# Front End Technologies Week 6 Coding Assignment

**Points possible:** 70

|  |  |  |
| --- | --- | --- |
| Category | Criteria | % of Grade |
| Functionality | Does the code work? | 25 |
| Organization | Is the code clean and organized? Proper use of white space, syntax, and consistency are utilized. Names and comments are concise and clear. | 25 |
| Creativity | Student solved the problems presented in the assignment using creativity and out of the box thinking. | 25 |
| Completeness | All requirements of the assignment are complete. | 25 |

**Instructions:** In VS Code, or an IDE of your choice, write the code that accomplishes the objectives listed below. Ensure that the code compiles and runs as directed. Take screenshots of the code and of the running program (make sure to get screenshots of all required functionality) and paste them in this document where instructed below. **Create a new repository on GitHub for this week’s assignments and push this document, with your JavaScript project code, to the repository. Add the URL for this week’s repository to this document where instructed and submit this document to your instructor when complete**.

**Coding Steps:**

1. **As a team, create a full CRUD application of your choice.** If you can use an existing API, use AJAX to interact with it. However, you do not have to use an API. If you do not use an API, store the entities you will create, read, update, and delete in an array.
   1. **Application should have at least 1 entity per team member**
   2. **Use a form to add new entities**
   3. **Build a way for users to update or delete entities**
   4. **Use Bootstrap and CSS to style your project**

**Screenshots of Code:**

**index\_hmwk\_week6.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Final Project Week6</title>

    <link rel="stylesheet" href="node\_modules/bootstrap/dist/css/bootstrap.css">

    <link rel="stylesheet" href="main.css">

</head>

<body class="container flex mx-auto rounded border border-primary m-5">

    <br><div id = "new-unicorn" class="jumbotron">

        <div class="container flex rounded border border-primary">

            <h1>Unicorn List</h1>

        </div>

        <br><h2>Create Unicorn</h2>

        <input type= "text" id="new-unicorn-name" class="form-control" placeholder="Unicorn Name"> <br>

        <input type= "text" id="new-unicorn-color" class="form-control" placeholder="Unicorn Color"> <br>

        <div class="col-md-3">

            <label for="new-unicorn-region" class="form-label">Region</label>

            <select class="form-select" id="new-unicorn-region" required>

              <option selected disabled value="">Choose...</option>

              <option>North</option><option>South</option><option>East</option>

              <option>West</option>

            </select>

            <div class="invalid-feedback">

              Please choose a region.

            </div><br>

        <button id="create-new-unicorn" class="btn btn-primary form-control">Create Unicorn</button>

    </div>

    <div id="app">

    </div>

    <script src="node\_modules/jquery/dist/jquery.min.js"></script>

    <script src="Index-hmwk\_week6.js"></script>

    <script src="node\_modules/bootstrap/dist/js/bootstrap.bundle.js"></script>

</body>

</html>

**Index\_hmwk\_week6.js**

class Unicorn {

    constructor(name, color, region) {

        this.name = name;

        this.color = color;

        this.region = region;

        this.powers = [];

    }

    addPower(name, effect, damage, strikes) {

        this.powers.push(new Power(name, effect, damage, strikes));

    }

}

class Power {

    constructor(name, effect, damage, strikes) {

        this.name = name;

        this.effect = effect;

        this.damage = damage;

        this.strikes = strikes;

    }

}

class UnicornService {

    static url =  "https://crudcrud.com/api/467446f2b41f4ea58e1a553fd9faaf72/unicorns";

    static getAllUnicorns() {

        return $.get(this.url);

    }

    static getUnicorn(id) {

        return $.get(this.url + `/${id}`);

    }

    static createUnicorn(unicorn) {

        return $.ajax({

            url: this.url,

            dataType: 'json',

            contentType: 'application/json',

            data: JSON.stringify(unicorn),

            type: 'POST'

        });

