

D424 – Software Engineering

Task 3



Capstone Proposal Project Name: PurpleCat PC Store Inventory Management Application

Student Name: Jin Luo

Table of Contents

<i>Table of Contents.....</i>	<i>2</i>
<i>Task 3 Design Document</i>	<i>4</i>
<i>Application Design and Testing.....</i>	<i>4</i>
Class Design.....	4
UI Design	6
<i>link to Web App Hosted.....</i>	<i>14</i>
<i>link to GitLab Repository.....</i>	<i>14</i>
<i>GitLab Repository Branch History</i>	<i>15</i>
<i>User Guide for Setting Up and Running the Application</i>	<i>16</i>
Introduction	16
System Requirements.....	16
Setting up Frontend (React)	16
Setting up Backend (Java Spring Boot).....	17
<i>User Guide for Running the Application from A User Perspective</i>	<i>18</i>
User Login Page.....	18
Home Page	19
User Logout:.....	19
Products Page.....	20
View Products	20
Add Products.....	21
Update Products	22

Delete Products	23
Parts Page.....	24
View Parts	24
Add Parts.....	25
Update Parts	26
Delete Parts	27
Generate Reports.....	29
Generate Reports for Products.....	29
Generate Reports for Parts	30
<i>Unit Test Plan.....</i>	<i>31</i>
Introduction	31
Purpose.....	31
Overview	31
Test Plan	31
Items	31
Features	32
Deliverables	32
Tasks	32
Needs	33
Pass/Fail Criteria	33
Specifications	34
Procedures	36
Results	38

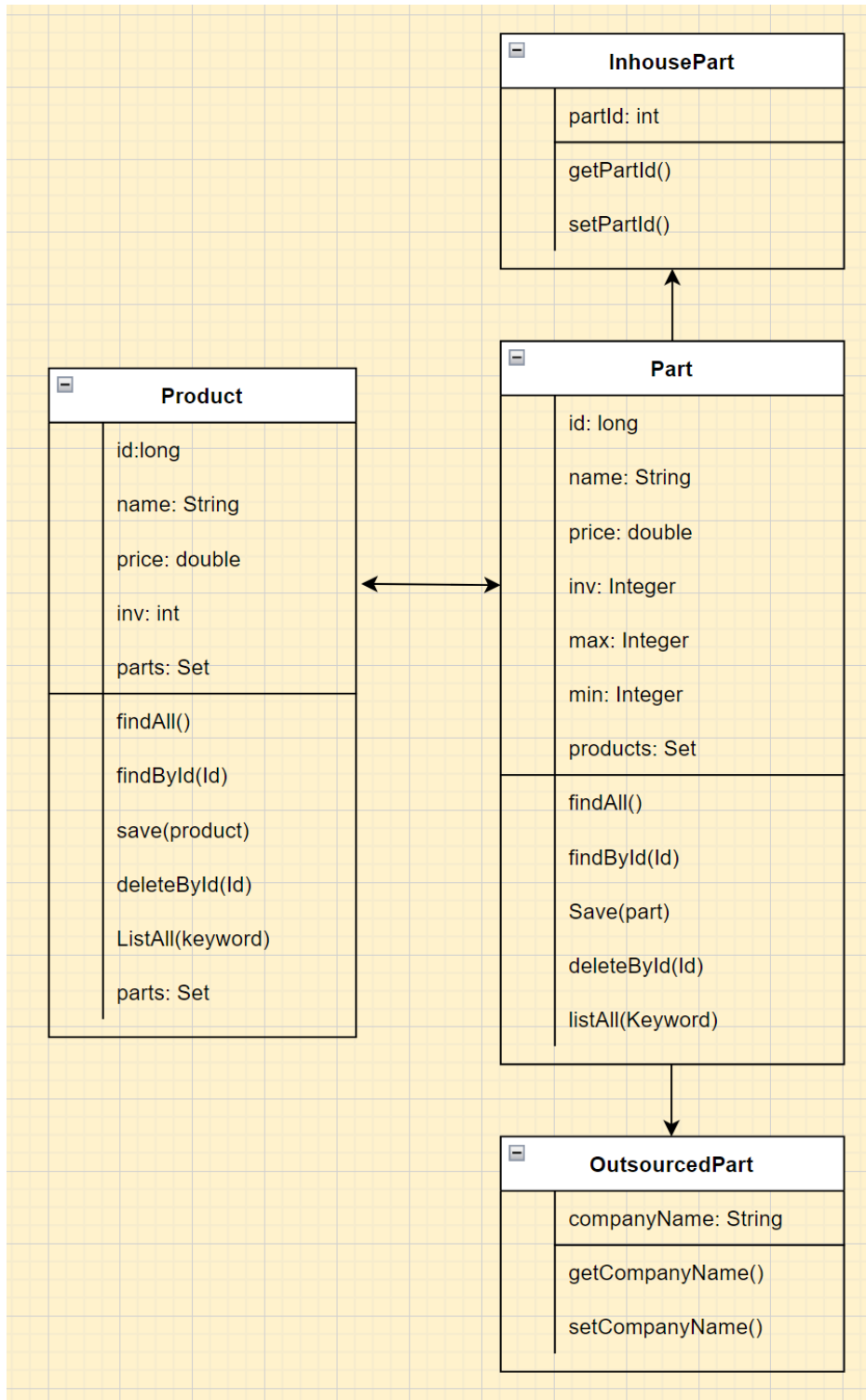
Task 3 Design Document

Application Design and Testing

Class Design

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.

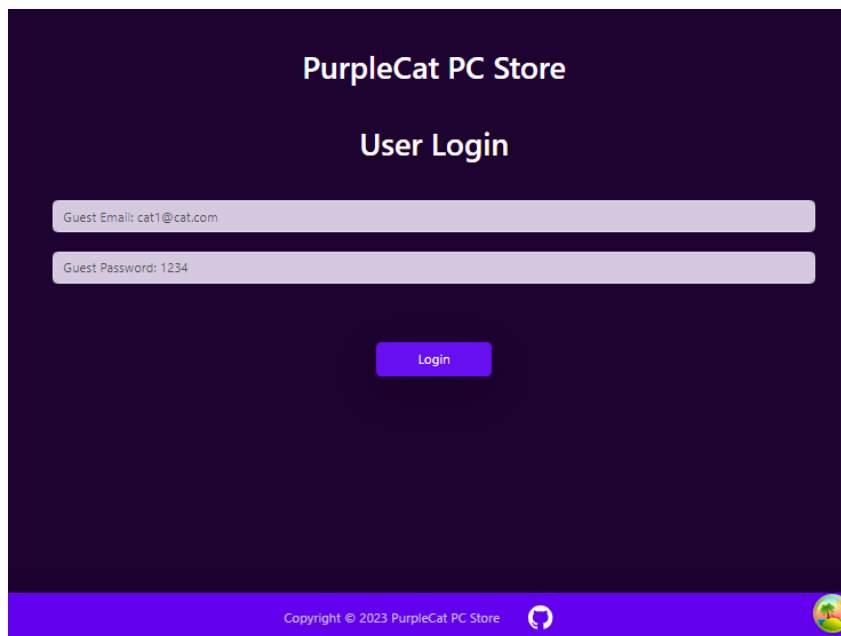
- **Product Class:** The Product class represents the products available in our inventory. It includes attributes such as product ID, name, price, and inventory. This class is responsible for linking associated parts through a many-to-many relationship, allowing us to define the parts required for assembling 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 supplier. This class is used to manage parts obtained from external vendors.



