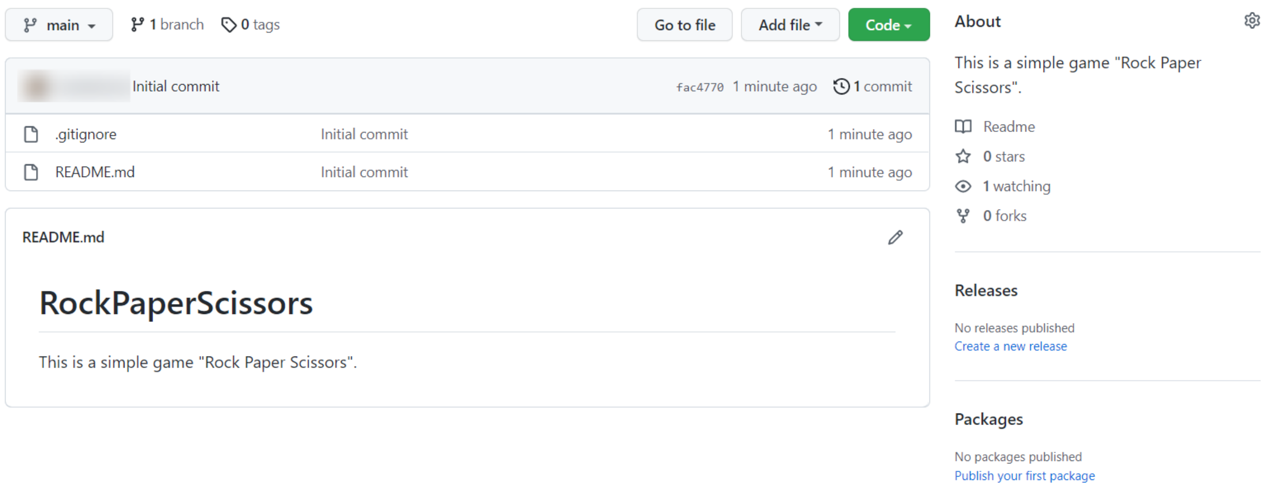
.gitignore template: VisualStudio

In Git projects, the **`.gitignore` file** specifies which files from your repo are not part of the source code and should be ignored (not uploaded in the GitHub repo). Typically in GitHub, we upload in the repo **only the source code,** and we don't upload the compiled binaries and temp files.



You are creating a public repository in your personal account.

Now your **repository is created** and looks like this:



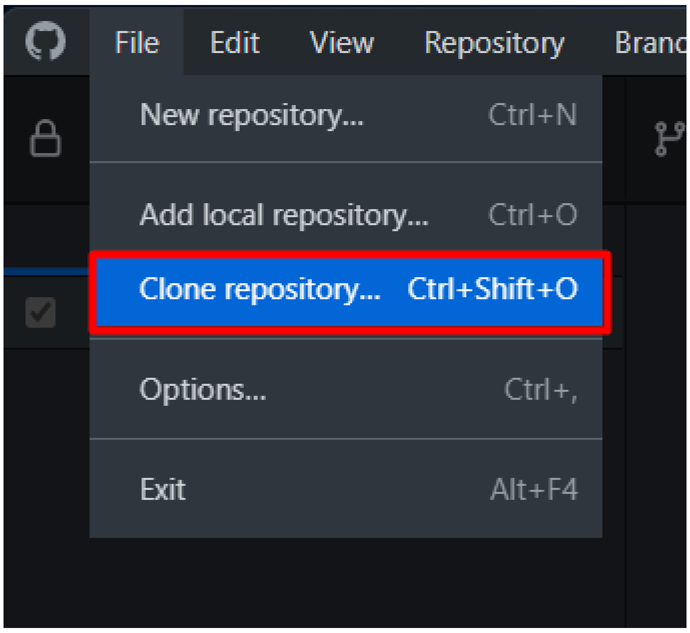
Now let's see how to **write the code** of our game.

2. Write the Game's Code

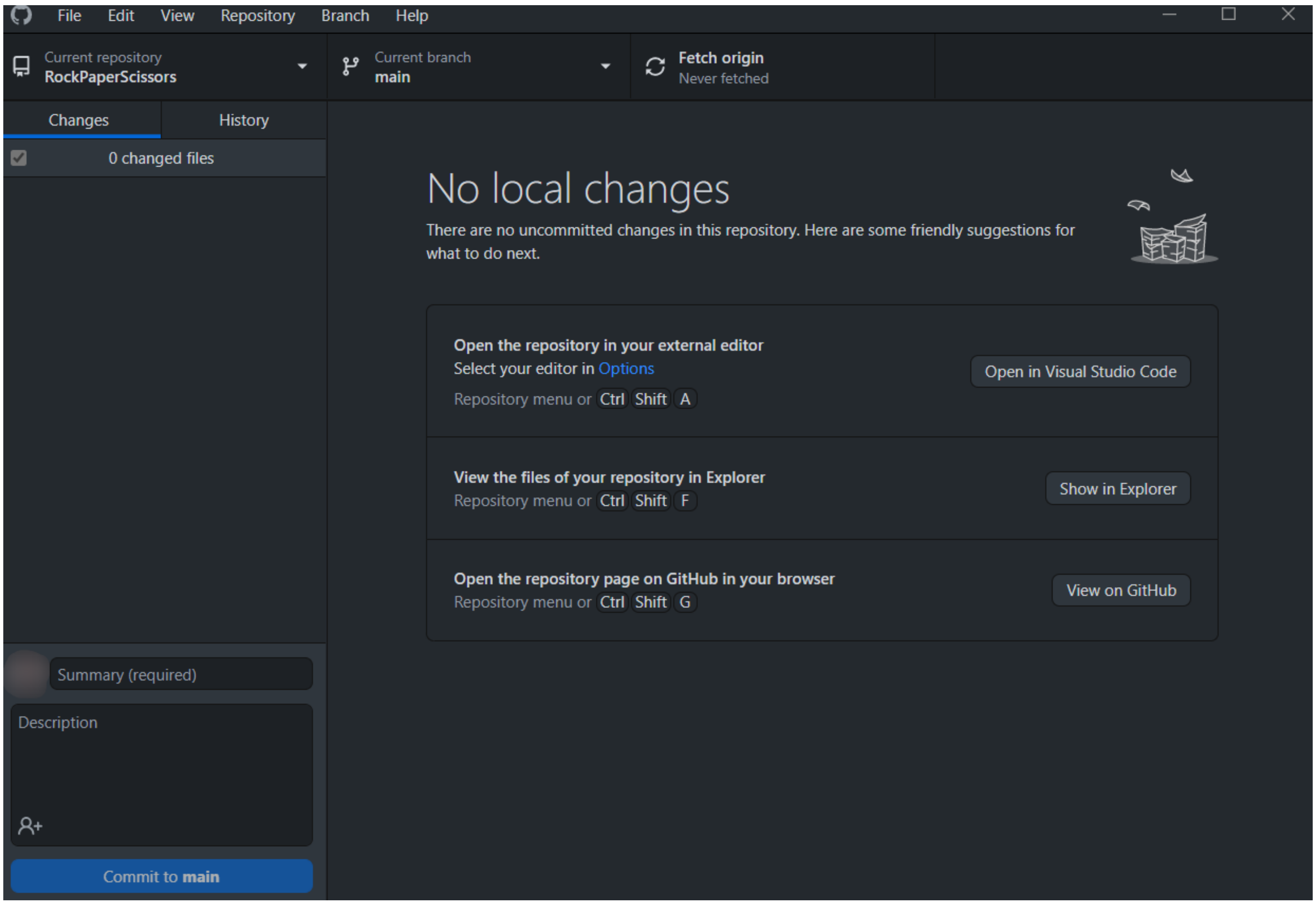
Let's create the game and play with it.

Create a Visual Studio Project

1) Open the folder from GitHub Desktop. If you don't have GitHub Desktop on your computer, download and install it from here: https://desktop.github.com/
2) Go to **"File"** and choose **"Clone repository".**



3) **Chose the repository** for the project, in our case, "RockPaperScissors" and hit the **"Clone"** button.



No local changes

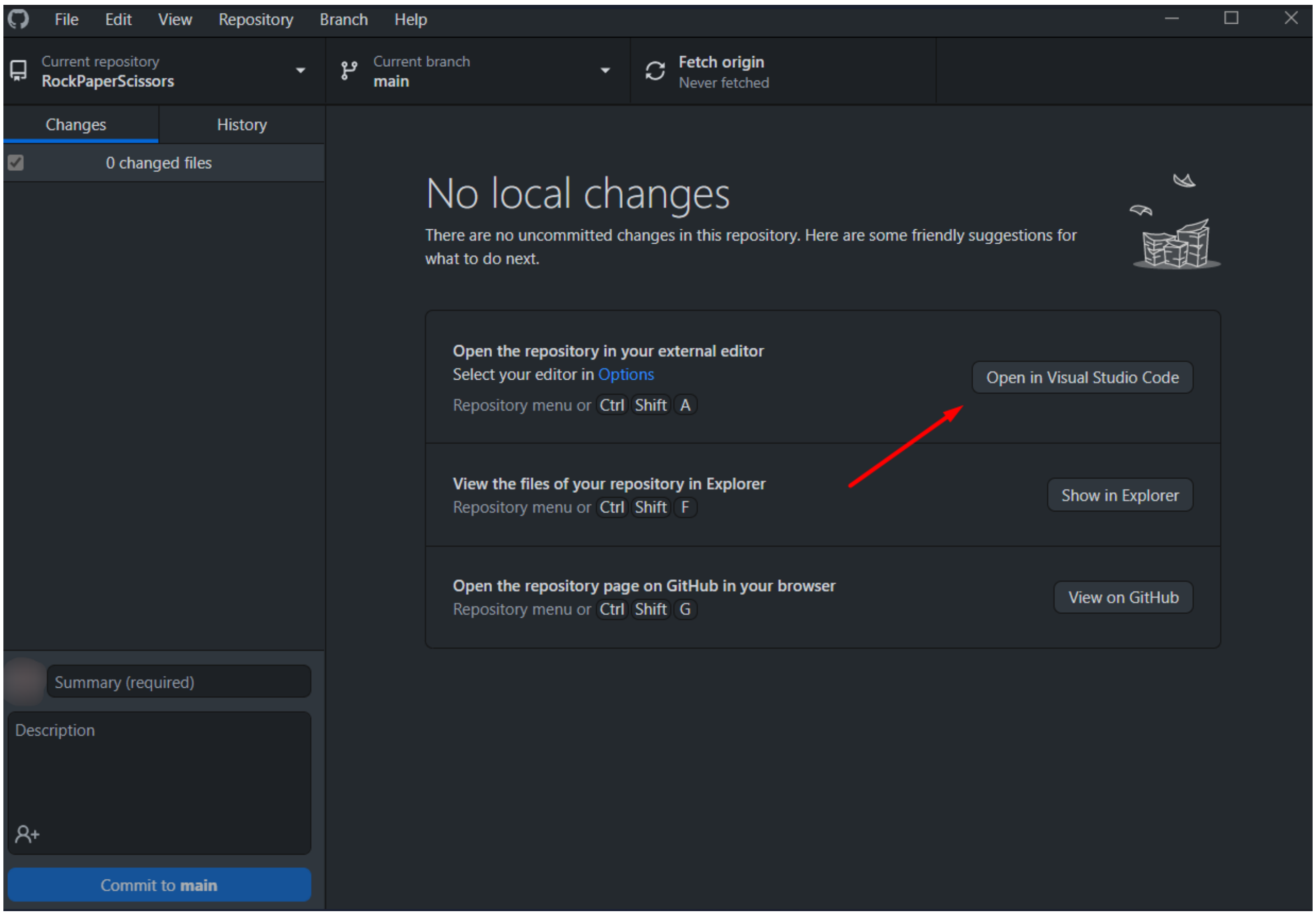
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A
Open in Visual Studio Code

