DS Projects

A client-server application that uses sockets

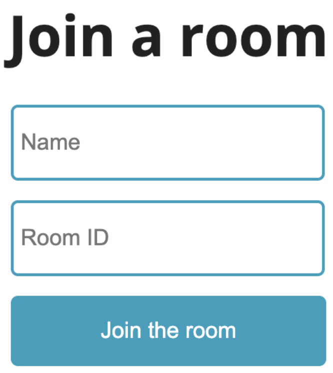
Technologies used for the client:
HTML, CSS, JavaScript (+ React library, socket.io-client package)

Technologies used for the server:
NodeJS (+ Express framework, socket.io package)

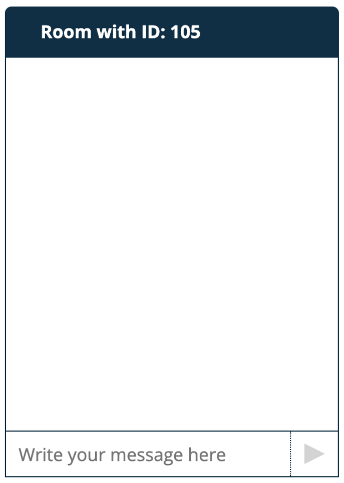
How to start the application in the development environment:
(Note: The server should be started first.)
Navigate to the /server folder and run the following command in the terminal: 'npm run start'.
Navigate to the /client folder and run the following command in the terminal: 'npm start'.

The client has 2 states:

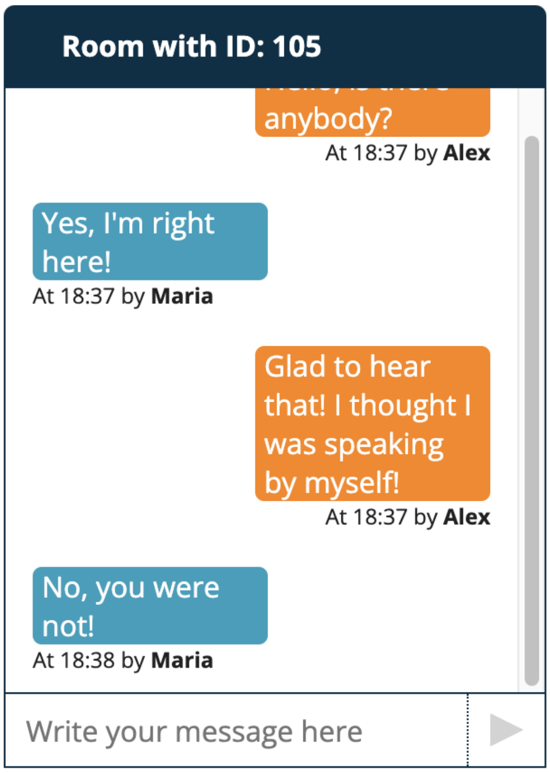
1. Before you join a chat room (with a name and a room ID)



2. After you are in the chat room

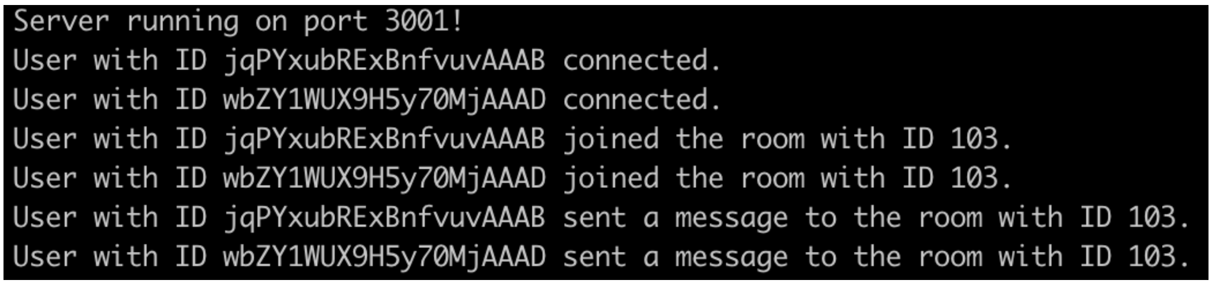


The chat provides additional information like the message author's name and the time it was sent:



Plus other UX like hitting the 'Enter' button in the input will send the message or automatically scrolling to the bottom of the chat.

