

Get App Name Programmatically

Language

rust

Last User Prompt

But that assumes the app knows where Cargo.toml might be, I like to get it at compile time, how to do that?

Most Possible Source Code

```
[dependencies]
cargo_toml = "0.9"
```

Test Code

```
use std::env;
use std::fs;
use std::path::Path;

fn main() {
    let manifest_dir = env::var("CARGO_MANIFEST_DIR").expect("CARGO_MANIFEST_DIR
not set");
    let cargo_toml_path = Path::new(&manifest_dir).join("Cargo.toml");
    let cargo_toml_content = fs::read_to_string(cargo_toml_path).expect("Unable to
read Cargo.toml");

    let cargo_toml: toml::Value =
toml::from_str(&cargo_toml_content).expect("Failed to parse Cargo.toml");
    let app_name = cargo_toml.get("package").and_then(|pkg|
pkg.get("name")).and_then(|name| name.as_str());

    if let Some(app_name) = app_name {
        println!("cargo:rustc-env=APP_NAME={}", app_name);
    }
}
```

Other Code Blocks

```
extern crate cargo_toml;
use std::path::Path;

fn main() {
    let manifest_path = Path::new("Cargo.toml");
    let manifest = cargo_toml::Manifest::from_path(manifest_path).unwrap();
```

```
let package_name = &manifest.package.as_ref().unwrap().name;
println!("App Name: {}", package_name);
}
```

Basho Object for Sumo

Language

javascript

Last User Prompt

please write basho.test.js

Most Possible Source Code

```
class Basho {
  constructor(id) {
    this.id = id; // To distinguish different bashos
    this.picks = {}; // dictionary mapping player names to their picks
  }

  // Function to allow a player to select a wrestler
  selectWrestler(playerName, wrestler) {
    // Check if player already has a pick
    if (this.picks[playerName]) {
      console.log(`Player ${playerName} already selected a wrestler for this
basho.`);
      return;
    }
    this.picks[playerName] = wrestler;
  }

  // Function to get a player's pick
  getPlayerPick(playerName) {
    return this.picks[playerName];
  }

  // Function to get all player's picks
  getAllPicks() {
    return this.picks;
  }
}
```

Test Code

```
const Basho = require('./Basho'); // Update with path to your Basho file
```

```
describe('Basho Class', () => {
  let basho;
  beforeEach(() => {
    basho = new Basho(1);
  });

  test('selectWrestler() should add player pick to the basho', () => {
    basho.selectWrestler('player1', 'wrestler1');
    expect(basho.getPlayerPick('player1')).toBe('wrestler1');
  });

  test('selectWrestler() should not overwrite existing pick', () => {
    basho.selectWrestler('player1', 'wrestler1');
    basho.selectWrestler('player1', 'wrestler2');
    expect(basho.getPlayerPick('player1')).toBe('wrestler1');
  });

  test('changePick() should change player pick', () => {
    basho.selectWrestler('player1', 'wrestler1');
    basho.changePick('player1', 'wrestler2');
    expect(basho.getPlayerPick('player1')).toBe('wrestler2');
  });

  test('getAllPicks() should return all picks', () => {
    basho.selectWrestler('player1', 'wrestler1');
    basho.selectWrestler('player2', 'wrestler2');
    expect(basho.getAllPicks()).toEqual({
      player1: 'wrestler1',
      player2: 'wrestler2'
    });
  });
});
```

Other Code Blocks

```
changePick(playerName, newWrestler) {
  if (!this.picks[playerName]) {
    console.log(`Player ${playerName} has not selected a wrestler for this
basho.`);
    return;
  }
  this.picks[playerName] = newWrestler;
}
```

Expressão Regular para Data

Language

java

Last User Prompt

em java pra cucumber junit?

Most Possible Source Code

```
^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/\d{4}$
```

Test Code

```
import io.cucumber.java.en.When;
import static org.junit.Assert.assertEquals;

import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class StepDefinitions {

    private String texto;
    private String dataFormatada;

    @When("a expressao regular extrai numeros e formata como data com barras")
    public void extrairNumerosEFormatarData() {
        // Expressão regular para extrair números
        Pattern pattern = Pattern.compile("\\d");
        Matcher matcher = pattern.matcher(texto);

        // Construindo a data formatada com barras
        StringBuilder dataBuilder = new StringBuilder();
        while (matcher.find()) {
            dataBuilder.append(matcher.group());
        }

        // Inserindo barras na data formatada
        dataFormatada = dataBuilder.toString().replaceAll("(\\d{2})(\\d{2})(\\d{4})", "$1/$2/$3");
    }

    // Métodos de apoio para configurar o cenário

    @When("a string contém o seguinte texto {string}")
    public void aStringContemOTexto(String texto) {
        this.texto = texto;
    }

    @When("a expressao regular é aplicada")
    public void aExpressaoRegularEhAplicada() {
        extrairNumerosEFormatarData();
    }
}
```

```

    @Then("o resultado deve ser a data formatada {string}")
    public void oResultadoDeveSerADataFormatada(String resultadoEsperado) {
        assertEquals(resultadoEsperado, dataFormatada);
    }
}

```

Other Code Blocks

```

^(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[0-2])/\d{4}$

```

```

import re

def formatar_data_com_barras(texto):
    # Expressão regular para extrair números
    numeros = re.findall(r'\d', texto)

    # Formatando os números como uma data com barras
    data_formatada = '/'.join(numeros)

    return data_formatada

# Exemplo de uso
texto = "A data é 25122023"
data_formatada = formatar_data_com_barras(texto)
print(data_formatada)

```

Deploying wavebeat on Replicate

Language

python

Last User Prompt

I want to make a cog to host a wavebeat model on Replicate.

