

Inventory Tracking Program

By: Team Java

Jordan Hidalgo, Jorge Sanchez,
Raymond Duenas, Zackary Bair

Stan State

SWE CS 4800

10/12/2021

Team Contributions: Jordan Hidalgo, Jorge Sanchez, Raymond Duenas, Zackary Bair

Team Member	Contribution
Jordan Hidalgo	<ul style="list-style-type: none"> - Captured System Requirements - Captured Meeting Minutes - Captured Solution Description - Assisted in Problem Identification - Assisted in Identifying Problem Solution - Assisted in Developing Plan of work - Assisted in Developing Timeline - Assisted in Design of Prototype - Captured Domain Analysis - Developed Mathematical Model - Assisted in Developing Sequence Diagram - Assisted in Developing Use Case Diagram - Assisted in Developing Acceptance test cases - Assisted in Developing Project Size Estimation - Captured Class Diagram - Captured System Architecture and System Design - Updates System Architecture and System Design - Updated Class Diagram - Captured design of test - Assisted in the non-functional/user interface requirements - Updated functional requirements - Updated user interface specifications - Updated project size estimation - Updated Acceptance test cases - Updated Meeting log
Jorge Sanchez	<ul style="list-style-type: none"> - Captured Problem Description - Assisted in Problem Identification - Assisted in Identifying Problem Solution - Assisted in Developing Plan of work - Assisted in Developing Timeline - Introduced Project Idea to Team - Assisted in Design of Prototype - Assisted in Developing Domain Analysis - Developed Mathematical Model - Assisted in Developing Sequence Diagram - Assisted in Developing Use Case Diagram

	<ul style="list-style-type: none"> - Assisted in Developing Acceptance test cases - Captured Project Size Estimation - Developed sequence diagram to required detail - Assisted in determining algorithms and data structures - Updated sequence diagram - Assisted in updating class diagram - Assisted in the non-functional/user interface requirements - Updated functional requirements - Updated traceability matrix - Updated use cases for login - Updated user interface specifications - Updated Overview - Updated project size estimation - Updated Acceptance test cases
Zackary Bair	<ul style="list-style-type: none"> - Ensured Plan of Work Execution - Captured Meeting minutes - Captured Project PowerPoint - Assisted in Problem Identification - Assisted in Identifying Problem Solution - Assisted in Developing Plan of work - Assisted in Developing Timeline - Assisted in Design of Prototype - Assisted in Developing Use Case Diagram - Created use cases 2, 3, and 6 - Captured Sequence Diagram - Assisted in Developing Project Size Estimation - Keeps table of contents updated - Keeps track of meetings while I am pleasant - Traceability matrix - Created Gantt chart - Assisted with updating class diagrams - Developed sequence initial sequence diagram - Assisted in system architecture and system design - Developed identify subsystems graphic - Assisted with updating class diagram - Assisted with history of work, current status, and future work - Updated table of contents - Added in references used
Raymond Dueñas	<ul style="list-style-type: none"> - Captured Timeline and Sprints - Presented Project Proposal - Assisted in Problem Identification

	<ul style="list-style-type: none">- Assisted in Identifying Problem Solution- Assisted in Developing Plan of work- Assisted in Developing Timeline- Captured Low Fidelity Prototype- Assisted in Developing Domain Analysis- Developed Mathematical Model- Assisted in Developing Sequence Diagram- Captured Use Case Diagram- Assisted in Developing Acceptance test cases- Assisted in Developing Project Size Estimation- Assisted in Developing Gantt chart- Captured data types and operation signature- Assisted in determining system architecture- Assisted updating system architecture and system design- Assisted in updating class diagram- Captured the coverage and functionalities of test- Updated Use Case Diagram- Updated Prototype- Captured History of work- Captured Key accomplishments- Captured Possible future directions
--	---

TABLE OF CONTENTS

A. Team Meeting Log	7
A.1 Discord Meeting Log	
1. Customer Problem Statement	12
a. Problem Statement	
b. Proposed Solution	
c. Timeline	
i. Sprint 1	
ii. Sprint 2	
d. Glossary of Terms	
2. System Requirements	14
a. Functional Requirements	
b. Non-Functional Requirements	
c. User-Interface Requirements	
d. Prototype	
e. Constraints	
3. Function Requirements Specification	17
a. Actors and Goals	
i. Actors	
ii. Goals	
b. Use Cases	
i. Add Item	
ii. Remove Item	
iii. Item Description	
iv. Sort Inventory	
v. Search	
vi. Login	
c. Use Case Diagram	
d. Traceability Matrix	
e. Sequence Diagrams	
i. Add Item	
ii. Remove Item	
iii. Search Item	
iv. Update Inventory	
v. Log in	
4. User Interface Specifications	27
a. Steps	
b. Overview	
5. Domain Analysis	29
a. Analysis Object Model	
b. Mathematical Model	
i. Add Item	
ii. Remove Item	
iii. Sort Item	

6. Project Size Estimations Based on Use Case Points	30
a. Totals	
b. UUCP	
c. TCF	
d. ECP	
7. Acceptance Use Cases	32
8. Plan of Work	37
a. Gantt Chart	
b. Ownership	
9. Data Types and Operation Signature	38
a. Inventory Control	
10. System Architecture and System Design	38
a. Architectural Style	
b. Identify Subsystems	
c. Mapping Subsystems to Hardware	
d. Persistent Data Storage	
e. Network Protocol	
f. Hardware Requirements	
11. Algorithms and Data Structures	39
a. Algorithms	
b. Data Structures	
12. User Interface Design and Implementation	40
a. Design of tests	
b. Test Coverage and Functionalities	
c. Non-Functional/User-Interface Requirements	
13. History of Work, Current Status, Future Work	43
a. History of Work	
b. Current Status	
c. Future work	
14. Key Accomplishments	44
a. Possible Future Directions	
15. References	44

A. Team Meeting Log

A.1. Discord Meeting Log:

Meeting Date: 09/01/2021

Meeting Time: 2:00 PM - 3:00 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- First discussion: Introductions and planning
- Came up with the idea for the project
- Defined the background, problem description, and proposed solution
- Created the low fidelity prototype
- Defined the teams deadlines as well as which part is each members responsibility
- Researched Java's SWING for the purpose of creating our GUI

Meeting Date: 09/04/2021

Meeting Time: 9:00 AM - 9:15 AM

Attendees: Jordan, Jorge, Raymond

- Discussed requirements of the program

Meeting Date: 09/05/2021

Meeting Time: 10:00 AM - 11:00 AM

Attendees: Jordan, Jorge, Raymond, Zackary

- Created GitHub repository
- Finalize functionality of the program

Meeting Date: 09/07/2021

Meeting Time: 5:30 PM - 5:50 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Reviewed project presentation

Meeting Date: 09/14/2021

Meeting Time: 3:15 PM - 5:00 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Created the cover page, team contribution, team meeting log, and the first two requirements of Report 1

Meeting Date 09/23/2021

Meeting Time: 3:30 PM - 5:30 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Installed Eclipse and TortoiseGit as well as made sure that all team members could access and upload to the GitHub repository

Meeting Date: 09/23/2021

Meeting Time: 8:00 PM - 9:45 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Added in sections 3 and 4 of the report.
- Planned further details for how the UI will interact with the program.

Meeting Date: 09/24/2021

Meeting Time: 12:30 PM -

Attendees: Jordan, Zackary

- Reviewed report before submitting.

Meeting Date: 09/28/2021

Meeting Time: 4:30 PM - 6:00 PM

Attendees: Jordan, Jorge, Raymond

- Developed Domain Analysis
- Developed Mathematical Model

Meeting Date: 10/03/2021

Meeting Time: 9:00 AM - 10:10 AM

Attendees: Jordan, Jorge, Raymond, Zackary

- Adjusted use case diagrams, sequence diagrams, and domain analysis diagram from professors feedback

Meeting Date: 10/04/2021

Meeting Time: 6:30 PM - 8:15 PM

Attendees: Jordan, Jorge, Raymond

- Developed Acceptance Test Scenarios

Meeting Date: 10/05/2021

Meeting Time: 12:00 PM - 2:00 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Added new Use Case, updated Use Case diagram, updated Analysis Object Module, added Sequence diagram, updated Traceability Matrix, completed Acceptance Test Cases, completed project size estimation.

Meeting Date: 10/11/2021

Meeting Time: 6:30 PM - 7:30 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Fixed domain analysis diagram and Gantt chart.

