COMP 307
Course Project
Due: December 8, 2020 on myCourses at 23:59 EST (website)
Due: December 11, 2020 by email at 23:59 EST (presentation)

You are building a complete website with a team of 2 or 3 students. A group of 3 is preferable. Your website must have as a **minimum**: a front-end, a backend, and a database. You can create your website using any language you like, even ones we did not cover in class. **However**, your website must run on the SOCS web server, so the backend technology choices are limited to: Apache, Python, C (or any compiled language – ie. C++), PHP and SQL. The front-end technology is open. You can contact help@cs.mcgill.ca to find out what backend technologies the SOCS server supports (specifically ask for "Ron Simpson" to answer you back). You will need to get my okay for your technology stack before you start the project (more on that later). Your project is to build a new School of Computer Science website, see http://www.cs.mcgill.ca .

You must code over 70% of your website from scratch. This is important. We are computer science people and we need to know how to program.

The best way to organize the work for the website is to divide the project into distinct areas and assign an area to one of your teammates. Make sure that each team member has been assigned an equal amount of web development work (creating PowerPoint slides and writing documentation does not count). This is important so that the working relationships and grading can go smoothly. If one team member fails to do their work, then **it will only affect their grade** since the work was divided into distinct equal sized units.

Your goal is to create a new and better School of Computer Science website. Your new website must keep the major features already existing in the current website, but you are designing a new professional look. It must be easy to use for new incoming students, students currently enrolled in the program, students who want to transfer into the program, professors who want to quickly find information, other people (like parents and academics from other universities and institutions) who want to quickly and easily find information important to them. A backend is also important that supports easy updates to the web pages.

Just because the website looks the way it does now does not mean it is the best way to present the information. Be creative and design not only a new look but a simpler and more friendlier website.

A team of 2 students must do the following:

- Design a new better layout for the website
- Write static pages for all the content
- Provide a simple backend: user accounts, login, secure private pages and the ability to modify some of the static public web pages
- Bonus: add an extra feature not already present in the current website (optional).

A team of 3 students must do the following:

- Design a new better layout for the website
- Write static pages for all the content
- Provide a complex backend: user accounts, login, secure private pages and the ability to modify all the static public web pages

- Bonus: add an extra feature not already present in the current website (optional).

Below are five images that describe important features that need to be kept, but you do not need to keep them in the same way. They can be presented differently or using different technologies or they can be located in different places. You are free to make things better, different and friendlier. How would you have liked the website to look like and behave like, when you came to McGill?

Image 1 and Image 2 describe the home page. Notice the following important things: SOCS logo, search bar, top menu, then a grid of four informational areas (teaching@cs showing important immediate things, latest@cs also showing important things but not immediate, events@cs that show current events of interest, and posting@cs for things like jobs etc), and a simple footer.

Do all these things need to be on the home page?
Is there a better layout?
What sort of information do you think is important (that is not there)?
What information currently on the home page should not be on the home page? Where should it be?
Is there a better color scheme or theme?
What about **responsive** design and **interactive** web pages?
Is this home page just wrong… and you have a great new idea!

Take time, with your team, to brainstorm about the home page and the general color and theme of the website. List some of your ideas and discuss them. Make sure the changes you plan are within your abilities.

