Website Purpose and Theme:

The purpose of this portfolio is to showcase my current skills, projects, and experience in the software development field. As an Integrations Specialist with 3 years of professional experience at Northwestern Polytechnic and a current 3rd-year BSc Computer Science student at Athabasca University, I aim to present my work to potential employers, freelance clients, and collaborators.

The site will evolve as I progress in my education and career, serving as both a static display of past achievements and a dynamic record of new learning. It will highlight programming skills, integration projects, technical blog posts, certifications, and professional goals.

The primary themes are professionalism, growth, clarity, and accessibility. The tone will be formal but friendly, the structure clean and navigable, and the layout responsive and accessible across various devices and browsers.

Personas:

1. Technical Recruiter (David)
    - Demographics
        - Name: David McMan
        - Age: 38
        - Photo:

          (Generated using https://www.vecteezy.com/)
        - Job Title: Senior Technical Recruiter
        - Technical Skill Level: Moderate — understands roles and tech stacks
    - Motivations:
        - Quickly assess my background information and determine my fitness for client positions.
        - Identify my core technologies, work history, and availability
        - Save time by finding essential info fast
    - Frustrations:
        - No downloadable or easy-to-share resume
        - Unclear tech stack or missing job titles
        - No contact link
    - Goals:
        - David will land on my about page and will start by scanning for keywords like frameworks, roles and certificates to assess if I am a fit for the jobs he is recruiting for.

    - Should I meet his expectations, he will then go over to my resume section where he will print the resume and send it to the hiring manager.
    - David will then visit the Contact me page to schedule an interview once the hiring manager approves.

2. **Software Development Team Lead (Angela)**
    - **Demographics**
        - **Name:** Angela Patel
        - **Age:** 35
        - **Photo:**

          (Generated using https://www.vecteezy.com/)
        - **Job Title:** Software Development Team Lead
        - **Technical Skill Level:** Advanced
    - **Motivations:**
        - Assess whether I follow modern development practices and industry standards
        - Review real-world examples of my code and see if I would be a good fit for her team
        - Find out if I'm a proactive, thoughtful developer who documents and reflects on my work
    - **Frustrations:**
        - Vague or shallow project descriptions without technical depth
        - Poorly structured UI/UX
        - Projects that look impressive visually but lack meaningful content or code
    - **Goals:**
        - Amira will first explore my **Projects** section to evaluate the complexity and quality of the work
        - She will dig into project write-ups and view GitHub links to inspect code and documentation
        - She will then review my **About Me** page to determine whether my background and personality align with her team's values and environment
        - If everything checks out, she might bookmark the site for future collaboration or reach out for a technical interview
3. **Computer Science Student (Carlos)**
    - **Demographics**
        - **Name:** Carlos Mendes
        - **Age:** 25
        - **Photo:**

(Generated using https://www.vecteezy.com/)

- **Job Title:** Computer Science Undergraduate Student
- **Technical Skill Level:** Intermediate

- **Motivations:**
  - Learn from the structure and presentation of my personal portfolio
  - Discover how to explain and reflect on coding projects
  - Get inspired by the types of tools and languages I've worked with
- **Frustrations:**
  - Jargon-heavy technical writing with no explanations
  - Lack of source code or context
  - Sites that are not responsive or mobile-friendly
- **Goals:**
  - Carlos will start by browsing the **Projects** section to read summaries and view code examples
  - He'll move on to the **Blog/Reflection** section where I share learning insights and challenges
  - If interested, he'll bookmark the site or return later when building his own portfolio

4. **Small Business Owner (Lori)**
   - **Demographics**
     - **Name:** Lori McAllister
     - **Age:** 29
     - **Photo:**

(Generated using https://www.vecteezy.com/)

- **Job Title:** Small Business Owner
- **Technical Skill Level:** Low

- **Motivations:**
  - Find a capable and professional freelance developer for her business website

    - Understand my services and see proof of work
    - Feel confident that I can communicate clearly and deliver quality products on time

- **Frustrations:**
    - Overly technical language or cluttered layout
    - Lack of clear pricing, offerings, or way to reach out
- **Goals:**
    - Lori will begin at the **Home** or **About Me** page to get a sense of who I am
    - She'll look for clear, easy-to-understand examples of previous work in the **Projects** section
    - If she's impressed, she'll visit the **Contact Me** page to ask for a quote or discuss a possible project
    - A testimonial section or client success stories will help build her confidence in hiring me

Scenarios:

1. David (Recruiter) Quick Candidate Evaluation
   David McMan, a seasoned technical recruiter, is scanning through LinkedIn profiles looking for a developer with integration experience and exposure to cloud platforms when he comes across my name. Curious, he clicks through to my portfolio site. He's on a tight schedule and wants to quickly assess whether I'm a potential fit for a role involving API integrations and cloud services.
   He lands on my **About Me** page, scanning for skills and keywords like "Azure," "REST APIs," and "integrations." Satisfied with the skills, he clicks over to the **Resume** section and schemes over my previous employment history. Before wrapping up, he opens the **Contact Me** page to send a message requesting availability for a screening interview.

2. David (Recruiter) Needs a Printable Resume Now
   It's 9:45 AM, and David has a candidate meeting with a hiring manager in 15 minutes. One of the names he's considering is mine. He quickly pulls up my portfolio site and navigates to the Resume section. He's relieved to find a neatly formatted, downloadable PDF of my resume, which he prints immediately to add to the candidate briefing folder.

