Warehouse Management System: Final Project Report

Intro to C Programming
CSCI-1110-01

Can You C My Screen

University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING, West Haven, CT

Submitted To:
Dr. Reza Sadeghi

Spring 2021

Final Project Report of Warehouse Management System

Team Name

Can You C My Screen

Team Members

1. Arely J. Parra López aparr3@unh.newhaven.edu (Team Head)
2. Alexander Vita avita3@unh.newhaven.edu (Team Member)
3. Kamryn Hammond khamm1@unh.newhaven.edu (Team Member)
4. Jarred Crystal jcrys1@unh.newhaven.edu (Team Member)
5. Sophie Ross sross7@unh.newhaven.edu (Team Member)

Roles of Team Members

1. Arely J. Parra López
    a. Allowing the user to view, request, and save items from the Warehouse Management System as well as providing the admin and guest user an exit function.
2. Alexander Vita
    a. Adding, deleting, and editing items with varied details (i.e., Type, Stored Time, ID, etc) from Warehouse Management System.
3. Kamryn Hammond
    a. Creating log-in page for admin and guest users to enter username & password for Warehouse Management System.
4. Jarred Crystal
    a. Generating user-friendly software that provides a welcome page, a menu of all functions that users have access to, and tabular format of all requested information from the Warehouse Management System.
5. Sophie Ross
    a. Allowing admin to view the list of borrowing requests as well as accept/reject borrowing requests made by guest users in the Warehouse Management System.

Table of Contents

Table of Lists

WMS Item Inventory

Inventory Format (Vertically): Item ID, Item Type, Item Name, Item Stored Time, Item Pick Out Time, Item's Provider Name, Item Quantity, Item's Place, Item's Price.

```
1234
Book
Stewart_Little
05/15/19_05:55
05/18/19_21:25
B&N
25
Shelf_5
$10.00
----------
5678
Food
Fiji_Apple
11/15/15_06:35
05/24/19_20:25
Dole
15
Shelf_7
$15.00
----------
9876
Book
Hello_Again
01/01/20_10:41
08/05/20_16:57
B&N
31
Shelf_1
$5.00
----------
```

Figure 1: WMS Item Inventory List from "inventory.txt"

WMS Admin & Guest User Data

User Format (Vertically): User's Encryption Key, Username, Password.

```
1  Arely
2  aparr3
3  1234
4  ----------
5  Alex
6  avita3
7  5678
8  ----------
9  Kamyrn
10 khamm1
11 9876
12 ----------
13 Jarred
14 jcrys1
15 5432
16 ----------
17 Sophie
18 sross7
19 1012
20 ----------
```

Figure 2: WMS Admin & Guest User List from "LogIn.txt"

WMS Guest User's Favorite Items List

Favorite Items Format (Vertically): Item ID, Item Type, Item Name, Item's Provider Name, Item Quantity, Item's Place, Item's Price.

```
1  1234
2  Book
3  Stewart_Little
4  B&N
5  25
6  Shelf_5
7  $10.00
8  ----------
9  5678
10 Food
11 Fiji_Apple
12 Dole
13 15
14 Shelf_7
15 $15.00
16 ----------
```

Figure 3: WMS Guest User's Favorite Items List from "List.txt"

WMS Guest User Item(s) Request

Request Format (Vertically): Item ID, Item Type, Item Name, Item's Provider Name, Item Quantity, Item's Place, Item's Price, Borrowing Time.

```
1234
Book
Stewart_Little
B&N
1
Shelf_5
$10.00
2_weeks
----------
```

Figure 4: WMS Guest User's Request List from "Requests.txt"

WMS Guest User Item(s) History View

History Format (Vertically): Item ID, Item Type, Borrower's Name, Taken Out (when the item was taken from the WMS), Returned (when the item was returned to the WMS).

```
1234
Book
Arely_Parra
04/01/2021
04/20/2021
----------
5678
Food
Sophie_Ross
03/28/2021
04/01/2021
----------
```

Figure 5: WMS Guest User's Item's History from "History.txt"

WMS Accepted/Denied Request Lists

Accepted/Denied Format (Vertically): Item ID, Accept (y/n), Deny (y/n).

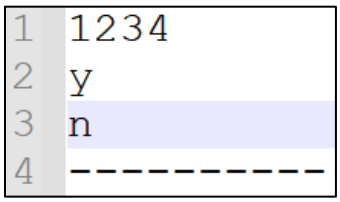

Figure 6: WMS Accepted or Denied Guest User's Requests from "acceptDeny.txt"

Table of Text Files

List of Login Information

A separate text file, “LogIn.txt”, that lists user information for either administrator or guest user: encryption key, username, and password.

tempLogin File

A separate text file, “tempLogin.txt”, that allows the administrator to edit or delete a user’s information from the WMS in a temporary text file prior to implementing the changes into “LogIn.txt”.

List of Item Inventory in the WMS

A separate text file, “inventory.txt”, that lists the WMS inventory. It involves all items and their associated details of Item ID, Item Type, Item Name, Item Stored Time, Item Pick Out Time, Item’s Provider Name, Item Quantity, Item’s Place, and Item’s Price.

tempinventory File

A separate text file, “tempinventory.txt”, that allows the administrator to edit or delete an item and its associated details from the WMS in a temporary text file prior to implementing the changes into “inventory.txt”.

List of Guest User’s Favorite Items from the WMS

A separate text file, “List.txt”, that lists the items that Guest User has favorited after looking into the WMS inventory. This list includes items with their associated details of Item ID, Item Type, Item Name, Item’s Provider Name, Item Quantity, Item’s Place, and Item’s Price.

List of Guest User’s Request of Items from the WMS

A separate text file, “Requests.txt”, that lists the items that Guest User wants to request after looking into the WMS inventory. This list includes items with their associated details of Item ID, Item Type, Item Name, Item’s Provider Name, Item Quantity, Item’s Place, Item’s Price, and Borrowing Time.