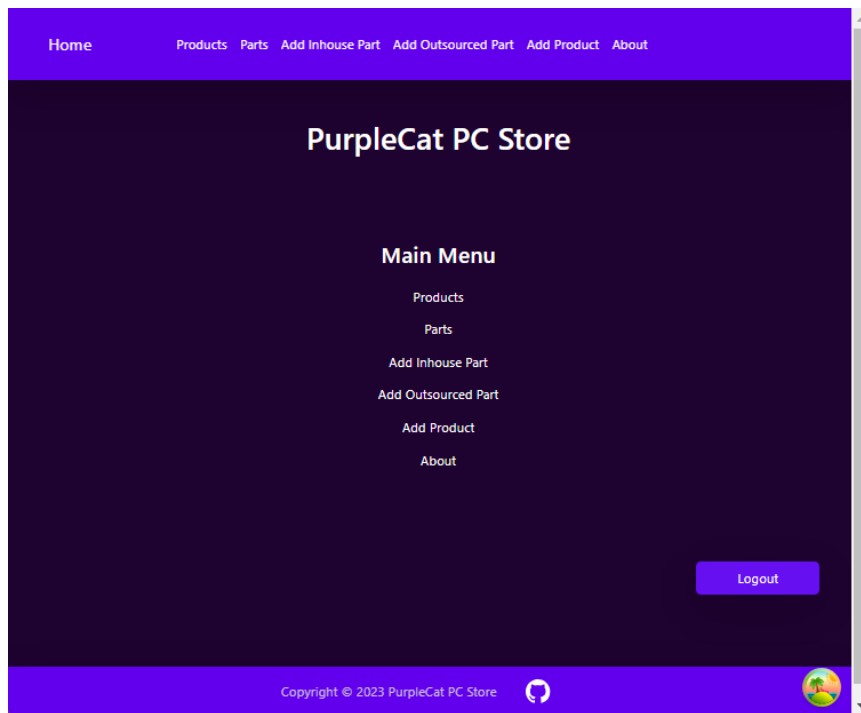
UI Design

In this section, we present the user interface design for our Inventory Management Application. Below, you'll find images that showcase our user interface design, providing an overview of the features and pages we've developed in our application.

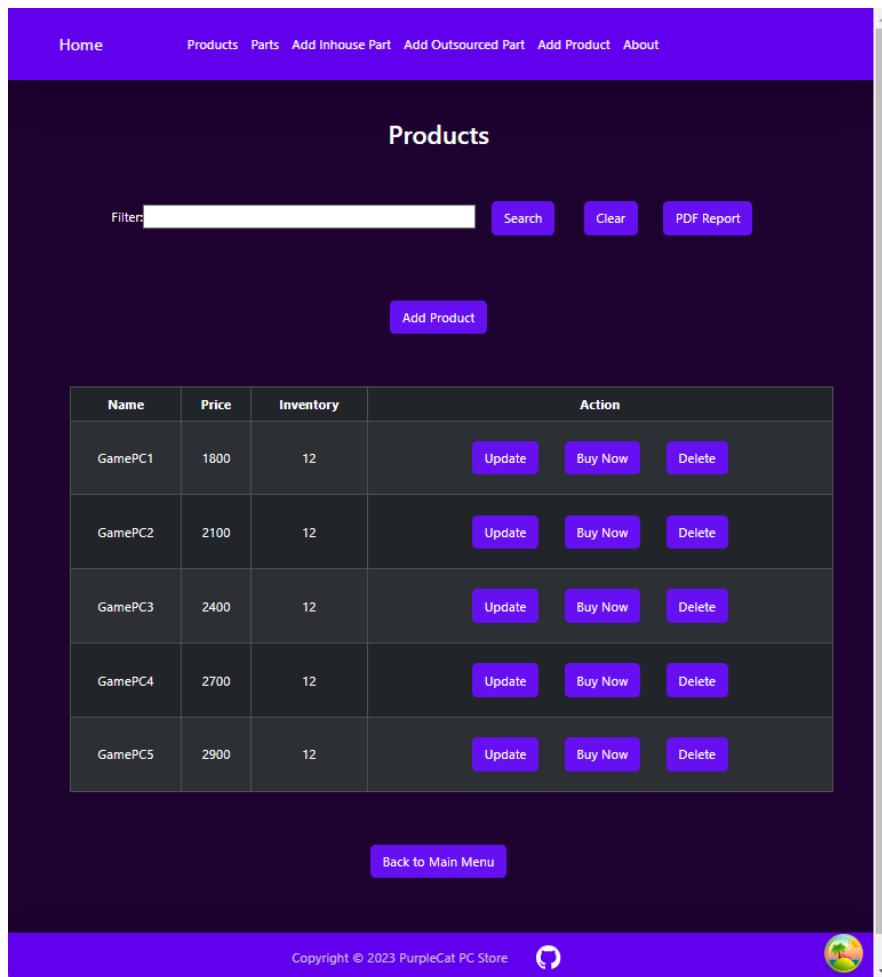
- **User Login Page:** Our User Login Page serves as the entry point to the application, ensuring secure access for authorized users.



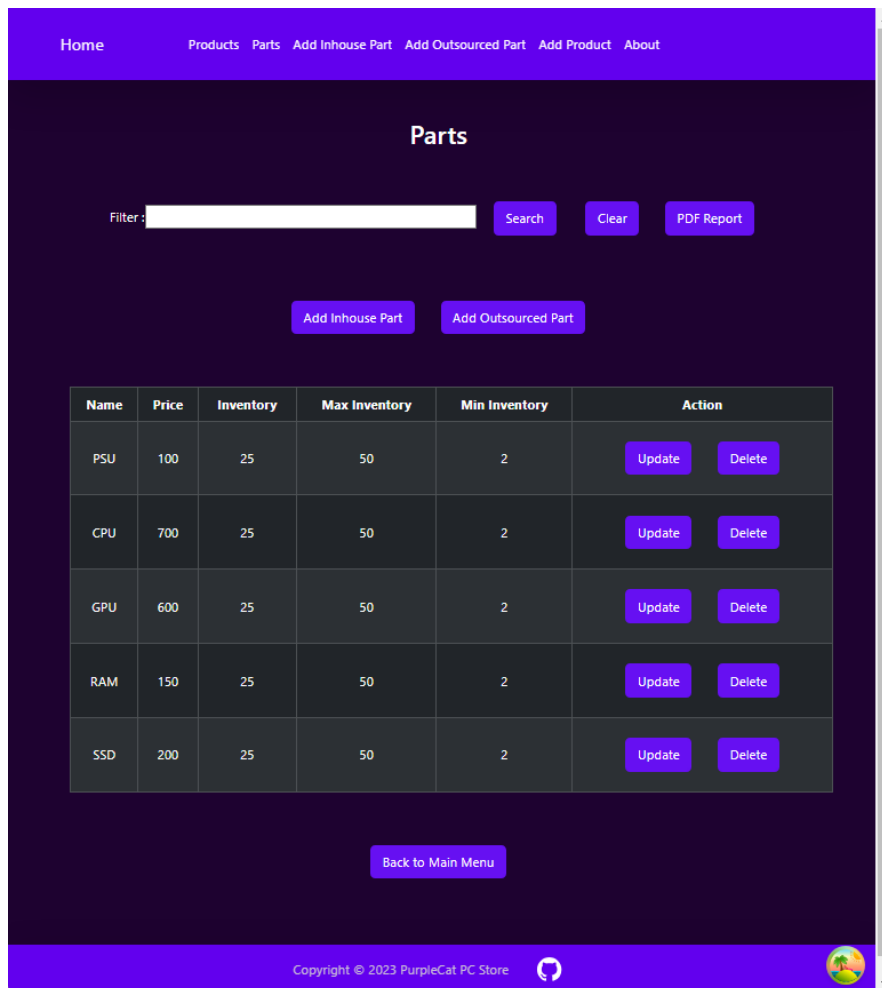
- **Home Page with Main Menu:** The home page serves as your central hub within the application. It offers a user-friendly interface with a main menu that provides quick navigation to other pages.



- **Products Page:** This is where users can manage products, including searching, adding, updating, buying, and deleting products. Additionally, users can generate reports to gain insights into product-related data.



