Project 1
Simple Rails website

Due: 1 March 2024

For the second project you'll build a small website in Ruby on Rails, making use of several of the major techniques we've seen (and building on our earlier work with HTML, CSS, and Javascript).

You will design and build a website for a small business, non-profit, or club. The group need not actually exist, but it should be believable (and if you pick a group that *does* actually exist you can use their actual information so that you don't have to invent any content).[1] The look and feel of the website should be functional and clean, and it should in general follow current web standards, although it will be on the simpler end of such websites.

The website you build should have the following properties:

- At least three pages, including a homepage or landing page (associated with the root route `'/'`).
- Each page should have a logo[2] that links back to the homepage.
- Each page should have a navigation bar with links to each of the pages on the site.
- Links to other pages on the site, whether in the navbar or elsewhere, should use `link_to` rather than hard-coding the links.
- All pages should have meaningful content unique to the page, appropriately structured and styled.
- All shared content (including header, navigation, footer, etc) should be shared using a Rails layout (in ERb).
- Any parameterised content (unique items embedded in what is otherwise a 'shared' area of the page) should be handled using `provide` and `yield` as appropriate. This includes at least the page title but possibly also some piece of the header or elsewhere.

[1] Note that if you use a real company's content and ever plan to deploy this work or make it public outside this course assignment, you need to get their permission to do so.

[2] The 'logo' can be styled text with the name of the group.

- The metadata in the `<head>` section of each page should be valid and appropriate.
- The site should make effective use of a Rails partial at least once to cleanly organise the design. Headers and footers are likely candidates for this (but any appropriate use is fine).
- All HTML should be as purely structural as possible, with the visual styling reserved for CSS/SASS.
- There should be some visual styling, though, with some visual design choices for how to arrange, highlight, and contrast various elements. [3]
- There should be some element (either on a single page or in one of the shared parts) whose display (whether it's visible on the page or not) is controlled via Javascript by clicking some other element of the page. (If the clickable element is also updated e.g. by swapping the words "show" and "hide" that's cool but not required)
- The project should include substantial use of the automatic testing facilities in Rails.
- The automatic tests should verify, among other things, that:
  - Pages exist and are routable
  - Pages and sections of pages contain certain content
  - Pages include certain links to other pages
- Broadly speaking, the more I can look at the test files and think, "If `rails test` generates zero failures, this project is *done*," the better. This is true even if the project is incomplete (and there are some test failures)—a good test case that is currently failing indicates that you understand what the requirements were and that you have a to-do list of what's left.

There are other things you've learned how to do in HTML/CSS/JS, and you're encouraged to use them too!

To the extent that you base your design ideas on other sites you've seen, *cite your sources* by indicating your inspirations in comments in the HTML. Similarly, any content lifted for this project should be cited either in comments or actually visibly in the page footer.

[3]You're not being judged on the aesthetics of this, though. I'm not a GAND professor. Just try not to make it actually visually clash.

Handing in

Hand this project in using the `handin` script for course `cmsc210` and assignment `proj1`. If you are doing your development on some machine other than the lab machines (totally ok, and not difficult—just install ruby, and then run the various `gem install` commands and so forth), upload your stuff to the server *and verify that it works in the lab* before handing in. I'll test your code by running `bundle install` if necessary and then running `rails test` and `rails server` (and also reading parts of your code to check things that wouldn't be visible from those two commands). So you should make sure it works there.

It's due at 8pm on 1 March.

Rubric

RUBRIC

Basics (35)	
5	Properly init'ed rails project, `rails s` works
5	At least one routable page
10	Homepage (`'/'`) + 2 other pages
5	Unique page content
5	Logo/wordmark with homepage link
5	Nav bar
HTML/CSS/JS (20)	
5	Valid metadata in single `<head>` section
5	Some visual styling, using CSS/SASS
5	HTML is structural, CSS is separate
5	JS to unhide/hide an element
Rails features (30)	
5	CSS, JS, images are served through asset pipeline
5	Some links use `link_to` appropriately
5	Nav bar consistently uses `link_to`
5	Shared layout, `yield` to include page content
5	Uses partial and `render` at least once to organise
5	Parameterised content with `provide` and `yield`
Tests (15)	
5	Pages exist and are routable
5	Page contents (and section contents)
5	Page links