InventoryPro

Requirements and Test Document

Aiden McCollum

March 28, 2025

CS 225, Spring 2025

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

INTRODUCTION:

InventoryPro is an inventory management application specializing in tracking, updating, and creating items for a warehouse or business. InventoryPro allows employees using the app to log into the program, create and edit items as they undergo the shipping and processing workflows, assign projects for tracking items associated with organizational objectives, and sort items into locations for easier finding in the future. This program allows warehouses and big box stores to optimize their operations and reduce lost or misplaced items.

BACKGROUND INFORMATION:

In this application, items can be one of two different types. An item can either be a product, which is an item sold by the company, or an expense, which is an item consumed by the company to execute their objectives. Additionally, projects are simply used as a method for storing items in a coherent way were they can be assigned to a person of contact. This allows users to track items that they may hold responsibility for. Locations can be considered a storage container for specific items. This helps bridge the gap between the physical and digital layout of the warehouse or business when it comes to inventory management. To properly control access to the program, users can log in and/or sign up. This helps the program provide information that should only be accessible or pertinent to the current user. The development of this application has been segmented into ten different user stories. For user story #1, no requirements could be developed for the user story. This is because the user story focuses on laying out the code framework of the program, which the user should never be able to see or verifiably test without fully flushed out methods.

REQUIREMENTS:

This section contains the requirements. Remember that requirements must be <u>unambiguous</u> (only interpreted in one way) and <u>testable</u> (able to be objectively measured via test). The section starts with a short description of the purpose of the section (presenting the requirements), and any other additional text that helps the reader understand what the purpose of the requirements are. You must have **one requirement per user story**. You may have more, but your grade is based on one per user story. The requirements shall be numbered or identified in some fashion. Each requirement shall be associated with the user story, or stories, that it relates to. Formats may vary; a recommended format is provided in Table 1. Expand Table 1 as needed as you record user stories and requirements during the software development process. If a user story cannot be tested, explain that here.

Table 1: Requirement Specifications

ID	Requirement Specification
1	As a developer, I want to create a basic layout of my program.
	No requirement due to lack of testability.
2	As a user, I want to be able to log into the platform or create an account if I don't already have an account.
	1.1: User shall only be able to access the program's main menu if they provide a valid employee ID value and a matching password value. All other incorrect or invalid entries should deny access to the program. 1.2: User shall be able to successfully create an account by providing valid values for the Employee ID, name, and password fields during the sign-up prompting process.
3	As a developer, I want to provide the user with a menu of options to perform.
	2.1: The program shall output a menu with the options as defined in the Software Design Document with the ability for a user to provide the correlating number to execute that option. If an invalid input is provided, the program should prompt the user to re-enter a valid input to select an option from the menu.
4	As a user, I want to be able to create an item with the required information.
	3.1 The user shall be able to create either a product item or an expense item with the required values being prompted to the user appropriately and then stored in the database.
5	As a user, I want to be able to search for and view an item within the program.
	4.1 The user shall be able to enter any valid item ID and returned with the item's specific information. If the Item ID is invalid, the user should be re-prompted to enter a new ID.

*[Shall be completed **for user stories actively worked in the current sprint**. Add rows as needed.]*

TEST CASES:

This section contains the actual test cases. **Students are required to test one requirement per user story**. **Each test shall be composed of a minimum of three test cases (so at least 3 test cases per user story)**. Students having additional requirements for a user story are not required to provide tests cases for the additional requirements. Test cases shall be constructed with specific values for input and expected behavior.

Some user stories may not require test cases. No requirement or test case is needed for that user story. **If a requirement cannot be written for a user story, or the user story does not result in testable software, justify why there is no requirement or test case in the background section** of this document. For example, a user story that involves doing background research prior to writing software does not need a requirement and cannot be tested.

User Table 1 to track what test cases are associated with which requirements.

Use Table 2 for the format of test cases. Naturally, the actual behavior and pass/fail columns don't get filled out until the tests are actually performed. Introductory text shall explain the table (you should never include a table or figure in a document without discussing it and referring to it).

Table 1: Test Case Summary

User Story ID	Requirement ID	Test Case ID	Date	Status Pass/Fail/Pending
2	1.1	1A	3/28	Pass
2	1.1	1B	3/28	Pass
2	1.1	1C	3/28	Pass
3	2.1	2A	3/28	Pass
3	2.1	2B	3/28	Pass
3	2.1	2C	3/28	Pass

Table 2: Test Case Results

Test Case ID: 1A	**Current Status: Pass**		**Date: 3/28**
Req. ID: 1.1 User shall only be able to access the program's main menu if they provide a valid employee ID value and a matching password value. All other incorrect or invalid entries should deny access to the program.			
Test Description: The following testing case verifies that the user can properly access the program with a valid employee ID and password.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	User shall run the InventoryPro.java file to start the program.	System will output a message welcoming the user to the program and ask whether they would like to (1) log in or (2) sign up.	
2	User shall select to log in to the program by inputting 1 into the prompt.	The program will prompt the user to enter their employee ID.	