Meeting Date: 10/12/2021

Meeting Time: 6:30 PM - 8:30PM

Attendees: Jorge, Raymond

- Created new report
- Transferred relevant information from report 1 to report 2
- Developed sequence diagram to meet report 2 requirements
- Completed Data types and operations signature

Meeting Date: 10/18/2021

Meeting Time: 6:00 PM - 7:00 PM

Attendees: Jordan, Raymond, Zackary

- Designed architecture and system design
- Identified subsystems
- Specified data storage

Meeting Date: 10/26/2021

Meeting Time: 4:00 PM - 5:20 PM

Attendees: Jordan, Jorge, Raymond, Zackary

- Added sequence diagram for log-in use case
- Refined class diagram
- Refined subsystems and system design

Meeting Date: 11/1/2021

Meeting Time: 10:00 AM - 11:00 AM

Attendees: Jordan, Raymond

- User Interface Design and Implementation
- Design of test
- Test coverage and their functionalities

Meeting Time: 11:00 PM - 11:30 PM

Attendees: Jordan, Jorge

- Non-Functional/User-interface Requirements:

Meeting Date: 11/4/2021

Meeting Time: 3:30 - 4:10 PM

Attendees: Jorge, Jordan, Raymond, Zackary

- Discussed how the workload will be separated
- Worked on log-in page
- Researched log-in page and databases

Meeting Date: 11/07/2021

Meeting Time: 8:00 - 10:00 AM

Attendees: Jorge, Jordan, Raymond, Zackary

- Problem solved with Eclipse
- Created project and pushed to Git
- Created demoOne.java, loginPage.java, and Test.java
- Started creating GUI for loginPage.java
- Hard coded log-in credentials

Meeting Date: 11/09/2021

Meeting Time: 4:00-5:00 PM

Attendees: Jorge, Jordan, Raymond

- Ironed out new issues with pushing to Git using Git Tortoise
- Continued building loginPage.java
- Began study of MongoDB and created database on Altus

Meeting Date: 11/12/2021

Meeting Time: 8:00 - 10:40 PM

Attendees: Jordan, Raymond, Zackary

- Continued study of MongoDB
- Began implementing code and manipulating database through Eclipse

Meeting Date: 11/13/2021

Meeting Time: 6:30 - 8:00 PM

Attendees: Jorge, Jordan, Zackary

- Built homepage GUI
- Finished work on basic login page GUI

Meeting Date: 11/15/2021

Meeting Time: 8:00 - 9:30 PM

Attendees: Jorge, Jordan, Raymond, Zackary

- Worked on PowerPoint presentation for Demo One presentation
- Worked on Demo One Reflections

Meeting Date: 11/18/2021

Meeting Time: 3:30 - 4:30 PM

Attendees: Jorge, Jordan, Raymond, Zackary

- Worked on Report three (key accomplishments, history of work, current status, and future work)

Meeting Date: 11/21/2021

Meeting Time: 6:30 - 7:00 PM

Attendees: Jorge, Jordan, Zackary

- Checked over and finalized Report Three

1. Customer problem statement:

a. Problem statement:

Small business owners have a hard time managing the business with very little man power at their disposal. That extra man power is costly leaving the owner to shoulder the extra work on their own. This extra work can unintentionally cause areas of neglect leading to inefficient management in a business. Manual documentation of available products is hard work and time consuming but it is a necessary part of business operations leaving very little breathing room for business owners to focus on other areas of the business. Neglecting the inventory documentation will lead to owners losing touch with their customer base, which dictate the demand. Business is supply and demand and neglecting the demand for a product is bad business.

b. Proposed Solution:

The (product name) is designed to help lessen the workload by simplifying the inventory process down to the click of a button so the user has more time to focus on other areas of his business without worry. (product name) will track growing and decreasing inventory and product loss across the stores available inventory as well as maintaining a count of what products are more popular so the user can make an educated decision whether to stock more of the popular products or drop the products that are not selling well. While these tasks can be done manually by hand, (product name) makes it faster and easier so the user has more time to focus on other tasks that require attention. Good business relies on good reliable information and (product name) displays that information for the user in a moment's notice right on their monitor to help make more educated and efficient decisions for the business. (product name) is the tool that business owners can rely on to help their business flourish.

c. Timeline:

The project turn over date is Monday December, 6th. In order to successfully complete the project by the given deadline, the team will be utilizing the SCRUM Software Development Life Cycle model. Separating the workload into two six to seven week sprints, we expect to complete a fully functioning Inventory tracking system before the turn over date. The execution of the sprints will be as follows.

i. **Sprint 1:**

Spanning six to seven week sprint 1 will be dedicated to determining functionality, architecture of class communication, design of graphic user interface and writing code for the core functions of the program. The sprint will begin with focused communication with the business owner to ensure desired functionality and interface are understood before moving to implementation. Once the expectations have been communicated and are understood the development team, utilizing case diagrams and flowcharts, will determine the flow of data through the program to develop an overall structure. Once it has been determined that the program structure and owner expectations are in alignment implementation will begin. Implementation for sprint one will only go so far as to code basic user interface and core operation functionalities including input capability, class communication, data storage and manipulation. Sprint one will conclude with a presentation of a low level prototype demonstrating the development team's execution of the core functionalities that the owner requested. The presentation will serve as an opportunity to ensure the project is progressing according to the owner's vision.

ii. **Sprint 2**

Being the final sprint, sprint 2 will also span six to seven weeks. Sprint 2 will initially be dedicated to rectifying any improper executions of the owner's vision revealed during the Sprint 1 presentation. After which the primary focus of Sprint 2 will be to complete the desired auxiliary functionalities of the program, developing the final graphical user interface, and debugging any unforeseen data flow issues. Week four of Sprint 2 will include an owner vision sync meeting. The goal of the mid sprint owner vision sync is to present a high-fidelity graphical user interface allowing for any miss interpretations of owner vision to be caught and addressed before the project turn over date.

d. **Glossary of Terms:**

- **Java:** a high level, object oriented programming language for developing programs.
- **User Interface (UI):** a series of screens, pages, visual elements and icons that allow a user to interact with a product.
- **SCRUM:** is a lightweight framework that helps people, teams and organizations generate value through adaptive solutions for complex problems.
- **Software Development Life Cycle:** a process used by the software industry to design, develop and test high quality software.
- **Class:** the basic building block of an object oriented language, is a template that describes the data and behavior associated with instances of that class.

2. System requirements:**a. Functional Requirements:**

REQ-X	Requirement	PW	Brief Description
REQ-1	Ability to add items to inventory	10	Users should be able to add items to their inventory.
REQ-2	Ability to remove items from inventory	8	Users should be able to remove items from their inventory.
REQ-3	Ability to search for items in inventory	9	Users should be able to search for items in their inventory.
REQ-4	Ability to state reason for item removal	5	Users should be able to determine why an item is being removed from inventory..
REQ-5	Ability to rank item by popularity	4	Program should be able to rank items by their amount sold.
REQ-6	Ability to display the amount of items in inventory	7	Program should be able to display the amount of an item in the inventory.
REQ-7	Ability to display the popularity of items in inventory	3	Program should be able to display the popularity of the items in inventory.
REQ-8	Login page for security	8	Users are prompted to Login into the program to keep information safe.

b. Non-functional requirements:

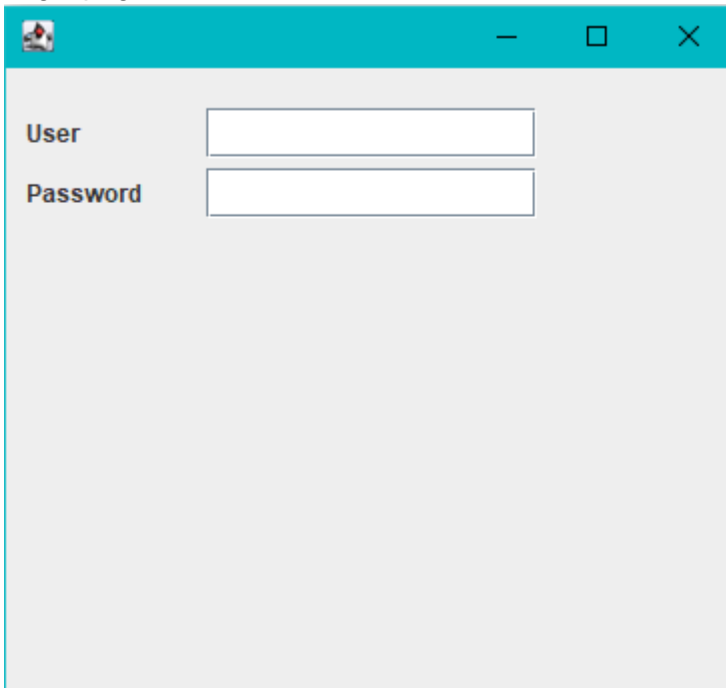
NF-REQ-X	Requirement	PW	Brief Description
NF-REQ-1	Simple, accessible interface	7	Interface should be simple and easy to use, with a maximum response time of 4 seconds.
NF-REQ-2	Reliable	10	Program should be able to save data in response to a crash and reboot within 10 seconds.
NF-REQ-3	Secure	8	Program should be secure with hash passwords with a quick login of 5 seconds.
NF-REQ-4	Saved Data	10	Data is saved to a local database within 3 seconds.

c. User-interface requirements

UI-REQ-X	Requirement	PW	Brief Description
UI-REQ-1	Display amount of item in inventory.	8	Program should be able to display the amount of an item in inventory.
UI-REQ-2	Ability to add, sell, search for, and remove items	10	Users should be able to add, sell, and remove items from inventory.
UI-REQ-3	View item popularity	7	Users should be able to view the popularity of a selected item in inventory.
UI-REQ-4	Sort by popularity	7	Users should be able to sort and view their inventory by popularity.

