**D424 – Software Engineering**

**Task 3**

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

**Table of Contents**

[Table of Contents 2](#_Toc149168821)

[Task 3 Design Document 4](#_Toc149168822)

[Application Design and Testing 4](#_Toc149168823)

[Class Design 4](#_Toc149168824)

[UI Design 4](#_Toc149168825)

[Unit Test Plan 5](#_Toc149168826)

[Introduction 5](#_Toc149168827)

[Purpose 5](#_Toc149168828)

[Overview 5](#_Toc149168829)

[Test Plan 5](#_Toc149168830)

[Items 5](#_Toc149168831)

[Features 5](#_Toc149168832)

[Deliverables 6](#_Toc149168833)

[Tasks 6](#_Toc149168834)

[Needs 6](#_Toc149168835)

[Pass/Fail Criteria 6](#_Toc149168836)

[Specifications 6](#_Toc149168837)

[Procedures 7](#_Toc149168838)

[Results 7](#_Toc149168839)

[Introduction 8](#_Toc149168840)

[Installation and Using the Application 8](#_Toc149168841)

[*Login and Signup (An example*) 9](#_Toc149168842)

[*Classes* 9](#_Toc149168843)

[*Create a New Class* 9](#_Toc149168844)

[*Reports* 10](#_Toc149168845)

**Task 3 Design Document**

D.  Explain how the software product was tested, including the following:

●  a test plan for a unit test, including screenshots

●  unit test scripts

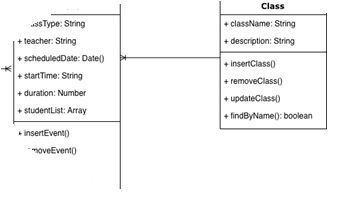
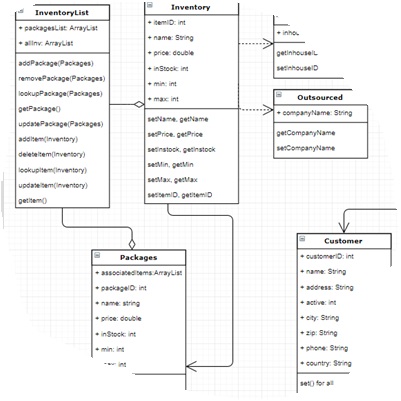
●  the results of the unit tests based on the provided test plan, including screenshots

●  summaries of changes resulting from completed tests

Remove the red instructions and any other example text before submitting.

# Application Design and Testing

## Class Design

Include images here that relate to the class design(s) that you used for the project. Include an introductory paragraph that describes what’s provided.

This section outlines the core classes used in our inventory management application. These classes are essential for defining the structure and functionality of our application. We have four classes: Product, Part (which is an abstract class), and two subclasses of Part - InhousePart and OutsourcedPart. Each class serves a specific role in managing products and parts within the system.

* Product Class: The Product class represents the products available in our inventory. It includes attributes such as product ID, name, price, and inventory. This class is responsible for linking associated parts through a many-to-many relationship, allowing us to define the parts required for assembling products.
* Part Class: The Part class is an abstract class that serves as the parent for both InhousePart and OutsourcedPart subclasses. It contains common attributes for all parts, including part ID, name, price, inventory, maximum and minimum stock levels, and a many-to-many relationship with products.
* InhousePart Class: The InhousePart class is a subclass of Part. It has an additional attribute, part ID, to represent the internal part source. This class allows us to identify parts produced in-house and their specific properties.
* OutsourcedPart Class: The OutsourcedPart class is another subclass of Part. It includes a "companyName" attribute to specify external part suppliers. This class is used to manage parts obtained from external vendors.

By organizing our classes in this manner, we create a flexible and efficient framework for managing products and parts within our inventory management system. These classes provide the foundation for our software's core functionality.

## UI Design

Include images here of your user interface design. You may include both low and high fidelity. Include an introductory paragraph that describes what’s provided.

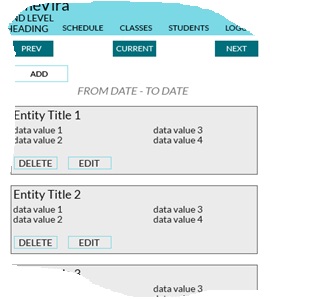
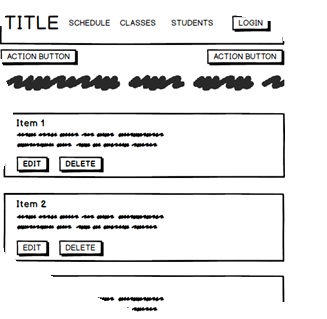


Figure 1: Low Fidelity

Figure 2: High Fidelity

# Unit Test Plan

## Introduction

### Purpose

Provide a brief description of the testing method(s) that you used and what the results it yielded. Also, what remediation was required if necessary and how it would be performed.

