✨ MiracleDex Project Report

Student Name: John Clements
Course: CSCI 270 – Web/Mobile App Development
Project Title: MiracleDex
Submission Date: 06/11/2025
GitHub Repo: https://github.com/JClements123/MiracleDex
Live Site (GitHub Pages): https://jclements123.github.io/MiracleDex/

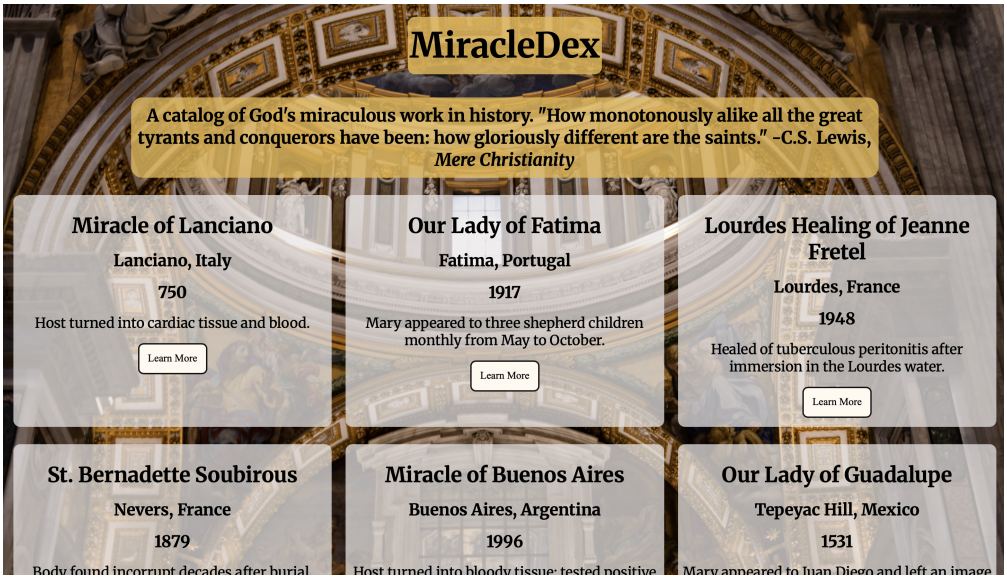
Note: I used Google's AI overview to learn how to turn this document into a PDF.