- **Parts Page:** The parts page has similar functionalities to products. Users can search for parts, add in-house or outsourced parts, update part information, delete parts, and generate reports.



- **Outsourced Part Detail Page:** This page offers a deep dive into the details of outsourced parts, provides information about outsourced parts.

The screenshot displays the 'Outsourced Part Detail' page. At the top, a navigation bar includes links for Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The main heading is 'Outsourced Part Detail'. Below this, there are seven input fields for Name, Price, Inventory, MaxInv, MinInv, and Company Name. An 'Add' button is positioned below the input fields, and a 'Back to Main Menu' button is located further down. The footer contains the copyright notice 'Copyright © 2023 PurpleCat PC Store', a GitHub icon, and a small globe icon.

Home Products Parts Add Inhouse Part Add Outsourced Part Add Product About

Outsourced Part Detail

Name

Price

Inventory

MaxInv

MinInv

Company Name

Add

Back to Main Menu

Copyright © 2023 PurpleCat PC Store

- Inhouse Part Detail Page: Here, you'll find detailed information about inhouse parts.

The screenshot displays the 'Inhouse Part Detail' page of the PurpleCat PC Store Inventory Management Application. The page features a dark blue header with a navigation menu containing links for Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The main content area has a dark blue background with the title 'Inhouse Part Detail' in white. Below the title, there are six light blue input fields for Name, Price, Inventory, Max Inventory, Min Inventory, and Part Inhouse ID. At the bottom of the form, there is a blue 'Add' button and a 'Back to Main Menu' button. The footer of the page is dark blue and includes the copyright notice 'Copyright © 2023 PurpleCat PC Store', a logo, and a small globe icon.

Home Products Parts Add Inhouse Part Add Outsourced Part Add Product About

Inhouse Part Detail

Name

Price

Inventory

Max Inventory

Min Inventory

Part Inhouse ID

Add

Back to Main Menu

Copyright © 2023 PurpleCat PC Store

- **Product Detail Page:** The Product Detail Page offers a comprehensive overview of the individual product. Users can add new products or update the details of existing ones. Users can associate available parts with the selected product. This page also provides a list of parts associated with the selected product.

[Home](#) [Products](#) [Parts](#) [Add Inhouse Part](#) [Add Outsourced Part](#) [Add Product](#) [About](#)

Product Detail

[Add](#) [Back to Main Menu](#)



Available Parts

Name	Price	Inventory	Max Inventory	Min Inventory	Action
PSU	100	25	50	2	Add
CPU	700	25	50	2	Add
GPU	600	25	50	2	Add
RAM	150	25	50	2	Add
SSD	200	25	50	2	Add

Associated Parts

Name	Price	Inventory	Max Inventory	Min Inventory	Action
------	-------	-----------	---------------	---------------	--------

Copyright © 2023 PurpleCat PC Store



[Home](#) [Products](#) [Parts](#) [Add Inhouse Part](#) [Add Outsourced Part](#) [Add Product](#) [About](#)

Product Detail

Update



Available Parts

Name	Price	Inventory	Max Inventory	Min Inventory	Action
GPU	600	25	50	2	Add
RAM	150	25	50	2	Add
SSD	200	25	50	2	Add

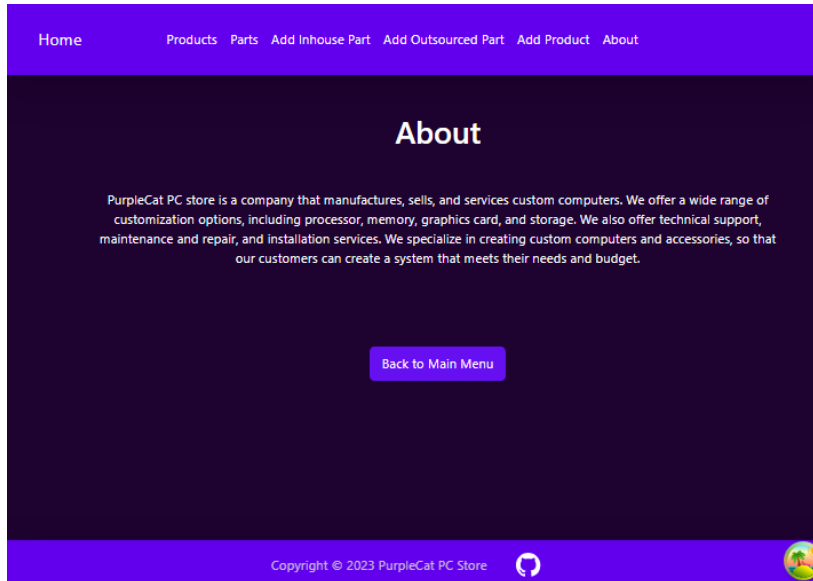
Associated Parts

Name	Price	Inventory	Max Inventory	Min Inventory	Action
PSU	100	25	50	2	Remove
CPU	700	25	50	2	Remove

Copyright © 2023 PurpleCat PC Store



- **About Page:** The About page provides information about PurpleCat PC Store.



















































link to Web App Hosted

<https://purplecat-pc-store.netlify.app>

link to GitLab Repository

https://gitlab.com/wgu-gitlab-environment/student-repos/jluo8/d424-software-engineering-capstone/-/tree/working_branch

GitLab Repository Branch History

Nov 07, 2023		
 Update frontend Jin Luo authored 17 minutes ago	fb2322dc	 
 update README.md Jin Luo authored 31 minutes ago	af28802a	 
 D. Add how the software product was tested Jin Luo authored 1 hour ago	0867ab85	 
 Merge branch 'working_branch' of... ... Jin Luo authored 1 hour ago	85ef1cc4	 
 D. Add how the software product was tested Jin Luo authored 3 hours ago	eb78b15c	 
 C. Add link to the web app, link to the GitLab repository, user guide. Jin Luo authored 3 hours ago	64c69506	 
 D. add unit test plan. scripts Jin Luo authored 3 hours ago	c0950986	 
 D. add unit test plan. scripts Jin Luo authored 3 hours ago	2b030167	 
Nov 06, 2023		
 D. add PartServiceTest.java, ProductServiceTest.java Jin Luo authored 1 day ago	b5c2db7d	 
 C. a design document including a class diagram and design diagram Jin Luo authored 1 day ago	979e4d8d	 
 B. Design and develop a fully functional full stack software product Jin Luo authored 1 day ago	85b265dc	 
 Add backend project Jin Luo authored 1 day ago	0d484c84	 
 first commit Jin Luo authored 1 day ago	d780f72a	 
Feb 27, 2023		
 Added Branching instructions to README Zohar Sajith authored 8 months ago	808c354c	 
Feb 13, 2023		
 Added WGU Template README.md Zohar Sajith authored 8 months ago	5c6cab33	 
 Initial commit Zohar Sajith authored 8 months ago	835201f0	 

User Guide for Setting Up and Running the Application For Maintenance Purposes

Installation and Using the Application

Introduction

This user guide is designed to assist users in setting up and using the PurpleCat PC Store Inventory Management Application in their local environment. The application consists of a frontend built with React and a backend developed using Java Spring Boot, connected to a MySQL database. Users are registered by the company's admin, so they do not need to go through a separate registration process; instead, they can use a guest login provided through the Supabase service. The guest login credentials are as follows:

- Email: ***cat1@cat.com***
- Password: ***1234***

System Requirements

Before you start, ensure that your system meets the following requirements:

- Integrated Development Environment (IDE) such as IntelliJ IDEA and VS Code.
- Node.js and npm (Node Package Manager) for running the frontend.
- Java Development Kit (JDK) 11 or higher
- Apache Maven
- MySQL database server

Setting up Frontend (React)

1. Download the source code.
2. Open a command prompt or terminal.
3. In the command prompt or terminal, navigate to the directory where you've stored the frontend code. ("my-frontend")
4. Run the following command to install dependencies: ***npm install***

5. Run the following command to start the React development server: ***npm start***

The frontend of PurpleCat PC Store Inventory Management Application should now be running at <http://localhost:3000> in your web browser.

