



Final Evaluation – Part 1: 25%

Course Identification

Name of program – Code:	COMPUTER SCIENCE TECHNOLOGY – PROGRAMMING (420.BP)
Course title:	WEB SERVER APPLICATIONS DEVELOPMENT I
Course number:	420-DW3-AS
Group:	07392
Teacher's name:	Jean-François Parent
Duration:	Extended
Semester:	Fall 2020

Student Identification

Name: _____ Student number: _____

Date: _____ 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 Competencies

Statement of the evaluated competency – Code

Develop transactional Web applications – 00SU

Evaluated elements of the competency 00SU

1. *Analyze the application development project*
 2. *Prepare the computer development environment*
 3. *Prepare the database*
 5. *Program the server-side application logic*
 7. *Control the quality of the application*
 9. *Produce the documentation*
-

Instructions

- Students will submit a NetBeans project and a cheat sheet individually.
 - It is the teacher's responsibility to identify language errors. If such errors are found, teacher may deduct up to 5% of the final 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).
 - Deadlines are shared on Omnivox in the assignment box and must be respected.
-

- Please see the detailed rubric in the document for the breakdown of the mark for the individual work.
-

TOTAL: 100 POINTS

Project 2

Guidelines

Note: This is **NOT** a group project

You have to do this project by implementing all the following working methods:

1. At the very top of **all the stored procedures** and **views** you submit you must write your **name** and **student number**. Also write the **date** of the file creation. You must also write **comment(s)** to describe what occurred at the written date(s). Use comments similar to the following ones (you may improve them, for example, by adding a VERSION column, like v1.0):

```
#Revision history:
#DEVELOPER          DATE          COMMENTS
#Bill Torvalds (1244556) 2020-03-14  Created the customers_insert
#                               stored procedure.
#
#Bill Torvalds (1244556) 2020-03-16  GRANT permission to webuser.
```

2. Your database and all the fields it contains must use **utf8mb4_general_ci** collation setting.
3. Give **clear names** to your database, **tables**, **fields**, **views** and **stored procedures**.
4. Create a **primary key** for all tables. Use a new **UUID** (Universally Unique Identifier) as the default value for all the primary keys.
5. Create **UNIQUE indexes** for all fields who require it.
6. Create **indexes** for all fields used for **sorting** or **filtering**.
7. **Prevent NULL values** when a field is mandatory.
8. All the SQL code must be placed in **stored procedures**.
9. Use **indentation** in the stored procedures. Your SQL code must never be written entirely in one single line. Press ENTER before SQL keywords.
10. Use **comments** in every table, view and stored procedure to include the **revision history** and to **explain** briefly what the SQL code is doing. Use only # or - - (you can use /* */ but only for debugging).
11. Create a **SQL user**. Use the word "user-" followed by your **student number** as the account name (for example **user-1244556**). **Give permissions** to this user so he/she will be able to call all the **stored procedures** and **views**. Don't give any other access to this user.

Description of the project

For this project, you have to build a database that will be used to improve the website you created in project 1. The database name should be

database-xxxxxxx (replace **xxxxxxx** with your **student number**).

Tables

You must create 3 tables. Read carefully the requirements to determine which fields are required. For all these fields, give a relevant **name**, select the correct **data type**, and an appropriate **field length** large enough to store all the required data, but not too big to waste space on disk (see ColumnStore Data Types in MariaDB doc).

Every table must have 2 extra fields to store the **date and time** of the object **creation** and **modification**. It is not acceptable for these fields to contain no value, so the default value for both of them should be the current date and time..

A)customers

We need to keep track of the customers who buy products on the website. We thus need the firstname and the lastname (20 characters maximum for both), the address, the city and the province (maximum 25 chars for all these fields, including the city (8 characters was not big enough)) and the postal code (7 chars max). It should also contain 2 fields to store the username (maximum 12 characters, and it is forbidden for 2 different customers to use the same username) and the password (maximum 255 characters) required for login.

B)products

This table is used to store all the products sold on your website. Each product is identified by a product code of maximum 12 characters. We need to store the description of the product (maximum 100 characters). We also need to store the price of the product (number with 2 decimals only, never save more than 2 decimals in the database. Maximum amounts should be 10000\$). We should also have a cost price (i.e. the price the company paid for it before selling it, use 2 decimals), but this field is optional.

C)purchases

This is used to store the purchases. It should contain foreign keys pointing to the primary keys of customers and products tables. The table should also contain the quantity sold (max 999) and the price of the product at the moment of the sale (so the past sales will not be affected if we change the price of the products in the database). All these fields are mandatory. Add another field for the comments (maximum 200 characters) which is optional.

Views

Create a view to show all the fields from the products table. Sort by product code.