🔷 Part 1 — HTML/CSS Layout and Responsive Design

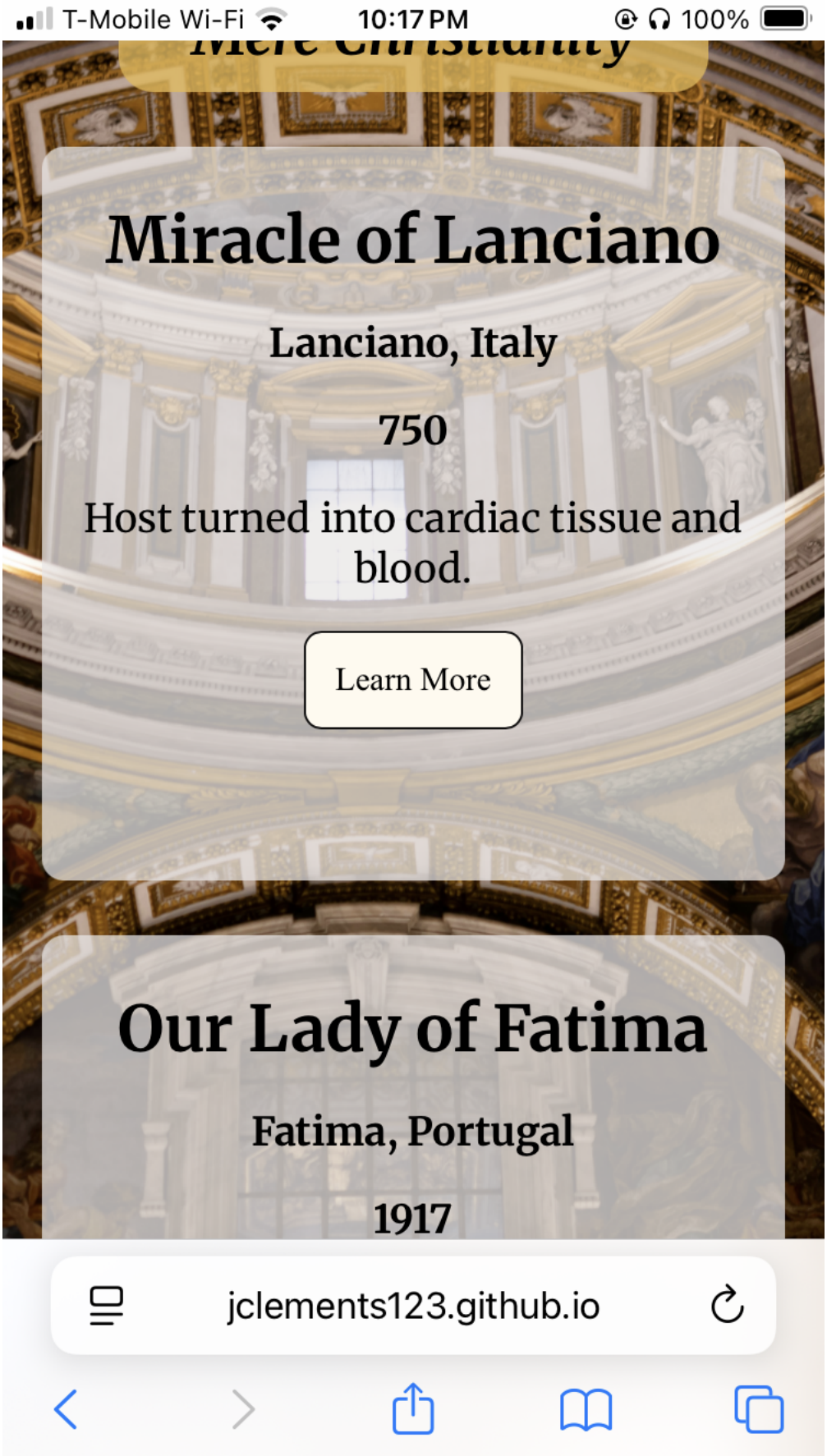
✅ Requirements Addressed

- [ ] Landing page created with heading and container
- [ ] Used Flexbox or Grid layout
- [ ] Mobile-friendly design with media queries
- [ ] Elegant, reverent styling with a Google Font
- [ ] External CSS file used

📸 Screenshots



Desktop layout of my MiracleDex website.



Phone layout of my MiracleDex website.

🔍 Code Snippets

> Include relevant HTML and CSS snippets here (layout structure, responsive styling, etc.).

HTML Layout Structure:

```
<main>
    <article id="article">
        <!-- ChatGPT helped me to learn about overlay elements. I wanted to learn how to darken the background
        behind the modal that I brought up. Using a div was suggested to me. ChatGPT also helped with the CSS.

        OpenAI. (2025). Darkening Background Behind Modal. ChatGPT (Version 4o). Retrieved from https://chatgpt.c
        <div id="overlay"></div>
        <!-- ChatGPT was utilized to suggest how to format a modal. A div was suggested to me.
        I went ahead and added my own close button after ChatGPT suggested that one should be included in a modal
        ChatGPT was utilized in understanding the basics of modals, and it also helped in generating the CSS for

        OpenAI. (2025). What is a Modal. ChatGPT (Version 4o). Retrieved from https://chatgpt.com. -->
        <div class="modal" id="modal">
            <button class="close-modal" id="close-modal" style="display:inline;">Close</button>
        </div>
        <!--End of modal that ChatGPT helped generate -->
        <!-- ChatGPT was utilized to ensure that this citation would be sufficient for a website.
         The quote was selected by myself, and I wrote the description. -->
        <!-- ChatGPT also aided me in suggesting to use the <em> tag around the book title. -->
        <!-- OpenAI. (2025). Set image background CSS. ChatGPT (Version 4o). Retrieved from https://chatgpt.com.
        <h3 class="sub-header">A catalog of God's miraculous work in history. "How monotonously alike
            all the great tyrants and conquerors have been: how gloriously different are the saints."
            -C.S. Lewis,
            <em>Mere Christianity</em>
        </h3>
        <!-- End of section where ChatGPT assisted me as noted above with the subtitle. -->
        <div class="card-grid" id="card-grid"></div>
        <button class="load-more" id="load-more">Load More Miracles</button>
    </article>
</main>
```

This screenshot demonstrates the overall layout of my app based on the HTML that I coded without the help of JavaScript.