List of WMS Item’s Borrowing/Purchase History

A separate text file, “History.txt”, that lists the items from the WMS inventory along with their respective histories regarding borrowing/purchase for the Guest User. This list includes items with their associated details of Item ID, Item Type, Borrower’s Name, Taken Out, and Returned.

List of Accepted and Denied Guest User's Requests

A separate text file, "acceptDeny.txt", that allows the administrator to write down which items from a Guest User's request list have been accepted or denied for borrowing/purchase from the WMS. This list includes items with their associated details of Item ID, Acceptance, and Denial.

README File

A separate file, "README.md", that contains information regarding the WMS program.

Table of Program Menus & Tables

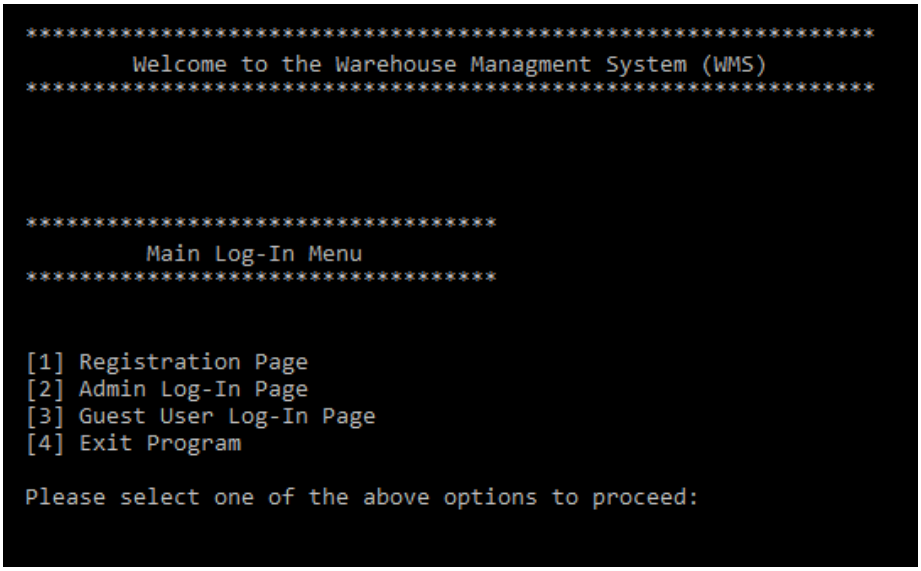

Figure 7: WMS Main Log-in Menu in Command Prompt

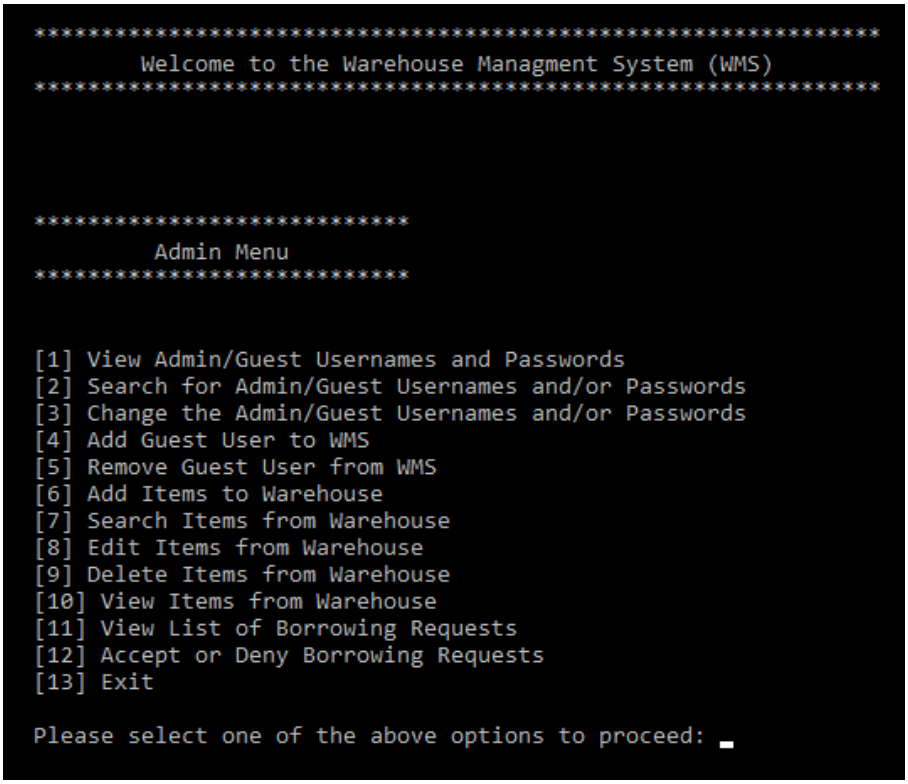

Figure 8: WMS Admin Menu in Command Prompt

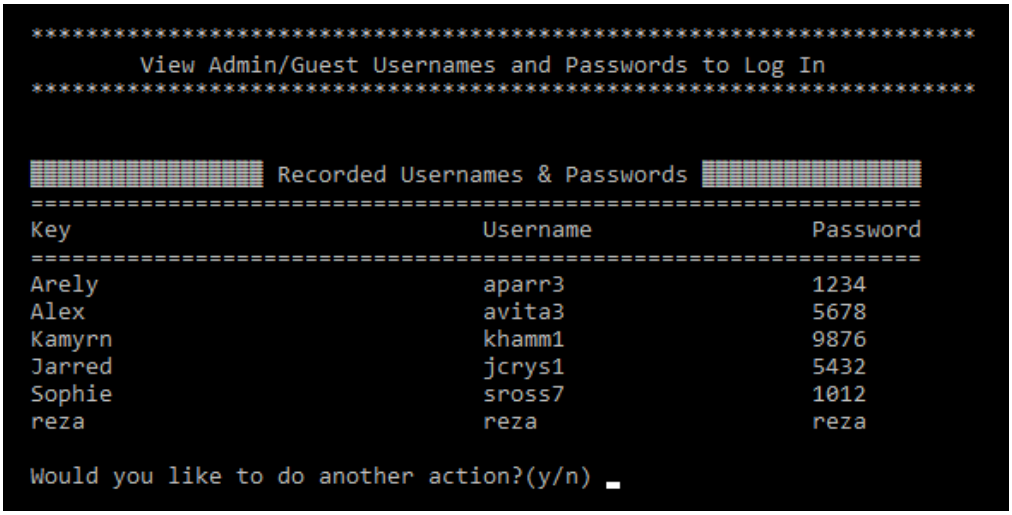

Figure 9: WMS Admin/Guest User Encryption Key, Username, & Password Table in Command Prompt