Stored procedures

You must create stored procedures to **SELECT**, **INSERT**, **UPDATE** and **DELETE** data in all the created tables (3 tables * 4 stored procedures = 12 stored procedures). The **SELECT** stored procedure should select all the data and **sort** it. The **UPDATE** and **DELETE** stored procedure needs to have a primary key passed as a parameter. Figure out which parameters are needed for the **INSERT** procedure.

In addition to the basic stored procedures, you must also create the following 2 stored procedures:

1. Get the password for a given username

This stored procedure should accept a **username** and return its corresponding **password** (which will be encrypted later) if it exists.

2. Filter the purchases

This stored procedure should accept an **optional** **DATETIME** parameter. If a date is passed as the parameter, you should return all the purchases made **on that date or later**. If **no parameter is specified**, or if **NULL** is passed as a parameter, it should return all the purchases for a specific customer. The fields should be sorted from the most recent to the oldest purchases.

In both cases (parameter passed or not) **sort** the returned data by **create DATETIME** from the most recent purchase to the oldest one.

Files to submit

Submit your **database backup** in a file called **database-xxxxxxx.sql** (replace **xxxxxxx** with your **student number**, for example database-1244556.sql).

This SQL file should be able to create the **database**, all the **tables**, all the **data** they contain (**INSERT INTO ...**), and all the **views** and **stored procedures**.

Submit another SQL file (or include it in the previous file) to **create the SQL user** required to run your website, and **all its permissions** (**GRANT**) for the stored procedures. The filename should be called **permissions-xxxxxxx.sql** (replace **xxxxxxx** with your **student number**, for example permissions-1244556.sql). Warning, HeidiSQL GUI cannot export the objects permissions, so if you want to get points for the user and permissions, use the **SHOW GRANTS** command.

Test your backup files before submitting them. When you are satisfied with your tests, **submit the 2 .sql files** (or 1 file if you merged them together) on Omnivox.

CORRECTION GRID FOR REQUIREMENTS

Competency : Deploy transactional Web applications – 00SU	
Elements of competencies: Analyze the application development project (00SU.1)	
Performance criteria	weight
Accurate analysis of design documents (00SU.1.1)	/3.5
Proper identification of the tasks to be carried out (00SU.1.2)	/4
Elements of competencies: Prepare the computer development environment (00SU.2)	
Performance criteria	weight
Proper installation of the Web development platform and the development database management system (00SU.2.1)	/2
Proper installation of software and libraries (00SU.2.2)	/2.5
Appropriate configuration of the version control system (00SU.2.3)	/2
Proper importing of the source code (00SU.2.4)	/3
Elements of competencies: Prepare the database (00SU.3)	
Performance criteria	weight
Suitable creation or adaptation of the database (00SU.3.1)	/6
Proper insertion of initial or test data (00SU.3.2)	/4.5
Compliance with the data model (00SU.3.3)	/4.5
Elements of competencies: Program the server-side application logic (00SU.5)	
Performance criteria	weight
Proper programming or integration of authentication and authorization mechanisms (00SU.5.1)	/3
Appropriate choice of clauses, operators, commands or parameters in database queries (00SU.5.3)	/5
Correct handling of database data (00SU.5.4)	/5
Proper application of internationalization techniques (00SU.5.6)	/1
Precise application of secure programming techniques (00SU.5.7)	/5
Elements of competencies: Control the quality of the application (00SU.7)	
Performance criteria	weight
Precise application of test plans (00SU.7.1)	/5.5
Thorough reviews of code and security (00SU.7.2)	/6
Relevance of the corrective actions (00SU.7.3)	/1.5
Compliance with issue tracking and version control procedures (00SU.7.4)	/2.5
Compliance with design documents (00SU.7.5)	/5
Elements of competencies: Participate in the deployment of the application on the Web host (00SU.8)	
Performance criteria	weight
Appropriate configuration of the application on the Web host (00SU.8.2)	/4
Proper application of the procedure for migrating the service onto the Web host (00SU.8.3)	/5
Precise application of security measures (00SU.8.4)	/4.5
Compliance with search engine indexing requirements (00SU.8.5)	/5
Elements of competencies: Produce the documentation (00SU.9)	
Performance criteria	weight
Proper identification of the information to be written up (00SU.9.1)	/2.5
Clear record of the work carried out (00SU.9.2)	/7.5

CORRECTION GRID FOR LANGUAGE

Clear Communication	Clear Comm., 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)	(Linguistic Code)	(Linguistic Code)	(Linguistic Code)
(≤2 mistakes / page)	(3-7 mistakes/page)	(8-10 mistakes/ page)	(>10 mistakes/page)
- 0	- 0,5 - 2.5	- 2.5 - 3.5	- 4