CSS Responsive Design and Media Queries:

```
.card-grid {
    margin-left: 10px;
    margin-right: 10px;
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(400px, 1fr));
    gap: 20px;
}
```

A screenshot demonstrating my use of Grid in order to generate a responsive design.

```css
@media (max-width: 768px) {
    .card-grid {
        display: flex;
        flex-direction: column;
    }

    .miracle-card * {
        font-size: 18px;
    }

    .miracle-card .card-title {
        font-size: 29px;
    }

    .miracle-card:hover .card-title {
        font-size: 30px;
    }
}
/* End of section where ChatGPT was utilized as noted above */
```

```css
@media (max-width: 768px) {
    body::before {
        background-size: 250%; /* Zoomed-out effect */
    }
}
```

The two screenshots above demonstrate my use of media queries in order to create both a desktop-friendly and mobile-friendly design. See CSS file for ChatGPT usage in media queries.

🪶 Reflection

What challenges did you face in designing a responsive layout? What did you learn about structuring HTML/CSS for real-world use?

Designing a responsive layout had easy aspects and more difficult aspects. It was very easy to implement the use of CSS Grid. There are only a few lines of CSS that you have to write to allow your cards to easily adapt to the layout of your website and have everything feel very organized.

The biggest challenge for me was making sure that my mobile layout would look just as beautiful as the layout for the computer. I ran into some challenges with this such as struggling with making sure that my media query would do what I needed it to do. The image background for the phone was very blurry and would shift around a lot when I loaded new cards, so I utilized ChatGPT to help me learn how to make my background just as clear as it was on my computer.

I found that structuring HTML for real-world use is actually very simple. You can set up a few divs here and there to act as containers for all of your contents, and your HTML actually becomes very simple. When you

are working with the JavaScript DOM, you can generate a lot of your HTML through JavaScript and not have it hard-coded into your HTML file. I actually wrote very little HTML for this project in my `index.html` file.

I found that structuring my CSS took a lot more work than building up the HTML. The CSS was where I ran into the most problems with struggling to make my website beautiful and responsive. I found that there were so many things that you have to take into account to let your website look good on several different layouts. I learned that you just have to keep testing and debugging as you write a lot of CSS to target each of the elements on your webpage.

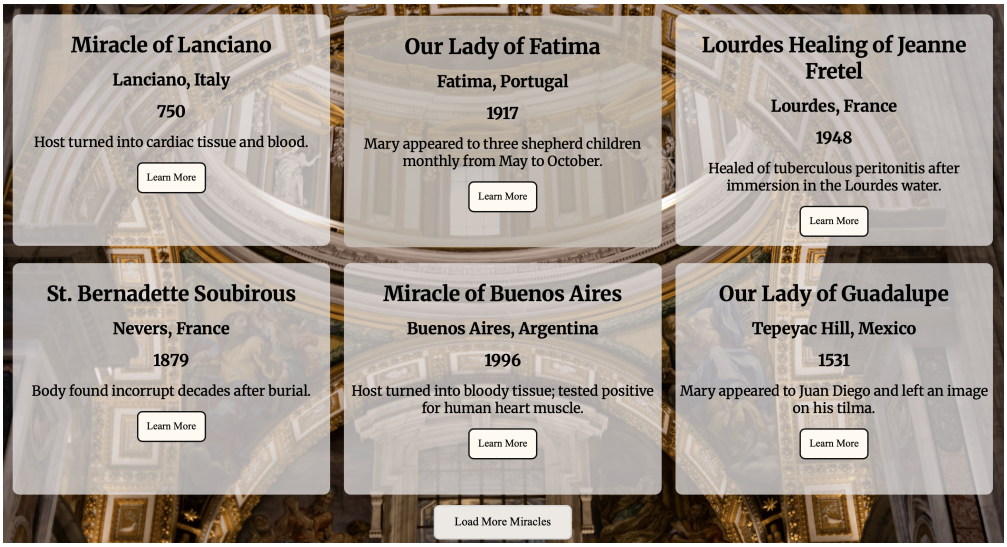
🔷 Part 2 — JavaScript + DOM + JSON Integration