The person with access to the running server can see all the activities made by the users using the sockets:



```
Server running on port 3001!
User with ID jqPYxubRExBnfvuvAAAB connected.
User with ID wbZY1WUX9H5y70MjAAAD connected.
User with ID jqPYxubRExBnfvuvAAAB joined the room with ID 103.
User with ID wbZY1WUX9H5y70MjAAAD joined the room with ID 103.
User with ID jqPYxubRExBnfvuvAAAB sent a message to the room with ID 103.
User with ID wbZY1WUX9H5y70MjAAAD sent a message to the room with ID 103.
```

The chat room's socket functionality was built with the help of the **socket.io** package on the server side and the **socket.io-client** for the client side. In both cases, they can listen to events (with the 'on' function) and send events (with the 'emit' function).

5 events were used in total (from which 3 were custom):
connection - when a user opens the client he automatically connects to the socket
join_room (custom) - when a user presses the 'Join a room' button in the client we are actually subscribing the socket to a given channel (with that specific ID)
send_message (custom) - when a user presses the arrow button in the client to send a message, we are sending the message with additional information to that specific channel (given by the room ID)
receive_message (custom) - we use this only to send the message with additional information (after being sent to the specific channel) from the server to the client
disconnect - when a user leaves the client

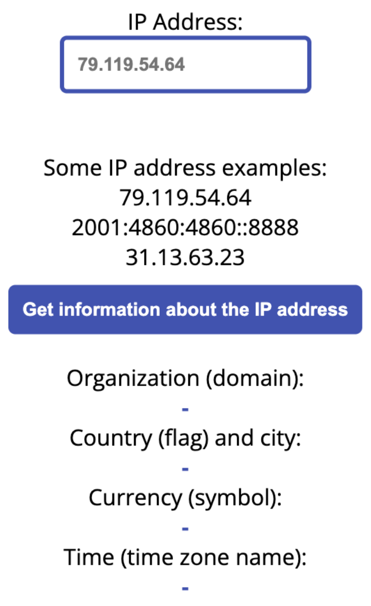
A client application for a PUBLIC Web service

Technologies used for the client:
HTML, CSS, JavaScript (+ ‘ipregistry’ as a public web service)

How to start the application in the development environment:
Open the ‘index.html’ in a browser

The application has 2 states:

1. The initial state



IP Address:

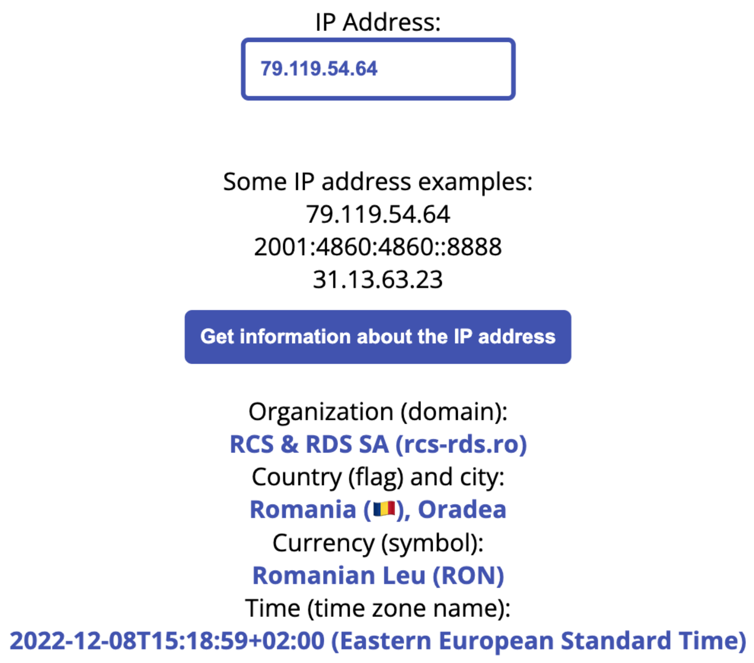
79.119.54.64

Some IP address examples:
79.119.54.64
2001:4860:4860::8888
31.13.63.23

Get information about the IP address

Organization (domain):
-
Country (flag) and city:
-
Currency (symbol):
-
Time (time zone name):
-

