Project Objective

A computer shop management system sells various types of hardware and software accessories such as Laptop, Antivirus and Keyboard etc. It's very hard for the shopkeeper to remember their cost price and in what price these items to be sold. As the market goes up and down for various products daily, so there should be some proper medium through which prices of these items can be changed frequently without any problem. There are many customers who will make their payment online, so there should be a medium to take their credit card as input and reduce exact amount from their account. This computer shop management system will enable its user's keep their customers report secure, so that next time these customers when arrive to their shop, do not have to enter their details again. Buyers will also able to select products options while purchasing through computer screen and can make their payment from that particular screen and pick up their materials which they have selected at the pickup center within the shop.

Class Descriptions

-Computer:

- **add_product :**this function is to add a new product to the inventory of the computer store .The function first prompts the user to enter the type of product, either "Software" or "Hardware", and validates the input. If the input is valid, the function pushes the product type to a list named type and calls AddSoftwareCategory() or AddHardwareCategory() functions to add details about the category of product being added. If the input is invalid, the function prompts the user to enter a valid input. Next, the function prompts the user to enter the name, price, quantity, and details of the product. It then stores this information in four separate vectors named NameProduct, PriceProduct, QuantityProduct, and DetailsProduct. Finally, the function generates a unique identifier for the product by using a random number generator and pushes it to a list named id .Overall , the add_product() function allows the user to add a new product to the computer store's inventory by prompting for and storing relevant product information in various lists.

- **DispalyAllProduct :**this function is to display all the product information stored in the class, including the product name, type, ID, price, quantity, and details .The function starts by initializing iterators for each of the lists that store the product information: NameProduct, type, id, PriceProduct, QuantityProduct, and DetailsProduct .The function then enters a while loop that continues until the NameProduct list is empty .During each iteration of the loop, the function prints out the product information using the values of the iterators. It also includes a separator line made of asterisks for clarity .At the end of each iteration, the function increments each iterator to the next value in its respective list .If the

function reaches the end of the NameProduct list, the loop breaks, and the function terminates.

- **GetSoftProduct :**this function is to search for and display all the products that are of type "Software" .The function first initializes a variable named "check" to 0.It then enters a for loop that iterates through the "type" list, looking for the string "Software" .If the loop finds a product with the type "Software", it sets the "check" variable to 1 and breaks out of the loop .If the "check" variable is 1, indicating that at least one "Software" product has been found, the function initializes a vector called "indices" .The function then enters another for loop that iterates through the "type" list again, this time storing the indices of all the products with the type "Software" in the "indices" vector .After the "indices" vector has been populated, the function enters a final for loop that iterates through the "indices" vector and displays the product information for each "Software" product .During each iteration of the loop, the function initializes iterators for each of the lists that store the product information, and then uses the "advance" function to move each iterator to the correct index in its respective list .The function then prints out the product information using the values of the iterators. It also includes a separator line made of asterisks for clarity .If no "Software" products are found, the function simply prints out a message indicating that there are no "Software" products in the inventory.

- **GetHardProduct :**It retrieves all the products of type "Hardware" from the NameProduct, id, PriceProduct, QuantityProduct, and DetailsProduct lists and prints them to the console .The function begins by initializing a variable check to 0. It then iterates through the type list using a range-based for loop and checks if the current element is equal to the string "Hardware". If it is, the check variable is incremented by 1, and the loop is terminated using a break statement .If check is not equal to 0, it means that there is at least one product of type "Hardware" in the system. In this case, the function creates a vector indices to store the indices of all the products of type "Hardware" in the type vector. It then iterates through the indices vector and retrieves the corresponding product information from the other vectors using advance and begin functions .Finally, the function prints the name, id, price, quantity, and details of each product to the console using cout. It also prints a line of asterisks to separate each product. If there are no products of type "Hardware" in the system, the function prints a message indicating this.

