Project Guide

☰ Tags	
☰ Module	

Project Goals

- Implement all the database-related skills that you learn in this course as well as CS290.
- Understand how to transform an idea into a database.
- Learn how to develop a web-based UI for your database for providing **Create-Read-Update-Delete (CRUD) functionalities**.
- Learn to work in Steps towards a bigger goal.

Specifications

The CS340 Project that you will submit at the end of this course should satisfy all these specifications:

- Your database should be pre-populated with sample data. At least three rows per table is expected. The sample data should illustrate a table's functionality, e.g. if the table is part of a many-to-many relationship, the sample data should depict M:M.
- Your database should have at least 4 entities and at least 4 relationships, one of which must be a many-to-many relationship. The entities and relationships should implement the operational requirements of your project.
- You are creating a web interface for data tables, so the primary user is the administrator of this database.
    - It is NOT a customer facing website; thus there is **no need** for *login page; sessions; register/password; shopping cart; check-out; etc.* While having those pages would be helpful in many customer facing applications, the purpose of this project is to provide a UI for your tables.
    - Put another way, if you had 4 entities that were implemented as 5 tables in a database, then we expect roughly 5 web app pages as a front end UX for each of the 5 tables in your database.
    - One exception is oftentimes it works better for the UX to have a single web page for a Many-to-Many relationship between 2 tables (so the user doesn't have to view two pages to complete a transaction in both tables). So in that case if you had 4 entities that were implemented as 5 tables, with 1 many-to-many relationship between 2 of those tables, and

    the 2 tables in that m:m were managed on a single web page, then we expect 4 web pages in the project.
  - Some students may choose to add a home page to their project, which is acceptable but not required. To continue the example from the previous item, adding a home page would be a 5th page in the project.
- It should be possible to INSERT entries into every table individually.
- Every table should be used in at least one SELECT query. For the SELECT queries, it is fine to just display the content of the tables. It is generally not appropriate to have only a single query that joins all tables and displays them.
- You need to include one DELETE and one UPDATE function in your website, for any one of the entities. In addition, it should be possible to add and remove things from at least one many-to-many relationship and it should be possible to add things to all relationships. This means you need SELECT & INSERT functionalities for all relationships as well as entities. And DELETE & UPDATE for least one m:m relationship.
- Note that it's not acceptable to require the user to enter IDs for foreign keys. Instead your website needs to use a dynamically populated drop-down list or have the ability to search using text instead of entering in the ID. This Dynamic drop-down/Search functionality should be present for at least one entity.
- In one relationship, you should be able to set the foreign key value to NULL using UPDATE, that removes the relationship. In case none of the one-to-many relationships in your database has optional participation, you would need to change that to make sure one can have NULL values. For example, in the table Orders, there may be two FKs: the employeeID and the customerID which create relations to the Employees and Customers tables. It may not be sensible for the Customer to be optional. But the Employee could be optional. Thus, we would expect that in the Orders INSERT and UPDATE pages it is possible to set the employeeID to a value or else to NULL.
- You should be able to DELETE a record from a M:M relationship without creating a data anomaly in the related tables. For example, DELETEing a Customer should handle any Orders that were made by the Customer. This can be done by either by setting the CustomerID to NULL, or else by DELETEing any Order(s) associated with that Customer. More on how this can be done in Week 5 when we cover MySQL CASCADE.
- To continue the example from above, if you have 5 tables in your schema, then at a minimum, we expect you to implement 5 SELECTs, 5 INSERTs, 1 UPDATE (1 NULLable relationship), 1 DELETE (M:M), and 1 Dynamic drop-down/Search for a total of 14 functions.

General Rules and Grading

- Can be developed using any technology platform which serves content over the web.

- Should use an SQL driven database back end (e.g. MySQL/MariaDB, PostgreSQL, MSSQL, etc.).
- You should write all the queries and not depend on an ORM or similar mechanism to generate any queries.
- You can host it anywhere as long as it is accessible to the graders. You are required to work on the Project Steps in a Project Group of two students. By the end of Week 1, you are required to submit the names of your team members. It's recommended that you have a set weekly meeting time to check on the progress and also use tools like version control to keep in sync. You can change groups at most 1 time through the quarter, and then only if you find someone else ready to swap groups.
- Please note that the CS340 database created for you will be deleted from the server at the end of the term.

You will be working on this project over several weeks, earning points for each step completed. The distribution of points for each step is shown in this table:

Project Step	**Description**	**Points for Draft Version**	**Group Review**	**Points for Final Version**	**Total Points**
0	Form your group	N/A	No	25	25
1	Proposal and outline with ERD	125	Yes	150	275
2	Normalized Schema + DDL with Sample Data	100	Yes	NA	100
3	Design HTML interface + DML SQL	100	Yes	200	300
4	Add CRUD to one M:M entity HTML page (e.g. Invoices)	100	Yes	NA	100
5	Submit final completed project	NA	NA	600	600
	TOTAL	**400**		**1000**	**1400**

Project Steps

Starting from Step 1, you will submit a PDF file along with other required deliverables for each Step. The PDF (and other deliverables) should be modified based on the feedback you received in the previous Step.