    }

    static updateUnicorn(unicorn) {

        return fetch(  `${this.url}/${unicorn.\_id}`, {

            method: 'PUT',

            headers : new Headers ({

                    'Content-Type': 'application/json'

            }),

            body: JSON.stringify({"name" : unicorn.name, "color" : unicorn.color, "region" : unicorn.region, "powers" : unicorn.powers}),

        });

    }

    static deleteUnicorn(id) {

        return $.ajax({

            url:  `${this.url}/${id}`,

            type: 'DELETE'

        });

    }

}

class DOMManager {

    static unicorns;

    static getAllUnicorns() {

        UnicornService.getAllUnicorns().then(unicorns => this.render(unicorns));

    }

    static createUnicorn(name,color,region) {

        console.log(`Creating a unicorn named: ${name}!`);

        UnicornService.createUnicorn(new Unicorn (name,color,region))

            .then(() => {

                return UnicornService.getAllUnicorns();

            })

            .then((unicorns) => this.render(unicorns));

    }

    static deleteUnicorn(id) {

        console.log(`Deleting a unicorn!`);

        UnicornService.deleteUnicorn(id)

            .then(() => {

                return UnicornService.getAllUnicorns();

            })

            .then((unicorns) => this.render(unicorns));

    }

    static addPower(id) {

        for (const unicorn of this.unicorns) {

            if (unicorn.\_id == id) {

                unicorn.powers.push(new Power($(`#${unicorn.\_id}-power-name`).val(), $(`#${unicorn.\_id}-power-effect`).val(), $(`#${unicorn.\_id}-power-damage`).val(), $(`#${unicorn.\_id}-power-strikes`).val()));

                console.log('Adding power:' + $(`#${unicorn.\_id}-power-name`).val());

                UnicornService.updateUnicorn(unicorn)

                    .then(() => {

                        return DOMManager.getAllUnicorns();

                    });

            }

        }

    }

    static deletePower(unicornId, powerName) {

        for (const unicorn of this.unicorns) {

            if (unicorn.\_id == unicornId) {

                for (let i = 0; i < unicorn.powers.length; i++) {

                    const power = unicorn.powers[i];

                    if (power.name == powerName) {

                        unicorn.powers.splice(i, 1);

                        console.log('Deleting power: ' + powerName);

                        UnicornService.updateUnicorn(unicorn)

                            .then(() => {

                                return DOMManager.getAllUnicorns();

                            });

                    }

                }

            }

        }

    }

    static decrementPower(unicornId, powerName) {

        for (const unicorn of this.unicorns) {

            if (unicorn.\_id == unicornId) {

                for (let i = 0; i < unicorn.powers.length; i++) {

                    const power = unicorn.powers[i];

                    if (power.name == powerName) {

                        if (unicorn.powers[i].strikes == 0) {

                            console.log(`No power: ${unicorn.powers[i].name} out of stock!`)

                        } else {

                            unicorn.powers[i].strikes -= 1;

                            console.log('Decremented ' + powerName + ' strikes.  New total: ' + `${unicorn.powers[i].strikes}`);

                            UnicornService.updateUnicorn(unicorn)

                                .then(() => {

                                    return DOMManager.getAllUnicorns();

                                });

                        }

                    }

                }

            }

        }

    }

    static incrementPower(unicornId, powerName) {

        for (const unicorn of this.unicorns) {

            if (unicorn.\_id == unicornId) {

                for (let i = 0; i < unicorn.powers.length; i++) {

                    const power = unicorn.powers[i];

                    if (power.name == powerName) {

                        unicorn.powers[i].strikes++;

                        console.log('Incremented ' + powerName + ' strikes.  New total: ' + `${unicorn.powers[i].strikes}`);

                        UnicornService.updateUnicorn(unicorn)

                            .then(() => {

                                return DOMManager.getAllUnicorns();

                            });

                    }

                }

            }

        }

    }

    static render(unicorns) {

        this.unicorns = unicorns;