- **ChangeQuantity :**allows the admin to change the quantity of a product in the inventory. The function first displays a menu with four options: show all products, show software products, show hardware products, or exit .If the admin chooses one of the first three options, the function will display a list of products, depending on the user's choice, using the corresponding function DispalyAllProduct(), GetSoftProduct(), or GetHardProduct(). The admin can

then enter the name of the product they want to change the quantity of. The function searches for the product by name in the NameProduct list, and if it is found, the admin is prompted to enter the new quantity for the product. The function validates the input to ensure that the new quantity is not negative. If the input is valid, the function updates the corresponding quantity in the QuantityProduct list .If the admin chooses the fourth option (exit), the function exits the while loop and returns to the calling function .Overall, this function allows the admin to interactively update the quantity of a product in the inventory.

- **ChangePrice :**this function is to allow the admin to change the price of a product in the computer's inventory .The function starts by displaying a menu that allows the admin to choose whether they want to see all products, software products, or hardware products. The admin is prompted to enter their choice, and if they enter an invalid choice, they are asked to enter a valid choice .If the admin chooses to see all products, the function calls the DispalyAllProduct() function to display all the products in the inventory. The admin is then prompted to enter the name of the product they want to change the price for. The function then searches for the product in the NameProduct list and sets the index variable to the index of the product. If the product is found, the admin is prompted to enter the new price for the product. If the new price is valid (i.e., greater than 0), the function updates the price of the product in the PriceProduct list .If the admin chooses to see only software products or only hardware products, the function calls the GetSoftProduct() or GetHardProduct() function, respectively, to display only the software or hardware products. The rest of the process for changing the price of a product is the same as when all products are displayed.

-Sign up:

- **GetName :**The function first declares a string variable called Name to store the user's input for their name .Then, it creates a regular expression object called regx using the regex class from the standard library. The regular expression matches any of the special characters: "@_!#$%^&*()<>?/|}{~:."The function then prompts the user to enter their name and uses getline() to read the entire line of input from cin, including any leading whitespace. It uses ws manipulator to skip any leading whitespace before reading the input.A while loop is used to keep prompting the user for their name until it meets the desired format. It first checks if the name contains any digits or special characters by using the any_of() function from the standard library and isdigit() function from the cctype library. If the name contains any digits or special characters, the user is prompted again to enter a valid name.If the inputted name is valid, the function converts all characters to lowercase using the for_each() algorithm from the standard library and then appends the name to a vector called name. In summary, this function

reads user input for their name and ensures it does not contain any digits or special characters, converts the name to lowercase, and appends it to a list..

- **GetPass :**It prompts the user to enter a password and stores it in a list called password that is a member variable of the SignUp class.The function uses a while loop to repeatedly prompt the user to enter a password until a password with at least 8 characters is entered. It first initializes a string variable pass to an empty string. It then displays a prompt message using cout, asking the user to enter a password .The function reads the user's input using getline function and the ws manipulator to ignore any leading white spaces. The input is then stored in the pass variable. The function then checks if the length of the entered password is less than 8 characters. If it is, the function displays an error message using cout and continue keyword is used to go back to the beginning of the loop to prompt the user again for a valid password .If the length of the password is at least 8 characters, the loop is exited using break and the password is stored in the password list using the push_back function .Overall, this function ensures that the user enters a password with at least 8 characters before proceeding with the sign-up process.

- **GetPhone :**It prompts the user to enter their phone number and then uses a while loop to keep asking the user for input until a valid phone number is provided .The function checks that the phone number meets certain criteria:
    - It should not be empty or contain whitespace characters only.
    - The first three digits must be "010", "011", "012", or "015".
    - The length of the phone number should be exactly 11 digits.
    - It should not contain any alphabetic characters.
    - It should not contain any special characters, except for hyphens.