I think that I should prepare two files: cog.yaml and scripts/predict.py

Let's look at what another ML model, omnizart, does to host:

omnizart/cog.yaml:

```

build: gpu: true python_version: "3.8" system_packages: - "libgl1-mesa-glx" - "libglib2.0-0" - "libsndfile1-dev" -
"ffmpeg" python_packages: - "ipython==7.30.1" - "numpy==1.21.4" run: - pip install -U pip - pip install --
upgrade cython - pip install omnizart - apt-get update && apt-get install -y fluidsynth - pip install
pyfluidsynth

```

```
predict: "scripts/predict.py:Predictor"
```

```
omnizart/scripts/predict.py:
```

```
""" To push this predictor to replicate.com, first run download_checkpoints() and save files to
omnizart/checkpoints. Then run cog push. Further documentation can be found at https://replicate.com/docs
"""
```

```
import os import tempfile import subprocess import shutil from pathlib import Path
```

```
import cog import scipy.io.wavfile as wave
```

```
from omnizart.remote import download_large_file_from_google_drive from omnizart.beat import app as bapp
from omnizart.chord import app as capp from omnizart.drum import app as dapp from omnizart.music import
app as mapp from omnizart.vocal import app as vapp from omnizart.vocal_contour import app as vcapp
```

```
class Predictor(cog.Predictor): def setup(self): self.SF2_FILE = "general_soundfont.sf2" if not
os.path.exists(self.SF2_FILE): print("Downloading soundfont...") download_large_file_from_google_drive(
"https://ftp.osuosl.org/pub/muscore/soundfont/MuseScore_General/MuseScore_General.sf2",
file_length=215614036, save_name=self.SF2_FILE, ) self.app = {"music": mapp, "chord": capp, "drum": dapp,
"vocal": vapp, "vocal-contour": vcapp, "beat": bapp} self.model_path = {"piano": "Piano", "piano-v2":
"PianoV2", "assemble": "Stream", "pop-song": "Pop", "": None}
```

```
@cog.input(
    "audio",
    type=Path,
    help="Path to the input music. Supports mp3 and wav format.",
)
@cog.input(
    "mode",
    type=str,
    default="music-piano-v2",
    options=["music-piano", "music-piano-v2", "music-assemble", "chord", "drum",
"vocal", "vocal-contour", "beat"],
    help="Transcription mode",
)
def predict(self, audio, mode):
    assert str(audio).endswith(".mp3") or str(audio).endswith(".wav"), "Please
upload mp3 or wav file."
    temp_folder = "cog_temp"
    os.makedirs(temp_folder, exist_ok=True)
    try:
        audio_name = str(os.path.splitext(os.path.basename(audio))[0])
        if str(audio).endswith(".wav"):
            wav_file_path = str(audio)
        else:
            wav_file_path = f"{temp_folder}/{audio_name}.wav"
            subprocess.run(["ffmpeg", "-y", "-i", str(audio), wav_file_path])
        model = ""
        if mode.startswith("music"):
```

```

        mode_list = mode.split("-")
        mode = mode_list[0]
        model = "-".join(mode_list[1:])

    app = self.app[mode]
    model_path = self.model_path[model]
    midi = app.transcribe(wav_file_path, model_path=model_path)

    if mode == "vocal-contour":
        out_name = f"{audio_name}_trans.wav"
    else:
        print("Synthesizing MIDI...")
        out_name = f"{temp_folder}/{audio_name}_synth.wav"
        raw_wav = midi.fluidsynth(fs=44100, sf2_path=self.SF2_FILE)
        wave.write(out_name, 44100, raw_wav)

    out_path = Path(tempfile.mkdtemp()) / "out.mp3" # out_path is
automatically cleaned up by cog
    subprocess.run(["ffmpeg", "-y", "-i", out_name, str(out_path)])
    finally:
        shutil.rmtree(temp_folder)
        if os.path.exists(f"{audio_name}.mid"):
            os.remove(f"{audio_name}.mid")
        if os.path.exists(f"{audio_name}_trans.wav"):
            os.remove(f"{audio_name}_trans.wav")
    return out_path

```

now let's see what we have in wavebeat repo:

csteinmetz1 Merge pull request #1 from nicolasanjanoran/main ... cfb9ed9 on Nov 11, 2021 Git stats 71 commits
Files Type Name Latest commit message Commit time checkpoints adding link to pre-trained model on
Zenodo 2 years ago docs/resources updating links 2 years ago util script to write out simple beat format for
gtzan 2 years ago wavebeat fix: pass use_gpu param to the predict_beats function 2 years ago .gitignore
adding webpage and updating pdf 2 years ago LICENSE Initial commit 2 years ago README.md adding link to
arXiv paper 2 years ago predict.py adding detail about 3.8 and remove import from predict.py 2 years ago
requirements.txt adding improvements for macOS support in requirements 2 years ago setup.py finalizing the
interface 2 years ago simple_test.py removing extra models 2 years ago train.py removing extra models 2
years ago train_cv.py training scripts for cross validation as well as multi-dataset 2 years ago README.md
WaveBeat End-to-end beat and downbeat tracking in the time domain.

| Paper | Code | Video | Slides |

Setup First clone the repo.

git clone <https://github.com/csteinmetz1/wavebeat.git> cd wavebeat Setup a virtual environment and activate it. This requires that you use Python 3.8.

python3 -m venv env/ source env/bin/activate Next install numpy, cython, and aiohttp first, manually.

pip install numpy cython aiohttp Then install the wavebeat module.

python setup.py install This will ensure that madmom installs properly, as it currently fails unless cython, numpy, and aiohttp are installed first.

Predicting beats To begin you will first need to download the pre-trained model here. Place it in the checkpoints/ directory, rename to get the .ckpt file.

```
cd checkpoints wget https://zenodo.org/record/5525120/files/wavebeat_epoch%3D98-step%3D24749.ckpt?download=1 mv wavebeat_epoch=98-step=24749.ckpt?download=1 wavebeat_epoch=98-step=24749.ckpt
```

Functional interface If you would like to use the functional interface you can create a script and import wavebeat as follows.

```
from wavebeat.tracker import beatTracker
```

beat, downbeats = beatTracker('audio.wav') Script interface We provide a simple script interface to load an audio file and predict the beat and downbeat locations with a pre-trained model. Run the model by providing a path to an audio file.

```
python predict.py path_to_audio.wav
```

wavebeat/predict.py:

```
import os import glob import torch import torchaudio import pytorch_lightning as pl from argparse import ArgumentParser
```

```
from wavebeat.dstcn import dsTCNModel
```

```
parser = ArgumentParser() parser.add_argument('input', type=str) parser.add_argument('--model', type=str, default="checkpoints/")
```

```
args = parser.parse_args()
```

find the checkpoint path

```
ckpts = glob.glob(os.path.join(args.model, "*.ckpt")) if len(ckpts) < 1: raise RuntimeError(f"No checkpoints found in {args.model}.") else: ckpt_path = ckpts[-1]
```

construct the model, and load weights from checkpoint

```
model = dsTCNModel.load_from_checkpoint(ckpt_path)
```

set model to eval mode

```
model.eval()
```

get the locations of the beats and downbeats

```
beats, downbeats = model.predict_beats(args.input)
```

print some results to terminal

```
print(f"Beats found in {args.input}") print("-" * 32) for beat in beats: print(f"{beat:0.2f}")
```

```
print() print(f"Downbeats found in {args.input}") print("-" * 32) for downbeat in downbeats: print(f"{downbeat:0.2f}")
```

as I yesterday played with installing and running wavebeat on a fresh VPS, I found this setup working (some steps are redundant and can be squashed):