Image 1 – Top of current home page

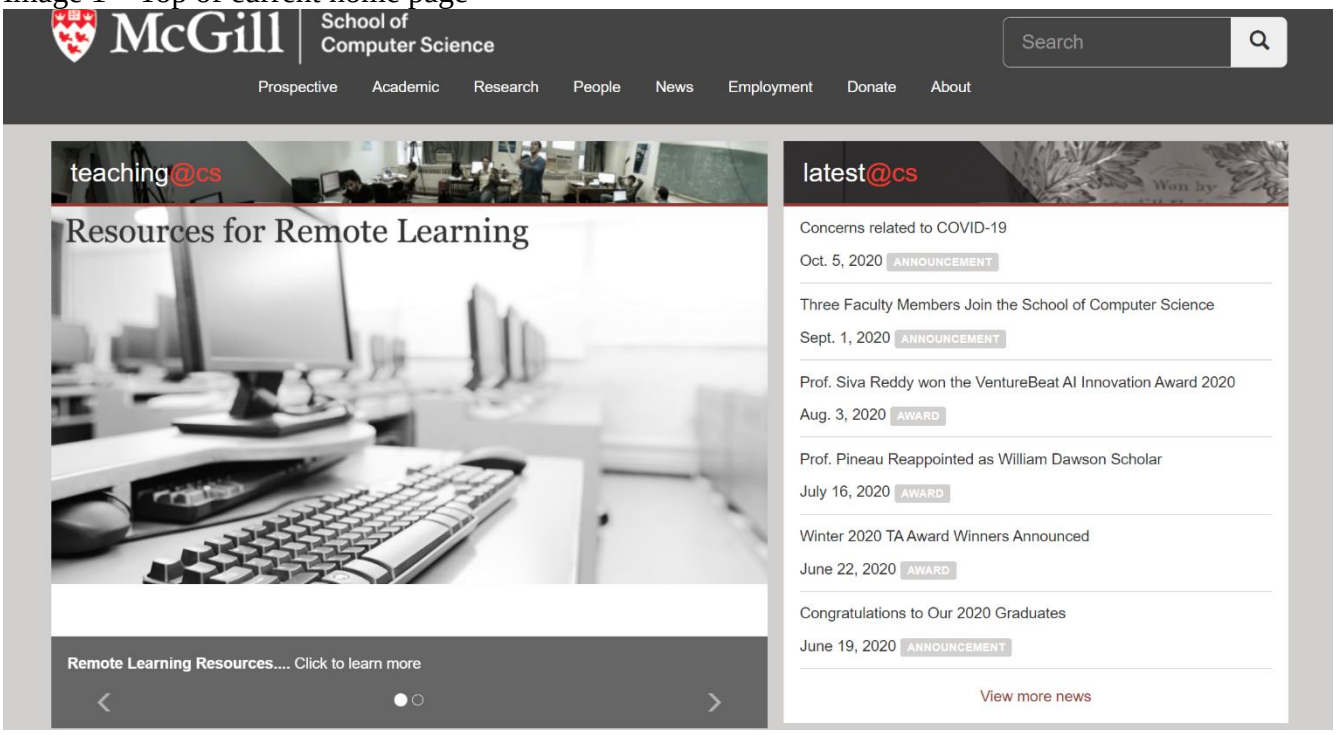


Image 2 – Bottom of current home page

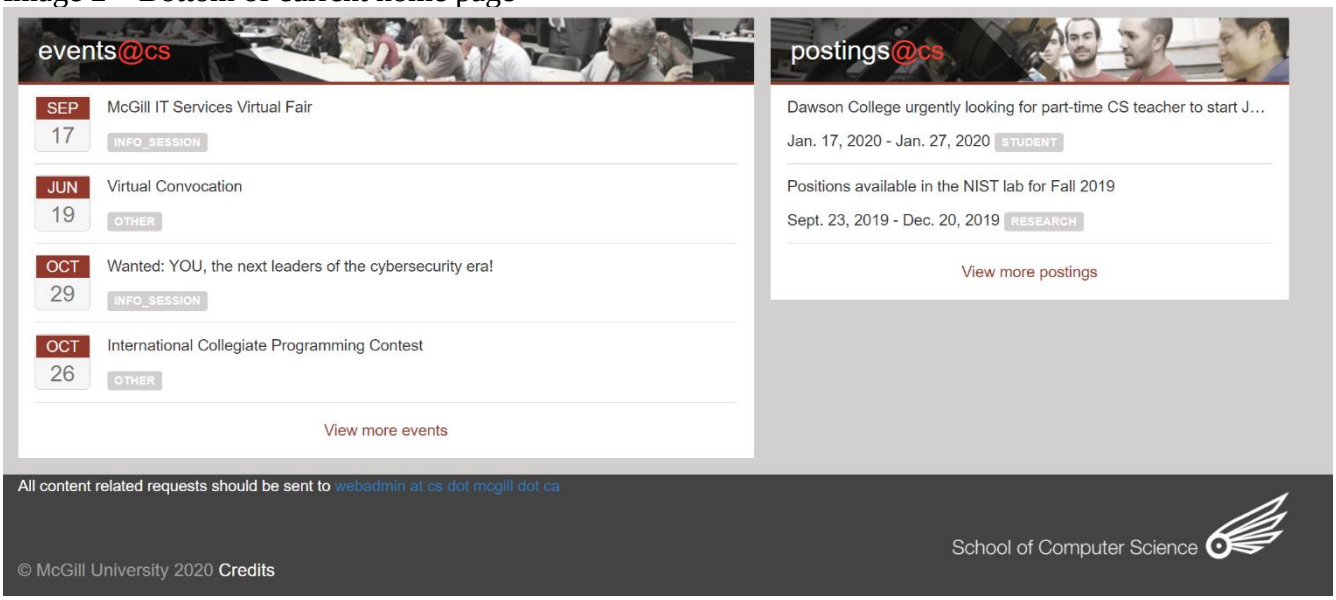


Image 3 demonstrates the main menu technology used by the current website. Specifically, it uses dropdowns to display the sub-menu choices. The user clicks on the top menu item to see the dropdown and then clicks on the sub-menu item to leave the home page. The selected web page is then displayed.

