Warehouse Management System

Introduction to Programming
CMPT 120L 114

Team ABC

Marist College
School of Computer Science and Mathematics

Submitted To:
Dr. Reza Sadeghi

Fall 2022

Project Progress Report #3 of Warehouse Management System

Table of Contents

Table of Figures

Project Description: Warehouse Management System

Summary: The warehouse management system (WMS) provides an organized way of storing different products and elements in a warehouse. You can consider a library as a warehouse, which maintains books' details and user libraries. A general WMS stores details of name and identification number of products, their store time, the required storage condition, price, weight, height, etc. following this, this system allows guest users to search for different content and request to borrow/buy them. Your WMS will store the data of different user types in distinct SQL tables. This system should at least support the following items:

1. Admin user is capable of:
    a. Having admin user and password for log in (a string of at least 8 characters)
    b. Changing the admin user and admin password
    c. Adding a guest user to WMS by creating a new username and password. a guest user is not able to define or remove other users.
    d. Removing users from WMS by removing their username, password, and corresponding recorded data.
    e. Adding an item to the warehouse with varied details, such as:
        i. Type: food, books, cars, etc.
        ii. Stored time in the warehouse
        iii. Pick out time from the warehouse
        iv. ID: each item in your library should have a unique identification number with a specific format
        v. Name
        vi. Provider/creator's name
        vii. Quantities: the number of available items. For instance, item x with a quantity of 2 is a sign of 2 available x items in your warehouse.
        viii. Place: where the item is stored
        ix. Price
    f. Deleting an item from the warehouse
    g. Editing an item in the warehouse
    h. Viewing the list of borrowing requests
    i. Accepting or rejecting a borrowing request
2. Each user should be able to:
    a. Search through WMS based on all items' details, such as id, name, and producer.
    b. Save a list of favorite items 7
    c. Request to borrow/buy some items for a specific time. For example, borrowing an item for 3 weeks.

    d. View the history of borrowed items
3. WMS should be a user-friendly software, such that:
    a. It shows a welcome page
    b. It provides a menu of all functions to the user on all pages
    c. It illustrates the reports in a tabular form. For instance, it displays a well-organized list of the requested items.
    d. WMS should provide an exit function and thank the user for using this software.
    e. It shows a warning if:
        i. The admin user tries to add a new item to the library with an existing ID.
        ii. A user search request returns null items.
4. WMS should protect the user information, such that
    a. Optional: WMS passwords and the recorded information should be ciphered. In the simplest case, you can use the caesar cipher methodology. The easiest way to understand the caesar cipher is to think of cycling the position of the letters. In a caesar cipher with a shift of 3, a becomes d, b becomes e, c becomes f, etc. When reaching the end of the alphabet it cycles around, so x becomes a, y becomes b, and z becomes c.

GitHub Repository Address

https://github.com/Shannonmaier/-CMPT-120L-112_Warehouse-Management-System_Team-ABC-

Graphical User Experience Design

The figure below outlines how the warehouse management system will function. The user starts by providing a username and password in order to log in. In the case that their username or password is incorrect, they will have to try logging in again. When the user logs in successfully, they will be directed to either the administrator menu, or regular user menu depending on their status within the program. If the person is an administrator, they will then get to choose from a variety of pages which will allow them to edit user information, products, or borrowing information. Each of these three pages has a subset of actions that they can choose from to perform different tasks. If the person is a normal user, they can search products, add a product to their favorites list, request to borrow a product, or view the borrowing history. Each individual page has the option to return to the main menu of either the user or administrator, through which the individual is able to exit the program if they so choose. Lastly, all of the values stored in the warehouse management system will be stored in a CSV file.