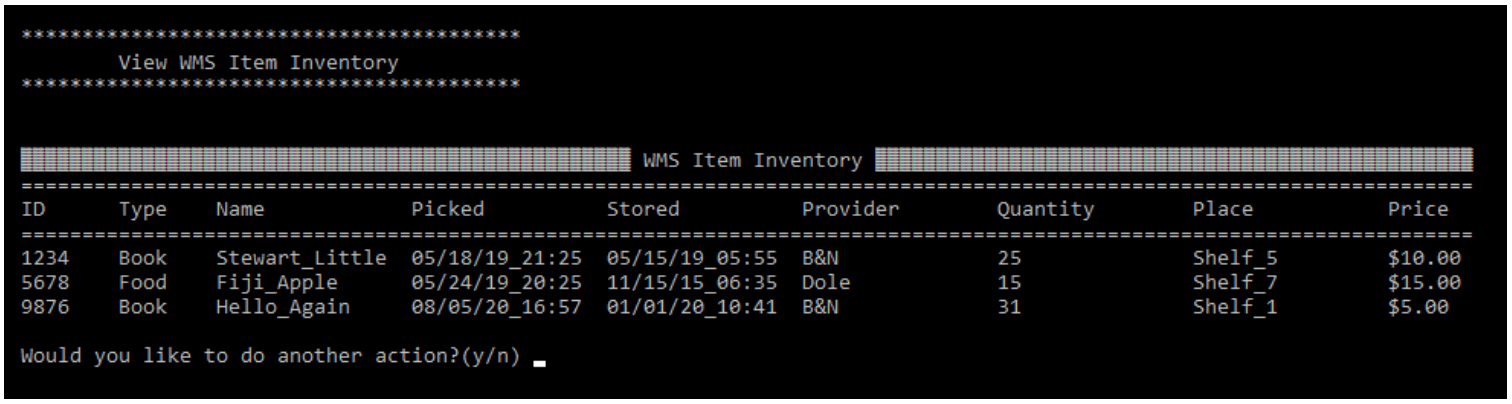

Figure 10: WMS Inventory Table in Command Prompt

Figure 11: WMS Guest User's Borrowing Requests Table in Command Prompt

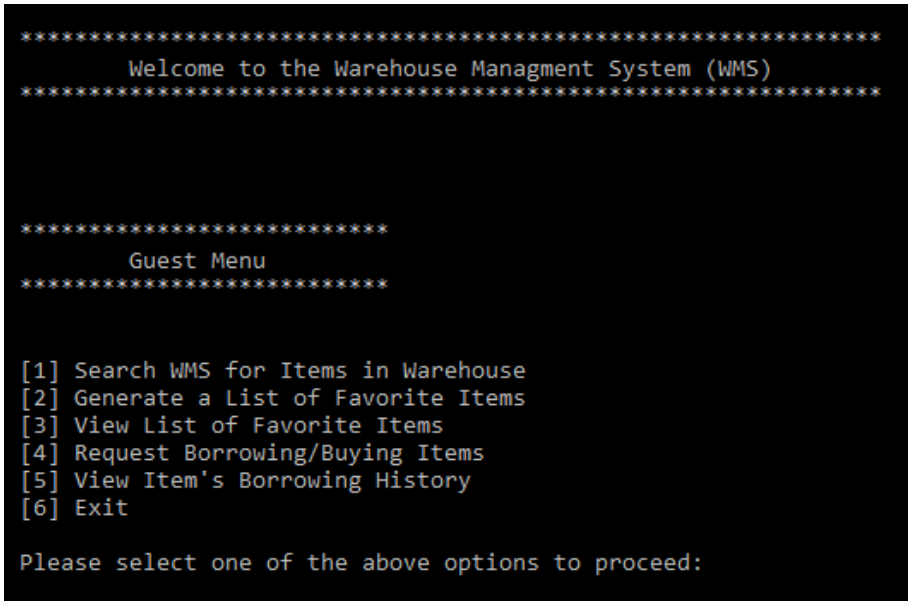

Figure 12: WMS Guest User Menu in Command Prompt

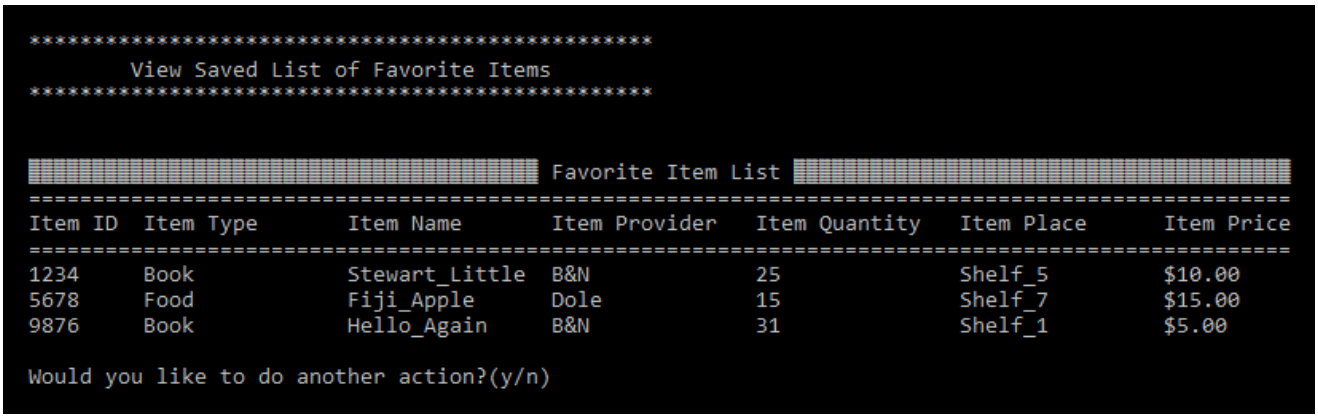

Figure 13: WMS Guest User's Favorite Items List Table in Command Prompt

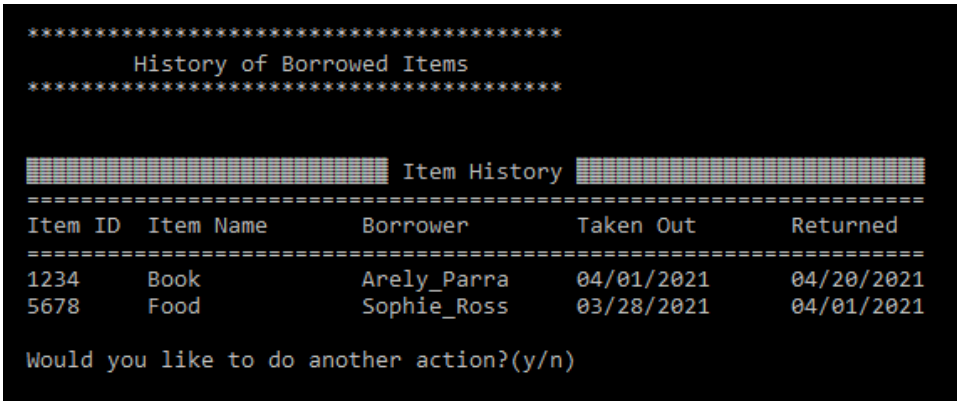

Figure 14: WMS Guest User's View of Item History Table in Command Prompt

Introduction

The Warehouse Management System (WMS) provides an organized, interactive, and user-friendly format of accessing this warehouse's contents. An individual can access the WMS using an administrator encryption key, username, and password or a guest user encryption key, username, and password. Users will be able to search through the warehouse inventory using item details such as Item ID, Item Type, Item Name, Item Stored Time, Item Pick Out Time, Item's Provider Name, Item Quantity, Item's Place, and Item's Price. Additionally, guest users will be able to create a favorite items list from the WMS inventory, request for borrowing/purchase of item(s) for a certain amount of time as well as view the history of item(s) from the WMS inventory.

Project Description

The WMS begins with welcoming the individual to the *Warehouse Management System.* The individual is then prompted to make a choice between registration, administrator login, or guest user login. If an individual is new with no pre-existing account, they can register themselves for an account with the REGISTRATION PAGE. If an individual already exists in the WMS, they can then choose between ADMIN LOGIN PAGE or GUEST USER LOGIN PAGE. It is important to note that if the administrator or guest user were to enter the wrong encryption key for their username and password, a warning message would appear to notify the individual that their encryption key is not in the system and to try again.

If the individual was the administrator and logged into the WMS from the ADMIN LOGIN PAGE, they will be presented with an organized list of commands. The administrator commands include: [1] VIEW ADMIN/GUEST USERNAMES AND PASSWORDS, [2] SEARCH FOR ADMIN/GUEST USERNAMES AND/OR PASSWORDS, [3] CHANGE THE ADMIN/GUEST USERNAMES/PASSWORDS, [4] ADD GUEST USER TO WMS, [5] REMOVE GUEST USER FROM WMS, [6] ADD ITEMS TO WAREHOUSE, [7] SEARCH ITEMS FROM WAREHOUSE, [8] EDIT ITEMS FROM WAREHOUSE, [9] DELETE ITEMS FROM WAREHOUSE, [10] VIEW ITEMS FROM WAREHOUSE, [11] VIEW LIST OF BORROWING REQUESTS, [12] ACCEPT OR DENY BORROWING REQUESTS, and [13] EXIT.