This PDF file is a version log of your work on the Project which would include:

- Feedback that you received from your peers and grader in previous Step.
- Changes that you decided to make to your Project based on the feedback.
- Reasons why you decided not act on the feedback.

Thus, it's a living document where you would have this information in reverse chronological order (previous Step first) on top. With completion of each Step, the document should still make sense as a whole. I recommend using a Google Doc or Box doc to collaborate on this document with your teammate.

Overview

You will create a web-based interface to a database with entities that you develop. You can choose to use the NodeJS, Python or use any technology as long as it satisfies the criteria listed above (e.g. no use of ORM for creating queries).

You also submit a PDF describing your Project and the Database you create. You will submit SQL statements and code which will be reviewed.

Steps 1 & 3

For Steps 1 and 3 there are THREE submissions each:

1. **Draft version** This is due early and you will earn points for turning in the work. You submit this to your Discussion group that the instructors will create for each step requiring a draft.
2. **Review** You will receive points by reviewing your peers' work. The review process will mean that you get reviewed by peers and you review multiple submissions. This gets you points for participation and helps you learn from others.
3. **Final version** After you receive peer feedback on the draft you will update your project and then turn in a final version. This will be graded by the TAs based on the rubric for validity.

Example case:

- You work on Step 1 Draft with your Project Group and post it in Discussion.
- You get a peer review telling you that one of your entities does not make sense and that the data type for one of your attributes is wrong.

- You include this review in your Step 1 Final version submission. You justify why you decided to keep the entity as-is but decide to act on the feedback about the wrong data type.
- You also change the data type in the Outline that you are submitting for Final version.
- You get feedback from the TA that another attribute that you have should have a different data type.
- You fix (*or don't fix and mention the reason*) the data type in your Outline when turning in Step 1. Thus for Step 1, not only does your ERD and Schema reflect this change in Project, but you also change your Database Outline.

Steps 2, & 4

For Steps 2 and 4, there are only TWO submissions:

1. **Draft version** You get points simply for posting a Draft version.
2. **Reviews** For every project step draft, you make a posting to the Discussion thread where groups can post links to their drafts. If you turn in the Draft version, you will get a chance to earn points by reviewing other groups' drafts. Find any **two** drafts and post your suggestions and comments as a reply. The graders will read your reviews and give you points accordingly.

Step 5 (portfolio assignment)

For Step 5, you turn in a completely working website URL and the accompanying PDF as described above. You will be graded by the graders and get points based on the rubric. This submission is intended to be something you include in a portfolio for potential employers to view your work.

Groups

Project Group

This is the group you work in to create the CS340 Project. Please refer to the Syllabus Policy on Group Work which has guidelines about group expectations. You decide whom you want to team up to work on the Project by the end of Week 1 and submit on Canvas. A Project Group can have maximum 2 students*. **Both members will get equal points for every Steps' Final Version but the points for Draft and Reviews may be based on individual participation.** We recommend using a collaboration tool such as GitHub which tracks contributions made by each student to the project.

If you don't choose a teammate, you will be assigned randomly with another student who doesn't have a teammate.

You can change your group only once through the term and to change you need to find another student willing to swap. To swap, email the instructor while copying teammates from both groups.

If your group member drops the class, contact the instructor and we will help you find a new group.

Exceptions will be made in special circumstances.

Frequently asked Questions

- **Q. How do I work on the Project? I don't understand/know HTML/ERD/Schema/this spec about M-to-M relationship updating/other such specs.**
- **A.** My advice is to keep an eye on the bigger picture but work on one Step at a time. The reason we implement the Project in multiple steps in this class is so that it's easy to take on this mammoth one step at a time. Don't worry about web development until you get to that week since your project is going to evolve with each step, reviews and feedback that you receive from your peers as well as graders. As your project evolves and we progress further in the class, many of the specs and how to implement something in your project will become clearer.
- **Q. I have a question about the specs for this Step...what do I do ?**
- **A.** Ask in office hours/Ed Discussion , describing the relevant details of your Project so that we can help you. Implementations are hard to describe in generic manner and it always helps to talk in terms of examples.
- **Q. How should my final website look like ?**
- **A.** The looks don't matter much for the Project. I would advise worrying about the design of your database and your queries right now and take care of the the HTML in the later weeks.
- **Q. Can I use X technology and Y framework to build the website for my CS340 Project?**
- **A.** Yes, you can use any technology and any framework to build your web app for the project in this class as long as you write all the queries yourself and do not use anything like an ORM to interact with your database. For example, Django defaults to using ORM for database layer, so you might want to avoid using that unless you can figure out a way to around it. The class will officially host content on NodeJS as well as a bit of Python and also provide sample code that you can build upon but I am happy to help with Python, PHP as well as Ruby. But do not completely rely on the TAs or me to debug your code or figure out a way to do something with any technology. It's your choice of technology and your job to figure out how to do things using it! :)
- **Q. I am getting my data from a live source (e.g. a web API). Do I still need to provide CREATE functionality?**
- **A.** No, however there should be at least two tables in your database that offer CREATE functionality. If at most two of your tables don't have it because the data is coming from a web API then that's OK as long as the other tables do.