2. The state after an IP address was submitted



IP Address:

79.119.54.64

Some IP address examples:
79.119.54.64
2001:4860:4860::8888
31.13.63.23

Get information about the IP address

Organization (domain):
RCS & RDS SA (rcs-rds.ro)
Country (flag) and city:
Romania (🇷🇴), Oradea
Currency (symbol):
Romanian Leu (RON)
Time (time zone name):
2022-12-08T15:18:59+02:00 (Eastern European Standard Time)

As we can see based on the IP Address the public web service gives us information like:

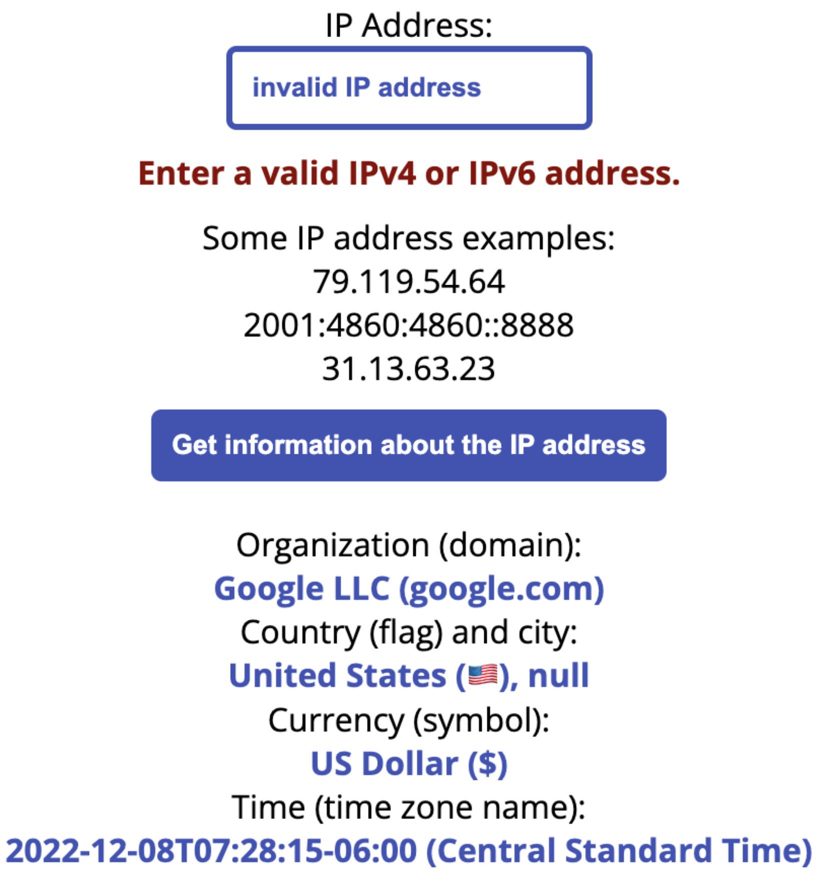
- the organization's (provider's) name and domain
- the country (with its flag) and the city
- the country's currency (with its symbol)
- the country's time (with the time zone name)

Even though we listed only these fields if the client would like further ones he should check the public web service's documentation (https://ipregistry.co/docs/fields) because there are several other fields that can be queried.

Note: There are some cases where the fields are empty (due to the lack of information the web service has). In these situations, the 'null' value will be printed.

The IP Address input field accepts IPv4 and IPv6 IP addresses too.

In order to make the UX better:
- there are some IP address examples listed below the input
- possible unexpected errors are handled (like empty input or invalid IP Address)



Enter a valid IPv4 or IPv6 address.