If the individual was a guest user and logged into the WMS from the GUEST USER LOGIN PAGE, they will be presented with an organized list of commands. The guest user commands include: [1] SEARCH WMS FOR ITEMS IN WAREHOUSE, [2] GENERATE A LIST OF FAVORITE ITEMS, [3] VIEW LIST OF FAVORITE ITEMS, [4] REQUEST BORROWING/BUYING ITEMS, [5] VIEW ITEM'S BORROWING HISTORY, and [6] EXIT.

To make the system more user-friendly, there are designated warning messages that will appear if the administrator attempts to add a new item to the warehouse with a pre-existing Item ID number, and if a user's search request returns null items. Furthermore, to ensure that all

information is stored and tracked in an organized fashion, all borrowed requests, list of favorite items, list of encryption keys/usernames/passwords, borrowed history, warehouse inventory, and accepted/denied requests will be logged in text files.

Administrators and guest users will be able to search through the warehouse inventory using item details such as Item ID, Item Type, Item Name, Item Stored Time, Item Pick Out Time, Item's Provider Name, Item Quantity, Item's Place, and Item's Price. Additionally, guest users will be able to create a favorite items list from the WMS inventory, request for borrowing/purchase of item(s) for a certain amount of time as well as view the history of item(s) from the WMS inventory.

List of Key Variables

Log-In Variables:

Jared Crystal:

----------MENU OPTIONS----------

int input

An integer data type that stores the individual's input based on their choice from the Login Menu.

----------MENU SWITCH CASE----------

int choice

An integer data type that stores the user's menu selection from the login menu.

char quit

A character data type that stores information whether the individual would like to perform another action.

Arely J. Parra López:

----------CASE 1----------

struct entity

A user-defined data type that stores encryption key, username, and password when an individual is registering a new account in the WMS.

char encrypt[20]

An array declaration with a character data type that stores the individual's encryption key up to 20 characters.

char username[20]

An array declaration with a character data type that stores the individual's username up to 20 characters.

char password[20]

An array declaration with a character data type that stores the individual's password up to 20 characters.

struct entity new

A user-defined data type that stores the new encryption key, username, and password when an individual is registering a new account in the WMS.

