**SODV1201** INTRODUCTION TO WEB PROGRAMMING

Project: Develop A   
Coworking Registry

# DESCRIPTION

In this course you will build a complete and fully functional **coworking registry** following modern web design best practices. "Coworking" is a sharing economy concept where a group of people share an office space, usually equipped with desks, internet service, meeting rooms, and all the other trappings of a modern workspace. This registry allows people who have office space suitable for coworking to find people who are seeking this kind of space. The business model is simple: the owner of the space charges a fee for the use of the space, and the coworking people get a space to run their business, hold meetings, get work done, and meet other like-minded coworkers. The costs for renting a coworking space are usually much lower than office leases, and the terms are much more flexible. This concept is very popular in Europe (especially Sweden) and is becoming more common in North America.

Here are the objectives of this registry:

* allow people who own coworking spaces to create and maintain a listing that describes their space
* give people interested in coworking a way to find a suitable space
* provide a basic way for owners and tenants to connect with each other

# INDUSTRY RELEVANCE

We're going to run this project in such a way that it is as **industry-relevant** as possible. This means you will essentially be creating the registry as if you were **working for a client** instead of completing a college assignment. This also means the architecture of the system is not simplified for assignment purposes -- we will build the system to near-industry standards of quality, performance, security, and scalability.

We'll spend the entire course working on this project, in addition to doing some **checkpoint assignments** to show competence and progress in various related technical areas. Along the way we'll pick up the necessary architecture, HTML, CSS, JavaScript, API, Node, testing, security, and performance optimization skills we need to build this system out.

You'll work in teams selected by your instructor (we can discuss alternatives case-by-case). It's generally up to each team to **manage task assignments** and **balance workload**. However, we will ensure that everyone in the course has equal opportunity to explore and develop skills in all of the important technologies that make up a modern web application.

This is a fairly complex project, with lots of moving parts. You'll need to leverage the programming and development skills you've picked up so far in the program to be successful. This includes (but is not limited to) your teamwork skills, your database design and SQL scripting skills, your knowledge of algorithms, object oriented design, and data structures. The primary programming language in this project will be **JavaScript** on both the frontend website and the backend layer. You'll naturally use SQL to create and manage your persistence layer in MSSQL.

This project will require a **significant** investment and time commitment from all members of your team, over and above classroom hours. We will set up regular out-of-class lab time, and of course your instructor team is here to assist you. Most of our in-class time will be devoted to working on this project and its related assignments; we will introduce new material as needed. This means your success on the project depends **largely on your initiative**. Don't wait to be directed to proceed on something; start work as soon as you know what is required (and reach out for assistance regularly!).

When your team completes this project, you can be confident knowing you are one step closer to joining your software development dream team, and your skills will be ready to rock.

# REQUIREMENTS

**Build A Sitemap/User Flow Diagram.** Create a sitemap or user flow diagram showing the navigation structure of your website. A sitemap is simply a graphical representation of how the content on your site is organized into distinct web pages, and how a user travels from one page to another. Your sitemap should show enough detail that it is clear how you plan to implement all required functionality and **user journeys**. Your sitemap does not need to show details unrelated to navigation -- you **don't** need to include logos, themes, detailed content, images, and so on. Also, your sitemap only needs to address the **minimum viable product (MVP)** as defined in scope (below). We'll cover some example sitemaps in class to get you started.

*Deliverable: a professional and neat sitemap/user flow diagram, showing all website pages and how they are related for the MVP edition of your registry.*

**Build A RESTful API Specification.** Create a spec that defines how the frontend (user experience layer) of your application will communicate with the backend (application and persistence layers). The API spec is a **contract** that allows your application to run on the web (and on mobile, if that was ever required). All modern websites use RESTful APIs to separate concerns. The "RESTful" part is simply a design pattern, and we will cover it in class. For your project, your team should discuss, design, validate, and ultimately sign off on a RESTful API that fulfills all the needed MVP functionality and follows best practice.