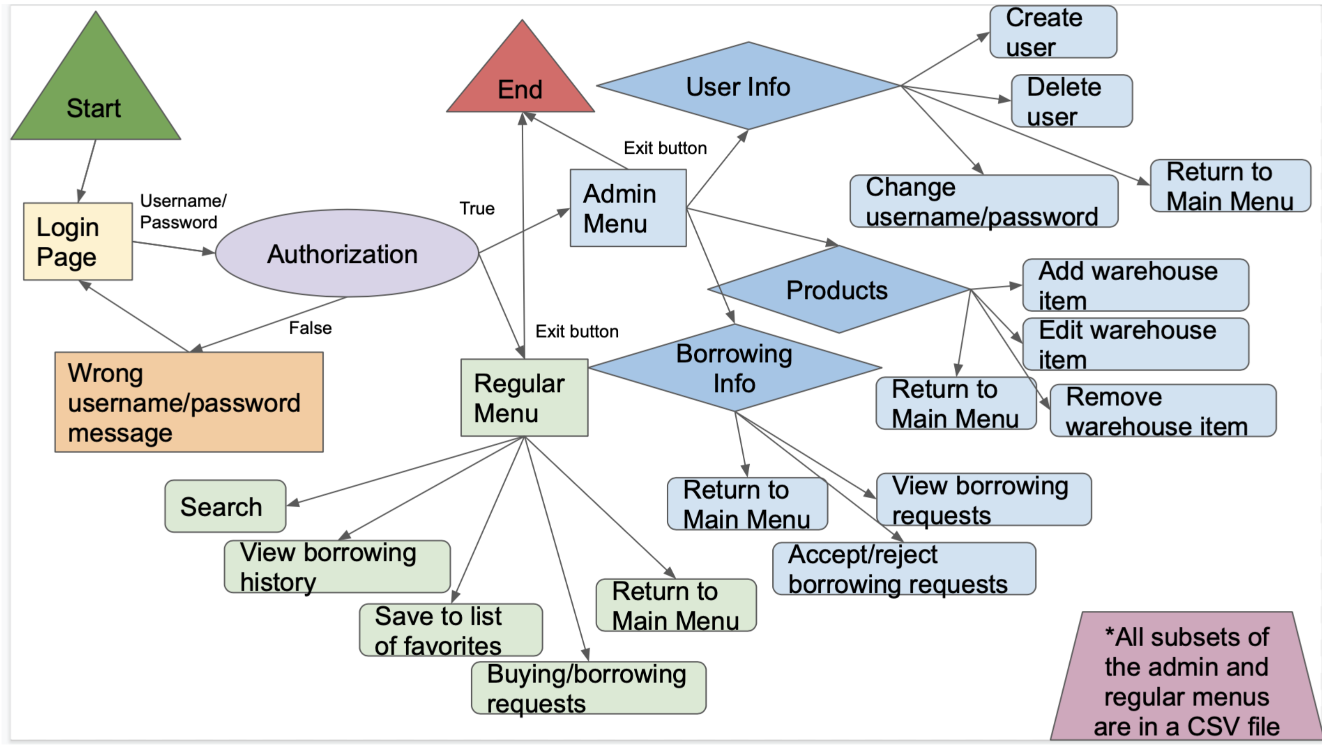


Figure 1 - Warehouse Management Systems Flowchart

Graphical User Interface Design

Figure 2, shown below, displays the login page that initializes the entire program and allows the user to get to either the Admin Main Menu or the User Main menu by putting in specific usernames and passwords and clicking the login button.

The author of this section of the code was Jaden Reasor. The code for the Login page first initializes the Admin Login and users CSV files with the passwords in them so that the files can be used in iteration. This only happens if the file does not already exist. Then the code defines the Login Page. The first definition within the Login Page is the deleteLogin, which will be used to destroy the page. Next, the code defines Login() in order for the login button to actually work. This function opens the AdminLogin.csv and checks if the username and password match up. If they do, then it navigates to the Admin Main Menu. Next, with the else statement, it goes through the users.csv and checks if the username and password match up there. If they do, it will open the User Main Menu. If they do not, it will display a message stating that the username and password are incorrect.

