COMP 307
Course Project
Multiple Due Dates at Middle of Document

This is your COMP 307 end of term team project.

This term's project has 3 development options:

- the SOCS Website Competition Project,
- the Free-Form Project, or
- the SOCS Website Project without the Competition.

You can optionally submit your team project for the **SOCS Website Competition**. The competition is described in detail at the end of this document.

Your team must consist of 3 to 5 students. A team of more than 5 will have extra requirements, please speak with the professor. The group cannot be smaller than 3.

Your website must have as a **minimum**: a front-end, a backend, and a database. Your website must use a web-stack. The stack can be one of the web stacks we covered in class (XAMPP or MERN), or you can use a different stack of your choice. You must get approval of your stack from the professor if different from what we saw in class. Students who want to run their website on mimi must speak to the professor to ensure the technology they are using for their stack is available. You are also permitted to host your website at some other location. Keep in mind, the competition projects must run on the SOCS machines. The ideal technologies supported by SOCS are: XAMPP, MERN, and Django.

Competition Project is to build a new School of Computer Science **Booking Tool**. We ask you to come up with a cool name for your booking website, since Calendly, Doodle, and MS Bookings are already taken. We would like to create a free booking tool for the professors and students in the school that provides special features that may be useful to professors, TAs, and students.

Free-Form Project allows teams to create a website of their choice within some basic requirements and theme. The basic requirements must be equivalent to the competition project description including the minimum mentioned above. The theme is "**useful tool for students or the planet**". This is a very open theme. You must get the approval from the professor. Very good projects in this category may also get used in some way. At the end of the semester, I might contact some of you. :)

You must code over 70% of your website by hand. This is important. We are computer science people; we need to know how to program. The remaining 30% can come from templates, frameworks, and libraries.

The best way to organize the work for the website is to divide the project into distinct areas and assign a single area to each 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 (more on this below). To help you divide the work, **remember the 3-tier website model**.

Your goal is to create a functional and useful Booking website or Free-Form website that adheres to the Theme and project requirements. You are designing a professional looking website: **responsive**, **dynamic**, **interactive**, **functional**, and **pretty**. It must be easy to use by students, professors, and TAs. Be creative and design not only a good look but a simple and easy-to-use website. Simplicity of design includes **simple code**, **fewer libraries**, and **optimized functions**.

This document will describe the basic layout and features of the website. Your job will be to **design the look**, select the **technology stack**, and **build** a functional application. You must provide the minimum required features, but you may include additional features. Make sure to get the professor's approval.

If you choose to submit your project to the competition, then your website needs to run well with very few bugs on SOCS servers.

You must submit the following:

- Submission 1 – Register your team
  - Due: November 15, 2024
  - Points: **5 points**
  - Registration form: https://forms.office.com/Pages/ResponsePage.aspx?id=cZYxzedSaEqvqfz4-J8J6ldtrJSsGchLsKruoBPFKv5UNjVXSTI3UzZIOE1FSkhRMTNBMFNaQkM4RS4u
  - Please register your team with the above link. You can also use the above link to make modifications to your team or leave a team.
  - For people who cannot find a team, please contact the professor. The due date can be extended in this case for Submission 1.
  - Please read the "Team Formation" section below.
  - Your grade for this section will be under "Project Registration" on myCourses.
- Submission 2 – Website Design Report
  - Due: November 25, 2024
  - Points: **15 points**
  - Submit to myCourses under "Project Design Report"
  - Your design report must have the following sections: (1) cover page with team name and team member names and ID numbers, with submission date. (2) Branding colors, font and logo. (3) Website storyboard with a short description of each storyboard box. (4) Screen mock-up of only important pages.
  - Please read the "Design Report" section below.
- Submission 3 – Demo & Running Program Tryout
  - Due: **Appointment** Between Dec 18 to Dec 20
  - Points: **50 points, bonus 3 points if earlier than Dec 18**.
  - Make an appointment with the TA. Each team will be assigned a TA.
  - At this appointment there will be two phases. Phase 1, the TA does not touch your program, and you will demo it to them. Phase 2, after the demonstration is completed, the TA will use the program without your help. In other words, you stay silent, and the TA will see if your program is intuitive to use and if bugs are encountered.
  - Grading: TA will enter your grade into myCourses under "Project Demo & Tryout"
- Submission 4 – Code Submission Verification
  - Due: Between Dec 18 to Dec 20
  - Points: **30 points, bonus 2 points if earlier than Dec 18**.

- Submit to myCourses under "Project Code Verification". Submit two things: (1) all source code, and (2) README.txt file containing the URL to the running website and a table. Each person in the team submits. In the table write the name of each team member and what they coded. Each team member must have completed at least two things on their own without help from other team members. Add you name to the source files.
- The TA will look to verify the 70% code requirement, the quality of the source code, and if the work each team member did on their own was good.