        $('#app').empty();

        for (let unicorn of unicorns) {

            $('#app').prepend(

                `

                <br><div id="${unicorn.\_id}" class="card border border-primary">

                    <div class="card-header">

                        <h2>${unicorn.name}</h2>

                        <h6> Region:  ${unicorn.region}</h6>

                        <button class="btn btn-danger" onclick="DOMManager.deleteUnicorn('${unicorn.\_id}')">Delete Unicorn</button>

                    </div>

                    <div class="card-body">

                        <div class="card">

                            <div class="row">

                                <div class="col-sm center">

                                    <input type="text" id="${unicorn.\_id}-power-name" class="form-control" placeholder="Unicorn Power"> <br>

                                </div>

                                <div class="col-sm">

                                    <input type="text" id="${unicorn.\_id}-power-effect" class="form-control" placeholder="Power Effect"><br>

                                </div>

                                <div class="col-sm">

                                    <input type="text" id="${unicorn.\_id}-power-damage" class="form-control" placeholder="Damage Points"><br>

                                </div>

                                <div class="col-sm">

                                    <input type="text" id="${unicorn.\_id}-power-strikes" class="form-control" placeholder="Number of Power Strikes"><br>

                                </div>

                            </div>

                            <br>

                            <button id="${unicorn.\_id}-new-power" onclick="DOMManager.addPower('${unicorn.\_id}')" class="btn btn-primary form-control">Add Power</button>

                        </div>

                    </div>

                </div><br>`

            );

            if (unicorn.powers == null) {

                console.log(`Power list for unicorn: ${unicorn.name} is empty!`);

            } else {

                $(`#${unicorn.\_id}`).find('.card-body').append(`<br>`);

                for (const power of unicorn.powers) {

                    $(`#${unicorn.\_id}`).find('.card-body').append(

                        `<p>

                            <span id="name-${power.name}"><strong>Unicorn Power: </strong> ${power.name}</span>

                            <span id="effect-${power.name}"><strong>Power Effect: </strong> ${power.effect}</span><br>

                            <span id="damage-${power.name}"><strong>Damage Points: </strong> ${power.damage}</span><br>

                            <span id="strikes-${power.name}"><strong>Number of Strikes: </strong> ${power.strikes}&nbsp;&nbsp;</span>

                            <button id="${unicorn.\_id}-${power.name}-increase number of strikes" onclick="DOMManager.incrementPower('${unicorn.\_id}', '${power.name}')"  class="btn btn-success">Increase Strikes</button>

                            <button id="${unicorn.\_id}-${power.name}-decrease number of strikes" onclick="DOMManager.decrementPower('${unicorn.\_id}', '${power.name}')"  class="btn btn-warning">Decrease Strikes</button>

                            <button id="${unicorn.\_id}-${power.name}-delete-power" onclick="DOMManager.deletePower('${unicorn.\_id}', '${power.name}')"  class="btn btn-danger">Delete Power</button>&nbsp;&nbsp;

                            `

                    );

                }

            }

        }

    }

}

$('#create-new-unicorn').on('click', () => {

    console.log("New Unicorn!");

    DOMManager.createUnicorn($('#new-unicorn-name').val(), $('#new-unicorn-color').val(), $('#new-unicorn-region').val());

    $('#new-unicorn-name').val('');

    $('#new-unicorn-color').val('');