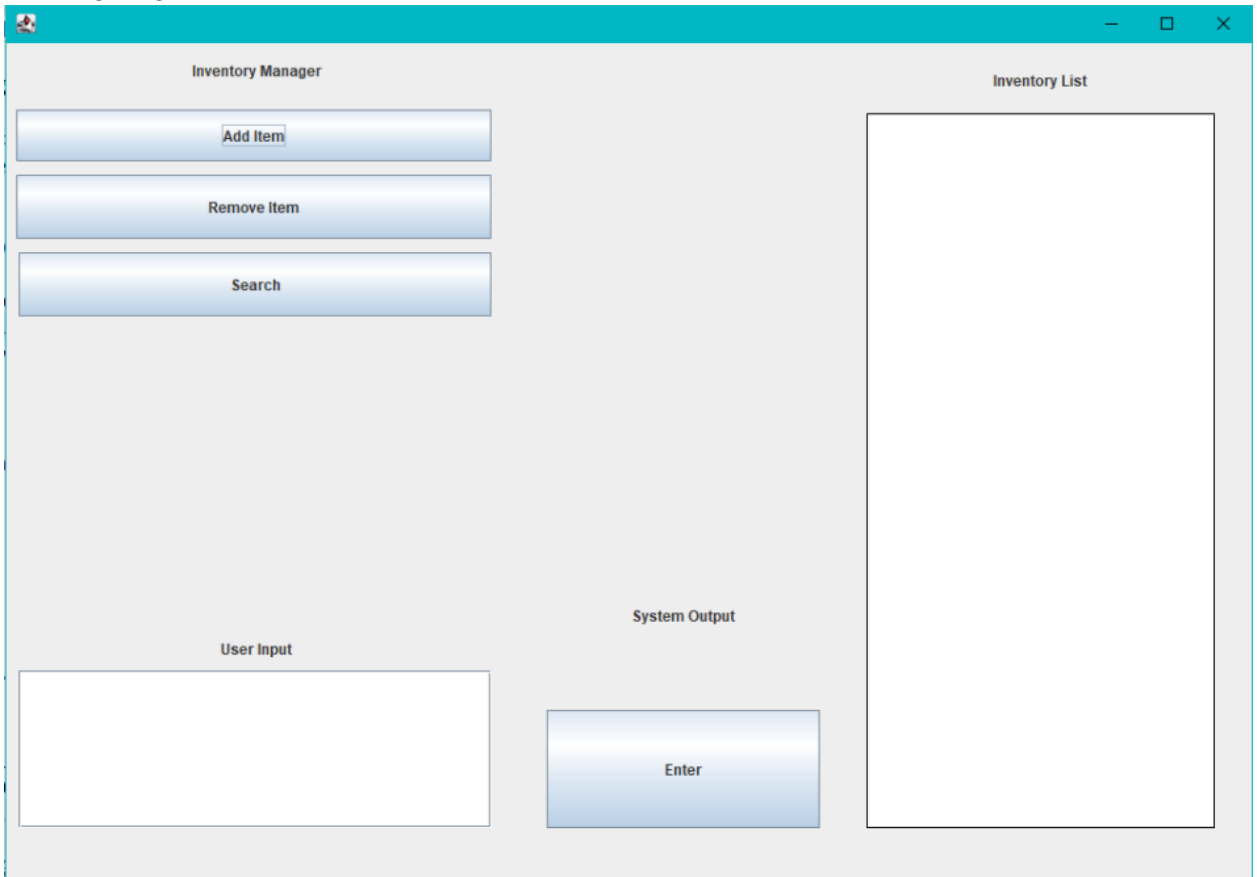
d. Prototype

- Login page



A prototype of a login page displayed in a window. The window has a teal title bar with standard minimize, maximize, and close buttons. The main area is light gray and contains two labels, "User" and "Password", each followed by a white rectangular input field.

- Landing Page



A prototype of a landing page displayed in a window. The window has a teal title bar with standard minimize, maximize, and close buttons. The main area is light gray and contains several elements:

- Inventory Manager**: A section on the left containing three stacked blue buttons labeled "Add Item", "Remove Item", and "Search".
- Inventory List**: A large, empty white rectangular area on the right.
- User Input**: A large white rectangular input field at the bottom left.
- System Output**: A label above a blue button labeled "Enter" at the bottom center.

e. Constraints

The primary IDE that we will be using is Eclipse. In Eclipse we will be adding the Java package as well as a package to allow us to create a meaningful and simplistic GUI. The constraints that we will face while using Eclipse are each file of the program can have only a single team member edit the file at a time. This constraint will be solved by having each team member dedicated to an assigned file, so that no two members are trying to edit a single file at the same time. To allow for proper management of the program's files, all members will be using a common GitHub repository to upload the most recent version of each file.

3. Function Requirements Specification:

3.1 Actors and Goals:

3.1.1 Actors:

- User: Business owner or employee using the app for inventory management

3.1.2 Goals:

- Add item to inventory
- Remove item from inventory
- Search for item in the inventory
- Display amount of item in inventory
- Display popularity of item in inventory

3.2 Use Cases:

3.2.1 Use case Scenarios

#1 Use Case Name: Add Item

Actors: User

Entry Condition: 1. User clicks the "Add Item" button from the right side of the UI.

Flow of Events: 2. The system responds by prompting the user for the name and amount of the item to be added, as well as a description of the item.
 3. The user enters the name and amount of the item.
 4. The system responds by adding the item to the system, or
 Increasing the amount of the item in the system.

Exit Condition: 5. The system displays "Your item has been added to the inventory."

#2 Use Case Name: Remove Item

Actors: User

- Entry Condition:** 1. User clicks the “Remove Item” button from the left side of the UI.
- Flow of Events:** 2. The system responds by presenting the user with new text fields.
 3. The User enters in the item to be removed, amount to be removed, and
 a check box for if the item is removed for a reason other than being
 sold.
 4. If box was checked, then User will fill a text field for why the item was
 removed.
- Exit Condition:** 5. System will then prompt the name of the item removed, how many
 were removed, and for what reason.

#3 Use Case Name: Item Description

Actors: User

- Entry Condition:** 1. User clicks “item” they want to check inventory from the inventory log.
- Flow of Events:** 2. System responds by opening a page for specific items clicked.
 3. System page displays all information regarding specific item in
 one page.
- Exit Condition:** 4. Users can exit the page by clicking return button displayed by the
 system.

#4 Use Case Name: Sort Inventory

Actors: User

Entry Condition: 1. User clicks the “Sort Inventory” button from the left side of the UI.

Flow of Events: 2. System brings up a sub menu for sorting options.
 3. User selects how they would like the items to be sorted from the submenu.
 4. System uses a sorting algorithm to sort the list of items on the right side of the UI to match the user’s desired sort e.g. popularity.

Exit Condition: 5. System displays the new sorted list of items.

#5 Use Case Name: Search

Actors: User

Entry Condition: 1. User navigates to the search button on the left side of the program home page.

Flow of Events: 2. User inputs the item name into the search bar and hits the enter button.
 3. Program filters out the specific item for the user and displays it.
 4. User is prompted to click the item for further description (call to the item description function).

Exit Condition: 5. The System will display a return button for the user to go back to main inventory page.

