WinPC

Computer Store System

Design Report

For Web Application

Version <1.1>

Revision History

Date	Version	Description	Author
March 28th, 2023	1.0	Create software requirement specification	Barry, Oumar Frost, Ian Niles Gao, Zhi Sehaumpai, Max Yeung, Richard
April 25th, 2023	1.1	Create design report	Barry, Oumar Frost, Ian Niles Gao, Zhi Sehaumpai, Max Yeung, Richard

Table of Contents

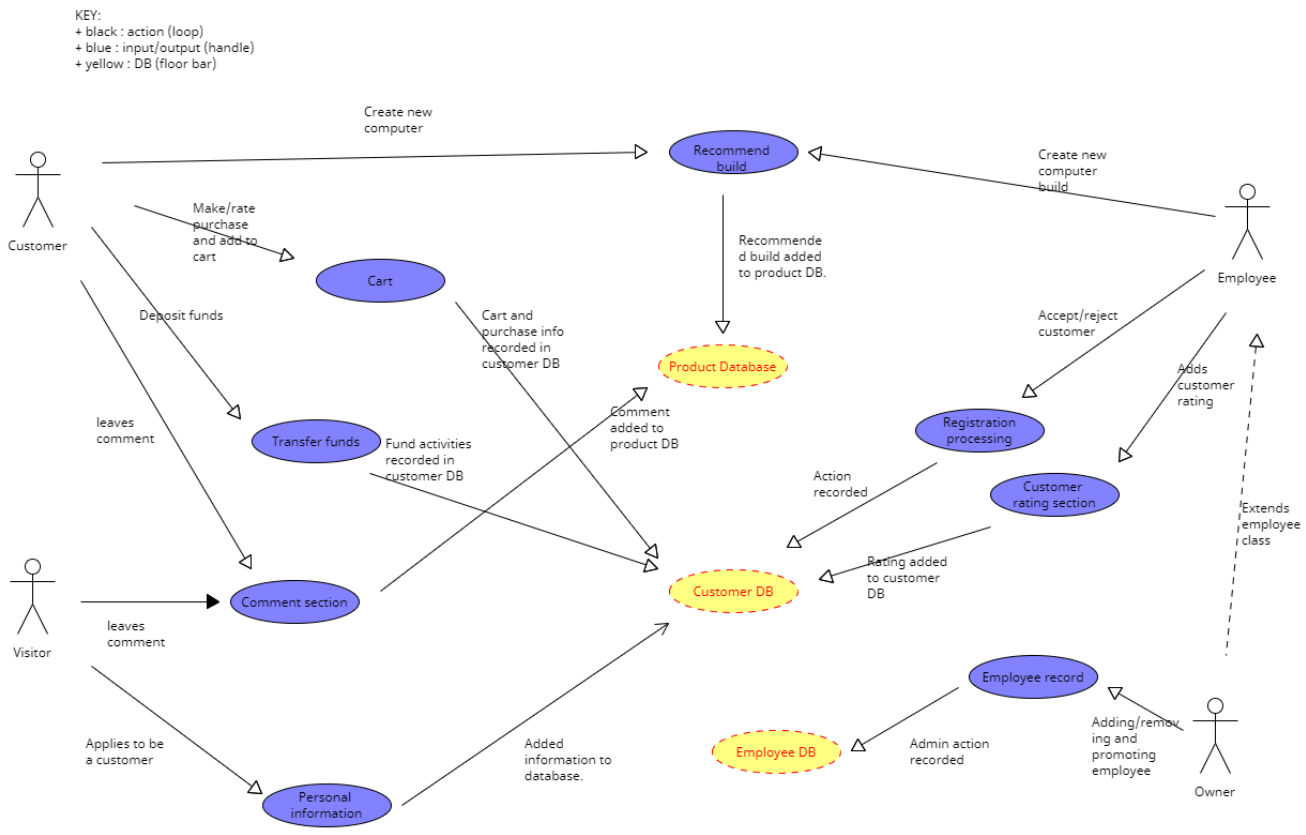
Design Report

1. Introduction

This design report will give an overview of the design and functionality of the entire Computer Store System by our company WinPC.

1.1 Collaboration Class Diagram

Collaboration diagrams are used to show how objects interact to perform the behavior of a particular use case, or a part of a use case. The collaboration class diagram below provides an overview of our Computer Store System.