Run on Ubuntu 20.04 with python3.8, numpy==1.19 and madmom==0.16.1 recompiled after installing numpy.

history

```
1 apt update
2 python3.8
3 git clone https://github.com/csteinmetz1/wavebeat.git
4 cd wavebeat
5 python3 -m venv env/
6 source env/bin/activate
7 apt install python3.8-venv
8 alias apt='apt -y'
9 python3 -m venv env/
```

```
10 source env/bin/activate 11 pip install numpy cython aiohttp 12 python setup.py install 13 pip install
scipy==1.11.2 14 pip install scipy==1.10.1 15 python setup.py install 16 pip --version 17 cat requirements.txt |
grep mat 18 pip install matplotlib==3.3.4 19 python setup.py install 20 apt install gcc python3.8-dev 21
python setup.py install 22 python3 predict.py /tmp/Carver\ Commodore\ -\ Drown\ Me\ In\ Emotions-
bLaipK9lisc.mp3 23 ack np.float 24 apt install ack 25 ack np.float 26 pip freeze 27 cat requirements.txt 28 pip
uninstall numpy 29 pip install numpy==1.19 30 pip install numpy==1.19.5 31 python3 predict.py
/tmp/Carver\ Commodore\ -\ Drown\ Me\ In\ Emotions-bLaipK9lisc.mp3 32 pip uninstall madmom 33 pip
install madmom==0.16.1 34 python3 predict.py /tmp/Carver\ Commodore\ -\ Drown\ Me\ In\ Emotions-
bLaipK9lisc.mp3 35 cd checkpoints/ 36 ll 37 wget
https://zenodo.org/record/5525120/files/wavebeat_epoch%3D98-step%3D24749.ckpt?download=1 38 mv
wavebeat_epoch=98-step=24749.ckpt?download=1 wavebeat_epoch=98-step=24749.ckpt 39 python3
predict.py /tmp/Carver\ Commodore\ -\ Drown\ Me\ In\ Emotions-bLaipK9lisc.mp3 40 cd .. 41 python3
predict.py /tmp/Carver\ Commodore\ -\ Drown\ Me\ In\ Emotions-bLaipK9lisc.mp3 42 python3 predict.py
/tmp/Carver\ Commodore\ -\ Drown\ Me\ In\ Emotions-bLaipK9lisc.mp3 > wavebeat.txt
```

Most Possible Source Code

```
nothing
```

Test Code

```
import os
import glob
import torch
from pathlib import Path
import cog
from wavebeat.dstcn import dsTCNModel

class Predictor(cog.Predictor):
    def setup(self):
        # find the checkpoint path
        ckpts = glob.glob(os.path.join("checkpoints", "*.ckpt"))
        if len(ckpts) < 1:
            raise RuntimeError(f"No checkpoints found in checkpoints.")
        else:
            self.ckpt_path = ckpts[-1]
        # construct the model, and load weights from checkpoint
        self.model = dsTCNModel.load_from_checkpoint(self.ckpt_path)
        # set model to eval mode
        self.model.eval()

    @cog.input("input", type=Path, help="Path to the input music. Supports wav format.")
    def predict(self, audio_path):
        # Ensure audio is in the supported format
        assert str(audio_path).endswith(".wav"), "Please upload wav file."

        # Predict beats and downbeats
        beats, downbeats = self.model.predict_beats(str(audio_path))

        # Create results as string
        result = "Beats:\n"
        for beat in beats:
            result += f"{beat:0.2f}\n"
        result += "\nDownbeats:\n"
        for downbeat in downbeats:
            result += f"{downbeat:0.2f}\n"

        # Save to file and return
        out_path = audio_path.parent / "result.txt"
        with open(out_path, 'w') as f:
            f.write(result)
        return out_path
```

Troubleshoot pytest warnings

Language

python

Last User Prompt

pytest is giving me these warnings:

```
EAB_tools/tests/io/test_display.py: 16 warnings
EAB_tools/tests/util/test_hashing.py: 3 warnings
c:\Users\mrubin8\Projects-Git\EAB_tools\EAB_tools\conftest.py:100:
RuntimeWarning: invalid value encountered in cast
  pytest.param(lambda arr: np rint(arr).astype(int), id="rint"), # any -> int
```

However, it's really hard for me to tell which tests are raising the warning. How can I get more info?

Most Possible Source Code

```
RuntimeWarning: invalid value encountered in cast
  pytest.param(lambda arr: np rint(arr).astype(int), id="rint"), # any -> int
```

Test Code

```
import warnings

def test_something():
    with warnings.catch_warnings(record=True) as w:
        # Trigger warning
        ...

    assert len(w) == 1
    assert isinstance(w[-1].category, RuntimeWarning)
    assert "invalid value encountered in cast" in str(w[-1].message)
```

Create Pick Object

Language

javascript

Last User Prompt

jesus - no, i will run npm test

Most Possible Source Code

```
// Using object literals
const pick = {
  wrestlerName: '',
  // You can add more properties relevant to the pick if needed
};

// Using a constructor function
function Pick(wrestlerName) {
  this.wrestlerName = wrestlerName;
  // You can add more properties relevant to the pick if needed
}

// Usage
const myPick = new Pick('Sumo Wrestler 1');
console.log(myPick.wrestlerName); // Output: Sumo Wrestler 1
```

Test Code

```
const Pick = require('./pick');

describe('Pick', () => {
  it('should set the wrestlerName property correctly', () => {
    const myPick = new Pick('Sumo Wrestler 1');
    expect(myPick.wrestlerName).toBe('Sumo Wrestler 1');
  });
});
```

Other Code Blocks

```
class Pick {
  constructor(wrestlerName) {
    this.wrestlerName = wrestlerName;
    // You can add more properties relevant to the pick if needed
  }
}

module.exports = Pick;
```

```
const Pick = require('./pick');

// Test case 1
const myPick = new Pick('Sumo Wrestler 1');
console.log(myPick.wrestlerName); // Output: Sumo Wrestler 1

// Test case 2
```

```
const anotherPick = new Pick('Sumo Wrestler 2');
console.log(anotherPick.wrestlerName); // Output: Sumo Wrestler 2
```

Render localStorage on load

Language

javascript

Last User Prompt

I need some place on the page to render the contents of localStorage on every page load. After I get this working I will want to add to my unit tests to ensure that this will always happen.

index.html

Welcome to Banzuke Surfing Game!

