D424 – Software Engineering

Task 3

Capstone Proposal Project Name: FretWorks Guitar Co. Inventory Management System

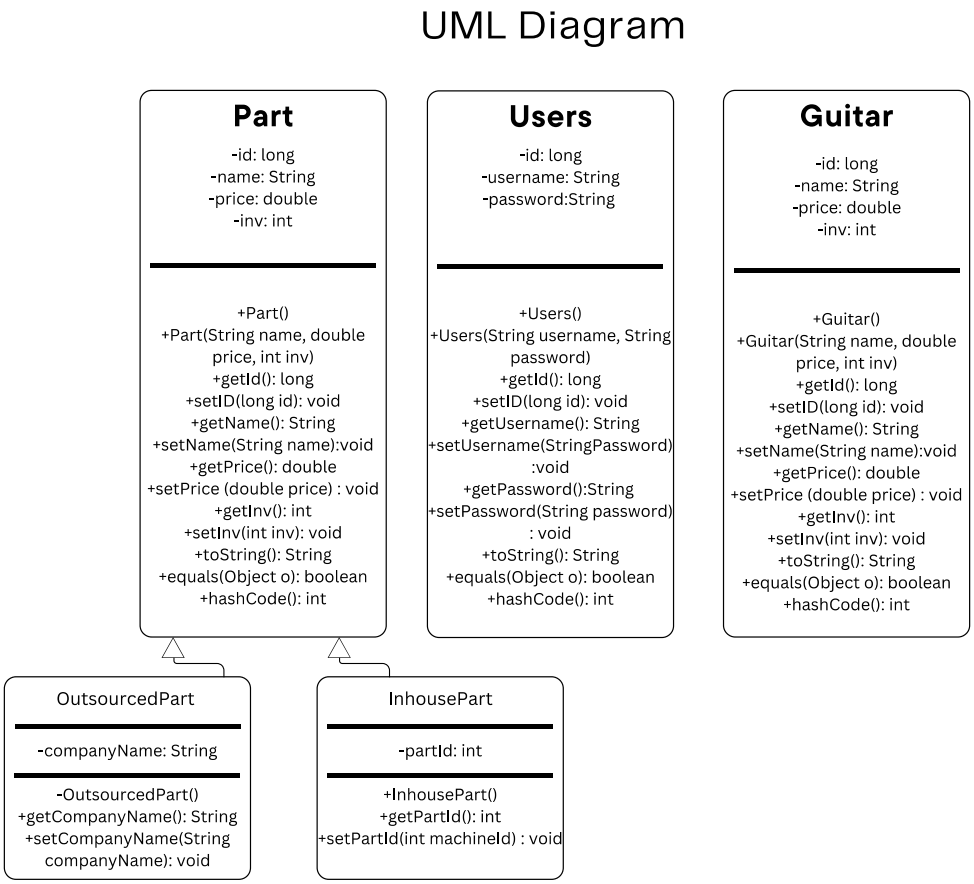
Student Name: Aislyn Mildenhall

Table of Contents

Application Design and Testing

Class Design

The UML diagram provided below displays the main classes that build the foundation for the Inventory Management System. The main classes include the Users, Guitar, and Part classes as well as the Inhouse Part and Outsourced Part subclasses. The diagram serves as a very useful tool in the making of the application as it displays the attributes, methods, and relationships used to create the classes and corresponding functionality.



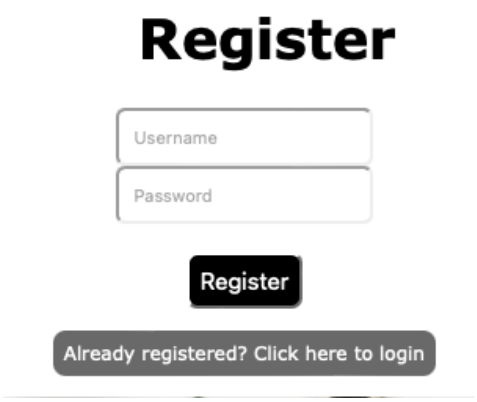
UI Design

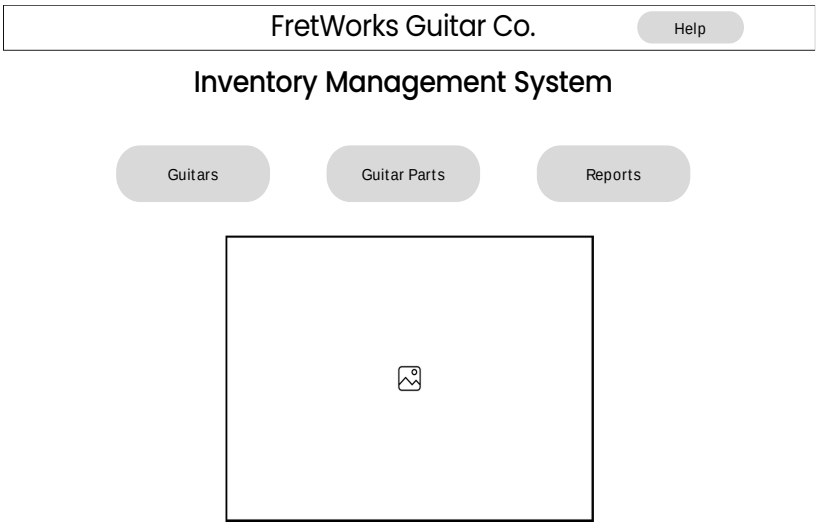
The User Interface for the Inventory Management System includes 12 main layouts; these layouts include the following displays:

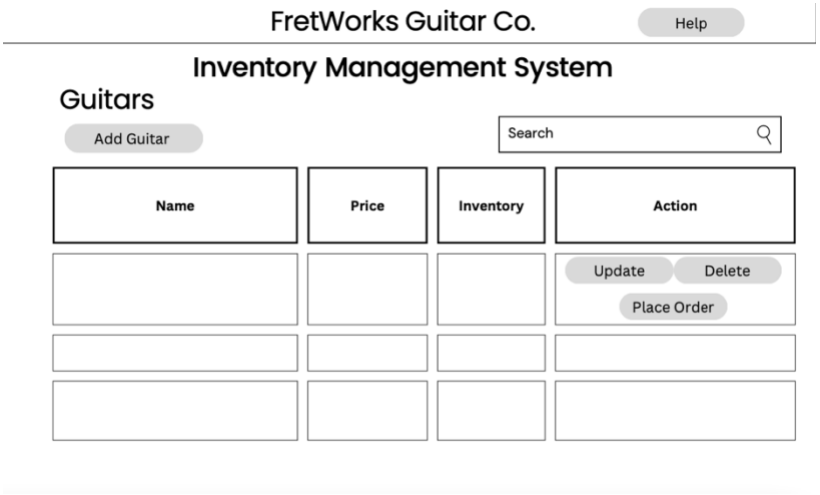
- Login – The user will login with their registered credentials to access the web application.
- Register –The user will register with their own username and password if they don't have one already.
- Error – The user will be directed to the error page if they don't provide the correct username and password for the login page or if their registration username/password have errors.
- Homepage—The user can access the main screens from the homepage after they login. They can also logout from this page.
- Guitar Inventory – The user can view the guitar inventory as well as place a guitar for order, update, and delete a guitar by accessing the guitar forms. The user can also search for specific guitars.
- Part Inventory –The user can view the part inventory as well as update and delete a part by accessing the part forms. The user can also search for specific parts.
- Guitar Details Form – The user is directed to this page to add or update a guitar with the designated guitar details.
- Part Details Form –The user is directed to this page to add or update a guitar part with the designated part details.
- Reports –The user may generate a guitar or guitar part report.

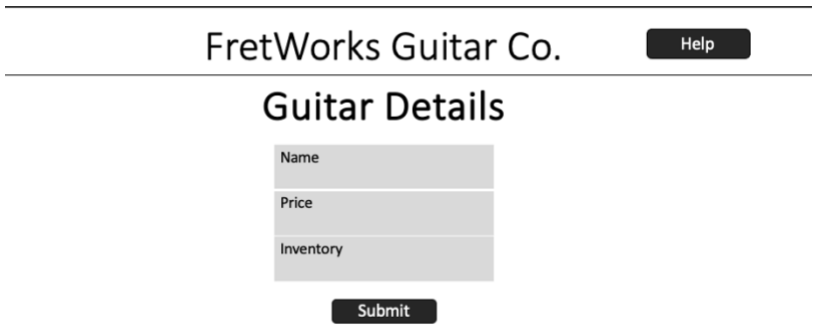
- Guitar Report – The user may filter through the guitar inventory as well as print the report.
- Part Report – The user may filter through the part inventory as well as print the report.
- Help Page – The user is provided with the IT contact information if they need technical support.

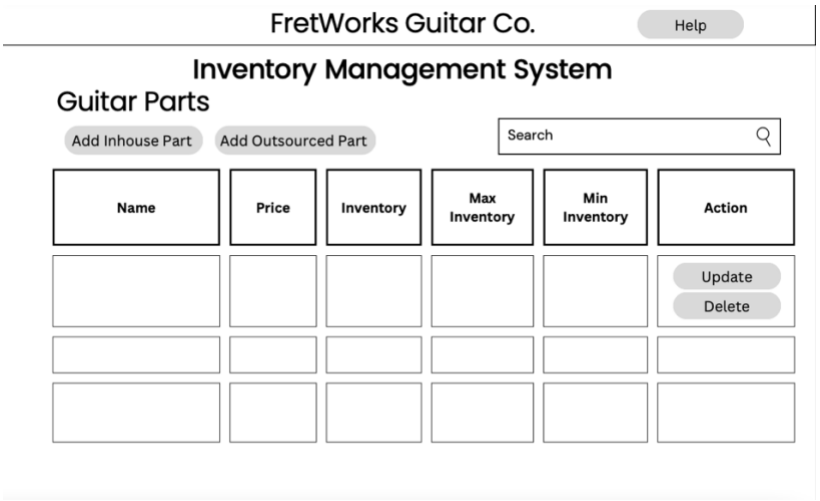
Displayed below are the low-fidelity wireframes displaying the UI for each of the discussed layouts. The wireframes closely follow the end result of each layout and prove to be simple, effective, and user-friendly. The wireframes provided include the login and register pages, the homepage, the guitar inventory page, the guitar details page, the part inventory page, the part details page, and the part report page.