View the files of your repository in Explorer
Repository menu or Ctrl Shift F
Show in Explorer

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G
View on GitHub

4) Click in GitHub Desktop on "Open in Visual Studio Code":



No local changes

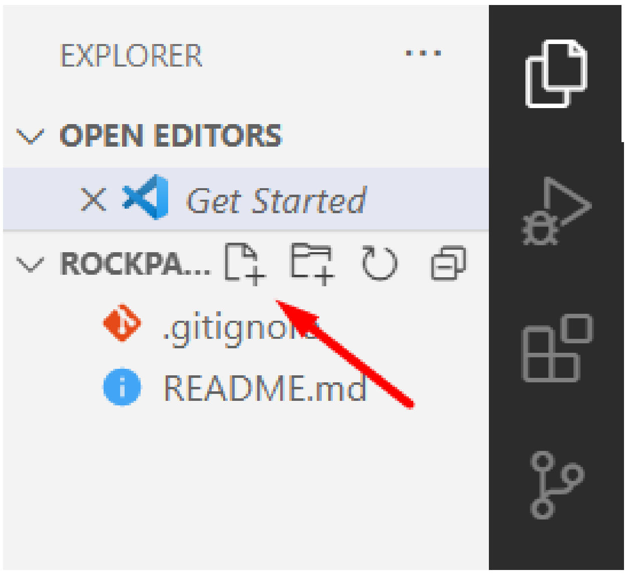
There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

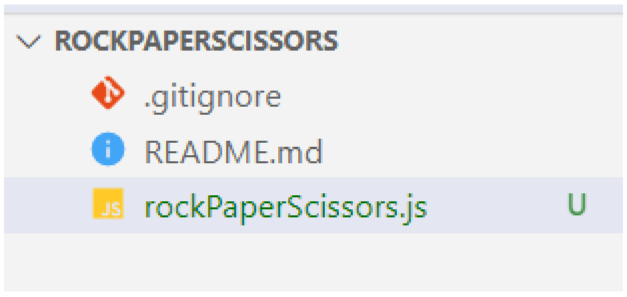
Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A
Open in Visual Studio Code

View the files of your repository in Explorer
Repository menu or Ctrl Shift F
Show in Explorer

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G
View on GitHub

5) When Visual Studio Code is open, click to create a new file





Implement the Game Logic

Read Player's Move

Now let's start working on our code.

Create **three constants** for our "**Rock**", "**Paper**" and "**Scissors**", which we will use later. **Constants** are values that **do not change** for the life of the program. They should look like this:

```
const rock = "Rock";
const paper = "Paper";
const scissors = "Scissors";
```

Match Player's Move with Possible Options

Now it is time to turn the user input into one of our **player's movement options**. To do this, create an **`if-else`** statement with the **possible moves** and change the variable value with our **`constants`**.

First, if the user has entered **"r"** or **"rock"**, then they chose **"Rock"**. Write it like this:

```
if (playerTurn == "r" || playerTurn == "rock") {
    playerTurn = rock;
}
```

And if they entered **"p"** or **"s"**, then they chose **"paper"** or **"scissors"** accordingly. Write the **`else-if`** statements by yourself:

```
else if (playerMove == "p" || playerMove == "paper")
{
    playerMove = Paper;
}
else if (playerMove == "s" || playerMove == "scissors")
{
    playerMove = Scissors;
}
```

Now we should cover the case where the user enters an **invalid value**. To do this, use **`else`** and **print** a message on the console and **stop the program execution**:

```
} else {
    console.log("Invalid Input. Try Again...");
}
```

Now let's **run** the app in the **console** and check whether our current code **works properly**. At the moment, we have **logic** only for the **incorrect input,** so the results should be as follow:

```
C:\Program Files\nodejs\node.exe .\rockPaperScissors.js
Invalid Input. Try Again...
```

Choose Computer's Move

Then, **use the method** "**`random`**", which will help us **choose a random number**. We will use this **number** so that the computer can randomly select from "**`rock`**", "**`paper`**" or "**`scissors`**":

```
let computerRandomNumber = Math.floor(Math.random() * 3) + 1;
```

You can learn a little more about it here:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

Choose the computer's **random move**, to make this happen, use the **conditional statements `switch-case`** or **`else-if`**. Also, check the **input of the player**, e. g.:

```
switch (computerRandomNumber)
{
    case 1:
        computerMove = Rock;
        break;
    case 2:
        computerMove = Paper;
        break;
    case 3:
        computerMove = Scissors;
        break;
}
```

Think about how you can complete these **conditional statements**.

Check and Write the Result

Write to the console what is the **random** selection of the computer. e. g. "**The computer chose {computerMove}.**". Now we need to **compare** the choice of the **player** and the **computer**, again using **conditional statements**.

```
console.log(`The computer chooses ${computerTurn}`);

if ((playerTurn === rock && computerTurn === scissors) ||
(playerTurn === paper && computerTurn === rock) || (playerTurn
=== scissors && computerTurn === paper)) {
    console.log("You win!");
}
```

You can use this table for the **possible moves**:

You	Computer	Outcome
rock	rock	Draw
rock	paper	You lose
rock	scissors	You win
paper	rock	You win
paper	paper	Drow
paper	scissors	You lose
scissors	rock	You lose
scissors	paper	You win
scissors	scissors	drow

Consider all the cases where the player **loses,** or the result between them is **equal,** and write down the **conditional statements**. That's all it takes for the **game to work**.

```
else if () {
    console.log("You lose!");
}
else {
    console.log("This game was a draw!");
}
```

After you run it, the game should look like this:

```
C:\Program Files\nodejs\node.exe .\rockPaperScissors.js
You choose Paper
The computer chooses Rock
You win!
```

```
C:\Program Files\nodejs\node.exe .\rockPaperScissors.js
You choose Rock
The computer chooses Rock
This game was a draw!
```

```
C:\Program Files\nodejs\node.exe .\rockPaperScissors.js
You choose Scissors
The computer chooses Rock
You lose!
```

3. Upload Your Project to GitHub

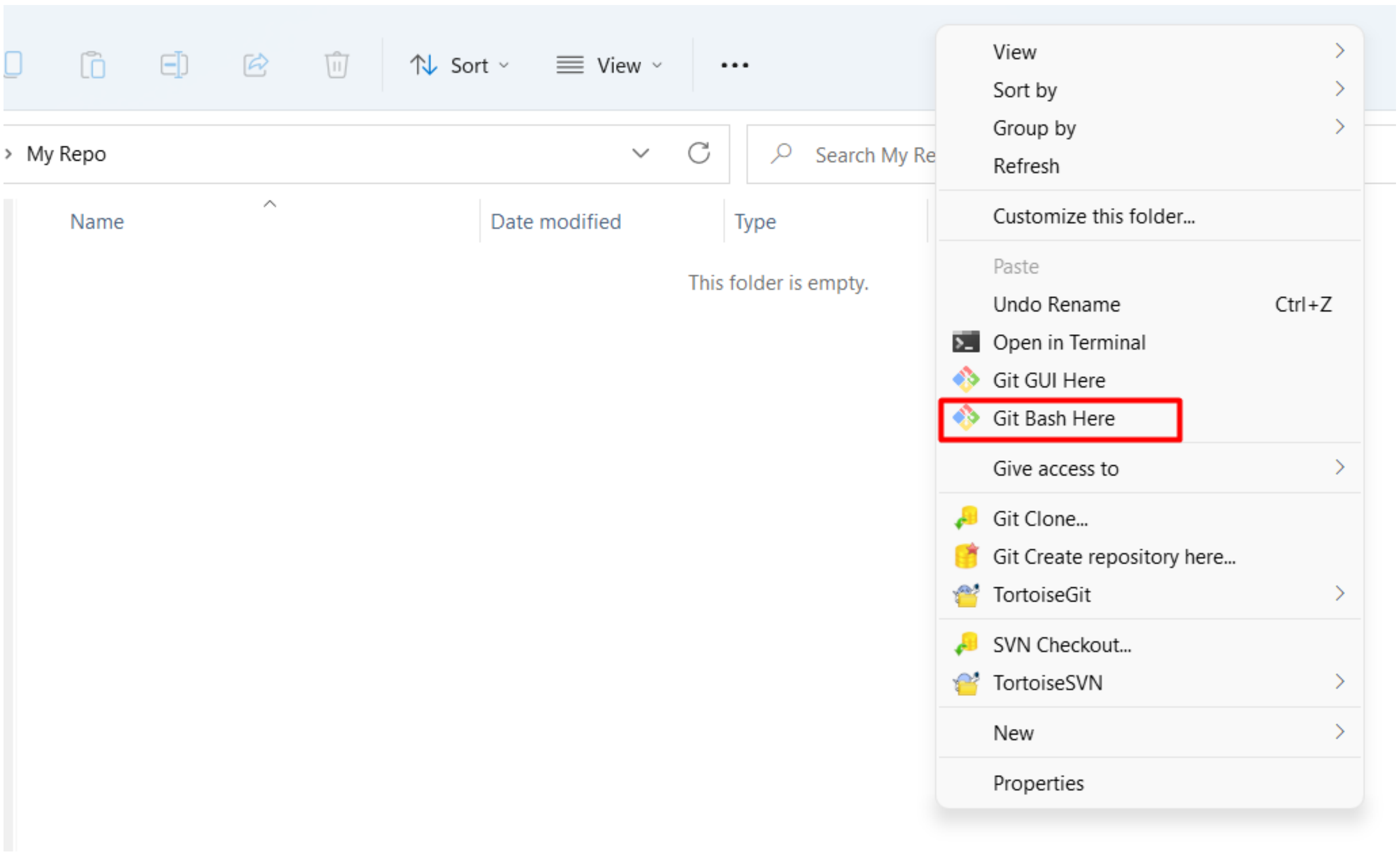
Now we want to deploy our project to **GitHub** so the other developers can see it, and if they want to test it, they can clone it and try it themself on their machine. You have **two options**, choose one and follow the steps.

Use Git Bash (Option 1)

If you don't use GitHub Desktop, you could use the "**Git Bash**" command line tool to upload your project to your **`GitHub`** repo.