Team Formation

To have a great team experience it is important to do the following:

1. Select a group of people you can work well with, and always be polite.
2. Divide the work from the start into non-overlapping areas. That way if someone fails to do their job, the prof can deduct those grades from only the offending student. When dividing the work, also make sure that dependencies are handled well. In other words, if a team member does not finish something it does not impact the other team members too much. Be smart in this and you will be very happy.
3. When something goes wrong in the team, do not let it go for a long time. Come talk to me. It is better to have a meeting with me than to have an end of semester "help us" meeting. There are many options when you come to me early (I am not scary).
4. Write a rough plan on paper. Doing this in writing is very important to make clear each other's jobs, **specify due dates**. Super, super important for a happy experience. Treat your deadlines like assignments. Don't be late in completing each deadline. If you need to be late, email your teammates to discuss (like you do with the prof when you are late with an assignment). Decide and redo the deadline schedule, tasks, workload, etc. One missed due date not handled well can cause many problems at the end of the course.

Design Report

An important step when developing a website it to form a plan. Before you begin programming the website we ask you to form a simple plan. Instead of asking you to write a formal report, we ask you to only include the following sections in your document:

(1) A cover page with team name, team member names with ID numbers, and submission date.
(2) Branding, which includes colors that will be used for all the public and private facing webpages; The title and text fonts, plus the font series for both title and text (If you have more fonts than title and text, then you must provide those fonts as well together with their font series); Last you must provide the logo.
(3) Create a website storyboard with a short description of each storyboard box. This will give us a good understanding of the scope of your website. Follow correct syntax. It will also give us an opportunity to comment on missing features or possible design decision concerns.
(4) Provide a screen mock-up of only the most important pages. This will give us an opportunity to comment on missing features or possible design decision concerns. This would give you the opportunity to adjust your website before the final submission.

Please submit this report as a PDF.

Minimum Project Expectations

Design and build the website using a web stack of your choice. The Competition projects must use the SOCS server running either XAMPP, MERN, or Django. For non-competition or Free-Form you can use any Stack, with the professor's approval, and the website can be hosted anywhere (the TAs need to have access to the code and database at that remote site).

See the "Website Description" section for a full description of the project, however, I am not asking you to build the entire project as described (**you may if you wish**). I have provided a large project description from which you will decide on a functional subset that your team will build. The minimum requirements for your project are described here:

- Rule 1: All public facing web pages, as described below.
- Rule 2: Minimum 1 private facing page per team member. From a storyboard point-of-view this means at least one square (HTML page) and one circle symbol (SQL code).
- Rule 3: The website records information within a database (SQL or Mongo).
- Rule 4: Minimum 2 extra features (with HTML page and SQL/Mongo or text file data)

How do you calculate your workload?

- Let us assume you are in a team of 4 people.
- Rule 1 says you need to implement all public facing web pages. The website description section identifies the following pages: landing, login/registration, select appointment, see list of selected appointments. You can add additional pages or features if you want.
- Rule 2 says that a minimum of 4 private pages need to be created. Select these features from the website description section under "private facing web pages" and "additional optional features". You can add even more features if you want.
- Rule 3 says you must use a database in your web application.
- Rule 4 says you must select 2 additional features that either interact with the database or a text file in some way. Select these features from the website description section under "private facing web pages" and "additional optional features". You can propose your own features if you want!
- Conclusion, if you have a team of 4 people, you must implement 6 features (4 of which must be private facing pages, 2 are "additional" features) plus the front facing web pages. Two of the six features do not have to use the database.

Website Description

A Booking website allows people to easily make appointments and meetings with one another. There are a few important use-cases I would like to highlight using a professor/student example: (1) A professor has office hours and wants to make a booking system where students can arrange to meet the professor throughout the semester during these regular periods. These times are available on a regular schedule, like every week or every month, on a specific day and time between a starting and ending date. The professor would want to create this recurring meeting time, share a URL with the class, and then the students would want an easy way to see when the next times are available and reserve a time slot. Optionally, there should be a way to specify if the meeting accepts only a single person or multiple people. (2) Another use case is a student who wants to meet a professor but cannot meet during one of the teacher's office hours and instead sends a request to the professor with possible alternate times. In this case the professor then picks from this list or has the option to make their own suggestion. (3) A

third use-case would be when the professor wants to make a special appointment period that is not recurring (a single event) and have people register for it, otherwise like use-case 1. (4) The professor wants to make a meeting poll, selecting dates and times when a meeting can take place, then sends a URL to those who are invited, and the invited use the URL to vote. (5) The last use-case is when there is equipment to borrow. In this case this type of appointment results in someone going to the professor's office to get something at a particular date and time. The difference here is that there is a return date and time when the equipment should be returned by the student.