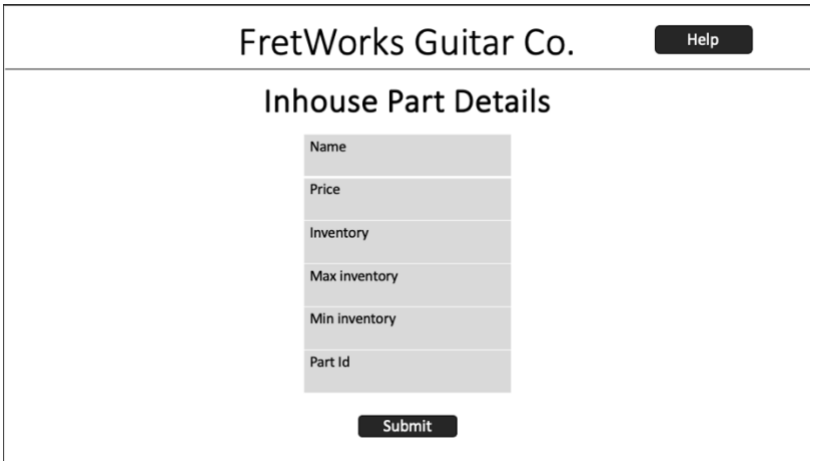


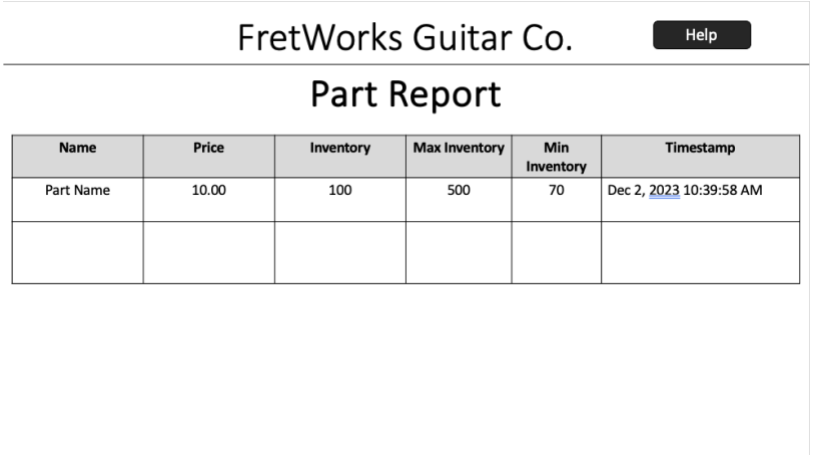

FretWorks Guitar Co.
Help
Inventory Management System
Guitars
Guitar Parts
Reports


FretWorks Guitar Co.
Help
Inventory Management System
Guitars
Add Guitar
Search
Name
Price
Inventory
Action
Update
Delete
Place Order


FretWorks Guitar Co.
Help
Guitar Details
Name
Price
Inventory
Submit


FretWorks Guitar Co.
Help
Inventory Management System
Guitar Parts
Add Inhouse Part
Add Outsourced Part
Search
Name
Price
Inventory
Max Inventory
Min Inventory
Action
Update
Delete


FretWorks Guitar Co.
Help
Inhouse Part Details
Name
Price
Inventory
Max inventory
Min inventory
Part Id
Submit



Unit Test Plan

Introduction

Purpose

Our team of software engineers started the testing process off by performing unit testing to ensure all units and methods were working correctly. Once all unit tests passed, manual testing began to ensure correct data input and output, data validation, and data deletion.

Displayed below are the test cases for the manual tests performed for each page, as well as the test scripts using Junit, and the test results with the corresponding screenshots.

Overview

Our team of software developers used Java, an object-oriented programming language, therefore it was beneficial to incorporate Unit Testing and test each unit and the methods within each unit. These methods include the CRUD capabilities within each class. After performing unit testing within the IDE, the software engineers retested any unit tests that failed and documented the specific unit test, failure, retest, and remediation. In this case, no unit test cases failed. Upon all unit tests passing, the software developers could move on to manual testing.

Manual testing included launching the application in its working environment, performing each user interaction within each page, and validating the function, appearance, performance, and output. The software developers retested any test cases that failed and documented the failure in the Test Results document with the test case, reason for failure, and remediation.

Test Plan

Needs

-Software requirements: Windows 10 or above, OS 12 or above

-Testing tools: Junit, IDE

-Source code and scripts

-Test data, provided within test scripts

Items

-Development Environment: IDE such as IntelliJ IDEA

-Viable working environment to run the application and IDE with Java 17 installed

-Valid Test Data

-Access to the database

-Test Framework: Junit

-Source Code

-Unit test plan and manual test cases

Deliverables

-Test Plan

-Test Cases

-Test Results

-Test Failures and Remediation

Tasks

To perform the testing process, unit tests must be written and ready to be run. Unit tests will then be run within a viable IDE such as IntelliJ IDEA. Unit tests will be run, and any failures will be documented, assigned to a developer, and fixed. This process will be repeated until all unit tests pass.

Manual testing will then take place by running the code in the working environment. The software engineer will go through each test case and run each test with each test script until all manual tests pass. Upon testing failure, the test case will be documented, fixed, and retested. This process will be repeated until all manual tests pass.

Pass/Fail Criteria

In the test plans displayed below, the description and test scripts are clearly described with the expected result for each test case. If the test case does not meet the expected result, then the test would be considered a fail. If the test case does meet the expected result, then the test is considered a pass.

Manual Testing:

Register Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
1	Register username and password with correct credentials	1. On login page, select the “Click here to register” button. 2. Type “admin” into username	Be redirected to login page after clicking register. No errors or validation messages	Successfully registered and redirected to login page without any errors or validation messages shown.	Pass

		3. Type “admin123!” into password 4. Click register	should be shown.		
2	Register username and password with blank fields	1. On login page, select “Click here to register” button. 2. Leave user name field blank. 3. Click register	User is prompted to fill out the empty username field.	User was prompted to fill out the empty username field.	Pass
3	Click go to Login	1. Click the “Already registered? Click here to login” button	User is redirected to the login page	User was redirected to the login page	Pass

Login Page:

Test Case:	Description	Steps:	Expected Results:	Actual Results:	Pass or fail:
4	Login with correct registered credentials	1. On login page, type “admin” into username field 2. Type “admin123!” into password field 3. Click “Login” button	Login and access application without any errors, be redirected to homepage	Login and access application without any errors, be redirected to homepage	Pass
5	Login with blank fields	1. Leave user name and password fields blank 2. Click “Login”	Be prompted to fill out blank fields	Prompted to fill out blank fields	Pass
6	Login with incorrect credentials	1. Type “admin” into username field 2. Type “admin” into password field 3. Click “Login”	Be redirected to error page	Redirected to error page	Pass
7	Go to register page	1. Click “Click here to register” button	Be redirected to register page	Redirected to register page	Pass

Error Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:

8	Go to login page	1. Click “Navigate to login” button	Navigates to login page	Navigates to login page	Pass
9	Go to register page	1. Click “Navigate to registration” button	Navigates to registration page	Navigates to registration page	Pass

Homepage:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
10	Logout of application	1. Click “logout” button in top right corner	Navigates back to login screen	Navigates back to login screen	Pass
11	Go to help page	1. Click “help” button in top right corner	Navigates to help screen with IT contact info displayed	Navigates to help screen with IT contact info displayed	Pass
12	Go to guitar inventory page	1. Click “Guitars” button	Navigates to guitar inventory pages	Navigates to guitar inventory page	Pass
13	Go to part inventory page	2. Click “Guitar parts” button	Navigates to part inventory page	Navigates to part inventory page	Pass
14	Go to reports page	3. Click “Reports” button	Navigates to reports page	Navigates to reports page	Pass

Guitar Inventory Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
15	Go to guitar details page via “Add new guitar” button	1. Click “Add Guitar” button	Be redirected to Guitar Details Page	Redirected to Guitar Details Page	Pass
16	Go to guitar details page via “Update” button	1. Click the “Update” button in the action column from one of the sample guitar rows	Be redirected to the Guitar Details Page with the correct guitar	Redirected to Guitar Details page with the correct guitar	Pass

			information in the form	information in the form	
17	Delete a guitar	1. Click the “Delete” button in the action column in one of the guitar rows 2. Click okay when asked if you are sure you want to delete the product 3. Press “back” button on deletion confirmation page	Delete guitar from guitar inventory and be successfully redirected to guitars page upon deletion confirmation. The deleted guitar should not be displayed in the table.	Deleted guitar from guitar inventory and redirected to guitars page upon deletion. The guitar does not show up in the guitar inventory.	Pass
18	Place a guitar for order	1. Click “Place order” under action column in one of the guitar rows 2. Press “back” button upon order confirmation 3. Acknowledge a decrement in inventory for that guitar	User should be redirected to a confirmation screen, and back to the guitar inventory screen with the guitar inventory decremented by 1	Redirected to a confirmation screen and back to the guitar inventory screen with the guitar inventory decremented by 1	Pass
19	Try to place a guitar for order without enough inventory	1. Click “Place order” under action column in the “sample0” guitar row with the inventory of 0 2. Press “back” on the order failure screen	User should be redirected to an error screen stating not enough inventory. User should be redirected to guitar inventory screen upon pressing back button.	Redirected to an error screen stating not enough inventory. Redirected to guitar inventory screen upon pressing back button.	Pass
20	Navigate to homepage	1. Click “Home” button in top right corner	Be redirected to Home page	Redirected to homepage	Pass