Setting up Backend (Java Spring Boot)

- Open your IntelliJ IDEA and open the project from the downloaded source code.
- Configure the application's database settings: Open the "***application.properties***" file located in the `src/main/resources` directory.
 - Replace "`spring.datasource.url`" with the URL of your MySQL database.
 - Replace "`spring.datasource.username`" with your MySQL database username.
 - Replace "`spring.datasource.password`" with your MySQL database password.

Run the application:

- In your IDE, navigate to the project root. Open a terminal window.
- Run the command: ***mvn clean install***
- Run the command: ***mvn clean spring-boot:run***

The backend of purpleCat PC Store Inventory Management Application should now be running at <http://localhost:8080> in your web browser.

➤ ***User Login***

Open your web browser.

1. In the address bar, enter <http://localhost:3000> to access the PurpleCat PC Store Inventory Management Application.
2. Enter the following guest login credentials:
 - Email: ***cat1@cat.com***
 - Password: ***1234***
3. Click the "Login" button. You will be redirected to the Home page.

By following this user guide, you can set up, log in, and use the PurpleCat PC Store Inventory Management Application on your local environment.

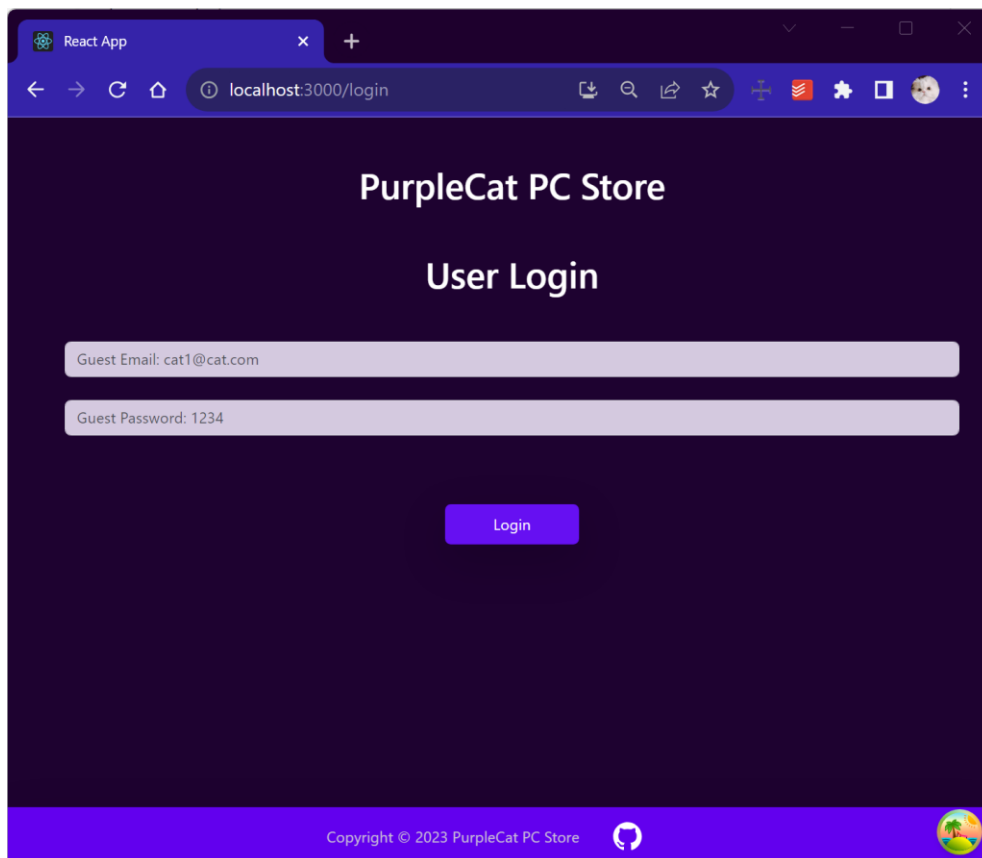
User Guide for Running the Application from A User Perspective

Using the Application

User Login Page

Open your web browser.

1. In the address bar, enter <http://localhost:3000> to access the PurpleCat PC Store Inventory Management Application.
2. Enter the following guest login credentials:
 - Email: **cat1@cat.com**
 - Password: **1234**
3. Click the "Login" button.
4. You will be redirected to the Home page.

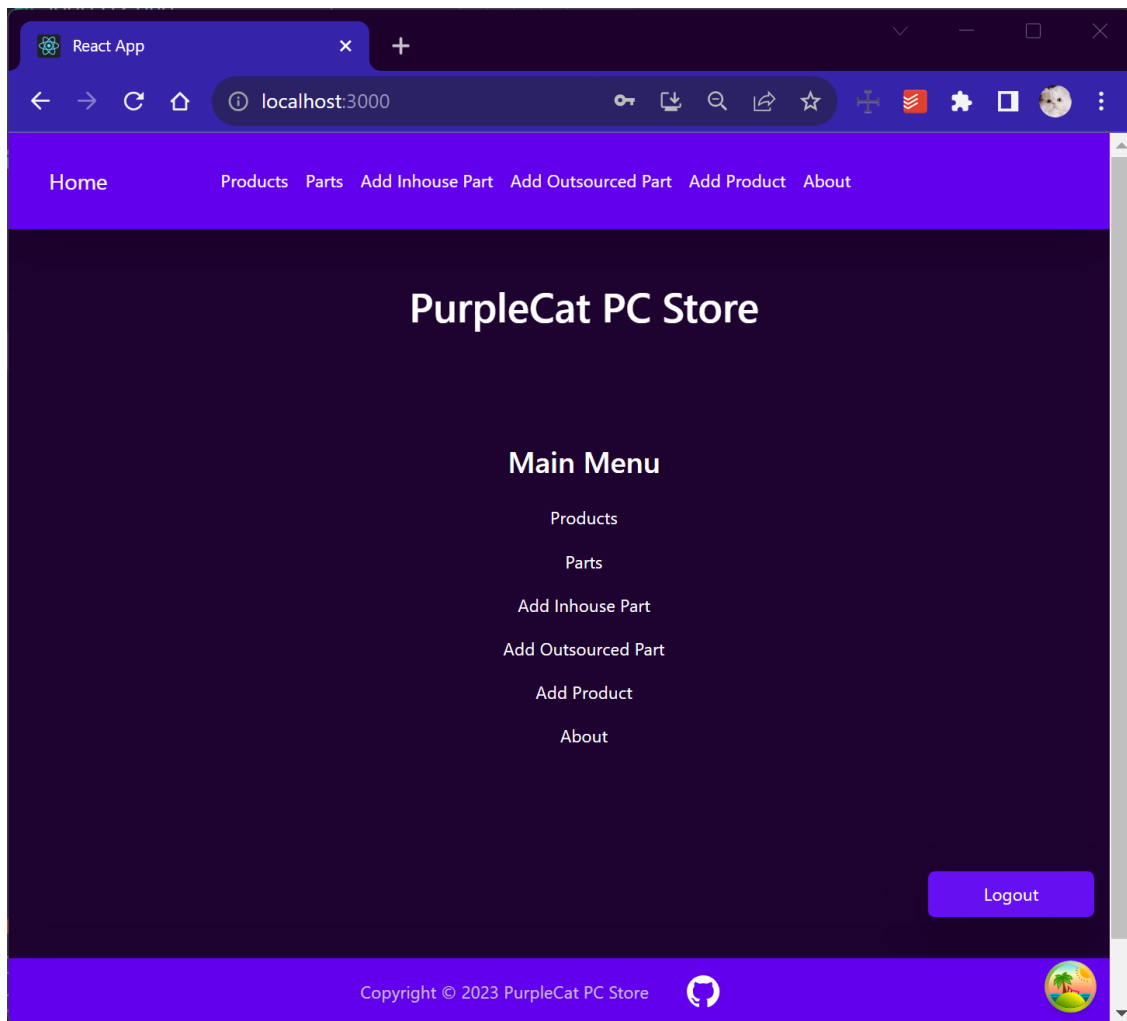


Home Page

Once logged in, you will be on the Home page of the application.