  If the entered phone number is invalid, the function prints a message asking the user to enter a valid phone number.
  If the entered phone number is valid, the function stores it in a string vector named phone.Note that the function uses a regular expression to check if the phone number contains any special characters. The regular expression pattern [@_!#$%^&*()<>?/|}{~:.] matches any of the special characters within the square brackets. The regex_search function is used to check if the phone number contains any matches to the pattern.

- **GetMail :**Its purpose is to prompt the user to enter their email address, validate the email address and store it in a list called mail .The function starts by declaring a string variable called Mail. Then it creates a regular expression object called regx that matches certain special characters that should not be included in an email address .Next, the function enters a loop that will keep running until the user enters a valid email address. Within the loop, the function first prompts the user to enter their email address by printing "Enter your e_mail:". The getline(cin

>> ws, Mail) statement is used to read the user's input and store it in the Mail variable .The function then checks if the Mail string contains both an "@" and "." symbol, which are necessary components of a valid email address. If the Mail string does not contain these symbols, the function prints an error message asking the user to enter a valid email address .Additionally, the function checks if the Mail string contains any of the special characters defined in the regx regular expression, and if it contains any whitespace characters. If it does, the function prints another error message asking the user to enter a valid email address.If the Mail string passes all these validation checks, the function breaks out of the loop and adds the Mail string to the mail list. The function then ends, having successfully obtained and validated the user's email address.

- **GetMoney :**this function is to prompt the user to input the amount of money they have in the bank to buy something, and then store that value in a list named money which is a member of the class Sign Up .The function uses a while loop to keep prompting the user to enter a value until a valid input is provided. Within the while loop, the function displays a message asking the user to input how much money they have in the bank, and then reads in the user's input using the cin command, and stores it in a variable named Money .The function then checks if the value of Money is less than 0. If it is, the function prints an error message asking the user to enter a valid value. If the value of Money is greater than or equal to 0, the function exits the while loop and appends the value of Money to the money list using the push_back command .Overall, this function ensures that the user inputs a valid value for the amount of money they have in the bank to buy something, and stores that value in the money list.

- **Sign :**The purpose of this function is to prompt the user to sign up as either a regular user or an admin, and then store their information (name, email, password, phone number, and bank balance) in the appropriate data structures .The function uses a while loop to keep prompting the user to select a sign-up option until a valid input is provided. Within the while loop, the function displays three options: sign up as a regular user, sign up as an admin, or exit. The user selects an option by entering the corresponding number, and the function reads in their input using the cin command, and stores it in a variable named choice .If the user selects option 1, the function adds the string "user" to a list named usertype , and then calls five other functions named GetName(), GetMail(), GetPass(), GetPhone(), and GetMoney() to prompt the user to input their name, email, password, phone number, and bank balance, respectively. After successfully getting all the required information, the function displays a success message and breaks out of the while loop .If the user selects option 2, the function enters another while loop that prompts the user to enter the name and password of an administrator. If the entered name and password match the predefined values for the admin, the function adds the string "admin" to the

usertype list, and then calls the same five functions as above to prompt the user to input their name, email, password, and phone number. After successfully getting all the required information, the function displays a success message and breaks out of the while loop .If the entered name and password do not match the predefined values, the function displays an error message and asks the user to enter valid data. The function will continue to prompt the user for admin credentials until valid credentials are entered .If the user selects option 3, the function simply breaks out of the while loop and terminates the sign-up process .Overall, this function enables users to sign up as either regular users or admins, depending on their choice, and then stores their information in the appropriate data structures for further processing.

- **LogInUser :**The function returns a tuple containing the user's name and phone number if the login is successful .The function first checks if there are any users registered in the database by iterating through the usertype list. If the list contains the string "user," it means there is at least one user in the database, and the login process can proceed. If there are no users in the database, the function calls the sign() function to prompt the user to sign up. Next, the function prompts the user to enter their email and password using getline(). It then iterates through the mail list to find the index of the user's email address. The password for that email address is then retrieved from the password list using the same index . the password entered by the user matches the password in the database, the function returns a tuple containing the user's name and phone number. If the login is unsuccessful, the user is prompted to enter their email and password again until they provide valid credentials .Overall, the function provides a basic login system for a user database, allowing users to access their account details if they have already signed up.

