Multiplayer trivia game

JavaScript and SocketIO part 1

Nathaniel Budijono

UMN ADC

October 12, 2021

UMN App Developers

Officer openings!

- Workshop instructors
- Marketing director

DM us on the discord!

https://z.umn.edu/ADCdiscord

A look ahead...

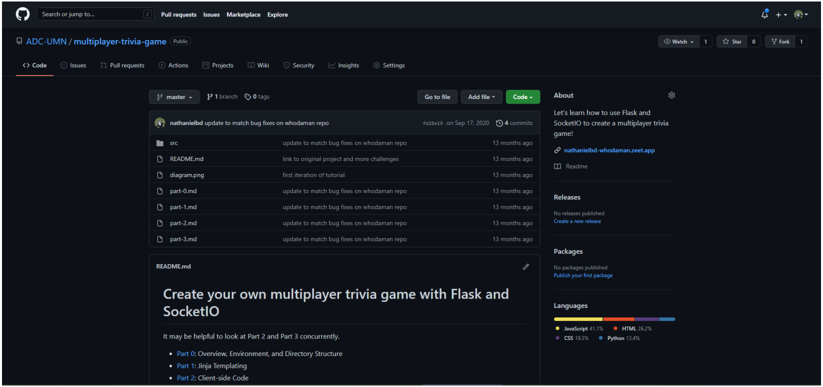
Guest Speaker Event: Postman API 101!

November 2nd, 5-6pm

Tate Hall B20

Follow the guide!

https://z.umn.edu/adc-mtg



nathanielbd update to match bug fixes on whodaman repo fd28e19 on Sep 17, 2020 4 commits

src	update to match bug fixes on whodaman repo	13 months ago
README.md	link to original project and more challenges	13 months ago
diagram.png	first iteration of tutorial	13 months ago
part-0.md	update to match bug fixes on whodaman repo	13 months ago
part-1.md	update to match bug fixes on whodaman repo	13 months ago
part-2.md	update to match bug fixes on whodaman repo	13 months ago
part-3.md	update to match bug fixes on whodaman repo	13 months ago

README.md

Create your own multiplayer trivia game with Flask and SocketIO

It may be helpful to look at Part 2 and Part 3 concurrently.

- Part 0: Overview, Environment, and Directory Structure
- Part 1: Jinja Templating
- Part 2: Client-side Code

About

Let's learn how to use Flask and SocketIO to create a multiplayer trivia game!

nathanielbd-whodaman.zeet.app

Readme

Releases

No releases published
Create a new release

Packages

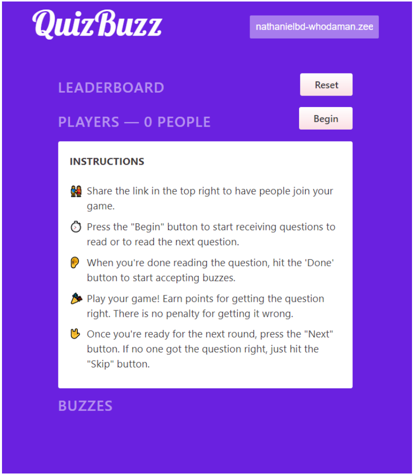
No packages published
Publish your first package

