ALN2 — ALN2 TASK 1: WEB-BASED SPRINT INVENTORY APPLICATION

JAVA FRAMEWORKS — D287
PRFA — ALN2

COMPETENCIES

4084.1.1 : Implements User Interfaces
The learner implements user interfaces.

4084.1.2 : Implements Frameworks
The learner implements object-oriented programming frameworks.

INTRODUCTION

Throughout your career in software design and development, you will be asked to create, customize, and maintain applications with various features and functionality based on business requirements. For this assessment, you will customize a Spring Framework application with an HTML front-end and a Java backend using the solution statements provided in the requirements section of this assessment.

The skills you showcase in your completed application will be useful in responding to technical interview questions for future employment. This application may also be added to your portfolio to show to future employers.

For this project, you will use the Integrated Development Environment (IDE) link in the web links section of this assessment to install the IDE, IntelliJ IDEA (Ultimate Edition). You will sign up for a free student license using your WGU.edu email address. Please see the "IntelliJ Ultimate Edition Instructions" attachment for instructions on how to do this. You will also use the GitLab link in the web links section to obtain the template code for this project, please review the "GitLab How-To" web link prior to starting your work. Next, you will clone your GitLab repository and open it in IntelliJ IDEA (Ultimate Edition).

SCENARIO

You are working for a company that licenses and customizes a software application to keep track of inventory in stores. Your job as a software developer is to customize this application to meet a specific customer's needs. You will choose any type of customer you would like, but it must sell a product composed of parts. An example of products versus parts would be a customer that's a bicycle shop: a bicycle is a product, and a set of two matching wheels is a part (do not use the bicycle shop example in your project).

You have been provided with a Spring application with a Java backend, a generic HTML user interface to use in the design and development of the system, and a UML class diagram to assist you in your work (see the attached "UML Class Diagram"). You can find a user guide to help assist with the inventory management application in the attachments (see "Shop Inventory Management User Guide").

REQUIREMENTS

Your submission must be your own original work. You may not use or reference other students' submissions for this task. For more information please review our Academic Authenticity policies and the Professionalism and Conduct Expectations for College of Information Technology Students.

You must use the rubric to direct the creation of your submission because it provides detailed criteria that will be used to evaluate your work. Each requirement below may be evaluated by more than one rubric aspect. The rubric aspect titles may contain hyperlinks to relevant portions of the course.

*Tasks may **not** be submitted as cloud links, such as links to Google Docs, Google Slides, OneDrive, etc., unless specified in the task requirements. All other submissions must be file types that are uploaded and submitted as attachments (e.g., .docx, .pdf, .ppt).*

Note: External plugins and libraries other than those specified in this task are not allowed.

Note: You cannot use a bicycle shop as a customer for your submission.

A. Create your subgroup and project by logging into GitLab using the web link provided and using the "GitLab How-To" web link, and do the following:
   - Clone the project to the IDE.
   - Commit with a message and push when you complete *each* of the tasks listed below (e.g., parts C to J).

     Note: You may commit and push whenever you want to back up your changes, even if a task is not complete.

   - Submit a copy of the Git repository URL and a copy of the repository branch history retrieved from your repository, which must include the commit messages and dates.

     Note: Wait until you have completed all the following prompts before you create your copy of the repository branch history.

B. Create a README file that includes notes describing where in the code to find the changes you made for *each* of parts C to J. *Each* note should include the prompt, file name, line number, and change.

C. Customize the HTML user interface for your customer's application. The user interface should include the shop name, the product names, and the names of the parts.

   Note: Do not remove any elements that were included in the screen. You may add any additional elements you would like or any images, colors, and styles, although it is not required.

D. Add an "About" page to the application to describe your chosen customer's company to web viewers and include navigation to and from the "About" page and the main screen.

E. Add a sample inventory appropriate for your chosen store to the application. You should have **five** parts and **five** products in your sample inventory and should not overwrite existing data in the database.

Note: Make sure the sample inventory is added only when both the part and product lists are empty. When adding the sample inventory appropriate for the store, the inventory is stored in a set so duplicate items cannot be added to your products. When duplicate items are added, make a "multi-pack" part.

F. Add a "Buy Now" button to your product list. Your "Buy Now" button must meet *each* of the following parameters:

- The "Buy Now" button must be next to the buttons that update and delete products.
- The button should decrement the inventory of that product by **one**. It should not affect the inventory of any of the associated parts.
- Display a message that indicates the success or failure of a purchase.

G. Modify the parts to track maximum and minimum inventory by doing the following:

- Add additional fields to the part entity for maximum and minimum inventory.
- Modify the sample inventory to include the maximum and minimum fields.
- Add to the InhousePartForm and OutsourcedPartForm forms additional text inputs for the inventory so the user can set the maximum and minimum values.
- Rename the file the persistent storage is saved to.
- Modify the code to enforce that the inventory is between or at the minimum and maximum value.

H. Add validation for between or at the maximum and minimum fields. The validation must include the following:

- Display error messages for low inventory when adding and updating parts if the inventory is less than the minimum number of parts.
- Display error messages for low inventory when adding and updating products lowers the part inventory below the minimum.
- Display error messages when adding and updating parts if the inventory is greater than the maximum.