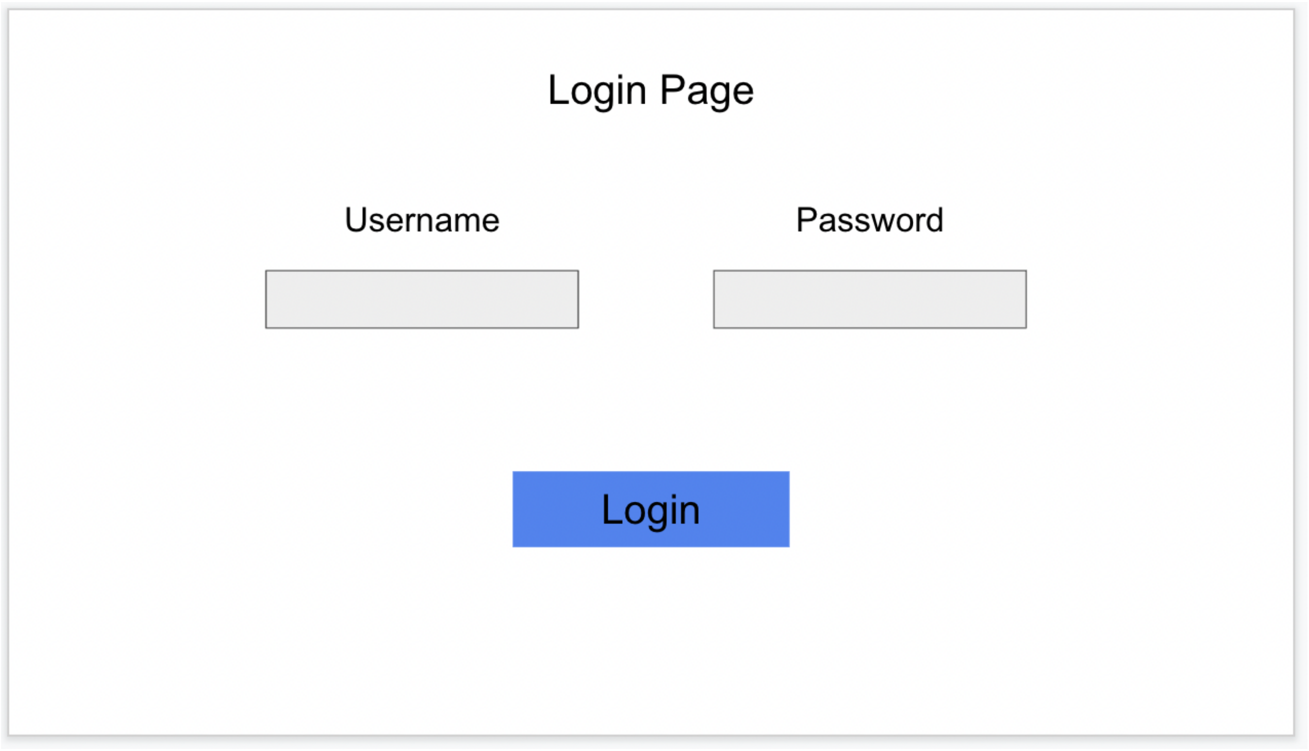


Figure 2 -Login Page

Figure 3, the Admin Main Menu, exhibits five buttons; “View Products and product actions”, “View Users and User Actions”, “View Borrow Requests”, “Change Admin User and Password”, and “Logout”. The first three buttons connect to completely separate pages. The “Change Admin User and Password” button connects to a popup window that is displayed at the same time as this page. The “Logout” button directs the user back to the Login Page.

The author of this section of the code was Cayleigh Goberman. The code for the Admin Main Menu page begins with the AdminMenu() function. This function first includes the function changePage() which creates the graphics for the page. This function also includes the changeLogin() function for the change button within that page. This function utilizes a list with the new username and password entered by the user, and then writes it in the AdminLogin.csv. The deleteMenu() function is also included in the Admin Main menu code; this allows the menu to be destroyed when new pages are opened. The “Logout” button uses the previously defined Login Page function to open the Login Page again.