✅ Requirements Addressed

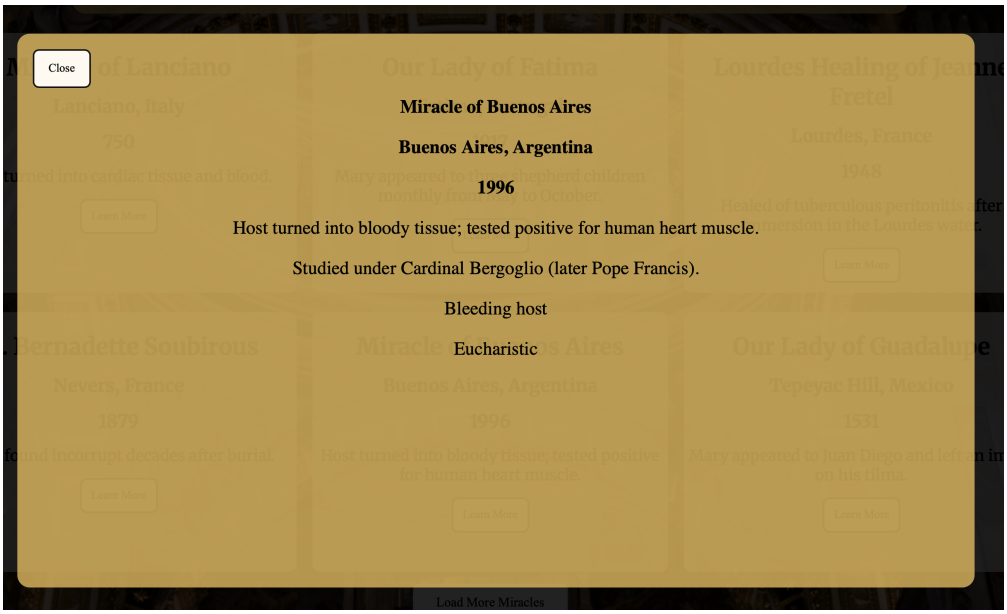
- [ ] Fetched data from: `https://gist.githubusercontent.com/trevortomesh/7bbf97b2fbae96639ebf1a254b6a7a70/raw/miracles.json`
- [ ] Rendered miracle title, location, year, summary
- [ ] Used `fetch()` and `async/await`
- [ ] Implemented "Load More" or pagination
- [ ] Added modal or expandable section with full miracle details

📸 Screenshots

> Show your miracle cards and one expanded view or modal.



A screenshot showing my miracle cards.



A screenshot showing the modal for one of my miracle cards.

🔍 Code Snippets

> Include the function you used to fetch the data, render the cards, and handle interaction.

```
// This is my async function that fetches the data from the JSON file
async function showMiracles() : Promise<void> { Show usages  John Clements
    try {
        const response : Response = await fetch( input: 'https://gist.githubusercontent.com/trevortomesh/7bbf97b
        if (!response.ok) throw new Error('HTTP error', response.status);
        const miracles = await response.json();
        display(miracles);
    } catch(error) {
        console.log('Fetch error:', error);
    }
}
```

