📝 Project 1 — The MiracleDex

**Due: Wed Jun 11, 2025 11:59pmDue: Wed Jun 11, 2025 11:59pm**

Ungraded, 30 Possible Points30 Points Possible

Attempt

In Progress

**NEXT UP: Submit Assignment**

Add Comment

**Unlimited Attempts Allowed**

**Details**

*Blessed Carlo Acutis (1991–2006)*

**“The Eucharist is my highway to Heaven.” – Blessed Carlo Acutis**

This project is inspired by the digital legacy of **Blessed Carlo Acutis**, a teenage programmer, gamer, and devout Catholic who created one of the first major digital archives of Eucharistic miracles. At just 14 years old, he coded a website cataloging miraculous Eucharistic events from around the world, aiming to bring others closer to Christ through the truth and beauty of the Church’s mysteries.

Carlo was beatified in 2020 by Pope Francis, and his canonization was originally scheduled for April 27, 2025, during the Jubilee of Adolescents. However, following the death of Pope Francis on April 21, 2025, the ceremony was postponed. A new date has not yet been announced.

As you begin this project, you are invited to walk in the footsteps of Blessed Carlo Acutis, using your technical gifts for something sacred. Like Carlo, you will use the tools of modern computing not to distract or entertain, but to illuminate and inspire.

The **MiracleDex** is a mobile-friendly, web-based application that presents a catalog of real Catholic miracles — including Eucharistic miracles, Marian apparitions, incorruptible saints, and more. You’ll build this using the core frontend technologies we’ve been learning:

* **HTML** to structure the content
* **CSS** (including Flexbox/Grid and media queries) to style it beautifully and responsively
* **JavaScript** to dynamically fetch and render miracle data from a remote JSON file

📝 Submission Instructions

Use the following Markdown template to write your project report:

**📥 Markdown Template:** [CSCI270-A1-Template.mdLinks to an external site.](https://gist.github.com/trevortomesh/c09eb8afa9f44d1e9b65ae03bfb96d41)

**You must:**

* Download and complete the Markdown template
* Include all screenshots, code snippets, and reflections
* **Convert the Markdown to a PDF** using Typora, Pandoc, VS Code plugin, or another tool
* **Submit the PDF file to Canvas**

---

🤖 On the Use of AI Tools

You are **allowed and encouraged** to use large language models (LLMs) such as ChatGPT, GitHub Copilot, or similar tools to assist you with this assignment — as long as your use complies with the [CSCI 270 LLM Usage GuideLinks to an external site.](https://github.com/trevortomesh/LLM-Guide).

**You must clearly cite:**

* What AI tool you used (e.g., ChatGPT, Gemini, Copilot)
* How you used it (e.g., “Generated initial draft of fetch logic”, “Helped reword README section”)
* Where the output appears in your submission (e.g., “Section 2 code block”)

**Examples of Acceptable Use:**

* Using ChatGPT to help you understand fetch() syntax
* Using Copilot to suggest JavaScript functions that you then edited
* Asking an AI to help you word your reflection or summarize code

**Examples of Unacceptable Use:**

* Submitting AI-generated code you don’t understand (no vibe coding!)
* Copying and pasting an entire solution without modification or citation
* Using AI to write the entire report with no input from you

Plagiarism or dishonest use of AI will be treated as an academic integrity violation.

---

🚨 AI Grading Transparency

Dr. T uses an AI assistant to suggest feedback and identify areas for improvement. **All final grading is done by Dr. T personally.**

If you do **not consent** to your project being reviewed with AI assistance, please write the following at the top of your PDF:

I DO NOT CONSENT TO AI ASSISTED GRADING

---

🔷 Project Overview

You will build a website that fetches miracle data from this live JSON file:

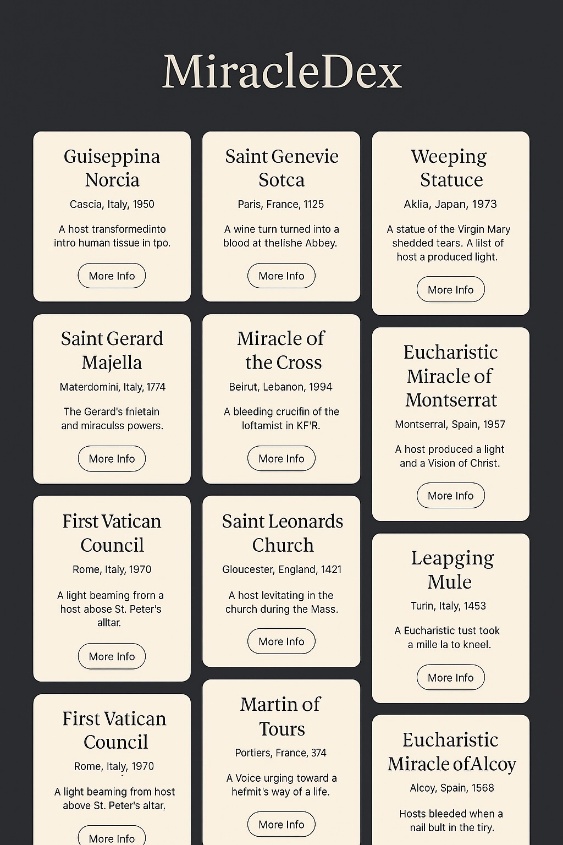
https://gist.githubusercontent.com/trevortomesh/7bbf97b2fbae96639ebf1a254b6a7a70/raw/miracles.json

The data includes Eucharistic miracles, Marian apparitions, healing miracles, and incorrupt saints. Your app will download this data and render each miracle as a “card.” Clicking a card should display more detailed information.