3. Angela (Dev Team Lead) Reviewing Code Quality

Amira, a team lead at a fast-paced software company, is vetting candidates for a new integration-focused role. She already has my name from a recruiter but wants to evaluate how I think and code before recommending me for an interview.
She browses the Projects section of my site, opening links to my GitHub repositories. She's impressed by the clear README files, sensible project structure, and concise write-ups about what I built and why. She bookmarks my site to return later with her junior developer for a second opinion.

4. Angela (Dev Team Lead) Evaluating Team Fit
   Later that week, Amira returns to my site not to look at code, but to see if I'd integrate well with her collaborative team. She reads through my About Me section and then opens my Blog, where I write about challenges I've faced, what I've learned from previous roles, and how I collaborate with designers and testers. The personal tone and thoughtful writing reassure her that I'd likely fit well with her team's culture.

5. Lori (Business Owner) Looking for a Freelancer
   Lori, who runs a small shoe business, needs someone to revamp her outdated website. A colleague recommends my name. Lori visits my portfolio and is pleased to see clean design, easy navigation, and relevant projects. She reads through the about me page to understand who she is dealing with.
   She then goes over to the Project section where she focuses on a project where I built a simple e-commerce site, reading how I approached client communication and mobile optimization. Convinced of my reliability, she heads to the Contact Me page to request a quote.

6. Carlos (CS Student) Looking for Portfolio Inspiration
   Carlos, a second-year computer science student, is browsing for portfolio ideas as part of a class project. He stumbles on my site through a student dev blog and spends time exploring every section.
   He's particularly interested in how I've structured my about me page. The tech stack icons, short summaries, and embedded GitHub activity gives him a solid idea of how to build his own. Inspired, he bookmarks the site and shares it with his peers.

Further requirements:

As I develop my personal portfolio, I will make deliberate choices to ensure it meets important ethical, technical, and institutional standards. I will consider color contrast, text alternatives, and keyboard navigation to ensure the site is usable for all visitors. I also want the site to authentically represent who I am as a developer and lifelong learner, while maintaining a professional tone that reflects my values and goals. Since I am currently employed at Northwestern Polytechnic, I will work with my manager and HR to ensure my site remains compliant with institutional

policies and does not result in any conflict of interest. On the technical side, I will design the site to be mobile-responsive, so it works seamlessly across devices. And because the course restricts server-side processing, I will focus on static technologies like HTML, CSS, and JavaScript, without relying on databases or user authentication.

Site Map / Mock-ups

My website is structured to provide a clear, professional, and engaging experience for various visitors, including recruiters, collaborators, and potential clients. The main entry point is the About Me page, which also acts as the homepage. From there, visitors can navigate to dedicated sections that serve specific purposes.

About Me

This is the landing page that gives a strong first impression. It includes:

- A contact card with essential links (LinkedIn, GitHub, email)
- A brief biography and professional background
- A list of technical skills and competencies
- Highlights of recent or featured projects
- (Optional) GitHub activity feed to show ongoing engagement in coding

Projects

This section showcases my personal and professional projects. It includes:

- A project filter (by technology or category)
- A project list/grid with screenshots, descriptions, and links to code or live demos

Resume

This section is structured for quick assessment by recruiters and includes:

- Education history
- Certifications and achievements
- Work experience and roles held
- A button to download the full resume as a PDF

Contact Me

Allows anyone to reach out directly through:

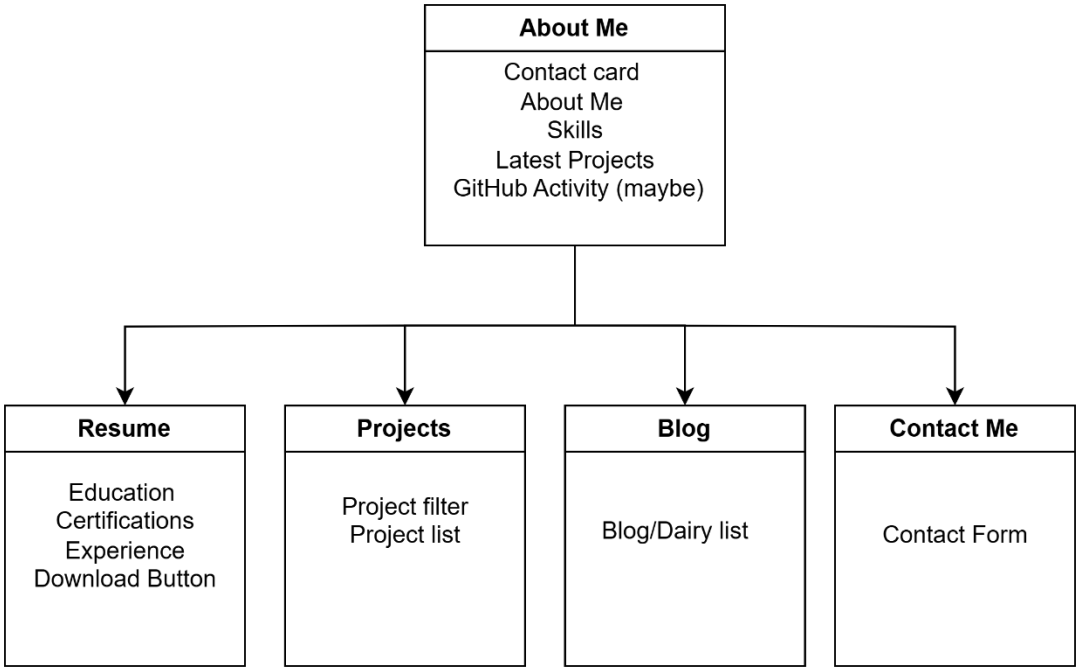
- A simple and accessible contact form

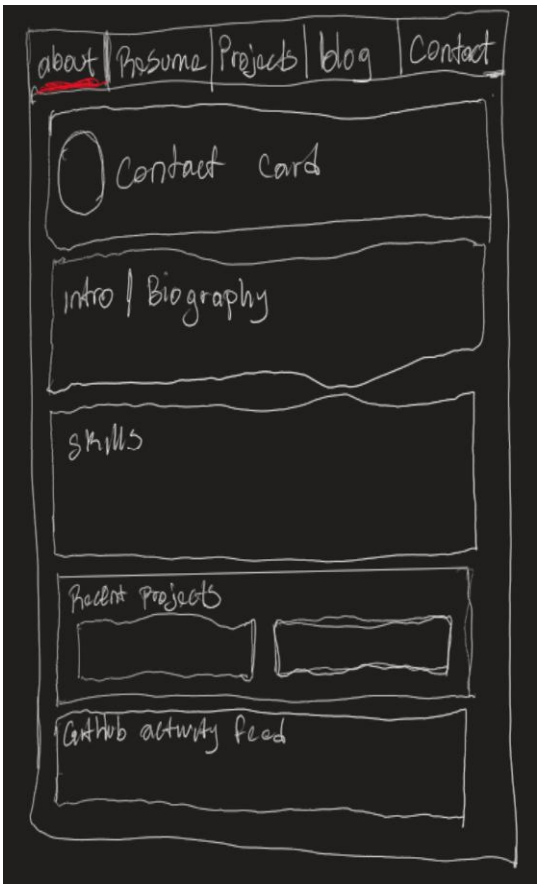
- Alternative contact methods (email, LinkedIn, etc.)

Blog

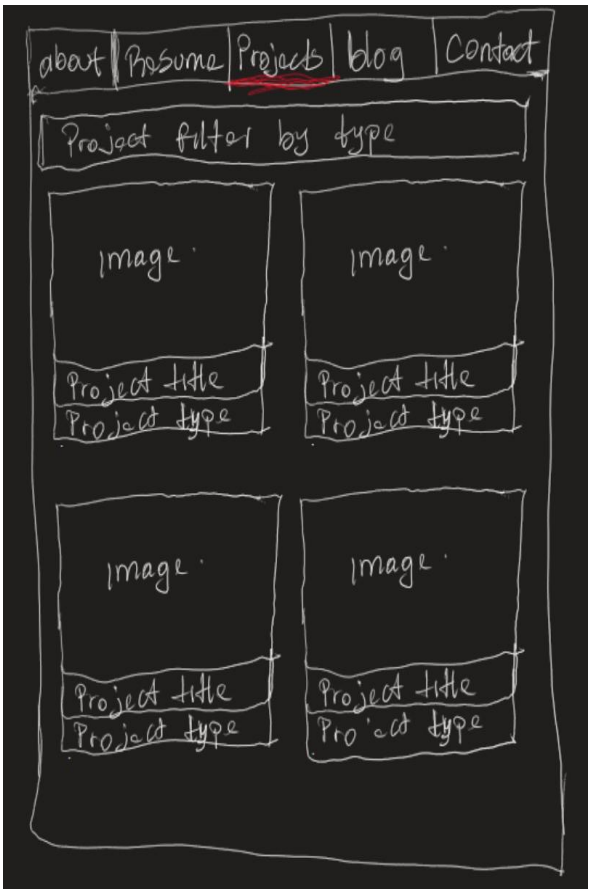
Also referred to as my **Reflection Dairy**, this section contains:

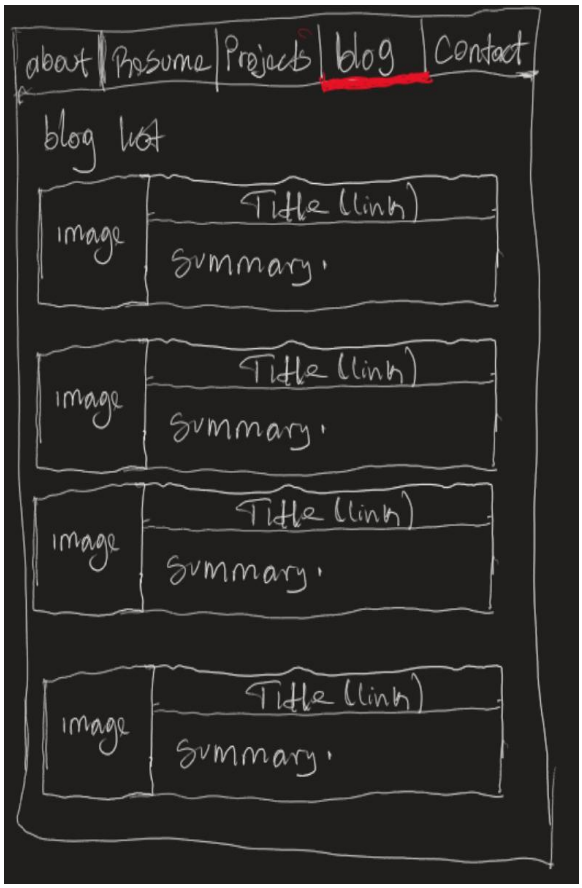
- A list of blog entries reflecting my learning progress through each unit of the COMP266 course
- Additional thoughts, technical breakdowns, or helpful resources.

