(25 points) Milestone 4: Final Deliverable

Due: 11:59pm, Sunday, Nov 30, 2025

No late submissions will be accepted.

Purpose:

- Write program(s) to interact with the database
- Implement all functionalities and necessary features to fulfill the project requirements
- Test and verify that your app works properly

This is the final development step!

You will work in your project group to develop a software that uses a relational database to provide services.

Minimum requirements:

- Write programs to interact with the database you designed and set up in previous milestones
- Implement all functionalities and necessary features you proposed to fulfill the project requirements
- Include **dynamic behaviors,** where the front end (user interface) respond to user inputs or actions (or web services) and updates the screen (interface) accordingly
- Allow users to perform the following functionalities. It is important to note that these functionalities must be done t**hrough your software;** users are not allowed to access your database directly. Manually performing any of the following functionalities (e.g., through phpMyAdmin or MySQL/SQL/DBMS command line) does not satisfy these requirements.
    - **Retrieve** data from the database
    - **Add** data to the database
    - **Update** data in the database
    - **Delete** data from the database
    - Do at least **one** of the following
        - Sort data
        - Search or filter data
        - Export data (from the database) in some forms (such as JSON, XML, HTML, or CSV)
        - Upload / import data (in some forms such as JSON, XML, HTML, or CSV) into the database
- Support **multiple users**, allowing users to use the app **simultaneously** without interfering with each other.
    - Your database must be shared or hosted on a public server to provide centralized services — allowing your app to **access a single shared database** that can be accessed by multiple users concurrently.

        - Setting up the same database locally on multiple machines is not directly considered a "shared database." In fact, they, simply, are duplicate databases that need to be synchronized either manually or using the concepts of distributed database systems.
        - To be realistic, try to avoid setting up a database that can only be accessed by a single machine and that needs to be synchronized manually.
        - To simplify the process and minimize the overhead (in terms of configuration), do not set up a distributed database system.
    - If you implement a web-based project **(recommended)**
        - A simple way to test this feature is to open multiple different web browsers, then on different web browsers try to access / use the service your app provides concurrently.
    - If you implement a non web-based project
        - Install your app on each user' machine. Host your database on a public server so that it can be shared. To test this feature, try to access / use the service your app provides from multiple machines concurrently.
        - Alternatively, you may set up multiple virtual environments on a machine; each environment represents each user's machine. To test this feature, try to access / use the service your app provides from multiple environments concurrently. Please spare time for troubleshooting and plan your time very well.

- Support **returning users** — allowing returning users to access / retrieve / manipulate their existing data.
- Set up **database security at the database level**
    - This typically involves limited access or privilege on the database system and may differentiate the levels of privileges of
        - Developers (you and your teammates) — which tables can be accessed and what operations an individual developer can perform on the tables
        - Users (end users who use your app) — which tables can be accessed by your app to provide services to an individual user (e.g., consider different categories of users: students vs. instructors; which tables / operations a student user can access / perform, which tables / operations an instructor user can access / perform)
- Incorporate **database security at the application level**
    - This may involve, for example,
        - User authentication to use some (or all) services your app provides
        - Password hashing
        - Control of execution flow such as redirecting a user to a certain portion of your service
        - Providing users with different views or interfaces, limiting access to certain data or services

- Your project must meet the minimum number of table requirements. All the normalized tables must be set up in the previous milestones and all of them must be utilized in some way by your app.

What to submit: Submit a final report in five parts

1) **Database design**
   - **a)** Include a final version of your E-R diagram
   - **b)** Include a final version of your tables written in schema statements
2) **Database programming**
   - **a)** Specify where you host your database
   - **b)** Specify where you host your app — the deployment environment needed to deploy and run your project
   - **c)** Describe and include all instructions / steps needed to deploy and run your project
     - **i)** Note: we may randomly set up, deploy, and run your project on a similar environment. It is important that you provide the description and a complete set of the instructions.
   - **d)** Refer to advanced SQL commands your wrote in the previous milestone, clearly discuss how the advanced SQL commands are incorporated in your app, what features of your app use them, and how they reflect the database
3) **Describe the database security at the database level**
   - **a)** Specify whether the security is set for developers or end users
   - **b)** Discuss how you set up security at the database level (access control)
   - **c)** Submit the SQL commands you use to limited / set privileges such as:

```
GRANT privilege-name ON object-name TO {user-name | PUBLIC | role-name}  or
REVOKE privilege-name ON object-name FROM {user-name | PUBLIC | role-name}
```

4) **Describe the database security at the application level**
   - **a)** Discuss how database security at the application level is incorporated in your project.
   - **b)** Submit code snippet(s) to illustrate how the security aspect is implemented and to support your discussion.
5) **Peer evaluation (due when this milestone is due)**
   - This is an individual task. Do not include it in the project's report.
   - The teaching team will consider this peer evaluation (along with the other peer evaluations and other deliverables) when assigning the project final grade to an individual team member. Each team member's grade may be adjusted by 0%-100% deduction, based on his/her contribution.
   - Submit your peer evaluation: https://docs.google.com/forms/d/e/1FAIpQLSfYSxIUOWoWsGDKaeEqZ1DlcIrZF0ED0jPYyRzJf7WSnmc1cQ/viewform?usp=sf_link
     - **Everyone is required to submit the peer evaluation**

- You are required to enter the names and NetIDs of all team members
- Once this form is closed, the form will not be reopened and we have to assign a zero grade to this section of the rubric.