#6 Use Case Name: Update Inventory

Actors: User

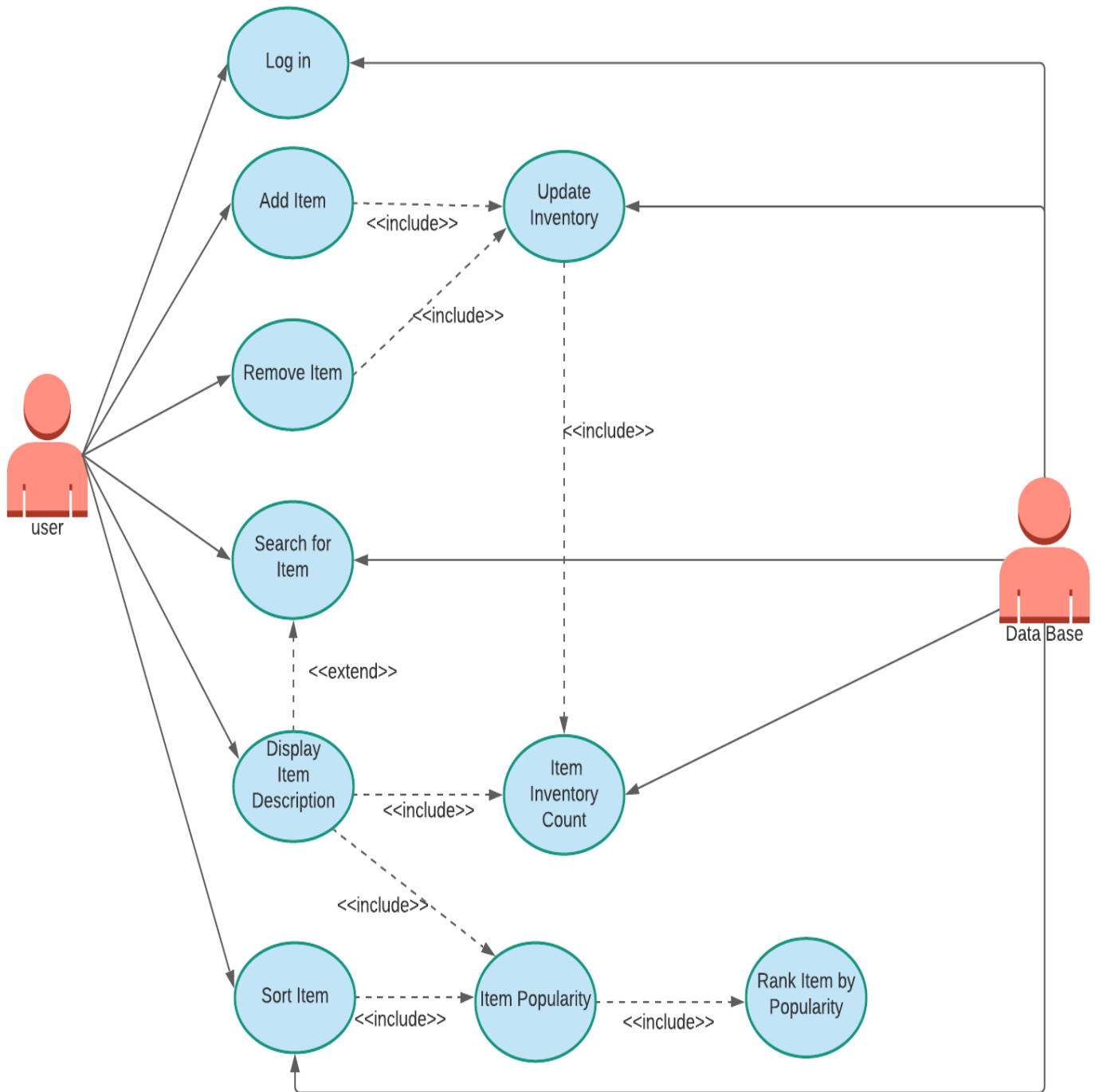
- Entry Condition:** 1. User has successfully finished the add or remove items use case.
- Flow of Events:**
2. Database finds the entity in the inventory database that is being modified.
 3. If inventory is being added and the item already exists in the database, the amount of items remaining in inventory is adjusted by the number of items added.
 4. If inventory is being added and the item does not exist in the database, a new entity is created and the description of the item along with the amount added is placed into the database.
 5. If inventory is being removed, the number of items is adjusted by the number of items to be removed.
 6. Reason for removal is updated according to the users removal reason.
 7. If all items from the entry are removed, the entry is deleted from the database.
- Exit Condition:** 8. Inventory of items in the item list of the UI is updated.

#7 Use Case Name: Login

Actors: User

- Entry Condition:** 1. User has successfully entered verified credentials
- Flow of Events:**
2. Credentials entered are compared to existing verified credentials in Database
 3. Credentials matching existing ones allow user entry into program
- Exit Condition:** 4. Login successful

3.3 Use Case Diagram:

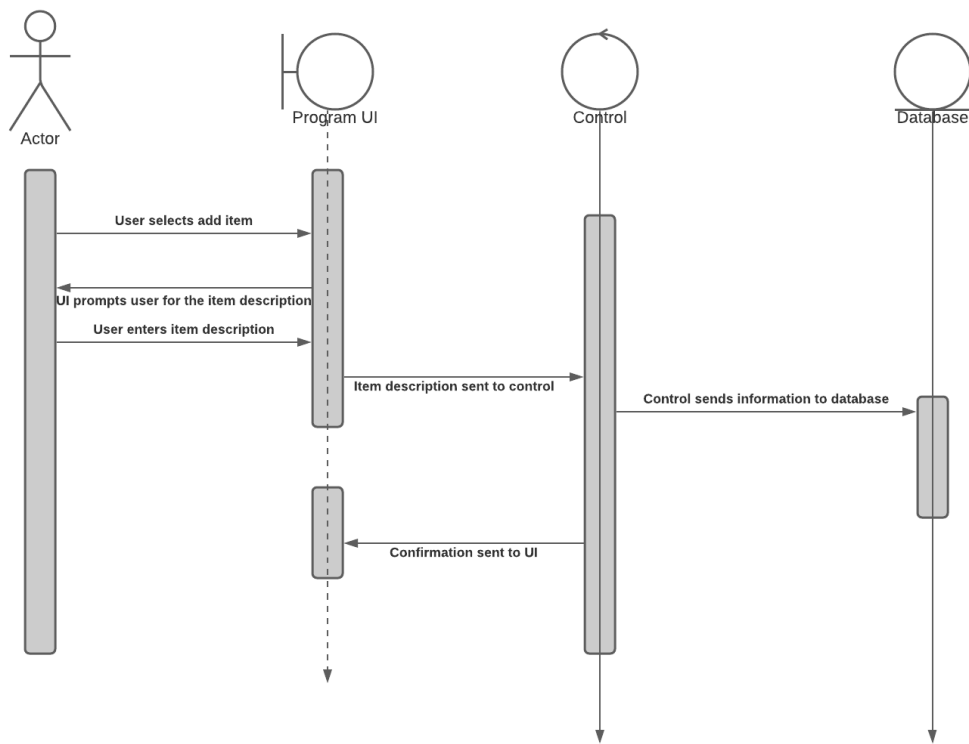


3.4 Traceability Matrix:

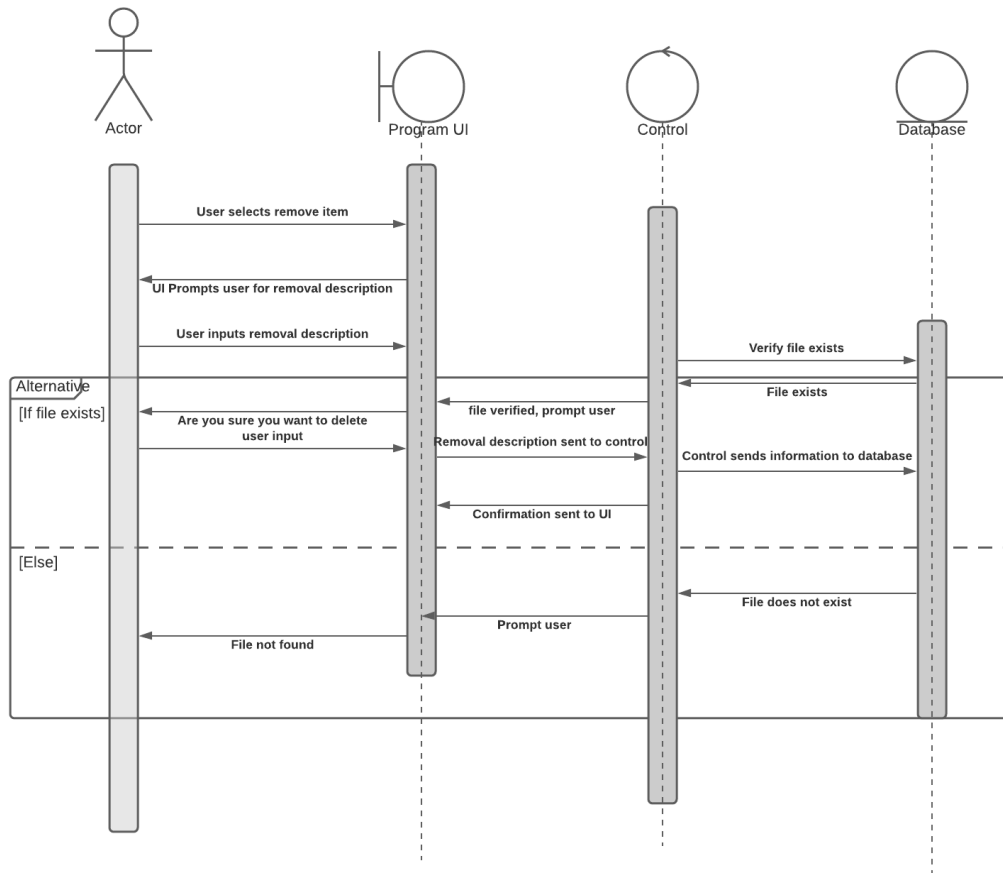
Req	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7
1	10	X						
2	8		X					X
3	9					X	X	
4	5		X					
5	4			X			X	
6	7			X	X			
7	3			X	X			
Max PW		10	8	7	7	9	9	8
Total PW		10	13	14	10	9	14	8