Select your Rikishi and start playing!

Rikishi 1 ▾

Start Playing

Backfilled Results:

Admin Panel

Switch user:

Switch User

Backfill contest results:

Enter contest name

Enter rikishi selection

Backfill Results

game.js export default class Game { constructor(initializeImmediately = false) { this.user = this.getUser(); if (initializeImmediately) { this.initialize(); } }

```
startPlaying() {
  const rikishi = document.querySelector('#rikishi').value;
  const picks = this.getPicks();
  const message = "You selected: " + rikishi + "\nPrevious Picks: " +
JSON.stringify(picks);
  this.updatePicks(rikishi); // Update the picks with the new selection
  return message;
}

getUser() {
  // get user from local storage
  let user = localStorage.getItem('user');
```

```
    if (!user) {
        user = 'admin';
        localStorage.setItem('user', user);
    }
    return user;
}

getPicks() {
    const picks = JSON.parse(localStorage.getItem(this.user));
    if (!picks) {
        return {};
    }
    return picks;
}

updatePicks(rikishi) {
    const picks = this.getPicks();
    const currentContest = new Date().getMonth();
    if ([0, 2, 4, 6, 8, 10].includes(currentContest)) {
        const contestName = new Date().toLocaleString('default', { month: 'long'
    }) + ' ' + new Date().getFullYear();
        picks[contestName] = rikishi;
        localStorage.setItem(this.user, JSON.stringify(picks));
    }
}

switchUser() {
    const newUser = document.querySelector('#userSwitch').value;
    localStorage.setItem('user', newUser);
    document.querySelector('#user').textContent = 'Current user: ' + newUser;
    this.user = newUser;
}

backfillResults() {
    const contestName = document.querySelector('#backfillContest').value;
    const rikishi = document.querySelector('#backfillRikishi').value;
    const picks = this.getPicks();
    picks[contestName] = rikishi;
    localStorage.setItem(this.user, JSON.stringify(picks));
    this.provideFeedback('Backfilled results for ' + contestName + ' with ' +
rikishi); // Provide feedback
    this.displayBackfilledResults(); // Display the updated results
}

displayBackfilledResults() {
    const picks = this.getPicks();
    const resultsElement = document.querySelector('#backfilledResults');

    // Clear previous results
    resultsElement.textContent = '';
```

```

    // Display each contest result
    for (const contest in picks) {
        const rikishi = picks[contest];
        const resultText = document.createTextNode(contest + ': ' + rikishi);
        const resultDiv = document.createElement('div');
        resultDiv.appendChild(resultText);
        resultsElement.appendChild(resultDiv);
    }
}

provideFeedback(message) {
    document.querySelector('#feedback').textContent = message;
}

initialize() {
    const userElement = document.querySelector('#user');
    if (userElement) {
        userElement.textContent = 'Current user: ' + this.user;
    }
    this.displayBackfilledResults(); // Display the initial results

    // Add event listeners
    document.querySelector("#startPlayingButton").addEventListener('click', () =>
this.startPlaying());
    document.querySelector("#switchUserButton").addEventListener('click', () =>
this.switchUser());
    document.querySelector("#backfillResultsButton").addEventListener('click', ()
=> this.backfillResults());
}

```

```

}

```

```

if (typeof window !== 'undefined') { window.game = new Game(); }

```

Most Possible Source Code

```

export default class Game {
    constructor(initializeImmediately = false) {
        this.user = this.getUser();
        if (initializeImmediately) {
            this.initialize();
        }
    }

    startPlaying() {
        const rikishi = document.querySelector('#rikishi').value;
        const picks = this.getPicks();
        const message = "You selected: " + rikishi + "\nPrevious Picks: " +
JSON.stringify(picks);
        this.updatePicks(rikishi); // Update the picks with the new selection
    }
}

```

```
        return message;
    }

    getUser() {
        // get user from local storage
        let user = localStorage.getItem('user');
        if (!user) {
            user = 'admin';
            localStorage.setItem('user', user);
        }
        return user;
    }

    getPicks() {
        const picks = JSON.parse(localStorage.getItem(this.user));
        if (!picks) {
            return {};
        }
        return picks;
    }

    updatePicks(rikishi) {
        const picks = this.getPicks();
        const currentContest = new Date().getMonth();
        if ([0, 2, 4, 6, 8, 10].includes(currentContest)) {
            const contestName = new Date().toLocaleString('default', { month:
'long' }) + ' ' + new Date().getFullYear();
            picks[contestName] = rikishi;
            localStorage.setItem(this.user, JSON.stringify(picks));
        }
    }

    switchUser() {
        const newUser = document.querySelector('#userSwitch').value;
        localStorage.setItem('user', newUser);
        document.querySelector('#user').textContent = 'Current user: ' + newUser;
        this.user = newUser;
    }

    backfillResults() {
        const contestName = document.querySelector('#backfillContest').value;
        const rikishi = document.querySelector('#backfillRikishi').value;
        const picks = this.getPicks();
        picks[contestName] = rikishi;
        localStorage.setItem(this.user, JSON.stringify(picks));
        this.provideFeedback('Backfilled results for ' + contestName + ' with ' +
rikishi); // Provide feedback
        this.displayBackfilledResults(); // Display the updated results
    }

    displayBackfilledResults() {
        const picks = this.getPicks();
        const resultsElement = document.querySelector('#backfilledResults');
```



```

    // Clear previous results
    resultsElement.textContent = '';

    // Display each contest result
    for (const contest in picks) {
        const rikishi = picks[contest];
        const resultText = document.createTextNode(contest + ': ' + rikishi);
        const resultDiv = document.createElement('div');
        resultDiv.appendChild(resultText);
        resultsElement.appendChild(resultDiv);
    }
}

provideFeedback(message) {
    document.querySelector('#feedback').textContent = message;
}

initialize() {
    const userElement = document.querySelector('#user');
    if (userElement) {
        userElement.textContent = 'Current user: ' + this.user;
    }
    this.displayBackfilledResults(); // Display the initial results

    // Add event listeners
    document.querySelector("#startPlayingButton").addEventListener('click', ()
=> this.startPlaying());
    document.querySelector("#switchUserButton").addEventListener('click', ()
=> this.switchUser());
    document.querySelector("#backfillResultsButton").addEventListener('click',
() => this.backfillResults());
}

if (typeof window !== 'undefined') {
    window.game = new Game();
}