- `LogInAdmin :`It allows the user to log in as an administrator if at least one administrator account has been created. If no admin accounts exist, the user is given the option to create one or to log in as the default administrator.The function starts by checking if there are any existing admin accounts by iterating through the usertype list. If an admin account is found, the function prompts the user to enter their email and password. It then compares the inputted values with the ones stored in the mail and password lists respectively, at the same index. If the email and password match an admin account, the user is logged in as an administrator .If no admin accounts exist, the user is prompted to either create one or log in as the default administrator. If the user chooses to create an admin account, the sign() method is called. If the user chooses to log in as the default administrator, they are prompted to enter the default administrator's name and password. If the inputted values match the default administrator's name and password, the user is logged in as an administrator .Overall, the LogInAdmin()

function allows users to log in as an administrator and perform administrative tasks if they have the proper credentials.

- **DisplayAllUsers :**The function starts by creating iterators for the name, mail, and phone lists using the begin() method. It then enters a while loop that continues to iterate through the vectors until there are no more elements in the name list. During each iteration, the function prints the corresponding name, email, and phone number of the user by dereferencing the iterators. It also prints a line of asterisks to visually separate the different users .The iterators are then incremented using the ++ operator. The loop continues until the N iterator reaches the end of the name vector, at which point the loop is exited .Overall, the DisplayAllUsers() function is used to display all of the user information that has been stored in the name, mail, and phone vectors. It is useful for displaying all user information in a formatted way.

- **AddMoney:** This function appears to be a method of a class called SignUp and its purpose is to add money to the account of a user whose name matches the input argument "auth" .The method first searches for the user's name in a list called "name" using a for loop and a linear search. Once it finds the matching name, it saves its index to a variable called "index" .It then uses the index to access the corresponding user's current account balance in a list called "money" using the "advance" function to move the iterator "add" to the correct position in the list .The method then enters a while loop that repeatedly prompts the user to enter a positive value to add to their account balance until a valid input is received .Once a valid input is received, the method adds the entered amount to the user's account balance and outputs a message confirming the transaction .Note that there is no error handling for cases where the input name is not found in the "name" list or if the "money" list is not the same length as the "name" list. Additionally, there are no checks to prevent adding more money than what is allowed or verifying the validity of the user's input.

-Hardware:

- **AddHardwareCategory:** The purpose of this function is to add a hardware category to the list of categories maintained by the Hardware class .The function first displays a menu of hardware categories to the user, numbered 1 to 17. It then prompts the user to enter a number corresponding to their choice of category. The function uses a while loop to ensure that the user enters a valid input - that is, a number - by catching any exceptions that might be thrown if the user inputs a string instead .Once a valid choice has been made, the function uses a switch statement to add the corresponding category to the category list maintained by the Hardware class. If an invalid choice is made, the function displays an error message and returns to the beginning of the loop to prompt the user again

.Overall, this function allows the user to add a new hardware category to the Hardware class, ensuring that only valid inputs are accepted.

- **GetHardCategory**: The purpose of this function is to display the list of hardware categories that have been added to the category list maintained by the Hardware class.

-Software:

- **AddSoftwareCategory:** The function first prints a menu of software categories to the console using cout. It then enters a while loop that repeatedly prompts the user to enter their choice of software category. The user input is stored in the choice string variable using getline(cin >> ws, choice), which reads a line of input from the console and removes any leading whitespace.The if-else statements inside the loop check the value of choice against each of the valid software categories. If the user enters a valid category, the corresponding string is added to the category list using push_back(), and the loop is exited using break. If the user enters an invalid category, an error message is printed to the console, and the loop continues to prompt the user for input.The function does not return anything, and it assumes that the category list is a member variable of the Software class.