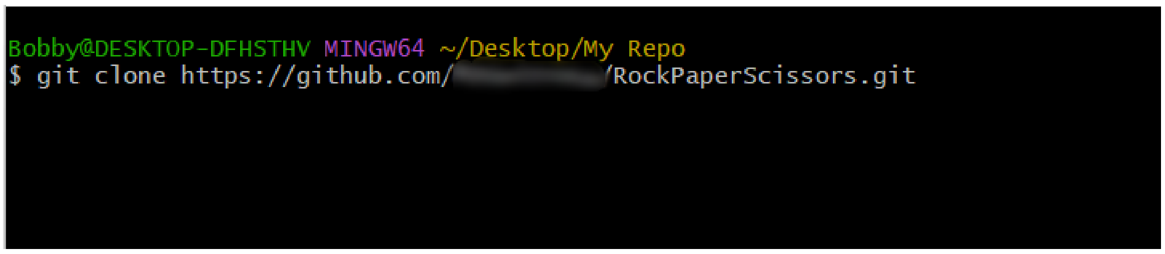
First, if you don't have **Git** on your **computer**, you should **install it** from https://git-scm.com/downloads.

Go to the desired **directory**, right-click on a blank space **anywhere** in the folder, and select "**Git Bash Here**" to open the Git command line console. If the "**Git Bash Here**" menu is missing, you should first install Git.



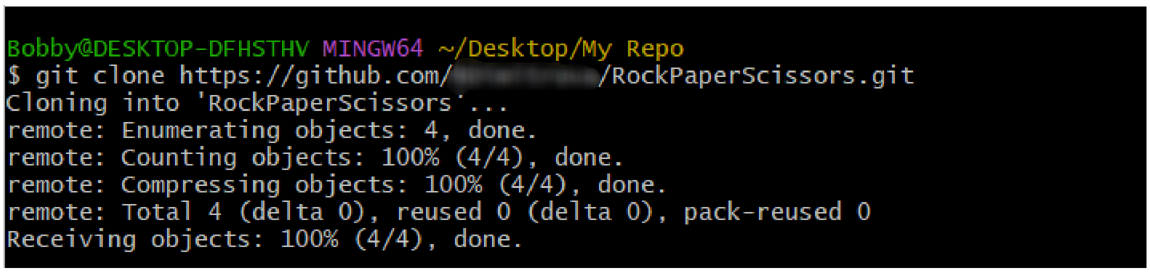
Type the "**`git clone`**" command followed by the link to your **repository**:

```
git clone
```



```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo
$ git clone https://github.com/[redacted]/RockPaperScissors.git
```

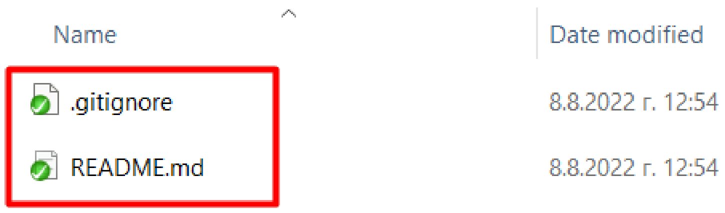
The result should be something like this:



```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo
$ git clone https://github.com/[redacted]/RockPaperScissors.git
Cloning into 'RockPaperScissors'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

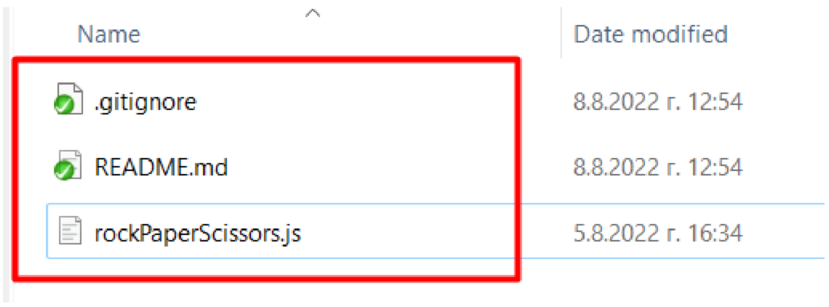
Your files from your GitHub repo will be downloaded to a **sub-folder** called your project in GitHub, "**RockPaperScissors**" in our case.

RockPaperScissors	8.8.2022 г. 12:54	File folder



Name	Date modified
.gitignore	8.8.2022 г. 12:54
README.md	8.8.2022 г. 12:54

The next thing to do is to **add** your **project files** to your **cloned repository folder**. It should look like this:



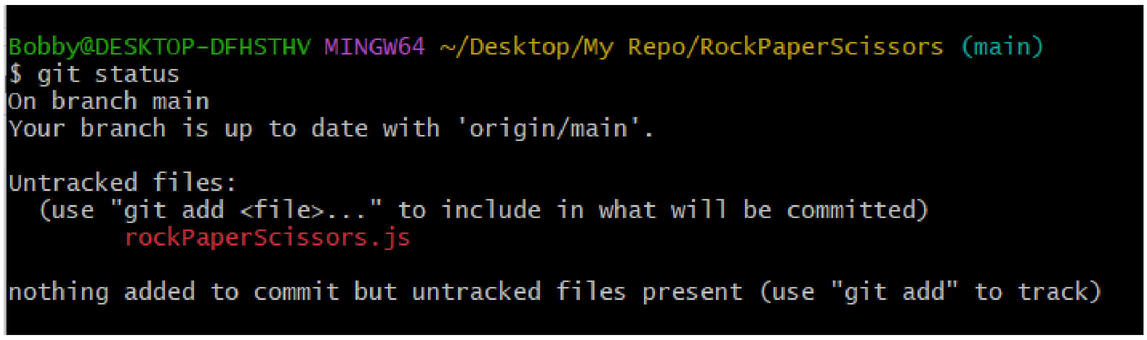
Name	Date modified
.gitignore	8.8.2022 г. 12:54
README.md	8.8.2022 г. 12:54
rockPaperScissors.js	5.8.2022 г. 16:34

Now we are ready to upload our changes from the "**Git Bash clone**". Go to the desired **folder**, right-click on a blank space anywhere in the folder, select "**`Git Bash Here`**" and run the following **commands**.

Type the following command:

```
git status
```

The **git status** command displays the state of the working directory and the **staging area**.

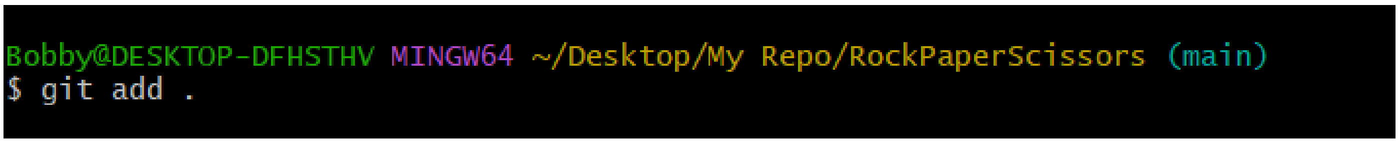


```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo/RockPaperScissors (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        rockPaperScissors.js

nothing added to commit but untracked files present (use "git add" to track)
```

Now type:

```
git add .
```

The above command **adds** all modified files to your local **Git repo**.

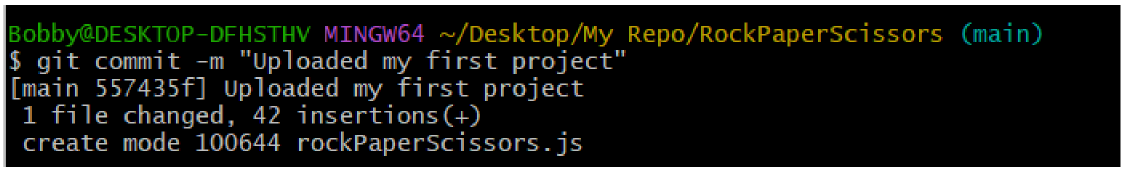


```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo/RockPaperScissors (main)
$ git add .
```

Now type:

```
git commit -m "Uploaded my first project."
```

This command **commits** your changes to your local **Git repo**. We also should **add** an appropriate **commit message**.

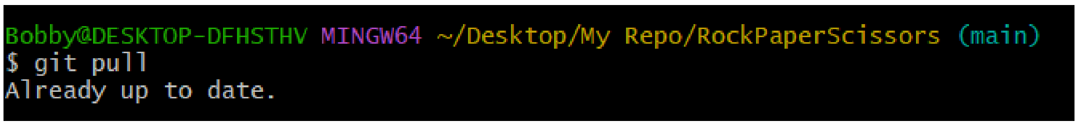


```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo/RockPaperScissors (main)
$ git commit -m "Uploaded my first project"
[main 557435f] Uploaded my first project
 1 file changed, 42 insertions(+)
 create mode 100644 rockPaperScissors.js
```

We have **two** more **commands** left. Second to the last type.

```
git pull
```

This command **updates** your local **repository** from GitHub. It downloads the latest project version from GitHub and merges it with your local copy.

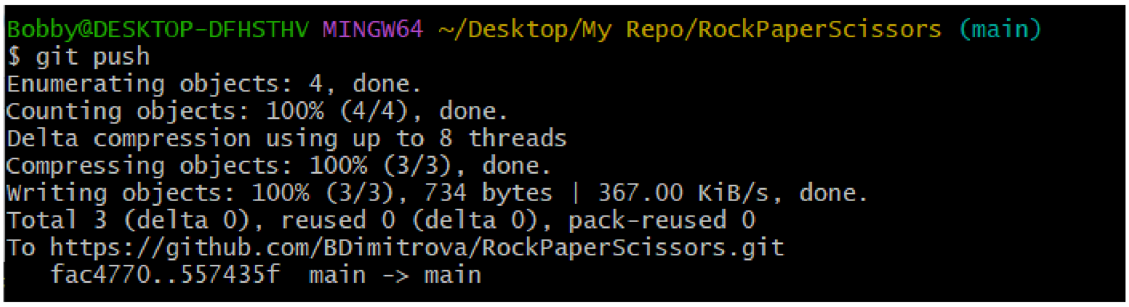


```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo/RockPaperScissors (main)
$ git pull
Already up to date.
```

Now the last thing that we should do is to **push** our changes by using the command.