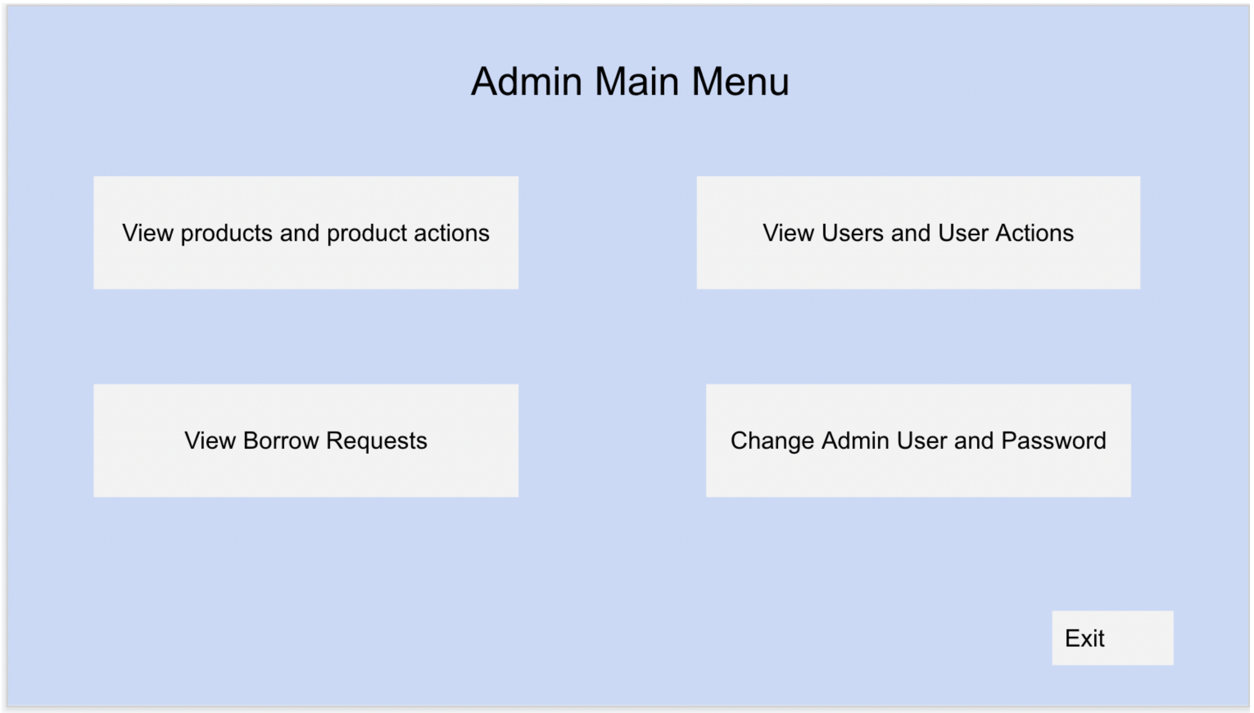

Admin Main Menu
View products and product actions
View Users and User Actions
View Borrow Requests
Change Admin User and Password
Exit

Figure 3 - Admin Main Menu

In Figure 4, the Admin Products and Actions Pages is shown. This page exhibits a text box on the left, where all of the warehouse items will be shown, and six buttons on the right, which contain the actions that the Admin can perform in regards to the products. The right side of the page also contains the "Back to Menu" button.

The author of this page of the code was Shannon Maier. The code for this page starts with the overall function for the page, AdminProducts(), which creates the graphics for all of the buttons. These buttons will utilize the functions that will be subsequently defined. In the Print() function, which is employed for the "Print" button, the code checks to see if there is a path to warehouse.csv. If this is true, the code opens the warehouse.csv file and deletes everything previously printed in the text box. It then inserts all the information from that file into the text box by utilizing a "for" loop. The addPage() function sets up a popup window to add products. It then defines the add() function for the "Add Product" button on that page. The add function uses the get() method in order to retrieve all of the inputs and add them to both the warehouse.csv file and the text box. In the editPage() function, the graphics for the popup window are created, and the edit() function is defined for the "Edit Product" button. The function uses the get() method and opens an empty list called "lines". It subsequently goes through the warehouse.csv and checks if the info does not equal the inputted information. If all of the input is not equal, then the line is added to the list and all the get() info is also attached to the list. Next, the entire warehouse.csv is removed, then reopened and written with all of the rows in "lines". After that, the code deletes the info in the text box and rewrites it with the updated product. The function removePage() creates the graphics for the page that allows the user to input the info that will be removed. The delete() function is used as the command for the button on the page. This function retrieves the name and ID number the user wants to delete, then opens the list called "lines". If all of the info in warehouse.csv does not match up with the ID and name, then that row gets added to the list. "lines" is written in warehouse.csv after it has been deleted, and the info is rewritten in the text box. The searchPage() function employs the get name and ID function, then searches the warehouse for either a name or an ID that matches. If the product matches, it then displays them in the text box.

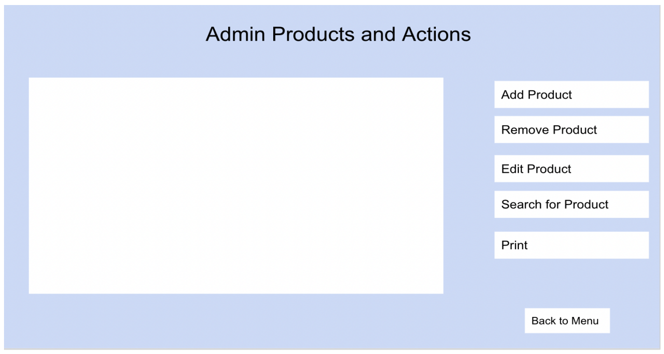


Figure 4 - Admin Products and Actions

In the Admin-User Window, a text box is displayed on the left with the list of added users. On the right, there is an "Add User" button, a "Remove User" button, and a "Back to Menu" button, which will take you back to the Admin Main Menu.

The author of this page of the code was David Penafiel. The code for this page initializes an overall function called AdminUserPage() that contains both the graphics and the remaining required functions. The AddPage() function creates a popup window for the admin to add the new user's info. This Add() function is utilized for the "Add User" button in this window. It retrieves the inputted information (name, password, username) and adds it to a new csv file called users.csv, which is then written in the textbox. Whenever the page is opened, all of the users that are already in the csv file will be automatically exhibited. The RemoveUserPage() creates the graphics for the remove page and contains the RemoveUser() function, which is used for the "Remove User" button in that page. The RemoveUser() function opens an empty list called "lines" and uses the get() method for the password and username. If the information in "lines" does not equal the inputted information, it then opens the users.csv and appends the row to "lines". Next, it completely removes users.csv and reopens it, writing only the information in "lines" into the new file. It subsequently takes this info from the file and types it into the textbox after deleting everything that had previously been there. The "Back to Menu" Button uses the destroy method to close this page, then reopens the menu using the AdminMenu() function.

