

**Capstone Project**

**Deadline: Wednesday, October 21st 2020, 09:00 AM**

[**GitHub Classroom Link**](https://classroom.github.com/g/CvvmF1pk)

Your task is to solve a problem with a full-stack web application. The problem that you solve will be your choice, but you must define the problem and demonstrate how your application solves the problem in your documentation and your presentation. You can choose your own tech stack, but you must be able to complete the project so be sure to choose the stack you are familiar with. The reason we have chosen to give such a simple assignment definition is to allow you to be creative and also to give you enough time to focus on the details. This is your final capstone assignment and should represent a professional level of quality. It is your primary portfolio piece from this program.

* **Your group size and membership need to be proposed and approved by an instructor BEFORE problem definition and application ideas are generated. You may opt to work alone, in pairs, or in a group.**
  + **Choose a group leader who will serve as the primary point of contact with TECHCareers staff during the capstone (obviously concerns can be brought forward by anyone).**
  + **Have the group leader email your initial group name and your membership roster proposal to** [**techc@ualberta.ca**](mailto:techc@ualberta.ca) **from their UAlberta email address by 12:00 Noon on September 30, 2020.**
* **Problem definition and application idea must be approved by an instructor BEFORE all other planning takes place.**
  + **Have the group leader email your initial project proposal to** [**techc@ualberta.ca**](mailto:techc@ualberta.ca) **from their UAlberta email address by 4:00 PM on September 30, 2020.**

# Requirements:

## Full-Stack Requirements

* Create Persistent Records
  + Must demonstrate at least one data constraint.

(Example: A user’s first name or last name cannot contain numbers)

* + - Data constraint must be protected at every layer of the application.
  + Records must exist after the user has logged out, cleared their browser cache, and returned to the application.
* Read Records From Database
  + Sample data or dummy data must be stored in a database.
  + One of the following may apply:
    - You must include a .sql script with INSERT statements for your database.
    - README must include instructions on how to run this script.

OR

* INSERT equivalent (using Entity Framework in .NET Core)
* README must include instructions for installation and setup.
* Update Persistent Records In Database
  + Updates must enforce the same data constraint(s) that were mentioned in the create.
  + Record’s state must remain updated after user has logged out, cleared their browser cache, and returned to the website (this includes dummy data or any starting data your app has).
* Delete Records in Database
  + Must demonstrate the safe deletion of records.
    - Child records must never be orphaned.
    - No view may display any part of the deleted record, or any part of the deleted child records.
  + Choosing to ‘archive’ records rather than delete them is acceptable only if:
    - Child records are also archived.
    - User is presented with text stating that the record was archived and not permanently deleted.
  + Record must remain deleted after user has logged out, cleared their browser cache, and returned to the website

## Project Management Principles and Practices

* Problem Definition and Scope must be included in README.
* Project Plan must include a breakdown of tasks.
  + Task breakdown hosted on Trello.
  + Each card in the completed column must have a Member attached to it so we can see who did what at a glance.
  + Link to public Trello must be included in the README.
* Project Plan must include some form of wireframe (MS Paint, Draw.io, Figma, Balsamiq, etc.)
  + Include the final wireframe(s) and planning documentation in a “Planning” folder in your GitHub repo.
* Testing Plan must include instructions on how to test each feature.
  + Each test case must define the input, action and expected result for each test case (AAA testing).

(Example: User with firstname == “B0b” cannot register, user with firstname == “Bob” can register, user with name == “B” can register)

(Example: Hamburger nav is visible on phone-sized device, full sized nav is visible on tablet-sized device, full sized nav is visible on desktop sized device).

* List of test cases and testing instructions are included in README.
* Scope of the project must be well managed.
  + Unfinished features must not be apparent anywhere in the application.
  + Dead code must be deleted.
  + The problem that you defined in your problem statement must be solved.
  + Any features not related to the problem or that are out of scope should be deprioritized or removed completely.

(Focus on solving one problem and solving it well)

* Presentation must include the following talking points:
  + Final Project Report
    - Did you complete all in-scope tasks?
    - Did you complete any extra tasks?
  + Satisfaction Assessment
    - Does your app solve your problem?
    - Have you or someone else started using the app? What do they think of it so far?
  + Lessons Learned
    - If you had to build another full-stack CRUD app what lessons would you apply that you have learned from this project?