```
git push
```

This command **pushes your local changes to GitHub**.



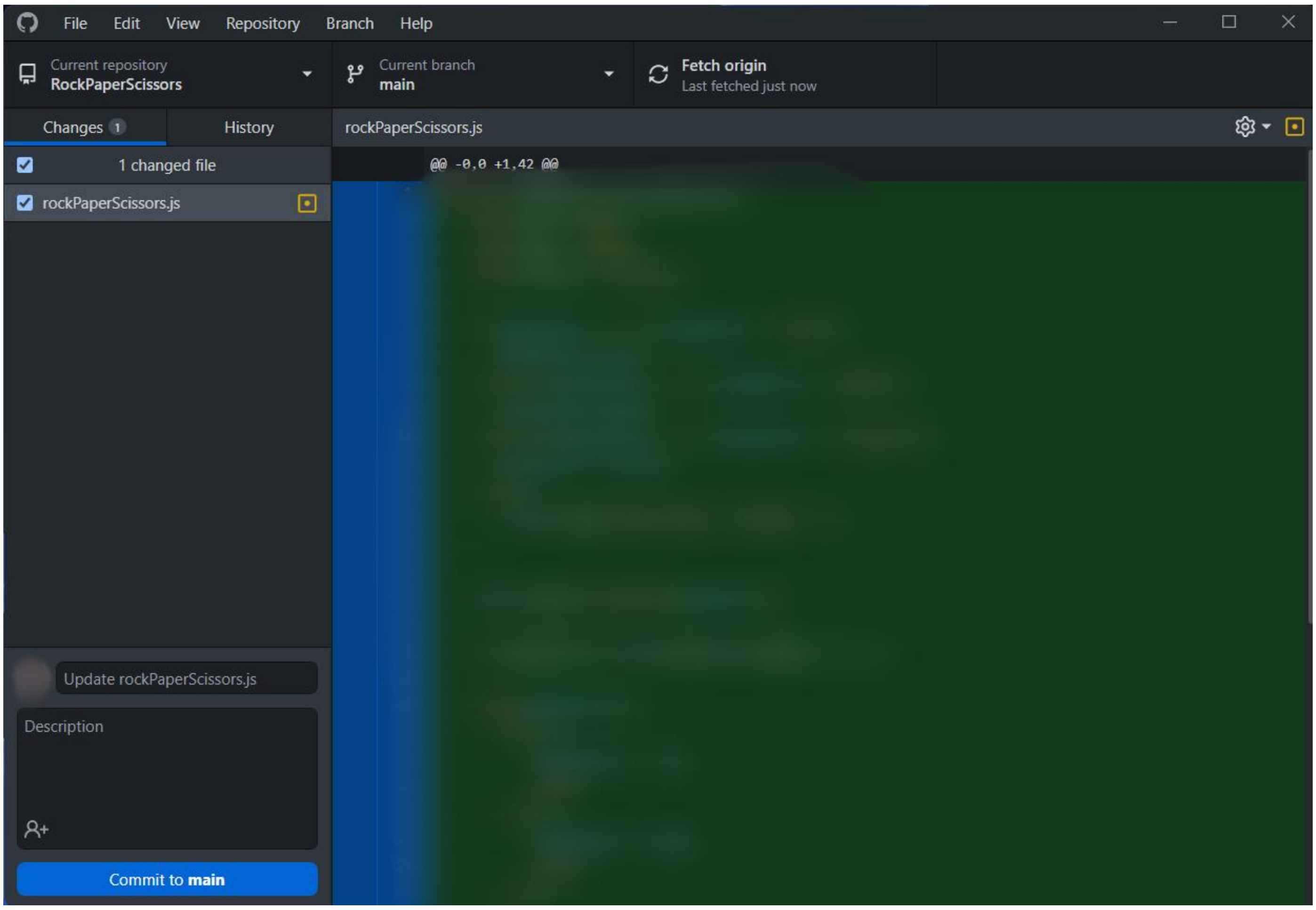
```
Bobby@DESKTOP-DFHSTHV MINGW64 ~/Desktop/My Repo/RockPaperScissors (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 734 bytes | 367.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/BDimitrova/RockPaperScissors.git
   fac4770..557435f  main -> main
```

This is all you need to **update** your **repository** using **`Git Bash`**.

A little more information about Git Bash: https://git-scm.com/about.

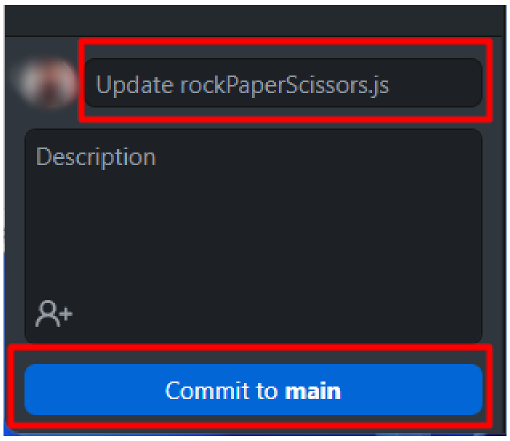
Use GitHub Desktop (Option 2)

1) If you use GitHub Desktop from the start, after you are done with your project, your GitHub Desktop will look like this:

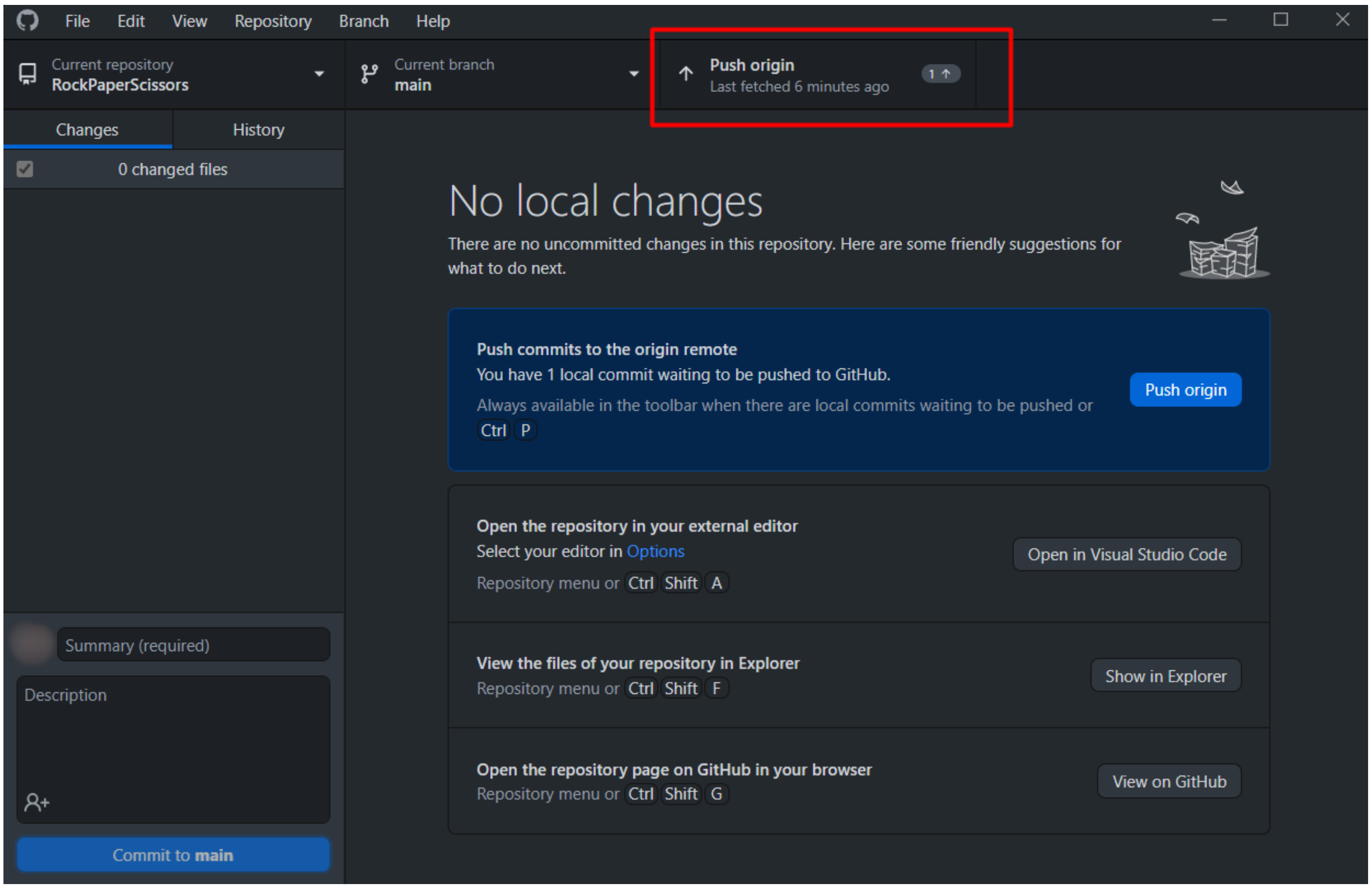


```
@@ -0,0 +1,42 @@
```

2) **Create a commit**, just like this.



Then **push the commit** to the repository.



No local changes

There are no uncommitted changes in this repository. Here are some friendly suggestions for what to do next.

Push commits to the origin remote
You have 1 local commit waiting to be pushed to GitHub.
Always available in the toolbar when there are local commits waiting to be pushed or Ctrl P

Open the repository in your external editor
Select your editor in Options
Repository menu or Ctrl Shift A

View the files of your repository in Explorer
Repository menu or Ctrl Shift F

Open the repository page on GitHub in your browser
Repository menu or Ctrl Shift G

This is all you need to **update** your **repository** using **GitHub Desktop.**

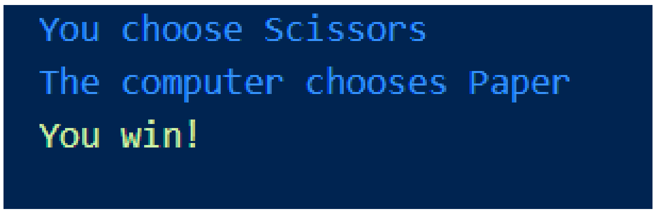
4. *Modify the Code, Write Your Features

⚠	This is your project. **Be unique**. Don't be a copy/paster! • Implement your **features**. • **Implement the code yourself**, using your coding style, code formatting, comments, etc. • Make the project **more interesting**. Learn by playing with the code and adding your changes.

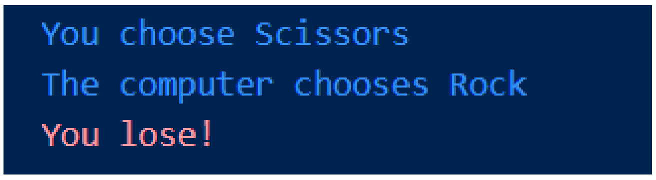
Below are a few **ideas** of what you can implement or modify in addition to your code.

Add Colors

You can modify the **text color** and **text background** in the console: https://blog.logrocket.com/using-console-colors-node-js/#implementing-console-colors-node-js-apps