Asynchronous function for fetching the data.

```
// This is my function that will help dynamically generate and display the miracle cards
function display(miracles) : void  {  Show usages  John Clements
    const card_grid : HTMLElement  = document.getElementById( elementId: 'card-grid');
    for (let i : number  = 0; i < 6; i++) {
        let div : HTMLDivElement  = document.createElement( tagName: 'div');
        div.className = 'miracle-card';
        card_grid.appendChild(div);
        div.id = 'card' + miracles[i].id;
        let title : HTMLHeadingElement  = document.createElement( tagName: 'h2');
        title.textContent = miracles[i].title;
        title.className = 'card-title';
        let title_id : string  = 'title' + miracles[i].id;
        title.id = title_id;
        // ChatGPT utilized for debugging in order to properly append my created h2 element. I was forgetting to
        // say appendChild() and was instead just updating the innerHTML.
        // OpenAI. (2025). Add text to h2. ChatGPT (Version 4o). Retrieved from https://chatgpt.com.
        div.appendChild(title);
        // End of section where ChatGPT was utilized
        let location_id : string  = 'location' + miracles[i].id;
        let location : HTMLHeadingElement  = document.createElement( tagName: 'h3');
        location.textContent = miracles[i].location;
        location.className = 'card-location';
        location.id = location_id;
        div.appendChild(location);
        let year_id : string  = 'year' + miracles[i].id;
        let year : HTMLHeadingElement  = document.createElement( tagName: 'h3');
        year.textContent = miracles[i].year;
        year.className = 'card-year';
        year.id = year_id;
        div.appendChild(year);
        let summary_id : string  = 'summary' + miracles[i].id;
        let summary : HTMLParagraphElement  = document.createElement( tagName: 'p');
        summary.textContent = miracles[i].summary;
        summary.className = 'card-summary';
        summary.id = summary_id;
```

```
        summary.id = summary_id;
        div.appendChild(summary);
        let button_id : string  = 'button' + miracles[i].id;
        let button : HTMLButtonElement  = document.createElement( tagName: 'button');
        button.innerHTML = '';
        button.textContent = 'Learn More';
        button.className = 'card-button';
        button.id= button_id;
        div.appendChild(button);
        document.getElementById(button.id).addEventListener( type: 'click',  listener: () : void  => {
            const overlay : HTMLElement  = document.getElementById( elementId: 'overlay');
            overlay.style.display = 'block';
            const modal : HTMLElement  = document.getElementById( elementId: 'modal');
            modal.style.display = 'block';
            let modal_title : HTMLHeadingElement  = document.createElement( tagName: 'h2');
            modal_title.id = 'modal_title' + miracles[i].id;
            modal_title.textContent = miracles[i].title;
            modal_title.style.textAlign = 'center';
            modal_title.style.marginTop = '10px';
            modal.appendChild(modal_title);
            let modal_location : HTMLHeadingElement  = document.createElement( tagName: 'h3');
            modal_location.id =  'modal_location' + miracles[i].id;
            modal_location.textContent = miracles[i].location;
            modal_location.style.textAlign = 'center';
            modal.appendChild(modal_location);
            let modal_year : HTMLHeadingElement  = document.createElement( tagName: 'h3');
            modal_year.id =  'modal_year' + miracles[i].id;
            modal_year.textContent = miracles[i].year;
            modal_year.style.textAlign = 'center';
            modal.appendChild(modal_year);
            let modal_summary : HTMLParagraphElement  = document.createElement( tagName: 'p');
            modal_summary.id =  'modal_summary' + miracles[i].id;
            modal_summary.textContent = miracles[i].summary;
            modal_summary.style.textAlign = 'center';
```

```
        modal_summary.style.textAlign = 'center';
        modal.appendChild(modal_summary);
        let details : HTMLParagraphElement = document.createElement( tagName: 'p');
        details.textContent = miracles[i].details;
        details.className = 'card-details';
        let details_id : string = 'details' + miracles[i].id;
        details.id = details_id;
        details.style.textAlign = 'center';
        modal.appendChild(details);
        let category : HTMLParagraphElement = document.createElement( tagName: 'p');
        category.textContent = miracles[i].category;
        category.className = 'card-category';
        let category_id : string = 'category' + miracles[i].id;
        category.id = category_id;
        category.style.textAlign = 'center';
        modal.appendChild(category);
        let type : HTMLParagraphElement = document.createElement( tagName: 'p');
        type.textContent = miracles[i].type;
        type.className = 'card-type';
        let type_id : string = 'type' + miracles[i].id;
        type.id = type_id;
        type.style.textAlign = 'center';
        modal.appendChild(type);
        const close : HTMLElement = document.getElementById( elementId: 'close-modal');
        close.addEventListener( type: 'click', listener: () : void => {
            overlay.style.display = 'none';
            modal.style.display = 'none';
            modal_title.remove();
            modal_location.remove();
            modal_year.remove();
            modal_summary.remove();
            category.remove();
            type.remove();
            details.remove();
```

```
        })
    })
}
const loadMore : HTMLElement = document.getElementById( elementId: 'load-more');
loadMore.addEventListener( type: 'click', listener: () : void => {
    let count : number = card_grid.children.length;