Languages

JavaScript 41.1% HTML 26.2% CSS 19.3% Python 13.4%

The questions workflow

Before



LEADERBOARD

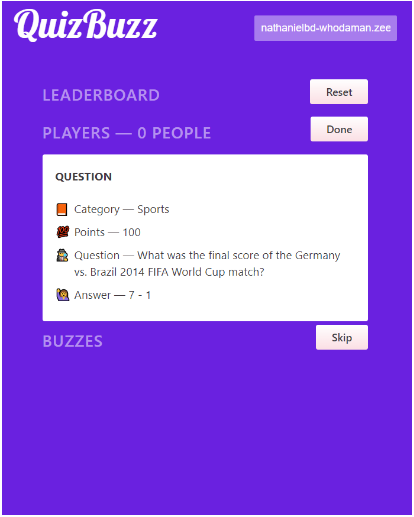
PLAYERS — 0 PEOPLE

INSTRUCTIONS

- Share the link in the top right to have people join your game.
- Press the "Begin" button to start receiving questions to read or to read the next question.
- When you're done reading the question, hit the 'Done' button to start accepting buzzes.
- Play your game! Earn points for getting the question right. There is no penalty for getting it wrong.
- Once you're ready for the next round, press the "Next" button. If no one got the question right, just hit the "Skip" button.

BUZZES

After



LEADERBOARD

PLAYERS — 0 PEOPLE

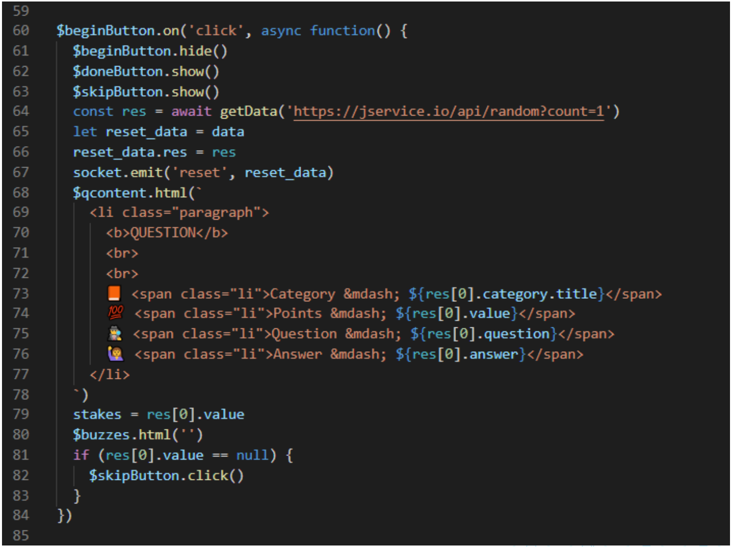
QUESTION

- Category — Sports
- Points — 100
- Question — What was the final score of the Germany vs. Brazil 2014 FIFA World Cup match?
- Answer — 7 - 1

BUZZES

The questions workflow

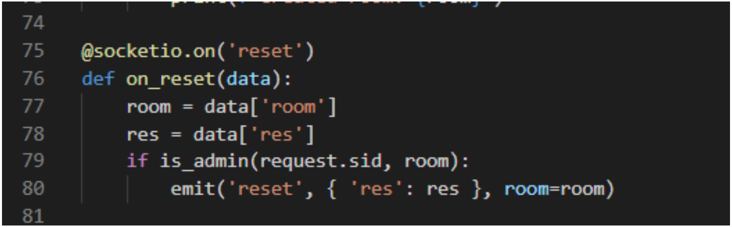
admin.js



```
$beginButton.on('click', async function() {
  $beginButton.hide()
  $doneButton.show()
  $skipButton.show()
  const res = await getData('https://jservice.io/api/random?count=1')
  let reset_data = data
  reset_data.res = res
  socket.emit('reset', reset_data)
  $qcontent.html(`
    <li class="paragraph">
      <b>QUESTION</b>
      <br>
      <br>
      📙 <span class="li">Category &mdash; ${res[0].category.title}</span>
      💯 <span class="li">Points &mdash; ${res[0].value}</span>
      🕵 <span class="li">Question &mdash; ${res[0].question}</span>
      🙋 <span class="li">Answer &mdash; ${res[0].answer}</span>
    </li>
  `)
  stakes = res[0].value
  $buzzes.html('')
  if (res[0].value == null) {
    $skipButton.click()
  }
})
```

