# Inventory Tracking Program

By: Team Java
Jordan Hidalgo, Jorge Sanchez,
Raymond Duenas, Zackary Bair
Stan State
SWE CS 4800
09/15/2021

https://github.com/BairZackary/SE_TermProject

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

| **Team Member** | **Contribution** |
|---|---|
| Jordan Hidalgo | - Captured System Requirements<br>- Captured Solution Description<br>- Assisted in Problem Identification<br>- Assisted in Identifying Problem Solution<br>- Assisted in Developing Plan of work<br>- Assisted in Developing Timeline<br>- Assisted in Design of Prototype<br>- Captured Domain Analysis<br>- Developed Mathematical Model<br>- Assisted in Developing Sequence Diagram<br>- Assisted in Developing Use Case Diagram<br>- Assisted in Developing Acceptance test cases<br>- Assisted in Developing Project Size Estimation |
| Jorge Sanchez | - Captured Problem Description<br>- Assisted in Problem Identification<br>- Assisted in Identifying Problem Solution<br>- Assisted in Developing Plan of work<br>- Assisted in Developing Timeline<br>- Introduced Project Idea to Team<br>- Assisted in Design of Prototype<br>- Assisted in Developing Domain Analysis<br>- Developed Mathematical Model<br>- Assisted in Developing Sequence Diagram<br>- Assisted in Developing Use Case Diagram<br>- Assisted in Developing Acceptance test cases<br>- Captured Project Size Estimation |
| Zackary Bair | - Ensured Plan of Work Execution<br>- Captured Meeting minutes<br>- Captured Project PowerPoint<br>- Assisted in Problem Identification<br>- Assisted in Identifying Problem Solution<br>- Assisted in Developing Plan of work<br>- Assisted in Developing Timeline<br>- Assisted in Design of Prototype<br>- Assisted in Developing Use Case Diagram |

| | |
|---|---|
| | - Created use cases 2, 3, and 6<br>- Captured Sequence Diagram<br>- Assisted in Developing Project Size Estimation<br>- Keeps table of contents updated<br>- Keeps track of meetings while I am pleasant<br>- Traceability matrix<br>- Created Gantt chart |
| Raymond Dueñas | - Captured Timeline and Sprints<br>- Presented Project Proposal<br>- Assisted in Problem Identification<br>- Assisted in Identifying Problem Solution<br>- Assisted in Developing Plan of work<br>- Assisted in Developing Timeline<br>- Captured Low Fidelity Prototype<br>- Assisted in Developing Domain Analysis<br>- Developed Mathematical Model<br>- Assisted in Developing Sequence Diagram<br>- Captured Use Case Diagram<br>- Assisted in Developing Acceptance test cases<br>- Assisted in Developing Project Size Estimation<br>- Assisted in Developing Gantt chart |

## TABLE OF CONTENTS

## 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.

1. **<u>Customer problem statement:</u>**

**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. |

**b.** Non-functional requirements:

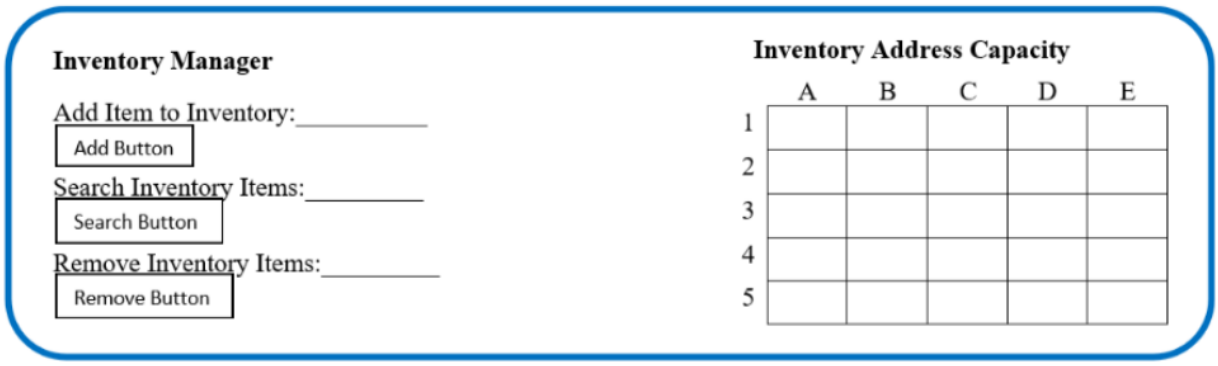
| NF-REQ-X | Requirement | PW | Brief Description |
|---|---|---|---|
| NF-REQ-1 | Simple, accessible interface | 7 | Interface should be simple and easy to use, with a maximum response time of 4 seconds. |
| NF-REQ-2 | Reliable | 10 | Program should be able to save data in response to a crash and reboot 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