The purpose of the Unit Test Plan is to verify the functionality of the PurpleCat PC Store application. By conducting a series of unit tests, we aim to ensure that each part of the application operates as intended and meets high-quality standards. This plan describes the tests, their associated features, the deliverables, the tasks involved, technical requirements, pass/fail criteria, and remediation procedures if any issues arise during testing.

### Overview

Here you go into more detail about the test(s) and how it related to the overall project. You should include if a similar method was used in other parts of the application or why this was unique for a certain aspect of the code. Then, go into detail about what functions were tested, how the tests were conducted, and how errors were dealt with.

The Unit Test Plan is an integral part of the software development process for the PurpleCat PC Store. It focuses on verifying the functionality of key components and features of the application, including user authentication, product management, part management, and report generation. Through rigorous testing, we ensure that the application performs correctly in isolation and as a whole.

## Test Plan

### Items

What is required to complete the test(s)?

Items To successfully conduct the tests, the following items are required:

* Development Environment: A development environment with appropriate software tools, including Java, Spring Boot, MySQL database system, and an integrated development environment (IDE).
* Testing Framework: JUnit, a widely used testing framework for Java applications.
* Test Data: Sample data for testing, including user accounts, product listings, part details, and report information.
* Database Setup: A configured database with predefined data to replicate real-world scenarios.

### Features

List the function/features that are part of each test.

The unit tests will cover the following features:

1. User Authentication: Testing user login. Ensuring that only authorized users can log in to the application.
2. Product Management: Verifying the ability to add, edit, and delete products. Ensuring that data validation and management are functioning as expected.
3. Part Management: Testing the functionality of adding, editing, and deleting parts. Ensuring accurate part management and data validation.
4. Report Generation: Verifying that users can generate accurate reports based on search criteria, including product and part details.

### Deliverables

List what the test(s) would produce. For example, documentation or code notations.

The Unit Test Plan will produce the following deliverables:

* Test Scripts: A collection of test scripts written in JUnit, including test classes and methods.
* Test Results: A detailed report summarizing test outcomes, including successful tests and any failed tests with relevant error messages.

### Tasks

List the tasks required to complete the testing and provide the outcomes you identified.

The testing process consists of the following tasks:

1. Environment Setup: Ensure that the development environment is correctly configured with all necessary tools and dependencies.
2. Test Script Preparation: Confirm the availability of test scripts covering user authentication, product management, part management, and report generation.
3. Database Initialization: Initialize the database with predefined test data, including user accounts, product listings, part details, and report information.
4. Test Execution: Execute the test scripts using JUnit, monitor test results, and capture any errors.
5. Results Analysis: Analyze test results, document issues, and evaluate test outcomes.

### Needs

Describe what was needed to be running or what support items had to be in place to perform the test? Specify versions if appropriate and other technical requirements. If a testing package and/or library was employed, be sure to identify it/them.

To execute the tests effectively, certain needs must be met:

* Software Requirements: Ensure that all necessary software and dependencies, including Java, Spring Boot, JUnit, and any related libraries, are correctly installed.
* Database Setup: Ensure that the database is properly configured with the required schema, tables, and initial test data.
* Access to Source Code: Access to the application's source code and test scripts to review and update them as needed during testing.
* Test Data: Prepare test data to populate the database, including user accounts, product listings, part details, and report information.

### Pass/Fail Criteria

Describe the criteria you used to determine the success of the test and what the protocol was for a positive result. Also describe what the recourse was if the test failed including remediation strategies and documentation requirements.

The criteria for determining the success or failure of each test are as follows:

* User Authentication:
  + Pass: Users can successfully log in with valid credentials, access authorized functionalities, and are denied access with invalid credentials.
  + Fail: Users encounter authentication issues due to incorrect credentials, failed access control, or other errors.
* Product Management:
  + Pass: Products can be added, edited, and deleted without errors. Data is correctly validated, and product listings are accurate.
  + Fail: Product management may encounter issues such as validation errors, data not saving, or incorrect product details.
* Part Management:
  + Pass: Parts can be added, edited, and deleted without errors. Data is correctly validated, and part details are accurate.
  + Fail: Part management may encounter issues such as validation errors, data not saving, or incorrect part details.
* Report Generation:
  + Pass: Users can generate accurate reports based on search criteria, and the reports are displayed correctly.
  + Fail: Report generation may result in problems such as inaccurate reports, search criteria not functioning, or errors in the report generation process.