```

Test Code

```

import Game from './game.js';

describe('Game', () => {
    beforeEach(() => {
        // Clear localStorage before each test
        localStorage.clear();
    });

    it('should render localStorage on initialization', () => {
        // Set something in localStorage
        localStorage.setItem('test', 'testValue');
    });
});

```

```
    const game = new Game(true);

    // Check if the initialize function has been called and content is
    rendered
    // Here, it assumes that you have a way to check whether the content is
    rendered on the page
    expect(game.initialize).toBeCalled();
  });

  it('should always call initialize on window load', () => {
    // Define a mock function for the initialize method
    const mockInitialize = jest.fn();

    // Replace the original method with the mock function
    Game.prototype.initialize = mockInitialize;

    // Simulate window load
    require('./game.js');

    // Check if initialize has been called
    expect(mockInitialize).toBeCalled();
  });
});
```

Other Code Blocks

```
<!DOCTYPE html>
<html>
<head>
  <title>Banzuke Surfing Game</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <!-- Link to the external CSS file -->
  <!-- Uncomment this if you have styles to include -->
  <!-- <link rel="stylesheet" href="styles.css"> -->
</head>
<body>
  <h1>Welcome to Banzuke Surfing Game!</h1>
  <p id="user"></p>
  <p>Select your Rikishi and start playing!</p>
  <select id="rikishi">
    <option value="1">Rikishi 1</option>
    <option value="2">Rikishi 2</option>
    <!-- more options here -->
  </select>
  <button id="startPlayingButton">Start Playing</button>
  <hr>
  <h2>Backfilled Results:</h2>
  <div id="backfilledResults"></div>
  <h2>Admin Panel</h2>
```

```

    <p>Switch user:</p>
    <input id="userSwitch" type="text">
    <button id="switchUserButton">Switch User</button>
    <p>Backfill contest results:</p>
    <input id="backfillContest" type="text" placeholder="Enter contest name">
    <input id="backfillRikishi" type="text" placeholder="Enter rikishi selection">
    <button id="backfillResultsButton">Backfill Results</button>
    <p id="feedback"></p>
    <!-- Link to the external JavaScript file -->
    <script type="module" src="game.js"></script>
  </body>
</html>

```

Conditional Rendering with React

Language

javascript

Last User Prompt

```

{field.meta.label}{' '} {field.required ? " : `(${t('not_required_short')}`) } {field.meta.subtitle &&
{field.meta.subtitle}}
{errorMessage && (
{errorMessage}

)) {Object.keys(field.meta.values).map((key: string) => (
☐ {field.meta.values[key]}
))}

```

If this is for example my react return component. On field.meta.ifOneOf I can have a object available with multiple ifOneOf or ifAllOf like this:

```
ifOneOf: { object_with_id: 'wrong', ifOneOf: { category: 'bar', subcategory: 'wrong', }, },
```

Where the keys (except the ifOneOf and ifAllOf) are field names that are registered in my form with react hook form (getValues(fieldName))

How can I make something that I only render the field if the ifOneOf or ifAllOf statement resolves to true? the value of the key is the value that the input should have

Most Possible Source Code

```

import { get } from 'lodash';
import { useFormContext } from 'react-hook-form';

const renderField IfOneOfOrIfAllOf = ({ field, t }) => {
  const { register, getValues } = useFormContext();
  const values = getValues();

```

```

const validateIfCondition = (condition, values) => {
  for (const key in condition) {
    if (key === 'ifOneOf' || key === 'ifAllOf') {
      if (key === 'ifOneOf') {
        if (!condition[key].some(key => values[key] === condition[key].value)) {
          return false;
        }
      } else if (key === 'ifAllOf') {
        if (condition[key].some(key => values[key] === condition[key].value)) {
          return false;
        }
      }
    } else {
      const nestedValues = get(values, key);
      if (nestedValues && !validateIfCondition(condition[key], nestedValues)) {
        return false;
      }
    }
  }
  return true;
};

const shouldRender = validateIfCondition(field.meta.ifOneOf ||
field.meta.ifAllOf, values);

return shouldRender && (
  <div>
    <input
      {...register(`${field.key}`, getValidators(field, t))}
      type="checkbox"
      id={`${field.key}`}
      value={field.key}
      aria-describedby={errorMessage ? `${field.key}-error` : undefined}
      defaultChecked={getDefaultValueCheckboxInput(field.key, field.key)}
    />
    <label htmlFor={`${field.key}`}>{field.meta.label}</label>
  </div>
);
};

```

Test Code

```

import React from 'react';
import { useFormContext } from 'react-hook-form';

const ConditionalFieldset = ({ field, t }) => {
  const { getValues, register } = useFormContext(); // react-hook-form context
  const errorMessage = ''; // Assume you have logic to retrieve the error message

  // Only render fieldset if conditions are met

```

```

    if (!shouldRenderField(field, getValues)) {
      return null;
    }

    return (
      <fieldset aria-invalid={!errorMessage}>
        <legend>
          {field.meta.label}{' '}
          <span> {field.required ? '' : `(${t('not_required_short')})`} </span>
          {field.meta.subtitle && <span>{field.meta.subtitle}</span>}
        </legend>
        {errorMessage && (
          <p
            id={`_${field.key}-error`}
            aria-live="assertive"
            style={{ color: 'red' }}
          >
            {errorMessage}
          </p>
        )}
        {Object.keys(field.meta.values).map((key) => (
          <div key={key}>
            <input
              {...register(`_${field.key}.${key}`, getValidators(field, t))}
              type="checkbox"
              id={`_${field.key}-${key}`}
              value={key}
              aria-describedby={errorMessage ? `_${field.key}-error` : undefined}
              defaultChecked={getDefaultCheckboxInput(field.key, key)}
            />
            <label htmlFor={`_${field.key}-${key}`}>
              {field.meta.values[key]}
            </label>
          </div>
        ))}
      </fieldset>
    );
  };
}

export default ConditionalFieldset;

```

Local Storage Rikishi Game

Language

javascript

Last User Prompt

I want this game to rely on local storage to remember who I am and who my picks were in previous contests. A contest is January, March, May, July, September, or November of a given year. The current contest is July

2023. We will assume I am in admin mode and I can switch users to record everyone's picks (which are visible to everyone) and backfill old results. Please add at least one new test.

index.html

Welcome to Banzuke Surfing Game!