FILE *myfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

char quit

A character data type that stores information whether the individual would like to register another encryption key, username, and password.

Kamryn Hammond:

----------*CASE 2*----------

static int output[2]

A static integer data type that will preserve the value of the output in the memory while the program is running after searching for the individual's Admin encryption key within an array declaration with up to 2 associated details.

char key[100]

An array declaration with a character data type that stores the individual's encryption key up to 100 characters.

char input[100]

An array declaration with a character data type that stores the individual's input up to 100 characters.

FILE *myfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

char myString[100]

An array declaration with a character data type that stores the individual's string up to 100 characters.

int line=0

An integer data type that initializes the WMS login line to zero when looking within the LogIn.txt file.

int find=0

An integer data type that initializes the WMS login find capability to zero when looking within the LogIn.txt file.

int fieldNumber=0

An integer data type that initializes the WMS login field number capability to zero when looking within the LogIn.txt file.

char quit

A character data type that stores information whether the administrator would like to enter the Admin Menu.

----------*CASE 3*----------

static int output[2]

A static integer data type that will preserve the value of the output in the memory while the program is running after searching for the individual's Guest User encryption key within an array declaration with up to two associated details.

char key[100]

An array declaration with a character data type that stores the individual's encryption key up to 100 characters.

char input[100]

An array declaration with a character data type that stores the individual's input up to 100 characters.

FILE *myfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

char myString[100]

An array declaration with a character data type that stores the individual's string up to 100 characters.

int line=0

An integer data type that initializes the WMS login line to zero when looking within the LogIn.txt file.

int find=0

An integer data type that initializes the WMS login find capability to zero when looking within the LogIn.txt file.

int fieldNumber=0

An integer data type that initializes the WMS login field number capability to zero when looking within the LogIn.txt file.

char quit

A character data type that stores information whether the guest user would like to enter the Guest User Menu.

Admin Variables:

Jared Crystal:

----------*MENU OPTIONS*----------

int input

An integer data type that stores the individual's input based on their choice from the Admin Menu.

----------MENU SWITCH CASE----------

int option

An integer data type that stores the user's menu selection from the admin menu.

char quit

A character data type that stores information whether the admin user would like to perform another action.

Kamryn Hammond:

----------CASE 1----------

struct entity

A user-defined data type that stores encryption key, username, and password for an individual's login within the WMS.

char encrypt[20]

An array declaration with a character data type that stores the user's encryption key up to 20 characters.

char username[20]

An array declaration with a character data type that stores the user's username up to 20 characters.

char password[20]

An array declaration with a character data type that stores the user's password up to 20 characters.

struct entity new

A user-defined data type that stores the new encryption key, username, and password in the WMS.

char myString[100]

An array declaration with a character data type that stores the Admin/Guest User's encryption key with the associated username and password as myString up to 100 characters.

int pline=0

An integer data type that initializes the WMS login printing line capability to zero when printing the Admin/Guest User login information found within the LogIn.txt file.

----------CASE 2----------

static int output[2]

A static integer data type that will preserve the value of the output in the memory while the program is running after searching for the user's encryption key within an array declaration with up to two associated details.

char key[100]

An array declaration with a character data type that stores the individual's encryption key up to 100 characters.

char input[100]

An array declaration with a character data type that stores the individual's input up to 100 characters.

FILE *myfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

char myString[100]

An array declaration with a character data type that stores the Admin/Guest User's encryption key with the associated username and password as myString up to 100 characters.

int line=0

An integer data type that initializes the WMS login line to zero when looking within the LogIn.txt file.

int find=0

An integer data type that initializes the WMS login find capability to zero when looking within the LogIn.txt file.

int fieldNumber=0

An integer data type that initializes the WMS login field number capability to zero when looking within the LogIn.txt file.

int pline=0

An integer data type that initializes the WMS login printing line capability to zero when printing the searched for encryption key, username, and password found within the LogIn.txt file.

----------*CASE 3*----------

int* place=search()

Defines the integer data type of place initialized to the search function by pointing to the address of Case 2 of the Admin Menu.

int line=place[1]

An integer data type that initializes the WMS to one when looking within the LogIn.txt file.

int find=place[0]

An integer data type that initializes the WMS to zero when looking within the LogIn.txt file.

char username[20], password[20]

An array declaration with a character data type that stores the user's username and password up to 20 characters.

FILE *invfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

FILE *temp

Defines the name of the temporary text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "tempLogin.txt").

char myString[100]

An array declaration with a character data type that stores the user's encryption key with associated details as myString up to 100 characters.

int pline=0

An integer data type that initializes the WMS login printing line capability to zero when printing the item information found within the tempLogin.txt file.

char quit

A character data type that stores information whether the Admin would like to change another user's encryption key, username, and password.

----------*CASE 4*----------

struct entity

A user-defined data type that stores encryption key, username, and password for an individual's login within the WMS.

char encrypt[20]

An array declaration with a character data type that stores the user's encryption key up to 20 characters.

char username[20]

An array declaration with a character data type that stores the user's username up to 20 characters.

char password[20]

An array declaration with a character data type that stores the user's password up to 20 characters.

struct entity new

A user-defined data type that stores the new encryption key, username, and password in the WMS.

FILE *myfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

char quit

A character data type that stores information whether the Admin would like to add another user's encryption key, username, and password.

----------*CASE 5*----------

int* place=search()

Defines the integer data type of place initialized to the search function by pointing to the address of Case 2 of the Admin Menu.

int line=place[0]

An integer data type that initializes the WMS to zero when looking within the LogIn.txt file.

int find=place[0]

An integer data type that initializes the WMS to zero when looking within the LogIn.txt file.

FILE *originalfile

Defines the name of the text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "LogIn.txt").

FILE *tempfile

Defines the name of the temporary text file containing the WMS's encryption keys, usernames, and passwords with a pointer (*) used to point to the address of the text file from the memory (myfile = "tempLogin.txt").

char myString[100]

An array declaration with a character data type that stores the user's encryption key with associated details as myString up to 100 characters.

int pline=0

An integer data type that initializes the WMS login printing line capability to zero when printing the item information found within the tempLogin.txt file.

char quit

A character data type that stores information whether the Admin would like to delete another user's encryption key, username, and password.