Remediation Strategies If a test fails during the testing process, the following remediation strategies are applied:

1. Identify the Root Cause: Investigate the failed test to determine the underlying cause of the issue. Examine error messages, logs, and any relevant information.
2. Document the Issue: Create a bug report or issue ticket to document the problem. Include specific test details, error messages, reproduction steps, and any other relevant information.
3. Refine the Application Logic: Update the application logic and test scripts to address the identified issues. This may involve debugging, code modifications, or data adjustments.

## Specifications

Provide sample code that represents what testing code was used. Screenshots are acceptable.

The Unit Test Plan will include sample code snippets that demonstrate the testing procedures and assertions. These code snippets will be part of the test documentation.

javaCopy code

@TestpublicvoidtestProductCreation(){ Productproduct=newProduct("Sample Product", 99.99, 10); assertNotNull(product); assertEquals("Sample Product", product.getName()); assertEquals(99.99, product.getPrice(), 0.001); assertEquals(10, product.getInventory()); }



## Procedures

Provide a detailed list of the steps you used to complete the testing process. Be sure to mention if iterations were/are part of the process used and when pass/fail results were provided.

The testing process involves the following procedures:

1. Test Case Preparation: Identify critical functionalities and features that require testing. Create test cases, define expected outcomes, and establish pass/fail criteria.
2. Test Environment Setup: Configure the testing environment, including tools, dependencies, and the application's server.
3. Test Execution: Run the test suite using JUnit, record the test results, and capture any failures or issues.
4. Test Results Review and Documentation: Document the test results, including pass/fail outcomes, identified issues, and any modifications made to the application code or test cases. Update the test plan and test case documentation to reflect the final test outcomes.

## Results

Here you will describe and provide examples of the testing results. If you were using a testing package include a screenshot of the interface. Screenshot work best.

The test results for the PurpleCat PC Store will be obtained after executing the test scripts using JUnit. The results will include detailed descriptions and screenshots, when applicable, to illustrate the testing outcomes, covering both successful tests and tests that encountered issues.

By following this detailed Unit Test Plan, we aim to ensure the reliability, accuracy, and functionality of the PurpleCat PC Store application, providing a robust and high-quality software solution.



*C2. Provide a link to where the web app is hosted with HTML code (if applicable).*

*C3. Provide a link to the GitLab repository of the code indicating the version included in this submission.*

C4. User guide for setting up and running the application for maintenance purposes.

C5.  User guide for running the application from a user perspective

. **User Guide**

## Introduction

Provide a description of the content you’re providing in the User Guide. This guide will include how to install, log into, sign up, and use all of the functions of the application. The steps need to be clearly defined and fully tested so the process works flawlessly for the evaluator.

## Installation and Using the Application

This procedural information should follow the basic rules of such technical references. While some procedures may provide for personal judgment yours should be clear and concise. Here are other rules to remember:

* Provide step-by-step sequences in the correct order.
* Follow the timing and sequencing of the actual operations.
* Provide visual stepping stones by using bullets or labeling steps.
* Strive to be concise. Avoid lengthy paragraphs but include enough detail so false assumptions are not made.
* Use common terms and jargon appropriate for the audience (someone with basic IT background).
* Explain why steps are completed or what they will yield as well as "How to" instructions.
* Test the instructions to ensure they match the actual product.
* Format the material for ease of reading and use graphic aids to clarify point/steps.
* Write in the present tense and the active voice.

## *Login and Signup (An example*)

1. *Click the "Log in" button in the top right corner of the app.*

**

1. *If you already have an account, log in with your account name and password. If you need an account, click on the link below that states “Need an account?”*
2. *If you need to create an account, choose a unique username and password. By default, the password requires at least 6 characters. This function could be changed to address new password requirements.*

## *Classes*

### *Create a New Class*

1. *Once logged in, click on the link at the top labeled “Classes”. This will enable you to create a new class of students.*

**

1. *Click on “+ Add Class”.*

**

1. *Enter a class name and its description. The class name must be unique.*
2. *Click “Add Class” to add the class, otherwise click “Cancel” or outside of the modal to cancel adding the class.*



## *Reports*

1. *To access the reporting feature, from the Schedule module, click on “Generate Report” near the top right of the page.*

**

1. *By default, all events are generated and displayed.*