User Logout:

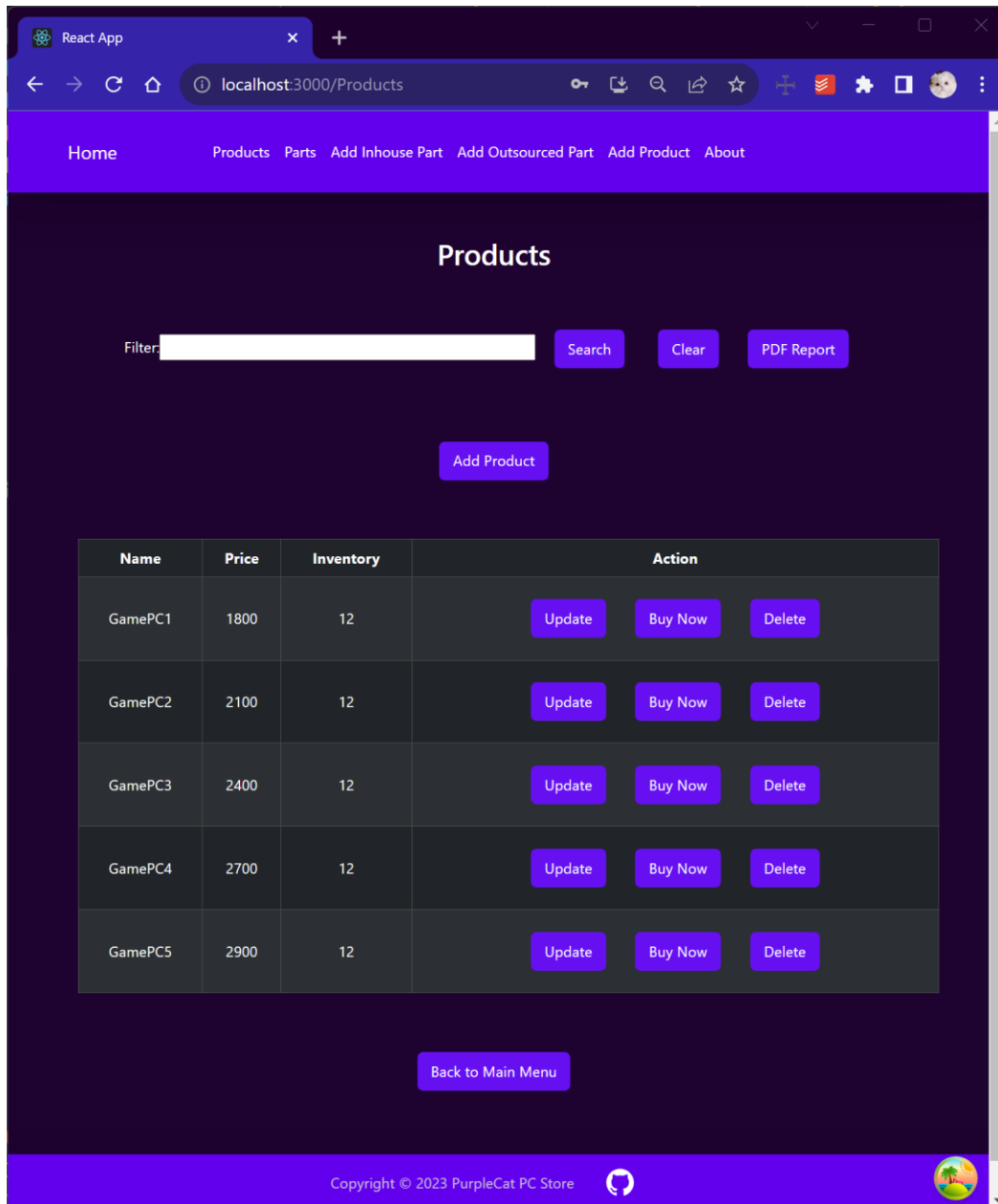
Click the “Logout” button on the Home page, and you will be logged out. It will take you to the “Login” page.



Products Page

View Products

- Click on the “Products” link in the navbar, it will take you to the Products Page where you will find a list of products.



Add Products

- To add a new product into the inventory, click the "Add Product" link in the navbar.
- Fill in the product details, including the product's name and price.
- Click the "Add Product" button to add the new product into the inventory.

The screenshot shows a web browser window with the URL `localhost:3000/ProductDetail`. The application has a dark blue header with a navigation bar containing links: Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The main content area is titled "Product Detail" and features two input fields for "Name" and "Price". Below these fields are two buttons: "Add" and "Back to Main Menu".

Below the form is a section titled "Available Parts" containing a table with the following data:

Name	Price	Inventory	Max Inventory	Min Inventory	Action
PSU	100	25	50	2	<button>Add</button>
CPU	700	25	50	2	<button>Add</button>
GPU	600	25	50	2	<button>Add</button>
RAM	150	25	50	2	<button>Add</button>
SSD	200	25	50	2	<button>Add</button>

Below the table is a section titled "Associated Parts" with a table structure that is currently empty:

Name	Price	Inventory	Max Inventory	Min Inventory	Action
------	-------	-----------	---------------	---------------	--------

The footer of the application displays the copyright notice "Copyright © 2023 PurpleCat PC Store" and a small logo of a cat's head.

Update Products

- To edit a product's details, click the "Update" button on the Products page. It will take you to the Product Detail page.
- Update the product information.
- Click the "Update" button to save the changes.

The screenshot shows a web browser window with the URL `localhost:3000/ProductDetail/5`. The application has a purple header with navigation links: Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The main content area is titled "Product Detail" and contains a form for editing the product "GamePC5". The form has three input fields: "Name" (containing "GamePC5"), "Price" (containing "2900"), and "Inventory" (containing "12"). Below the form is a blue "Update" button.

Below the form, there are two tables:

Available Parts

Name	Price	Inventory	Max Inventory	Min Inventory	Action
CPU	700	25	50	2	<button>Add</button>
GPU	600	25	50	2	<button>Add</button>
RAM	150	25	50	2	<button>Add</button>
SSD	200	25	50	2	<button>Add</button>