```
You choose Scissors
The computer chooses Paper
You win!
```



```
You choose Scissors
The computer chooses Rock
You lose!
You choose Scissors
The computer chooses Scissors
This game was a draw!
```

Scoring System

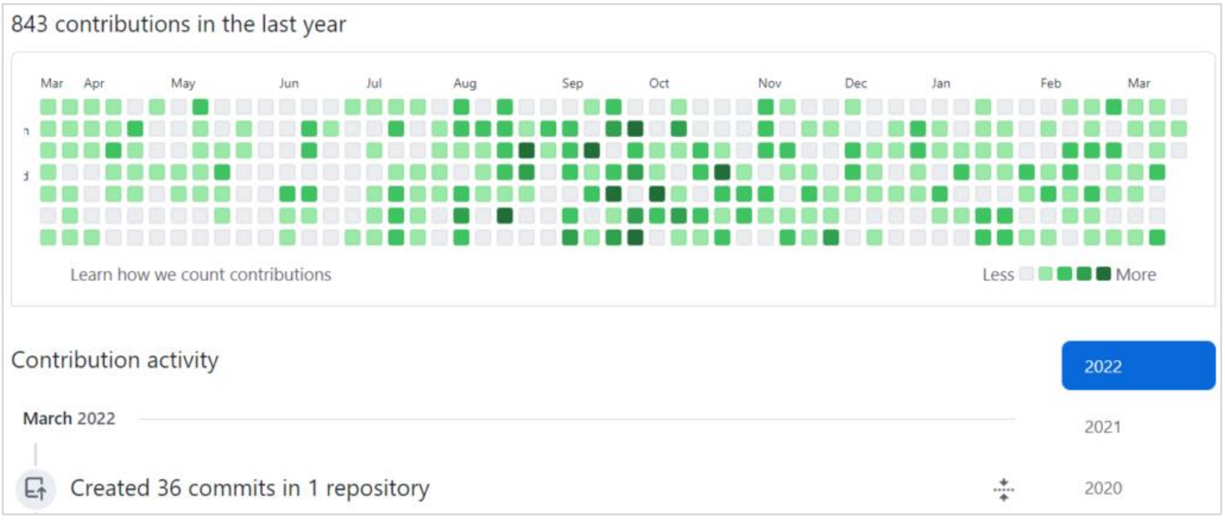
You can add a **scoring system** and display the player's and the computer's scores after each game session.

Additional Ideas

- Can you change your logic, so you can **increase the chances of the player winning**?
- Can you add **anything else** to your code based on your ideas?

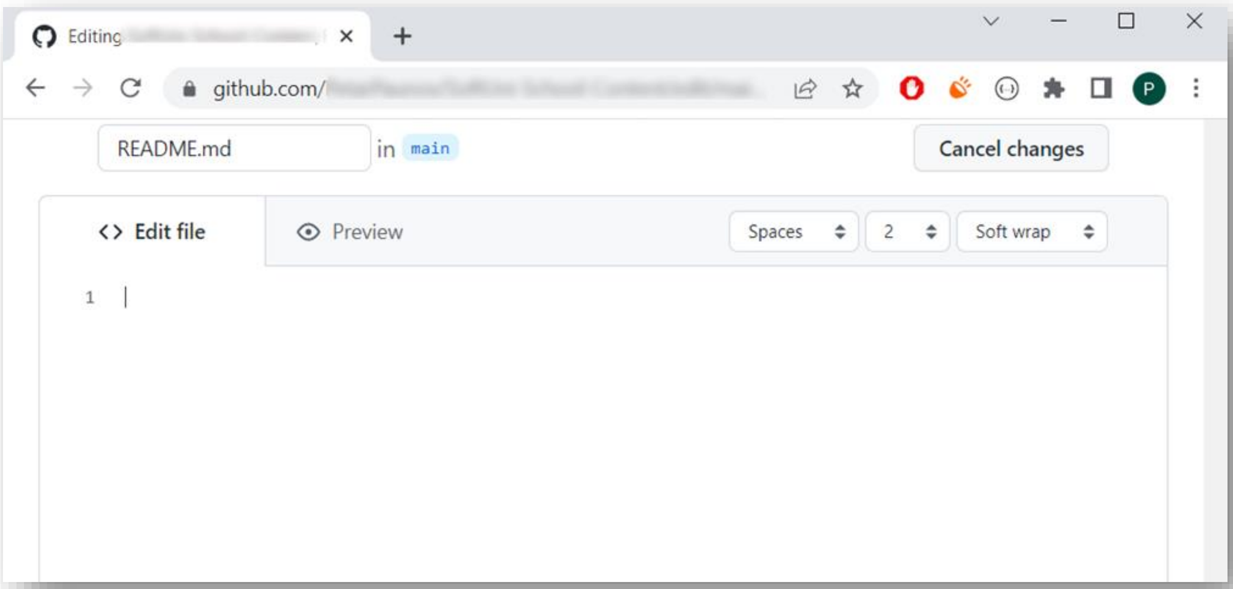
Commit to GitHub

Now **commit and push your code changes** to your GitHub repo!

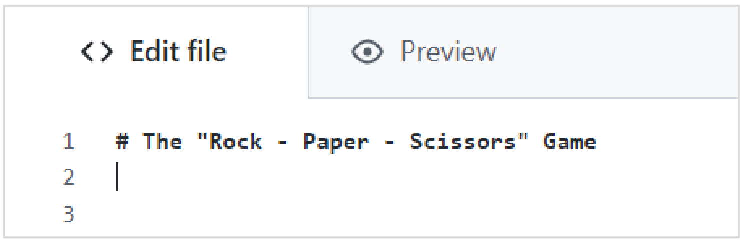
⚠	It is very important to **commit your code frequently** to GitHub. This way, you create a **rich commit history** for your project, and your **GitHub contribution graph** is growing:  843 contributions in the last year Contribution activity March 2022 Created 36 commits in 1 repository

5. Create a README.md File

It's highly recommended to provide **documentation as part of your project on GitHub** to describe what the project is **doing**. So, let's make one for this **project**. Let's start by editing the **`README.md`** file from our repo on GitHub:

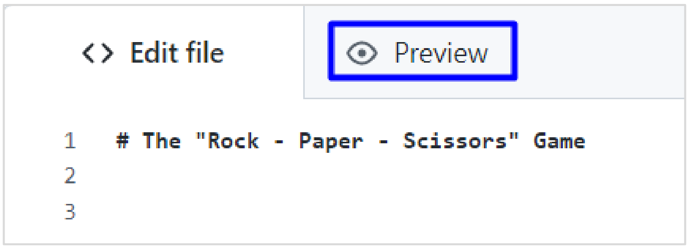


Add a project name. Use **"#"** in front of the text to indicate the **title**:

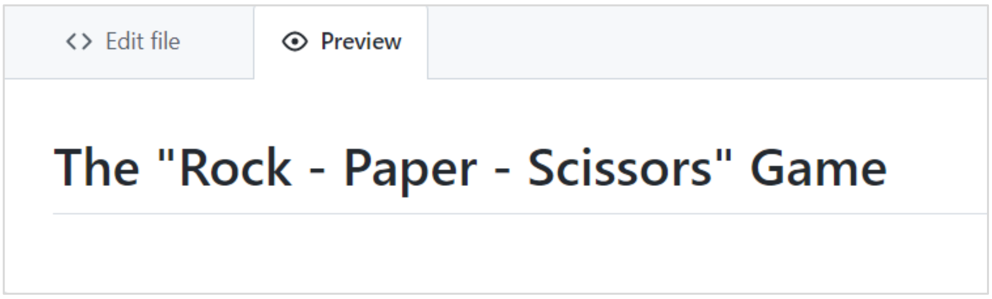


```
# The "Rock - Paper - Scissors" Game
```

You can **view** the current progress by pressing the **[Preview]** button:



```
# The "Rock - Paper - Scissors" Game
```



Documentation Sections

Add **information** about your project in your **README.md** file: project goals, technologies used, screenshots, live demo, etc. Typically, you should have the following **sections**:

- **Project title** (should answer the question "What's inside this project)
- **Project goals** (what problem do we solve, e. g., we implement a certain game)

- **Solution** (should describe how we solve the problem → **algorithms**, **technologies**, **libraries**, **frameworks**, **tools**, etc.)
- **Source code link** (give a direct link to your source code)
- **Screenshots** (add screenshots from your project in different scenarios of its usage)
- **Live demo** (add a one-click live demo of your code)

Use Markdown

Note that the GitHub **`README.md`** file is written in the **Markdown language**. Markdown combines text and special formatting tags to describe formatted text documents.

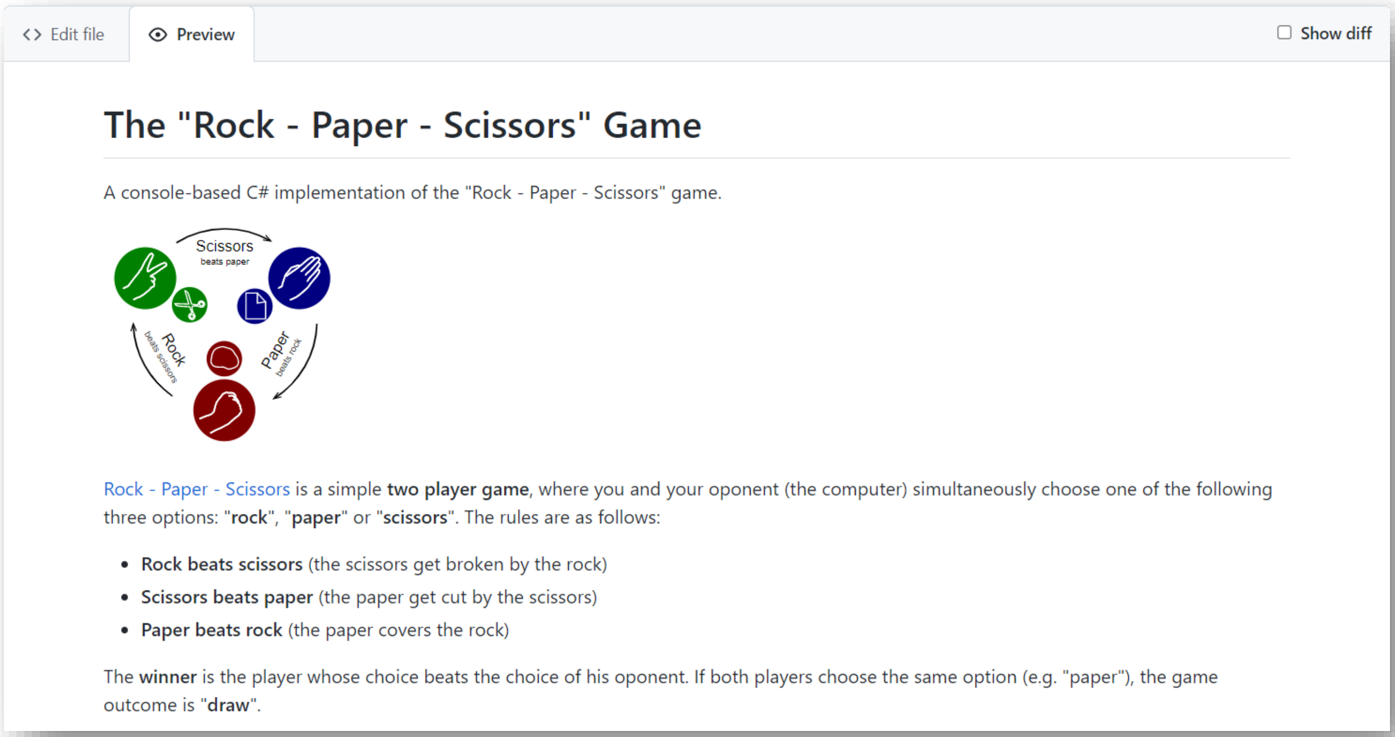
You can learn more about **Markdown** here: https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax.

Project Goals

Start your documentation by describing your **project goals**. What problem does your project solve?

Sample Documentation

This is an **example** of how you can document your project. Don't copy-paste it!



The "Rock - Paper - Scissors" Game

A console-based C# implementation of the "Rock - Paper - Scissors" game.

Rock - Paper - Scissors is a simple **two player game**, where you and your oponent (the computer) simultaneously choose one of the following three options: "**rock**", "**paper**" or "**scissors**". The rules are as follows:

- **Rock beats scissors** (the scissors get broken by the rock)
- **Scissors beats paper** (the paper get cut by the scissors)
- **Paper beats rock** (the paper covers the rock)

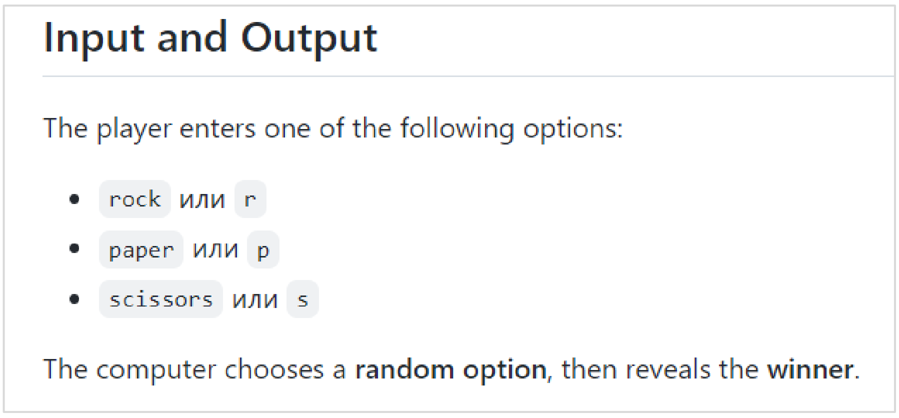
The **winner** is the player whose choice beats the choice of his oponent. If both players choose the same option (e.g. "paper"), the game outcome is "**draw**".

⚠	**Write the project documentation yourself**. Don't copy/paste it! This is your **unique GitHub profile** and your unique project. **Be different** from others.

You can add **appropriate images** to make your documentation better. You can add an **image** as follows:

```
<img alt="Image" width="200px" src="[redacted]" />
```

You can add information about the **inputs** and **outputs** of the project:


Input and Output

The player enters one of the following options:

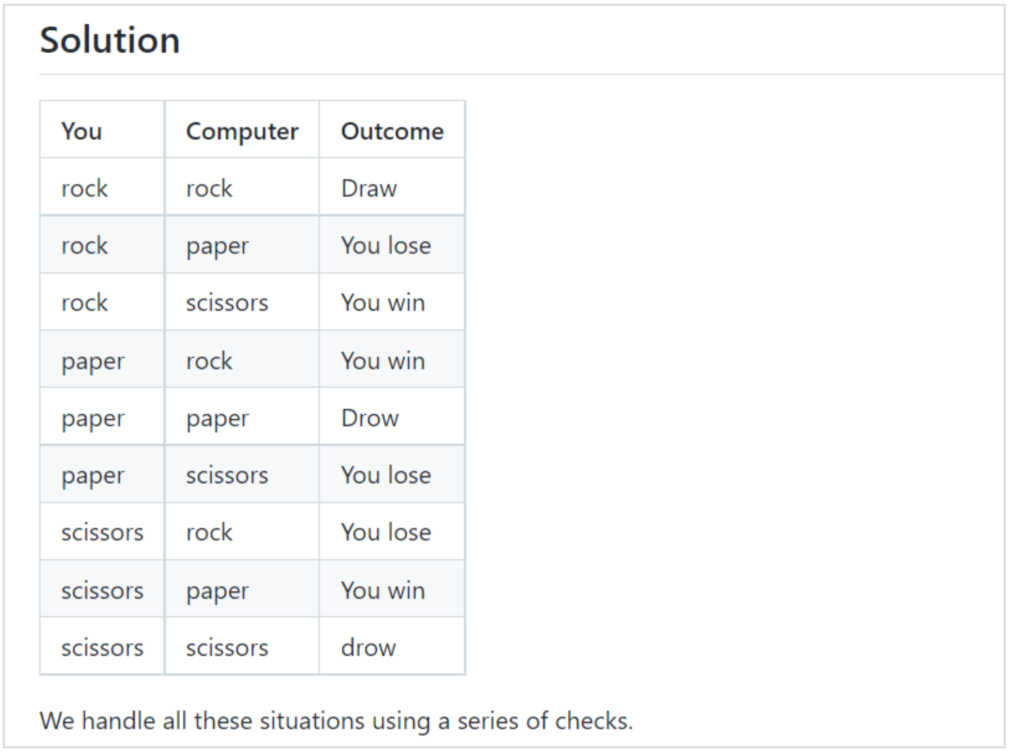
- `rock` или `r`
- `paper` или `p`
- `scissors` или `s`

The computer chooses a **random option**, then reveals the **winner**.

Your Solution

Describe how you **solve** the problem: **algorithms**, **technologies**, **libraries**, **frameworks**, **tools**, etc.

For example, for our simple game, you may analyze all possible game **situations** in a **table**:


Solution

You	Computer	Outcome
rock	rock	Draw
rock	paper	You lose
rock	scissors	You win
paper	rock	You win
paper	paper	Drow
paper	scissors	You lose
scissors	rock	You lose
scissors	paper	You win
scissors	scissors	drow

We handle all these situations using a series of checks.

Link to the Source Code

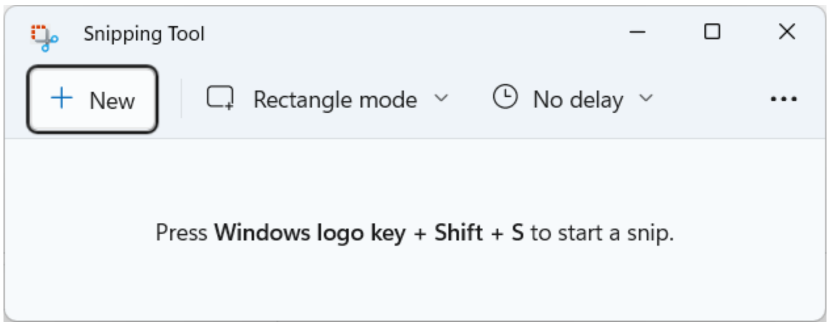
Add a **link** to your **source code** as follows:

[Source Code](rock_paper_scissors.py)

Screenshots

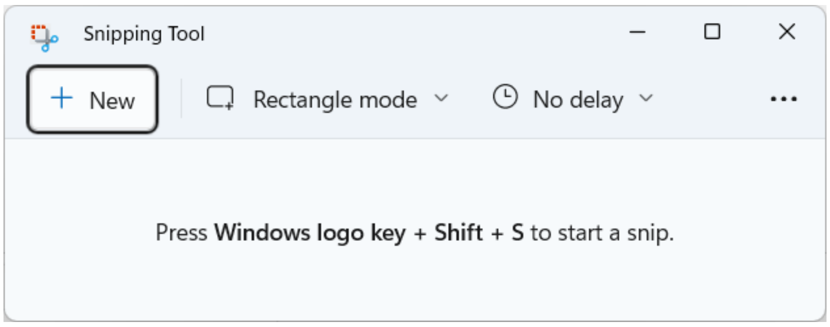
Add **screenshots** of your project:

1. **Take a screenshot** with your favorite tool (e.g., the Snipping Tool in Windows).



Press **Windows logo key** + **Shift** + **S** to start a snip.

2. **Paste** the screenshot in the GitHub Markdown editor using **`[Ctrl+V]`**:
3. **Take a screenshot** with your favorite tool (e.g., the Snipping Tool in Windows).



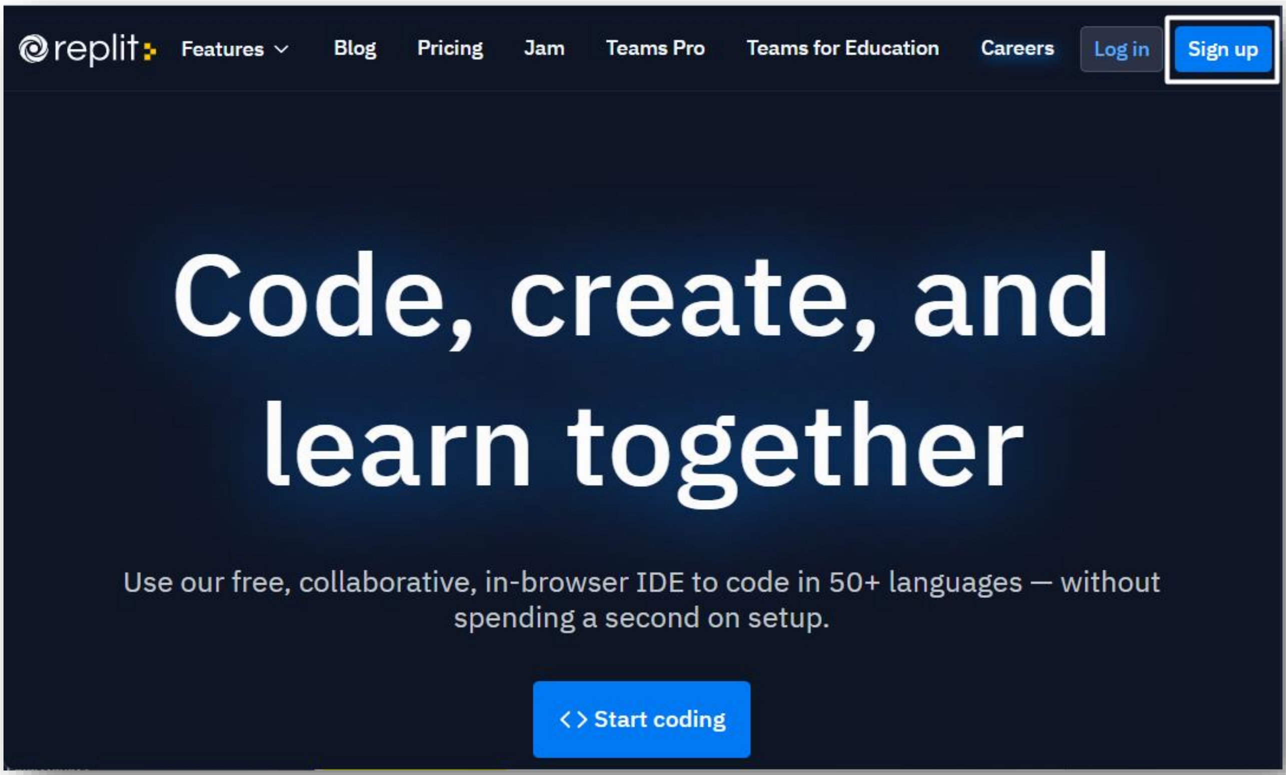
Press **Windows logo key** + **Shift** + **S** to start a snip.

4. **Paste** the screenshot in the GitHub Markdown editor using **`[Ctrl+V]`**:

6. Upload Your App to Replit

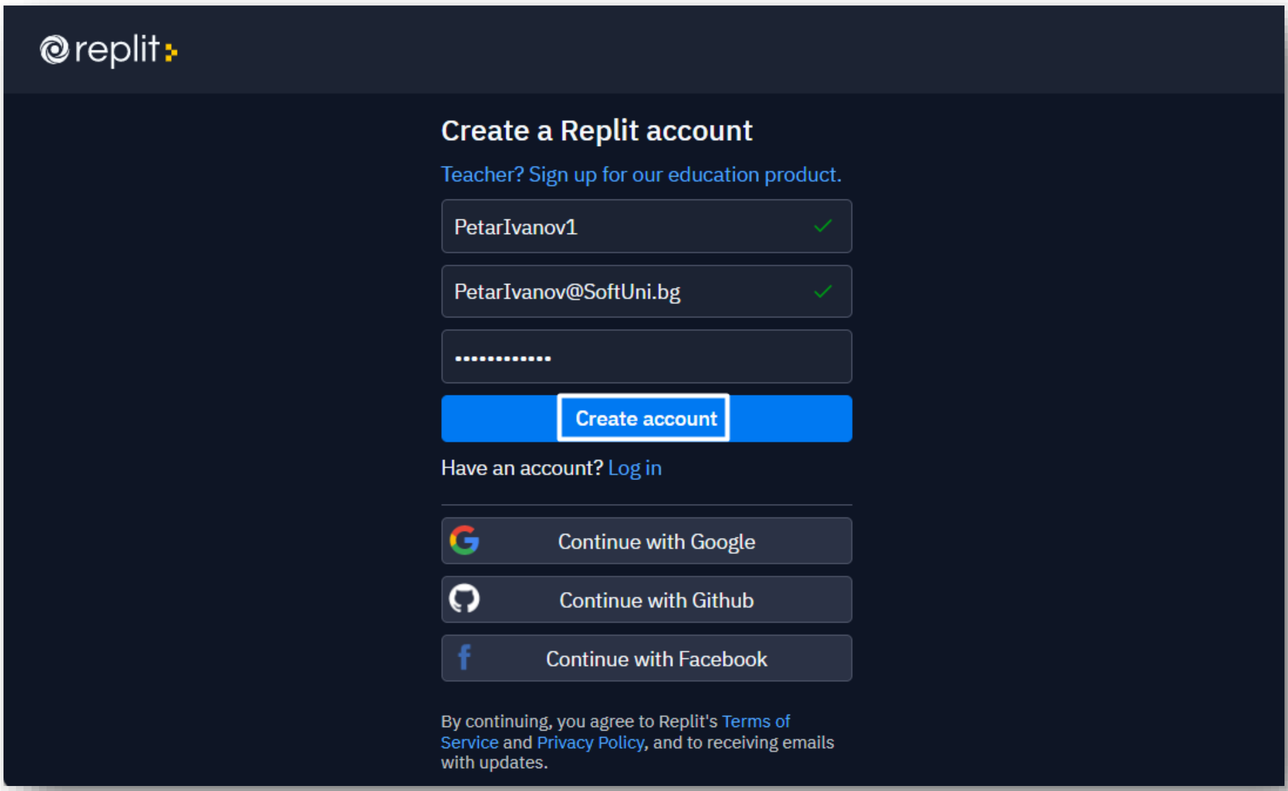