- **GetSoftCategory**: The function uses a for loop that iterates over each element of the category list using a range-based for loop syntax. Inside the loop, the current element is assigned to the variable i, and then it is printed to the console using cout. The string "Category:" is printed to the console, followed by the value of i, and then a new line is added using endl.The function does not take any input parameters, and it assumes that the category list is a member variable of the Software class.

-Buying:

- **BuyingSteps:** the function that handles the process of buying a product, which takes two strings as inputs representing the name and password of the user. The function uses a loop to get the name of the product from the user and searches for it in a list of products. If the product is found, the function displays the details of the product and asks the user to choose one of three options: buy the product and have it delivered, reserve the product, or exit .If the user chooses to buy the product and have it delivered, the function first searches for the user in a list of users. If the user is found, the function asks the user to enter the quantity of the product they want to buy. If the quantity is valid, the function calculates the cost of the purchase and checks if the user has enough money to cover it. If the user has enough money, the function deducts the cost from the user's account,

updates the quantity of the product, and asks the user to enter the location where they want to receive the product. Finally, the function asks the user if they want to make a report about the product or service .If the user chooses to reserve the product, the function asks the user to enter the quantity of the product they want to reserve. If the quantity is valid, the function updates the quantity of the product and displays a message that the reservation was successful .Overall, the code seems to be functioning correctly, but there are some minor issues that could be improved, such as handling invalid inputs and improving the user interface.

- **DisplayReservation:** The function uses a range-based for loop to iterate over the elements of the reservation list, and for each element, it prints the value to the console using the cout object. The endl manipulator is used to insert a newline character after each value is printed, which causes each value to be printed on a separate line .In summary, the purpose of the DisplayReservation() function is to print the contents of the reservation vector to the console.

- **DisplayLocationOrders:** The function loops through the elements of the LocationOrder container using a range-based for loop and prints each element to the standard output followed by a newline character using the cout object and the endl manipulator.Assuming that LocationOrder is a container of strings representing orders for a particular location, this function displays all the orders for that location.