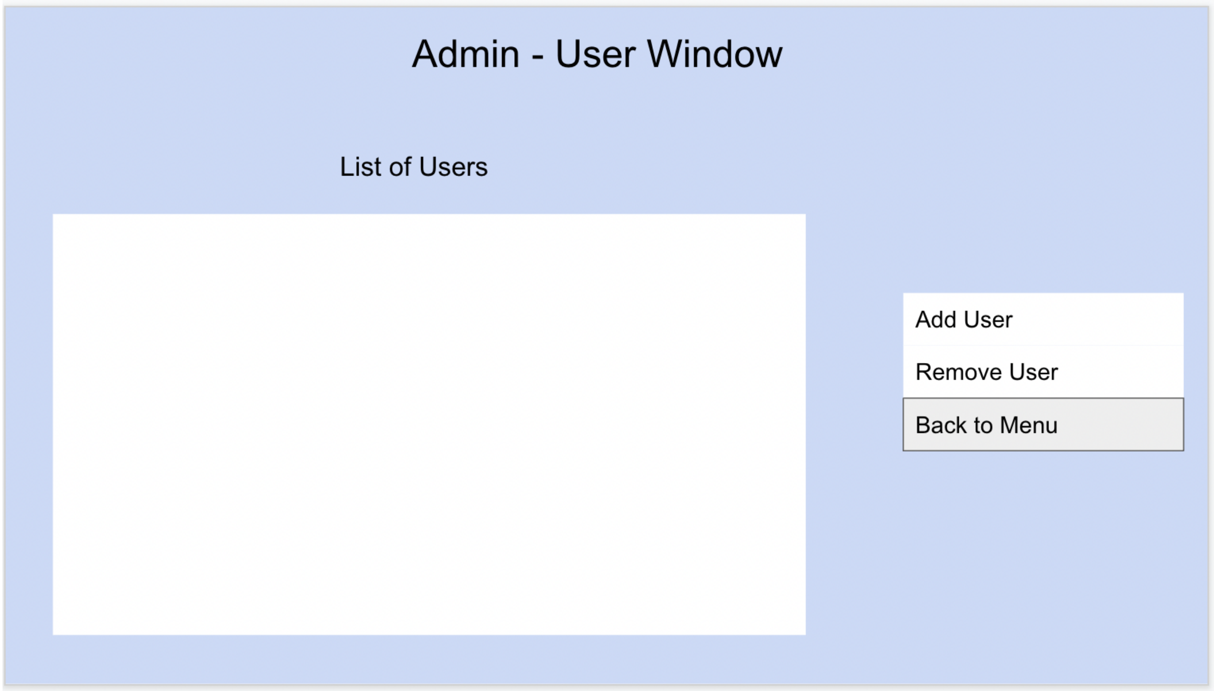


Figure 5 - Admin User Editing Window

In Figure 6, the Admin - View Borrow Requests Window, there is a text box displayed on the left that contains all of the requests. There are also three buttons on the right: “Accept”, “Decline", and “Back to Menu”. The “Accept” and “Decline” buttons show popup windows to accept or decline the requests, and the “Back to Menu” button brings you back to the Admin Main Menu.

The code for this page is under the overarching function of BorrowReqPage(), which creates the graphics. In this function, the AcceptPage() function is created for the “Accept” button. This AcceptPage() function creates the popup window and uses the acceptReq() function to change the information. The acceptReq() utilizes the inputted length, name, and ID, and opens a list called “lines". It opens the accepts.csv and writes the info into the CSV file. Next, it opens the borrowrequest.csv and checks if the name, ID, and length match up in the rows. If they do not, then the row is added to the lines list. The original borrowrequest.csv is then deleted and reopened. The “lines” list is written in this CSV file, then the file is written into the text box. In the declinePage(), the declineReq() function is defined, which is similar to the acceptReq() function, but the info is not added to a new CSV file; it is only deleted from the borrowrequest.csv. When the admin opens this page, all of the requests will automatically be shown, if any have been made.