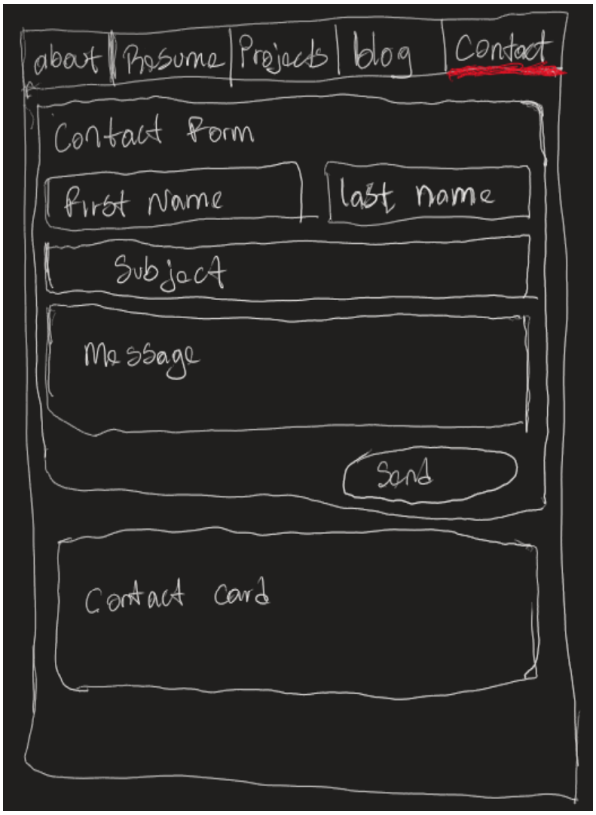



about
Resume
Projects
blog
Contact
Contact Card
intro / Biography
skills
Recent projects
Github activity feed


about
Resume
Projects
blog
Contact
Education history
certifications / achievements
work experience
Download PDF


about
Resume
Projects
blog
Contact
Project filter by type
Image
Project title
Project type
Image
Project title
Project type
Image
Project title
Project type
Image
Project title
Pro'ect type


about
Resume
Projects
blog
Contact
blog list
image
Title (link)
Summary.
image
Title (link)
Summary.
image
Title (link)
Summary.
image
Title (link)
Summary.


about
Resume
Projects
blog
Contact
Contact form
First name
Last name
Subject
Message
Send
Contact card

Unit 1 Reflection

As part of the work for Unit 1, I focused on laying a strong foundation for my personal portfolio website. I began by defining the purpose of the site; to showcase my skills, experience, and personal growth as a software developer. I then created three distinct personas: a technical recruiter, a fellow developer looking for collaboration, and a peer student. These personas helped me think critically about the range of users who might visit my site. I also wrote detailed user scenarios to imagine how each of these personas would interact with the website in a real-world context. Using these insights, I developed a structured and user-focused site map, with clear sections such as "About Me," "Projects," "Resume," "Contact Me," and "Blog." This sitemap reflects the key user goals identified in the personas and scenarios and is designed to support smooth navigation and quick access to relevant information.

The rationale behind my design choices is deeply rooted in the personas and scenarios I developed. For David, the technical recruiter, the priority is efficiency. He needs quick access to my key skills, certifications, and a downloadable resume. This directly influenced my decision to consolidate relevant details on the About Me page and include a clear, printer-friendly Resume section. Angela, the software team lead, is more technically advanced and interested in the depth of my projects and adherence to modern development standards. To support her goals, I designed the Projects section to include code summaries, GitHub links, and documentation for each project. Carlos, the computer science student, is looking to learn and be inspired, so I planned a Blog section where I'll reflect on my development journey and explain decisions in simple, accessible terms. Lastly, Lori, the small business owner, has limited technical background but wants to feel confident in my reliability. For her, I kept the design clean, jargon-free and ensured the Contact Me page was simple to navigate. These user needs guided every major content and layout decision I made in this unit.

The research process for Unit 1 has significantly influenced how I view user-centered design. I learned from Interaction Design Foundation's blog on personas that personas are not just fictional placeholders, but research-based representations that help in designing more empathetic and effective user experiences. This shifted my thinking—from typically grouping users by roles to instead focusing on specific, individual behaviors and goals. Another valuable source was their user scenarios article, which clarified the difference between broad use cases and narrow, story-based scenarios. This helped me create realistic, goal-driven narratives for each persona.

To ensure these personas and scenarios reflected real-world contexts, I browsed various Quora threads and articles about personal portfolios, learning what users expect when they visit such sites. I also used resources like Justinmind's guide on scenarios to write well-structured, meaningful stories. Initially, I found it challenging to switch my thinking from jumping into site maps and wireframes to first investing time into personas and scenarios. However, once those foundational elements were in place, the sitemap and layout came together smoothly. It's a lesson I'll carry forward into future projects.

One major challenge was resisting the urge to start coding immediately. I had to constantly remind myself to stay focused on the design and planning process. But now that the design stage is complete, I feel well-prepared and excited to begin the actual development. This early work has not only clarified my project's direction but also improved my overall approach to web development.

Sample1.html Issues

- Missing <!DOCTYPE html> declaration at the top of the page.
- The html tag is missing the lang attribute for Accessibility.
- Empty title tag degrading the user experience.
- Multiple tags were left open.
- Broken list tags <ul is invalid.
- List tags are not properly nested.
- <td> tags were left open.
- Using the attribute border=1 is highly discouraged.
- Images lack alt text for accessibility.
- Despite HTML not being case sensitive, use of uppercase tags like <H1> is discouraged in HTML5.
- No <meta *charset*="UTF-8"> in the <head> for proper character encoding. Adding this will prevent the browser from misinterpreting non-English characters.
- No <meta viewport> in the <head> for responsiveness on other devices.
- Paragraphs are too long. Limit the length to about 80 characters a line.

