Inventory Management Program
for UTM CSCI 352 Spring 2018

Brayden Faulkner
Johnathon Sulcer

Abstract

With this program the goal is to provide an inventory management system that can be used by anyone for a variety of types of inventory. The program will allow the user a baseline level of functionality through which they can interact with their inventory. The program will be robust, easy to use, and understand to those who may not be familiar with databases or similar software programs.

1. Introduction

With this program we want to help the user track the contents of their inventory as efficiently as we can and provide features to aid them. This means we will have a user specific database filled with the user's inventory and information provided by the user, such as what the inventory is for. This program will be designed with the intention of being usable by virtually anyone who has an inventory to manage. At the base level users will be able to add, remove, and modify items in their inventory. The program will be easy to use, and setup in an easy to learn fashion.

1.1. Background

The main reason we decided to do this program is we thought about all the similar programs we had done in the past, and decided to adapt the idea behind some of them into a usable end product. We also thought of all the places that still use ineffective methods to keep track of their inventory due to fear of technology or the view that modernizing would be too expensive. By simple we mean that even someone with next to no computer skills would be able to use the program with much difficulty. By efficient we mean the program should be able to run smoothly and without crashing in any conceivable case.

1.2. Challenges

Two of the biggest problems we anticipate are dealing with working with the databases and the login procedure. Since we plan to use a login, We'll need the login procedure to store what the client has in their inventory and what kind of inventory they have. The databases are sort of an x factor here as since neither of us have extensive experience with databases, so we're not really sure what to expect.

2. Scope

As a minimum, we would like the program to be able to track each user's inventory, list its contents, and be able to add and remove items from the inventory as well. The user's inventory will be stored in a database, and each user should have their own table. Each item in the inventory should store the name, the amount, and id of the item. The user should also be able to modify the amount of items that are already in their inventory. As a stretch goal, we will add in features specialized for retail stores, including tracking profits and the dates items were received. There will also be some features specialized for warehouses, like tracking where and when items are being shipped. Finally, we would like to add in features specialized for a library, such as tracking who has what book and when they should return it. As another stretch goal we would like to add a basic setting screen to change simple things such as text size and background color.

2.1. Requirements

The program should give the user the ability to add, remove, and modify the amount of items in his or her inventory. Each item in the inventory should store a string for the name, an int for the amount, and a string for the id. Users should only be able to access their inventory after logging in to their account. The accounts should be also be stored in a database. Users that have not used the program before should be able to create a new account. Each account should have its own separate corresponding table in the database that can only be accessed by that account.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Basic Inventory	Admin	Hard	1
2	Retail Inventory	Admin	Med	2
3	Warehouse Inventory	Admin	Med	2
4	Library Inventory	Admin	Med	2
5	Visual Settings/Appeal	Normal User	Easy	3

TABLE 1. INVENTORY PROGRAM USE CASES

2.1.1. Functional.

- User needs to have a private inventory
- User needs to be able to save and load the correct inventory every time they login
- User needs to be able to modify their inventory.

2.1.2. Non-Functional.

- Security – each user should only be allowed to access their account with the proper username and password
- Login – each user should have unique username and password

2.2. Use Cases

Use Case Number: 1
Use Case Name: Remove Item
Description: A user wants to remove an item from his or her inventory

1) User logins to his or her account
2) User navigates to the remove window
3) User enters in the name of the item he or she would like to remove and hits enter
4) User inventory is updated to reflect the removal of said item.

