**Project Step 3 Draft Version: Design HTML Interface + DML SQL (Group / On Ed Discussion )**

* Due Thursday by 11:59pm
* Points 100
* Submitting a website url
* Available until Feb 14 at 11:59pm

**Aim**

In this step you create the HTML front-end for your web app to provide all the CRUD functionalities as described in the [CS340 Project Guide](https://canvas.oregonstate.edu/courses/1987790/pages/cs340-project-guide) while writing the queries that you want to run on the Database for providing each functionality.

**Deliverable**

A **ZIP** archive containing the 4 THINGS described below in detail:

1. A **.PDF** file containing information from previous steps
2. A **URL** to an index.html listing all your HTML pages. Put this URL on the top of your PDF file as well as on your discussion post.
3. The **.SQL file containing your Data Manipulation Queries.** These are the queries that your website uses to let your users interact with data. Thus SELECT, INSERT, UPDATE and DELETE queries to provide the functionalities described in the [CS340 Project Guide](https://canvas.oregonstate.edu/courses/1987790/pages/cs340-project-guide).
4. The **.SQL file containing your Data Definition Queries** and the **sample data INSERT statements** from the previous step

Make sure to name your zip file like **​projectgroupX\_stepX\_DRAFT/FINAL.zip.** e.g. **ProjectGroup42\_Step3\_DRAFT.zip**

**PDF File**

**a) Project Outline and Database Outline, ERD Schema & Sample Data Updated Version**

This section should contain the updated outline, ERD, Schema and sample data based on the feedback from the grader and your peers as well as any design decisions that you decided to make on your own. It's required that you apply the various design tools you learned until now (like Normalization steps, ON DELETE CASCADE, etc.) to review and fix your ERD and Schema. Your Final Project submission will be graded based on your application of these concepts.

**b) Fixes based on Feedback from Previous Steps:**

This section in your PDF should detail what all things you were told to fix in the feedback by the grader and your peer reviewers in previous steps and how you fixed them. If you chose not to fix things based on the feedback, this section would contain the reasons why you think things stay as they are. If you didn't receive any feedback for a specific step from grader or reviewers, please mention that.

Apart from this, if you are making any changes based on your own changed design decisions, they should also be listed here along with the reasoning.

**URL to an index.html page**

You need to **submit a URL of an index page linking all the other static pages** in your website (e.g. http://classwork.engr.oregonstate.edu:9124 *--or--* https://web.engr.oregonstate.edu/~yourONID/cs340/index.html) with a short but informative title for each page like "Browse existing ships", "Add new ship", "Manage certifications for each character".

These are the static HTML pages which the user will use to interact with your Project Website. Your HTML pages should be the front-end implementation for all the functionalities described in the [CS340 Project Guide](https://canvas.oregonstate.edu/courses/1987790/pages/cs340-project-guide). These can be just plain old HTML or if you wish to build them using the templating engine of the web technology (e.g. handlebars files) you choose that is fine, as long as each page is browsable and has the CRUD forms you plan to implement.

So basically, you need to create the client-side version of your site containing forms, any client-side validation that you want to perform (though not required), and any images, etc. that you want to be part of the website. Thus, the project website ***without*** any back-end or database interaction working.

You can simply deploy this website on the ENGR server inside the **public\_html** directory, or you can create a server running with a port number as you did in [Activity 2 - Connect webapp to database (Individual)](https://canvas.oregonstate.edu/courses/1987790/assignments/9888486). Keep in mind that the public\_html servers have outdated versions of Node.JS and Python which students cannot update. So, anything other than HTML (e.g. a templating engine such as handlebars, jinga or react) requires a server running with a port number on the ENGR server, similar to what you did in [Activity 2 - Connect webapp to database](https://canvas.oregonstate.edu/courses/1987790/assignments/9888486).

You need to include this URL in your PDF on the first page as well as on your discussion post.

For a sample website using the BSG database, I would turn in the URL to an **index.html** page listing all the other pages in the website with a short description of what functionality is available in what page. Here is [an example of what those files would look like](https://canvas.oregonstate.edu/courses/1987790/files/108859901?wrap=1)

[Download an example of what those files would look like](https://canvas.oregonstate.edu/courses/1987790/files/108859901/download?download_frd=1).

**One .SQL file should contain the Data Manipulation Queries:**

These are the queries that your website uses to let your users interact with data. Thus SELECT, INSERT, UPDATE and DELETE queries to provide the functionalities described in the [CS340 Project Guide](https://canvas.oregonstate.edu/courses/1987790/pages/cs340-project-guide).

Anything that is a variable that you expect the user to fill or to be computed by your back-end code and passed to MySQL/MariaDB should be shown using some special characters around it. This variable is generally used by your back-end code to pass some data to SQL server. You can use any bunch of special characters. Just make a SQL comment (like in the example below) showing what characters you will be using to denote the sample part.

**For example,**the [bsg\_sample\_data\_manipulation\_queries.sql](https://canvas.oregonstate.edu/courses/1987790/files/108860507?wrap=1)

[Download bsg\_sample\_data\_manipulation\_queries.sql](https://canvas.oregonstate.edu/courses/1987790/files/108860507/download?download_frd=1)

contains the following query.

-- Query for add a new character functionality with colon : character being used to   
-- denote the variables that will have data from the backend programming language  
  
INSERT INTO bsg\_people (fname, lname, homeworld, age)  
VALUES (:fnameInput, :lnameInput, :homeworld\_id\_from\_dropdown\_Input, :ageInput);

Here, :fnameInput is a way to denote the variable that will have the value of First Name in it, when the NodeJS code processes it. Again, you can choose any such way of denoting these variables.

