LaSalle College
Montréal

Final Evaluation: 40%

Course Identification

Name of program – Code:	COMPUTER SCIENCE TECHNOLOGY - PROGAMMING (420.BP) INFORMATION TECHNOLOGY - PROGRAMMER-ANALYST (LEA.3Q)
Course title:	**WEB SERVER APPLICATIONS DEVELOPMENT I**
Course number:	420-DW3-AS
Group:	07250
Teacher's name:	Marc-Eric Boury
Duration:	3 periods (150 minutes)
Semester:	Winter 2023

Student Identification

Name: _____ Student number: _____

Date: 25/04/2023, 8:00 AM Result: _____

☐ I declare that this is an original work, and that I credited all content sources of which I am not the author (online and printed, images, graphics, films, etc.), in the required quotation and citation style for this work.

Standard of the Evaluated Competency

Statement of the evaluated competency – Code

Develop transactional Web applications – 00SU

Evaluated elements of the competency

1. Analyze the application development project.
2. Prepare the computer development environment.
3. Prepare the database.
4. Program the Web interface.
5. Program the server-side application logic.
6. Program the client-side application logic.
7. Control the quality of the application.

Instructions

- Permitted equipment: anything
- No break is allowed during this exam. Students are not allowed to exit the examination room before half of the allotted time has passed. Once a student has exited the classroom, he/she may not re-enter (IPEL – Article 5.12.4).
- The teacher will not answer questions during the exam.
- Students must remain silent during the exam.
- It is the teacher's responsibility to identify language errors. If such errors are found, teachers may apply a penalty of up to 10% of the grade (IPEL – Article 5.7).
- Plagiarism, attempts at plagiarism or complicity in plagiarism during a summative evaluation results in a mark of zero (0). In the case of recidivism, in the same course or in another course, the student will be given a grade of '0' for the course in question. (IPEL – Article 5.16).
- Please write clearly.

Mark Breakdown

This evaluation is on 100 points, distributed as follows:

- Development of a transactional web application. For a total of 100 points

TOTAL: 100 POINTS

Reminders:

- Create your own code. Copying from **any** source will not be tolerated.
- Do not forget to export your database and add the generated script to your project before submitting this evaluation

Context

You are tasked with creating a small web application to implement the four basic operations (create, read, update, delete) with video game entities and store the data in a database. The application must have two web pages: one containing a form for entity creation, and one containing a form for data display, edition and deletion.

To achieve this, **you have received an existing project containing organized files and some preexisting code that you must complete**. The project is marked with "TODO" tags indicating where work is required.

Requirements

- The application must be written in PHP, HTML, JavaScript and optionally CSS.
- No framework or other external libraries can be used in the application.
- With the exception of anything that was received in the exam package, no code can be copied from any external source.
- The application default page (index page) must redirect to the entity creation page.
- Use of Object-oriented PHP is required.
- The application must make use of a MySQL Database to store and retrieve data from.
- Use parameterized statements in all SQL queries and statements.
- Use PDO as the PHP database access extension.

Part 1: Setup (5%)

- Extract the exam project from the provided zip archive in your *htdocs* directory of your XAMPP installation.
- Ensure that the directory structure from the zip extraction is correct and matches the following:

  <your htdocs directory>
  └> *<project directory>*
  └> *<project files (including index.php) and sub-directories>*
- Complete the index page (located in the project root directory) by filling your name and student number, and by implementing a basic redirection to the creation page.

Part 2: Creation of the database (20%)

- Create a database named "420dw3_final_summative".
- In that database, create a table named "games" with the following columns:

Col. Name	Col. Type	Extra informations
id	INT	NOT NULL, PRIMARY KEY, AUTO-INCREMENT
title	VARCHAR(128)	NOT NULL
genre	ENUM	ENUM VALUES : Strategy, Simulation, RPG, FPS
developer	VARCHAR(128)	NOT NULL
release_year	YEAR(4)	NOT NULL
date_created	DATETIME	NOT NULL, DEFAULT : CURRENT_TIMESTAMP()

Resulting in a table schema like this:

Insert the following test values in your created table:

id	title	genre	developer	release_year	date_created
(auto)	Counter-Strike	FPS	Valve	2000	(DB default)
(auto)	Dark Souls	RPG	FromSoftware	2011	(DB default)
(auto)	Hearts of Iron II	Strategy	Paradox Development Studio	2005	(DB default)

Part 3: Developing the Business Logic (20%)

- Complete the “VideoGame” class to serve as a model and object type for the required entity (located in *private/src/models/VideoGame.php*).
  - Use encapsulation methods to validate acceptable values for the model.
  - You will need to implement in some way the four basic entity database interactions (create, read, update, delete).
- Complete the “PdoConnector” class to serve as a service to create and obtain PDO connection objects for use in your code (located in */private/src/PdoConnector.php*). You **must** use the PDO PHP extension to handle database interactions.
  - When creating connections to the database, **you must read the connection parameters from the *config.json* file given in the project package** (located at */private/config/config.json*). Use the PHP file input/output and JSON management functions to directly read the data from the file. You can, if made necessary by your own system setup, make changes to the connection data in the file such as the username/password

or port number.

- Complete the “api.php” PHP script (located at *public/endpoints/api.php*) that analyzes the received requests and implement the three required operations (create, update, delete) in interactions with the database.
- **All business logic must be handled with objects; data received from the client must be converted, inserted or wrapped into model object instances.**

Part 4: Entity creation page

- Complete the entity creation page (located at *public/pages/create.php*) with an HTML form for the creation of entities.
  - The form inputs must be of a correct type for the data they handle.
  - The form inputs must implement basic client-side validation to limit acceptable values.
- Submitting the form must trigger the creation in the database of a record for an entity with the user input values.
  - use a manually defined AJAX “POST” request, not the default form behaviour, to transmit the data to the server.
    - Achieve this by completing the creation page’s JavaScript script (located at *public/js/create.js*).

Part 5: Entity display, edition and deletion page

- Complete the entity view/modification/deletion page (located at *public/pages/view.php*) with two HTML forms, one to view and modify a specific entity and the other to delete it.
  - The page display the information about a specific entity; so it must receive an id from the request.
  - The view/edition form must be filled with the entity data when the page is loaded (retrieve the entity from the request’s passed id).
  - The forms inputs must be of a correct type for the data they handle.
  - The forms inputs must implement basic client-side validation to limit acceptable values or block undesired user interaction.
  - For the view/modification form, make certain that any entity properties that should not be editable by the user are made not editable (any property whose value is handled by the database: id, creation date...).
- Submitting the forms must respectively trigger the modification or deletion in the database of the entity record.
  - use manually defined AJAX “POST” requests, not the default form behaviour, to transmit the data to the server.
    - Achieve this by completing the view page’s JavaScript script (located at *public/js/view.js*).

Part 6: Testing your application

- Using your completed and functional web pages, create a new game model with the following values: **title**: “The Sims 3”, **genre**: “Simulation”, **developer**: “The Sims Studio”, **release_year**: “2009”.
- Verify the creation of the record in the database.
- Verify the correct display of the record in the redirected-to page.
- From the edition page, modify the record’s **title** to be “The Sims 4”, its **developer** to be “Maxis” and its **release_year** to be “2014”.
- Verify the modification in the database.
- Delete the modified record record from the database.
- Verify the deletion in the database.

Submission directives

1. Export your database into a SQL file. Make sure to check the following custom options (keep the other options as they are by default):
   a. Format-specific options:
      1. ***“Disable foreign key checks”*** is **ON**
   b. Object creation options:
      1. ***“Add CREATE DATABASE / USE statement”*** is **ON**
      2. ***“Add DROP TABLE / VIEW / PROCEDURE / FUNCTION / EVENT / TRIGGER statement”*** is **ON**
      3. ***“IF NOT EXISTS (less efficient as indexes will be generated during table creation)”*** is **ON**
2. Copy your exported SQL database script somewhere inside your code project directory.
3. Zip the entirety of the project directory (which should be named “420DW3_07250_Final”) that contains your code, database script and any other front-end files. Keep the same name for the archive as the project directory.
4. Submit the zipped archive on LEA.