Some IP address examples:
79.119.54.64
2001:4860:4860::8888
31.13.63.23

Organization (domain):
Google LLC (google.com)
Country (flag) and city:
United States (🇺🇸), null
Currency (symbol):
US Dollar ($)
Time (time zone name):
2022-12-08T07:28:15-06:00 (Central Standard Time)

- hitting the 'Enter' key in the input does the same thing as clicking the 'Get information about the IP address' button

Note: Even though the web service is 'free' it has a limited number of free requests (100,000).

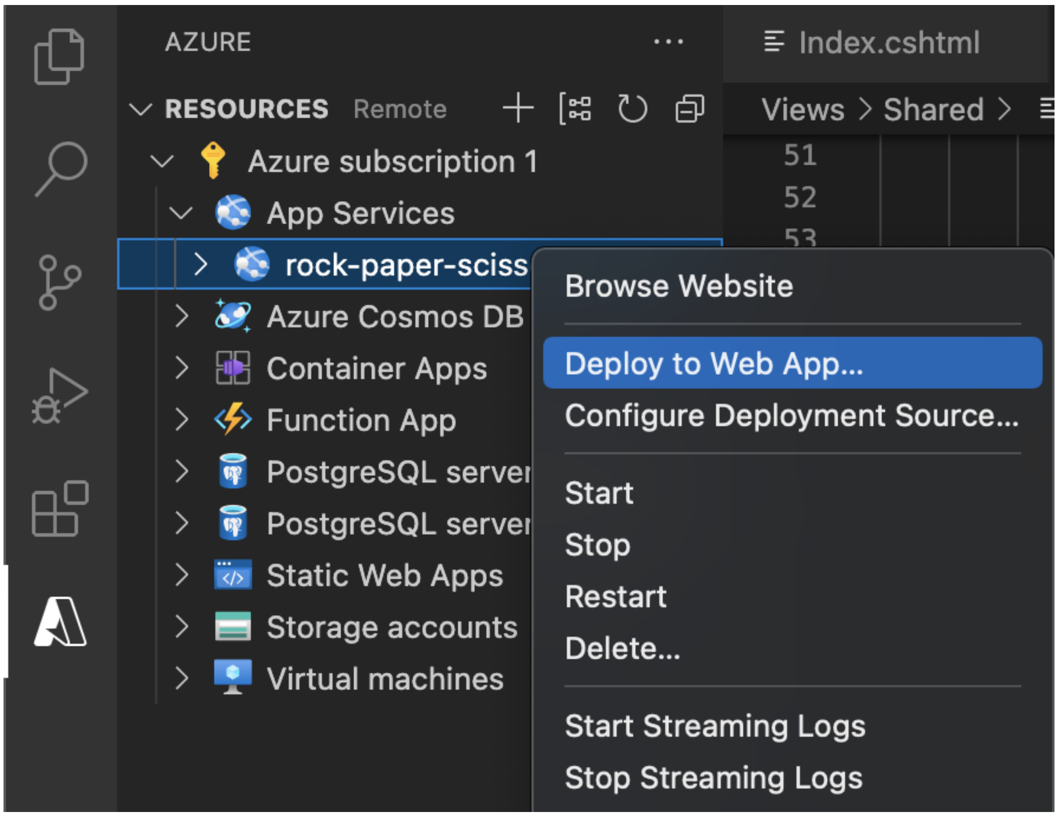
Google App Engine or Microsoft Azure application

Technologies used for the client:
HTML, CSS, JavaScript

Technologies used for the server:
.NET Framework (v6)

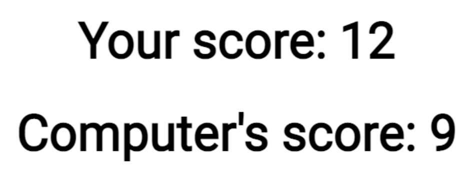
How to start the application in the development environment:
Navigate to the root folder and run the following command in the terminal: ‘dotnet run’.

