Full Stack Cat App IX

Times Played Button / Screen

* Let's add a button underneath the reset button in the rest-of-site.ejs view. Let's also later use the same button on our profile "modal" view. It will say something like "How many times have I played?" and return two bits of information, the times played that the score has changed (games) and the times played that are only matches (matches). In order to do this, we will need to modify the modal to record the matches as well.

* In models/GamePlayer.js copy and paste the times_played field and change the name to games_played. The times_played field will record the matches and the games_played will record the games.

````
const GamePlayerSchema = new mongoose.Schema({
    cat_id: {
        type: String,
        required: true,
        trim: true
    },
    user_id: {
        type: String,
        required: true,
        trim: true
    },
    score: {
        type: Number,
        required: true,
        default: 0,
        trim: true
    },
    times_played: {
        type: Number,
        required: true,
        default: 0,
        trim: true
    },
    games_played: {
        type: Number,
        required: true,
        default: 0,
        trim: true
````

```
    },
```

* Next, let's pass this information along to rest-of-site.ejs, but in a hidden input. We will need to do this so that we can pass it to app.js. In app.js we will be able to update the information as well as the database. We will not need to render the data (although we easily could) inside rest-of-site.ejs. Instead, we will build a new view that will open when the button is clicked.
* The two ways into rest-of-site.ejs are the routes/site/game.js router.get'/game/game/:id' route and the router.post'/game/' route. In the gamePlayer.findOne() extract the games_played and pass it in both the renders of each route (each route has two renders). Here is the GET route:

```
  router.get('/game/:id', ensureAuthenticated, (req, res) => {
    let id = req.params.id;
    CatFancier.findById({ _id: id }, (err, data) => {
        if (err) {
            console.log(err);
        } else {
            let name = data.name;
            let age = data.age;
            let id = data._id;
            let fci = data.favoriteCatImg;
            let user_id = data.user_id;
            GamePlayer.findOne({ cat_id : id }, (err, data) => {
                if (err) {
                    console.log(err);
                } else {
                    if (data) {
                        let player_id = data._id;
                        let score = data.score;
                        let times_played = data.times_played;
                        let games_played = data.games_played;
                        res.render('layouts/site/rest-of-site.ejs', { name, id, age, fci, user_id, player_id,
score, times_played, games_played });
                    } else if (!data) {
                        let gamePlayer = new GamePlayer({
                            cat_id : id,
                            user_id : user_id
                        });
                        gamePlayer.save((err, data) => {
                            if (err) {
                                console.log(err);
```

```
                        } else {
                            let player_id = data._id;
                            let score = data.score;
                            let times_played = data.times_played;
                            let games_played = data.games_played;
                            res.render('layouts/site/rest-of-site.ejs', { name, id, age, fci, user_id,
player_id, score, times_played, games_played });
                        }
                    });
                }
            }
        });
    }
  });
});
```

* times_played is already passed to app.js, but now let's pass games_played. First we have to make a hidden input for it on rest-of-site.ejs. We can copy and paste the times_played input and modify it accordingly. Here is rest-of-site.ejs, I have moved all the hidden divs to the top, for clarity:

```
<div hidden id="age" data-value="<%= age %>"></div>
<div hidden id="id" data-value="<%= id %>"></div>
<div hidden id="player_id" data-value="<%= player_id %>"></div>
<div hidden id="times_played" data-value="<%= times_played %>"></div>
<div hidden id="games_played" data-value="<%= games_played %>"></div>
<div class="profile-bar-div">
  <%- include('./partials/profile-bar.ejs') %>
</div>
<div class="portal-container site-container" style="background-image:url(<%= fci %>);">
  <div class="portal-content-wrapper game-content-wrapper">
    <div class="portal-content game-content">
      <div id="gameDiv" class="game-div">
```

* In app.js let's grab this value, again, just copy and paste. console.log() the result so we can see later that it works. Also change timesPlayed and gamesPlayed to 'let' so that we can change the value. Later we will move them to the let section of variables:

````
const timesPlayedDiv = document.getElementById("times_played");
let timesPlayed = timesPlayedDiv.getAttribute('data-value');
console.log("timesPlayed: " + timesPlayed);
const gamesPlayedDiv = document.getElementById("games_played");
let gamesPlayed = gamesPlayedDiv.getAttribute('data-value');
console.log("gamesPlayed: " + gamesPlayed);