Alexander Vita:

----------*CASE 6*----------

struct entity

A user-defined data type that stores Item ID, Item Type, Item Name, Item Stored Time, Item Picked Out Time, Item Provider, Item Quantity, Item Place, and Item Price for the WMS inventory.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char store[100]

An array declaration with a character data type that stores the item's stored time up to 100 characters.

char pick[100]

An array declaration with a character data type that stores the item's picked out time up to 100 characters.

char provider[100]

An array declaration with a character data type that stores the item's provider up to 100 characters.

char quantity[100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char place[100]

An array declaration with a character data type that stores the item's place up to 100 characters.

char price[100]

An array declaration with a character data type that stores the item's Price up to 100 characters.

struct entity new

A user-defined data type that stores the new Item ID, Item Type, Item Name, Item Stored Time, Item Picked Out Time, Item Provider, Item Quantity, Item Place, and Item Price for the WMS inventory.

FILE *invfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (invfile = "inventory.txt").

char quit

A character data type that stores information whether the Admin would like to add another item to the WMS's inventory.

----------*CASE 7*----------

static int output[2]

A static integer data type that will preserve the value of the output in the memory while the program is running after searching for the Item's ID within an array declaration with up to 2 item associated details.

char key[100]

An array declaration with a character data type that stores the Item's ID up to 100 characters.

char input[100]

An array declaration with a character data type that stores the individual's input up to 100 characters.

FILE *invfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (invfile = "inventory.txt").

char myString[100]

An array declaration with a character data type that stores the individual's string up to 20 characters.

int line = 0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

int find = 0

An integer data type that initializes the WMS inventory find capability to zero when looking within the inventory.txt file.

int fieldNumber = 0

An integer data type that initializes the WMS inventory field number capability to zero when looking within the inventory.txt file.

int pline=0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

----------*CASE 8*----------

int* place=searchItems()

Defines the integer data type of place initialized to the searchItems function by pointing to the address of Case 7 of the Admin Menu.

int line = 0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

int find = 0

An integer data type that initializes the WMS inventory find capability to zero when looking within the inventory.txt file.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char store[100]

An array declaration with a character data type that stores the item's stored time up to 100 characters.

char pick[100]

An array declaration with a character data type that stores the item's picked time up to 100 characters.

char provider [100]

An array declaration with a character data type that stores the item's provider/creator up to 100 characters.

char quantity [100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char place [100]

An array declaration with a character data type that stores the item's storage location up to 100 characters.

char price [100]

An array declaration with a character data type that stores the item's price up to 100 characters.

FILE *invfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (invfile = "inventory.txt").

FILE *tempfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (tempfile = "tempinventory.txt").

char myString[100]

An array declaration with a character data type that stores the individual's string up to 20 characters.

int pline=0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

char quit

A character data type that stores information whether the guest user would like to edit another item to the inventory.

----------*CASE 9*----------

int* place=searchItems()

Defines the integer data type of place initialized to the searchItems function by pointing to the address of Case 7 of the Admin Menu.

int line=place[1]

An integer data type that initializes the WMS to one when looking within the inventory.txt file.

int find=place[0]

An integer data type that initializes the WMS to zero when looking within the inventory.txt file.

FILE *invfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (invfile = "inventory.txt").

FILE *tempfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (tempfile = "tempinventory.txt").

char myString[100]

An array declaration with a character data type that stores the individual's string up to 20 characters.

int pline=0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

char quit

A character data type that stores information whether the guest user would like to remove another item form the inventory.

----------*CASE 10*----------

FILE *invfile

Defines the name of the text file containing the WMS's inventory with a pointer (*) used to point to the address of the text file from the memory (invfile = "inventory.txt").

struct entity

A user-defined data type that stores Item ID, Item Type, Item Name, Item Stored Time, Item Picked Out Time, Item Provider, Item Quantity, Item Place, and Item Price for the WMS inventory.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char store[100]

An array declaration with a character data type that stores the item's stored time up to 100 characters.

char pick[100]

An array declaration with a character data type that stores the item's picked time up to 100 characters.

char provider [100]

An array declaration with a character data type that stores the item's provider/creator up to 100 characters.

char quantity [100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char place [100]

An array declaration with a character data type that stores the item's storage location up to 100 characters.

char price [100]

An array declaration with a character data type that stores the item's price up to 100 characters.

struct entity new

A user-defined data type that stores new Item ID, Item Type, Item Name, Item Stored Time, Item Picked Out Time, Item Provider, Item Quantity, Item Place, and Item Price for the WMS inventory.

c**har myString[100000]**

An array declaration with a character data type that stores the individual's string up to 10000 characters.

int pline=0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

Sophie Ross:

----------*CASE 11*----------

FILE myfile*

Defines the name of the text file containing the WMS's borrow request list with a pointer (*) used to point to the address of the text file from the memory (myfile ="Requests.txt").

struct entity

A user-defined data type that stores Item ID, Item Type, Item Name, Item Stored Time, Item Picked Out Time, Item Provider, Item Quantity, Item Place, and Item Price in the guest user's borrowing request list.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char Provider [100]

An array declaration with a character data type that stores the item's provider/creator up to 100 characters.

char Quantity [100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char Place [100]

An array declaration with a character data type that stores the item's storage location up to 100 characters.

char Price [100]

An array declaration with a character data type that stores the item's price up to 100 characters.

char Length [100]

An array declaration with a character data type that stores the item's requested borrowing time to 100 characters.

struct entity new

A user-defined data type that stores new Item ID, Item Type, Item Name, Item Stored Time, Item Picked Out Time, Item Provider, Item Quantity, Item Place, and Item Price in the guest user's borrowing request list.

char myString[10000]

An array declaration with a character data type that stores the user's string up to 10000 characters.

int pline=0

An integer data type that initializes the item's borrowing request list history printing line capability to zero when printing the item information found within the Request.txt file for the admin user.