How to deploy the application into a production environment:
Navigate to the root folder and run the following command in the terminal: ‘dotnet build’ (or ‘dotnet run’).
After the build is done we can use the Visual Studio Code interface to quickly deploy the application to production:



After the deployment, the application will be available on the following URL:
https://rock-paper-scissors-game.azurewebsites.net/

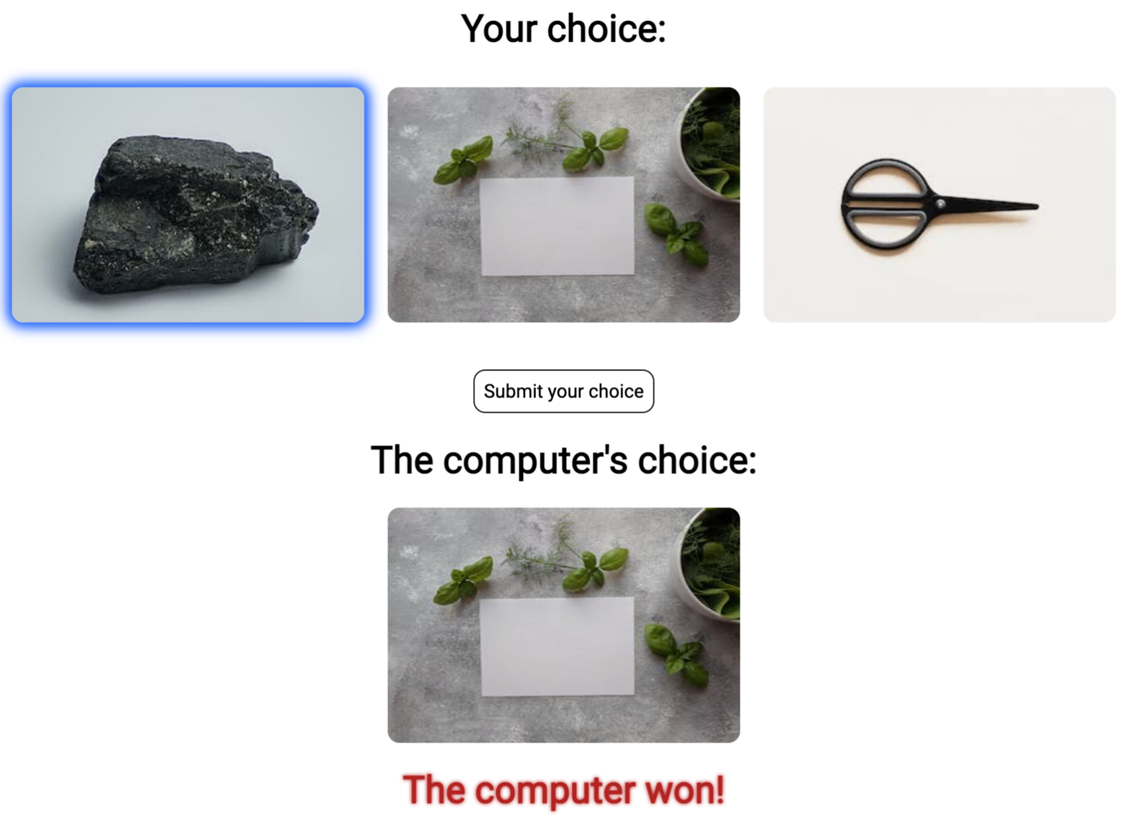
The application is a simple ‘Rock Paper Scissors’ game with score tracking (which can be reset with a page refresh).


Your score: 12

Computer's score: 9

It has some additional UI/UX features like

1. Adding a blue shadow around the user's choice
2. Adding a black shadow around the 'Submit your choice' button when it is being hovered
3. Adding a green color for the message when the user won the game
4. Adding a red color for the message when the user lost the game

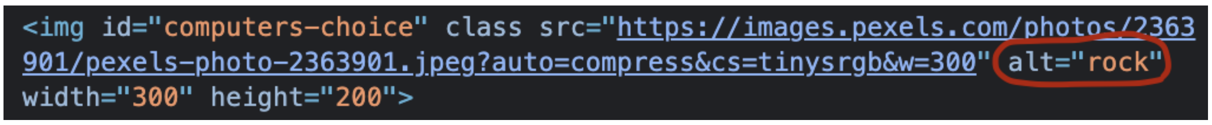


Your choice:

The computer's choice:

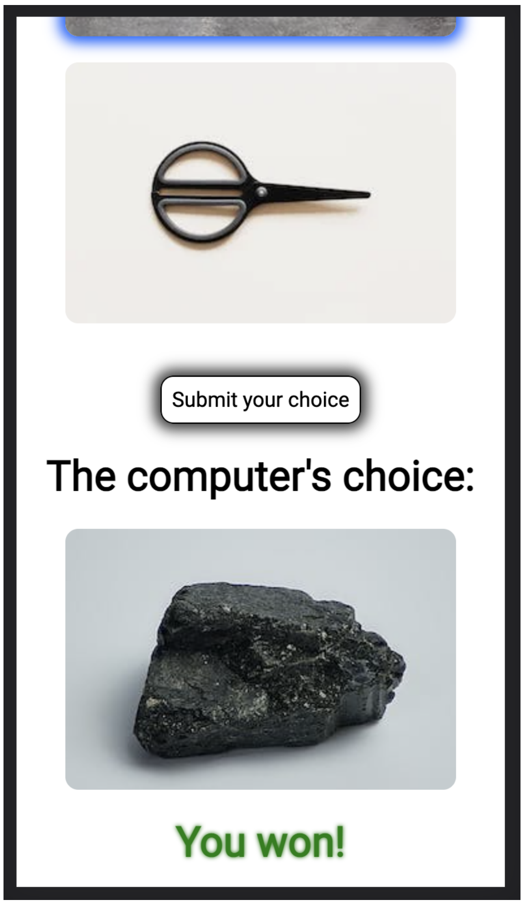
The computer won!

Moreover, all the images have their corresponding 'alt' attribute (even the computer's choice), meaning that even visually impaired persons can use the application.



```
<img id="computers-choice" class src="https://images.pexels.com/photos/2363901/pexels-photo-2363901.jpeg?auto=compress&cs=tinysrgb&w=300" alt="rock" width="300" height="200">
```

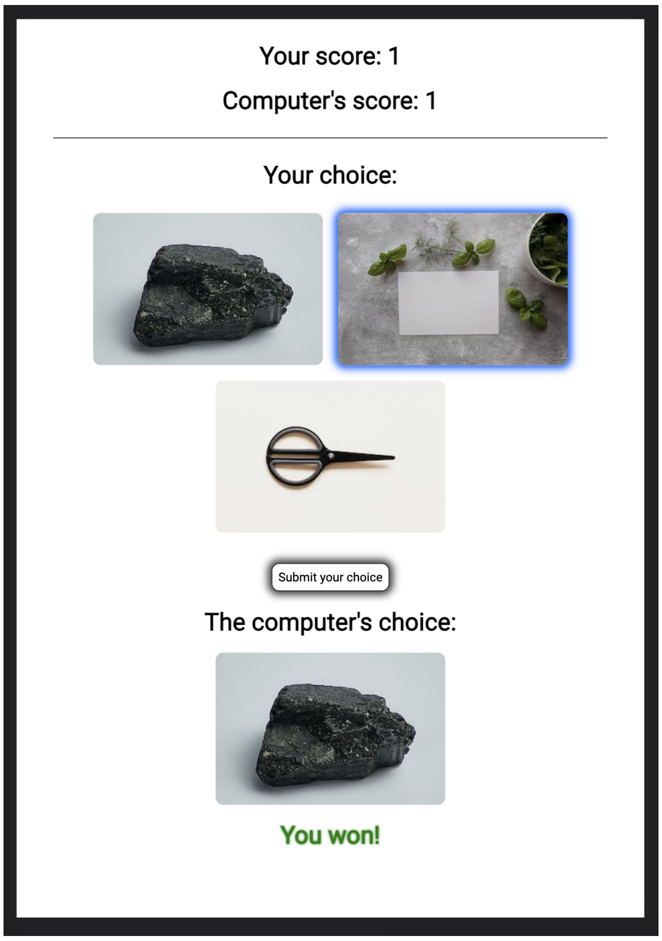
Finally, the application has a fully responsive design, meaning that it looks great on all devices:



The computer's choice:

You won!

← **iPhone SE**


Your score: 1

Computer's score: 1

Your choice:

The computer's choice:

You won!

← **iPad Air**