CORRECTION GRID

Part 1: Setup (5%)

Element of competency: Prepare the computer development environment (00SU.2).	
Performance criteria	**weight**
2.1 Proper installation of the Web development platform and the development database management system	/3
Element of competency: Control the quality of the application (00SU.7).	
Performance criteria	**weight**
7.5 Compliance with design documents	/2

Part 2: Creation of the database (20%)

Element of competency: Analyze the application development project (00SU.1).	
Performance criteria	**weight**
1.1 Accurate analysis of design documents	/5
Element of competency: Prepare the database (00SU.3).	
Performance criteria	**weight**
3.1 Suitable creation or adaptation of the database	/5
3.2 Proper insertion of initial or test data	/5
3.3 Compliance with the data model	/5

Part 3: Developing the business logic (20%)

Element of competency: Analyze the application development project (00SU.1).	
Performance criteria	**weight**
1.2 Proper identification of the tasks to be carried out	/5
Element of competency: Program the server-side application logic (00SU.5).	
Performance criteria	**weight**
5.1 Proper programming or integration of authentication and authorization mechanisms	/1
5.3 Appropriate choice of clauses, operators, commands or parameters in database queries	/5
5.4 Correct handling of database data	/5
5.5 Appropriate use of data exchange services	/2
Element of competency: Control the quality of the application (00SU.7).	
Performance criteria	**weight**
7.2 Thorough reviews of code and security	/2

Part 4: Entity creation page (15%)

Element of competency: Program the Web interface (00SU.4).	
Performance criteria	**weight**
4.1 Appropriate use of markup langage	/2
4.4 Suitable creation of Web forms	/3

Element of competency: Program the client-side application logic (00SU.6).	
Performance criteria	**weight**
6.3 Proper programming of interactions between the Web interface and the user	/5
6.5 Web forms in compliance with usability requirements	/5

Part 5: Entity display, edition and deletion page (20%)

Element of competency: Program the Web interface (00SU.4).	
Performance criteria	**weight**
4.1 Appropriate use of markup langage	/5
4.4 Suitable creation of Web forms	/5
Element of competency: Program the client-side application logic (00SU.6).	
Performance criteria	**weight**
6.3 Proper programming of interactions between the Web interface and the user	/5
6.5 Web forms in compliance with usability requirements	/5

Part 6: Testing your application (15%)

Element of competency: Analyze the application development project (00SU.1).	
Performance criteria	**weight**
1.1 Accurate analysis of design documents	/5
Element of competency: Control the quality of the application (00SU.7).	
Performance criteria	**weight**
7.2 Thorough reviews of code and security	/5
7.5 Compliance with design documents	/5

Submission (5%)

Element of competency: Control the quality of the application (00SU.7).	
Performance criteria	**weight**
7.5 Compliance with design documents	/5

Sub-total on 100 (before errors)	/100
Penalty for language errors (0.5 point each /maximum 10%)	/100
Evaluation total on 100	/100
Evaluation total on 40	/40

CORRECTION GRID FOR LANGUAGE

Clear Communication	Clear Communication, **most of the time**	Vague Communication	Unclear Communication
- 0	- 0,5	- 1,5	- 2
(Word Choice) Use of precise and rich vocabulary	(Word Choice) Use of precise vocabulary	(Word Choice) Use of imprecise vocabulary	(Word Choice) Use of inappropriate vocabulary
- 0	- 0,5	- 1,5	- 2
(Format/Type of work) Respect of norms	(Format/Type of work) Respect of **most of the** norms	(Format/Type of work) Non-respect of the norms	(Format/Type of work) Inappropriate in relation to the required norms
- 0	- 0,5	- 1,5	- 2
(Linguistic Code) (≤2 mistakes / page)	(Linguistic Code) (3-7 mistakes/page)	(Linguistic Code) (8-10 mistakes/ page)	(Linguistic Code) (>10 mistakes/ page)
- 0	- 0,5 - 2.5	- 2.5 - 3.5	- 4