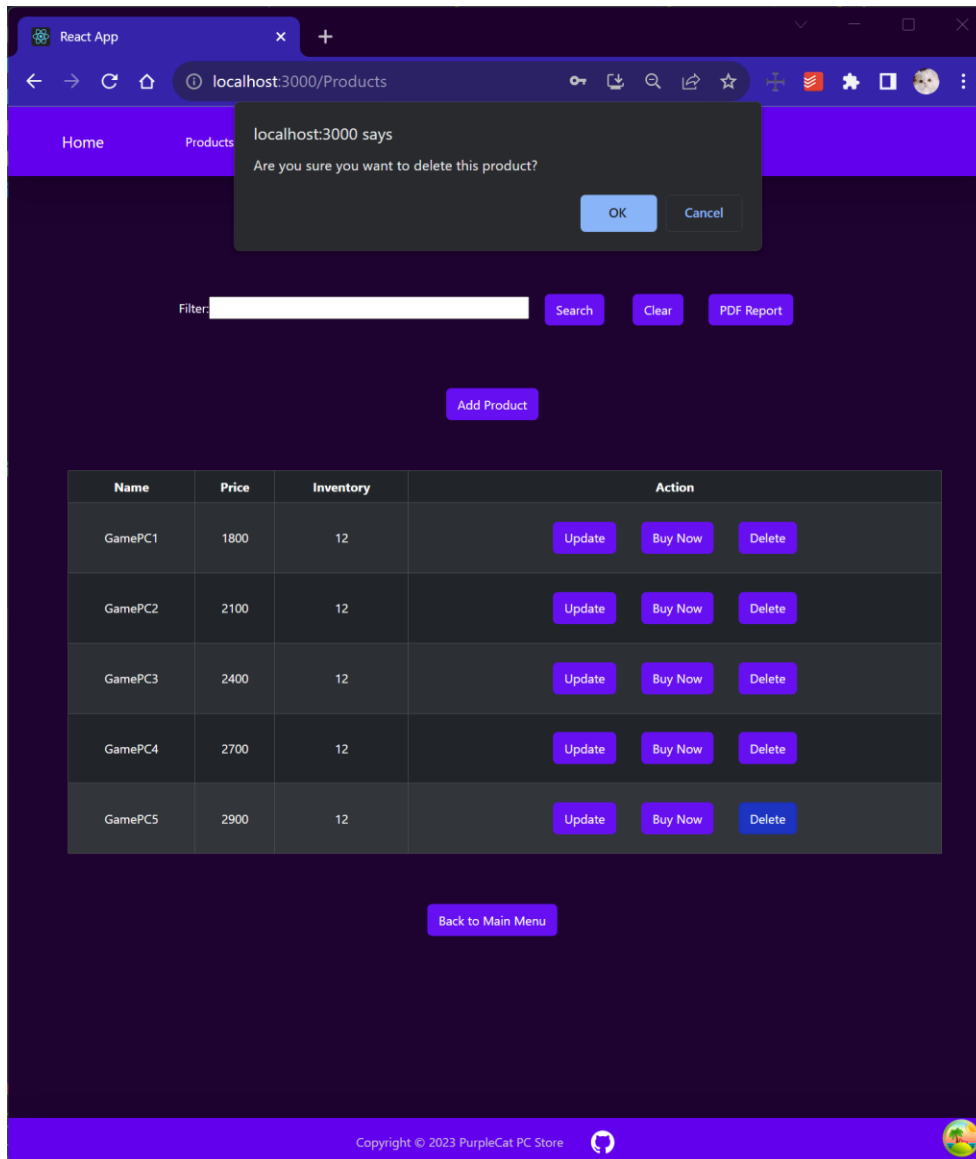
Associated Parts

Name	Price	Inventory	Max Inventory	Min Inventory	Action
PSU	100	25	50	2	<button>Remove</button>

The footer of the application contains the text "Copyright © 2023 PurpleCat PC Store" and a small logo.

Delete Products

- To delete a product from the inventory, click the "Delete" button on the Products page.
- Confirm the deletion when prompted.



Parts Page

View Parts

- Click on the “Parts” link in the navbar, it will take you to the Parts Page where you will find a list of parts.

React App

localhost:3000/Parts

Home Products Parts Add Inhouse Part Add Outsourced Part Add Product About

Parts

Filter: Search Clear PDF Report

Add Inhouse Part Add Outsourced Part

Name	Price	Inventory	Max Inventory	Min Inventory	Action
PSU	100	25	50	2	<button>Update</button> <button>Delete</button>
CPU	700	25	50	2	<button>Update</button> <button>Delete</button>
GPU	600	25	50	2	<button>Update</button> <button>Delete</button>
RAM	150	25	50	2	<button>Update</button> <button>Delete</button>
SSD	200	25	50	2	<button>Update</button> <button>Delete</button>

Back to Main Menu

Copyright © 2023 PurpleCat PC Store

Add Parts

- To add a new part into the inventory, click the "Add Inhouse Part" or "Add Outsourced Part" link in the navbar.
- Fill in the part details, including the part's name, price, inventory, Max inventory, Min inventory and part inhouse ID or company name.
- Click the "Add" button to add the new part into the inventory.

The screenshot shows a web browser window with the URL `localhost:3000/inhousePartForm`. The application has a purple header with a navigation bar containing links: Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The main content area is titled "Inhouse Part Detail" and contains a form with the following fields: Name, Price, Inventory, Max Inventory, Min Inventory, and Part Inhouse ID. Below the form are two buttons: "Add" and "Back to Main Menu". The footer of the application displays "Copyright © 2023 PurpleCat PC Store" and a small globe icon.

The screenshot shows a web browser window with the URL `localhost:3000/OutsourcedPartF...`. The application has a purple header with a navigation bar containing links: Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The main content area is titled "Outsourced Part Detail" and contains a form with the following fields: Name, Price, Inventory, Max Inventory, Min Inventory, and Company Name. Below the form are two buttons: "Add" and "Back to Main Menu". The footer of the application displays "Copyright © 2023 PurpleCat PC Store" and a small globe icon.

Update Parts

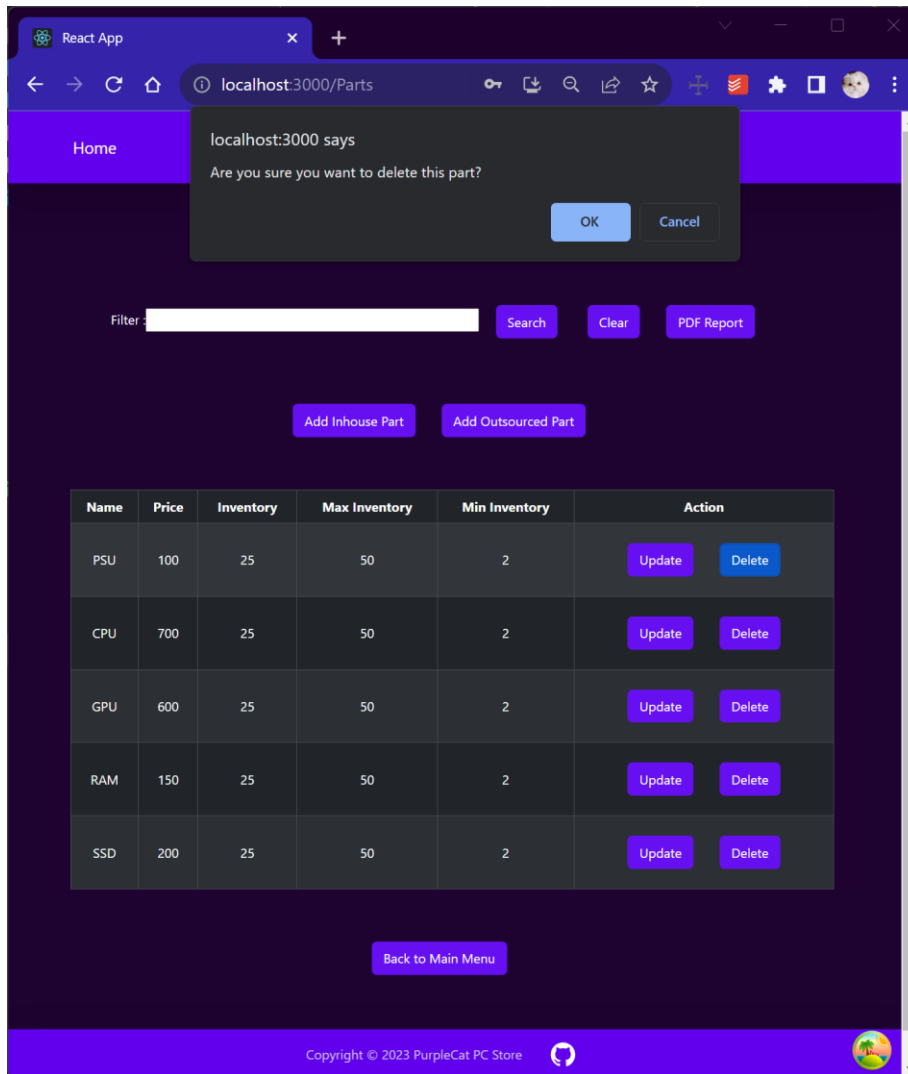
- To edit a part's details, click the "Update" button on the Parts page. It will take you to "Outsourced Part Detail" page or "Inhouse Part Detail" page.
- Update the part information.
- Click the "Update" button to save the changes.