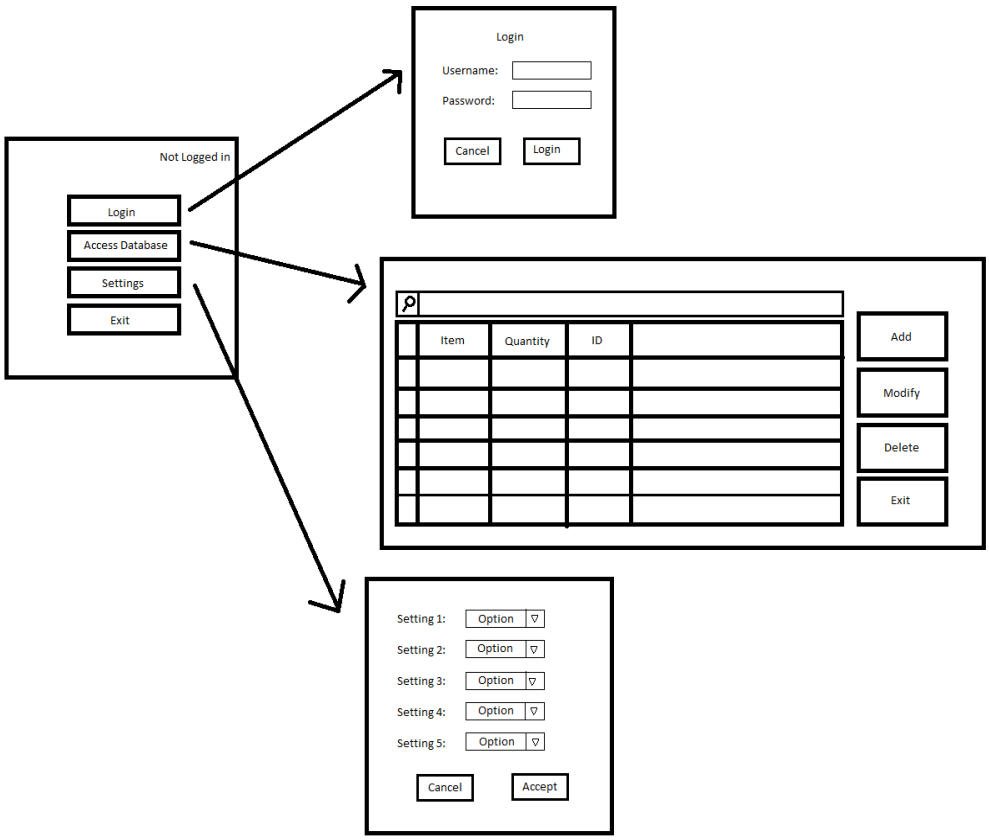
Termination Outcome: The user no longer has the requested item stored in his or her inventory.
Use Case Number: 2
Use Case Name: Add Item
Description: A user wants to add an item to his or her inventory

1) User logs in to his or her account
2) User navigates to the add window
3) User enters in the name, amount, id of the item they wish to add, and hit enter
4) User's inventory is updated to reflect the added item

Termination Outcome: The user's inventory now contains the requested item
Use Case Number: 3
Use Case Name: Visual Settings/Appeal
Description: The user may change visual settings to make the program more usable or appealing.

1) User logs in to his or her account
2) User navigates to a settings page.
3) User changes to desired visual appeals.
4) Program visuals are updated to reflect the desired visual changes for the program.

2.3. Interface Mockups



3. Project Timeline

Design: The design of the program, including interface mockups and UML's was completed by March 15.

Implementation: The program all front end functionality by April 15. The back end functionality, loading data from and writing data to the databases, was functional by April 25.

Verification: We verified that the program was working as intended on April 25. Maintenance: The program underwent maintenance from April 25 to April 29.

4. Project Structure

The front end of the program is made up of a main window that provides access to various other windows each in charge of a specific function of the program. For example adding the an item to the list requires the user to navigate to the add window then enter the information about the item. The list is stored in a database, with each member of the list being stored in its own column. Each user has their own table in the database that is created when the user registers. The only time the database and the front end of the program interact is when the user either logs in or exit the program. When the user logs in the list is filled with the information from the database, when the user logs out the list is written out to the database so that the updated information can be loaded next time.

4.1. UML Outline

4.2. Design Patterns Used

1 Decorator pattern used to modify the data grid on the database window. The decorator is made up of a block widget class, a block field class, a block decorator class, and a solo decorator class. The color decorator class takes a block

Figure 1. The idea of how modified databases will be implemented, such as specific databases for a library, retail store, or warehouse, using the decorator design pattern.

widget as an argument, then set its internal instance of the data grid equal to the one in the widget. The color decorator then changes the color of the given data grid to the one requested by the user on the settings window.

2 Factory Pattern used to create new windows that create windows to the users specification. Each window type has its own dedicated factory, for example, addfac creates add windows. When the a instance of a factory is created, any information the factory or the window it is creating will need is passed as an argument into the factory's constructor. When build window is called a new instance of the desired window type is created, then the background color is changed to the color requested by the user on the settings window.

5. Results

The first thing that was achieved was the design of the windows was completed, allowing us to see what kind of interface we needed to design our code around. All of the basic functionality, such as being able to add, modify, and remove items, was achieved early into the programs construction. Saving and loading the inventory to/from the database was implemented in the last development sprint before presentation. While all of our goals were achieved, many were harder to implement than we had anticipated. We did not have the time to implement any of our stretch goals other than the ability to change the background color of the window. The program came together fairly well, but what we ended up with changed from what we had initially designed through the development process. The interface over went a major overhaul towards the end of the development cycle, changing almost entirely from what we had initially intended it to look like.

5.1. Future Work

We would like to implement more of our stretch goals, such as implementing the specialized features for retail stores and libraries. We would also like to be able for our users to upload their databases to the internet so they could access it anywhere they desire.