*Deliverable: a professional and neat RESTful API specification for your registry site, with all services defined to fulfill the scope of the MVP edition.*

**Design And Implement The User Interface.** Using HTML, CSS, JavaScript, your sitemap, and your RESTful API specification, go ahead and build out the **user experience** part of your site. This is the frontend of your application. Users will interact directly with this site, which in turn is communicating with your backend through your API. You have complete control over how you style your frontend using CSS, so use your full creativity. **Important**: make sure your user interface renders appropriately on landscape desktop (i.e. laptop or desktop displays) and on portrait mobile (i.e. phones and tablets in portrait orientation). You should consider this requirement early on in your design, and we'll touch on this in class. Your final user interface should closely match your sitemap (minor deviations are expected) and it should use all the RESTful APIs you defined in your specification.

*Deliverable: a functional, usable, creative, and performant user interface that fulfills all scoped requirements for the MVP edition of the registry. Your user interface must be usable in landscape desktop (typically 1920x1200) and portrait mobile (typically 640x900).*

**[INSTRUCTOR DISCRETION] Design And Implement The Application Layer & Persistence Layer.** Using Node, JavaScript, SQL, and your RESTful API specification, go ahead and build out the **backend** part of your application. This is where you implement all the support that responds to requests from your frontend, validates them, enforces security, and manages the database. This *includes* designing and implementing a simple MSSQL database to store listings in your coworker registry, and a small set of stored procedures to manipulate it.

*Deliverable: a functional and performant backend that fulfills all RESTful APIs you defined in your specification. The application layer and database is to be published to a server (TBD) and the source code is to be managed in a BitBucket repo.*

# SCOPE

The word **scope** in this context means "what features does this system need to provide?". Scope is important, because we only want to bite off what is feasible in our course. When you take your Agile Project Management course (term 3) you'll learn more about managing scope in a software project. For now, we'll define scope with a set of **user stories**, which are short customer-focussed statements of what the system is supposed to do. Make sure your design considers all of these user stories! If anything is unclear **do not skip it**, but reach out for clarification!

We introduced the concept of **minimum viable product** earlier. What does that mean? MVP is a way of saying "a product that is finished only as far as it needs to be to **satisfy scope** and **minimize technical debt**". As your team builds out your registry, you are expected to apply best practice (clear code, good commenting, standard design patterns) and implement all required features. You are not expected to implement features that are not required. For instance, you will see below there is no login system for this registry. This is intentional -- we can demonstrate functionality to a client without requiring users to log in. In real life, a subsequent iteration of the product would add login and authentication, but it is **not required** in this project.

**User Stories.** In the user stories below, there are two roles: **owner** (a person who has an available coworking space) and **user** (a person seeking a coworking space). Your team needs to consider all of these user stories.

1/ As an owner I want to create, inspect, update, and delete a listing for my space.

2/ As an owner I want to have the following details attached to my listing:

* Name of building (e.g. Bowness GroupSpace)
* Name of neighborhood or subdivision (e.g. Bowness)
* Exact geographic location (not the postal address)
* Type of coworking space (desk in shared space, desk in private room)
* Amenities (wifi, printer, fax, showers, bike locker, meeting room, projector)
* Price (per month in CAD$)
* [BONUS MARKS] Ratings from past users (1 to 5)
* [BONUS MARKS] Photos

3/ As an owner I want to be able to have multiple listings since I'll likely have more than one space to rent.

4/ As an owner I want to be able to mark a listing "rented" or "not rented".

5/ As a user or an owner I want to filter and rank listings using these criteria:

* Proximity to my location (does **not** need to consider routing)
* Price
* Availability (e.g. rented or not rented)
* Matching amenities (e.g. has a meeting room)
* Type of coworking space
* [BONUS MARKS] Minimum rating (1 to 5)
* [BONUS MARKS] Has photo (yes or no)

6/ As a user or owner I want to use this service on my phone or on my desktop using any popular browser.

