Vanier College **Computer Science Technology**
Faculty of Science & Technology **420-411-VA eCommerce**

Term Project (LIA)

We use projects to optimize authentic learning in College courses. These projects allow the student to apply/synthesize most or all course competencies and to see the results of what they learned.

To promote your employability as soon-graduating students it is highly suggested that you preserve these tangible results in a portfolio of your accomplishments to show prospective employers next semester. Toward this goal, the Term Project for eCommerce will be developped completely on your personal GitHub account (if you don't already have one, get a free account to complete the work in this project/course). Note that throughout all phases of planning and implementation of your project, proper use of a GitHub repository and associated tools will be evaluated.

Throughout all phases of this project, as per the Vanier College SPLI policy, the expression in the language of instruction (English) will be evaluated for a weight of at least 10% of each component which uses English as the language of communication. This includes most Term Project deliverables, but excludes the implementation.

Note that the Term Project is designed to allow students to demonstrate their achievement of the course key learning outcomes within the achievement context defined for the competency(ies) it covers.

Term Project Description (Learning Integration Assessment)

In teams of 2 or 3, students develop, implement, and deploy a transactional Web application integrating the following aspects:

- User registration, authentication, and authorization, including two-factor authentication.
- Prevention of common security risks such as SQL injection and cross-site scripting (XSS).
- Internationalization and localization.
- Transactions with a database dedicated to this Web application.
- Reading data from an external Web service.
- Developed using a Behaviour-Driven Development methodology, complete with a test suite automated with an appropriate framework, such as Codeception.

The Web application scope should be representative of a typical small Web application found online that can be programmed by a team of 2 people. In this project, each student implements 8 user requirements, expressed as gherkin feature files written by the students themselves. The implementation will be executed using XHTML/CSS/JavaScript for the front-end technologies and PHP/MySQL (MariaDB) for the back-end technologies. The programming will follow the MVC programming pattern as it will be presented in class.

Deliverables:

- Deliverable 1: Project proposal
- Deliverable 2: Feature suite: Feature behaviours written using a language such as gherkin for consumption in automated testing frameworks and database ERD
- Deliverable 3a: 50% progress checkpoint
- Deliverable 3b: Completed implementation
- Deliverable 4: In-class presentation and demonstration
- Deliverable 5: Final report including updated features and a user guide

Grading Scheme

A total of 30% of your final grade will be awarded for the Term Project. This 30% will be distributed according to the following detail:

Description	Grade Value	Completion Date

Deliverable 1: Project proposal	2%	Week 3
Deliverable 2: Feature suite and ERD	3%	Week 5
Deliverable 3a: 50% progress checkpoint	5%	Week 9
Deliverable 3b: Completed Implementation	15%	Week 14
Deliverable 4: Presentation and demo	2%	Week 14
Deliverable 5: User guide	3%	Week 15

Notes:

1. The Term Project will be executed in teams of 2 or 3 students.
2. **In order for your Term Project grade to count toward your final grade for the course, you must first achieve an average of at least 60% on the Midterm and Final Exams.**
3. Late work (any work counting toward grades) is not accepted unless prior arrangement with the teacher. In this case, the penalty for submitting late work is 10% per day (all 7 days of the week count toward this penalty).

Deliverables

In this section we explain each deliverable and show the evaluation rubrics for these.

Project Proposal (2%)

A project proposal is a written document outlining everything stakeholders should know about a project, including the timeline, budget, objectives, and goals. Your project proposal should summarize your project details and sell your idea, so stakeholders buy in to the initiative.

For this specific project the stakeholders are your teacher and your team and the timeline is outlined below in the grading scheme. You have to define your budgeted work hours, as well as objectives and goals.

In teams of 2 or 3 you will write a Markdown document stored in your GitHub repository with the following contents:

- Name of your team
- Name of the team members
- A description of the clientele, the general goal of the project, and a list of all the features that will be supported by your final Web application. For each student in the team, there must be a total of **8** new features written. A new feature excludes the features that were completely coded in class with your teacher, e.g. login, logout, registration, 2-factor authentication.
- An estimate of the number of hours you will spend implementing this Web application.

See the following Markdown markup language guide: https://www.markdownguide.org/getting-started/

An example proposal:

MarketPlace

We will build a marketplace Web application where registered sellers can list and sell products and buyers can purchase these products online.

Our project will support the following stories:

1. As a seller, I can register, login, and logout (these are not new features - they are given in class).
2. As a seller, I can add and modify products for sale and track sales of these products (3 features).
3. As a seller, I can create and modify my store profile (1 feature).
4. As a seller, I can view product purchases and mark them as shipped while adding tracking information (2 features).
5. As a seller, I can view client service requests on sales and respond (2 features).
6. As a user, I can search the product catalog (1 feature).

7. As a user, I can see product details (1 feature).
8. As a user, I can add/delete/modify quantities for products to my shopping cart (3 features).
9. As a user, checkout my order (1 feature).
10. .... complete this, bringing the total to **8** features per team member.

