# SOEN 387: WebBased Enterprised Applications Design Assignment 2 on Layered Architecture, Data Formats, and using Databases

Fall 2021, sections F

October 27, 2021

## Contents

# 1 General Information

**Date posted:** Wednesday October 27$^{\text{th}}$, 2021.

**Date due:** Thursday November 18$^{\text{th}}$, 2021, by 23:59[1].

**Weight:** 8% of the overall grade.

# 2 Introduction

This assignment targets implementing web applications with an emphasis on 1) layered architecture, 2) handling data and file formats, and 3) using databases.

# 3 Ground rules

You are allowed to work on a team of 3 students at most (including yourself). Each team should designate a leader who will submit the assignment electronically. See Submission Notes for the details.

ONLY one copy of the assignment is to be submitted by the team leader. Upon submission, you must book an appointment with the marker team and demo the assignment. All members of the team must be present during the demo to receive the credit. Failure to do so may result in zero credit.

# 4 Overview

In this assignment we extend the Poll system we created in assignment I and add the following features: 1) Adding a data persistency to support archiving and restoring Polls, 2) Supporting Users and Authentication, 3) Configurations

**Notes:**

1. The detailed description of the additional system functionalities are explained in the next session (see 5).

[1] see submission notes

2. You may re-use the skeleton and most components you created in **Assignment I**.

3. You may want to perform a proper refactoring, so that the new requirements are fully addressed.

# 5 Your Assignment

This assignment extends the web-based poll application system that you made in the previous assignment. The assignment consists of the following parts: 1) The Poll Business Layer, 2) Data Persistency, 3) The Poll Servlet, 4) The Web Front-End, 5) Documentation and Source Control.

The added features are provide in the following sections. See also the **non-functional requirements** listed under the "Implementation Platform and Requirements" section in the previous assignment.

## 5.1 The Poll Business Layer

Similarly to the previous assignment, the business layer is to be implemented in a stand alone class library project. This business layer will be responsible for the poll system functionality and should not have any dependency on any web technology. This assignment does not specify any specific classes to implement. You may use any design or any number of classes to implement the business layer (see 1).
The functionality of the new Poll System is outlined in the following.

The system it to be implement as a "prototype", and as a result, not all features of the poll system are available (e.g. user registration). The options to those functions must be present in the system; however, when the user wishes to select such unimplemented features, a proper error message must be given to the user indicating that the feature is not implemented at the moment.

***Users and Authentication***

The user management component is in a prototype state and is not fully functional. The

system currently uses a set of pre-defined users with pre-set passwords. In future, users will be able to **sign-up**, **modify** their accounts, and **change their password**.
**Note:** All passwords must be hashed and not kept in plain text. To do so, you may use any hashing algorithm such as **MD5** or **SHA** (see 3, 4).

***User and Polls***

The new system allows every user create a poll or vote on a poll (created by self or others). The system allows multiple votes be created and run at the same time. All votes are anonymous. Logging in to the system is required for voting. Polls votes are persistent.
An authenticate user has a *Poll Manager* role (see previous assignment). All previous functions and business rules apply, with the following exceptions:

1. Create a Poll: automatically time-stamped by the system, a poll is assigned a randomly unique identifier (a **non-case-sensitive** 10**-char long random string**, containing letters **A-Z** (excluding [ILOU]) and digits **0-9**). This ID will be used by participants for voting[2].
2. Delete a Poll: a Poll may be deleted only by the user who has created it and only if it has not been voted on.
3. Access a Poll: identified by the 10-char poll id, to vote on or to view the poll result.
4. Close a Poll: Archives the Poll and its data. The Poll may no longer be used. Only a person who created the Poll may view the poll data.
5. Get List of Polls: to view all polls created by the user, including their statuses (i.e. created, running, etc.).

Any user whether authenticated or not may be considered as a voting *User* (a person who can vote). A user may do the following:

1. Access a Poll: identified by the 10-char poll id, to vote on or to view the poll result. See the previous assignment on when a poll may be accessed.

[2]Note that there are 32 combinations. You may create a random 50-bit string and represent it as a 10-char string

2. Vote: provided the 10-char poll id, a user may vote on the poll. To support vote updates, a user is given a 6-**digit randomly generated PIN#**, identifying the user's session, specific to this very Poll. Users may later on use this very pin to update their votes. See Request PIN.
3. Request PIN#: provided the 10-char poll id, a user may request a PIN#. Every user is eventually assigned a PIN#. A PIN# is used by the user to identify the voting session. A user is assigned a different PIN# for every Poll. The PIN# is only unique within a Poll. It is possible that same PIN#s may be used by different users in different Poles. A returning user may enter a previously generated PIN for the pole ID and continue as the user who already voted. A Poll manager, as a user, may be assigned a PIN# to vote.

## 5.2 Data Persistency

The data persistency may be implemented within the Business layer or in a completely separated layer.