3.5 Sequence Diagrams:

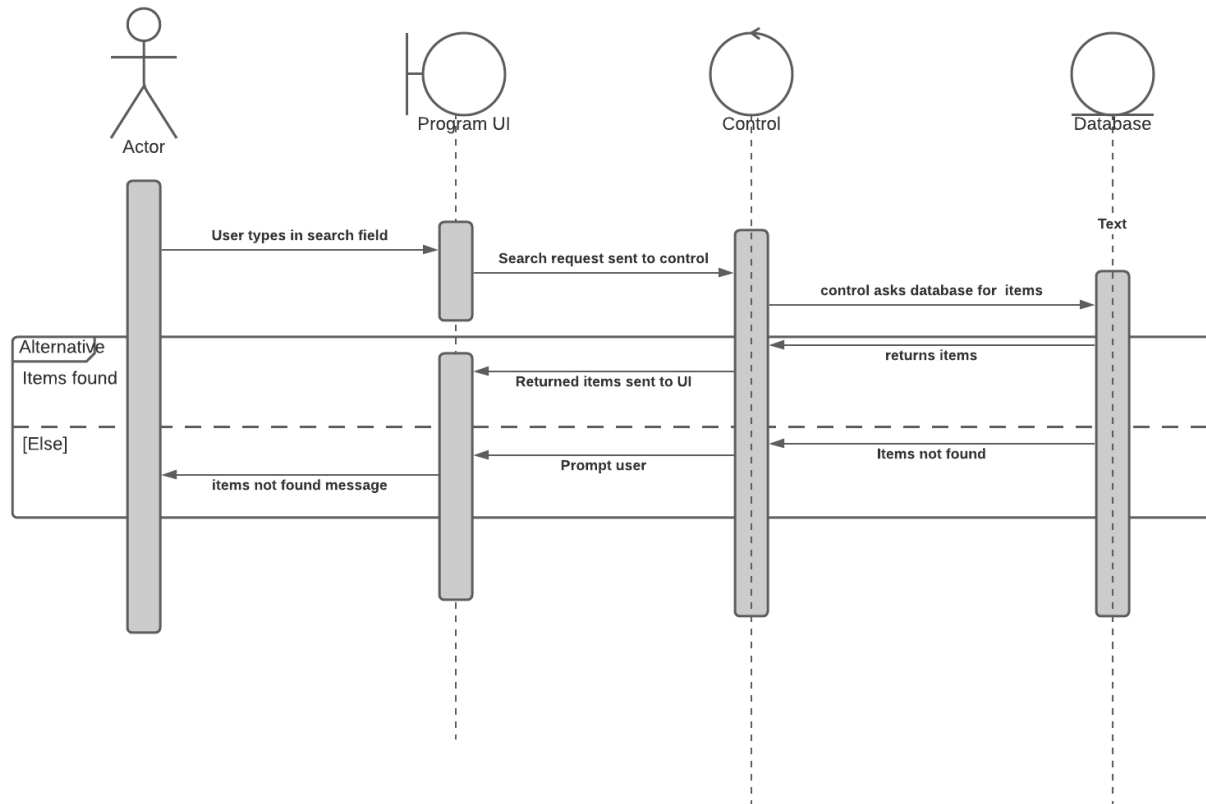
3.5.1 Add Item Use Case



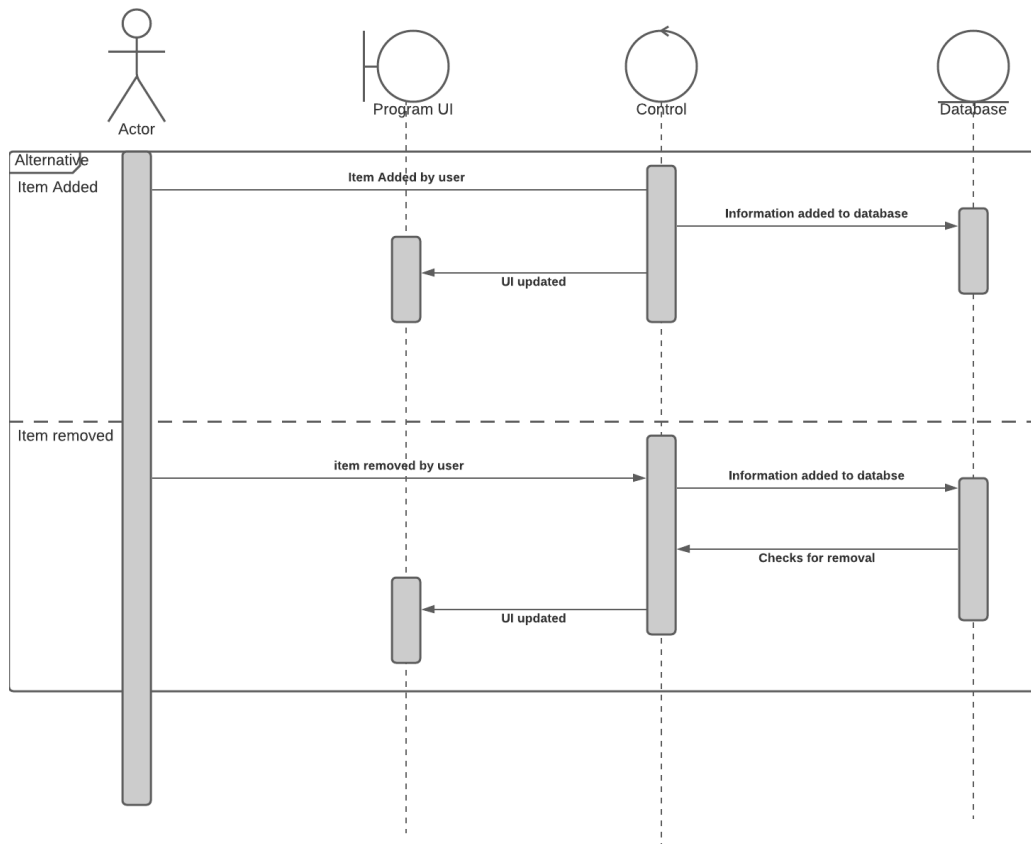
3.5.2 Remove Item Use Case



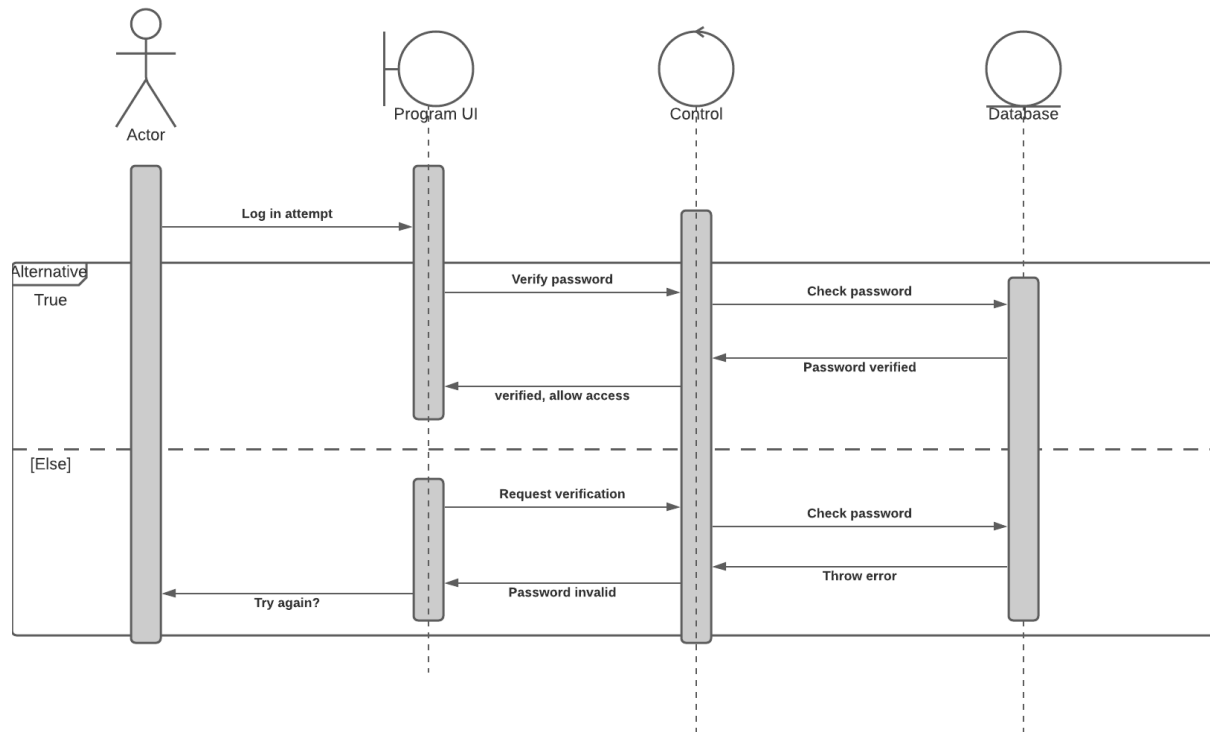
3.5.3 Search Use Case



3.5.4 Update Inventory Use Case



3.5.5 Log In



4. User Interface Specifications:

4.1 Steps:

4.1.1 User opens the program. Home page of the program being the inventory

4.1.2 User adds store inventory using the add item functionality

4.1.2.1 User fills out item description and starting quantity

4.1.2.2 System saves data information

4.1.2.3 System prompts user to return to home page after information is added

4.1.2 User removes items from inventory using the remove item functionality

4.1.2.1 User selects item they want removed from inventory

4.1.2.2 System prompts user if they want to remove item and why?

4.1.2.3 System removes item

4.1.2.3 System kicks back user to inventory

4.1.3 User sorts items using sort functionality

4.1.3.1 If the user wants to sort items differently from default sort of amount they click the sort button.

4.1.3.2 System submenu shows different sorting options to user.

4.1.3.3 User chooses desired option.

4.1.3.4 System changes inventory view to match desired option.

4.1.4 User search functionality

4.1.4.1 User inputs name of item in search bar

4.1.4.2 System filters out products that do not match name

4.1.4.3 Inventory page only displays item that matched the name

4.1.5 Login functionality

4.1.5.1 User fills out credentials

4.1.5.2 System checks if credentials are valid

4.15.3 Once credentials are verified System allows user to enter

4.2 Overview:

4.2.1 The presentation of our software will be very simple and easy to understand. The layout. Buttons for user interaction will be placed in the top left quadrant of the screen. These buttons will include the Add, Search, Remove, and Sort buttons. On the right side of the screen we will provide the list of items in the system, which can be sorted to the user's preference. In the bottom left quadrant we will have an area for system messages that will inform the user when actions such as adding and removing have been performed.

4.2.1.1 **Adding an Item:** A button labeled "Add Item" will be displayed in the top left corner. Our program will have nothing to search, sort, or remove without items in the system, so this will be the first button the user sees. Clicking the button will prompt the user for an item and the amount to be added.

4.2.1.2 **Search for an Item:** Once items have been added to the system, searching for an item is the next logical step for a user, to confirm that items have been added and that the amount displayed is correct. This "Search" button will be placed in the same area below the "Add" button. Clicking this button will display the item in inventory, and any available statistics

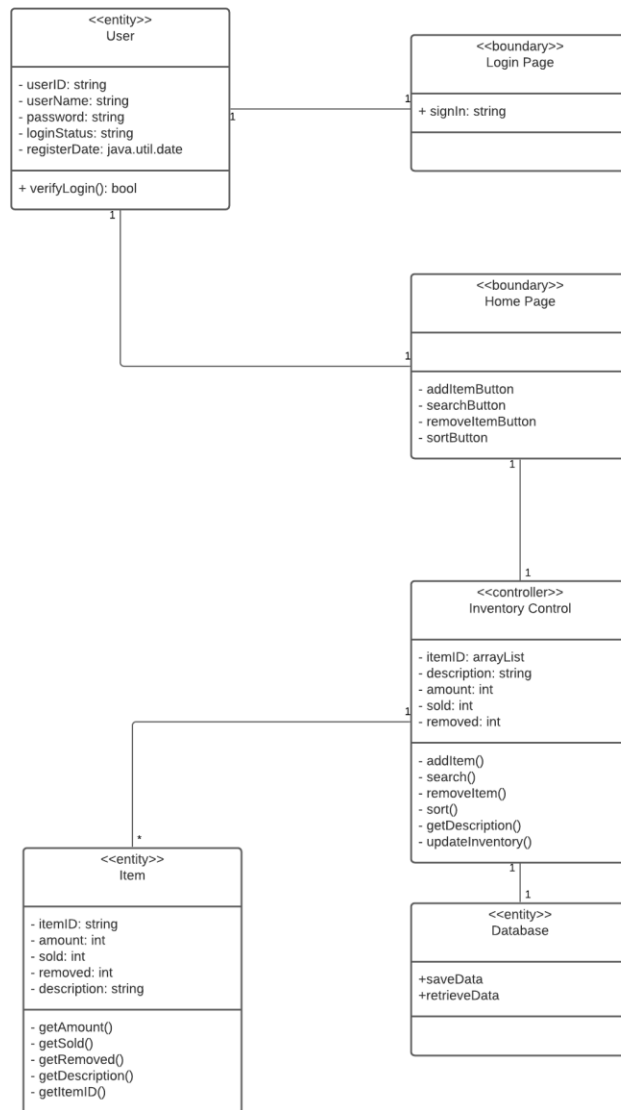
4.2.1.3 **Sort Items:** When the inventory of a business has been established, a user may desire to sort their items by popularity or the amount in stock. The "Sort" button will be placed in the same upper left quadrant under the "Search" button. When pressed the program will sort items and display them on the right side of the screen.

4.2.1.4 **Removing items:** Finally, the user will be able to remove items from their inventory using the "Remove" button. This button will be located under the "Sort" button in the upper left quadrant of the UI. Upon pressing this button, users will be prompted to enter the item they wish to remove. After this, the user will be prompted to select a reason for removal, such as a sale or product spoilage.

