# CS5200: Milestone 4 (Team)
# Final Implementation

Spring 2025

## 1 Purpose

In this assignment, your team will construct a basic web application that interacts with the database that you have built in previous milestones. By completing this assignment, you will gain experience working with web applications that interact with databases and construct a basic interface that allows users to interact with the data-access operations that you defined in the previous assignment.

## 2 Deliverables

Your team will construct a JSP web application that implements several interactions with your game database, described further below. You will have two primary deliverables for this milestone: you must upload a copy of the source code for your application, and you must demonstrate a working instance of your application in class on the due date.

### 2.1 Submission

Your team must upload a single ZIP file to Canvas that contains the following items:

- Model classes (as Java source code) for those tables in your database that are necessary to implement the database operations. These will most likely be very similar if not identical to those you submitted as part of PM3, but please include these files in your PM4 submission as well. Note that you only need include model files for those tables that you need to access to support the operations described below. If a particular table is not used in these operations, you do not need to include its model class.
- DAO classes (as Java source code) for those tables necessary to implement the required database operations. You may use the DAO classes you submitted as part of PM3 as a starting point, but you should expect to implement additional methods.
- JSP files defining the web pages displayed in your application.
- Servlet clases (as Java source code) that implement the behavior of the JSP web pages.
- A Driver.java file that defines a class with a `main` method that creates the tables in your database and populates them with at least 10 sample records per table. (For enumeration tables, the requirement of 10 records does not apply; instead, you should insert exactly those values into your enumeration tables as required by the application.)

As with PM3 and HW5, you are encouraged to use the ConnectionManager.java and Utils.java files from the sample application, with a comment giving credit as appropriate.

We should be able to compile your code, run your Driver program, deploy your application to Tomcat as demonstrated in class, run your application, and interact with your web pages.

As discussed above, the point of this project is *not* to produce a polished, modern web application; that is out of scope for this course. (Web application development is a useful and valuable skill, but it is covered in CS5610.) In this course, we are concerned entirely with the application's functionality, so a user interface similar to that of the sample application I have distributed is sufficient. If you wish to improve on the application's look and feel, you are free to do so, but I encourage you to ensure that the core features are implemented fully and correctly first, as that is where the bulk of your grade comes from.

## 2.2 Required Operations

Your application should be able to perform the following operations:

1. Produce a meaningful list-based report showing a high-level summary of some selection of entities in your database. This report must support the following functionality:
    - Users must be able to filter the list on at least one attribute that is meaningful to an end user (i.e., author name, not internal author ID).
    - The filtering operation must support partial matches.
    - The list must display values that are meaningful to a human reader thinking in terms of the entities described in the original problem definition. That is, avoid displaying synthetic keys.
    - Each item in the list must contain a hyperlink that expands into the report in operation #2.
    - The items in the list must be sorted in a meaningful way. (You can choose to allow the user to specify the sort key(s), but be careful here!)
2. Produce a meaningful detail report on one of the entities in the list in the previous section.
    - The user must be able to access this report by clicking on one of the links in the list generated by the previous operation.
    - The detail report must be complicated and thorough enough to require combining information from at least three separate tables.
    - The detail report must include both data from the original summary report as well as more detailed information about the selected entity.
    - Again, in information displayed to your application's user, avoid displaying synthetic keys.
3. Allow the user to update at least one table based on your reports.
    - There should be a link or button on the detailed report that allows the user to navigate to this update operation.
    - The table that you update in this operation must not be an enumeration table.

**All submitted code must use the techniques demonstrated in lecture to guard against SQL injection attacks. We will impose a penalty of up to 50 points for insecure code.**

## 2.3 Example Operations

A concrete example may help clarify some of these requirements. If we were to construct the application based on the blogging database we've been using as a running example in class, then the application could do something like the following:

1. List summary information (poster, title, date created) for each blog post in the system.
    - Allow users to filter the list based on title or author.

- If the user chooses filter the list based on title, they should be able to type in a few words and display all posts whose titles contain that phrase.
- The list should not display postID values to the user.
- The post title in each row in this list should be a link to a page that displays more detailed information about the post (see point #2 below)
- The list of posts should be sorted by author, or by date, or by title (possibly according to the user's selection).

2. Provide detailed information about a single blog post.
    - In addition to including the information from the summary list in the previous point, we could also display the author's full name (and not just their userID) and a list of all the comments that people have written for this post.
3. Allow the user to update some property of the post displayed in operation #2. For instance, you could allow the user to edit the post's title.

## 2.4 Presentation

One of your team members must demonstrate a working copy of your team's application in class on April 17th or 18th. As before, we'll have a Teams meeting to which presenters can connect in order to display their application on the classroom screens. (We've had problems in the past with screen sharing when attempting to run Teams entirely within the browser; we recommend using the free standalone program instead.)