Build a Triva Game using WebSockets with Typescript

Build the Server Code

First, we will need to create some types but what will they be?

- **IGame** - this is the main interface where other types will branch from

  ```
    export interface IGame {
      id: string;
      currentQuestionId: string;
      done?: boolean;
      hasStarted: boolean;
      timer: number;
      clients: IClient[];
      questions: IQuestion[];
      settings: ISettings;
    }
  ```

  - **id** - a distinct identifier to represent each game
  - **currentQuestionId** - keeps track of the current question
  - **done** - boolean value to know if the game is finished
  - **hasStarted** - boolean value to know if the game has started
  - **timer** - how long a person has to answer each question
  - **clients** - the person playing the game w/ other related info
  - **questions** - all the questions for the game with other related info
  - **settings** - settings related to the game

- **IClient** - each person (computer, phone, etc) at the url of the game

  ```
    export interface IClient {
      id: string;
      name: string;
      questionsAnswered?: eQuestionAnswered[];
      score: number;
      ws?: WebSocket;
    }
  ```

  - **id** - a distinct identifier to represent the client
  - **name** - the display name of the client

  - **questionsAnswered** - keeps track of all answered questions where
    * 1 - answered correctly
    * 0 - answered incorrectly
    * -1 - not answered
  - **score** - the current score
  - **ws** - the websocket for each client, which allows us to send data to the correct client

- **IQuestion** - each question

```
export interface IQuestion {
  id: string;
  seq: number;
  text: string;
  answers: IAnswer[];
  done: boolean;
  hasFirstCorrectAnswer: boolean;
  clientIdsWhoAnswered: string[];
}
```

  - **id** - a distinct identifier to represent the question
  - **seq** - helps to keep the order of the questions
  - **text** - display text
  - **answers** - answers for each question and related info
  - **done** - boolean value to know if the question is done
  - **hasFirstCorrectAnswer** - boolean value to know if the question has been answered correctly at least once
  - **clientIdsWhoAnswered** - list of clients who answered the question

- **IAnswer** - each answer per question

```
export interface IAnswer {
  id: string;
  text: string;
  isCorrect: boolean;
}
```

  - **id** - a distinct identifier to represent the answer

  - **text** - display text
  - **isCorrect** - boolean value to know if answer is correct

- **ISettings** - settings for the game

  ```
    export interface ISettings {
      questionsCount: number;
      timePerQuestion: number;
      timeBreakPerQuestion: number;
    }
  ```

  - **questionsCount** - how many questions per game
  - **timePerQuestion** - the amount of time to answer each question
  - **timeBreakPerQuestion** - the amount of time to break before the next question loads

Create router.ts file to be the entry point for all of my websocket request to go to

- First we parse the message into json with `const obj = JSON.parse(message);`
-