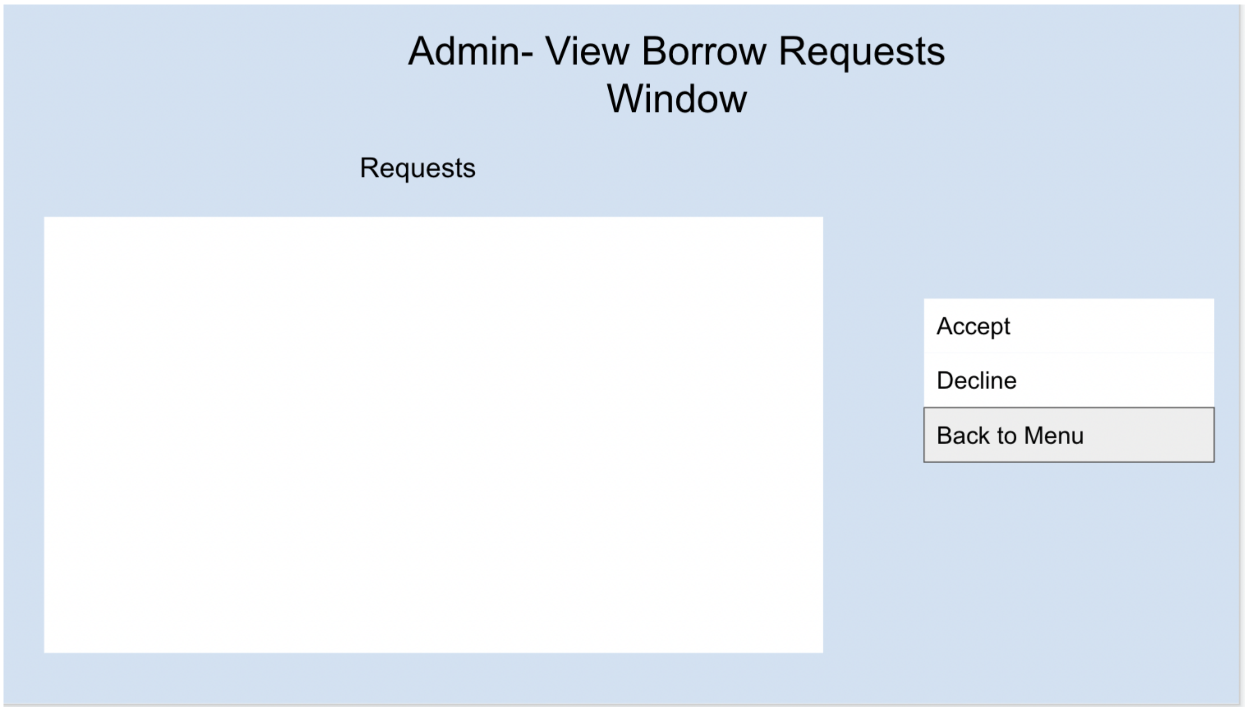


Figure 6 - Admin - View Borrow Requests

In Figure 7, the User Main Menu, there are four buttons. These are the “View products and user product actions” button, the “View Favorites List” button, the “View Borrowed Items History” button, and the “Exit” button. The first two buttons bring the user to new pages, while the “Borrowed History” button creates a popup window.

The author of this page of the code was Dan Aulbach. The code for this page begins with the overall function of userMenu(), which creates the graphics for the page and inputs all the functions for the Tkinter buttons. The function BorrowHisotryPage() is then made, which creates the popup window that displays the borrowed items history. It does this by going through the accept.csv and printing the file in the text box in the popup window. The Favorites() function creates a new window that displays a text box, which will be filled with all of the products the user adds to their favorites list by reading through the favorites.csv and printing it in the textbox. The products and product actions are explained in the next figure.