Unit2 Reflection

For Unit 2, I created the HTML foundation of my personal portfolio website using XHTML 1.0 Strict. The site currently includes:

- index.html (About Me)
- resume.html
- projects.html
- contact.html
- blog.html

Each page features consistent navigation, meaningful heading levels, hyperlinks (both internal and external), an image with alt text, properly marked-up lists, a semantic data table, a contact form, and the use of and tags. Each page is designed to be easily extended or styled later using CSS. I used relative URLs for internal linking and an absolute URL for external navigation (LinkedIn).
The full site is deployed at: https://student.athabascau.ca/~macho1/comp266/index.html

The design and structure of the site were guided by the personas and scenarios developed in Unit 1:

- David the recruiter can quickly access my technical skills on the About Me page and download a printable resume.
- Angela the dev team lead can view my GitHub projects and assess both code and team compatibility by reviewing my blog entries.

- Lori the small business owner can see examples of my client-oriented work and easily request a quote via the Contact page.
- Carlos the CS student gains inspiration from the structure and formatting of my portfolio.

Challenges:

- I spent over a month resolving access to the school's student web server. Even after gaining access via WinSCP, I encountered frustrating permission issues, which significantly slowed progress and as of now has still not been resolved.
- Another area I spent time researching was the difference between HTML5 and XHTML. I ultimately chose XHTML because of its stricter rules, which I believe will help instill better habits as I continue learning. Some helpful resources included:
    - https://www.w3schools.com/html/html_xhtml.asp
    - https://www.geeksforgeeks.org/html/difference-between-xhtml-and-html5/
    - https://landing.athabascau.ca/pages/view/110055/html

What I Would Do Differently:

- I would begin server testing earlier in the unit to avoid being blocked late in the process.

While I will be adding additional HTML where need be as the course continues, I believe I've built a solid foundation. The code is structured, accessible, and ready to support styling and scripting in upcoming units. I am really excited to proceed to the next unit and start styling my pages.

Learning Dairy

I began by refreshing my knowledge of CSS through YouTube tutorials (http://youtube.com/watch?v=wRNinF7YQqQ and https://www.youtube.com/watch?v=wsTv9y931o8). These videos helped me quickly recall the basics of styling, layout, and positioning. I also used W3Schools, which provided structured examples I could directly test and understand better.

I created a style.css file for global styles (fonts, background, navigation bar, etc.) and separate CSS files for individual pages (style-resume.css, style-about.css, and style-contact.css, style-project.css). This separation made my code easier to maintain and let me fine-tune each page's unique layout without affecting the others. I relied heavily on flexbox for arranging content, since it easily adapted to varying screen sizes. However, I found css grid useful and easy to setup on the programs section where a structured, two-dimensional alignment was needed. I made the navigation bar sticky, so it remains accessible at the top of the page. This decision was based on usability: it allows users to quickly move between sections without scrolling back up, which is especially helpful on longer pages or smaller screens.

To ensure the site met accessibility standards, I used the WebAIM Contrast Checker (https://webaim.org/resources/contrastchecker/). This guided my color choices and helped me achieve strong contrast between text, backgrounds, and buttons, making the site more usable for visitors with visual impairments.

Using Chrome's built-in device emulator, I tested how the site appeared across different screen sizes. I was pleased to see that only small tweaks were needed for mobile devices. This confirmed that my reliance on flexbox and grid naturally supported responsive design.

While my primary task was to use CSS for styling, I made a few structural adjustments to elements to better group content and apply consistent styling

I still have not been able to gain permission to my site on the school server at this time.

Reflection

Through this unit, I gained confidence in separating content (HTML) from presentation (CSS), which is an essential principle of modern web design. I also learned how to balance global and page-specific styles, improving maintainability while allowing customization. The use of flexbox and grid layouts gave me practical experience with responsive design patterns, while accessibility tools like the contrast checker reminded me of the importance of inclusive design. Overall, the process reinforced the importance of testing across devices and designing with the end-user in mind. The sticky navigation, clear visual hierarchy, and responsive adjustments all came directly from thinking about the scenarios and personas defined earlier.

Unit 4 Learning Dairy

I integrated a small client-side validation script into my contact form to support my Unit 1 personas (recruiter, collaborator contacting me). The script from the geeksforgeeks tutorial (https://www.geeksforgeeks.org/javascript/form-validation-using-javascript/) validates several inputs with regular expressions and show inline error messages.

What the code does:

- It grabs the form by ID (myForm) and attaches a submit listener. On submit it calls e.preventDefault() so the browser doesn't post the form until validation succeeds. I wrapped the code around DOMContentLoaded listener to make sure it runs only after the full HTML document has loaded.
- It then reads seven input values from the DOM (first and last name, username, email, password, phone, and date of birth) and stores them in local constants so they can be validated. Before validating, it resets the text content of all error placeholders so old error messages are removed.
- For each field, it defines a regex pattern or rule and tests the user's value against the pattern.
- If any test fails, it writes a friendly message into the corresponding -error span and flips isValid to false.
- If no rules fail (isValid remains true), it shows a success alert() (but still prevents submission in the original snippet—there's no form.submit() call).

What's good:

- The code uses dedicated elements to display error messages instead of intrusive alerts().
- Each field report its own reason for error making it easier for the user to figure out what caused the error.
- The code is clearly written, properly indented and follows a single naming convention throughout.