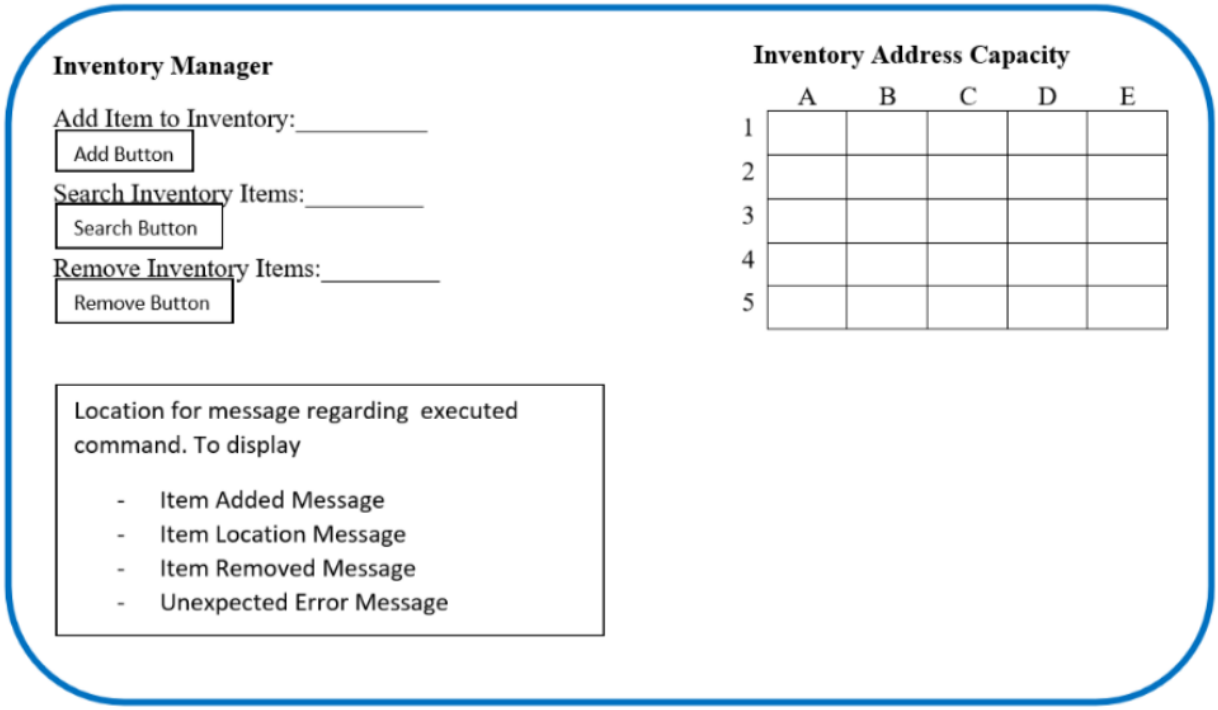
| UI-REQ-X | Requirement | PW | Brief Description |
|---|---|---|---|
| UI-REQ-1 | Display amount of item in inventory. | 8 | Program should be able to display the amount of an item in inventory. |
| UI-REQ-2 | Ability to add, sell, search for, and remove items | 10 | Users should be able to add, sell, 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