`Replit` is an online coding environment (online IDE) that allows you to **write** software projects, **share** them through a simple link, and **run** your projects directly in the Web browser. We shall upload our project in **`Replit`** to allow the users to **run and interact with the project** with just **one click**.

Create your **`Replit`** profile so you can show your **projects** to your friends and put "**live demo links**" in your **`GitHub`** project documentation. Create a **Replit** account for **free**: https://replit.com.



Code, create, and learn together

Use our free, collaborative, in-browser IDE to code in 50+ languages — without spending a second on setup.



Create a Replit account

Teacher? Sign up for our education product.

PetarIvanov1

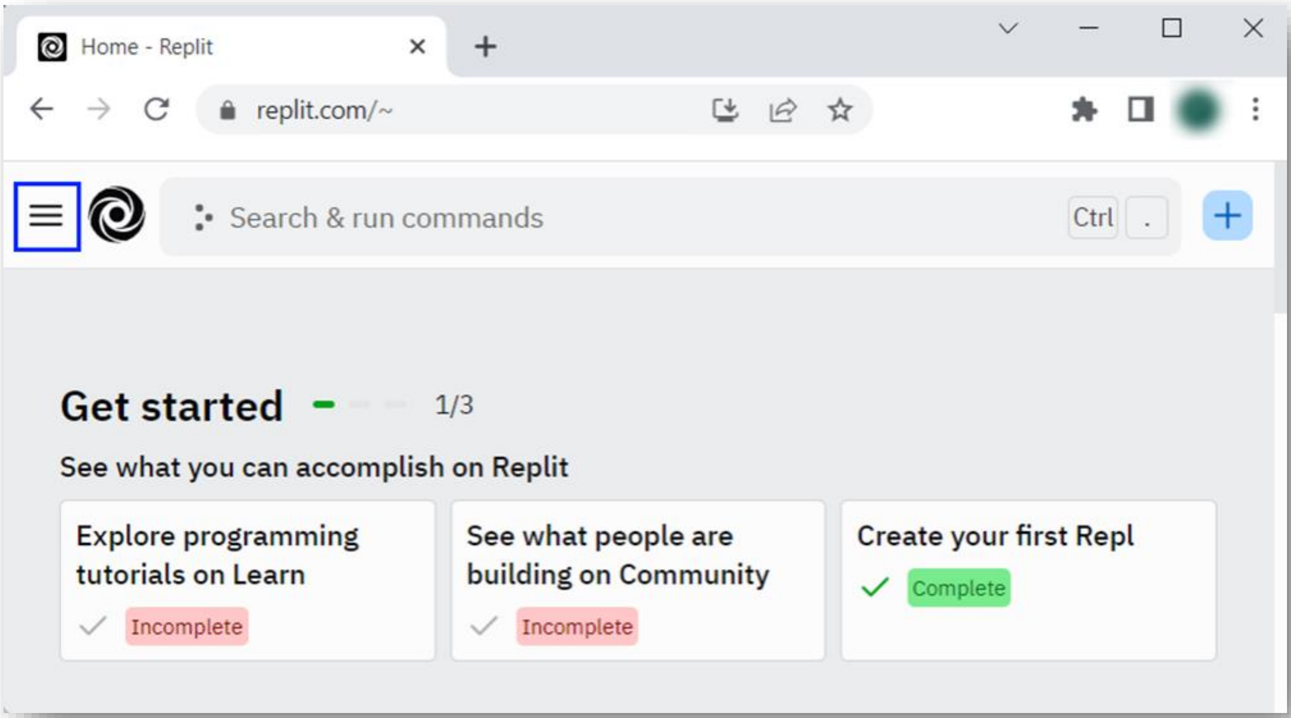
PetarIvanov@SoftUni.bg

•••••••••••••

Have an account? Log in

By continuing, you agree to Replit's Terms of Service and Privacy Policy, and to receiving emails with updates.

Create a **new project** in **`Replit`**, open the **menu** in the upper **left corner**.

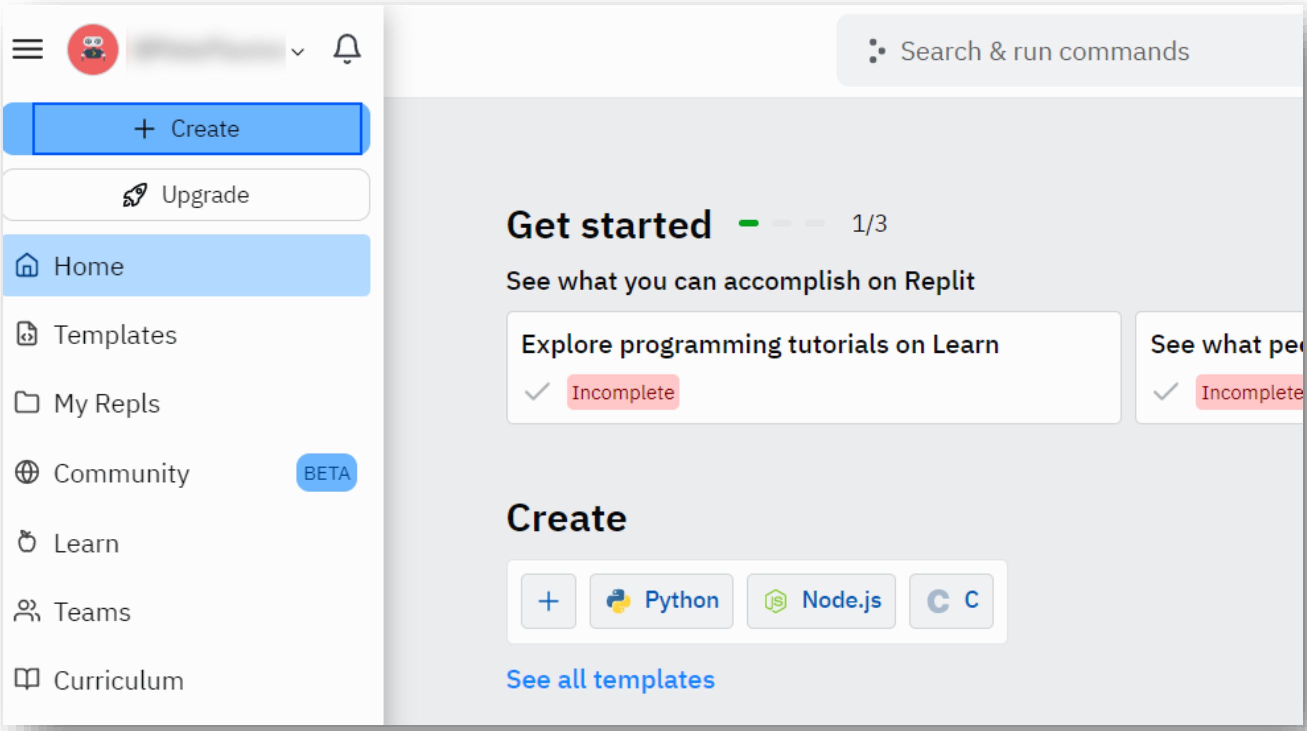


Get started 1/3

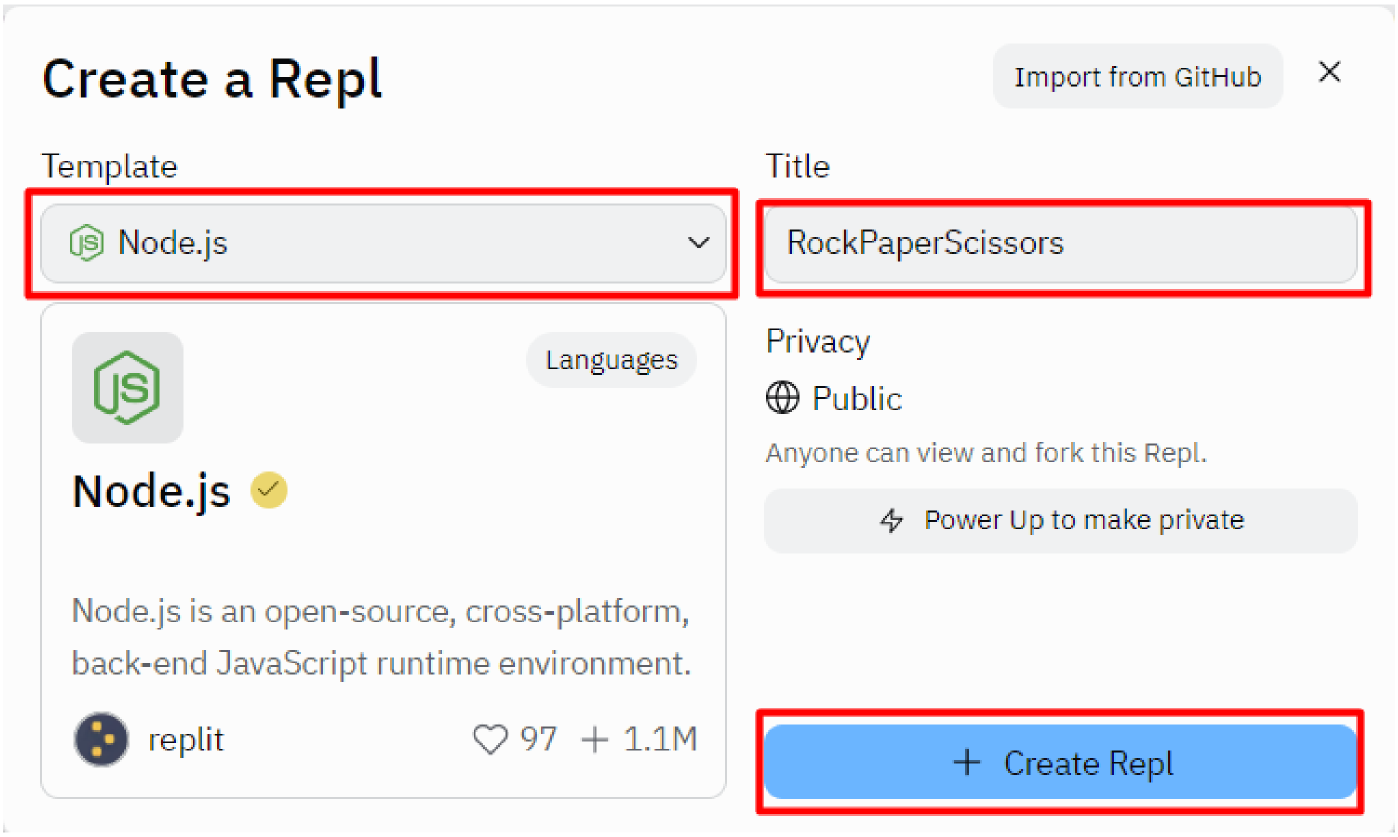
See what you can accomplish on Replit

Explore programming tutorials on Learn	See what people are building on Community	Create your first Repl
Incomplete	Incomplete	✓ Complete

Click **[Create]**, then select the **language** in which your project is **written**, select a name, and **create** the project.

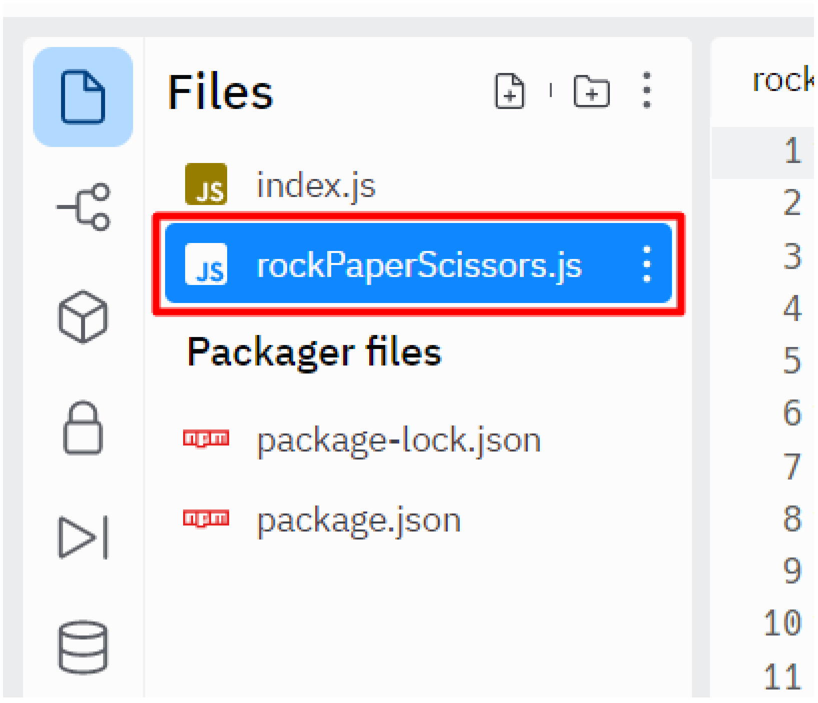


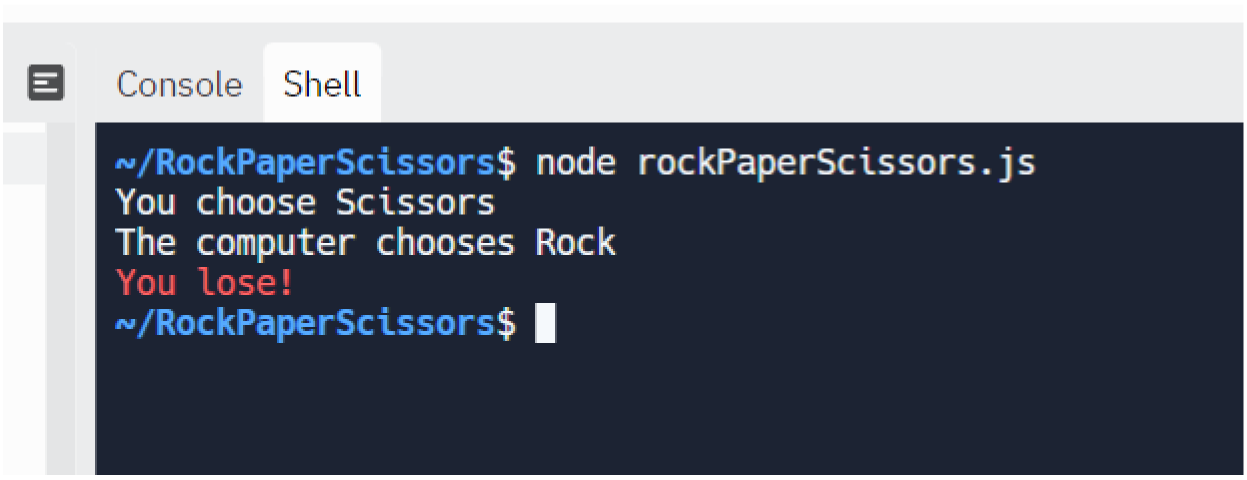
Chose "**`Node.js" for your project`**.



Add a meaningful **name** to your **`Replit`** project, e.g., "**RockPaperScissors** ".

Paste your code in the "**`rockPaperScissors.js`**" file:





```
~/RockPaperScissors$ node rockPaperScissors.js
You choose Scissors
The computer chooses Rock
You lose!
~/RockPaperScissors$
```

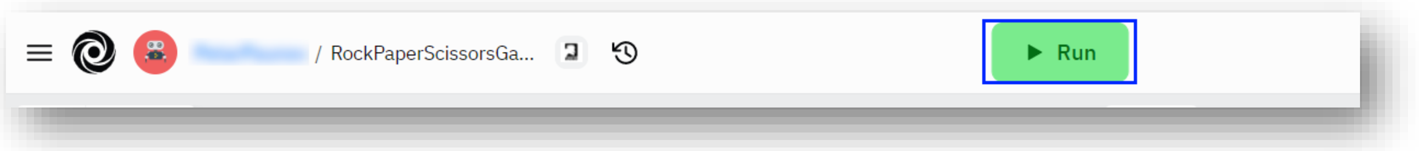
7. Add Replit Link to Your README.md

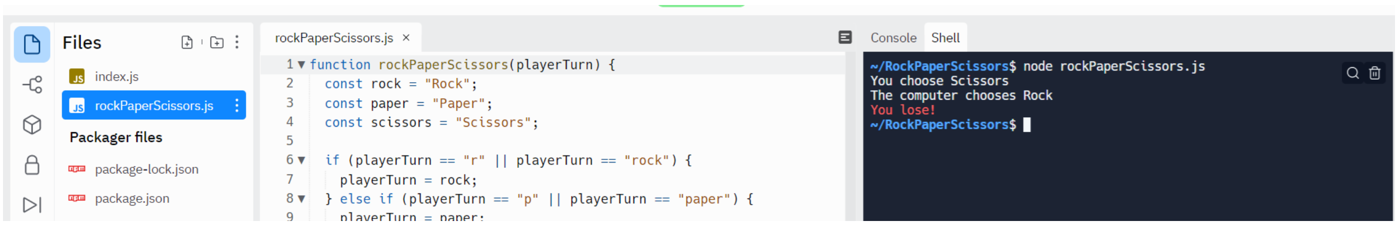
Now add a "**one-click live demo**" of your project from your GitHub project documentation. You can do it as follows:

```
## Live Demo

You can play the game directly in your Web browser here:

[<img alt="Play Button" src="https://user-images.githubusercontent.com/85368212/167706726-d027f056-fc2b-47b7-bfad-8ff8a3aa7688.png" />](https://replit.com/@PetarPaunov/Rock-Paper-Scissors-Game#Main.cs)
```

You can take a **screenshot** from Replit.com and **paste it** into the GitHub documentation editor directly with **[Ctrl+V]**.

When the [**Run**] button is clicked, you will be redirected to your demo in **`Replit`**.





```
function rockPaperScissors(playerTurn) {
  const rock = "Rock";
  const paper = "Paper";
  const scissors = "Scissors";

  if (playerTurn == "r" || playerTurn == "rock") {
    playerTurn = rock;
  } else if (playerTurn == "p" || playerTurn == "paper") {
    playerTurn = paper;
```

```
~/RockPaperScissors$ node rockPaperScissors.js
You choose Scissors
The computer chooses Rock
You lose!
~/RockPaperScissors$
```

Now we have completed our **first console game,** and we have our first project in our GitHub portfolio.