What can be Improved:

- Adding a DOMContentLoaded listener to make sure the code only runs after the full html document is loaded so form is not null.
- The code could benefit from some inline comments in addition to the tutorial notes.
- Ensuring the form is submitted once all validations are completed.

My Changes:

- Wrap the code in DOMContentLoaded listener so form is never null.
- I removed all the fields that were not related to my page and repurpose the first name validation to the name validation. Updated the regex to also allow special characters.
- Added the message validation regex to ensure the message is never null.

- Submitted the form once all validations where completed.
- Styled the error message in my style.css file.

Unit 4 Reflection

In Unit 4, I focused on enhancing the functionality and usability of my portfolio website by incorporating JavaScript form validation. I began by searching the internet for ideas and decided to implement form validation because it ensures that recruiters or potential clients submit complete and accurate information when contacting me.

To implement this feature, I adapted code from GeeksforGeeks(https://www.geeksforgeeks.org/javascript/form-validation-using-javascript/) and customized it to fit my project.

While working on this I used w3schools to refresh my knowledge on JS.

Unit 5 Reflective Learning Dairy

Based on the work already performed in unit 1, I came up with the following three ideas on how to improve my site using javascript:

1) Dynamic Project Gallery with Filters and Sorting

   As described in the project section of my sitemap in unit 1 I would like to create a dynamic project gallery that allows users to filter projects by category (e.g., "Web Development," "Mobile Apps," "integration Projects") and sort them by date or name. Using JavaScript, I can implement a switch for each filter type and a drop-down menu to select the sorting preference. This would make navigating my portfolio much smoother and more interactive compared to static HTML lists. I may also add small animations (such as fade-in/fade-out effects) to make the transitions visually engaging.

   This feature relates strongly to my personas and scenarios. For the recruiter and team lead persona, it allows them to quickly find the type of project they care about (e.g., "Mobile Apps" if they're hiring for app developers). For the student persona, it allows them to learn from or compare relevant projects without going through unrelated work.

2) Personalized Greeting and Theme Customization

   Another idea is to use JavaScript to detect the time of day and display a personalized greeting such as "Good morning, welcome to my portfolio!" or "Good evening, thanks for visiting." Additionally, I could allow visitors to toggle between light and dark mode or choose a preferred accent color, with their choice saved in local storage so that it persists across sessions.

   This connects directly to usability and accessibility for my personas. All my persona's benefits from a professional but welcoming site that feels tailored to them, which may leave a stronger impression. They also benefit from theme customization since they may prefer a dark mode while browsing at night. In the scenario where a visitor returns to my site multiple times, the fact that their theme preferences are remembered shows attention to detail and improves their overall user experience.

3) Back to Top Button & Active Navigation Highlight

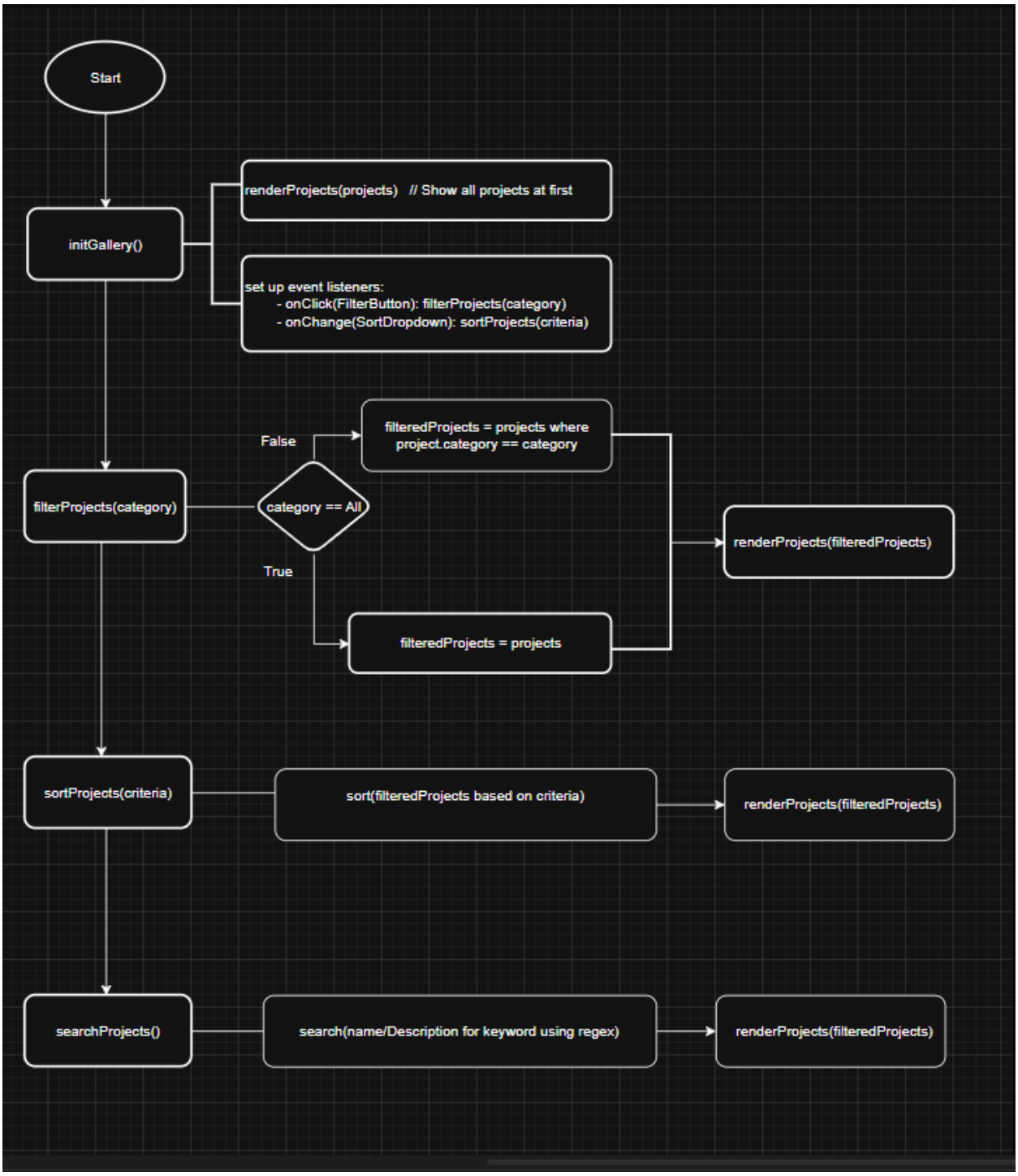
   I can also add a floating "Back to Top" button that appears at the bottom corner of the page when a user scrolls down. Clicking this button would smoothly scroll the user back to the top of the page. Along with this, I plan to dynamically highlight or underline the navigation link for the page the user is currently on. For example, if the visitor is on the "Projects" page, the "Projects" link in the navigation bar will be visually marked. This can be achieved with JavaScript by detecting the current URL or scroll position and adding/removing CSS classes to the appropriate navigation items.