21	Navigate to help page	1. Click “Help” button in top right corner	Be redirected to Help page	Redirected to Help page	Pass
22	Logout	1. Click “Logout” button in top right corner	Be redirected to Login page	Redirected to Login page	Pass
23	Search a valid guitar	1. Type “Electric” into search input field 2. Click “Search” button	Only guitars with the keyword “Electric” should be displayed	Only guitars with the keyword “Electric” are displayed	Pass
24	Search an invalid guitar	1. Type “Rockstar” into the search input field 2. Click “Search” button	No guitars should be displayed in the table	No guitars are displayed in the table	Pass
25	Clear search	3. After searching for keyword “Rockstar” click the “Clear” button	Page should reset with search bar blank and table back to all guitar inventory	Page reset with search bar blank and table back to all guitar inventory	Pass

Guitar Details Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
26	Enter correct guitar details	1. Enter “Guitar1”, “100.00”, “20” into the name, price and inventory fields 2. Click “Submit” button 3. Click “Back” button	User should be directed to a confirmation page upon submitting. After pressing back, the new guitar should be populated in the inventory table	Redirected to confirmation page upon submitting. New guitar is populated in the inventory table.	Pass
27	Add guitar with invalid inputs	1. Type “Guitar2”, “-20”, “inventory” into form fields	Fields should give an input error	Redirected to login/register error page.	Fail

		2. Click “Submit” button	prompting user to fix fields.		
28	Retest: Add guitar with invalid inputs	1. Type “Guitar2”, “-20”, “inventory” into form fields 2. Click “Submit” button	Fields should give an input error prompting user to fix fields.	Fields give an error message for invalid input fields	Pass
29	Add guitar with blank fields	1. Leave input fields blank 2. Click “Submit” button	User should be prompted to fill out all empty fields	Prompted to fill out all empty fields	Pass
30	Update a guitar	1. On guitar inventory page, click “Update” under the action column in one of the guitar rows 2. Type “Updated”, “100”, “100” into fields. 3. Click “Submit” button 4. Click “Back” button	Guitar should be updated to the newly input information displayed in the guitar inventory page.	Guitar information was update to the newly input information in the guitar inventory page.	Pass
31	Go to the home page	1. Click “Home” button in the top right corner	User should be redirected to the homepage	Redirected to the homepage	Pass
32	Go to the help page	1. Click “Help” button in the top right corner	User should be redirected to the help page	Redirected to the help page	Pass
33	Logout	1. Click “Logout” in the top right corner	User should be redirected to Login page	Redirected to login page	Pass

Part Inventory Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
34	Go to Inhouse Part Details Page	1. Click “Add Inhouse Part” button	User is redirected to Inhouse Part Details Page	Redirected to Inhouse Part Details Page	Pass

35	Go to Outsourced Part Details Page	1. Click “Add Outsourced Part” button	User is redirected to Outsourced Part Details Page	Redirected to Outsourced Part Details Page	Pass
36	Go to part details page upon updating	1. Click “Update” button under action column in one of the guitar rows	User is redirected to part details page with all details from guitar already input	Redirected to part details page with all details from guitar already input	Pass
37	Delete a part	1. Click “Delete” button under action column 2. Click “Ok” 3. Click “Back”	User is prompted to confirm deletion, user is redirected back to Part inventory page, Part is no longer in table	Prompted to confirm deletion, user is redirected back to Part inventory page, Part is no longer in table	Pass
38	Search for a valid part	1. Type “Sample” in search field 2. Click “Search” button	All parts with the keyword “Sample” should be displayed	All parts with the keyword “Sample” are displayed	Pass
39	Search for an invalid part	1. Type “none” in search field 2. Click “Search” button	No parts should show up in table	No parts shown in table	Pass
40	Clear search	1. Upon searching, click the “Clear” button	Table should reset with all parts shown in table	Table reset with all parts shown	Pass

Inhouse Part Details Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
41	Add new Inhouse	1. Type “Inhouse1”, 100, 100, 500, 50 into form fields	New inhouse part should be populated in	New inhouse part is added	Pass

	Part with valid input	2. Click “Submit” button 3. Click “Back” button	the part inventory table	into the part inventory table	
42	Add new Inhouse Part with invalid input	1. Type “Inhouse2”, “-100”, “-100”, “-500”, “-50” into form fields 2. Click “Submit”	Errors should show below each incorrect input	Correct errors are shown below each incorrect input	Pass
43	Add new Inhouse Part with blank fields	1. Leave input fields blank 2. Click “Submit” button	User should be prompted to fill out blank fields	Prompted to fill out blank fields	Pass
44	Update an Inhouse Part	1. On Part Inventory page, click “Update” button under action column next to a part 2. Type “Update” into name field 3. Click “Submit” button 4. Click “Back” button	Part should be updated with the name “Update”	Part is updated with the name “Update”	Pass
45	Check inventory is between maximum and minimum input validation	1. Type “sample” in name field 2. Type “100” in price field 3. Type 10 into the inventory field 4. Type 100 into the minimum inventory field 5. Type 500 into the maximum inventory field	Error should show stating inventory must be between max and min values	Error shows stating inventory value should be between max and min values	Pass
46	Check that maximum inventory value is greater than minimum	6. Type “sample” in name field 7. Type 100 in price field 8. Type 400 into the inventory field 9. Type 500 into the minimum inventory field	Error should show that the maximum inventory must be greater than the minimum value	No error was shown	Fail

		Type 100 into the maximum inventory field			
47	Retest: Check that maximum inventory value is greater than minimum	1. Type “sample” in name field 2. Type 100 in price field 3. Type 400 into the inventory field 4. Type 500 into the minimum inventory field 5. Type 100 into the maximum inventory field	Error should show that the maximum inventory must be greater than the minimum value	Error shows that the maximum value should be greater than the minimum value	Pass

Outsourced Part Details Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
48	Add new Outsourced Part with valid input	1. Type “OutsourcedPart”, 100, 100, 500, 50, “Company” into form fields 2. Click “Submit” button 3. Click “Back”	New outsourced part should be populated in part inventory table	New outsourced part is populated in the part inventory table	Pass
49	Add new Outsourced Part with invalid input	1. Type “OutsourcedPart” -100, -100, -500, -50, “Company” into form fields 2. Click “Submit”	Errors should show below invalid fields	Errors show below invalid fields	Pass
50	Add new Outsourced Part with blank fields	1. Leave form fields blank 2. Click “Submit”	User should be prompted to fill out empty fields	Prompted to fill out empty fields	Pass
51	Update an Outsourced Part	1. On Part Inventory Page click “Update” under action column in a part row 2. Update the name to “Update” and	User should be directed to an update confirmation page and then directed to the part inventory	Part is successfully updated with correct details	Pass

		the price to "5000.00" 3. Click "Submit" button 4. Click "Back" button	page with the part updated to the newly input details		
52	Check inventory between minimum and maximum input validation	1. Type "Part" into name field, "100" into price field, "10" into inventory field, "100" into minimum inv field, and "500" into maximum inv field 2. Click "Submit" button	Error should show that inventory must be between max and min inventory values	Error shows that inventory should be between max and min values	Pass
53	Check maximum and minimum inventory validation	1. Type "Part" into name field, "100" into price field, "10" into inventory field, "500" into minimum inv field, and "100" into maximum inv field 2. Click "Submit" button	Error should show that maximum value must be greater than the minimum value	Error shows that maximum value should be greater than minimum value	Pass

Reports Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
54	Go to guitar reports page	1. Click "Generate Guitar Report" Button	Go to guitar reports page	Redirects to guitar report page	Pass
55	Go to part reports page	1. Click "Generate Part Report" button	Go to part reports page	Redirects to part report page	Pass

Guitar Report Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
56	Current Date and Time	Identify Date and time in upper left corner of report	Date and time should be current	Date and time are current	Pass
57	Filter report based on searching valid keyword	1. Type keyword "Electric" into search input 2. Click "Search" Button	Only guitars with searched keyword should show up in report	Only guitars with searched keyword show up in report	Pass
58	Filter report based on searching invalid keyword	1. Type "none" into search input 2. Click "Search" button	No guitars should show up in report	No guitars show up in report	Pass
59	Clear Search	1. Click "Clear" button upon searching	Guitar report should reset to displaying all guitars	Guitar report resets to displaying all guitars	Pass
60	Print Report	1. Click "Print" button	Page should popup with printing details	Print details popup to print report	Pass

Parts Report Page:

Test Case:	Description:	Steps:	Expected Results:	Actual Results:	Pass or Fail:
61	Current Date and Time	Identify Date and time in upper left corner of report	Date and time should be current	Date and time are current	Pass