Landing Page




Page after command execution




**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**

---

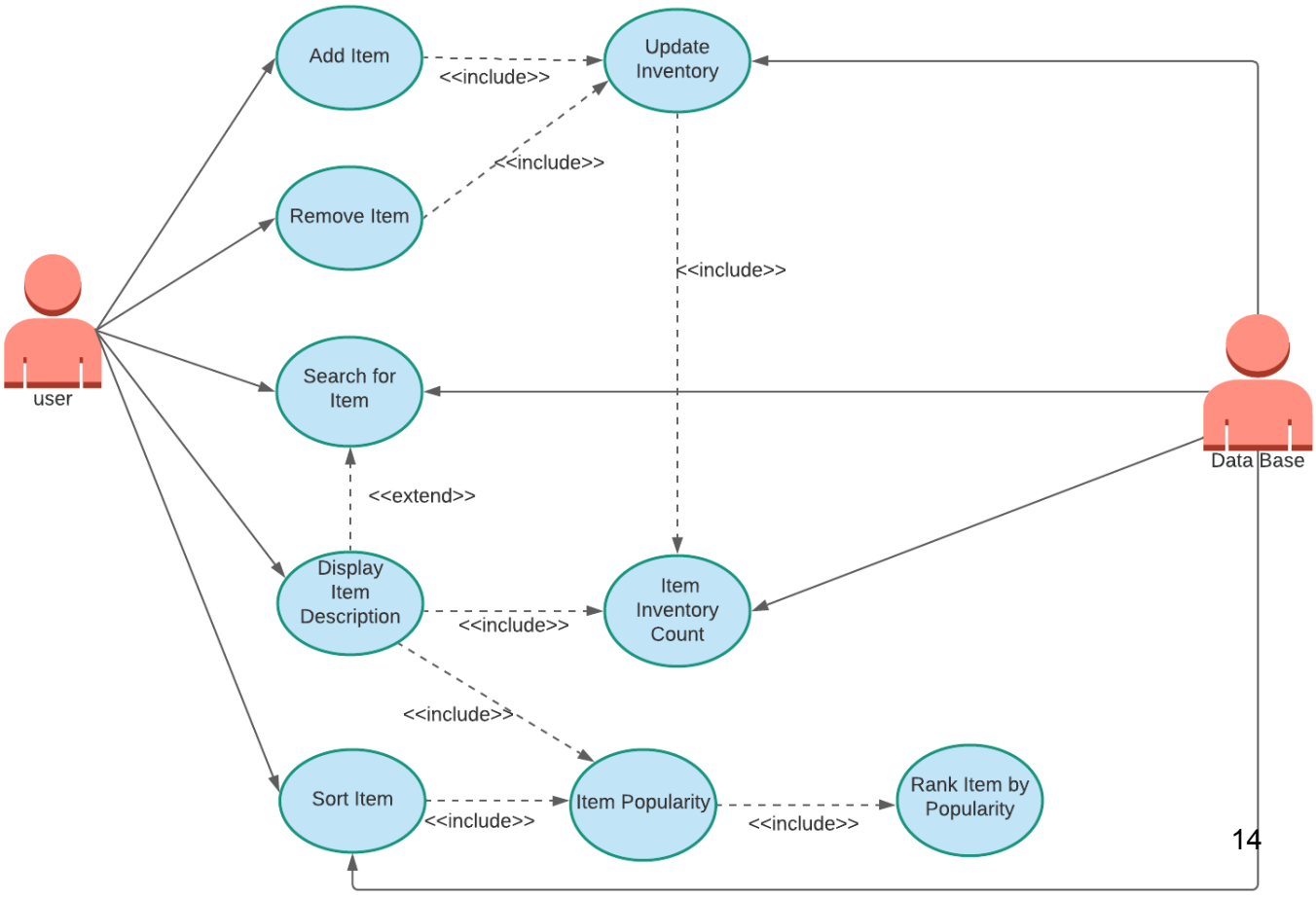
**Actors: User**

**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.

### 3.3 Use Case Diagram:

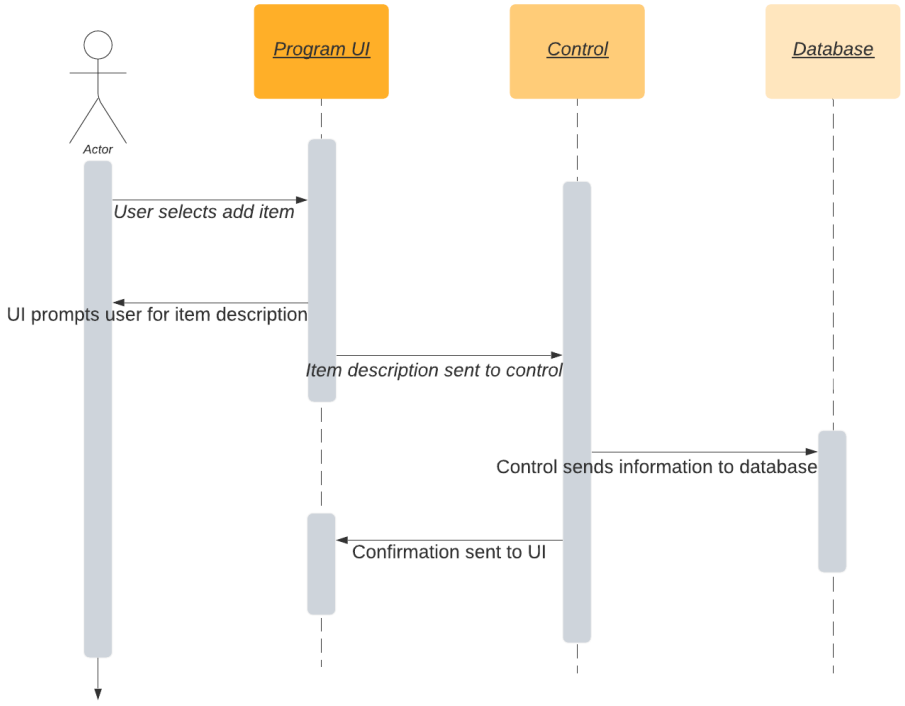


### 3.4 Traceability Matrix:

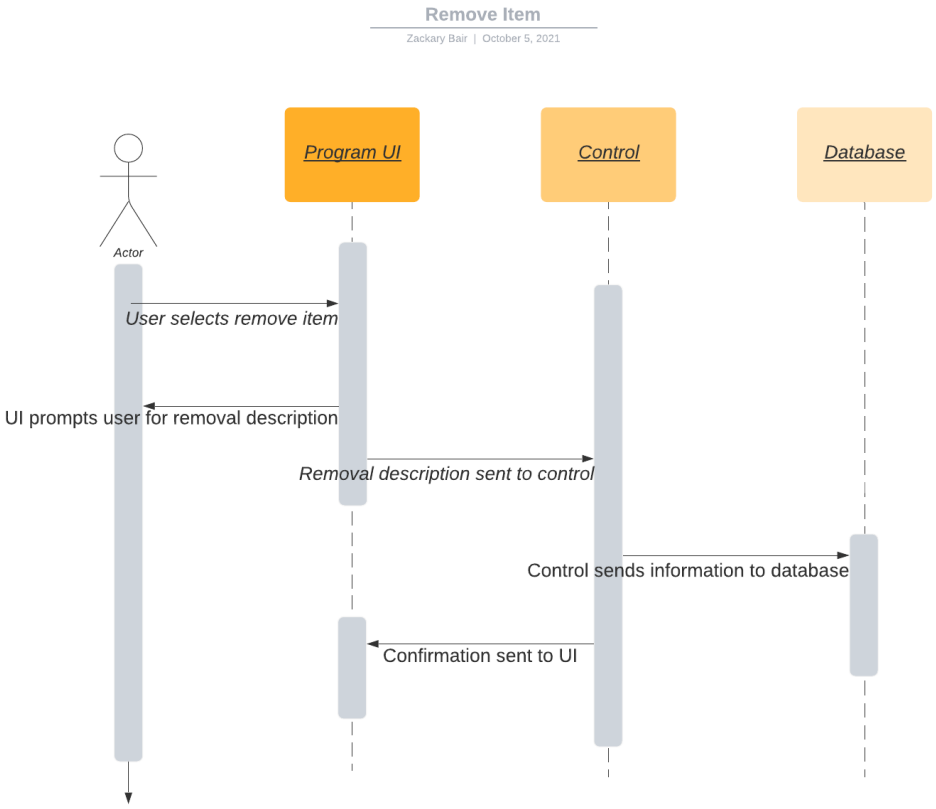
| Req | PW | UC1 | UC2 | UC3 | UC4 | UC5 | UC6 |
|---|---|---|---|---|---|---|---|
| 1 | 10 | X | | | | | |
| 2 | 8 | | X | | | | |
| 3 | 9 | | | | | X | X |
| 4 | 5 | | X | | | | |
| 5 | 4 | | | X | | | X |
| 6 | 7 | | | X | X | | |
| 7 | 3 | | | X | X | | |
| **Max PW** | | 10 | 8 | 7 | 7 | 9 | 9 |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