The screenshot shows the 'Inhouse Part Detail' form in a web browser. The browser's address bar shows 'localhost:3000/InhousePartF...'. The application has a purple header with navigation links: Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The form itself has a title 'Inhouse Part Detail' and contains six input fields with the following values: CPU, 700, 25, 50, 2, and 101. Below the fields are two buttons: 'Update' and 'Back to Main Menu'. The footer of the application shows 'Copyright © 2023 PurpleCat PC Store' and a logo.

The screenshot shows the 'Outsourced Part Detail' form in a web browser. The browser's address bar shows 'localhost:3000/OutsourcedP...'. The application has a purple header with navigation links: Home, Products, Parts, Add Inhouse Part, Add Outsourced Part, Add Product, and About. The form itself has a title 'Outsourced Part Detail' and contains six input fields with the following values: PSU, 100, 25, 50, 2, and PurpleCat. Below the fields are two buttons: 'Update' and 'Back to Main Menu'. The footer of the application shows 'Copyright © 2023 PurpleCat PC Store' and a logo.

Delete Parts

- To delete a part from the inventory, click the "Delete" button on the Parts page.
- Confirm the deletion when prompted.



Users cannot delete a part if it is associated with products.

React App

localhost:3000/Parts

Home Products Parts Add Inhouse Part Add Outsourced Part Add Product About

Parts

Filter: Search Clear PDF Report

Add Inhouse Part Add Outsourced Part

Name	Price	Inventory	Max Inventory	Min Inventory	Action
CPU	700	25	50	2	<button>Update</button> <button>Delete</button>
GPU	600	25	50	2	<button>Update</button> <button>Delete</button>
RAM	150	25	50	2	<button>Update</button> <button>Delete</button>
SSD	200	25	50	2	<button>Update</button> <button>Delete</button>

Part cannot be deleted due to existing associations with products.

Back to Main Menu

Copyright © 2023 PurpleCat PC Store

Generate Reports

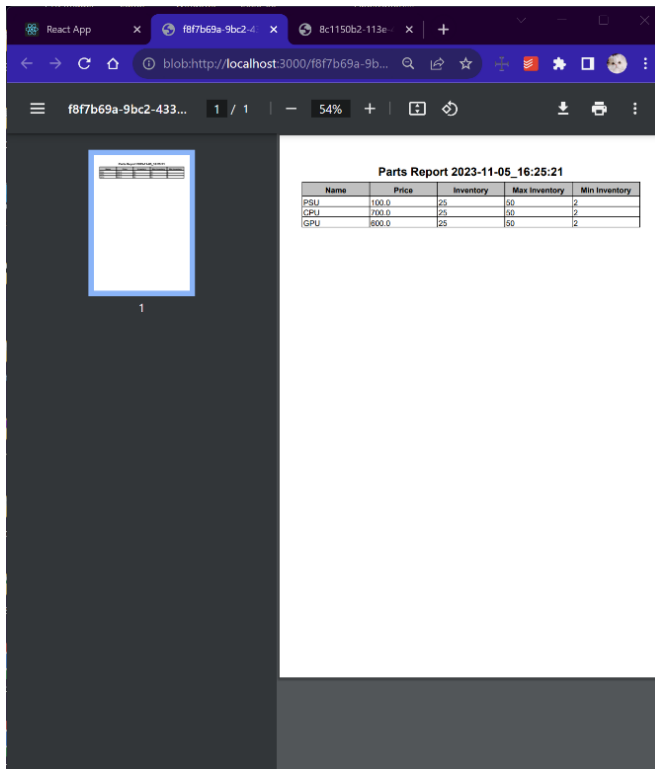
Generate Reports for Products

On the Products page, you can enter a keyword in the search bar, click 'Search' to display the results on the page, and then click 'PDF Report' to navigate to the PDF report page, where you can download or print the report.

Products Report 2023-11-05_16:17:27

Name	Price	Inventory
GamePC1	1800.0	12
GamePC2	2100.0	12
GamePC3	800.0	12
GamePC4	2700.0	12
GamePC5	5000.0	12

Generate Reports for Parts



The screenshot shows a web browser displaying a 'Parts Report' for the date 2023-11-05_16:25:21. The report is presented as a table with the following data:

Name	Price	Inventory	Max Inventory	Min Inventory
PSU	100.0	25	50	2
CPU	700.0	25	50	2
GPU	800.0	25	50	2

On the Parts page, you can enter a keyword in the search bar, click 'Search' to display the results on the page, and then click 'PDF Report' to navigate to the PDF report page, where you can download or print the report.

By following this user guide, you can effortlessly navigate and manage your inventory within the PurpleCat PC Store Inventory Management Application. Whether you are viewing, adding, editing, deleting, or generating reports for products or parts, this guide should help you interact with the application's user-friendly interface.

Unit Test Plan

Introduction

Purpose

The purpose of the Unit Test Plan is to verify the functionality of the PurpleCat PC Store Inventory Management 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

The Unit Test Plan is an important part of the software development process for the PurpleCat PC Store inventory management application. It focuses on verifying the functionality of essential components and features of the application, including product management and part management. Through thorough testing, we ensure that the application performs correctly in isolation and as a whole.

Test Plan

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 products list, parts list, and other information.
- **Database Setup:** A configured database with predefined data to replicate real-world scenarios.

Features

The unit tests will cover the following features:

1. Product Management: Verifying the ability to view, find by ID, update, and delete products, ensuring accurate product management and data validation.
2. Part Management: Verifying the ability to view, find by ID, update, and delete parts, ensuring accurate part management and data validation.

Deliverables

The Unit Test Plan will include 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

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 product management and part management.
3. Database Initialization: Initialize the database with predefined test data, including products list and parts list.
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

- **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 products list and parts list.

Pass/Fail Criteria

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

- **Product Management:**
 - **Pass:** Products can be viewed, found by ID, updated, and deleted without errors. The data is correctly validated, and the products list is accurate.
 - **Fail:** Product management may encounter issues such as validation errors, data not being updated, or incorrect product details.
- **Part Management:**
 - **Pass:** Parts can be viewed, found by ID, and deleted without errors. The data is correctly validated, and the parts list is accurate.
 - **Fail:** Part management may encounter issues such as validation errors, data not being updated, or incorrect part details.

If a test fails, we will:

- Debug and identify the cause of the failure.
- Modify the source code to fix the issue.
- Execute the test and analyze test results to verify that the problem is resolved.