Should the menu be at the top of the screen?
Is it interactive enough? Maybe it should not be interactive at all?
When you read the menu items is it easy to understand the meaning of the selections?
Should each menu item have a popup explanation (appearing interactively without a click)?
Is there a better way to do this that does not need a menu?

Image 3 – Menu dropdown of current home page

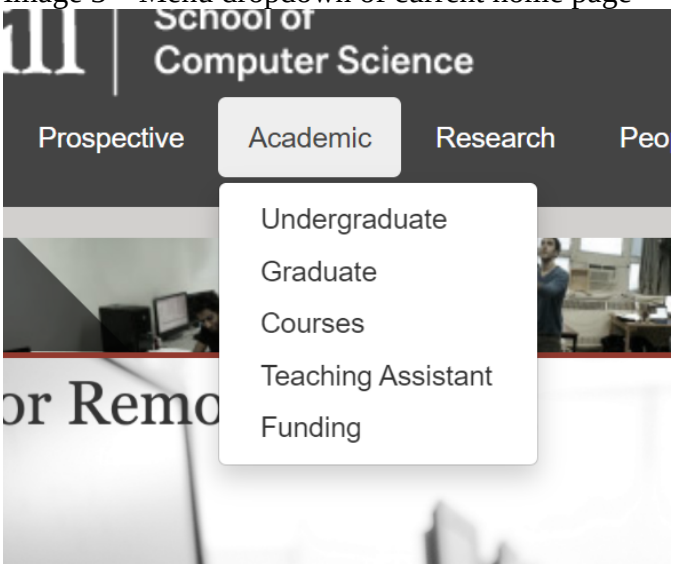
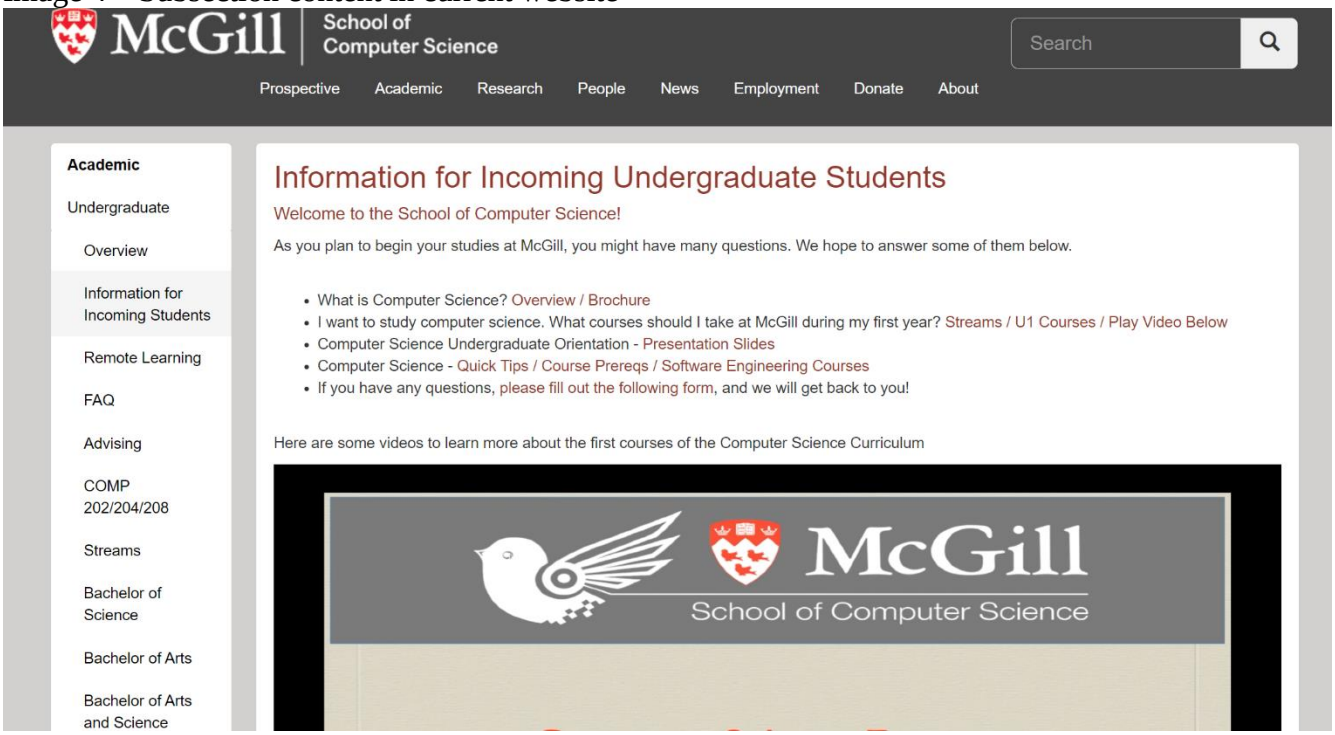