````

Next, In app.js in lose() and win() add timesPlayed ++ . Similarly, in matchScores() add gamesPlayed ++ after the score-- and the score++ :

````

function lose() {
    losses++;
    timesPlayed++;
    console.log("timesPlayed: " + timesPlayed);
    lossesTextP.innerText = "Losses: " + losses;
    matchScores();
    setUpGame();
}

function win() {
    wins++;
    timesPlayed++;
    console.log("timesPlayed: " + timesPlayed);
    winsTextP.innerText = "Wins: " + wins;
    matchScores();
    setUpGame();
}

function matchScores() {
    if (losses === 3) {
        score--;
        gamesPlayed++;
        console.log("gamesPlayed: " + gamesPlayed);
        updateScore(score);
        wins = 0;
        losses = 0;
        winsTextP.innerText = "Wins: " + wins;
        lossesTextP.innerText = "Losses: " + losses;
```

```
    } else if (wins === 3) {
        score++;
        gamesPlayed++;
        console.log("gamesPlayed: " + gamesPlayed);
        updateScore(score);
        wins = 0;
        losses = 0;
        winsTextP.innerText = "Wins: " + wins;
        lossesTextP.innerText = "Losses: " + losses;
    }
}

````

* Before we make routes to update the database, let's see if the variables update when we play the game. Play the game for a while, and watch the console.
* If everything is working, let's make POST routes for gamesPlayed and timesPlayed to capture that information.
* In routes/site/game.js copy and past the router.post'/updateScore/:id' twice. Change them to update the gamesPlayed and the timesPlayed:

````
router.post('/updateTimesPlayed/:id', ensureAuthenticated, (req, res) => {
    let playerId = req.params.id;
    let timesPlayed = req.body.timesPlayed;
    GamePlayer.findByIdAndUpdate({ _id : playerId }, { $set:{ times_played : timesPlayed } }, {
new: true }, (err,data) => {
        if (err) {
            console.log(err);
        } else {
            let cat_id = data.cat_id;
            res.redirect('/restOfSite/game/game/' + cat_id);
        }
    });
});

router.post('/updateGamesPlayed/:id', ensureAuthenticated, (req, res) => {
    let playerId = req.params.id;
    let gamesPlayed = req.body.gamesPlayed;
    GamePlayer.findByIdAndUpdate({ _id : playerId }, { $set:{ games_played : gamesPlayed } }, {
new: true }, (err,data) => {
        if (err) {
            console.log(err);
```

```
        } else {
            let cat_id = data.cat_id;
            res.redirect('/restOfSite/game/game/' + cat_id);
        }
    });
});

````

* While we are here, let's make a new file in routes/site called updateGame.js . Move the updateScore, updateTimesPlayed and updateGames played into that file. In site/index.js make the necessary changes:

````
const router = require('express').Router();
const gameRoutes = require('./game');
const updateGameRoutes = require('./updateGame');

router.use('/game', gameRoutes);
router.use('/updateGame', updateGameRoutes);


module.exports = router;

````

* As well as to the axios route in app.js. Once the axios route is changed, copy and paste it for timesPlayed and gamesPlayed:

````
function updateTimesPlayed(timesPlayed) {
    let newTimesPlayed = timesPlayed;
    return axios.post('/restOfSite/updateGame/updateTimesPlayed/' + playerId, {
        timesPlayed: newTimesPlayed
    })
    .then((data) => {
        if (data.status === 200){
            console.log("newTimesPlayed: " + newTimesPlayed);
        } else {
            console.log("there has been a problem");
        }
    })
    .catch((err) => console.log(err));
```

```
}

function updateGamesPlayed(gamesPlayed) {
    let newGamesPlayed = gamesPlayed;
    return axios.post('/restOfSite/updateGame/updateGamesPlayed/' + playerId, {
        gamesPlayed: newGamesPlayed
    })

    .then((data) => {
        if (data.status === 200){
            console.log("newGamesPlayed: " + newGamesPlayed)
        } else {
            console.log("there has been a problem");
        }
    })
    .catch((err) => console.log(err));
}

function updateScore(score) {
    let newScore = score;
    return axios.post('/restOfSite/updateGame/updateScore/' + playerId, {
        score: newScore
    })
    .then((data) => {
        if (data.status === 200){
            return scoreDBDiv.innerText = "Score: " + score;
        } else {
            console.log("there has been a problem");
        }
    })
    .catch((err) => console.log(err));
}


````
* Remove the console.logs in win() lose() and matchScores() in app.js and call the respective functions in their place. In the .then() the console.log should still execute while playing.

````
function win() {
    wins++;
    timesPlayed++;
    updateTimesPlayed(timesPlayed);
```

```
    winsTextP.innerText = "Wins: " + wins;
    matchScores();
    setUpGame();
}
```

```
* Test these routes out for a while, and if everything is good, as always, save to GitHub.

%%%%%%%%%%%%%%%%%% Artistic change of direction
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

* Originally, I was going to make a button on rest-of-site.ejs, but instead, let's make a counter and put it above the score div.
* There will still be a button on third-page.ejs (the profile page).
* Rename rest-of-site.ejs to rest-of-site.html so that it will be easier to work in.
* Make a wrapper containing two <p>, one for times played and the other for games played.
* Rename rest-of-site.ejs:

```

                    <p id="score" data-value="<%= score %>">
                    <button id="resetBtn" class="btn reset-btn">Reset</button>

```