3	The user shall provide a valid employee ID, such as Emp1018.	The program will prompt the user for a password.	
4	The user shall provide a valid password, such as cheeseFries1!	The program will welcome the user and display the menu. Program will continue running as normally expected.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
Welcome! Would you like to (1) log in or (2) sign up: 1
Enter Employee ID: Emp1018
Enter Password: cheeseFries1!



Welcome back, John Smith!
```

Test Case ID: 1B	**Current Status: Pass**	**Date: 3/28**

Req. ID: 1.1 User shall only be able to access the program's main menu if they provide a valid employee ID value and a matching password value. All other incorrect or invalid entries should deny access to the program.

Test Description: The following testing case verifies that the user cannot access the program with a valid employee ID but not a valid password.

Step #	Operator Action	Expected Results	Comments
1	User shall run the InventoryPro.java file to start the program.	System will output a message welcoming the user to the program and ask whether they would like to (1) log in or (2) sign up.	
2	User shall select to log in to the program by inputting 1 into the prompt.	The program will prompt the user to enter their employee ID.	
3	The user shall provide a valid employee ID, such as Emp1018.	The program will prompt the user for a password.	
4	The user shall provide an invalid password, such as chiliFries52$	The program will alert the user that authentication could not be performed and the program will halt running	

Screenshots:

```
amccollum@Aidens-MacBook-Pro final_project % java InventoryPro

=== InventoryPro: Inventory Management System ===
Welcome! Would you like to (1) log in or (2) sign up: 1
Enter Employee ID: Emp1018
Enter Password: chiliFries52$
failed to authenticate.
```

Test Case ID: 1C	**Current Status: Pass**		**Date: 3/28**
Req. ID: 1.1 User shall only be able to access the program's main menu if they provide a valid employee ID value and a matching password value. All other incorrect or invalid entries should deny access to the program.			
Test Description: The following testing case verifies that the user cannot access the program with a valid employee ID and a valid password, but the two do not match the same user profile.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	User shall run the InventoryPro.java file to start the program.	System will output a message welcoming the user to the program and ask whether they would like to (1) log in or (2) sign up.	
2	User shall select to log in to the program by inputting 1 into the prompt.	The program will prompt the user to enter their employee ID.	
3	The user shall provide a employee ID, such as Emp6061.	The program will prompt the user for a password.	
4	The user shall provide a valid password, such as cheeseFries1! (which is linked to a different user account than the provided employee ID)	The program will alert the user that authentication could not be performed and the program will halt running	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
Welcome! Would you like to (1) log in or (2) sign up: 1
Enter Employee ID: Emp6061
Enter Password: cheeseFries1!
failed to authenticate.
```

Test Case ID: 2A	Current Status: Pass		Date: 3/28
Req. ID: 2.1 The program shall output a menu with the options as defined in the Software Design Document with the ability for a user to provide the correlating number to execute that option. If an invalid input is provided, the program should prompt the user to re-enter a valid input to select an option from the menu.			
Test Description: The following testing case verifies that the user can see the menu and enter a valid number to select option.			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	User shall run the InventoryPro.java file to start the program.	System will output a message welcoming the user to the program and ask whether they would like to (1) log in or (2) sign up.	
2	User shall select to log in to the program by inputting 1 into the prompt.	The program will prompt the user to enter their employee ID.	
3	The user shall provide a valid employee ID, such as Emp1018.	The program will prompt the user for a password.	
4	The user shall provide a valid password, such as cheeseFries1!	The program will welcome the user and display the menu.	
5	User should select option 1 by entering the number 1.	The program will print a message saying “option 1 selected”	Note that this would usually execute the actual task, but that code has not been written yet
Screenshots:			

```
=== InventoryPro: Inventory Management System ===
Welcome! Would you like to (1) log in or (2) sign up: 1
Enter Employee ID: Emp1018
Enter Password: cheeseFries1!



Welcome back, John Smith!

=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 1
Selected Choice 1
```

Test Case ID: 2B	Current Status: Pass	Date: 3/28

Req. ID: 2.1 The program shall output a menu with the options as defined in the Software Design Document with the ability for a user to provide the correlating number to execute that option. If an invalid input is provided, the program should prompt the user to re-enter a valid input to select an option from the menu.