Select your Rikishi and start playing!

Rikishi 1 ▾

Start Playing

game.js function startPlaying() { var rikishi = \$('#rikishi').val(); var message = "You selected: " + rikishi; return message; }

module.exports = { startPlaying };

game.test.js

global.\$ = jest.fn(() => ({ val: jest.fn(() => '1') }));

const { startPlaying } = require('./game');

test('check if startPlaying is defined and returns expected value', () => { const result = startPlaying() expect(result).toBe("You selected: 1"); });

Most Possible Source Code

```
<!DOCTYPE html>
<html>
<head>
  <title>Banzuke Surfing Game</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
  <h1>Welcome to Banzuke Surfing Game!</h1>
  <p>Select your Rikishi and start playing!</p>
  <select id="rikishi">
    <option value="1">Rikishi 1</option>
    <option value="2">Rikishi 2</option>
    <!-- more options here -->
  </select>
  <button onclick="startPlaying()">Start Playing</button>
  <script src="game.js"></script>
</body>
</html>
```

Test Code

```
// You need to install jest-localstorage-mock to mock localStorage
// You can install it via npm: npm install --save-dev jest-localstorage-mock

require('jest-localstorage-mock'); // import the local storage mock

global.$ = jest.fn(() => ({
  val: jest.fn(() => '1')
}));

const { startPlaying } = require('./game');

beforeEach(() => {
  localStorage.clear(); // clear localStorage before each test
});

test('check if startPlaying is defined and returns expected value', () => {
  localStorage.setItem('user', 'testUser');
  localStorage.setItem('testUser', JSON.stringify({ 'July 2023': '1' })); //
  Assume a previous pick
  const result = startPlaying();
  expect(result).toBe("You selected: 1\nPrevious Picks: {\"July 2023\": \"1\"}");
});
```

Other Code Blocks

```
function startPlaying() {
  var rikishi = $('#rikishi').val();
  var message = "You selected: " + rikishi;
  return message;
}

module.exports = { startPlaying };
```

Module Error in Browser.

Language

javascript

Last User Prompt

Please write an updated version of game.test.js

```
const jsdom = require('jsdom'); const { JSDOM } = jsdom;
```

```
const dom = new JSDOM('<!doctype html>', { url: "http://localhost/", resources: "usable", runScripts:
"dangerously", });
```

```
global.window = dom.window; global.document = dom.window.document; global.localStorage =
dom.window.localStorage;
```

```
const { startPlaying, switchUser, backfillResults, initialize } = require('./game');
```

```
beforeEach(() => { localStorage.clear(); // clear localStorage before each test // Reset the HTML body before
each test document.body.innerHTML = <p id="user"></p> <select id="rikishi"> <option
value="1">Rikishi 1</option> <option value="2">Rikishi 2</option> </select> <input
id="userSwitch" type="text"> <input id="backfillContest" type="text"> <input
id="backfillRikishi" type="text"> ; initialize(); });
```

```
test('check if startPlaying is defined and returns expected value', () => { localStorage.setItem('user', 'testUser');
localStorage.setItem('testUser', JSON.stringify({ 'July 2023': '1' })); // Assume a previous pick
document.querySelector('#rikishi').value = '1'; const result = startPlaying(); expect(result).toBe("You selected:
1\nPrevious Picks: {"July 2023":"1"}"); });
```

```
// More tests for switchUser() and backfillResults()...
```

Most Possible Source Code

```
window.startPlaying = startPlaying;
window.switchUser = switchUser;
window.backfillResults = backfillResults;
window.initialize = initialize;
```

Test Code

```
const jsdom = require('jsdom');
const { JSDOM } = jsdom;

const dom = new JSDOM('<!doctype html><html><body></body></html>', {
  url: "http://localhost/",
  resources: "usable",
  runScripts: "dangerously",
});

global.window = dom.window;
global.document = dom.window.document;
global.localStorage = dom.window.localStorage;

// Now we are requiring the whole script, not destructuring it
const Game = require('./game');

let game;

beforeEach(() => {
  localStorage.clear(); // clear localStorage before each test
  // Reset the HTML body before each test
  document.body.innerHTML = `
```



```

    <p id="user"></p>
    <select id="rikishi">
      <option value="1">Rikishi 1</option>
      <option value="2">Rikishi 2</option>
    </select>
    <input id="userSwitch" type="text">
    <input id="backfillContest" type="text">
    <input id="backfillRikishi" type="text">
  `;

  game = new Game(); // Initialize a new game for each test
});

test('check if startPlaying is defined and returns expected value', () => {
  localStorage.setItem('user', 'testUser');
  localStorage.setItem('testUser', JSON.stringify({ 'July 2023': '1' })); //
  Assume a previous pick
  document.querySelector('#rikishi').value = '1';
  const result = game.startPlaying();
  expect(result).toBe("You selected: 1\nPrevious Picks: {\"July 2023\": \"1\"}");
});

// More tests for switchUser() and backfillResults()...

```

Other Code Blocks

```

// module.exports = { startPlaying, switchUser, backfillResults, initialize };

```

```

function startPlaying() {
  var rikishi = document.querySelector('#rikishi').value;
  var user = getUser();
  var picks = getPicks(user);
  var message = "You selected: " + rikishi + "\nPrevious Picks: " +
  JSON.stringify(picks);
  updatePicks(user, rikishi); // Update the picks with the new selection
  return message;
}

function getUser() {
  // get user from local storage
  var user = localStorage.getItem('user');
  if (!user) {
    user = 'admin';
    localStorage.setItem('user', user);
  }
  return user;
}

function getPicks(user) {