We will model our solution on the Amazon online buying tools and online store experiences. We estimate that we will spend a total of 120 hours building this product, i.e., 60 hours per team member.

The Project Proposal will be evaluated according to the following rubric:

Criterion	**Good**	**OK**	**Bad**
Appropriate use of version control (30%)	The document was stored and revised on the repository and written using the Markdown markup language.	The document was stored on the repository and written using the Markdown markup language.	The document was not stored and revised on the repository.
Relevance of the proposal to the context of the Web (30%)	The proposal uses existing Web application feature types and viable revenue streams.	The proposal uses mostly existing Web application feature types and revenue streams.	The proposal uses some existing Web application feature types or revenue streams.
Feasibility of the proposal to the context of the Web (30%)	The features rely on available data storage and passing techniques and locally tractable algorithms.	The features mostly rely on available data storage and passing techniques and tractable algorithms.	The features don't rely on available data storage and passing techniques and tractable algorithms.
(SPLI) Quality of expression in English (10%)	Error-free, clear expression with understandable text.	Mostly error-free, mostly clear expression with mostly understandable text.	Many errors, or unclear expression.

Feature Suite and ERD (3%)

A Feature Suite consists of the set of business-domain description for all planned features to implement. When written correctly using a standardized language such as gherkin, such business-domain descriptions can be consumed by automated testing frameworks such as Codeception to produce automated tests as part of a test-driven development methodology.

For this deliverable, you will write a set of **8** new features per student in your team, written correctly using the gherkin language. These will be consumable by the Codeception automated testing framework. Refer to https://codeception.com/docs/07-BDD for a quick introduction to gherkin. In a nutshell consider the following:

```
Feature: checkout process
  In order to buy products
  As a customer
  I want to be able to buy several products

  Scenario:
    Given I have product with $600 price in my cart
    And I have product with $1000 price
    When I go to checkout process
    Then I should see that total number of products is 2
    And my order amount is $1600
```

Above, the feature described is the "checkout process". In the 3 lines below, it is documented and explained as "`In order to buy products, As a customer, I want to be able to buy several products`". Next comes a scenario which is a description of an acceptance test. The acceptance test describes an exact test that will be run each time the automated testing framework will be asked to run tests. It is concievable that a single scenario could be enough to test a feature, however, in many cases, multiple scenarios will be required. The scenrios are split in 3 parts using the Given-When-Then approach: Given..And.. are initial conditions, When..And.. are actions that change the initial conditions, and Then..And.. are expected results.

We will address the topic in more depth in class.

Moreover, provide a database ERD that describes the database that supports the planned features.

This deliverable will be evaluated according to the following rubric:

Criterion	**Good**	**OK**	**Bad**
Appropriate use of version control (15%)	The documents were stored and revised on the repository in an appropriate folder.	The documents were stored on the repository in an appropriate folder.	The documents were not stored and revised on the repository.
Compatibility of the feature with the acceptance tests (25%)	The provided scenarios are compatible with the functionality of all features.	The provided scenarios are mostly compatible with the functionality of all features.	The provided scenarios are not compatible with the functionality of all features.
Completeness of the acceptance test coverage for the described features (25%)	The provided scenarios cover the positive and negative cases to test all functionality of all features.	The provided scenarios mostly cover the positive and negative cases to test all functionality of all features.	The provided scenarios do not correctly cover the positive and negative cases to test all functionality of all features.
Feature support by the database described by the ERD (25%)	The database described in the ERD has the correct entities with the correct relations to store the data in support for the planned features.	The database described in the ERD mostly has the correct entities with the correct relations to store the data in support for the planned features.	The database described in the ERD does not have the correct entities with the correct relations to store the data in support for the planned features.
(SPLI) Quality of expression in English (10%)	Error-free, clear expression with understandable text.	Mostly error-free, mostly clear expression with mostly understandable text.	Many errors, or unclear expression.

50% Progress Checkpoint (5%)

For the 50% Progress Checkpoint, you will demonstrate to the teacher that 50% of the planned features have been implemented and tested in accordance with the delivered feature suite and contained scenarios. Any deviation from this plan needs to be justified to the satisfaction of the teacher.

Completed Implementation (15%)

For the completed Implementation, you will provide to the teacher an implementation where 100% of the planned features have been implemented and tested in accordance with the delivered feature suite and contained scenarios. Any deviation from this plan needs to be justified to the satisfaction of the teacher.