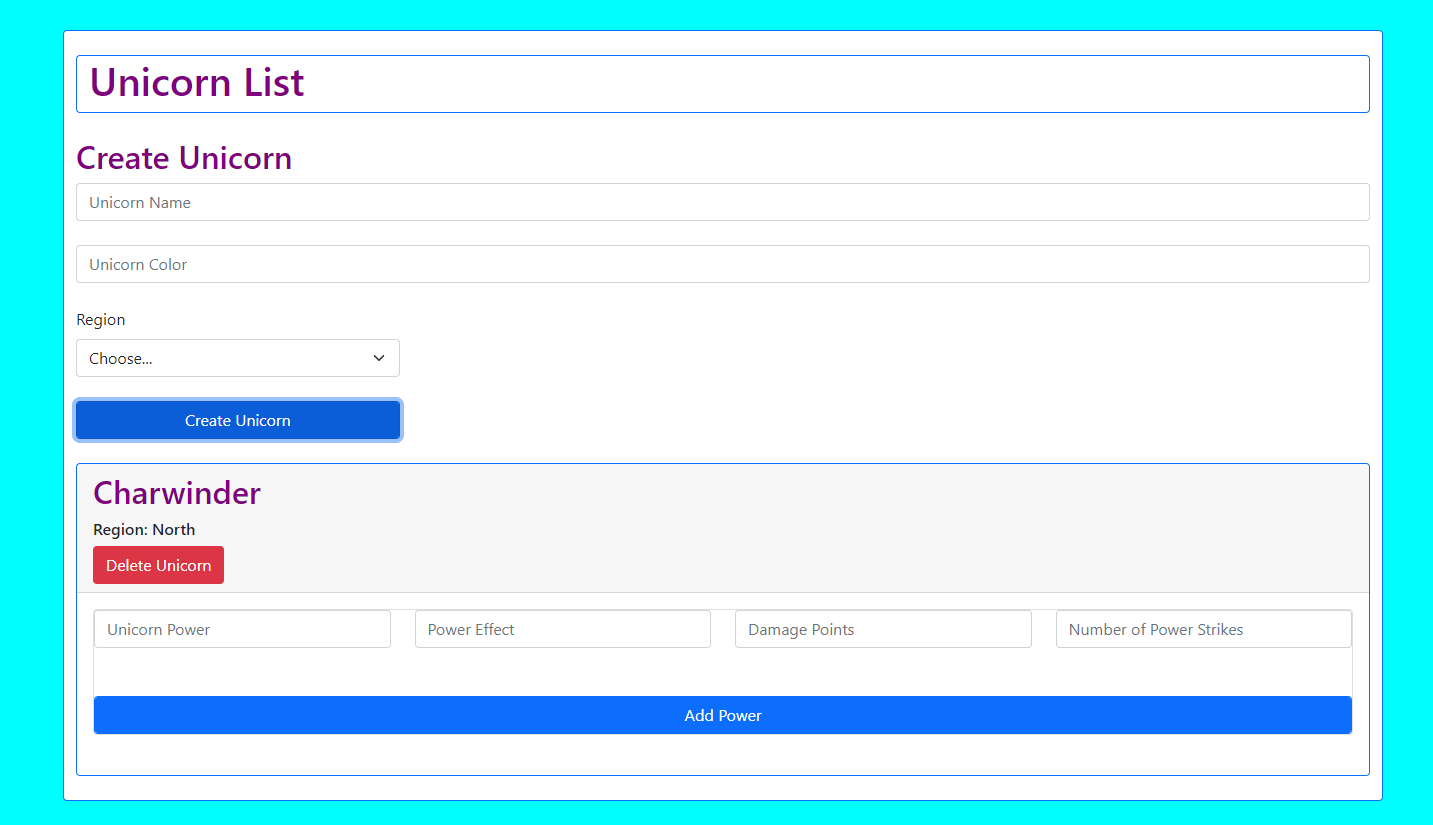
    $('#new-unicorn-region').val('');