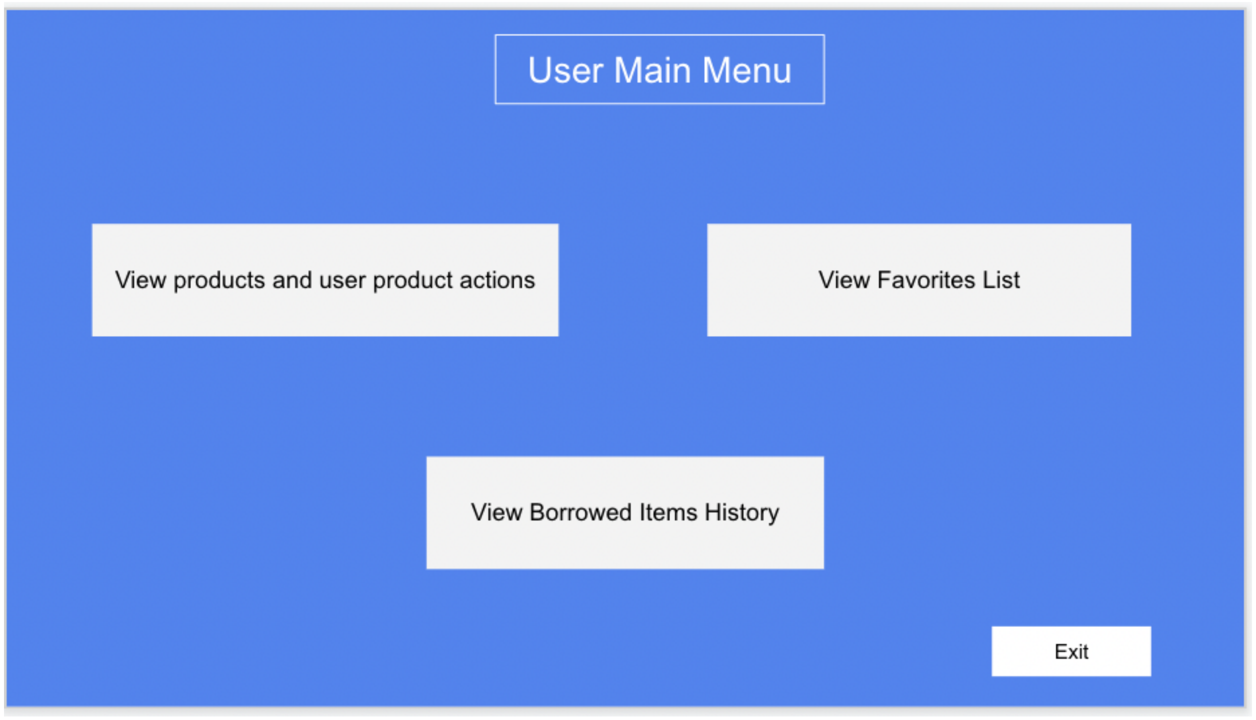

User Main Menu
View products and user product actions
View Favorites List
View Borrowed Items History
Exit

Figure 7 - User Main Menu

Figure 8, the User Products Window, outlines what the user will see after choosing to view products and user product actions in the User Main Menu. From this screen, the user will be able to add an item to their favorites list, request to borrow an item, search their product library, or return to the main menu. The user will go into a new page if they wish to add an item to their favorites list, request to borrow an item, or search for a particular item throughout the warehouse inventory.

The author of this page of the code was Lauren Lietzke. The code for this page is initialized with the UserProductPage() function that has all of the graphics coded in it. The next function defined within this function is the BorrowPage() function, which creates the popup window for the user to request to borrow a product. The function BorRequest() is used in the button on that window, and utilizes the get() method in order to go through the warehouse.csv and check if the inputs match any of the rows. If a row does match the input, a new file is opened called borrowrequest.csv and the row is added there. The function for SearchProd() creates the graphics and initializes the function Search(). The Search() function employs the user inputs and all of the available info and goes through the warehouse.csv and checks if any of it matches. If it does, that row will be displayed in the text box. The next function is defined as FavePage(), which sets up the popup page to add something to the favorites list and creates the function to actually do that. The addFav() function takes the name and ID that the user inputs and goes through the warehouse.csv to check if it matches with any of the rows. If it does, then that row is added to a new file called favorites.csv. When the user first opens this window, the products are automatically shown in the text box. The back-to-menu button uses the destroy method to close this window and open the User Main Menu.

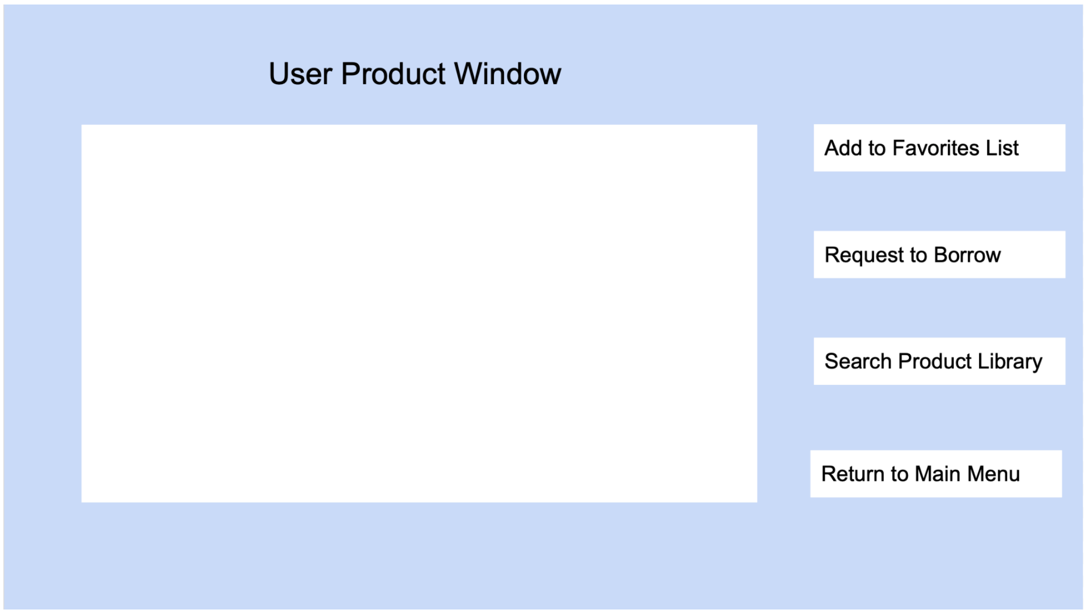
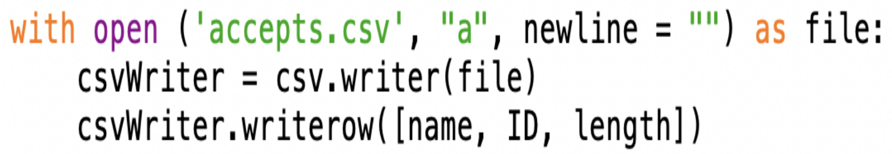