Specifications

The Unit Test Plan will include sample code that demonstrates the testing procedures and assertions. This code will be a part of the test documentation.

```
26
27 @RunWith(MockitoJUnitRunner.class)
28 public class PartServiceTest {
29     @Mock
30     private PartRepository partRepository;
31     @InjectMocks
32     private PartServiceImpl partService;
33
34     @Test
35     public void findAll() {
36         when(partRepository.findAll()).thenReturn(Arrays.asList(
37             new InhousePart( cat1: "Pink1", i: 11, i1: 10, i2: 22, i3: 2, i4: 10001),
38             new OutsourcedPart( cat2: "Pink2", i: 22, i1: 20, i2: 22, i3: 2, company1: "Company1")
39         ));
40         List<Part> parts = partService.findAll();
41         assertEquals( expected: "Pink1", parts.get(0).getName());
42         assertEquals( expected: 22, parts.get(1).getPrice());
43     }
44
45     @Test
46     public void findById() {
47         Part part = new InhousePart( cat1: "Pink3", i: 11, i1: 10, i2: 22, i3: 2, i4: 10001);
48         when(partRepository.findById(100L)).thenReturn(Optional.of(part));
49         Part result = partService.findById( theId: 100);
50         assertEquals( expected: "Pink3", result.getName());
51     }
52
53     @Test
54     public void save() {
55         InhousePart part = new InhousePart( l: 400L, pink1: "Pink4", i: 11, i1: 10, i2: 22, i3: 2, i4: 10001);
56         partService.save(part);
57         verify(partRepository, times( wantedNumberOfInvocations: 1)).save(part);
58     }
59
60     @Test
61     public void deleteById() {
62         Part part = new InhousePart( l: 500, pink1: "Pink5", i: 11, i1: 10, i2: 22, i3: 2, i4: 10001);
63         partService.deleteById( theId: 500);
64         verify(partRepository, times( wantedNumberOfInvocations: 1)).deleteById(500L);
65     }
66 }
```

```
18
19
20 @RunWith(MockitoJUnitRunner.class)
21 public class ProductServiceTest {
22     @Mock
23     private ProductRepository productRepository;
24     @InjectMocks
25     private ProductServiceImpl productService;
26     @Test
27     public void findAll() {
28         when(productRepository.findAll()).thenReturn(Arrays.asList(
29             new Product( name: "Cat1", price: 11, inv: 10),
30             new Product( name: "Cat2", price: 22, inv: 20)
31         ));
32         List<Product> products = productService.findAll();
33         assertEquals( expected: "Cat1", products.get(0).getName());
34         assertEquals( expected: 22, products.get(1).getPrice());
35     }
36
37     @Test
38     public void findById() {
39         Product product = new Product( id: 100, name: "Cat3", price: 11, inv: 10);
40         when(productRepository.findById(100L)).thenReturn(Optional.of(product));
41         Product result = productService.findById( theId: 100);
42         assertEquals( expected: "Cat3", result.getName());
43     }
44
45     @Test
46     public void save() {
47         Product product = new Product( id: 400, name: "Cat4", price: 11, inv: 10);
48         productService.save(product);
49         verify(productRepository, times( wantedNumberOfInvocations: 1)).save(product);
50     }
51
52     @Test
53     public void deleteById() {
54         Product product = new Product( id: 500, name: "Cat5", price: 11, inv: 10);
55         productService.deleteById( theId: 500);
56         verify(productRepository, times( wantedNumberOfInvocations: 1)).deleteById(500L);
57     }
58
59
```

ProductServiceTest > findById()

Procedures

➤ Product Management Testing Process:

❖ Test Product findAll (Iteration):

- The functionality of finding all the products was tested.
- An initial test was conducted to verify successfully finding of all the products.
- The test result was provided immediately after the test, indicating a "Pass."
- An iteration occurred when testing the finding of all products with invalid data.
- After multiple iterations, the test results were provided, indicating "Fail" for invalid product data.

❖ Test Product findById (Iteration):

- The functionality of finding the product by ID was tested.
- An initial test was conducted to verify successfully finding of the products by ID.
- The test result was provided immediately after the test, indicating a "Pass."
- An iteration occurred when testing the finding of the product by ID with invalid data.
- After multiple iterations, the test results were provided, indicating "Fail" for invalid product data.

❖ Test Save Product (Not Iteration):

- The functionality of saving a product was tested by adding a new product.
- An initial test was conducted to verify the successful addition of the product.
- The test result was provided immediately after the test, indicating a "Pass."

❖ Test Delete Product (Not Iteration):

- The functionality of deleting a product was tested by removing a product.
- An initial test was conducted to verify the successful deletion of the product.
- The test result was provided immediately after the test, indicating a "Pass."

➤ Part Management Testing Process:

❖ Test Part findAll (Iteration):

- The functionality of finding all the parts was tested.
- An initial test was conducted to verify successfully finding of all the parts.
- The test result was provided immediately after the test, indicating a "Pass."
- An iteration occurred when testing the finding of all parts with invalid data.
- After multiple iterations, the test results were provided, indicating "Fail" for invalid part data.

❖ Test Part findById (Iteration):

- The functionality of finding the part by ID was tested.
- An initial test was conducted to verify successfully finding of the part by ID.
- The test result was provided immediately after the test, indicating a "Pass."
- An iteration occurred when testing the finding of the part by ID with invalid data.
- After multiple iterations, the test results were provided, indicating "Fail" for invalid part data.

❖ Test Save Part (Not Iteration):

- The functionality of saving a part was tested by adding a new part.
- An initial test was conducted to verify the successful addition of the part.
- The test result was provided immediately after the test, indicating a "Pass."

❖ Test Delete Part (Note Iteration):

- The functionality of deleting a part was tested by removing a part.
- An initial test was conducted to verify the successful deletion of the part.
- The test result was provided immediately after the test, indicating a "Pass."

In summary, iterations were an important part of the testing process, and test results were provided after each iteration, indicating whether the tests passed or failed. These iterations were essential for identifying and addressing issues with the application's functionality.

Results

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

➤ Product Management Test Results:



❖ *Product findAll() Test:*

- Purpose: This test ensures that all products can be found successfully.
- Result: The test passed when all products were found successfully.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

❖ *Product findById() Test:*

- Purpose: This test verifies that product information can be successfully found by ID.
- Result: The test passed when the product was successfully found by ID.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

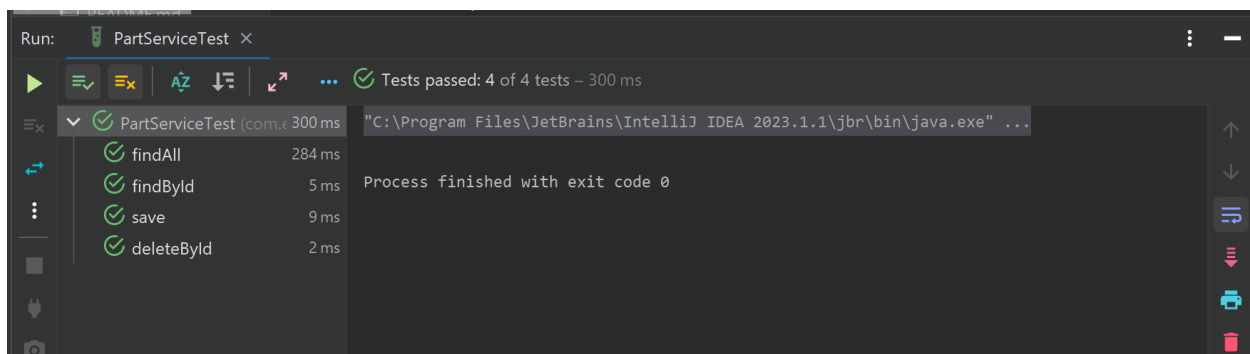
❖ *Product save() Test:*

- Purpose: This test verifies that product information can be saved successfully.
- Result: The test passed when a product was successfully saved.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

❖ *Product deleteById() Test:*

- Purpose: This test ensures that products can be deleted successfully.
- Result: The test passed when a product was successfully deleted.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

➤ Part Management Test Results:



❖ *Part findAll() Test:*

- Purpose: This test ensures that all parts can be found successfully.
- Result: The test passed when all parts were found successfully.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

❖ *Part findById() Test:*

- Purpose: This test verifies that part information can be successfully found by ID.
- Result: The test passed when the part was successfully found by ID.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

❖ *Part save() Test:*

- Purpose: This test verifies that part information can be saved successfully.
- Result: The test passed when a part was successfully saved.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

❖ *Part deleteById() Test:*

- Purpose: This test ensures that parts can be deleted successfully.
- Result: The test passed when a part was successfully deleted.
- Pass/Fail Criteria: Pass
- Remediation: No remediation required.

In summary, the unit tests for product management and part management all passed successfully. Since all tests have passed, no remediation is required at this stage. These tests confirm that the application's core features are functioning as expected.