***Users Data***

The credentials of authenticated users (poll managers) are stored in a readonly data store (a lightweight database or a text file in **CSV**, **XML**, **JSON**, or any other readable format). Users information contain: user-id, full name, email address, and hashed password.

***Polls Data***

Polls Data are stored in a separate database. It contains The data for the Polls and the Votes. Every vote is associated a PIN number (see section 5.1).

***Configuration***

All system data (i.e. path to the system files, connection string to the database, etc.) must be stored and managed in a *configuration file*.

## 5.3 The Poll Servlet

In addition to the functionality, specified in the previous assignment, The servlet exports the data in two additional formats: **XML**, **JSON**. The format is given by the format string query parameter.

## 5.4 The Web Front-End

Implement the new UI component to support additional features as listed in section 5.1. Note that the system does **NOT** provide a "Search for Poll" feature.
It is assumed that a voting user, is already given the poll-id. The poll id may be entered in a text-box. An separate text-box must be available so that the user enters a previously given PIN#. Note that the PIN# is optional and in case is missing, the system displays a randomly generated PIN# that may be later on used to modify the vote. A current user, may always be able to switch to a different PIN# to identify themselves as a different user.

***Unimplemented Features***
*Unimplemented features* such as user sign-up and registration links must be present on the Web-UI. However, the service is unavailable. Upon clicking on those links, the system may display an error message indicating that the feature is not yet implememnted.

## 5.5 Documentation and Source Control

### 5.5.1 Source Control

This assignment too must be done using a source control system. It is recommended that a new branch is created for this assignment by forking the code from the previous workspace. Document changes and store them in the source control. Include the following diagrams in the source control as well.

### 5.5.2 The README File

The **README.md** file contains the team information, the description of the application, and the release notes (instructions on how to install the software components);

### 5.5.3 UML Class Diagrams

After the project is done, draw the complete updated UML class diagram of the Business Layer and include it in your submission. The class diagram contains classes and associations in the business layer. In case you use a separate Persistency Later, separate classes into logical packages and include the dependencies among classes.

### 5.5.4 The Entity Relationship Model

Provide the **Entity Relationship Model** of the system (see 2) in the form of an ERD or a UML diagram.

### 5.5.5 Sequence Diagrams

Provide the **Sequence Diagrams** for the following scenarios. Include all classes from all layers that are involved in each case.

1. A poll data download scenario.
2. A create poll scenario.

# 6 What to Submit

The whole assignment is submitted by the due date under the corresponding assignment box. It has to be completed by ALL members of the team in one submission.

## Submission Notes

Clearly include the names and student IDs of all members of the team in the submission. Indicate the team leader.

IMPORTANT: You are allowed to work on a team of 3 students at most (including yourself). Any teams of 4 or more students will result in 0 marks for all team members. If your work on a team, ONLY one copy of the assignment is to be submitted. You must make sure that you upload the assignment to the correct assignment box on Moodle. No email submissions are

accepted. Assignments uploaded to the wrong system, wrong folder, or submitted via email will be discarded and no resubmission will be allowed. Make sure you can access Moodle prior to the submission deadline. The deadline will not be extended.

Naming convention for uploaded file: Create one zip file, containing all needed files for your assignment using the following naming convention. The zip file should be called a#_studids, where # is the number of the assignment, and studids is the list of student ids of all team members, separated by (_). For example, for the first assignment, student `12345678` would submit a zip file named `a1_12345678.zip`. If you work on a team of two and your IDs are `12345678` and `34567890`, you would submit a zip file named `a1_12345678_34567890.zip`.
Submit your assignment electronically on Moodle based on the instruction given by your instructor as indicated above: `https://moodle.concordia.ca`
Please see course outline for submission rules and format, as well as for the required demo of the assignment. A working copy of the code should be submitted for the tasks that require them. Archive all files with any archiving and compressing utility, such as WinZip, WinRAR, tar, gzip, bzip2, or others. You must keep a record of your submission confirmation. This is your proof of submission, which you may need should a submission problem arises.

# 7 Grading Scheme

| | |
|---|---|
| Data Persistency | 15 marks |
| The Business Layer | 20 marks |
| The Web-UI | 20 marks |
| The Servlet | 15 marks |
| Proper Error Handling | 10 marks |
| Diagram and Documentation | 20 marks |

**Total:** 100 marks.

# References


1. Martin Fowler, "Patterns of Enterprise Application Architecture (EAA)", ch. 2: Organizing Domain Logic, 2002, ISBN-10: 0321127420, ISBN-13: 978-0321127426
2. The Entity Relationship Model:

   `https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model`
3. Java support for MD5: `https://www.baeldung.com/java-md5`
4. Java support for SHA: `https://www.baeldung.com/sha-256-hashing-java`
5. Online UML tools:

   - `https://yuml.me/diagram/scruffy/usecase/draw`
   - `https://yuml.me/diagram/scruffy/class/draw`
   - `https://plantuml.com/state-diagram`
   - `https://sequencediagram.org/`
   - `https://plantuml.com/ie-diagram`