62	Filter report based on searching valid keyword	3. Type keyword “Part” into search input 4. Click “Search” Button	Only parts with searched keyword should show up in report	Only parts with searched keyword show up in report	Pass
63	Filter report based on searching invalid keyword	3. Type “none” into search input 4. Click “Search” button	No parts should show up in report	No parts show up in report	Pass
64	Clear Search	2. Click “Clear” button upon searching	Part report should reset to displaying all parts	Part report resets to displaying all Parts	Pass
65	Print Report	2. Click “Print” button	Page should popup with printing details	Print details popup to print report	Pass

Unit Testing:

Guitar Test:

```java
package com.D424webApp.D424webApp.domain;


import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

Aislyn Mildenhall
public class GuitarTest {
    23 usages
    Guitar guitar;
    Aislyn Mildenhall
    @BeforeEach
    public void setUp() {guitar = new Guitar();}

    Aislyn Mildenhall
    @Test
    void getId() {
        Long idValue=4L;
        guitar.setId(idValue);
        assertEquals(guitar.getId(), idValue);
    }

    Aislyn Mildenhall
    @Test
    void setId() {
        Long idValue=4L;
        guitar.setId(idValue);
        assertEquals(guitar.getId(), idValue);
    }
    Aislyn Mildenhall
    @Test
    void getName() {
        String name="Test guitar";
        guitar.setName(name);
```

```java
    @Test
    void setName() {
        String name="Test guitar";
        guitar.setName(name);
        assertEquals(name,guitar.getName());
    }
    Aislyn Mildenhall
    @Test
    void getPrice() {
        double price=1.0;
        guitar.setPrice(price);
        assertEquals(price,guitar.getPrice());
    }
    Aislyn Mildenhall
    @Test
    void setPrice() {
        double price=1.0;
        guitar.setPrice(price);
        assertEquals(price,guitar.getPrice());
    }
    Aislyn Mildenhall
    @Test
    void getInv() {
        int inv=5;
        guitar.setInv(inv);
        assertEquals(inv,guitar.getInv());
    }
    Aislyn Mildenhall
    @Test
    void setInv() {
        int inv=5;
        guitar.setInv(inv);
        assertEquals(inv,guitar.getInv());
    }
    Aislyn Mildenhall
```

```java
    void setInv() {
        int inv=5;
        guitar.setInv(inv);
        assertEquals(inv,guitar.getInv());
    }
    Aislyn Mildenhall
    @Test
    void testToString() {
        String name = "Test Guitar";
        guitar.setName(name);
        assertEquals(name,guitar.toString());
    }
    Aislyn Mildenhall
    @Test
    void testEquals(){
        guitar.setId(1L);
        Guitar newGuitar = new Guitar();
        newGuitar.setId(1L);
        assertEquals(guitar,newGuitar);
    }
    Aislyn Mildenhall
    @Test
    void testHashCode(){
        guitar.setId(1L);
        Guitar newGuitar = new Guitar();
        newGuitar.setId(1L);
        assertEquals(guitar.hashCode(), newGuitar.hashCode());
    }
}
```

Inhouse Part Test:

```java
package com.D424webApp.D424webApp.domain;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assert

Aislyn Mildenhall
public class InhousePartTest {
    5 usages
    InhousePart ip;
    Aislyn Mildenhall
    @BeforeEach
    void setUp() { ip= new InhousePart(); }

    Aislyn Mildenhall
    @Test
    void getPartId() {
        int idValue=4;
        ip.setPartId(idValue);
        assertEquals(ip.getPartId(), idValue);
    }

    Aislyn Mildenhall
    @Test
    void setPartId() {
        int idValue=4;
        ip.setPartId(idValue);
        assertEquals(ip.getPartId(), idValue);
    }
}
```

Outsourced Part Test:

```java
package com.D424webApp.D424webApp.domain;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertE

Aislyn Mildenhall
public class OutsourcedPartTest {
    5 usages
    OutsourcedPart op;

    Aislyn Mildenhall
    @BeforeEach
    void setUp() { op= new OutsourcedPart(); }

    Aislyn Mildenhall
    @Test
    void getCompanyName() {
        String name="test company name";
        op.setCompanyName(name);
        assertEquals(name,op.getCompanyName());
    }

    Aislyn Mildenhall
    @Test
    void setCompanyName() {
        String name="test company name";
        op.setCompanyName(name);
        assertEquals(name,op.getCompanyName());
    }
}
```

Part Test:

```java
package com.D424webApp.D424webApp.domain;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEq

Aislyn Mildenhall
public class PartTest {
    31 usages
    Part partIn;
    31 usages
    Part partOut;
    Aislyn Mildenhall
    @BeforeEach
    void setUp() {
        partIn=new InhousePart();
        partOut=new OutsourcedPart();
    }
    Aislyn Mildenhall
    @Test
    void getId() {
        Long idValue=4L;
        partIn.setId(idValue);
        assertEquals(partIn.getId(), idValue);
        partOut.setId(idValue);
        assertEquals(partOut.getId(), idValue);
    }

    Aislyn Mildenhall
    @Test
    void setId() {
        Long idValue=4L;
        partIn.setId(idValue);
        assertEquals(partIn.getId(), idValue);
        partOut.setId(idValue);
        assertEquals(partOut.getId(), idValue);
    }

    Aislyn Mildenhall
    @Test
    void getName() {
        String name="test inhouse part";
        partIn.setName(name);
        assertEquals(name,partIn.getName());
        name="test outsourced part";
        partOut.setName(name);
        assertEquals(name,partOut.getName());
    }

    Aislyn Mildenhall
    @Test
    void setName() {
        String name="test inhouse part";
        partIn.setName(name);
        assertEquals(name,partIn.getName());
        name="test outsourced part";
        partOut.setName(name);
        assertEquals(name,partOut.getName());
    }

    Aislyn Mildenhall
    @Test
    void getPrice() {
        double price=1.0;
        partIn.setPrice(price);
        assertEquals(price,partIn.getPrice());
        partOut.setPrice(price);
        assertEquals(price,partOut.getPrice());
    }

    @Test
    void setPrice() {
        double price=1.0;
        partIn.setPrice(price);
        assertEquals(price,partIn.getPrice());
        partOut.setPrice(price);
        assertEquals(price,partOut.getPrice());
    }

    Aislyn Mildenhall
    @Test
    void getInv() {
        int inv=5;
        partIn.setInv(inv);
        assertEquals(inv,partIn.getInv());
        partOut.setInv(inv);
        assertEquals(inv,partOut.getInv());
    }

    Aislyn Mildenhall
    @Test
    void setInv() {
        int inv=5;
        partIn.setInv(inv);
        assertEquals(inv,partIn.getInv());
        partOut.setInv(inv);
        assertEquals(inv,partOut.getInv());
    }
    Aislyn Mildenhall
    @Test
    void testToString() {
        String name="test inhouse part";
        partIn.setName(name);
        assertEquals(name,partIn.toString());
        name="test outsourced part";
```

```java
    @Test
    void testEquals() {
        partIn.setId(1l);
        Part newPartIn=new InhousePart();
        newPartIn.setId(1l);
        assertEquals(partIn,newPartIn);
        partOut.setId(1l);
        Part newPartOut=new OutsourcedPart();
        newPartOut.setId(1l);
        assertEquals(partOut,newPartOut);

    }

    Aislyn Mildenhall
    @Test
    void testHashCode() {
        partIn.setId(1l);
        partOut.setId(1l);
        assertEquals(partIn.hashCode(),partOut.hashCode());
    }

    Aislyn Mildenhall
    @Test
    void getMinInv() {
        int minInv=3;
        partIn.setMinInv(minInv);
        assertEquals(minInv,partIn.getMinInv());
        partOut.setMinInv(minInv);
        assertEquals(minInv,partOut.getMinInv());
    }

    Aislyn Mildenhall
    @Test
    void setMinInv() {
        int minInv=3;
```

```java
    Aislyn Mildenhall
    @Test
    void setMinInv() {
        int minInv=3;
        partIn.setMinInv(minInv);
        assertEquals(minInv,partIn.getMinInv());
        partOut.setMinInv(minInv);
        assertEquals(minInv,partOut.getMinInv());
    }
    Aislyn Mildenhall
    @Test
    void getMaxInv() {
        int maxInv=3;
        partIn.setMaxInv(maxInv);
        assertEquals(maxInv,partIn.getMaxInv());
        partOut.setMaxInv(maxInv);
        assertEquals(maxInv,partOut.getMaxInv());
    }
    Aislyn Mildenhall
    @Test
    void setMaxInv() {
        int maxInv=3;
        partIn.setMaxInv(maxInv);
        assertEquals(maxInv,partIn.getMaxInv());
        partOut.setMaxInv(maxInv);
        assertEquals(maxInv,partOut.getMaxInv());
    }
```

Users Test:

```java
package com.D424webApp.D424webApp.domain;

import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

Aislyn Mildenhall
public class UsersTest {
    19 usages
    Users user;

    Aislyn Mildenhall
    @BeforeEach
    void setUp() { user=new Users(); }

    Aislyn Mildenhall
    @Test
    void getId() {
        Long idValue=4L;
        user.setId(idValue);
        assertEquals(user.getId(), idValue);
    }

    Aislyn Mildenhall
    @Test
    void setId() {
        Long idValue=4L;
        user.setId(idValue);
        assertEquals(user.getId(), idValue);
    }
```

```java
    @Test
    void getUsername() {
        String username="test username";
        user.setUsername(username);
        assertEquals(username,user.getUsername());
    }

    Aislyn Mildenhall
    @Test
    void setUsername() {
        String username="test username";
        user.setUsername(username);
        assertEquals(username,user.getUsername());
    }

    Aislyn Mildenhall
    @Test
    void getPassword() {
        String password="test password";
        user.setPassword(password);
        assertEquals(password,user.getPassword());
    }

    Aislyn Mildenhall
    @Test
    void setPassword() {
        String password="test password";
        user.setPassword(password);
        assertEquals(password,user.getPassword());
    }
```

```java
    Aislyn Mildenhall
    @Test
    void testEquals() {
        user.setId(1l);
        Users newUser=new Users();
        newUser.setId(1l);
        assertEquals(user,newUser);
    }

    Aislyn Mildenhall
    @Test
    void testHashCode() {
        user.setId(1l);
        Users newUser=new Users();
        newUser.setId(1l);
        assertEquals(user.hashCode(),newUser.hashCode());
    }

    Aislyn Mildenhall
    @Test
    void testToString() {
        String username="test user";
        user.setUsername(username);
        assertEquals(username,user.toString());
    }
}
```