```
* In app.js target the two  tags with document.getElementById()  and set the innerText to "Matches played: " + timesPlayed and "Games played: " + gamesPlayed, respectively:

```
const timesPlayedDiv = document.getElementById("times_played");
const timesPlayedText = document.getElementById("timesPlayedText");
const gamesPlayedDiv = document.getElementById("games_played");
const gamesPlayedText = document.getElementById("gamesPlayedText");

```

* Get rid of the console.log() s in the variable section and move the let timesPlayed = and let gamesPlayed = down to the let variables, with let score = . In setUpGame() set the innerText :

````
    timesPlayedText.innerText = "Matches played: " + timesPlayed;
    gamesPlayedText. innerText = "Games played: " + gamesPlayed;
    winsTextP.innerText = "Wins: " + wins;
    lossesTextP.innerText = "Losses: " + losses;
    scoreDBDiv.innerText = "Score: " + score;
    playerNumber.innerText = "Player Number: " + playerNum;
    targetNumber.innerText = "Game Number: " + targetNum;

````

* finally in updateTimesPlayed() and updateGamesPlayed() change the .then() console.log() to render the gamesPlayedText.innerText and the timesPlayedText.innerText:

````
function updateGamesPlayed(gamesPlayed) {
    let newGamesPlayed = gamesPlayed;
    return axios.post('/restOfSite/updateGame/updateGamesPlayed/' + playerId, {
        gamesPlayed: newGamesPlayed
    })

    .then((data) => {
        if (data.status === 200){
            gamesPlayedText. innerText = "Games played: " + gamesPlayed;
        } else {
            console.log("there has been a problem");
        }
    })
    .catch((err) => console.log(err));
}

````

* Adjust the css and make sure that all the modal links go to the right paths. especially how-to-ejs (as the paths have moved) and then save to GitHub.

** Finally, the button!

* On third-page.ejs rename the file to '.html' and make a button that says "How many times have I played this game?". Link the button to '/restOfSite/updateGame/getTimesPlayed/' + <%= id

%>. (This route does not exist yet). Rename the file '.ejs' and make any css adjustments. The button will not work yet (because the route does not exist).

* Let's make the route! Go to routes/site/updateGame.js and make a GET route called getTimesPlayed/:id. The route should call on CatFancier since we have the CatFancier _id and we need to extract the name and the favorite cat image. So also CatFancier will need to be imported into updateGame.js . Then we can do GamePlayer findOne using the cat_id = id . We can then take out the times_played and the games_played and pass them and the name into the render of times-played.ejs (which does not exist yet.) ALSO if there is a new user the route will throw an error because they can push the button before they have a GamePlayer profile. To avoid this, test to see if there is data. If there is no data, new GamePlayer.save().

````
router.get('/getTimesPlayed/:id', ensureAuthenticated, (req, res) => {
    let id = req.params.id;
    CatFancier.findById({ _id : id }, (err, data) => {
        if (err) {
            console.log(err);
        } else {
            let name = data.name;
            let fci = data.favoriteCatImg;
            let user_id = data.user_id;
            GamePlayer.findOne({ cat_id: id }, (err, data) => {
                if (err) {
                    console.log(err);
                } else if (data) {
                    let timesPlayed = data.times_played;
                    let gamesPlayed = data.games_played;
                    let id = data.cat_id;
                    res.render('layouts/site/times-played.ejs', {
                        name, fci, id, gamesPlayed, timesPlayed
                    });
                } else if (!data) {
                    let gamePlayer = new GamePlayer({
                        cat_id : id,
                        user_id : user_id
                    });
                    gamePlayer.save((err, data) => {
                        if (err) {
                            console.log(err);
                        } else {
                            let timesPlayed = 0;
                            let gamesPlayed = 0;
````

```
                        res.render('layouts/site/times-played.ejs', { name, id, fci, timesPlayed,
gamesPlayed });
                    }
                });
            }
        });
    }
  });
});

````

* Make the layouts/site/times-played.ejs file and type in: my name is <%= name %>, my id is <%= id %>, matches are <%= timesPlayed %> and games are <%= gamesPlayed %>.
* See if the page renders with all the variables correct.
* To keep the styling consistent, copy and past name.ejs into times-played.ejs, keeping all the div intact but changing the content to indicate the users name and how many matches and how many games they have played. The closing X should be to href="/profile/third-page/<%= id %>", BUT there still is no GET route to there, only a POST route.
* Again, for ease of formatting change the name to .html rather than .ejs, and after you have it configured, change it back.

* Make a GET route for the button. Copy and paste the routes/profile/profile.js POST 'third-page/:id' route. CAREFULLY remove the CatFancier.findByIdAndUpdate() because you do not want to update the cat picture. Also remove the const src = . Change the post to a get and you should be good.

CAREFULLY go back through all the routes and check that the paths line up. Take your time. Push all buttons and make sure everything works! AND you are done! --As soon as you save to GitHub!