I. Add at least **two** unit tests for the maximum and minimum fields to the PartTest class in the test package.

J. Remove the class files for *any* unused validators in order to clean your code.

K. Demonstrate professional communication in the content and presentation of your submission.

File Restrictions

File name may contain only letters, numbers, spaces, and these symbols: ! - _ . * ' ( )
File size limit: 200 MB
File types allowed: doc, docx, rtf, xls, xlsx, ppt, pptx, odt, pdf, csv, txt, qt, mov, mpg, avi, mp3, wav, mp4, wma, flv, asf, mpeg, wmv, m4v, svg, tif, tiff, jpeg, jpg, gif, png, zip, rar, tar, 7z

RUBRIC

A:GIT REPOSITORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A GitLab repository is not provided.	The subgroup and project are created in Gitlab, but the submission does not include *all* of the given requirements, or one or more of the given requirements contain errors.	The subgroup and project are created in GitLab correctly and the submission includes *each* of the given requirements. *Each* requirement is error-free.

B:README FILE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
A README file is not provided.	A README file is created and includes notes describing where in the code to find the changes made for *each* of parts C to J, but does not include the prompt, file name, line number, or change. The submission does not include *all* the changes made or does not accurately describe the code.	A README file is created and includes notes accurately describing where in the code to find the changes made for *each* of the parts C to J. The notes include the prompt, file name, line number, and change.

C:CUSTOMIZE USER INTERFACE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The HTML user interface is not customized for the customer's application.	The HTML user interface is customized for the customer's application, but the user interface is missing the shop name, the product names, or the names of the parts. Or the user interface is not customized correctly for the customer, or the user interface does not function correctly.	The HTML user interface is correctly customized for the customer's application. The user interface includes the shop name, the product names, and the names of the parts.

D:ABOUT PAGE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The "About" page is not added to the application.	The "About" page is added to the application, but the page does	The "About" page added to the application accurately describes the chosen customer's company

	not accurately describe the chosen customer's company to web viewers, or the application does not include navigation to and from the "About" page to the main screen. Or the navigation does not function properly.	to web viewers. The application includes navigation to and from the "About" page to the main screen and functions properly.

E:SAMPLE INVENTORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The sample inventory is not added to the application.	The sample inventory is added to the application, but the inventory is not appropriate for the chosen store, or the inventory includes four or fewer products or four or fewer parts. Or the sample inventory overwrites the existing data in the database.	The sample inventory is added to the application. The inventory is appropriate for the chosen store and includes five products and five parts. The inventory does not overwrite existing data in the database.

F:BUY NOW BUTTON

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The "Buy Now" button is not added to the product list.	The "Buy Now" button is added to the product list, but the button does not meet *each* of the given parameters. Or the button is not functional.	The "Buy Now" button is added to the product list. The button meets *each* of the given parameters, and the button is fully functional.

G:MAX AND MIN INVENTORY

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not modify the parts to track minimum and maximum inventory.	The submission modifies the parts to track the maximum and minimum inventory but does not include *each* of the given parameters. Or the modifications are not functional or contain inaccuracies.	The submission accurately modifies the parts to track maximum and minimum inventory. The modification of the parts includes *each* of the given parameters, and the modifications are fully functional.

H:VALIDATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
Validation is not added for between or at the maximum and minimum fields.	Validation is added for between or at the maximum and minimum fields, but the validation does not include *each* of the given parameters. Or the error messages are not appropriate, or the validation is not functional.	Validation is added for between or at the maximum and minimum fields. The validation includes *each* of the given parameters and the appropriate error messages. The validation is fully functional.

I:UNIT TESTS

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
At least two unit tests are not added for the maximum and minimum fields.	At least two unit tests are added for the maximum and minimum fields, but one or *both* of the unit tests are not added to the PartTest class. Or one or *both* of the unit tests are not functional.	At least two functional unit tests are added for the maximum and minimum fields. *Both* unit tests are added to the PartTest class in the test package.

J:CLEAN CODE

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
The submission does not remove the class files for *any* unused validators.	Not applicable.	The submission removes the class files for *any* unused validators to clean the code.

K:PROFESSIONAL COMMUNICATION

NOT EVIDENT	APPROACHING COMPETENCE	COMPETENT
This submission includes professional communication errors related to spelling, grammar, punctuation, and sentence fluency. For best results, please focus on the specific Correctness errors identified by Grammarly for Education to help guide your revisions. If you need additional	This submission includes professional communication errors related to spelling, grammar, punctuation, and/or sentence fluency. For best results, please focus on the specific Correctness errors identified by Grammarly	This submission demonstrates correct use of spelling, grammar, punctuation, and sentence fluency. You have demonstrated quality professional communication skills in this submission.

assistance preparing your submission, please contact your Instructor.

for Education to help guide your revisions.

WEB LINKS

Integrated Development Environment – IntelliJ IDEA (Ultimate Edition)
Sign up for free student account using WGU.edu email address.

GitLab
Access Inventory Management Application Template code in this environment

GitLab How-To
GitLab Instructions Knowledge Base Article

SUPPORTING DOCUMENTS

Shop Inventory Management User Guide.pdf

UML Diagram.pdf

IntelliJ Ultimate Edition Directions.pdf