Image 4 presents a common sub-menu page. Currently, all the web pages of the SOCS website that is not the Home Page and not the Internal pages look identical to this example. A static vertical menu, on the left, displaying the top-menu selections and sub-menu selection and the title of the vertical menu. There are then a list of items that can be clicked. Each clicked item cases the menu item to be highlighted and the right display area presents the information. The information color and theme has been standardized as you see: primary heading, secondary heading, content (paragraphs, lists, anchors, images, imbedded videos). The homepage's page header and page footer is maintained.

Is there a better way to do this?
Is there a better color or theme?
From the backend point of view is there an easy way to maintain and update this information?
Should only experienced HTML developers modify and update these webpages?
How much can or should we automate?

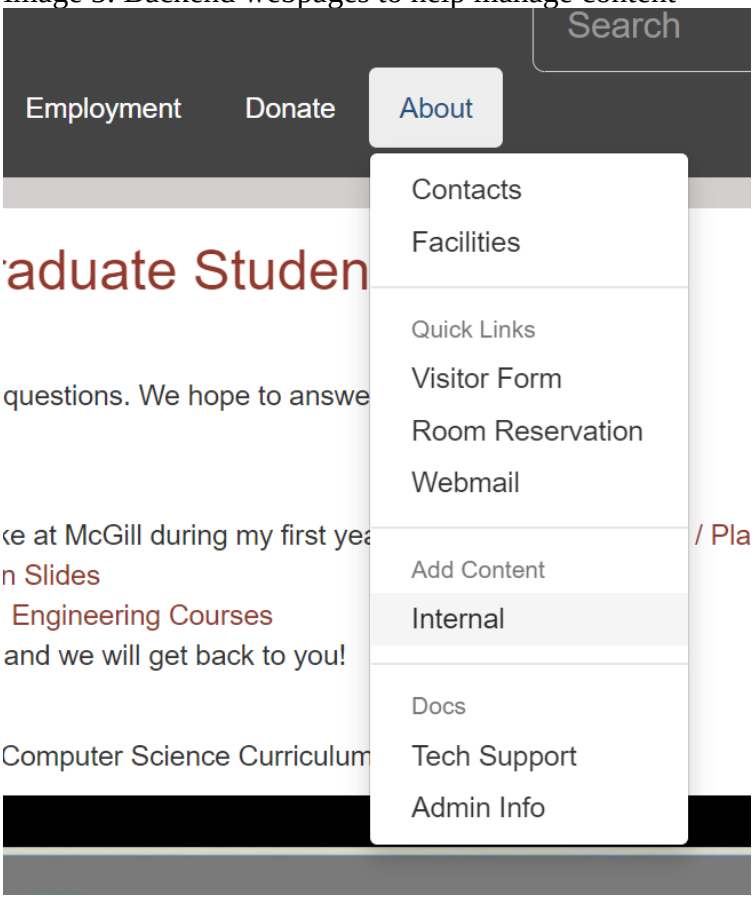
Image 4 – Subsection content in current website



Finally, image 5 shows you the link to the backend. I, unfortunately, cannot show you the backend. But, clicking on this links sends the user to a login screen. When they successfully login, they are presented with a totally new website that is secure. It has links to all the static web pages and articles. Users can click on those links to access a web text editor that allows them to write plain HTML. When pressing save the default color and theme is applied to the updated HTML. There are also create new web page buttons and delete old web page buttons. The current backend does not permit easy layout changes only content changes. Layout changes have been left to proper HTML developers.