----------*CASE 12*----------

struct entity

A user-defined data type that stores Item ID, Accept Item Request, and Deny Item Request from the guest user's borrowing request list.

char ID[20]

An array declaration with a character data type that stores the Item's ID up to 20 characters.

char accept[20]

An array declaration with a character data type that stores the Item's accepted request up to 20 characters.

char deny[20]

An array declaration with a character data type that stores the Item's denied request up to 20 characters.

struct entity new

A user-defined data type that stores new Item ID, Accept Item Request, and Deny Item Request from the guest user's borrowing request list.

FILE *myfile

Defines the name of the text file containing the WMS's accept or deny list with a pointer (*) used to point to the address of the text file from the memory (myfile = "AcceptDeny.txt").

char quit

A character data type that stores information whether the admin would like to accept/deny another guest user's borrowing request list.

Guest Variables:

Jared Crystal:

----------*MENU OPTIONS*----------

int input

An integer data type that stores the individual's input based on their choice from the Admin Menu.

----------*MENU SWITCH CASE*----------

int task

An integer data type that stores the user's menu selection from the guest menu.

char quit

A character data type that stores information whether the guest user would like to perform another action.

Arely J. Parra López:

----------*CASE 1*----------

static int output[10]

A static integer data type that will preserve the value of the output in the memory while the program is running after searching for the Item's ID within an array declaration with up to 10 item associated details.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char input[100]

An array declaration with a character data type that stores the item's ID as an input up to 100 characters.

FILE *myfile

Defines the name of the text file containing the WMS's items with associated details with a pointer (*) used to point to the address of the text file from the memory (myfile = "inventory.txt").

char myString[100]

An array declaration with a character data type that stores the item's ID number with associated details as myString up to 100 characters.

int line=0

An integer data type that initializes the WMS inventory line to zero when looking within the inventory.txt file.

int find=0

An integer data type that initializes the WMS inventory find capability to zero when looking within the inventory.txt file.

int fieldNumber=0

An integer data type that initializes the WMS inventory field number capability to zero when looking within the inventory.txt file.

int pline=0

An integer data type that initializes the WMS inventory printing line capability to zero when printing the item information found within the inventory.txt file for the guest user.

----------*CASE 2*----------

struct entity

A user-defined data type that stores Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Price for a guest user's favorite items list from the WMS.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char Provider[100]

An array declaration with a character data type that stores the item's provider up to 100 characters.

char Quantity[100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char Place[100]

An array declaration with a character data type that stores the item's place up to 100 characters.

char Price[100]

An array declaration with a character data type that stores the item's Price up to 100 characters.

struct entity new

A user-defined data type that stores the new Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place for a guest user's favorite items list from the WMS.

FILE *myfile

Defines the name of the text file containing the Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place for a guest user's favorite items list from the WMS with a pointer (*) used to point to the address of the text file from the memory (myfile = "List.txt").

char quit

A character data type that stores information whether the guest user would like to add another item to their favorite item's list.

----------*CASE 3*----------

FILE *myfile

Defines the name of the text file containing the Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place for a guest user's favorite items list from the WMS with a pointer (*) used to point to the address of the text file from the memory (myfile = "List.txt").

struct entity

A user-defined data type that stores Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Price for a guest user's favorite items list from the WMS.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char Provider[100]

An array declaration with a character data type that stores the item's provider up to 100 characters.

char Quantity[100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char Place[100]

An array declaration with a character data type that stores the item's place up to 100 characters.

char Price[100]

An array declaration with a character data type that stores the item's Price up to 100 characters.

struct entity new

A user-defined data type that stores the new Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place for a guest user's favorite items list from the WMS.

char myString[10000]

An array declaration with a character data type that stores the Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place as myString up to 10000 characters.

int pline=0

An integer data type that initializes the guest user's favorite items list printing line capability to zero when printing the item information found within the List.txt file for the guest user.

----------*CASE 4*----------

struct entity

A user-defined data type that stores Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Price for a guest user's request list from the WMS.

char Type[100]

An array declaration with a character data type that stores the item's type up to 100 characters.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char Provider[100]

An array declaration with a character data type that stores the item's provider up to 100 characters.

char Quantity[100]

An array declaration with a character data type that stores the item's quantity up to 100 characters.

char Place[100]

An array declaration with a character data type that stores the item's place up to 100 characters.

char Price[100]

An array declaration with a character data type that stores the item's Price up to 100 characters.

char Length[100]

An array declaration with a character data type that stores the item's borrowing time up to 100 characters.

struct entity new

A user-defined data type that stores the new Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place for a guest user's request list from the WMS.

FILE *myfile

Defines the name of the text file containing the Item ID, Item Type, Item Name, Item Provider, Item Quantity, Item Place, and Item Place for a guest user's request list from the WMS with a pointer (*) used to point to the address of the text file from the memory (myfile = "Requests.txt").

char quit

A character data type that stores information whether the guest user would like to add another item to their request list.

----------*CASE 5*----------

FILE *myfile

Defines the name of the text file containing the Item ID, Item Name, Borrower Name, Taken, and Returned for a guest user to see an item's borrowing history from the WMS with a pointer (*) used to point to the address of the text file from the memory (myfile = "History.txt").

struct entity

A user-defined data type that stores Item ID, Item Name, Borrower, Taken, and Returned for a guest user to see an item's borrowing history from the WMS.

char ID[100]

An array declaration with a character data type that stores the item's ID up to 100 characters.

char Name[100]

An array declaration with a character data type that stores the item's name up to 100 characters.

char Borrower[100]

An array declaration with a character data type that stores the borrower's name up to 100 characters.

char Taken[100]

An array declaration with a character data type that stores the date and time when an item was taken from the WMS up to 100 characters.

char Returned[100]

An array declaration with a character data type that stores the date and time when an item was returned to the WMS up to 100 characters.

struct entity new

A user-defined data type that stores the new Item ID, Item Name, Borrower, Taken, and Returned for a guest user to see an item's borrowing history from the WMS.

char myString[10000]

An array declaration with a character data type that stores the Item ID, Item Name, Borrower, Taken, and Returned for a guest user to see an item's borrowing history from the WMS as myString up to 10000 characters.

int pline=0

An integer data type that initializes the item's borrowing history list printing line capability to zero when printing the item information found within the History.txt file for the guest user.