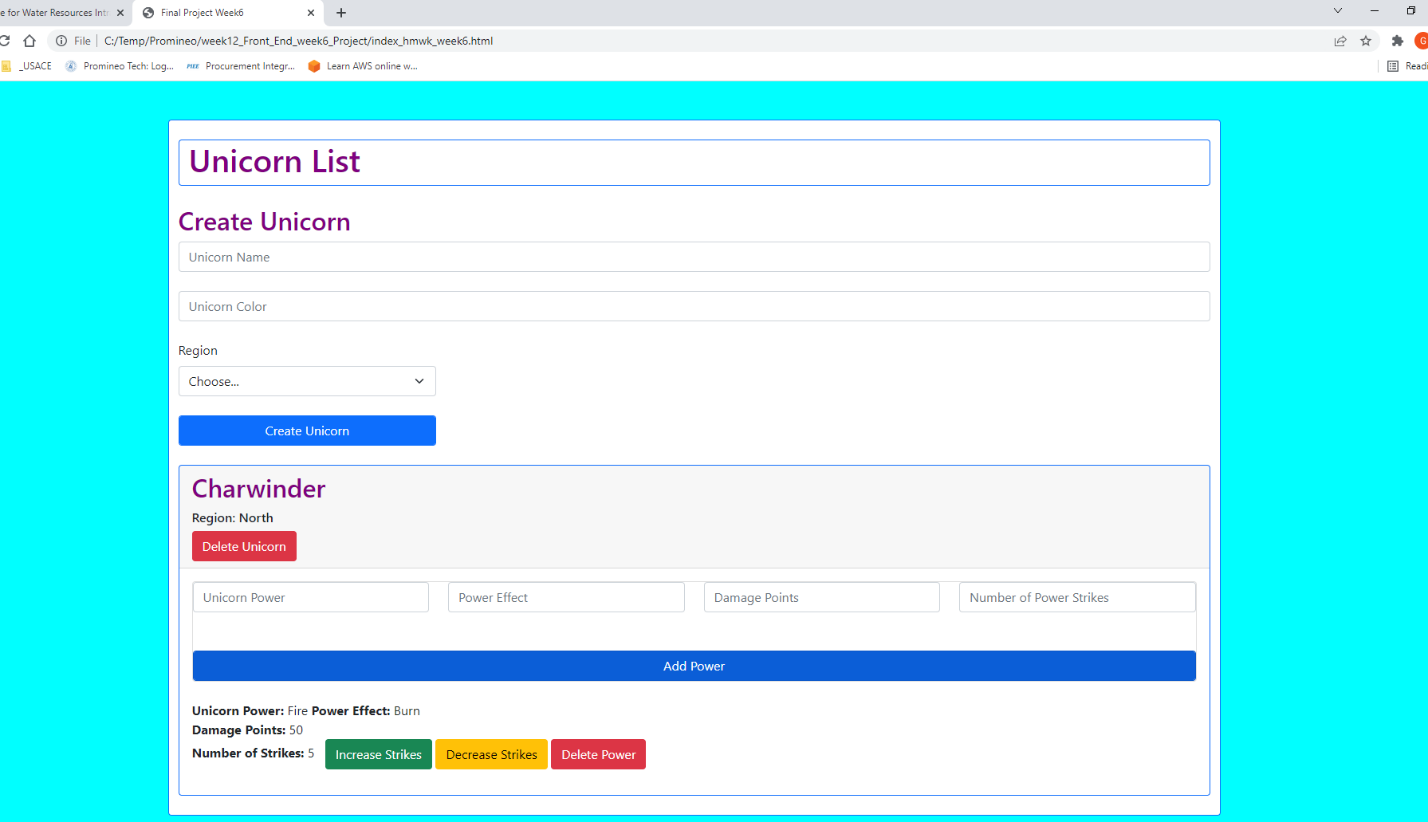
});