These implementation deliverables will be evaluated according to the following rubric, where the grade will be prorated with the number of user stories implemented on the expected number of user stories (8):

Criterion	**Good**	**OK**	**Bad**
Appropriate use of version control (10%)	The commit history shows at least one commit per feature and per method. Developers do not modify multiple methods in commits.	The commit history mostly shows at least one commit per feature and per method. Developers do not modify multiple methods in commits.	The commit history dows not show at least one commit per feature and per method. Developers do not modify multiple methods in commits.
Respect of the selected design pattern (15%)	Correct separation of concerns as per MVC design pattern guidelines.	Mostly correct separation of concerns as per MVC design pattern guidelines.	Incorrect separation of concerns as per MVC design pattern guidelines.
Correct selection and implementation of algorithms (15%)	Optimal use of SQL and PHP to process and display data.	Correct use of SQL and PHP to process and display data.	Incorrect use of SQL and PHP to process and display data, e.g., loops used to filter data instead of SQL.
Correct validation of data inputs (10%)	All data inputs are correctly validated to force correct data into the system. Validation on client and server sides.	Most data inputs are correctly validated to force correct data into the system. Validation on client and server sides.	Data inputs are not correctly validated to force correct data into the system. Validation on client and server sides.
Appropriate corrective measures to ensure platform security (10%)	Common security risks such as SQL injection and cross-site scripting are prevented.	Common security risks such as SQL injection and cross-site scripting are mostly prevented.	Common security risks such as SQL injection and cross-site scripting are not prevented.
Reasonable consideration of User Experience (10%)	Completed internationalization, intuitive user interaction with few clicks.	Mostly completed internationalization, mostly intuitive user interaction with few clicks.	Incomplete internationalization or non-intuitive user interaction.
Appropriate design of User Interface (10%)	Ergonomic user interfaces with easy-to-find controls	Mostly ergonomic user interfaces with easy-to-find	User interfaces that lack in ergonomics and with links and controls that are not

	and links that is easy on the eyes.	controls and links that is easy on the eyes.	easy to find or with a design that is not easy on the eyes.
Appropriate automated test suite (10%)	Correct use of the testing framework to import the feature suite, produce automated tests, and run them with successful results.	Mostly correct use of the testing framework to import the feature suite, produce automated tests, and run them with successful results.	Incorrect use of the testing framework to import the feature suite, produce automated tests, and run them with successful results.
Absence of execution and logical errors (10%)	No execution or logical errors.	Almost no execution or logical error.	Clear presence of execution and logical errors.

Presentation and Demonstration (2%)

In front of all your classmates, you will deliver a presentation to introduce and demonstrate your completed Web application to all students and teachers in attendance. The presentation will last at most 5 minutes per student on your team and during this presentation you will address the following topics:

A. Explain what need your application addresses and why this needs to be addressed.
B. Show us a demonstration of all user stories. DO NOT demonstrate all checks, validations, and safeties.
C. Emphasize the most interesting feature.
D. Explain the hardest feature to implement and what made it difficult to implement.

This deliverable will be evaluated according to the following rubric:

Criterion	**Good**	**OK**	**Bad**
Project objectives, need addressed, and target user-base. (30%)	Clear communication of topic.	Needs some clarification.	Unclear communication of topic.
Difficulties met in the project and solutions (30%)	Clear communication of topic.	Needs some clarification.	Unclear communication of topic.
Lessons learned from interesting and hard features to implement (30%)	Clear communication of topic.	Needs some clarification.	Unclear communication of topic.
(SPLI) Quality of expression in English (10%)	Error-free, clear expression with understandable presentation.	Mostly error-free, mostly clear expression with mostly understandable presentation.	Many errors, or unclear expression.

User Guide (3%)

Write a user guide in your GitHub repository Wiki. A proper user guide does the following for all your application features:

1- Present the feature.
2- Explain the goal of the feature.
3- Explain the use of the feature.

Guidelines:

- Make sure the instruction terminology uses the same words used on buttons and links in the Web application.
- Present instructions as step-by-step procedures with visual stepping stones (e.g. Step 1, Step 2 etc.) complete with screen captures to decompose COMPLEX instructions.
- Provide step-by-step sequences in the correct order and write these while doing the actual task.
- Make instructions short.
- Use everyday words and terms: no jargon.
- Explain what a function or feature is for (in basic practical terms) as well as "How to" instructions.
- Explain symbols, icons and codes early.
- Do not assume the user has prior experience or knowledge.
- Write in the present tense and the active voice.
- Write the user manual in synch with the product's development timeline — not under pressure of shipping deadlines.
- Test the instructions yourself, pretending you don't know anything while doing it.

This deliverable will be evaluated according to the following rubric:

Criterion	**Good**	**OK**	**Bad**
Appropriate use of version control (30%)	The document was stored and revised on the repository and written using the Markdown markup language.	The document was stored on the repository and written using the Markdown markup language.	The document was not stored and revised on the repository.
Correct presentation of work completed (30%)	All features of the Web application are showcased and introduced.	Needs some clarification.	Unclear communication of topic.
Correct demonstration of product usability (30%)	The use of all features of the Web application is explained.	Needs some clarification.	Unclear communication of topic.
(SPLI) Quality of expression in English (10%)	Error-free, clear expression with understandable documentation.	Mostly error-free, mostly clear expression with mostly understandable documentation.	Many errors, or unclear documentation.