Test Description: The following testing case verifies that the user gets reprompted if they enter a number not on the menu

Step #	Operator Action	Expected Results	Comments
1	User shall run the InventoryPro.java file to start the program.	System will output a message welcoming the user to the program and ask whether they would like to (1) log in or (2) sign up.	
2	User shall select to log in to the program by inputting 1 into the prompt.	The program will prompt the user to enter their employee ID.	
3	The user shall provide a valid employee ID, such as Emp1018.	The program will prompt the user for a password.	
4	The user shall provide a valid password, such as cheeseFries1!	The program will welcome the user and display the menu.	
5	User should enter an invalid number, such as 8.	The program will reprompt the menu.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
Welcome! Would you like to (1) log in or (2) sign up: 1
Enter Employee ID: Emp1018
Enter Password: cheeseFries1!



Welcome back, John Smith!

=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 8
Invalid option. Please try again.

=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice:
```

Test Case ID: 2C	**Current Status: Pass**	**Date: 3/28**

Req. ID: 2.1 The program shall output a menu with the options as defined in the Software Design Document with the ability for a user to provide the correlating number to execute that option. If an invalid input is provided, the program should prompt the user to re-enter a valid input to select an option from the menu.

Test Description: The following testing case verifies that the user is reprompted the menu if a non-integer value is provided

Step #	Operator Action	Expected Results	Comments
1	User shall run the InventoryPro.java file to start the program.	System will output a message welcoming the user to the program and ask whether they would like to (1) log in or (2) sign up.	
2	User shall select to log in to the program by inputting 1 into the prompt.	The program will prompt the user to enter their employee ID.	
3	The user shall provide a valid employee ID, such as Emp1018.	The program will prompt the user for a password.	
4	The user shall provide a valid password, such as cheeseFries1!	The program will welcome the user and display the menu.	

5	User should enter an invalid input, such as the word “four”	The program will reprompt the menu.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
Welcome! Would you like to (1) log in or (2) sign up: 1
Enter Employee ID: Emp1018
Enter Password: cheeseFries1!



Welcome back, John Smith!

=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: four
Invalid input. Please enter a number.

=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice:
```

Test Case ID: 3A	**Current Status: Pass**		**Date: 4/4**
Req. ID: 3.1 The user shall be able to create either a product item or an expense item with the required values being prompted to the user appropriately and then stored in the database.			
Test Description: The following testing case verifies that the user is able to create a product item			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	Upon being prompted for the menu, the user should select option 1 (add item)	The system will then request a series of generic values for an item.	
2	User shall enter the following values for the following prompts: Name: CRV, Unit Cost: 23687, Quantity: 12, Supplier: Honda, Status: Purchased	The program will prompt the user to select whether the item is a product or an expense.	

3	The user should enter “p” for product.	The program will then prompt the user for a sale price and customer.	
4	The user should enter the sale price as 43228 and customer as John Dunphy.	The program will redisplay the menu. In the backend, the program will show the new item.	
5			

Screenshots:

```
Welcome back, Testy McTesttest!

=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 1
Enter item name: CRV
Enter unit cost: 23687
Enter quantity: 12
Enter supplier: Honda
Enter the status of the item: Purchased
is this item a product (p) or an expense (e): p
Enter sale price: 43228
Enter customer: John Dunphy
```

```
p81649,CRV,23687.0,12,Honda,Purchased,43228.0,John Dunphy,false
```

Test Case ID: 3B	**Current Status: Pass**	**Date: 3/28**

Req. ID: 3.1 The user shall be able to create either a product item or an expense item with the required values being prompted to the user appropriately and then stored in the database.

Test Description: The following testing case verifies that the user is able to create an expense item

Step #	Operator Action	Expected Results	Comments
1	Upon being prompted for the menu, the user should select option 1 (add item)	The system will then request a series of generic values for an item.	
2	User shall enter the following values for the following prompts:	The program will prompt the user to select whether the	

	Name: paper towels, Unit Cost: 12.34, Quantity: 33, Supplier: Brawn, Status: Purchased	item is a product or an expense.	
3	The user should enter “e” for expense.	The program will then prompt the user for an item type and lifespan.	
4	The user should enter the expense type as “Consumable” and lifespan as 120.	The program will redisplay the menu. In the backend, the program will show the new item.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 1
Enter item name: Paper Towels
Enter unit cost: 12.34
Enter quantity: 33
Enter supplier: Brawn
Enter the status of the item: Purchased
is this item a product (p) or an expense (e): e
Enter expense type: Consumable
Enter item timespan: 120
```