    for (let i : number = count; i < (count + 6); i++) {
        if (i >= miracles.length) {
            loadMore.remove();
            break;
        }
        let div : HTMLDivElement = document.createElement( tagName: 'div');
        div.className = 'miracle-card';
        card_grid.appendChild(div);
        div.id = 'card' + miracles[i].id;
        let title : HTMLHeadingElement = document.createElement( tagName: 'h2');
        title.textContent = miracles[i].title;
        title.className = 'card-title';
        let title_id : string = 'title' + miracles[i].id;
        title.id = title_id;
        // ChatGPT utilized for debugging in order to properly append my created h2 element. I was forgettin
        // say appendChild() and was instead just updating the innerHTML.
        // OpenAI. (2025). Add text to h2. ChatGPT (Version 4o). Retrieved from https://chatgpt.com.
        div.appendChild(title);
        // End of section where ChatGPT was utilized
        let location_id : string = 'location' + miracles[i].id;
        let location : HTMLHeadingElement = document.createElement( tagName: 'h3');
        location.textContent = miracles[i].location;
        location.className = 'card-location';
        location.id = location_id;
        div.appendChild(location);
        let year_id : string = 'year' + miracles[i].id;
```

```
        let year_id : string  = 'year' + miracles[i].id;
        let year : HTMLHeadingElement  = document.createElement( tagName: 'h3');
        year.textContent = miracles[i].year;
        year.className = 'card-year';
        year.id = year_id;
        div.appendChild(year);
        let summary_id : string  = 'summary' + miracles[i].id;
        let summary : HTMLParagraphElement  = document.createElement( tagName: 'p');
        summary.textContent = miracles[i].summary;
        summary.className = 'card-summary';
        summary.id = summary_id;
        div.appendChild(summary);
        let button_id : string  = 'button' + miracles[i].id;
        let button : HTMLButtonElement  = document.createElement( tagName: 'button');
        button.textContent = 'Learn More';
        button.className = 'card-button';
        button.id= button_id;
        div.appendChild(button);
        document.getElementById(button.id).addEventListener( type: 'click',  listener: () : void  => {
            const overlay : HTMLElement  = document.getElementById( elementId: 'overlay');
            overlay.style.display = 'block';
            const modal : HTMLElement  = document.getElementById( elementId: 'modal');
            modal.style.display = 'block';
            let modal_title : HTMLHeadingElement  = document.createElement( tagName: 'h2');
            modal_title.id = 'modal_title' + miracles[i].id;
            modal_title.class = 'modal_title'
            modal_title.textContent = miracles[i].title;
            modal_title.style.textAlign = 'center';
            modal_title.style.marginTop = '10px';
            modal.appendChild(modal_title);
            let modal_location : HTMLHeadingElement  = document.createElement( tagName: 'h3');
            modal_location.id =  'modal_location' + miracles[i].id;
            modal_location.textContent = miracles[i].location;
```

```
            modal_location.id =  'modal_location' + miracles[i].id;
            modal_location.textContent = miracles[i].location;
            modal_location.style.textAlign = 'center';
            modal.appendChild(modal_location);
            let modal_year : HTMLHeadingElement  = document.createElement( tagName: 'h3');
            modal_year.id =  'modal_year' + miracles[i].id;
            modal_year.textContent = miracles[i].year;
            modal_year.style.textAlign = 'center';
            modal.appendChild(modal_year);
            let modal_summary : HTMLParagraphElement  = document.createElement( tagName: 'p');
            modal_summary.id =  'modal_summary' + miracles[i].id;
            modal_summary.textContent = miracles[i].summary;
            modal_summary.style.textAlign = 'center';
            modal.appendChild(modal_summary);
            let details : HTMLParagraphElement  = document.createElement( tagName: 'p');
            details.textContent = miracles[i].details;
            details.className = 'card-details';
            let details_id : string  = 'details' + miracles[i].id;
            details.id = details_id;
            details.style.textAlign = 'center';
            modal.appendChild(details);
            let category : HTMLParagraphElement  = document.createElement( tagName: 'p');
            category.textContent = miracles[i].category;
            category.className = 'card-category';
            let category_id : string  = 'category' + miracles[i].id;
            category.id = category_id;
            category.style.textAlign = 'center';
            modal.appendChild(category);
            let type : HTMLParagraphElement  = document.createElement( tagName: 'p');
            type.textContent = miracles[i].title;
            type.className = 'card-type';
            let type_id : string  = 'type' + miracles[i].id;
            type.id = type_id;
```

```
            type.className = 'card-type';
            let type_id : string = 'type' + miracles[i].id;
            type.id = type_id;
            type.style.textAlign = 'center';
            modal.appendChild(type);
            const close : HTMLElement = document.getElementById( elementId: 'close-modal');
            close.addEventListener( type: 'click', listener: () : void => {
                overlay.style.display = 'none';
                modal.style.display = 'none';
                modal_title.remove();
                modal_location.remove();
                modal_year.remove();
                modal_summary.remove();
                category.remove();
                type.remove();
                details.remove();
            })
        })
    }
})
}

showMiracles();
```

