**D424 – Software Engineering**

**Task 3**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Inventory Management System |
| **Student Name:** | Brady Boyd |

Table of Contents

[Application Design and Testing 4](#_Toc137395671)

[Class Design 4](#_Toc137395672)

[UI Design 5](#_Toc137395673)

[Unit Test Plan 5](#_Toc137395674)

[Introduction 5](#_Toc137395675)

[Purpose 5](#_Toc137395676)

[Overview 6](#_Toc137395677)

[Test Plan 6](#_Toc137395678)

[Items 6](#_Toc137395679)

[Features 6](#_Toc137395680)

[Deliverables 6](#_Toc137395681)

[Tasks 6](#_Toc137395682)

[Needs 7](#_Toc137395683)

[Pass/Fail Criteria 7](#_Toc137395684)

[Specifications 7](#_Toc137395685)

[Procedures 7](#_Toc137395686)

[Results 8](#_Toc137395687)

[Application Link 9](#_Toc137395688)

[GitLab Link 9](#_Toc137395689)

[Maintenance Guide 9](#_Toc137395690)

[Introduction 9](#_Toc137395691)

[Requirements 9](#_Toc137395692)

[Project Setup 9](#_Toc137395693)

[Running Tests 10](#_Toc137395694)

[Updating the Application 10](#_Toc137395695)

[User Guide 11](#_Toc137395696)

[Introduction 11](#_Toc137395697)

[System Requirements 11](#_Toc137395698)

[Logging In 12](#_Toc137395699)

[Parts 12](#_Toc137395700)

[*Add a New Part* 12](#_Toc137395701)

[*Update an Existing Part* 13](#_Toc137395702)

[*Delete a Part* 14](#_Toc137395703)

[Products 15](#_Toc137395704)

[*Add a New Product* 15](#_Toc137395705)

[*Attach Parts to Products* 16](#_Toc137395706)

[*Remove Attached Parts to Products* 17](#_Toc137395707)

[*Update an Existing Product* 17](#_Toc137395708)

[*Sell a Product* 18](#_Toc137395709)

[*Delete a Part* 18](#_Toc137395710)