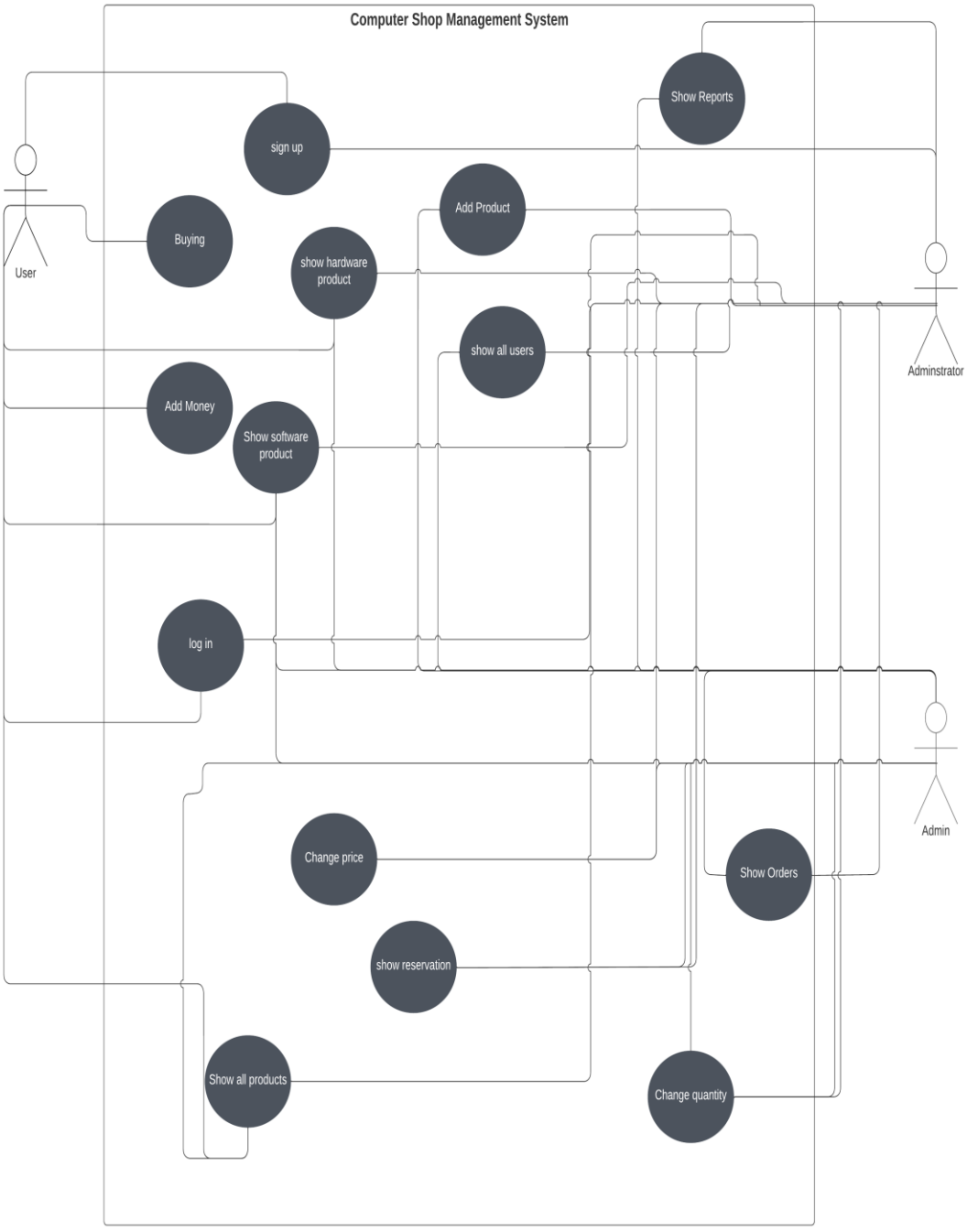
- **DisplayReports:** The function loops through the elements of the report container using a range-based for loop and prints each element to the standard output followed by a newline character using the cout object and the endl manipulator .Assuming that report is a container of strings representing reports related to buying activities, this function displays all the reports currently stored in the container.

- **BuyingProcess:** buying process that takes two parameters as input; n and p which represent the name and password of the user respectively. The function starts with a while loop that continues until the user chooses to exit. Inside the loop, the function presents the user with a menu that includes several options:
    - Show all products
    - Show software products
    - Show hardware products
    - Searching
    - Exit

  Based on the user's input, the function takes different actions. If the user chooses to display all products (option 1), the function calls the DispalyAllProduct() function, which displays all the products stored in the database, then calls the

BuyingSteps(n, p) function to proceed with the buying process .If the user chooses to display software products (option 2), thefunction checks if there are any software products stored in the database. If there are, it calls the GetSoftProduct() function, which displays all the software products, then calls the BuyingSteps(n, p) function to proceed with the buying process. If there are no software products, the function displays a message indicating that there are no software products .If the user chooses to display hardware products (option 3), the function checks if there are any hardware products stored in the database. If there are, it calls the GetHardProduct() function, which displays all the hardware products, then calls the BuyingSteps(n, p) function to proceed with the buying process. If there are no hardware products, the function displays a message indicating that there are no hardware products .If the user chooses searching (option 4), the function checks if there are any products stored in the database. If there are, it calls the BuyingSteps(n, p) function to proceed with the searching process. If there are no products, the function displays a message indicating that the database is empty.If the user chooses to exit (option 5), the loop breaks, and the function terminates.

Use case


Computer Shop Management System
Show Reports
sign up
Add Product
Buying
show hardware product
User
Adminstrator
show all users
Add Money
Show software product
log in
Admin
Change price
Show Orders
show reservation
Show all products
Change quantity

Class diagram