### 3.5 Sequence Diagrams:

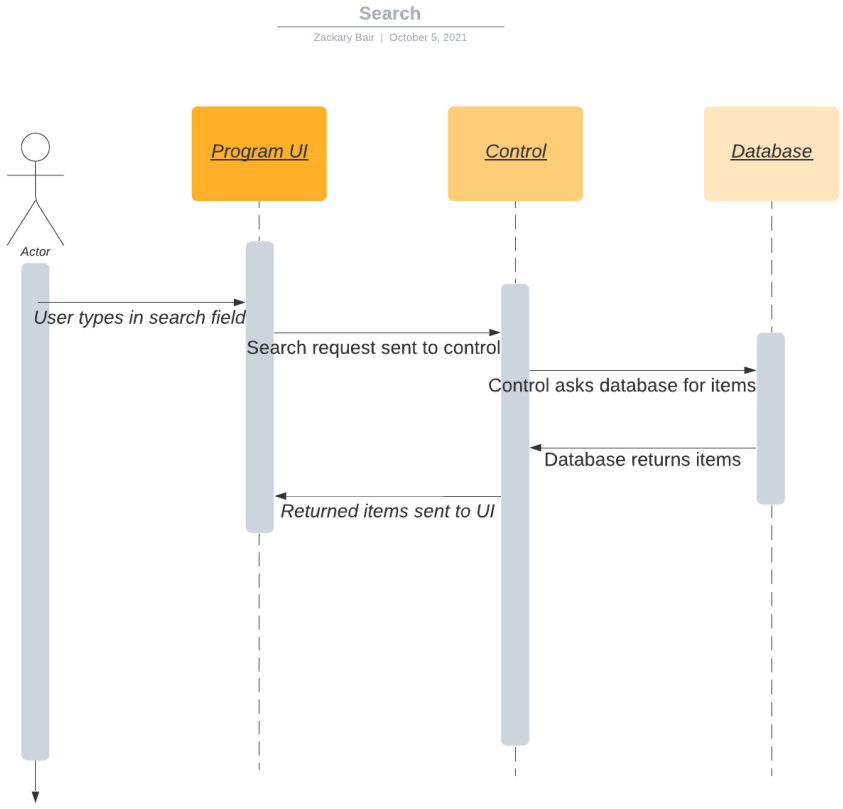
#### 3.5.1 Add Item Use Case



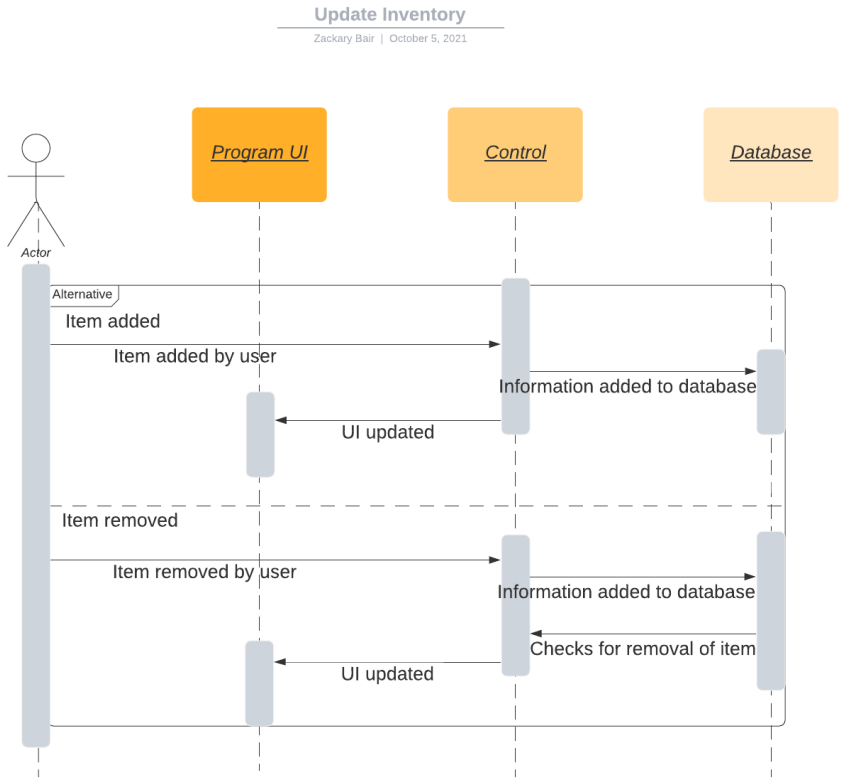
### 3.5.2 Remove Item Use Case



### 3.5.3 Search Use Case



3.5.4 Update Inventory Use Case




## 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

## 4.2 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.

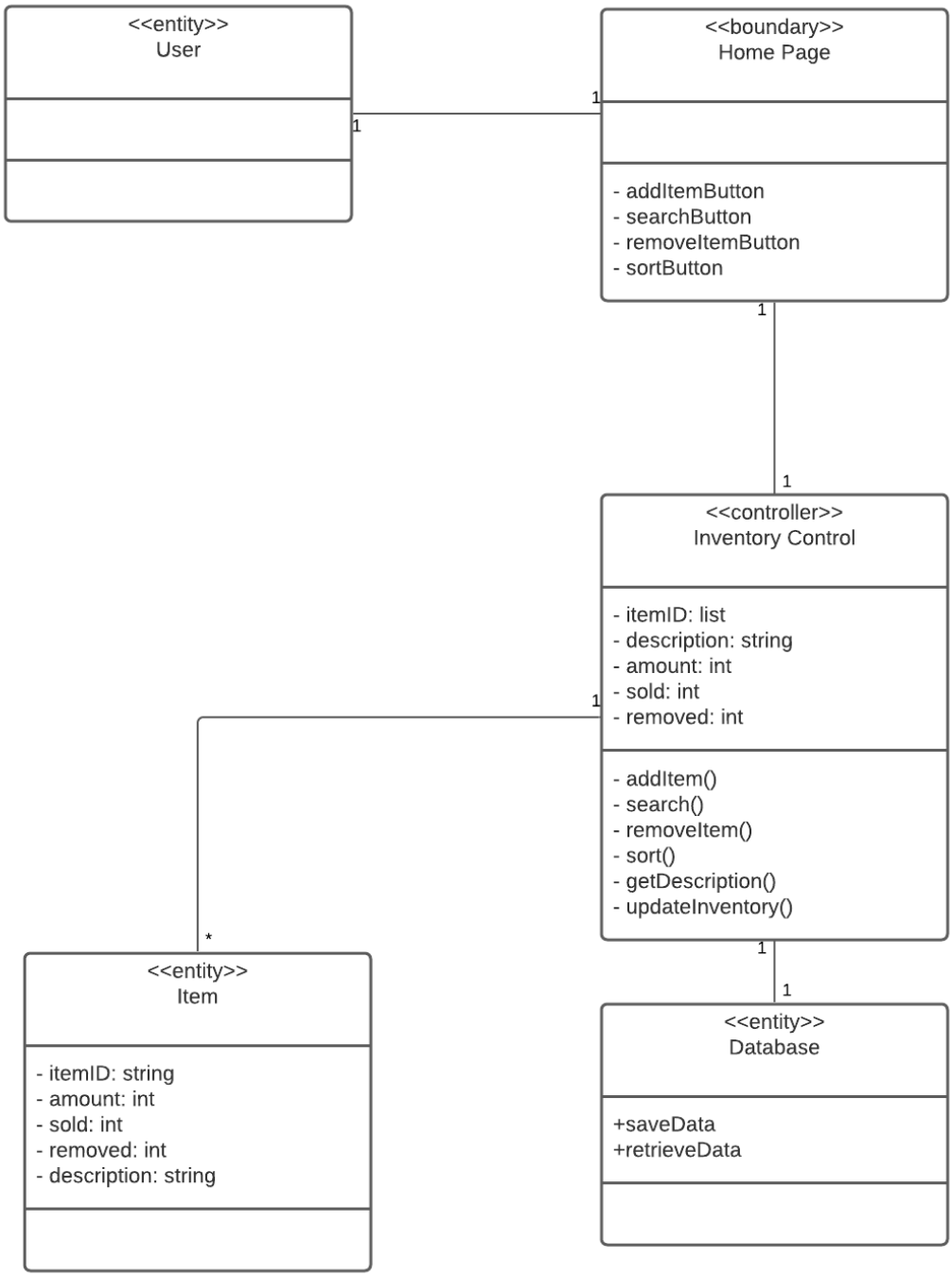
4.2.1.2 **Search for an Item:** Once items have been added to the system, searching for an item is the next logical step for a user, to confirm that items have been added and that the amount displayed is correct. This “Search” button will be placed in the same area below the “Add” button. Clicking this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