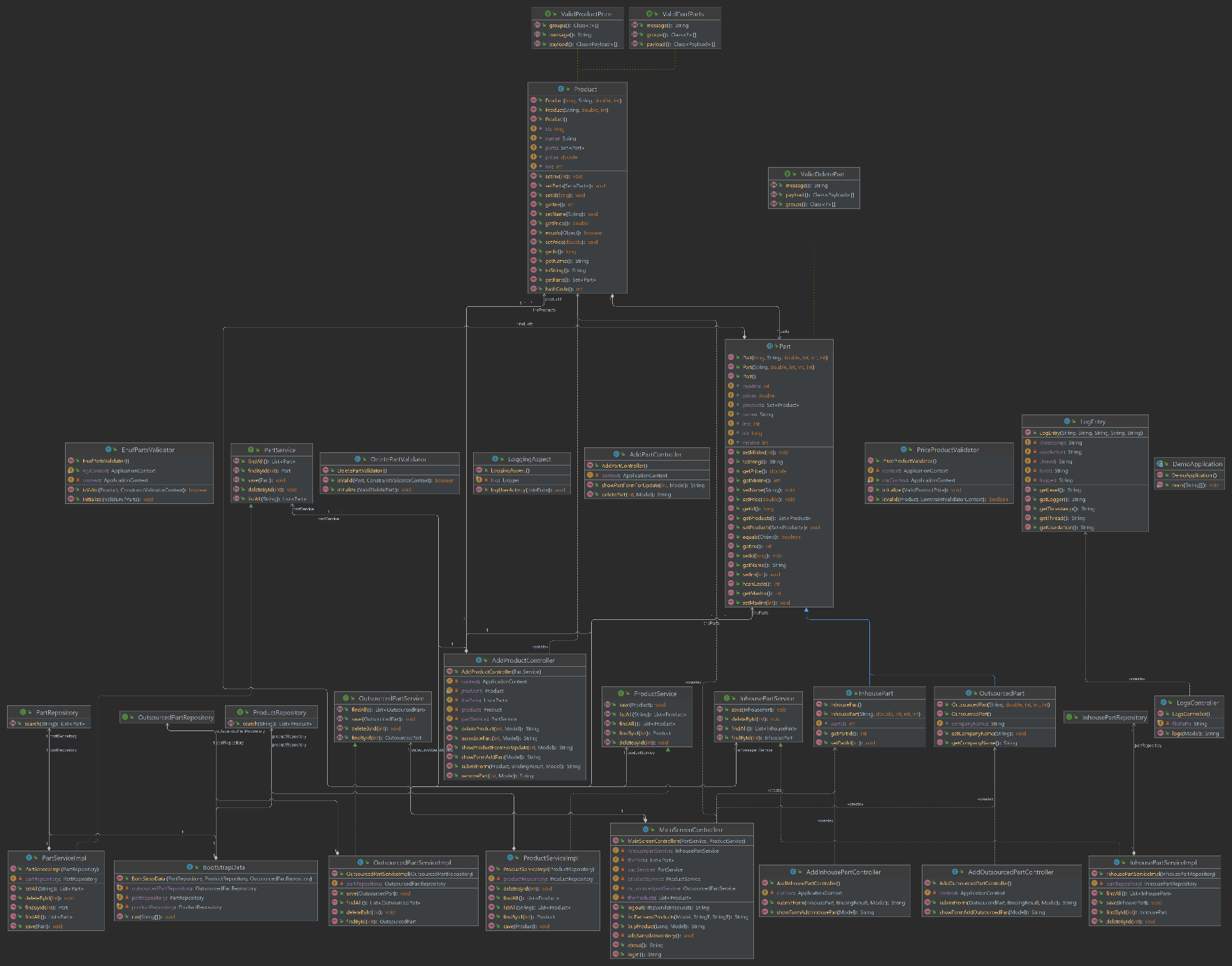
[Generate Log/Report 19](#_Toc137395711)

[Searching 19](#_Toc137395712)

[Logout 20](#_Toc137395713)

# Application Design and Testing

## Class Design

After the project was complete, a UML class diagram was generated using the Diagram tool in IntelliJ IDEA. The diagram shows the relationship between the part and product entities and how they interact with the controllers, services, repositories, and tests in the application. The diagram is a visual representation of the application’s structure, and it highlights the dependencies and interaction among the components of the application.

## UI Design

A screenshot of a computer

Description automatically generatedThe UI is intentionally simplistic as users of varying levels of technical expertise will be using the application. The home screen consists of a Parts table and Product table with straightforward navigation buttons that make it easy to locate the operation the user wishes to perform. Included is a low fidelity wireframe that was used as a mockup for later design stages. Additionally, the a high fidelity wireframe demonstrates the final design stages.

A screenshot of a computer

Description automatically generated

Figure 2: High Fidelity

Figure 1: Low Fidelity

# Unit Test Plan

## Introduction

### Purpose

The purpose of this unit test was to validate the functionality of the LogEntry Class. The tests aimed to verify that each field is returned correctly from the respective getter methods. Any inconsistencies or invalid results would cause further debugging, defeating the purpose of the addition of the class and its logging functionality counterparts.

### Overview

Because the LogEntry class is responsible for managing log entries’ details, it is a crucial component of understanding the application’s behavior. A script was written to test each getter method in the class individually to ensure that an error in one function would not affect another’s test. All errors are logged and should be analyzed in order to debug the code base.

## Test Plan

### Items

The JUnit 5 testing framework was used to test the LogEntry class

### Features

1. Test the timestamp getter function.
2. Test the user action getter function.
3. Test the level getter function.
4. Test the thread getter function.
5. Test the logger getter function.