For our project, the use cases 1 to 3 are the most important, with use case 4 being something nice to have, and use case 5 is less important in Computer Science.

A more general description of the project is detailed below in the following bullet list (remember, you are only implementing a useable subset, not everything – you get to decide on that subset):

- I use the term "user" to mean a person who may or may not be a member of the website but does interact with the website. I use the term "member" to mean a person who has registered with a username and password. I use the term "public" to mean a user who is not a member.
- Registering users supply an email address and a password. Members of this website can only belong to McGill community; therefore, the only valid email addresses are those that end with @mcgill.ca and @mail.mcgill.ca. The user can provide any password.
- Members can only create bookings; however, the public can reserve a meeting using a provided booking URL.
- Members:
    - May create the following bookings: weekly recurring meeting periods, special one-time meeting dates, and a meeting poll.
    - Create a URL that can be shared with the public unique to the booking.
    - Optional attachments to a booking:
        - A view only checklist for the user to prepare for the meeting
        - An online fillable form for the user to fill out before coming to the meeting with a way to see the data.
- Users:
    - Use the supplied URL to pick when they would like to meet.
    - Send a request for an alternative period,
- The following should happen in general always (but are optional for our project):
    - Appointment notifications,
    - Appointment confirmations,
    - Add appointment to Google calendar,
    - A centralized location to see all the user's appointments by type (reservation, meeting, poll),
    - A centralized location to see the history of all the meetings the user has had in the past.
- Minimum web pages:
    - Public facing web pages:
        - Landing page
        - Login and register page with a ticket system for security
        - Book a meeting using URL page
    - Private facing web pages:
        - Create booking page: recurring weekly or special one-time dates and create the invite URL.
        - Send a request for a meeting with someone outside their availability page.

    - For all users
        - Central location page to see all active appointments and history of appointments.
- Additional optional features:
    - The creator of the regular booking may not be available for a particular date and would need a way to indicate that or remove those dates from the regular booking schedule.
    - You can propose your own additional features. Please confirm with the professor.

COMPETITION INSTRUCTIONS

If you registered for the competition and then decline at the end, make sure to tell the professor.

At the end of December, the professor will select 3 best projects. These 3 projects will be announced to everyone in class by email. The professor will then submit these 3 projects to the School of Computer Science. The school reserves the right to not select any of the 3 projects. If they do select one of the projects, then an email will be sent to the class announcing the winner. This will probably happen by the end of January. At the same time, the winning project will be placed on Professor Vybihal's Hall of Fame webpage. The winning project will either: (1) SOCS will adopt the students project as-is!!!, (2) go through additional development (through a COMP 400 project, a paid position, or a master's student). The winning team might be invited to help deploy the website.

HOW IT WILL BE GRADED

- For the project to be graded at all, the PASS / FAIL items must all be completed.

- PASS / FAIL
    - Filled out the Registration Form.
    - Team of 3 to 5 people.
    - 70% of project was coded from scratch.
    - Existence of frontend
    - Existence of backend
    - Existence of database
    - Each person spoke in presentation describing their section.
    - All competition projects must submit a 5-minute recording of someone using your website, with audio. This will help us when presenting the projects to the School of Computer Science.

- REDUCTIONS:
    - -20 points for not following instructions (proportional)
    - Standard late penalty (note there are multiple due dates)

- AWARDING GRADES: The TA will award points proportionally. This means, if your feature or style is only implemented partially, then you will be awarded partial points for that feature. For example, if you completed 25% of the feature, you would receive 25% of the points.

- WEBSITE DESIGN REPORT (POIINTS 15)
  - +2 = Cover page
  - +4 = Branding
  - +5 = Storyboard
  - +4 = Mock-ups

- DEMO & RUNNING PROGRAM TRYOUT (POINTS 50)
  - Phase 1 – Demo
    - PASS/FAIL = It runs
    - +5 = Appearance
    - +5 = Simplicity
    - +5 = Features work
  - Phase 2 – Tryout
    - +10 = Easy to use?
    - +25 = Features run well (Bugs?)

- CODE SUBMISSION VERIFICATION PER INDIVIDUAL (POINTS 30)
  - PASS/FAIL = Source code present
  - PASS/FAIL = README.txt present and filled out correctly
  - +10 = Individual code (70% code and works)
  - +10 = Quality of coding (style, optimization, bugs)
  - +10 = Uses the database

Rubric

POINTS	DESCRIPTION
Full	A Level - The student answers the point exactly as described.
Partial	B Level - Solution is very good but missing 1 element. (80% points) C Level - Solution is okay but missing 2 elements. (70% points) D Level - Solution is only partly completed. (50% points) F Level - Solution is not good. (10% to 25%)
0	Solution was not submitted, or it is completely wrong.