```

```
    var picks = JSON.parse(localStorage.getItem(user));
    if (!picks) {
        picks = {};
    }
    return picks;
}

function updatePicks(user, rikishi) {
    var picks = getPicks(user);
    var currentContest = new Date().getMonth();
    if ([0, 2, 4, 6, 8, 10].includes(currentContest)) {
        var contestName = new Date().toLocaleString('default', { month: 'long' })
+ ' ' + new Date().getFullYear();
        picks[contestName] = rikishi;
        localStorage.setItem(user, JSON.stringify(picks));
    }
}

function switchUser() {
    var newUser = document.querySelector('#userSwitch').value;
    localStorage.setItem('user', newUser);
    document.querySelector('#user').textContent = 'Current user: ' + newUser;;
}

function backfillResults() {
    var user = getUser();
    var contestName = document.querySelector('#backfillContest').value;
    var rikishi = document.querySelector('#backfillRikishi').value;
    var picks = getPicks(user);
    picks[contestName] = rikishi;
    localStorage.setItem(user, JSON.stringify(picks));
}

function initialize() {
    var user = getUser();
    var userElement = document.querySelector('#user');
    if (userElement) {
        userElement.textContent = 'Current user: ' + user;
    }
}

initialize();

// Add the functions to the window object
window.startPlaying = startPlaying;
window.switchUser = switchUser;
window.backfillResults = backfillResults;
window.initialize = initialize;
```

Debugging Console Errors

Language

javascript

Last User Prompt

Add test coverage and fix as needed

```
game.test.js import 'jest-localstorage-mock'; import Game from './game.js';
```

```
let game;
```

```
beforeEach(() => { localStorage.clear(); // set localStorage values localStorage.setItem('user', 'testUser');  
localStorage.setItem('testUser', JSON.stringify({ 'July 2023': '1' }));
```

```
document.body.innerHTML = `  
  <p id="user"></p>  
  <select id="rikishi">  
    <option value="1">Rikishi 1</option>  
    <option value="2">Rikishi 2</option>  
  </select>  
  <input id="userSwitch" type="text">  
  <input id="backfillContest" type="text">  
  <input id="backfillRikishi" type="text">  
`;  
game = new Game();
```

```
});
```

```
test('check if startPlaying is defined and returns expected value', () => {  
  document.querySelector('#rikishi').value = '1'; const result = game.startPlaying(); expect(result).toBe("You  
selected: 1\nPrevious Picks: {\"July 2023\":\"1\"}"); });
```

```
// More tests for switchUser() and backfillResults()...
```

Most Possible Source Code

```
startPlaying() {  
  var rikishi = document.querySelector('#rikishi').value;  
  var picks = this.getPicks();  
  var message = "You selected: " + rikishi + "\nPrevious Picks: " +  
  JSON.stringify(picks);  
  this.updatePicks(rikishi); // Update the picks with the new selection  
  this.provideFeedback(message); // Provide feedback  
}  
  
switchUser() {  
  var newUser = document.querySelector('#userSwitch').value;  
  localStorage.setItem('user', newUser);  
  document.querySelector('#user').textContent = 'Current user: ' + newUser;  
  this.user = newUser;  
  this.provideFeedback('Switched user to ' + newUser); // Provide feedback
```

```

}

backfillResults() {
  var contestName = document.querySelector('#backfillContest').value;
  var rikishi = document.querySelector('#backfillRikishi').value;
  var picks = this.getPicks();
  picks[contestName] = rikishi;
  localStorage.setItem(this.user, JSON.stringify(picks));
  this.provideFeedback('Backfilled results for ' + contestName + ' with ' +
    rikishi); // Provide feedback
}

```

Test Code

```

test('check if switchUser is defined and changes the user', () => {
  document.querySelector('#userSwitch').value = 'newUser';
  game.switchUser();
  expect(localStorage.getItem('user')).toBe('newUser');
});

test('check if backfillResults is defined and backfills results', () => {
  document.querySelector('#backfillContest').value = 'August 2023';
  document.querySelector('#backfillRikishi').value = '2';
  game.backfillResults();
  let expectedResults = { 'July 2023': '1', 'August 2023': '2' };
  expect(JSON.parse(localStorage.getItem('testUser'))).toEqual(expectedResults);
});

```

Other Code Blocks

```

backfillResults() {
  var contestName = document.querySelector('#backfillContest').value;
  var rikishi = document.querySelector('#backfillRikishi').value;
  var picks = this.getPicks();
  picks[contestName] = rikishi;
  localStorage.setItem(this.user, JSON.stringify(picks));
  this.provideFeedback('Backfilled results for ' + contestName + ' with ' +
    rikishi); // Provide feedback
  this.displayBackfilledResults(); // Display the updated results
}

```

```

<!DOCTYPE html>
<html>
<head>
  <title>Banzuke Surfing Game</title>
  <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

```

```

    <!-- Link to the external CSS file -->
    <!-- Uncomment this if you have styles to include -->
    <!-- <link rel="stylesheet" href="styles.css"> -->
</head>
<body>
  <h1>Welcome to Banzuke Surfing Game!</h1>
  <p id="user"></p>
  <p>Select your Rikishi and start playing!</p>
  <select id="rikishi">
    <option value="1">Rikishi 1</option>
    <option value="2">Rikishi 2</option>
    <!-- more options here -->
  </select>
  <button id="startPlayingButton">Start Playing</button>
  <hr>
  <h2>Admin Panel</h2>
  <p>Switch user:</p>
  <input id="userSwitch" type="text">
  <button id="switchUserButton">Switch User</button>
  <p>Backfill contest results:</p>
  <input id="backfillContest" type="text" placeholder="Enter contest name">
  <input id="backfillRikishi" type="text" placeholder="Enter rikishi selection">
  <button id="backfillResultsButton">Backfill Results</button>
  <!-- Link to the external JavaScript file -->
  <script type="module" src="game.js"></script>
</body>
</html>

```

Restructuring Code for User

Language

javascript

Last User Prompt

ok i lied about the order of operations. I have populated your work into user.js. please write user.test.js here is the content of game.test.js

```
import 'jest-localstorage-mock'; import Game from './game.js';
```

```
let game;
```

```
beforeEach(() => { localStorage.clear(); // set localStorage values
localStorage.setItem('user', 'testUser');
localStorage.setItem('testUser', JSON.stringify({ 'July 2023': '1' }));
```

```

document.body.innerHTML = `
  <p id="user"></p>
  <select id="rikishi">
    <option value="1">Rikishi 1</option>
    <option value="2">Rikishi 2</option>

```

```

    </select>
    <button id="startPlayingButton">Start Playing</button>
    <button id="switchUserButton">Switch User</button>
    <input id="userSwitch" type="text">
    <input id="backfillContest" type="text">
    <input id="backfillRikishi" type="text">
    <button id="backfillResultsButton">Backfill Results</button>
    <div id="backfilledResults"></div>
    <p id="feedback"></p>
  `;
  game = new Game();

```

```

});