4.2.1.5 **Login:** the login page will add security to the program and give the user one less thing to worry about. This page will be the first thing a user sees every time they use the program. This too will be kept simple for the user, the page will have two buttons "login" and "create" the latter of which is for new users to create their own personal credentials, the former will simply prompt the user to enter valid username and password.

5. Domain Analysis:

Analysis object model:



Mathematical Model:

5.1.1 Add Item:

-Variables

- IA = Item Amount

- AA= Amount Added

- Adding more items to inventory database

- Equation: $IA = IA + AA$

5.1.2 Remove Item:

-Variables

- IA = Item amount

- ID = Item deleted/subtract

-Subtracting items from inventory database

- Equation: $IA = IA - ID$

5.1.3 Sort Item:

-Variable

- IRP = Item Removal for Purchase

-Utilize inequalities to compare IRP number of individual items. Sorting items from greatest IRP number at top to lowest at the bottom.

6. Project Size Estimations Based on Use Case Points:

Totals:

UCP	(UUCP * TCF * ECF) = 34.83
UUCP	39
TCF	0.77
ECF	1.16

UUCP:

UAW	4
Simple actors	1
Complex actors	3

UUCW	45
Add item	5
Remove item	5
Search	5
Sort	5
Item description	5
User	10
Login Page	10

TCF:

	TCF	0.77
T1	Distributed system	6
T2	Response time/performance objectives	1
T3	End-User efficiency	1
T4	Internal Processing complexity	1
T5	Code reusability	0
T6	Easy to install	1
T7	Easy to use	.5
T8	Portability to other platforms	0
T9	System maintenance	1
T10	Concurrent/parallel processing	0
T11	Security features	4
T12	Access for third parties	0
T13	End user training	0
Sum Total:		15.5

ECF:

	ECF	1.16
E1	Familiarity with development process used	3
E2	Application experience: somewhat familiar	1
E3	Object oriented experience	2
E4	Lead analyst capability	1
E5	Motivation of the team: high	1
E6	Stability of requirements	2
E7	Part-time staff	0
E8	Difficult programming language	-2
Sum Total:		8

7. Acceptance Test Cases:**Add existing Item vs. add New Item**

Test- Case Identifier: TC-1.01 Use Case Tested: UC-1, "Add Item" Pass/fail Criteria: The test passes if the user enters a valid Inventory ID number. Input Data: Alphanumeric code, Inventory item ID	
Test Procedure:	Expected Result:
Step 1. Type an inventory ID Alphanumeric code that does not exist in inventory and click "Add Button".	System prompts the user that the imputed Item does not currently exist in inventory. "Would you like to create a new inventory Item?"
Step 2. Type an inventory ID Alphanumeric code that exists in the inventory and click "Add Button".	System prompts the user that the imputed item has been added to inventory.

Test- Case Identifier: TC-1.02 Use Case Tested: UC-1, "Add Item" Pass/fail Criteria: The test passes if the user enters a valid Inventory ID number. Input Data: Alphanumeric code, Inventory item ID	
Test Procedure:	Expected Result:
Step 1. User clicks the "Add item" button with no correct item code or "new" in the field.	System prompts the user to input correct item code or input "new" in the field
Step 2. Type new into add Item input line and click "Add Button" then inputs new item inventory ID, item name and item description.	System prompts the user that the imputed item has been created in the inventory log.