```
e49018,Paper Towels,12.34,33,Brawn,Purchased,Consumable,120
```

Test Case ID: 3C	**Current Status: Pass**	**Date: 3/28**

Req. ID: 3.1 The user shall be able to create either a product item or an expense item with the required values being prompted to the user appropriately and then stored in the database.

Test Description: The following testing case verifies that the user is reprompted when selecting not a product or expense

Step #	Operator Action	Expected Results	Comments
1	Upon being prompted for the menu, the user should select option 1 (add item)	The system will then request a series of generic values for an item.	
2	User shall enter the following values for the following prompts:	The program will prompt the user to select whether the	

	Name: Cheese, Unit Cost: 23.23, Quantity: 2, Supplier: Fromage, Status: Purchased	item is a product or an expense.	
3	The user should enter “a” , which is not a key	The program will reprompt for a selection.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 1
Enter item name: Cheese
Enter unit cost: 23.23
Enter quantity: 2
Enter supplier: Fromage
Enter the status of the item: Purchased
is this item a product (p) or an expense (e): a
is this item a product (p) or an expense (e): a
is this item a product (p) or an expense (e):
```

Test Case ID: 4A	**Current Status: Pass**		**Date: 4/4**
Req. ID: 4.1 The user shall be able to enter any valid item ID and returned with the item’s specific information. If the Item ID is invalid, the user should be re-prompted to enter a new ID.			
Test Description: The following testing case verifies that the user is able to enter a product item ID			
Step #	**Operator Action**	**Expected Results**	**Comments**
1	Upon being prompted for the menu, the user should select option 2 (find item)	The system will then request an item ID.	
2	User shall enter the following item ID: “p98070”.	The program will output the information about the product, including product specific information.	The p in the ID indicates it’s a product.
Screenshots:			

```
=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 2
Enter item ID: p98070

Item found:
    Item ID: p98070
  Location:
  Name: Accord
  Unit Cost: $12345.00
  Quantity: 12
  Supplier: Honda
  Status: Purchased
   Sale Price: 32123.0
  Buyer: John
  On sale: false
```

Test Case ID: 4B	**Current Status: Pass**	**Date: 3/28**

Req. ID: 4.1 The user shall be able to enter any valid item ID and returned with the item's specific information. If the Item ID is invalid, the user should be re-prompted to enter a new ID.

Test Description: The following testing case verifies that the user is able to enter an expense item ID

Step #	Operator Action	Expected Results	Comments
1	Upon being prompted for the menu, the user should select option 2 (find item)	The system will then request an item ID.	
2	User shall enter the following item ID: "e00387".	The program will output the information about the expense, including expense specific information.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 2
Enter item ID: e00387

Item found:
    Item ID: e00387
  Location:
  Name: Kleenex
  Unit Cost: $12.44
  Quantity: 23423
  Supplier: Tissue Co
  Status: Purchased
  Expense Type: Consumable
  Timespan: 1234
```

Test Case ID: 4C	**Current Status: Pass**	**Date: 3/28**

Req. ID: 4.1 The user shall be able to enter any valid item ID and returned with the item's specific information. If the Item ID is invalid, the user should be re-prompted to enter a new ID.

Test Description: The following testing case verifies that the user is able to enter an expense item ID

Step #	Operator Action	Expected Results	Comments
1	Upon being prompted for the menu, the user should select option 2 (find item)	The system will then request an item ID.	
2	User shall enter the following item ID: "z43012".	The program will inform the user that there is no item with that ID and prompt the user if they would like to try again.	

Screenshots:

```
=== InventoryPro: Inventory Management System ===
1. Add an item
2. Find an item
3. List all items
4. List all products
5. Create a project
6. Find project by ID
7. Find my projects
99. Exit
Enter your choice: 2
Enter item ID: z43012
No item found with ID: z43012
Try again? (y/n):
```

*[Shall be **completed for user stories actively worked, and completed, in the current sprint**.]*

REFERENCES:

All sources cited previous sections are listed in this section. If the project required no sources, keep this section but leave it blank. Sources might be papers and texts in the general problem domain of the project, code snippets, libraries incorporated in the project, or even algorithmic solutions to specific parts of the project.

[Shall be completed by deliverable P3, and edited as needed for future deliveries.]

APPENDICES:

This is optional, but may include external sources, source code, input data files, or other related material.

[Shall be completed by deliverable P3, and edited as needed for future deliveries]