2. Use Case Analysis

In this section, we provide a detailed overview of the use cases mentioned in the specification report. For each use case, we will provide a collaboration class diagram or petri-net diagram as a way to explain more specifically how the system works.

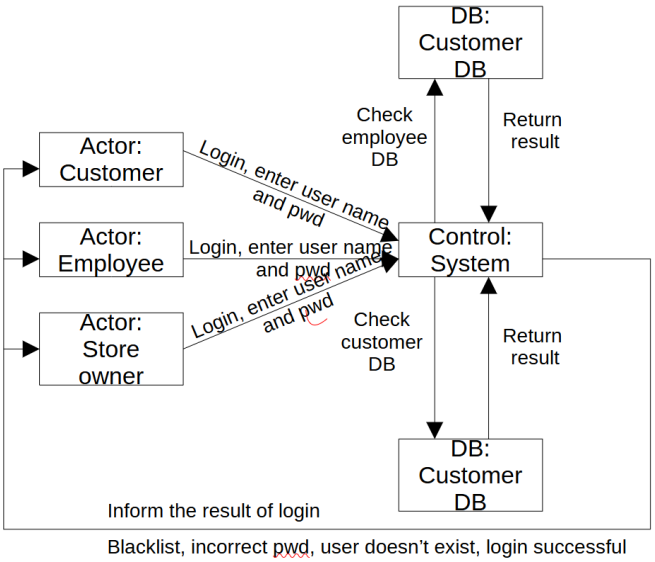
2.1 Login

Normal case:

In normal scenario, customers, employees, and boss can come to the login page and enter their id/password. The system performs checking on the customer DB and employee DB (depending on user) and proceed to the login success state.

Exceptional case:

If the user has a ban a list or if the id/password is correct, the page returns to the login page where they can try again until reaching a fail state.



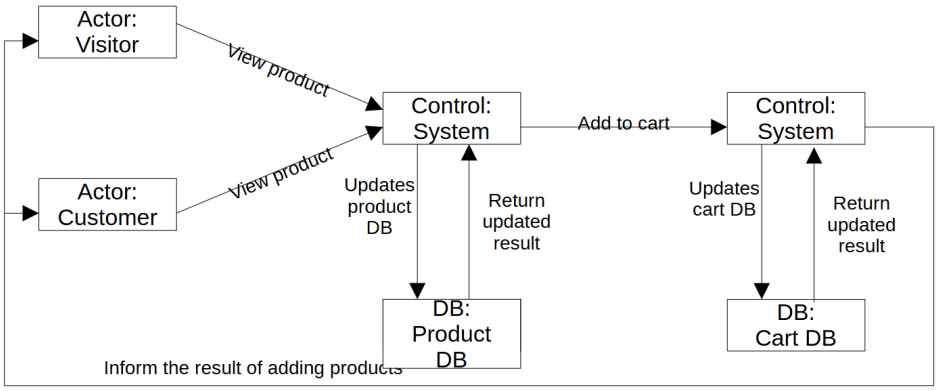
2.2 Add to cart

Normal case:

When logged in, customers and visitors can add items to cart. These data will be added to the customer database as they are merely personal choices.

Exceptional case:

Visitors can add to card with an exception that visitors cannot actually make a purchase unless they later decide to create an account and become a customer.



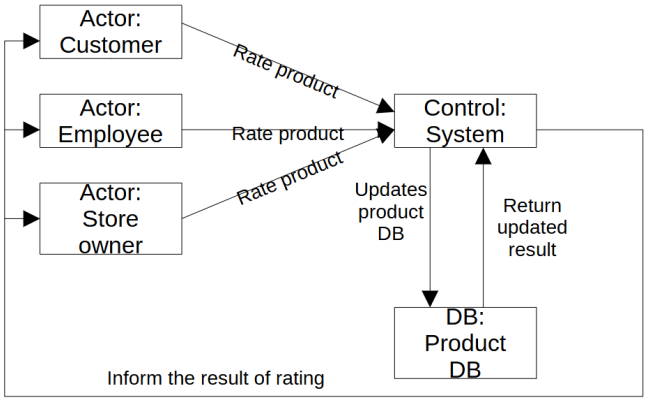
2.3 Rate build

Normal case:

Customers, employees, and the owner (which extends the employee class) can rate builds after purchase. These data will be stored in the product DB which others can see.