Test- Case Identifier: TC-1.03 Use Case Tested: UC-1, "Add Item" Pass/fail Criteria: The test passes if the user enters a valid Inventory ID number. Input Data: Alphanumeric code, Inventory item ID	
Test Procedure:	Expected Result:
<p>Step 1. Type "new" into add Item input line and click "Add Button" then inputs new item name, item description and item inventory ID but does not follow inventory ID convention.</p> <p>Step 2. Type new into add Item input line and click "Add Button" then inputs new item inventory ID, item name and item description.</p>	<p>System prompts the user that the imputed Item ID does not follow Inventory ID format .</p> <p>System prompts the user that the imputed item has been created in the inventory log.</p>

Test- Case Identifier: TC-1.04 Use Case Tested: UC-1, "Add Item" Pass/fail Criteria: The test passes if the user enters a valid Inventory ID number. Input Data: Alphanumeric code, Inventory item ID	
Test Procedure:	Expected Result:
<p>Step 1. Type "new" into add Item input line and click "Add Button" then inputs new item inventory ID and item description but leaves Item name blank.</p> <p>Step 2. Type new into add Item input line and click "Add Button" then inputs new item inventory ID, item name and item description.</p>	<p>System prompts the user that the imputed Item can not be added without the Item name.</p> <p>System prompts the user that the imputed item has been created in the inventory log.</p>

Test- Case Identifier: TC-1.05 Use Case Tested: UC-1, "Add Item" Pass/fail Criteria: The test passes if the user enters a valid Inventory ID number. Input Data: Alphanumeric code, Inventory item ID	
Test Procedure:	Expected Result:
Step 1. Type new into add Item input line and click "Add Button" then inputs new item inventory ID and item name but leaves item description blank.	System prompts the user that the imputed Item can not be added without item description.
Step 2. Type new into add Item input line and click "Add Button" then inputs new item inventory ID, item name and item description.	System prompts the user that the imputed item has been created in the inventory log.

Test- Case Identifier: TC-2.01 Use Case Tested: UC-2 "Remove item" Pass/fail Criteria: The test passes if selected item is removed from database and available product list. The test fails if the item has no inventory to remove Input Data: Alphanumeric code, Inventory ID	
Test Procedure:	Expected Result:
Step 1: input item code that does not exist in the system and click "Remove item" button	System does not remove item since there is no item to remove. Notifies the user the item is not in inventory
Step 2: Input item code that does exists in the system and click "Remove item" button	System removes item from the list of available stock. System lets the user know the item was removed.

Test- Case Identifier: TC-2.02 Use Case Tested: UC-2 "Remove item" Pass/fail Criteria: The test passes if selected item is removed from database and available product list. The test fails if the item has no inventory to remove Input Data: Alphanumeric code, Inventory ID	
Test Procedure:	Expected Result:
Step 1: input item code that is in the system but has zero items in stock and clicks "Remove item" button	System does not remove item since there are zero items to remove. Notifies the user the item has zero items in inventory
Step 2: input item code that is in the system but has items in stock and clicks "Remove item" button	System removes item from the list of available stock. System lets the user know the item was removed.

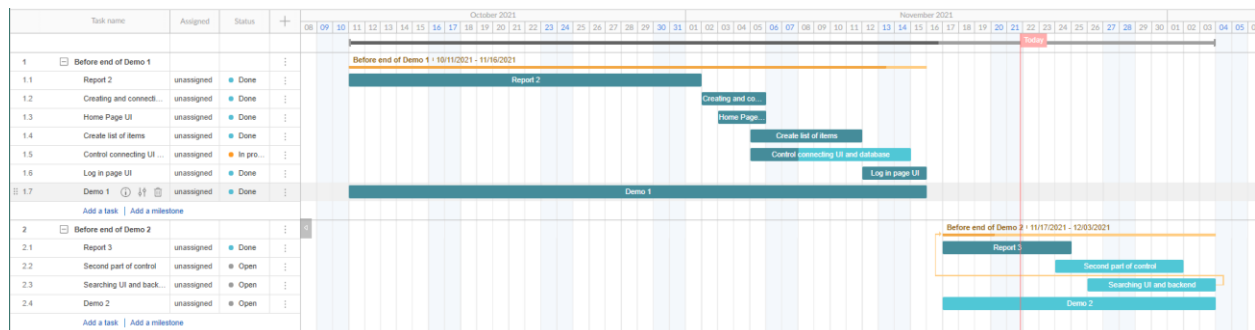
Test- Case Identifier: TC-4 Use Case Tested: UC-4 "search" Pass/fail Criteria: The test passes if the item inputted into the search bar is found. The test fails if the item was not found in the database Input Data: Alphanumeric code	
Test Procedure:	Expected Result:
Step 1 User types in incorrect item code into the search bar	System attempts to match the item searched. Returns "item not found" message to user
Step 2 User types in correct item code into the search bar	System matches the item searched to items in the database and returns a match. Displays the item for the user.

Test- Case Identifier: TC-7 Use Case Tested: UC-7 "login page" Pass/fail Criteria: The test passes if the correct username and password are entered. The test fails if the incorrect information is entered. Input Data: Alphanumeric code	
Test Procedure:	Expected Result:
Step 1 User types in incorrect credentials	System attempts to match the username and password to valid existing credentials. Does not return a match. User entry denied. System outputs "Username and login do not match"
Step 2 User types in correct credentials	System matches the username and password to valid existing credentials. System outputs "Login successful"

8. Plan of Work:

8.1 Gantt Chart:

ID	Task Name	Predecessor	Time Estimate			Expected Time (T_e)
			Opt. (O)	Normal (M)	Press.	
1	Creating and connecting database	-	1.5	2	4	2.25
2	Home page UI	-	0.5	1	2	0.66
3	Create list of items	-	0.5	0.5	1.5	1.08
4	Create control linking UI and database	1, 2	1	2	4	2.16
5	Second part of control	1, 2	1.5	3	5	3.08
6	Searching UI and backend	1, 2, 3	0.5	1	4	1.41
7	Login page UI					



8.2 Ownership:

- 8.2.1. Jordan Hidalgo:
- 8.2.2. Jorge Sanchez:
- 8.2.3. Raymond Dueñes:
- 8.2.4. Zackary Bair:

9. Data types and Operation signature:

i. Inventory Control

1. Operations:

- addItem() - item is added to inventory stored in database
- search() - item is located in inventory database
- removeItem() - item is removed from database
- sort() - inventory from database is displayed in a specified order
- getDescription() -
- updateInventory() -

2. Attributes:

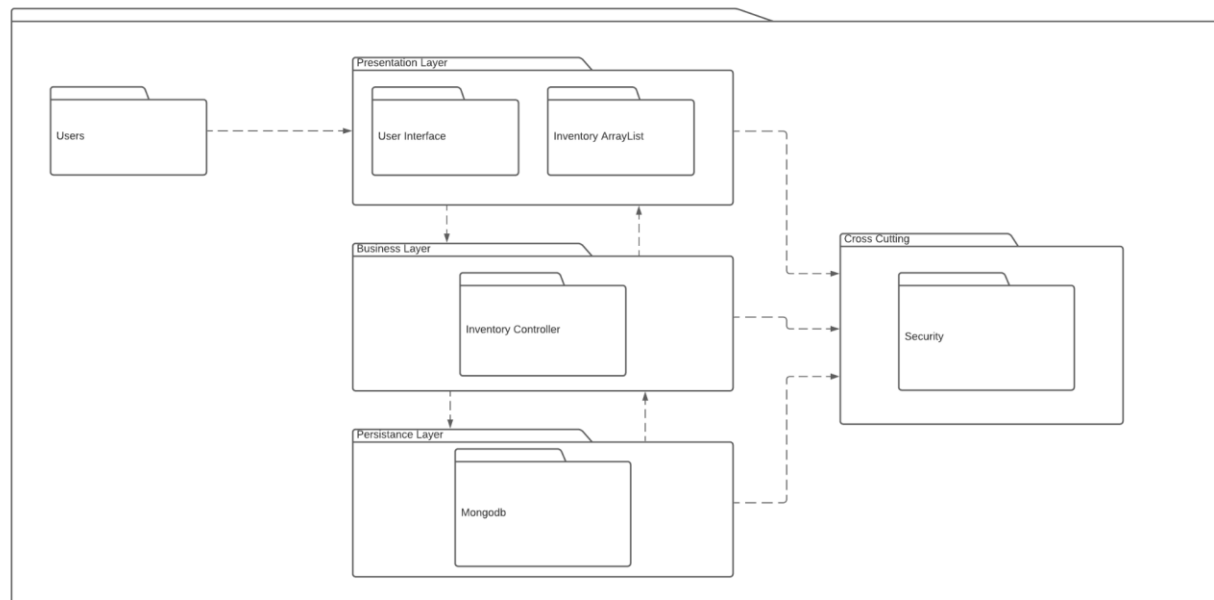
- List itemID - this will hold the list of inventory item id
- String description - this will describe the inventory item
- Int amount - this will be the amount of an item currently in the inventory
- Int sold - this will be the amount of times an item has been sold
- Int remove - this will be the quantity of an item that is being removed from the inventory