Guitar Repository Test:

```java
package com.D424webApp.D424webApp.repository;

import ...

Aislyn Mildenhall
class GuitarRepositoryTest {

    3 usages
    GuitarRepository guitarRepository;
    Aislyn Mildenhall
    @BeforeEach
    void setUp() { guitarRepository=mock(GuitarRepository.class); }

    Aislyn Mildenhall
    @Test
    void findAll() {
        Guitar guitar = new Guitar();
        List guitarData = new ArrayList();
        guitarData.add(guitar);
        when(guitarRepository.findAll()).thenReturn(guitarData);
        List<Guitar> guitars = (List<Guitar>) guitarRepository.findAll();
        assertEquals(guitarData.size(), actual: 1);
    }
    }
```

Inhouse Repository Test:

```java
Aislyn Mildenhall
public class InhousePartRepositoryTest {
    3 usages
    InhousePartRepository inhousePartRepository;
    Aislyn Mildenhall
    @BeforeEach
    void setUp() { inhousePartRepository=mock(InhousePartRepository.class); }
    Aislyn Mildenhall
    @Test
    void findAll() {
        InhousePart part=new InhousePart();
        List partData=new ArrayList();
        partData.add(part);
        when(inhousePartRepository.findAll()).thenReturn(partData);
        List<InhousePart> parts=(List<InhousePart>)inhousePartRepository.findAll();
        assertEquals(partData.size(), actual: 1);
    }
}
```

Outsourced Repository Test:

```
class OutsourcedPartRepositoryTest {

    3 usages
    OutsourcedPartRepository outsourcedPartRepository;
    Aislyn Mildenhall
    @BeforeEach
    void setUp() { outsourcedPartRepository=mock(OutsourcedPartRepository.class); }

    Aislyn Mildenhall
    @Test
    void findAll() {
        OutsourcedPart part=new OutsourcedPart();
        List partData=new ArrayList();
        partData.add(part);
        when(outsourcedPartRepository.findAll()).thenReturn(partData);
        List<OutsourcedPart> parts=(List<OutsourcedPart>)outsourcedPartRepository.findAll();
        assertEquals(partData.size(), actual: 1);
    }
}
```

Parts Repository Test:

```
class PartRepositoryTest {

    3 usages
    PartRepository partRepository;
    Aislyn Mildenhall
    @BeforeEach
    void setUp() { partRepository=mock(PartRepository.class); }

    Aislyn Mildenhall
    @Test
    void findAll() {
        InhousePart part=new InhousePart();
        List partData=new ArrayList();
        partData.add(part);
        when(partRepository.findAll()).thenReturn(partData);
        List<Part> parts=(List<Part>)partRepository.findAll();
        assertEquals(partData.size(), actual: 1);
    }


}
```

Users Repository Test:

```
@Test
public void testFindByUsernameAndPassword() {
    // Given
    Users user = new Users( username: "testUser",  password: "password123");
    usersRepository.save(user);

    // When
    Optional<Users> foundUser = usersRepository.findByUsernameAndPassword( username: "testUser",  password: "password123");

    // Then
    assertTrue(foundUser.isPresent()); // Expecting a user to be found
    assertEquals( expected: "testUser", foundUser.get().getUsername());
    assertEquals( expected: "password123", foundUser.get().getPassword());
}

Aislyn Mildenhall
@Test
public void testFindByUsernameAndPassword_NotFound() {
    // When
    Optional<Users> notFoundUser = usersRepository.findByUsernameAndPassword( username: "nonExistentUser",  password: "wrongP

    // Then
    assertTrue(notFoundUser.isEmpty()); // Expecting no user to be found
}
```

Results

The manual testing results are displayed in the above test cases. Displayed below, are the unit test results.

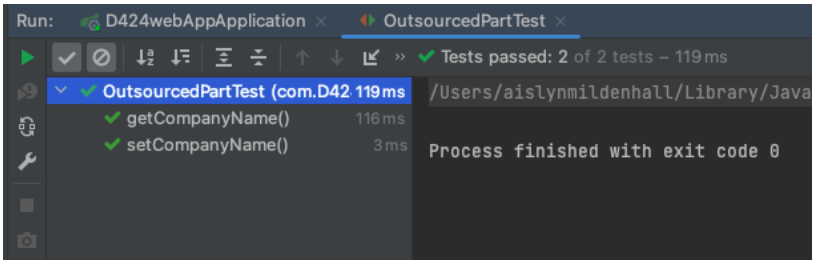
Guitar Test Results:

```
Run:  D424webAppApplication  GuitarTest
Tests passed: 11 of 11 tests – 68 ms
GuitarTest (com.D424webApp.[ 68 ms        /Users/aislynmildenhall/Library/Java
    getInv()                 45 ms
    testToString()            5 ms        Process finished with exit code 0
    setInv()                  2 ms
    getName()                 1 ms
    getId()                   2 ms
    setId()                   3 ms
    testHashCode()            2 ms
    testEquals()              2 ms
    setPrice()                2 ms
    getPrice()                2 ms
    setName()                 2 ms
```

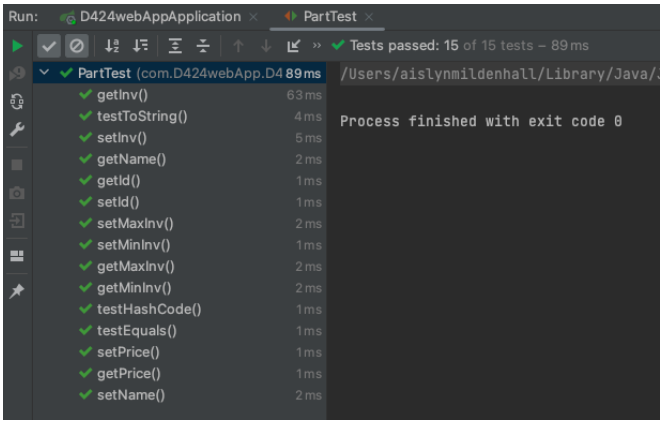
Inhouse Part Test Results:

```
Run:  D424webAppApplication  InhousePartTest
Tests passed: 2 of 2 tests – 48 ms
InhousePartTest (com.D424wel 48 ms        /Users/aislynmildenhall/Library/Java/
    setPartId()              46 ms
    getPartId()               2 ms        Process finished with exit code 0
```

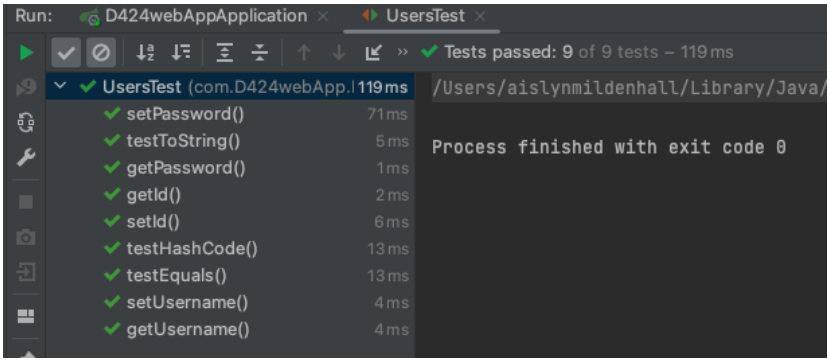
Outsourced Part Test Results:



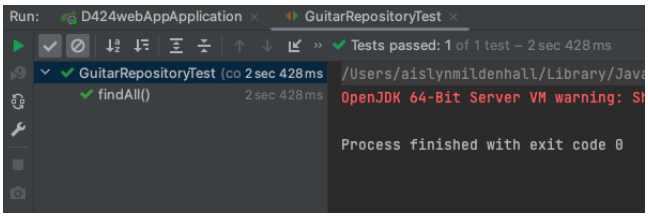
Part Test Results:



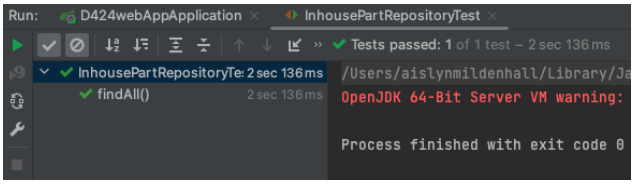
Users Test Results:



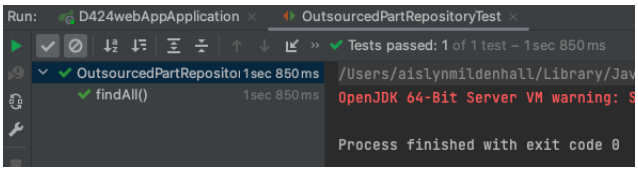
Guitar Repository Test Results:



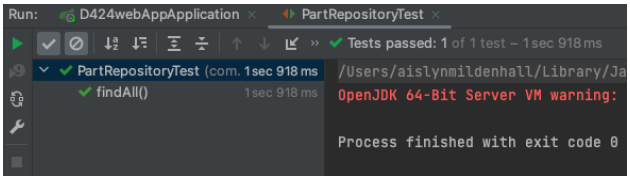
Inhouse Repository Test Results:



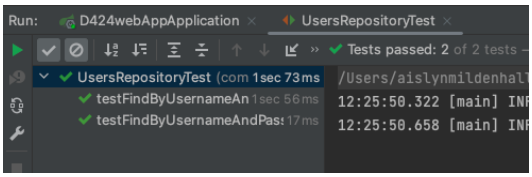
Outsourced Repository Test Results:



Parts Repository Test Results:



Users Repository Test Results:



Summary:

All unit tests passed and no remediation was necessary. Two manual tests failed, they were subsequently retested and documented within the manual test cases.