The questions workflow

app.py



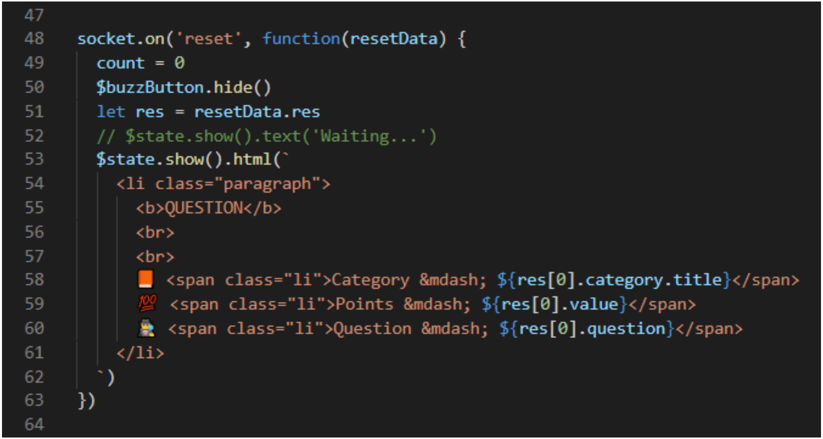
```python

@socketio.on('reset')
def on_reset(data):
    room = data['room']
    res = data['res']
    if is_admin(request.sid, room):
        emit('reset', { 'res': res }, room=room)

```

The questions workflow

play.js



```
socket.on('reset', function(resetData) {
  count = 0
  $buzzButton.hide()
  let res = resetData.res
  // $state.show().text('Waiting...')
  $state.show().html(`
    <li class="paragraph">
      <b>QUESTION</b>
      <br>
      <br>
      📙 <span class="li">Category &mdash; ${res[0].category.title}</span>
      💯 <span class="li">Points &mdash; ${res[0].value}</span>
      🕵 <span class="li">Question &mdash; ${res[0].question}</span>
    </li>
  `)
})
```

Stuff to think about for features

Think about global variables needed in `app.py` or `<client>.js`

Stuff to think about for features

Think about global variables needed in `app.py` or `<client>.js`

1. `<client>.js` emits an event, perhaps with data

Stuff to think about for features

Think about global variables needed in `app.py` or `<client>.js`

1. `<client>.js` emits an event, perhaps with data
2. `app.py` performs security checks and broadcasts the event

Stuff to think about for features

Think about global variables needed in `app.py` or `<client>.js`

1. `<client>.js` emits an event, perhaps with data
2. `app.py` performs security checks and broadcasts the event
3. `<client>.js` listens for event, updates global variables, and updates the UI

Futher reading

You can look at the guides for details on the other uses of SocketIO!

https://github.com/ADC-UMN/multiplayer-trivia-game/blob/master/part-2.md

https://github.com/ADC-UMN/multiplayer-trivia-game/blob/master/part-3.md

A challenge

Try implementing new features!

A challenge

Try implementing new features!

Fork https://github.com/ADC-UMN/whodaman-ce to get started!

A challenge

Try implementing new features!

Fork https://github.com/ADC-UMN/whodaman-ce to get started!

Some ideas:

- Try implementing ‘Daily Doubles’!

A challenge

Try implementing new features!

Fork https://github.com/ADC-UMN/whodaman-ce to get started!

Some ideas:

- Try implementing ‘Daily Doubles’!
- Try implementing real-time chat for the players!

A challenge

Try implementing new features!

Fork https://github.com/ADC-UMN/whodaman-ce to get started!

Some ideas:

- Try implementing 'Daily Doubles'!
- Try implementing real-time chat for the players!
- Add a feature to the landing page to display rooms in progress

A challenge

Try implementing new features!

Fork https://github.com/ADC-UMN/whodaman-ce to get started!

Some ideas:

- Try implementing 'Daily Doubles'!
- Try implementing real-time chat for the players!
- Add a feature to the landing page to display rooms in progress
- Add a menu to the admin page to configure things like what types of questions are used, a timer/max number of questions system, or other settings!