### Deliverables

The test code and the documentation of the test case and test results are the deliverables.

### Tasks

1. Set up the test environment (IntelliJ).
2. Write the test script and test cases for each feature.
3. Locate the test in the test folder of the project’s directory.
4. Right-click LogEntryTest and click Run.
5. Document results.
6. If a test fails, identify the cause, and correct it.

### Needs

These tests require Java 8 or above and JUnit 5.

### Pass/Fail Criteria

If the returned value matches the expected value for each of the test’s methods, the test is considered a success. In the event of a failure, the code will be analyzed, debugged, and corrected before the test is run again.

## A screenshot of a computer program Description automatically generated with medium confidenceSpecifications

## Procedures

1. Imported necessary libraries and LogEntry class into LogEntryTest.
2. Created a separate test case for each getter function of LogEntry.
3. Initialized an object of the LogEntry class in the setUp() method to be used in the test cases.
4. Called each getter method and compared its output to the expected output using the assertEquals function.
5. Ran each test by right-clicking LogEntryTest and clicking Run.
6. Documented the results.

## Results

All the unit tests were successful. The functions of the LogEntry class were thoroughly tested, covering the getter methods for timestamp, userAction, level, thread, and logger. Each of these tests returned the expected result when compared assertEquals from JUnit. Because all tests passed, confidence has increased in the reliability of LogEntry objects, and no code changes are needed currently.

A screenshot of a computer program

Description automatically generated with medium confidence

# Application Link

* <http://perfectpens.us-east-1.elasticbeanstalk.com/>

# GitLab Link

* <https://gitlab.com/wgu-gitlab-environment/student-repos/brady-wgu/d424-software-engineering-capstone>

# Maintenance Guide

## Introduction

The purpose of this guide is to inform team members of the steps needed to update and maintain the application for the purposes of error resolution and system improvement. Below is a list of system requirements and steps needed to redeploy the updated version of the application to AWS.

## Requirements

* Java Development Kit (JDK) 11 or newer
* Apache Maven
* Any Java IDE

## Project Setup

1. Create a new project using your IDE from version control (VCS).
2. Supply the IDE with the GitLab repository link.
3. After the project is successfully cloned, navigate to the root directory in the terminal and run ‘**mvn clean install**”.
4. Next, run ‘**mvn spring-boot:run**’. This will enable you to access the application vial **localhost:8080**.

## Running Tests

1. The application uses JUnit for testing. To run all tests, navigate to the project’s root directory and, in the terminal, run ‘**mvn test**’.
2. Alternatively, individual unit tests can be performed by navigating to the **test** folder of the project, right-click the unit test you wish to perform and, click ‘**Run’.**

## Updating the Application

1. The application uses Docker for image/container management.
2. After completing all necessary changes, navigate to the project’s root directory and run ‘**mvn package’** in the terminal to generate a new .jar file.
3. Ensure that the new name of the .jar matches that which is named in the Dockerfile.
4. Next, build a new Docker image by running ‘**docker build -t perfectpens:latest**’.
5. Log in to the ECR repository using the AWS CLI: ‘**aws ecr get-login-password --region us-east-1 | docker login --username AWS –'**
6. After the new image is built, push the updated image to AWS ECR ‘**docker push public.ecr.aws/p2i9t3d5/perfectpens:latest**’.
7. Navigate to Elastic Beanstalk from AWS console.
8. Locate the perfectpens environment and click ‘**Upload and Deploy**’.
9. Reupload the Dockerrun.aws.json file, and Elastic Beanstalk will take care of the rest.

# User Guide

## Introduction

The PerfectPens Inventory Management System (IMS) is cloud-hosted and prepopulated with the inventory data previously provided by PerfectPens. Because the application is already installed on the web server and the database was configured by the developer, there are no additional installation or setup instructions aside from the login instructions below.