GitLab Repository and Branch History:

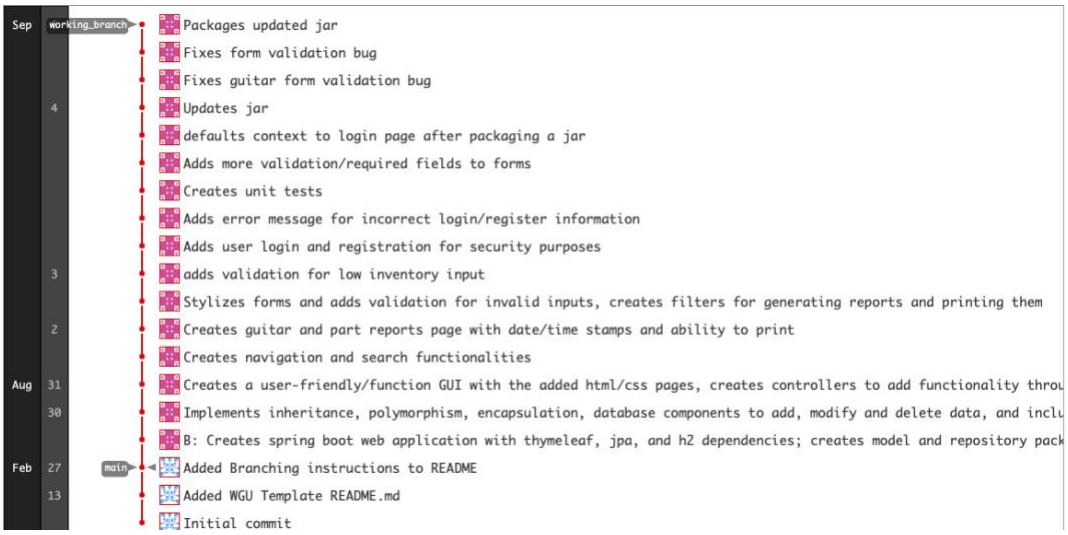
Link to Gitlab Repository:

https://gitlab.com/wgu-gitlab-environment/student-repos/amilde3/d424-software-engineering-capstone.git

Panopto video link:

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=6f8abc42-59ef-462a-bcc2-b075016ad973

Branch History:



User Guide

Introduction

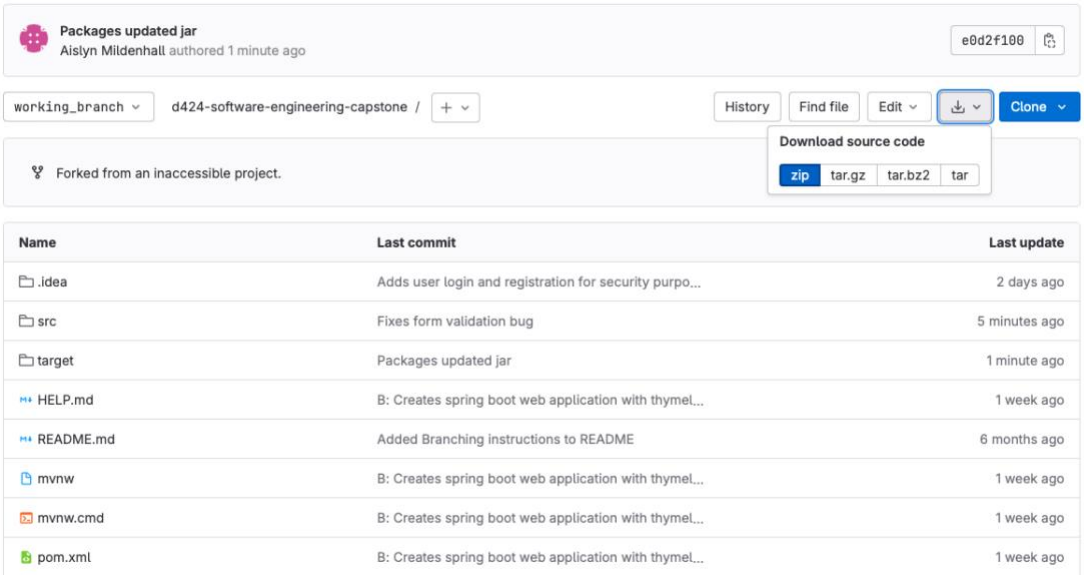
The following user guide includes instructions on how to set up and install the application. After setting up and installing the application, you can access the application with the following user guide that holds instructions for running the application.

Set-Up and Installation

Note: Before downloading and running the jar file-you need to have Java 17 installed on your device.

1. Download the zip file from GitLab:

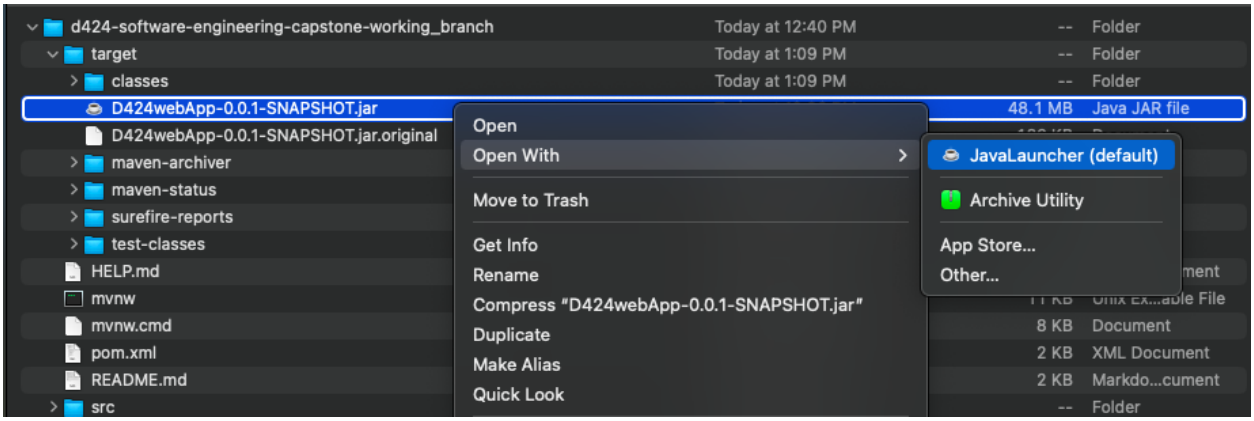
   -Make sure the selected branch is the working_branch



2. Once the zip file is downloaded, locate the zip file in your files and extract the files by double clicking the zip file on a Mac or clicking “Extract All” on Windows.
3. Navigate to the folder called “d424-software-engineering-capstone-working_branch”, open up the folder and find the “target” folder, then find the jar file called “D424webApp-0.0.1-SNAPSHOT.jar”

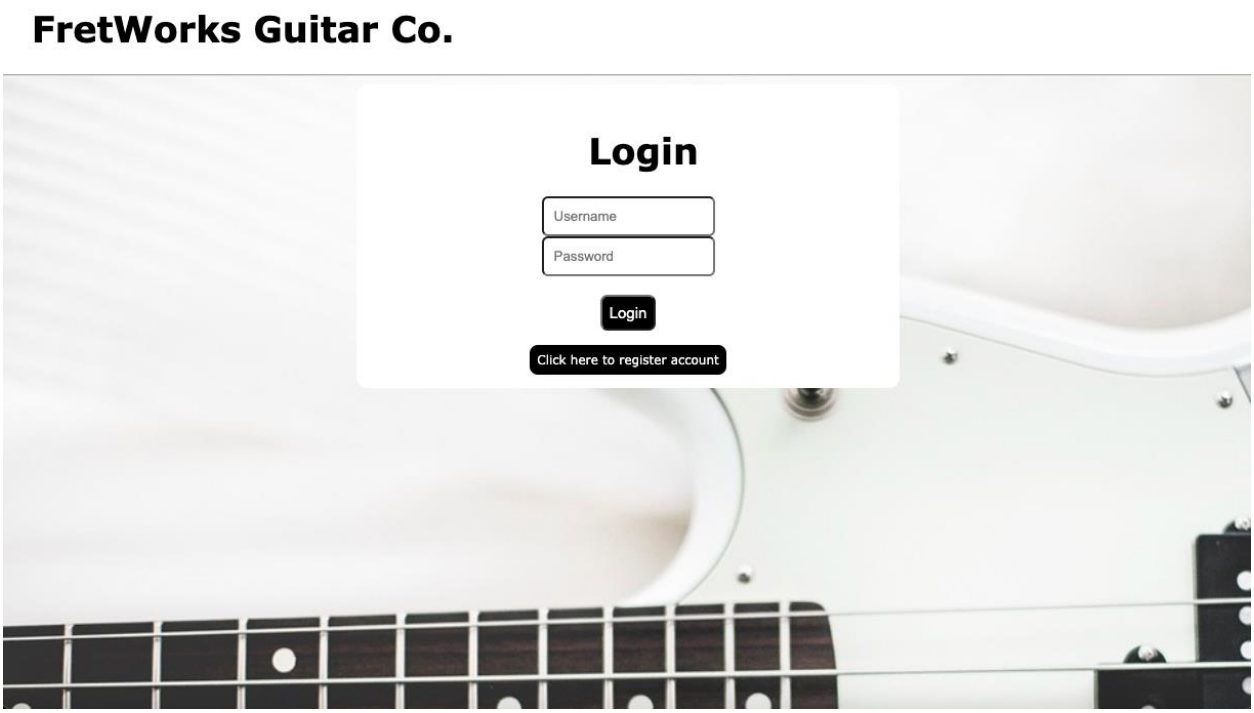
4. Right click on the jar file and select “Open with”

5. Select JavaLauncher on Mac or Java Platform SE on Windows.



6. Open up a browser and type localhost:8080

7. You should then be presented with the login page



Running and using the application

Login and Register

1. *Click the "Click here to register account" button on the bottom of the login form.*

2. *Type "admin" into the username and "admin123!" into the password*
3. *Click the "Register" button under the register form*

You will then be redirected to the login screen to log in with the account you just made.