7/ As a user or owner I want to see the location of a listing on a map (use Google) directly in the site (i.e. I don't want to have to go someplace else to see the location on a map).

8/ As an owner I want to ensure I cannot enter missing or invalid information on my listing.

# TOOLS & TECH

You’re free to use almost any tools you want for this project. The only technology and tooling constraints are:

* BitBucket for repo management (shared with your team and instructor)
* HTML5/CSS3/JavaScript in frontend
* [Instructor discretion] Node/JavaScript in backend
* [Instructor discretion] MSSQL 2016/SQLite/MongoDB for the database

# PROJECT DETAILS

Assignment Type: Team

Format: Published online website connected to published backend

Original Work: Required from team

Submission: D2L dropbox as directed

D2L Dropbox: As directed in class

Due Date: Final deployment due **August 20, 2021**, intermediate checkpoints announced in class

Writing Standard: Business professional

English Level: Equivalent to TOEFL score of 83 or CLB level 8 or higher

Length: Does not apply

Cover Page: Not required

Author Name: All team members must be clearly identified on all submissions

Course Code: Required on all submissions

APA Style: Does not apply

# HOW TO GET TOP MARKS

**This is an important project for you. It will significantly boost your development skills.** Start work on this project immediately, and plan on devoting a significant amount of time to it over the term. If you procrastinate, the game is over. If your team doesn't mesh, you need to deal with that immediately. If you assume "someone else" will do the work, you will likely fail the project and the course.

Having said all that, here's how to score well: **take initiative** and move things forward every day; **invest the time** into creating a distinct, engaging, performant solution that showcases best design and implementation practice; **comment** your code completely and without spelling errors; make a **team charter** to ensure all members of your team are committed; reach out for assistance frequently; attend all classes; attend as many **labs** as you need to (send at least one team person each lab); make sure you satisfy all **user stories**; don't waste time on unnecessary features; test test test test test test; organize your codebase in BitBucket; cooperate and collaborate with your team; use tools like **Teams** to stay connected; plan on solving many problems you did not anticipate; plan on doing research online when you are stuck; plan on **a** **little** **frustration** from time to time but don't give up; **rise** to the challenge; try to nail the **bonus** **mark** features if you can; and keep in mind this is a **portfolio-grade** project.

# 

# ASSESSMENT CHECKLIST Coworking Registry Sitemap

|  |  |  |
| --- | --- | --- |
|  |  | Score |
| Submission method & format |  | /2 |
| Team identification, all team members' names on work product |  | /6 |
| Indicates how major user stories are satisfied |  | /16 |
| Indicates where listings are displayed |  | /4 |
| Indicates major navigation paths |  | /4 |
| Work product grade |  | /8 |
| Total |  | /40 |
| Plagiarized or unoriginal work (0% + academic sanction) |  |  |
| Total adjusted |  | /40 |

# ASSESSMENT CHECKLIST Coworking Registry Front End

|  |  |  |
| --- | --- | --- |
|  |  | Score |
| Submission format (zip file to D2L) |  | /4 |
| Required user stories are achieved (8@4ea) |  | /32 |
| Code hygiene evaluation (comments, indentation, clarity) |  | /32 |
| Bonus: users can rate listings and search by minimum rating |  | /8 |
| Bonus: professional layout that renders on desktop and mobile |  | /8 |
| Overall impression of design and implementation |  | /16 |
| Total |  | /100 |
| Plagiarized or unoriginal work (0% + academic sanction) |  |  |
| Total adjusted |  | /100 |

# ASSESSMENT CHECKLIST Coworking Registry Individual Contributions (Presentation)

|  |  |  |
| --- | --- | --- |
|  |  | Score |
| Can explain a block of random code (up to three attempts) |  | /8 |
| Can explain how a random feature is implemented |  | /8 |
| Total |  | /16 |
| Individual contribution grade factor |  |  |