The above seven screenshots show the function that I used to display all of the miracle cards.

🖋️ Reflection

What did you learn about asynchronous JavaScript? What debugging techniques did you use or discover?

I learned that using asynchronous JavaScript is actually very simple. My function that fetches all of the data for my entire website is very short. I call `await fetch()` and `await response.json()` in order to get all of the information that I need to generate my miracle cards. I learned to keep the asynchronous function separate from the function that displays the miracle cards to increase readability of the code. The difficult part of the JavaScript was figuring out how to update the Document Object Model with all of the information that I was retrieving from the JSON file, but the easy part was getting that data. This assignment helped to reinforce what I had already learned about the Fetch API and asynchronous JavaScript in the lectures.

One of the biggest debugging techniques that I utilized throughout this project was using ChatGPT when I was particularly stuck on a problem that I was having. I would often try to solve the problem myself and to spend some time with the problem before utilizing ChatGPT, but ChatGPT came in handy to solve problems that I couldn't immediately sort out.

I also tried to utilize `console.log()` as much as I could, as this helped me to make sure that I had actually retrieved the data from the website. I utilized Chrome DevTools to see where my design was breaking and how my website was coming along. Chrome DevTools is especially helpful for CSS problems, but it also can help you see that updating the DOM changes the structure of your HTML.

🔷 Part 3 — GitHub Repository and Documentation

✅ Requirements Addressed

- ☐ GitHub repo created and pushed
- ☐ GitHub Pages deployed
- ☐ `README.md` contains project description, instructions, and screenshots

📎 Links

- **GitHub Repo:** https://github.com/JClements123/MiracleDex
- **Live GitHub Pages Site:** https://jclements123.github.io/MiracleDex/

🪶 Reflection

How did using GitHub affect your development process? What new Git or GitHub skills did you gain?

GitHub made my development process feel clean and organized. I could slowly update my GitHub repository as I made changes throughout the project. This made me feel like I could keep track of the changes as I was making them. Sometimes I would forget after having not worked on the project for a little while what changes I had made, but I could go back and look at my commit history to check.

I learned how to utilize GitHub Pages to make my website go live for the public. I also continued to practice and improve with using the git commands in my workflow. I sometimes made changes in WebStorm and committed and pushed there. I learned that when I made a new directory to store my screenshots, it didn't seem like I could commit and push the new directory from WebStorm, so I went to the command line to use `git add .`.

🧠 Final Reflection

Imagine you're explaining this project to a friend who doesn't code. What does your app do? What are you most proud of? What was the hardest part to get working?

Reflect on both the technical and the spiritual aspects of building a project about Catholic miracles.

My app gets information from a separate file in order to create a tiled display of cards that teach the user of the website about various miracles throughout history. The website displays a title and subtitle followed by 6 miracle cards at the beginning. The user can click load more to load 6 more cards at a time while there is more data to generate new cards. When the user wants to learn more about a miracle, they can click the Learn More button to read about the details, category, and type of the miracle.

I am most proud of my design for this app. I really tried to generate a beautiful app that would provide a great space for the user to learn more about the miracles. I was glad that I utilized AI to help me learn how to add an image as a background, because I thought it was very cool that I got to add a picture of Saint Peter's Basilica in the background. I think that the website feels clean, uncluttered, and easy to use.

I believe that the hardest part of my website to get working was making it to have a responsive design. I was having trouble with getting the media query to work, and I felt that while the design looked great on the computer, the design was nowhere near as good on the phone. I believe that I was able to debug this well and make it so that there is no sacrifice when the user utilizes their phone rather than their computer.

A lot of work clearly has to go into building a website. It likely took longer because I was just starting to learn how to build a website, but I spent several hours trying to get this website to look as good as possible. There is a lot that goes into building a website even as relatively simple as this MiracleDex site. I believe that this project provides me with the skills I need to start building larger, more complex websites. It was a blessing to get to learn about the miracles as I was building this website. I especially liked learning about the

Miracle of Buenos Aires which was studied under Cardinal Bergoglio, as noted in the card. It's amazing that the host turned into a human heart. It is great to use our technical skills to bear spiritual fruits, and I hope that anyone who sees our MiracleDex websites will also benefit spiritually.