## System Requirements

1. Device Compatibility:
   * Desktop, laptop, or mobile device.
   * Operating Systems: Windows, macOS, Linux, or mobile OS (iOS, Android).
   * Recommended screen resolution: 1280x800 pixels or higher.4
2. Web Browser:
   * Supported web browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari (latest versions)
   * Enable JavaScript and cookies in the browser settings.
3. Internet Connection:
   * Stable and reliable internet connection with sufficient bandwidth to access web applications.

## Logging In

*A screenshot of a login screen

Description automatically generated with medium confidence*

1. In your web browser, navigate to: http://perfectpens.us-east-1.elasticbeanstalk.com/.
2. Enter the login credentials provided in the application’s documentation.
3. Click ‘**Sign In**’.
4. *Note: This login information can only be changed by the developer. Please contact technical support to add users or change login information.*

## Parts

### *Add a New Part*

A screenshot of a computer

Description automatically generated

1. On the home screen, choose either the ‘**Add Inhouse Part**’ or ‘**Add Outsourced Part**’ button.

A screenshot of a computer screen

Description automatically generated with low confidence

1. On the ‘**Outsourced Part Detail**’ page, enter the new part’s details into the form.
2. Click the ‘**Submit**’ button.

### *Update an Existing Part*

A screenshot of a computer

Description automatically generated

1. To update an existing part, find the part in the parts table on the home screen.
2. Click the ‘**Update**’ button.

A picture containing text, screenshot, font, number

Description automatically generated

1. Enter the new information into the form.
2. Click the ‘**Submit**’ button.

### *Delete a Part*

A screenshot of a computer

Description automatically generated

1. To delete an existing part, find the part in the parts table on the home screen.
2. Click the ‘**Delete**’ button.

## Products

### *Add a New Product*

A screenshot of a computer

Description automatically generated

1. On the home screen, click the ‘**Add Product**’ button.

A screenshot of a product

Description automatically generated with low confidence

1. On the ‘**Product Detail**’ page, enter the new product’s details and click the ‘**Submit**’ button.

### *Attach Parts to Products*

A picture containing text, screenshot, font

Description automatically generated

1. To associate a part with a product, find the part on the product’s details page, and click the ‘**Add**’ button.
2. *Note: If you decrease the inventory of a product with associated parts, the inventory of the associated part will also be decremented.*

### *Remove Attached Parts to Products*

A picture containing text, screenshot, font

Description automatically generated

1. To remove an associate part from a product, find the part in the ‘**Associated Parts**’ table on the bottom of the product’s details page, and click the ‘**Remove**’ button.

### *Update an Existing Product*

A screenshot of a computer

Description automatically generated

1. To update an existing product, find the product in the product table on the home screen.
2. Click the ‘**Update**’ button.

A screenshot of a product

Description automatically generated with low confidence

1. Edit the information for the product on the ‘**Product Detail**’ page.
2. Click the ‘**Submit**’ button.

### *Sell a Product*

1. To sell a product, find the product in the product table on the home screen.
2. Click the ‘**Sell**’ button.
3. *Note: Selling a product decrements the inventory of the product by one.*

### *Delete a Part*

A screenshot of a computer

Description automatically generated

1. To delete a product, find the product in the product table on the home screen.
2. Click the ‘**Delete**’ button.

## Generate Log/Report

A screenshot of a computer

Description automatically generated

1. To generate a log for troubleshooting purposes, click the ‘**Application Log**’ button at the bottom of the page on the home screen.

## Searching

A screenshot of a computer

Description automatically generated

1. To search for parts and products, locate the search bar above the appropriate table.
2. Enter the search term.
3. Click the ‘**Search**’ button.
4. To clear the search term and results, press the ‘**Clear**’ button.

## Logout

A screenshot of a computer

Description automatically generated

1. To logout, click the ‘**Logout**’ button at the bottom of the page on the home screen.