---

🔶 Part 1 – HTML & CSS Layout (10 pts)

**From Lectures: Chapters 2–6**



Mock-up of the MiracleDex interface

This section focuses on building the structure and layout of your MiracleDex web app. You're not just designing a container for data — you're setting the visual tone for the entire experience. Think about reverence, readability, and mobile accessibility.

* **Landing page with heading and subtitle:** Your app should open with a clear, welcoming title like *“MiracleDex”* and a short subtitle such as “A catalog of miraculous signs throughout history.” Use appropriate semantic HTML elements like <header>, <h1>, and <main>.
* **Miracle card container:** Create a <section> or <div> that will hold dynamically generated miracle cards. You'll fill this later with JavaScript, but go ahead and give it structure and style now using a CSS class like .card-grid.
* **Responsive layout:** Use Flexbox or CSS Grid to control how cards are spaced and wrapped. Aim for a multi-column layout on desktop and a single-column stack on smaller screens. (See Lecture 5 for examples.)
* **Media queries:** Add at least one media query to adapt spacing or font size on smaller devices. Make sure your layout looks good on phones — this is required.
* **Typography & color:** Use at least one Google Font to elevate the feel of your app. Choose colors that evoke reverence and harmony — nothing neon or garish. Consider a soft beige background with cards in white or off-white.

If you finish early, consider adding some gentle animations or transitions to card hover states or buttons — but this is optional.

---

🔶 Part 2 – JavaScript + JSON Integration (15 pts)

**Related Lectures:**

* **Chapter 9:** JavaScript Basics (variables, types, expressions)
* **Chapter 10:** Arrays and the DOM
* **Chapter 11:** Loops (especially forEach)
* **Chapter 12:** JavaScript Objects
* **Chapter 13:** The DOM (creating and inserting elements)
* **Chapter 14:** Event Listeners
* **Chapter 15:** AJAX and the Fetch API

**Goal:** Use JavaScript to load miracle data from a remote JSON file and display it interactively.

📡 Step 1 — Fetch the JSON

Use fetch() and async/await to load this file:

https://gist.githubusercontent.com/trevortomesh/7bbf97b2fbae96639ebf1a254b6a7a70/raw/miracles.json

**Example:**

async function loadMiracles() {

const res = await fetch("https://gist.githubusercontent.com/trevortomesh/7bbf97b2fbae96639ebf1a254b6a7a70/raw/miracles.json");

const data = await res.json();

renderMiracles(data);

}

If it fails with a CORS error, download a local copy of the JSON file for testing.

🧱 Step 2 — Render Miracle Cards

Use createElement(), classList, and appendChild() to generate one card per miracle.

**Each card must include:**

* Title
* Location
* Year
* Summary

Cards must be generated dynamically — no hardcoded HTML miracle entries allowed.

🖱️ Step 3 — Modal or Expandable View

Clicking a card should either expand it in place or open a modal with full miracle information.

**Expanded view must include:**

* Details
* Category
* Type

Include a close button if you use a modal.

🔁 Step 4 — "Load More" Button or Pagination

Only show a few cards at first (e.g. 6). Add a button to load more cards in batches.

* Don’t reload the page
* Hide the button when all miracles are visible
* You may use slice() or a counter to manage what’s shown

📌 Other Requirements

* Images (if any) must be styled responsively (e.g. max-width: 100%)
* Card layout must adjust for mobile using CSS Grid or Flexbox
* Use textContent (not innerHTML) for inserting text

❗ Common Pitfalls

* Hardcoding miracles into HTML instead of using JavaScript
* Calling await outside an async function
* Rendering too many miracles at once (lack of pagination)
* Not testing mobile layout in DevTools

🆘 If You Get Stuck

* Revisit lectures 10–15 and live coding demos
* Check the [MDN Fetch GuideLinks to an external site.](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch)
* Ask in class or post a question in Discord

---

🔶 Part 3 – GitHub & Deployment (5 pts)

**From Lectures: Chapters 7–8**

* Initialize a GitHub repo and commit code regularly
* Enable GitHub Pages and deploy your site
* Include a README.md with:
  + Live site link
  + Description of your project
  + Setup instructions
  + Any known issues

---

🧠 Final Reflection

In the Markdown template, you’ll reflect on what you learned—technically and spiritually. What parts of the assignment challenged you? What part are you proud of? Did any miracle stand out to you?

---

🧾 Grading Rubric (30 Points Total)

| **Category** | **Points** | **Criteria** |
| --- | --- | --- |
| 🔶 Part 1 – HTML & CSS Layout | **10 pts** | Quality of structure, semantic HTML usage, responsive layout, and styling |
| 🔶 Part 2 – JavaScript & JSON | **15 pts** | Functionality, proper use of fetch(), dynamic rendering, and interactivity |
| 🔶 Part 3 – GitHub & Deployment | **5 pts** | GitHub usage, live deployment, and inclusion of required documentation |

🔍 Additional Requirements

* **🧠 Final Reflection** – Must be thoughtful and complete to pass the assignment
* **🤖 AI Use Disclosure** – Clearly cite AI use; incomplete or deceptive usage may affect your grade
* **🚫 AI Grading Consent** – To opt out of AI-assisted feedback, include:  
  I DO NOT CONSENT TO AI ASSISTED GRADING at the top of your PDF submission

🛑 Final Reminders

* **Submit only a PDF** generated from the Markdown template
* **Clearly cite any use of AI tools** (see above)
* **Your code must be functional, committed to GitHub, and publicly deployed**