## Technical Requirements

* Data Validation must be done in each layer of the application.
  + Webforms must not allow users to enter invalid data.
  + Webforms must display validation errors with tips on how to resolve the errors.
  + Server layer must check all input values BEFORE processing or storing data.
  + Database must not allow invalid data to be stored.
* Data Persistence
  + All scripts to create the database and tables are included in the git repository.
  + All scripts to insert data are included in the git repository.
* Intuitive User Interface
  + Consistent navigation across all pages.
    - Links don’t grow or shrink or run away from the mouse.
    - Links are in the same place every time.
    - The link for the page we are currently on is highlighted in navigation in some way.
  + Buttons do what they say they are going to do.
  + Error messages explain how to fix the error.
  + Web controls are used appropriately and consistently.
  + On-screen instructions - if necessary - are easy to understand.
* Mobile-First
  + All content is legible on a small screen.
  + Content is contained on the screen without horizontal scrolling.
  + Buttons and forms all work on mobile.
    - Buttons are not too close together.
  + Form fields and labels are visible while typing with on-screen keyboard.
  + All key features needed to solve your problem are available in mobile mode.
* Responsive
  + Uses min-width media queries to manage different browser viewport sizes.
  + Uses fluid units of measurement consistently.
    - May use px measurements for min-width.
  + All content is legible.
  + Content is contained on the screen without horizontal scrolling.
  + All HTML pages have the appropriate meta tag for accessing the device’s width.
* Accessible
  + Source code and website passes [WCAG validator](https://achecker.ca/checker/index.php).
* W3C Compliance
  + HTML passes [W3C Validator](https://validator.w3.org/).
  + CSS passes [W3C Validator](https://jigsaw.w3.org/css-validator/).
* Separation of Concerns
  + Presentation layer contains only presentation code, and some data validation code to prevent users from entering erroneous data.
  + Business logic held in appropriate services, single-purpose principle is applied throughout the project.
  + Data storage and data access layer does not contain any data transformations and reinforces data validation.
* Browser Console
  + No error messages when using the application.
  + No console log messages while using the application.
  + Does not reveal sensitive user information.
* Code is commented
  + Citations must include full urls for any code found, borrowed, modified, from the internet, from the class, or from any source other than yourself. If you borrow code from a book, provide the title of the book, the author(s), and the ISBN.
    - Citations must be duplicated in README.
  + Comments describe what methods are meant to do.
  + Comments do not contain old/dead code.
* Naming Conventions
  + Proper casing on file names.
  + Proper casing on variables/functions/modules.
  + Variables/functions/modules have descriptive names.
* General Good coding practices
  + Look for blocks of code with more than a few lines of code that look similar. Is it possible to refactor to reduce duplication? DRY your code!
  + Simplify “too smart” and over-engineered code.
  + No hardcoded values, use constants values.
  + User input is sanitized (escape characters, leading and trailing whitespace, and inappropriate values are removed).
  + Spelling and Grammar
    - Contents of each web page is spell-checked.
    - Contents of each web page is grammar-checked.
    - Contents of README are spell-checked.
    - Contents of README are grammar-checked.
  + Use up-to-date language features

(const and let instead of var, section instead of div, string interpolation instead of string concatenation, etc.)

## Design Requirements

* Contrast
  + Font colours must pass the [Web AIM contrast checker](https://webaim.org/resources/contrastchecker/).
  + Contrast does not cause eye strain (black on white, red on green, etc).
  + Size contrasts are used correctly if size contrast is used at all.

(big buttons for important actions, small buttons for unimportant actions)

* Alignment
  + Content does not overflow from it’s boundaries.
  + Content appears organized and structured.
  + Text content is not center-aligned.

(exceptions permitted for poetry and wedding invitations)

* Repetition
  + Navigation placement is consistent across the whole application.
  + Button and link styles are consistent across the whole application.
  + Button and link behaviour is consistent across the whole application.
  + Text styles are consistent across the application.
* Proximity
  + Alike content items are grouped closer to one another.
  + No content is crowded against other content or the edges of the browser.
  + Controls are grouped together logically with their labels.
  + Proper use of whitespace.

# Deliverables

* GitHub link
  + README.md
    - Name of the project is included.
    - List of contributors is included.
    - Problem Statement is included.
    - Description of how the app solves the stated problem is included.
    - Instructions for installing the application are complete.
    - Instructions for using the application are complete.
    - List of all citations is complete.
    - List of test cases and testing instructions are included.
    - Link to the public Trello board is included.
  + Master branch contains all final code needed to run the project.
* Presentation
  + Final Project Report
    - Did you complete all in-scope tasks?
    - Did you complete any extra tasks?
  + Satisfaction Assessment
    - Does your app solve your problem?
    - Have you or someone else started using the app? What do they think of it so far?
  + Lessons Learned
    - If you had to build another full-stack CRUD app what lessons would you apply that you have learned from this project?
* Group Member Performance Review
  + Each group member has been evaluated.
  + All fields filled out.
* Personal Reflection
  + Written reflection.
  + Video reflection.