4. *Type "admin" into the username and "admin123!" into the password.*
5. *Click the "Login" button under the login form.*

You will then be redirected to the homepage.

Get Help

The help screen is available from any screen throughout the website, outside of the login and registration pages. If at any point you need to access the IT help desk contact information, click the "Help" button in the upper right-hand corner.

Homepage

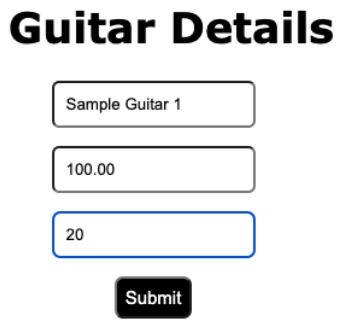
1. *Click the "Guitars" button to go to the Guitar Inventory Page*

Guitars

Create a New Guitar

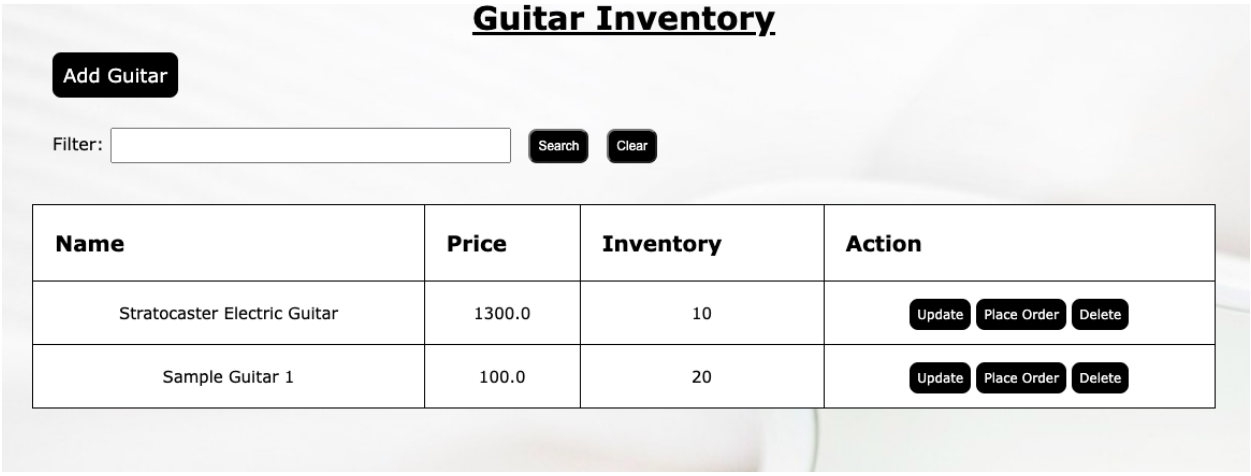
1. *Click the "Add Guitar" button in the top left corner of the screen. This will take you to the Guitar details screen where you can add details for the new guitar.*

2. *Enter a guitar name, price, and inventory number, then click "Submit", otherwise, click "Back" in the top left corner to go back to the Guitar Inventory Screen.*



3. *You will be notified that the guitar has been added. Click the "Back" button to go back to the guitar inventory page.*

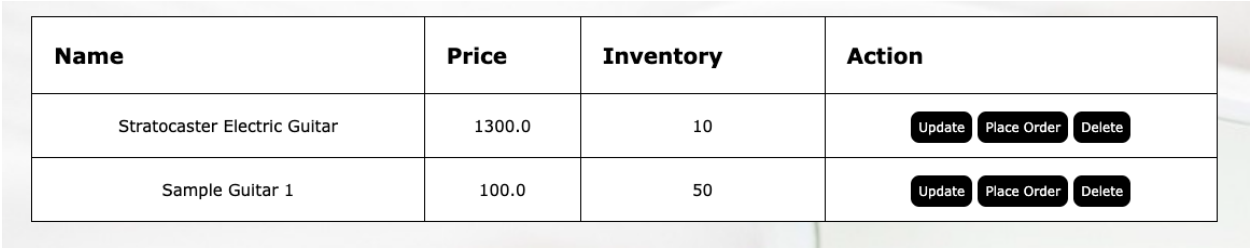
4. *Your new guitar should be populated in the table along with the sample guitar "Stratocaster Electric Guitar".*



Update a Guitar

1. *Click "Update" to update the inventory of this guitar. This will take you back to the guitar details form with the details populated into the form. Change the Inventory value to 50. Click "Submit".*

You should then be notified that the guitar was updated and click the "Back" button to go back to the guitar inventory screen. Note that the guitar inventory was updated to 50.

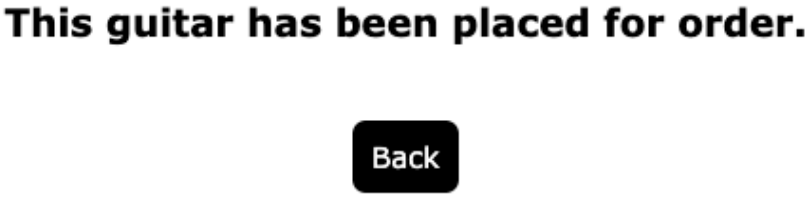


Name	Price	Inventory	Action
Stratocaster Electric Guitar	1300.0	10	Update Place Order Delete
Sample Guitar 1	100.0	50	Update Place Order Delete

Place a guitar for order

When a customer buys a guitar, you may place that guitar for order to track the inventory.

1. *Do this by clicking the "Place Order" button in the row of the Sample Guitar 1 to place it for order. You will then be redirected to a confirmation if the order can be placed.*

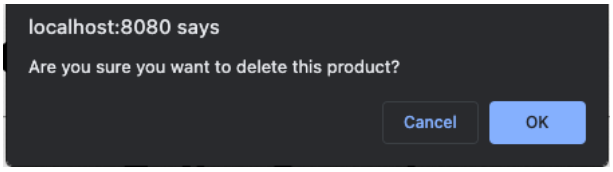


The order tracking is successful because the inventory number decreased:

Name	Price	Inventory	Action
Stratocaster Electric Guitar	1300.0	10	Update Place Order Delete
Sample Guitar 1	100.0	49	Update Place Order Delete

Delete an inventory item

1. *Click the "Delete" button in the Sample Guitar 1 row*
2. *You will be prompted to confirm the deletion. Click "OK"*



3. *A confirmation that the guitar was deleted will pop up. Click "Back" to go back to the guitar inventory page.*

The guitar has been successfully deleted.

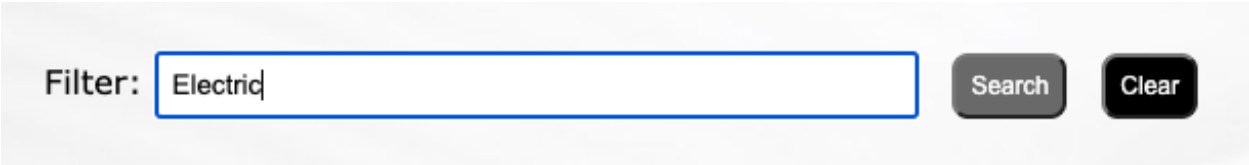
Back

Upon going back, we can see the Sample Guitar 1 has been deleted.

Name	Price	Inventory	Action
Stratocaster Electric Guitar	1300.0	10	Update Place Order Delete

Search for a guitar

1. *To filter guitars based on keywords, type the keyword in the search field and click the "Search" button.*



All guitars with the keyword in the name will appear in the table, otherwise, no data will appear in the table. If you search the keyword "none", no guitars will show up.

2. *Click the "Clear" button to reset the search. The search will be reset and all guitars will reappear in the guitar inventory table.*

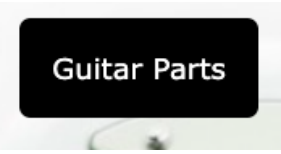
Navigate to Homepage

1. *Click the "Home" button in the top right corner*

   This will take you back to the homepage so you can access the Parts and Reports pages.