We only want to see SQL here. Do*not*include any of the JavaScript or PHP or any other programming language's code used to process the data.

All queries should be syntactically correct (apart from the variable part described above) and relevant to other parts of your Project and the version of Data Definition Queries that you submit here. This means all your queries should work if the variable names are replaced with actual data.

**How do I turn in this assignment?**

1. Go to Ed Discussion and create a post called "Group XX Project Step 3 Draft". categorize your posting as *Project Step 3 Drafts & Reviews*. Communicate with your team so that you know who is posting the draft. Be sure that include *Both members' names and your Group number.*
   1. *Post the deliverables described (the ZIP file and the URL of your website) in the Discussions inside the Ed Discussion post for Project Step 3 Draft Version.*
2. Next, copy the URL of your Ed post and post it here on Canvas. Only one teammate needs to submit for the group.  Failing to post to canvas will result in a deduction.
3. Each team member should then [review two other drafts](https://canvas.oregonstate.edu/courses/1987790/assignments/9888513) -reviews are an important part of the process you are helping others while you are also learning. Strive to make thoughtful and substantial suggestions.

**How should I get started?**

I would suggest making a list of all CS340 Project functionalities in terms relevant to your database. For example "Add entries to each table" could be "Add a new BSG person", "Add a new Certification", and so on.

Then, draw a prototype of how you would your HTML page look like. For example, an "Add new BSG person" page would contain fields for first name, last name, a dropdown for Homeworld, and a field for age. Also, remember that you need not implement all your functionalities in different pages. I can have a delete button for each entry on the page listing all bsg\_people.

**Frequently Asked Questions:**

**Q. *I am using handlebars/some other templating engine for developing my project website. What should I submit?***

**A.** Just make sure the handlebar pages are browse-able with the right template syntax inserted.

For example, for the sample web app, if I were to submit only the handlebar file for bsg\_people, I would make available the people.handlebars file with the URL I submit as well as an index.html file listing all the pages and a short description of what they are.

**Q. *How many web pages should my project have?***

**A.** There is neither a minimum or maximum number of HTML pages that your project should have but it is desirable that your project is spread across different pages rather than a single one. You may combine or separate functionalities from the [CS340 Project Guide](https://canvas.oregonstate.edu/courses/1987790/pages/cs340-project-guide) across pages as you deem necessary.

**Q. *Do I need to show sample data on the HTML pages?***

**A.** While not necessary, it would be great. You may want to use some of the sample data from your previous project step. A webpage should definitely indicate with the correct section headings whether a section lists rows from a table or is a form for adding new rows to the tables or provides some other functionality.

**Q. *What do you mean by "front-end implementation for a functionality" on a static HTML page?***

**A.** For example, consider the Add people form for bsg\_people entity. The HTML page would contain a form with fields for all the columns (but not the ID since it's auto-incremented) and also a submit button. Of course, you would also have labels telling what each of these textboxes or dropdown are for. And a header on the page specifying that this form is for adding new person. You could also go one step ahead and make sure that the fields are displayed in a neat order, probably using a table since there will be points for styling in your Final Project submission.

**Q. *So these HTML pages will interact with the database?***

**A.** Well, they will actually interact with your backend programming language or framework like Flask/NodeJS which then interacts with the database. More on that in Week 7. We will get there soon enough.

**Q. *Should this particular field in my form be a textbox/textarea/drop-down/some new UX style fancy element ?***

**A.** This is your choice as the user should easily be able to understand how to use it (maybe if it's too difficult you could provide tooltips about how to use it). Your reviewers/graders might disagree and you should be able to justify the usage of one over the other.

**Q. *How "done" does the design of the HTML pages need to be for this step?***

**A.** "Done" could be answered by asking the question: Does the page provide all the necessary HTML elements for implementing a specific functionality? For example, the BSG People page would provide all the things necessary for implementing the Add New row to the bsg\_people.

**Q. *My website front-end uses AJAX to display rows and the website can't function without it/I have already written the server-side code to make my website work. Do I need to remove/hide these functionalities?***

**A.** You do not have to remove/hide any "extra" functionality or features for this submission that makes your front end work. You also do not need to scale down your project or disconnect the server-side code if it's already working. It's perfectly fine to implement a website with more functionalities than required for this Step.

All those extra things will simply not be considered by your reviewers and the graders when the Final version is graded. For example, the form's fields, the presence of a Submit button and the presence of relevant headers and labels on your form to make sure that anyone looking at the form understands what it is for and how to interact with the form, will be the only things reviewed and graded.

**Q. *Why do we have to work on this HTML thing? Isn't this course about databases? Why not just let me write SQL and get points for it?***

**A.** Databases do not exist in isolation. Unless all your users are comfortable writing SQL for tasks like inserting, and browsing data, you definitely need to provide them with an interface to interact with this database. And that is where HTML comes in pictures. Forms facilitate a way for providing data to put in your INSERT and UPDATE statements. The data that you get using SELECT statements will be displayed using tables in your HTML pages. And there probably will be a button somewhere which allows a user to DELETE a record in your table.

**Q. *I have already learned database interaction using a web technology like NodeJS in CS290. Why am I learning this now?***

**A.**CS290 offers an introduction to database interaction using a web technology, while this course gives you an opportunity to create a full-fledged database-drive website.

**Q. *My question was not answered in this Q&A. Where can I go to find the answer?***

**A.** Ask on Ed Discussion . We update the FAQ sections every quarter.

**Points**

This is worth 100 points.

You get points for simply turning in the work.