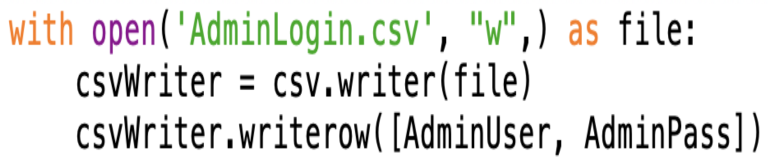
Figure 8 - User Products and Actions Page

Data Storage

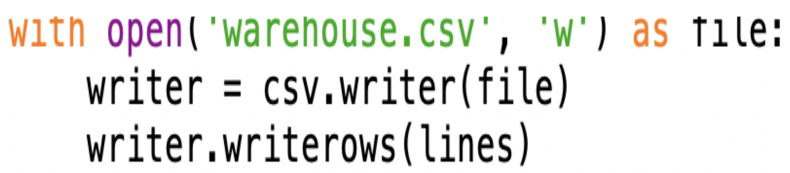
All of the data in this program is stored within six separate CSV files. These files include users.csv, Adminlogin.csv, warehouse.csv, borrowrequests.csv, accepts.csv, and favorites.csv. All products that are added by the Admin are put in the Adminlogin.csv and this is updated depending on any changes made. The Admin login information is placed in Adminlogin.csv, and the user login information is placed in users.csv. All of the borrow requests made by the user are placed in the borrowrequests.csv so that the Admin can accept or decline them. The accepts.csv holds all of the accepted borrow requests. Lastly, favorites.csv hold all of the favorites that a user adds to their list.



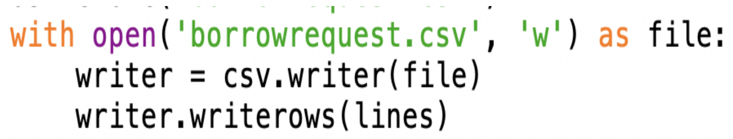
```
with open ('accepts.csv', "a", newline = "") as file:
    csvWriter = csv.writer(file)
    csvWriter.writerow([name, ID, length])
```



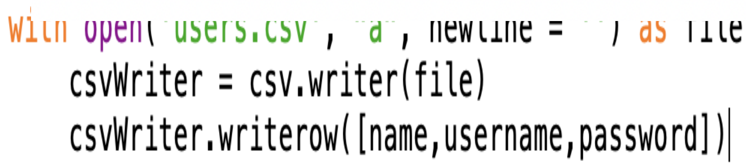
```
with open('AdminLogin.csv', "w",) as file:
    csvWriter = csv.writer(file)
    csvWriter.writerow([AdminUser, AdminPass])
```



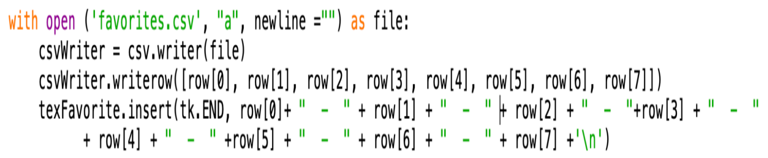
```
with open('warehouse.csv', 'w') as file:
    writer = csv.writer(file)
    writer.writerows(lines)
```



```
with open('borrowrequest.csv', 'w') as file:
    writer = csv.writer(file)
    writer.writerows(lines)
```



```
with open('users.csv', "a", newline = "") as file:
    csvWriter = csv.writer(file)
    csvWriter.writerow([name,username,password])
```



```
with open ('favorites.csv', "a", newline ="") as file:
    csvWriter = csv.writer(file)
    csvWriter.writerow([row[0], row[1], row[2], row[3], row[4], row[5], row[6], row[7]])
    texFavorite.insert(tk.END, row[0]+ "  -  " + row[1] + "  -  " + row[2] + "  -  "+row[3] + "  -  "
        + row[4] + "  -  " +row[5] + "  -  " + row[6] + "  -  " + row[7] +'\n')
```