Should elements of the web page design be left to proper HTML developers?
Should users be forced to write only simple HTML, or should they be permitted to change themes?
Should there be a markup language instead of a simple HTML editor?

Image 5: Backend webpages to help manage content



Does the SOCS website complete with the best! Or does it need help?

Project Instructions

Please follow these steps:

Step 1: Get your team proposal accepted by the professor (No later than October 31st)

- Team: select a team of 2 to 3 students.
- Do either the group of 2 project or the group of 3 project (see 1st page)
- Select a team leader to communicate with the professor
- Determine your technology stack (maybe contact help@cs.mcgill.ca)
- Determine a draft (tentative) layout for website
- Team leader meets with the professor to get the stack & layout approved
- Add your team to the Google doc (please populate all the **student** tabs)
  - https://docs.google.com/spreadsheets/d/1vm2JQUB820R6PtxK6_nF6qpzpvxe1qbxjRZUYDG72Bs/edit?usp=sharing
  - Write the first.last@mail.mcgill.ca instead of the student's name
  - Write down the tasks specifically assigned to that student (grading will follow this)
  - Columns exist for all the above, you can change the information later if needed
  - The team captain should only access this Google doc

Step 2: Your project

- Remember 70% of the project needs to be coded from scratch
- 30% of the code base can be from libraries, templates, and other sources
- Remember that the 2 person team requirements are not the same as the 3 person team requirements (see 1st page in this document)
  - However, everyone builds a front and backend that uses a database
  - The project must run on the SOCS server. This means you can test it from your own public_html directory.
  - Check with hellp@cs.mcgill.ca to verify that the technology you want to use is installed and if they want to install it
- Your website must use HTML5, CSS and JavaScript (or a framework) as a minimum
- You have all **November** to build your project
- Once **December** arrives, you should be finishing up the project and preparing the final presentation and report.

Step 3: Submission December 8

- Submit your project to myCourses, but also have a working website in the public_html directory of one of your team mates (maybe the captain)
- A readme.txt file with:
  - your team member names,
  - description of how to run your site
- An HTML file that links directly to your website home page
- A ZIP of the backend source and databases/files
- A ZIP of the front-end source and databases/files
- **All team members submit the entire project**. This is important because it is easier to enter a grade in myCourses when a student has submitted something than when they have not. The TA will use the Google doc to go to the captain's submission to grade from there.

Step 4: Video demo and written report December 11

- Purpose: to demo the running program and the code
- Your video can be recorded on your phone, using screen capture, Camtasia or YouTube (but keep it private since it is a McGill website)
- Each presentation must be 15 minutes long. All your team members must speak. Your presentation is 10 minutes long, then a short demo.
- Provide only the following slides for the video presentation:
  - (1) Have a title slide
    - Single page
    - Display project name and your team member's names,
  - (2) Division of work
    - Single page
    - Describe what each team member worked on
  - (3) Stack
    - Single page for the front-end
    - Single page for the backend including the database
  - (4) The experience

        - Single page for what worked well
        - Single page for what you will never do again
        - Single page to highlight a single technology that stood out for the team
    - (5) You must demo your application.
    - Diagrams are often better than text in this type of presentation.
- Report
    - Purpose: to help SOCS implement your website as their website if they select your project (also to help the TA if further analysis is needed)
    - Five-page report
    - Title page with project and team member names and ID numbers
    - A description of the technology stack
    - A description of how to install the website
        - Front-end, backend and database
    - A description of how to update the website
        - Using the backend interface
        - Coding by hand

HOW IT WILL BE GRADED

- TOTAL: 100 points

- POINTS:
    - (Pass/Fail) Professor's okay
    - (Pass/Fail) Team of 2 or 3 students
    - (Pass/Fail) 70% of project was coded from scratch
    - (Pass/Fail) Front-end (existence of)
    - (Pass/Fail) Back-end (existence of)
    - (Pass/Fail) Database or equivalent (existence of)
    - +10 points for a fully running website (partial running loses these 10 points)
    - +90 points for meeting the 2 or 3 person team requirements
        - Front-end code quality (HTML, CSS, JS and related langauges)
        - Front-end layout quality (responsive, interactive, page design, easy to use)
        - Front-end color and theme (pretty?)
        - Backend code quality (good COMP 303 techniques, good SE techniques)
        - Backend usefulness quality (does it work, is it working well, is it easy to use)
        - Good database usage (meet project requirements)
    - +5 points for bonus technology (does not qualify if it is simple)

- REDUCTIONS:
    - -10 points for not following instructions

- Points are awarded proportionally