10. System Architecture and system design:

a. Architectural style:

- i. The architectural style that we will be using is the Layered Pattern. We will decompose our project into groups of subtasks that are contained within their own respective classes. For Sprint One, our group will be focusing on creating and connecting the database, creating the home page UI, and creating a list of items. Sprint Two will be spent focusing on creating controllers to link the UI and database, as well as any additional functionalities such as search and ranking.

b. Identify subsystems:



- c. Mapping subsystems to hardware:
Our system will require multiple machines, one machine will be the local computer which our user physically interacts with. This machine will run every subsystem besides the database. The second machine will be the server utilized by mongodb which will contain the program's database subsystem.
- d. Persistent data storage:
We will store our data using an online database provided by Mongodb. Early versions of our system may store data locally; however, it is our hope to get our online database running as soon as possible. In doing so we hope to avoid relying on local storage and pull all data from the online database.
- e. Network protocol:
Our system will use the Transmission Control Protocol, TCP for short. This network protocol was chosen for its reliability of data transfer. While most of the software is going to be run from the user's local machine, the TCP network protocol will allow for accurate and reliable connection to Mongodb. This is meant to ensure that the business is able to order inventory when stock is low without ordering unnecessary stock.
- f. Hardware requirements:
For our system to work properly, a good network connection will be required with a desktop or laptop running Windows.

11. Algorithms and Data Structures

- a. Algorithms:

We will utilize an algorithm for sorting the inventory elements in a variety of ways, comparing all elements in the database and placing them in order based on their ranking with respect to a selected criterion. Ranking will be determined by comparing the integer values for the selected characteristic of the various elements within the inventory database. We will also be utilizing algorithms for adding inventory items and removing inventory items from the database.
- b. Data Structures:

The data types that we will implement in the execution of our program include list, string, int and boolean. List will be used to hold inventory items as objects containing multiple characteristics within the database. The string data type will be utilized to assign inventory item names, item descriptions, and describe the reasons for the removal of inventory items. The inventory objects of the list will also hold int values to track the amount of an item currently in stock, and the number of times an item is removed for a particular reason. Int will also be used to determine the ranking of items with respect to each other by comparing the int values for particular characteristics of objects within the list. Boolean will be used in displaying a sorted list.

Data Types: ArrayList, String, Int, Boolean

12. User Interface Design and Implementation

a. Design of tests

1. Test coverage and their functionalities

1.1

Function	1) Ability to add items to inventory
Description	- When adding an item the user should be able to press the add item button and input their data into the text box. That data should be saved into the database.
Purpose	- New items should be saved into the database.

1.2

Function	2) Ability to remove items from inventory
Description	- When removing an item the user should be able to press the remove item button and input their data into the text box. That data should be saved into the database.
Purpose	- Removed items should be removed from the database.

1.3

Function	3) Ability to search for items in inventory
Description	- Given a predetermined data set of items, the user should be able to use the search button to locate any of the items in inventory. This item and its attributes should be properly displayed in the list of items.
Purpose	- System should properly search and display the user's target item.

1.4

Function	4) Ability to state reason for item removal
Description	- When using the remove item button, user should be able to determine the reason for the item's removal
Purpose	- System should allow user to properly specify the reason for an item's removal

1.5

Function	5) Ability to rank items by popularity
Description	- Given a predetermined list of items and their attributes, the system should be able to rank these items by amount in inventory, amount sold, or removed when the sort button is pressed by the user.
Purpose	- System should properly rank items when prompted by the user

1.6

Function	6) Ability to display amount of items in inventory
Description	- Given a predetermined data set of items and their attributes, the program should be able to display the amount of said items properly in the list of items and item description.
Purpose	- System should properly display item amount

1.7

Function	7) Ability to display popularity of items in inventory
Description	- Given a predetermined data set of items and their attributes, the program should be able to display the popularity of an item (calculated from amount sold) properly in the list of items and item description.
Purpose	- System should properly display the item popularity

1.8

Function	8) Ability to access program features using username and password
Description	- Given a predetermined data username and password, the program should be able to login properly to access the inventory list, data, and program functions.
Purpose	- System should properly log the user into the system

2. Test coverage and functionalities:

Our first report outlined seven different functional requirements including: 1) Ability to add items to inventory, 2) Ability to remove items from inventory, 3) Ability to search for items in inventory, 4) Ability to state reason for item removal, 5) Ability to rank item by popularity, 6) Ability to display the amount of items in inventory, 7) Ability to display the popularity of items in inventory. We have since added an eighth function in recognition of the need for a secure login feature. 8) Ability to access program features using username and password. We have developed test cases with respect to these functions to ensure proper functionality of the program. Our tests outline to the best of our ability the most accurate expected result for each function of our program when prompted by user input.

3. Non-Functional/User-interface Requirements:

This project's goal is to develop an effective inventory management system for small business owners. The program is made with simplicity in mind and aims to minimize the effort for the user by keeping the majority of information on the home screen. The program has alerts that are color coded in order to quickly take note which items require the most attention. This program includes features such as adding items, removing items, searching for items, and sorting the inventory list by using the buttons displayed beside the inventory list. The end goal is an easy to use, simple to manage system that allows the user to properly keep track of and maintain their inventory stock.

13. History of Work, Current Status, Future Work

a. History of Work

- i. Up to this point in our progress towards completing the project our application has the ability to log in, access the landing page of the application and communicate with a database. The application is connected to a virtual database called MongoDB and has sample data loaded into it. During the development of acceptance test cases, we were able to identify holes in our original design structure and began to add additional functions and catch designs to handle these previously unforeseen situations. As we progressed through this project, we also realized that we would have to switch the format in which we stored the data from a linked list to an array list. We also received feedback that it would be advantageous to add a drop-down menu to our application allowing for an easier less error prone user interaction with the database. While we thought that we had this interaction handled with our catches developed from the acceptance test cases we will definitely heed the advice given. Up to this point in the project schedule we have developed a functioning log in page that has the ability to prompt the user when incorrect credentials are inputted and allow access to the landing page when the correct credentials are inputted. We have a user interface landing page up and running with dummy buttons. We also have the ability to access and manipulate content within our database from the source code of our application but do not have the ability to access and manipulate data in the database from our landing page.

b. Current Status

Currently we have done all of our planning and analysis for the project. Team Java was able to meet our expectations for Demo One and more so. A login page was not a part of our initial implementation, however; due to customer requests a login page has been added to our software. A rough outline of all UI elements have been implemented, a connection to MongoDB has been established, and we are prepared to finish all programming during Demo Two.

c. Future Work

Moving forward in our development process, we plan to first add functionality to all of the UI elements. While the UI has been created, it is not yet fully functional. Completing UI aspects' will be the first priority when moving forward. After the functionality of the UI is implemented, we plan to make the home page UI more attractive looking. Any leftover things that need to be implemented will be added here. After each major goal is met (i.e. add functionality to UI elements), our team will all peer review the new code. Once all major goals have been met, if there is still remaining time in our deadline, we will add additional functionality as sees fit.

14. Key accomplishments

- Developed a operational GUI login page
 - Developed a GUI landing page
 - Successfully connected to MongoDB Atlas
 - Successfully upload to and manipulate data in MongoDB from our application.
- a. Possible future directions
- i. With the ease of use that this inventory management program can provide to its potential users we can totally see posting the code as a bundle to be downloaded and implemented. Inviting users to provide feedback on their experience with this application and then further developing the application based on the constructive feedback of our users. Our project handled in this way would provide a mutual party benefit. The users of our application would get a great inventory management program, initially developed by some college kids with near no inventory management experience, but through time it will have been, in a very real sense, developed by the professional users of this product. We would benefit by having an extremely robust project to add to our portfolio with a legend of updates and versions to be spoken to during a future interview.
 - ii.

15. References:

- a. **Sequence diagram design**
 - i. <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>
- b. **Network Protocol**
 - ii. <https://www.omnisecu.com/tcpip/transmission-control-protocol-tcp.php>
- b. **Swing**
 - i. https://www.youtube.com/watch?v=eyRIW_LGe-Q
 - ii. <https://www.youtube.com/watch?v=7HChmKwOUzw>
- c. **MongoDB**
 - i. <https://www.youtube.com/watch?v=GiNMF18wnlq>
 - ii. <https://www.youtube.com/watch?v=-56x56UppqQ>
 - iii. <https://www.youtube.com/watch?v=pDqTmTVG9o>
 - iv. <https://www.youtube.com/watch?v=WlhhzbbF170>
 - v. <https://www.youtube.com/watch?v=nOUqH4oFjlk>
 - vi. <https://www.mongodb.com/java>
- d. **User Login**
 - i. https://www.youtube.com/watch?v=QwQuro7ekvc&ab_channel=Amigoscode