   This improvement makes navigation faster and more user-friendly, especially for recruiters who may be scrolling through long pages of my portfolio. Instead of manually scrolling back, the floating button provides a quick and accessible way to return to the top. The active

navigation highlighting also helps visitors know exactly where they are on the site, improving clarity and orientation.

Dynamic Project Gallery with Filters and Sorting Design

1. **Data**
    - projects (global): An array of project objects that hold all project data (name, category, date, description, imageUrl)
    - filteredProjects (global) : an array to store currently displayed projects
    - selectedCategory (global) : a string that holds the current filter (e.g . “Mobile Apps”)
    - sortOrder (global): a string that holds the current sorting selection (“dateAsc”, “dateDesc” etc.)
2. **Functons and Classes**
    - Project(name, category, date, description, imageUrl)
        - Class constructor used to create a project object.
        - Scope: global
    - renderProjects(projectList)
        - Input: array of projects
        - Output: updates the DOM with project cards.
        - Scope: global
    - filterProjects(category)
        - Input: category string
        - Output: filters the project by the specified category, stores it in filteredProjects and updates the UI via renderProjects(filteredProjects)
    - sortProjects(criteria)
        - Input: sorting criteria ("dateAsc", "dateDesc", "nameAsc", "nameDesc")
        - Output: sorts the filteredProjects and updates the UI via renderProjects()
    - searchProjects()
        - Input: keyword typed in the search bar
        - Output: filters the project by the keyword using regex and updates the UI
    - initGallery()
        - called on page load to load all projects when the user first visits the page and attaches event listeners to filter buttons and sort dropdown.
3. **Program Flow**



Reflection

When I began working with JavaScript for my portfolio site, my main goal was to make the pages more interactive and useful for the personas and scenarios I designed in Unit 1. For example, the recruiter persona wanted to quickly browse and sort through my projects, while the fellow developer persona wanted an easy way to filter or search projects based on their interests. This

made it clear to me that I needed features such as filtering by category, sorting by name or date, and a search function.

My Learning Process

I found the design stage of this project much easier than I expected because I could map the features directly to the personas and scenarios, I created in Unit 1. Once I finalized the design, I revisited JavaScript fundamentals to adapt my knowledge to the front-end. I refreshed myself using W3Schools, focusing on DOM manipulation, object handling, event listeners, array methods (filter, sort), and regular expressions.

After that refresher, I began developing the features incrementally. I first implemented project rendering, then sorting and filtering, and finally a search bar that uses regex to find matches in project titles and descriptions. Breaking the tasks into smaller steps made the process more manageable and allowed me to build on successes.

Obstacles I Faced

Even with my background, I ran into obstacles. For example, I initially used inline event handlers in the HTML but realized this wasn't the best practice for clean, maintainable code. I switched to adding all event listeners in JavaScript once the DOM was loaded, which improved both the structure and separation of concerns. Another challenge was handling regex search properly—my first implementation was case-sensitive, which excluded relevant results. Adding the /i flag fixed this.

A key challenge was ensuring that renderProjects() only ran after the DOM had fully loaded. Running it too early meant the script tried to manipulate elements that didn't yet exist.

Another challenge was adapting my code to **XHTML 1.0 strict compliance**. Initially, I used custom data-category attributes to store project categories, but I discovered that these attributes are not valid in XHTML 1.0 strict. To meet the course requirements, I learned about the rel attribute and used it as a valid alternative for storing category values.

Debugging Process

For debugging, I used Visual Studio Code, which made it much easier to identify and fix both compilation and runtime errors. I relied on breakpoints and ran my code in debug mode to step through execution and inspect variable values at key points. This gave me deeper insight into the control flow and helped me catch logical mistakes early.

Overall, this part of the course was an opportunity not just to practice JavaScript syntax, but also to expand the way I apply it—moving from backend integrations into user-facing web interactivity. That shift has strengthened both my technical skills and my appreciation of how code connects directly to user experience.