## 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 | 35 |
|---|---|
| Add item | 5 |
| Remove item | 5 |
| Search | 5 |
| Sort | 5 |
| Item description | 5 |
| User | 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<br>**Use Case Tested:** UC-1, “Add Item”<br>**Pass/fail Criteria:** The test passes if the user enters a valid Inventory ID number.<br><br>**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<br>**Use Case Tested:** UC-1, “Add Item”<br>**Pass/fail Criteria:** The test passes if the user enters a valid Inventory ID number.<br><br>**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.<br><br>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 to input correct item code or input “new” in the field<br><br>System prompts the user that the imputed item has been created in the inventory log. |

| | |
|---|---|
| **Test- Case Identifier:** TC-1.03<br>**Use Case Tested:** UC-1, “Add Item”<br>**Pass/fail Criteria:** The test passes if the user enters a valid Inventory ID number.<br><br>**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 name, item description and item inventory ID but does not follow inventory ID convention.<br><br>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 ID does not follow Inventory ID format .<br><br>System prompts the user that the imputed item has been created in the inventory log. |

| | |
|---|---|
| **Test- Case Identifier:** TC-1.04<br>**Use Case Tested:** UC-1, “Add Item”<br>**Pass/fail Criteria:** The test passes if the user enters a valid Inventory ID number.<br><br>**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 description but leaves Item name blank.<br><br>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 can not be added without the Item name.<br><br>System prompts the user that the imputed item has been created in the inventory log. |

| | |
|---|---|
| **Test- Case Identifier:** TC-1.05<br>**Use Case Tested:** UC-1, “Add Item”<br>**Pass/fail Criteria:** The test passes if the user enters a valid Inventory ID number.<br><br>**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.<br><br>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 can not be added without item description.<br><br>System prompts the user that the imputed item has been created in the inventory log. |

| |
|---|
| **Test- Case Identifier:** TC-2.01<br>**Use Case Tested:** UC-2 “Remove item”<br>**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<br><br>**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. |

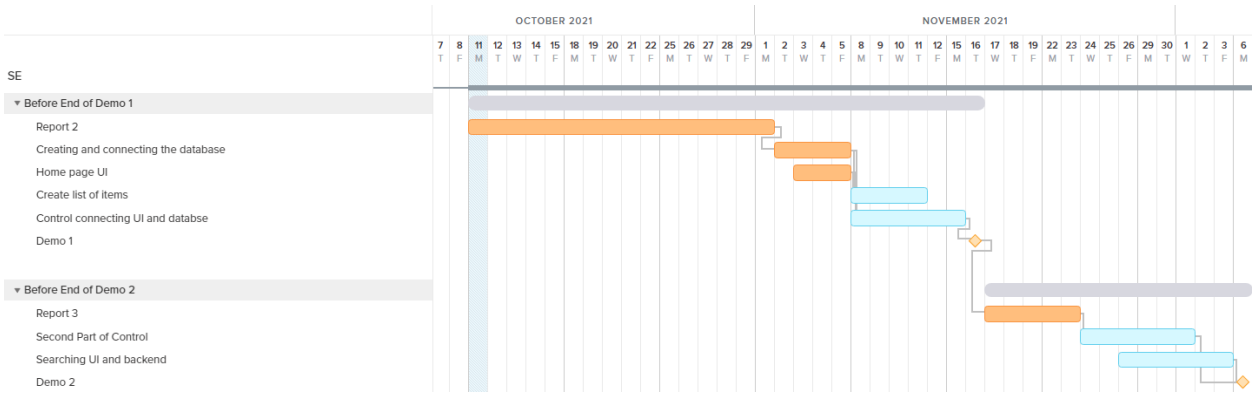
| | |
|---|---|
| **Test- Case Identifier:** TC-4<br>**Use Case Tested:** UC-4 “search”<br>**Pass/fail Criteria:** The test passes if the item inputted into the search bar is found. The test fails if the item was not found in the database<br><br>**Input Data:** Alphanumeric code | |
| **Test Procedure:** | **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. |

**8. Plan of Work:**

8.1 Gantt Chart:

| **ID** | **Task Name** | Predecesor | Time Estimate | | | Expected Time ($T_e$) |
|---|---|---|---|---|---|---|
| | | | Opt. (0) | 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 |




8.2 Ownership: (TBD)

8.2.1. Jordan Hidalgo:

8.2.2. Jorge Sanchez:

8.2.3. Raymond Duenes:

8.2.4 Zackary Bair:

**9. References: (TBD):**