```

```

test('check if startPlaying is defined and returns expected value', () => {
  document.querySelector('#rikishi').value = '1'; const result = game.startPlaying(); expect(result).toBe("You
  selected: 1\nPrevious Picks: {"July 2023": "1"}"); });

```

```

test('check if switchUser updates the user in UI', () => { document.querySelector('#userSwitch').value =
  'newUser'; game.switchUser(); expect(document.querySelector('#user').textContent).toBe('Current user:
  newUser'); });

```

```

test('check if backfillResults updates the results in UI', () => { document.querySelector('#backfillContest').value
  = 'May 2023'; document.querySelector('#backfillRikishi').value = '2'; game.backfillResults(); let resultsElement =
  document.querySelector('#backfilledResults'); expect(resultsElement.textContent).toContain('May 2023: 2'); });

```

Most Possible Source Code

```

export class User {
  constructor(name) {
    this.name = name;
  }

  getPicks() {
    const picks = JSON.parse(localStorage.getItem(this.name));
    return picks || {};
  }

  updatePicks(rikishi) {
    const picks = this.getPicks();
    const currentContest = new Date().getMonth();
    if ([0, 2, 4, 6, 8, 10].includes(currentContest)) {
      const contestName = new Date().toLocaleString('default', { month:
      'long' }) + ' ' + new Date().getFullYear();
      picks[contestName] = rikishi;
      localStorage.setItem(this.name, JSON.stringify(picks));
    }
  }

  backfillResults(contestName, rikishi) {

```

```

        const picks = this.getPicks();
        picks[contestName] = rikishi;
        localStorage.setItem(this.name, JSON.stringify(picks));
    }
}

export default class Game {
    constructor() {
        this.user = new User(this.getUser());
    }

    // Rest of the code...
    // Replace every instance of "this.user" to "this.user.name" except for the
    constructor
    // Replace every instance of "this.getPicks()" to "this.user.getPicks()"
    // Replace every instance of "this.updatePicks(rikishi)" to
    "this.user.updatePicks(rikishi)"
    // Replace every instance of "this.backfillResults()" to
    "this.user.backfillResults(contestName, rikishi)"
}

```

Test Code

```

import 'jest-localstorage-mock';
import User from './user.js';

let user;

beforeEach(() => {
    localStorage.clear();
    // set localStorage values
    localStorage.setItem('user', 'testUser');
    localStorage.setItem('testUser', JSON.stringify({ 'July 2023': '1' }));

    document.body.innerHTML = `
        <p id="user"></p>
        <select id="rikishi">
            <option value="1">Rikishi 1</option>
            <option value="2">Rikishi 2</option>
        </select>
        <button id="startPlayingButton">Start Playing</button>
        <button id="switchUserButton">Switch User</button>
        <input id="userSwitch" type="text">
        <input id="backfillContest" type="text">
        <input id="backfillRikishi" type="text">
        <button id="backfillResultsButton">Backfill Results</button>
        <div id="backfilledResults"></div>
        <p id="feedback"></p>
    `;
    user = new User('testUser');
});

```

```
test('check if user is defined and picks are correct', () => {
  const picks = user.getPicks();
  expect(picks).toEqual({ 'July 2023': '1' });
});

test('check if switchUser updates the user correctly', () => {
  user.switchUser('newUser');
  expect(localStorage.getItem('user')).toBe('newUser');
});

test('check if backfillResults updates the results correctly', () => {
  user.backfillResults('May 2023', '2');
  let picks = user.getPicks();
  expect(picks['May 2023']).toBe('2');
});
```

Other Code Blocks

```
export default class User {
  constructor(name) {
    this.name = name || this.getStoredUser();
  }

  getStoredUser() {
    let user = localStorage.getItem('user');
    if (!user) {
      user = 'admin';
      localStorage.setItem('user', user);
    }
    return user;
  }

  getPicks() {
    const picks = JSON.parse(localStorage.getItem(this.name));
    return picks || {};
  }

  updatePicks(rikishi) {
    const picks = this.getPicks();
    const currentContest = new Date().getMonth();
    if ([0, 2, 4, 6, 8, 10].includes(currentContest)) {
      const contestName = new Date().toLocaleString('default', { month:
'long' }) + ' ' + new Date().getFullYear();
      picks[contestName] = rikishi;
      localStorage.setItem(this.name, JSON.stringify(picks));
    }
  }

  backfillResults(contestName, rikishi) {
    const picks = this.getPicks();
```



```

        picks[contestName] = rikishi;
        localStorage.setItem(this.name, JSON.stringify(picks));
    }

    switchUser(newUser) {
        localStorage.setItem('user', newUser);
        this.name = newUser;
    }

    displayBackfilledResults() {
        const picks = this.getPicks();
        const resultsElement = document.querySelector('#backfilledResults');

        // Clear previous results
        resultsElement.textContent = '';

        // Display each contest result
        for (const contest in picks) {
            const rikishi = picks[contest];
            const resultText = document.createTextNode(contest + ': ' + rikishi);
            const resultDiv = document.createElement('div');
            resultDiv.appendChild(resultText);
            resultsElement.appendChild(resultDiv);
        }
    }
}

```

```

export class User {
    constructor(name) {
        this.name = name;
    }

    getPicks() {
        const picks = JSON.parse(localStorage.getItem(this.name));
        return picks || {};
    }

    updatePicks(rikishi) {
        const picks = this.getPicks();
        const currentContest = new Date().getMonth();
        if ([0, 2, 4, 6, 8, 10].includes(currentContest)) {
            const contestName = new Date().toLocaleString('default', { month:
'long' }) + ' ' + new Date().getFullYear();
            picks[contestName] = rikishi;
            localStorage.setItem(this.name, JSON.stringify(picks));
        }
    }

    backfillResults(contestName, rikishi) {
        const picks = this.getPicks();
        picks[contestName] = rikishi;
    }
}

```

```
        localStorage.setItem(this.name, JSON.stringify(picks));
    }
}

export default class Game {
    constructor() {
        this.user = new User(this.getUser());
    }

    // Rest of the code...
    // Replace every instance of "this.user" to "this.user.name" except for the
    constructor
    // Replace every instance of "this.getPicks()" to "this.user.getPicks()"
    // Replace every instance of "this.updatePicks(rikishi)" to
    "this.user.updatePicks(rikishi)"
    // Replace every instance of "this.backfillResults()" to
    "this.user.backfillResults(contestName, rikishi)"
}
```