List of Functions

Log-In Functions:

Jared Crystal:

int loginMenu(void)

This function allows for new registration of users, and the selection of either the admin or guest login prompt.

void main(void)

This function provides the main interface to the login menu dictating which function will be executed following user input.

Arely J. Parra López:

void registration(void)

This function allows new individuals to create a new account within the WMS by providing information for a new encryption key, username, and password.

Kamryn Hammond:

int* adminLogin(void)

This function allows admin to search for their login from the WMS inventory using their private encryption key.

int* guestLogin(void)

This function allows guest users to search for their login from the WMS inventory using their private encryption key.

Admin Functions:

Jarred Crystal:

int adminMenu(void)

This function provides the menu options within the admin menu.

void admin(void)

This function provides the main interface to the admin menu while dictating which function will be executed following Admin's input.

Kamryn Hammond:

void view(void)

This function provides an organized view of all the Admin & Guest User's encryption keys, usernames, and passwords.

int* search(void)

This function allows the Admin to search the WMS LogIn.txt file for a specific encryption key, username, and password.

void edit(void)

This function allows the Admin to select a specific user from the WMS and edit their login information.

void add(void)

This function allows the Admin to add a new user to the WMS with an encryption key, username, and password.

void deletes(void)

This function allows the Admin to select and delete a user from the WMS.

Alexander Vita:

void addItems(void)

This function allows the Admin to add an item to the inventory while specifying the ID, type, name, stored time, picked time, provider/creator, quantity, storage location, and price.

int* searchItems(void)

This function allows the Admin to search the WMS inventory using assigned characteristics.

void editItems(void)

This function allows the Admin to select an item from the inventory and edit its characteristics.

void deleteItems(void)

This function allows the Admin to select and delete an item from the inventory.

void viewItems(void)

This function provides an organized view of all the items and their characteristics within the inventory.

Sophie Ross:

void viewBorrowing(void)

This function allows the Admin to view the list of borrowing requests from the guest user.

void AcceptDeny(void)

This function allows the Admin to accept or deny the items from the guest user's borrowing list.

Guest User Functions:

Jarred Crystal:

int guestMenu(void)

This function provides the menu options within the guest menu.

void guestMain(void)

This function provides the main interface to the guest menu dictating which function will be executed following Guest User's input.

Arely J. Parra López:

Int* searchWMS(void)

This function allows guest users to search for an item from the WMS inventory using key search identifiers.

void makeList(void)

This function allows guest users to generate a list of favorite items from the WMS inventory using Item ID and other associated item details.

void viewList(void)

This function allows guest users to view the list of favorite items that they generated from the WMS inventory.

void requestItems(void)

This function allows guest users to request to borrow or purchase an item from the WMS inventory.

void viewHistory(void)

This function allows guest users to view the borrowing/purchase history of items from the WMS inventory.

Findings

Jared Crystal:

I learned that some computer programming can be very complicated, and that there are much harder programs out there too write. I also learned that I tend to create a lot of errors when I am writing code which is fine, because I utilize the error function recognizer and go back to correct the code that I have gotten wrong. Overall, I have had some struggles in this class with computer programming, but I have resorted to peers or the internet for guidance when I am not fully understanding something when I am coding.

Kamryn Hammond:

My main takeaway from this project is that coding takes a lot of hard work and focus. I also learned how functions and variables work together within the program to create a working code. It was very interesting to learn the ins and outs of stuff that I use on the daily. I did struggle in this class and during this project, but I gained helpful tips and tricks for if I ever encounter coding in the future.

Alexander Vita:

Creating this program with my group solidified that teamwork and organization are very important when programming large projects. On countless occasions we would share and evaluate each other codes thus improving on our processes and our programs effectiveness. In addition to our teamwork, organization played a big role in breaking this project up into smaller more manageable pieces for everyone.

One specific part of programming that I learned during this project is file handling. File handling played a big role in this project as we stored all our data in .txt files. This means that we constantly had to evaluate and edit our files while maintaining what was already within them. This took an extraordinary amount of time as we all needed to learn how to search, edit, and delete specific lines in a file without reformatting them.

Sophie Ross:

My takeaway from this assignment was to learn how to work in a group, learn how to be adaptable, and pay close attention to detail to code. I learned how to manipulate text files with

code and how to code more efficiently for user and programmer ease. I also gained knowledge by learning from other students and how to troubleshoot with a group of people.

Arely J. Parra López:

During the progression of creating the WMS program, I learned that certain functions are required to be placed at the end or at the beginning of a C file in order to allow the entire program to run. This was something I learned when attempting to place the Admin Menu functions and the Guest User Menu functions within the same C file as the Login Menu functions.

Another thing I learned was the ability to call C files in the same manner as header files. To lessen the complexity that our C file was having upon first look, I separated the Admin Menu Functions and the Guest User Menu Functions from the Login Menu functions into their own respective C files. I then used *#include "GuestMenu.c"* and *#include "AdminMenu.c"* within the LoginMenu.c File to ensure that the individual who logs in is able to use their respective menu and functions.

References

Fischer, A. E., Fischer, M. J., & Eggert, D. W. (2016, August 30). *Applied Introductory C Programming - University of New Haven*. Chapters from 2nd edition. http://eliza.newhaven.edu/apc/.

Sadeghi, R. (2020, December 14). *RezaSadeghiWSU/LMS*. GitHub. https://github.com/RezaSadeghiWSU/LMS.

Sadeghi, R. (2020, December 15). *RezaSadeghiWSU/Calendar-Management-System*. GitHub. https://github.com/RezaSadeghiWSU/Calendar-Management-System.

Sadeghi, R. (2021). passwordSystem.c. West Haven; Reza Sadeghi.