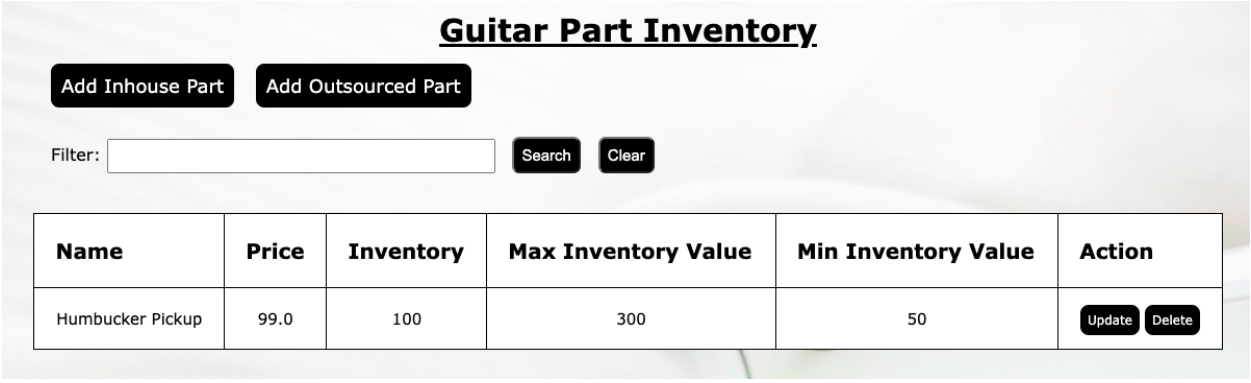
Guitar Parts

1. *On the homepage, click on the middle button "Guitar Parts" to navigate to the guitar parts inventory page.*

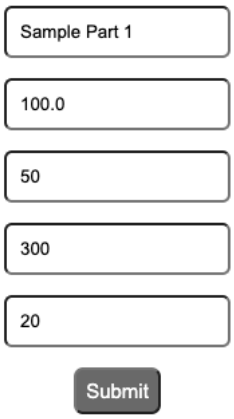


Add an Inhouse Part

1. *On this page, you have similar actions to the guitar inventory page. Click the "Add Inhouse Part" button in the top right to add an inhouse part.*



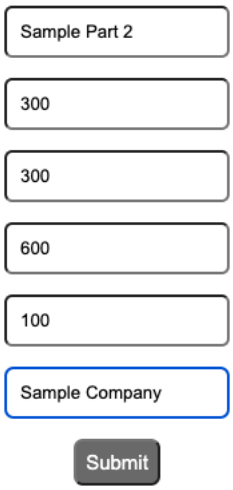
2. *Type a part Name, Price, Inventory value, maximum inventory value, and minimum inventory value. Note that the maximum inventory value must be greater than the minimum inventory value, and the inventory value must be within the maximum and minimum range. If not, you will be prompted with an error. Click "Submit"*



Your new part should be confirmed and updated in the part inventory table.

Add an Outsourced Part

1. *Follow the same steps for adding an outsourced part as you did for the inhouse part. The only difference is you may add a company to the outsourced part details form. Note that the company name is not required.*



Search for Parts

1. *Search for a part the same way you did for a guitar. Type a keyword like "Sample" into the search field and click the "Search" button. All parts with the keyword "Sample" will show up in the table. Otherwise, no parts will show up.*
2. *Click "Clear" to clear the search and reset the table to show all parts.*

Update and Delete Parts

1. *Update and delete parts the same way you did with the guitars.*

Navigate back to Homepage

1. *Click the "Home" button in the top right corner to navigate back to the homepage.*

Reports

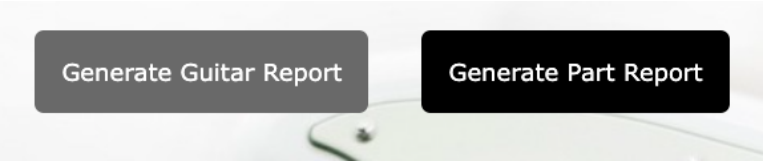
1. *To access the reports, from the homepage click the "Reports" button.*

Here, you can select to generate a guitar report or generate a part report

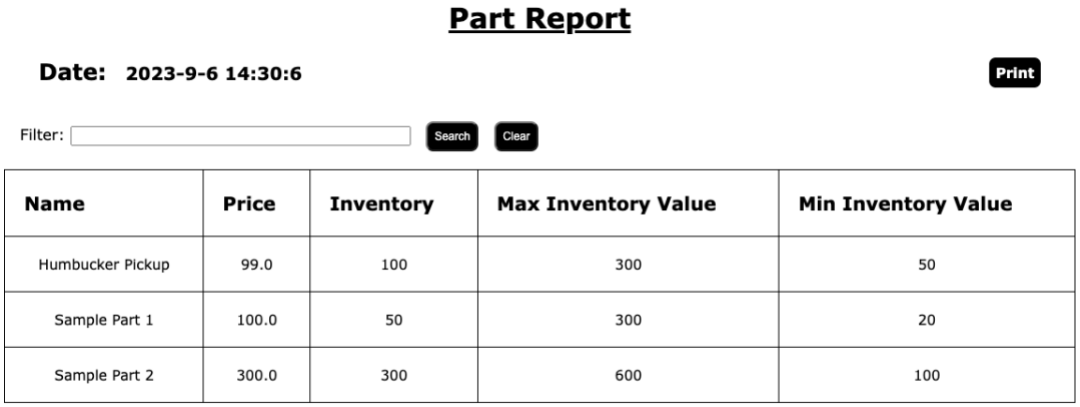
Generate a report

Generating a guitar and part report involve the same process. Demonstrated below are the steps to generate a part report. Generating a guitar report will require the exact same steps.

1. *Depending on which report you want to generate, click the corresponding button.*



2. *You will be redirected to the report page which displays the current date and time of the report.*



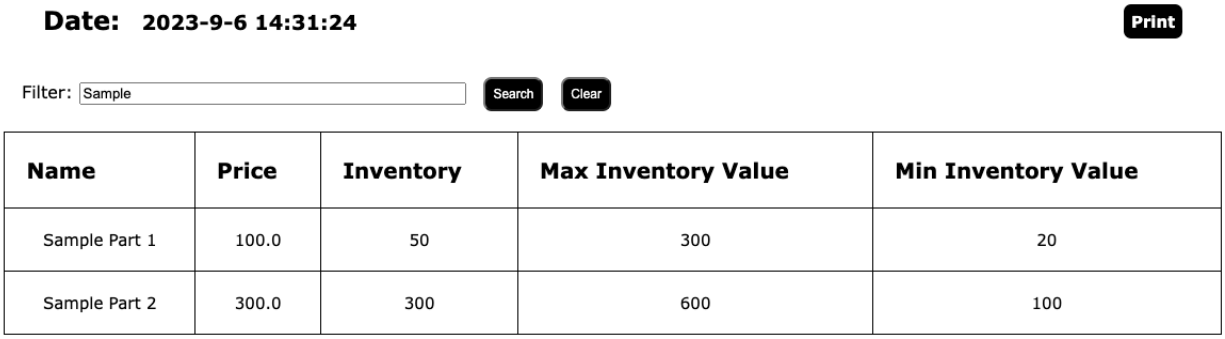
Part Report

Date: **2023-9-6 14:30:6** Print

Filter: Search Clear

Name	Price	Inventory	Max Inventory Value	Min Inventory Value
Humbucker Pickup	99.0	100	300	50
Sample Part 1	100.0	50	300	20
Sample Part 2	300.0	300	600	100

3. *Filter the report by typing a keyword into the search field and clicking the "search" button. Here, we are creating a report for the "Sample" guitars.*

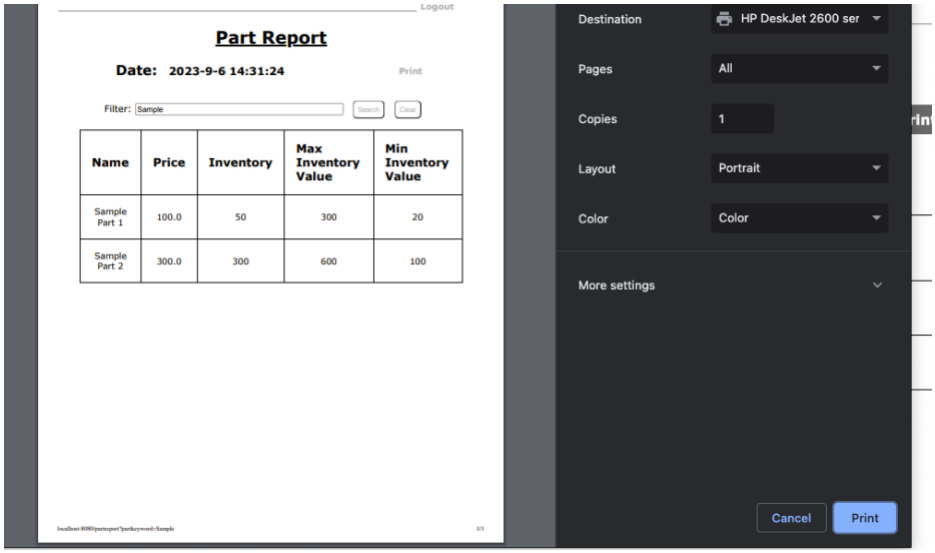


Date: **2023-9-6 14:31:24** Print

Filter: Sample Search Clear

Name	Price	Inventory	Max Inventory Value	Min Inventory Value
Sample Part 1	100.0	50	300	20
Sample Part 2	300.0	300	600	100

4. *Print the report by clicking the print button in the top right corner.*
5. *Your web browser should then load the print details for this report.*



6. *Clear the filter by clicking the "Clear" button next to the "search" button. This will reset your report to include all products in the report.*

Logout

You may logout from any screen within the web application. Click the "Logout" button in the top right corner of the application. You will then be redirected to the login screen.

https://d424capstone.onrender.com/

No sources were used in the creation of this document.