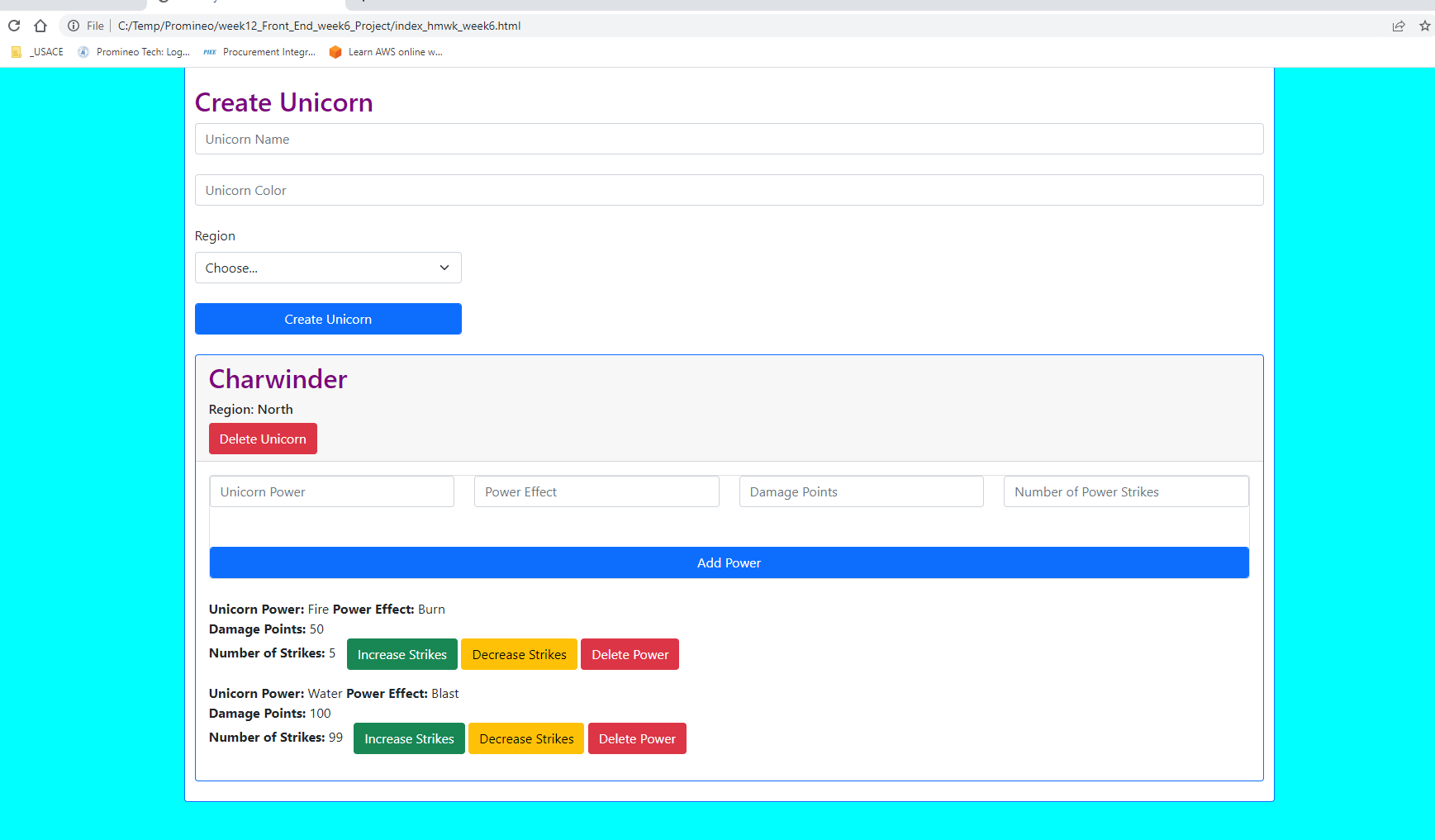
DOMManager.getAllUnicorns();