Unit 6 Reflective Learning Dairy

I chose to download a local copy of jQuery into my scripts folder, following the process guide, so that my site does not depend on external availability and so I retain control over updates.

Proposal for jQuery Enhancements

For this unit, I plan to extend the functionality of my portfolio site by incorporating the jQuery library to improve usability and navigation. Based on the requirements I identified in Unit 1 (ease of navigation, accessibility for recruiters/employers, and smooth user experience), I will implement the following enhancements:

Tabbed Navigation with AJAX Loading:

Instead of reloading the entire page when switching between sections (About, Resume, Projects, Blog, Contact), I will use jQuery to convert my navigation into tabs. Each tab will dynamically load its corresponding content into the page using jQuery's .load() function, providing a faster and more seamless browsing experience.

Floating "Back to Top" Button:

I will add a floating button that appears in the bottom-right corner of the screen when the user scrolls down. Using jQuery, the button will fade in and out depending on scroll position, and when clicked, it will smoothly scroll the user back to the top of the page. This feature will improve accessibility and ease of navigation on longer pages.

Dynamic Navigation Highlighting

I will use jQuery to automatically highlight the navigation link for the section currently being viewed. This will provide users with a clear context of their location within the site, making navigation more intuitive.

Unit 6 Reflection

This unit really helped me see how jQuery can make a website feel much more interactive. One of the first big lessons came when I ran into a CORS policy issue while trying to dynamically load content. At first, I thought I had completely broken my site, but after reading about it, I learned to launch Chrome against Live Server. That small discovery not only solved the problem but also gave me a better understanding of browser security restrictions and how to troubleshoot my setup in Visual Studio.

Adding a "Back to Top" button was one of the features I really enjoyed working on. I found a working example on JSFiddle and studied how the code handled scrolling and smooth animation. From there, I customized it for my site so it fit my design. It was satisfying to see it working smoothly, and it also reminded me how useful it is to learn from existing code instead of always starting from scratch.

I also spent a lot of time reading the official jQuery documentation. It was interesting to see how much could be done with core methods and event handling, and it gave me confidence to restructure my navigation and page-loading features. The W3Schools .load() guide was another really helpful resource, because it gave me a straightforward example, I could adapt to dynamically load page content into my site.

Another thing I focused on was making sure the site would still work if jQuery or my main.js file failed to load. At first this felt like extra work, but I realized it made my site more reliable. That experience showed me how important it is to design for failure and think about progressive enhancement, rather than assuming everything will always load perfectly.

What stood out the most to me was how concise jQuery is compared to plain JavaScript. I liked how a few lines of jQuery could do the same job as many lines of vanilla JavaScript. It made my code much cleaner and easier to follow. At the same time, I can see that relying on an external library has trade-offs, so in future projects I may consider whether using vanilla JavaScript would be enough.

References

- mnQb6. (n.d.). *Back to Top example*. JSFiddle. Retrieved from https://jsfiddle.net/mnQb6/
- jQuery Foundation. (n.d.). *Learning jQuery*. Retrieved from https://learn.jquery.com/
- W3Schools. (n.d.). *jQuery .load() Method*. Retrieved from https://www.w3schools.com/jquery/ajax_load.asp

Unit 7 Reflective Learning Dairy

Unit 7 Proposal

For Unit 7, I plan to integrate two external web services into my personal portfolio website in order to enhance functionality and better serve the personas and scenarios identified in Unit 1.

1. GitHub API Integration

I will use the GitHub REST API to display my latest repositories or commit activity directly on my site. This feature supports the needs of my recruiter persona, who wants to quickly assess my ongoing engagement in coding projects without leaving the site. By dynamically fetching and presenting this information, my portfolio will appear more current and interactive, reinforcing my credibility as an active developer while providing immediate value to visitors.

2. EmailJS Service Integration

I will enhance my existing contact form by integrating the EmailJS service, which allows form submissions to be sent directly to my email without requiring a backend server. This aligns with the needs of both recruiters and potential collaborators, who value a reliable and straightforward way to reach me. The integration not only ensures that messages are delivered but also provides users with confirmation feedback, improving trust and usability while keeping the experience consistent with the overall look and feel of the site.

By adding these two integrations, my portfolio will both consume external data (GitHub) and send data (EmailJS), demonstrating practical API use while directly improving the experience for my target personas.

Reflection

For Unit 7, I enhanced my portfolio site by integrating two external services: EmailJS for handling contact form submissions and the GitHub API for displaying my public activity feed. Both services required working with asynchronous requests, which gave me more hands-on experience with AJAX and JavaScript.

The first integration was EmailJS, which allowed me to connect my contact form to an email service without needing a custom backend. I learned how to properly initialize the library, secure my public key, and handle form submissions with meaningful feedback for the user. This directly tied into my original persona goals, where recruiters and collaborators needed an easy way to contact me.

The second integration involved the GitHub API. At first, I used the modern fetch() method to retrieve my public events, but then I switched to jQuery's $.ajax() so that my implementation explicitly demonstrated AJAX usage, as outlined in the course unit. I learned how to handle success and error callbacks in AJAX, parse the returned JSON, and dynamically update my site with the most recent events. This added a live, professional element to my portfolio and highlighted my ongoing coding activity.

A key realization during this process was how valuable my Unit 1 planning turned out to be. Back then, I had already identified a GitHub feed and a contact form as potential features. Because of

that foresight, when Unit 7 required external services, I was able to select and implement these two features without hesitation. The early planning gave me a clear roadmap and prevented me from scrambling to come up with ideas at the last minute.