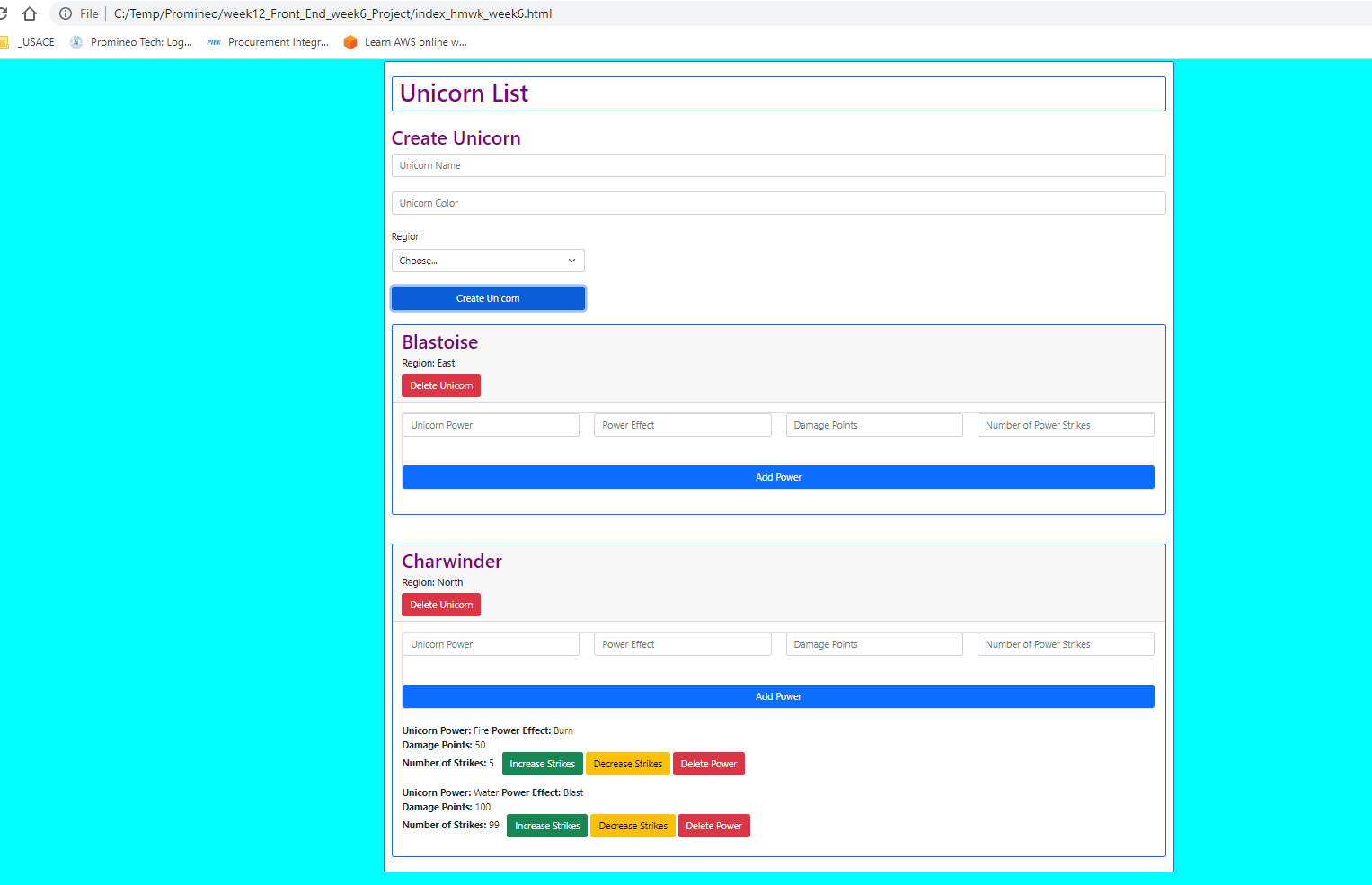
**Screenshots of Running Application:**

****

****

****

****

****

**URL to GitHub Repository:**

[**https://github.com/emcis/https-github.com-emcis-https-github.com-emcis-week\_12\_repository**](https://github.com/emcis/https-github.com-emcis-https-github.com-emcis-week_12_repository)