Exceptional case:

None



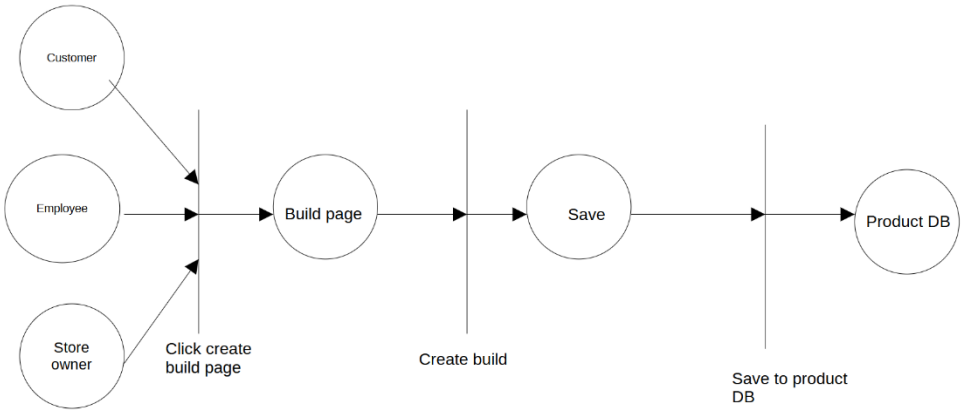
2.4 Create build

Normal case:

Customers, employees, and the owner (which extends the employee class) can create builds. These data will be stored in the product DB which others can see.

Exceptional case:

None



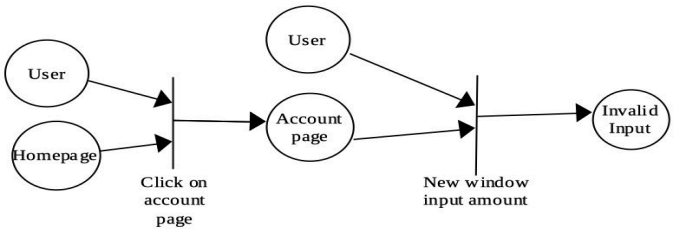
2.5 Transfer Funds

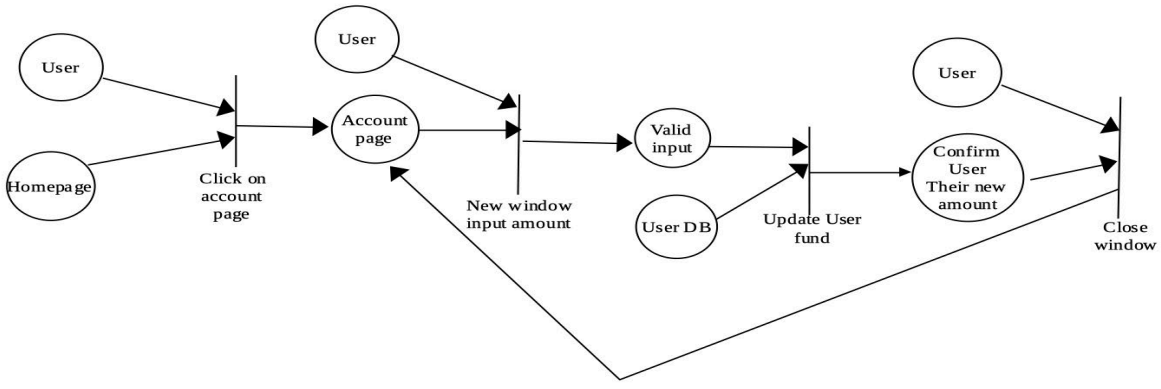
Normal case:

User can be either customer or employee, from homepage, clicks on their account information. From there, they can see how much they have and click on a button to add more funds into their account. A window will appear, prompting them to add their desired amount to their account. Normally, some bank or credit card info would be needed as well. If the appropriate amount is entered, then that amount is taken from their account and added to their account fund. Once the amount is accepted, we confirm with the user, such as a window saying "x amount has been added". The window is closed and the new page reflects the new amount.

Exception case:

Usually, the amount entered is checked with their account to see if the amount can be withdrawn or if it has an illegal input like a character.





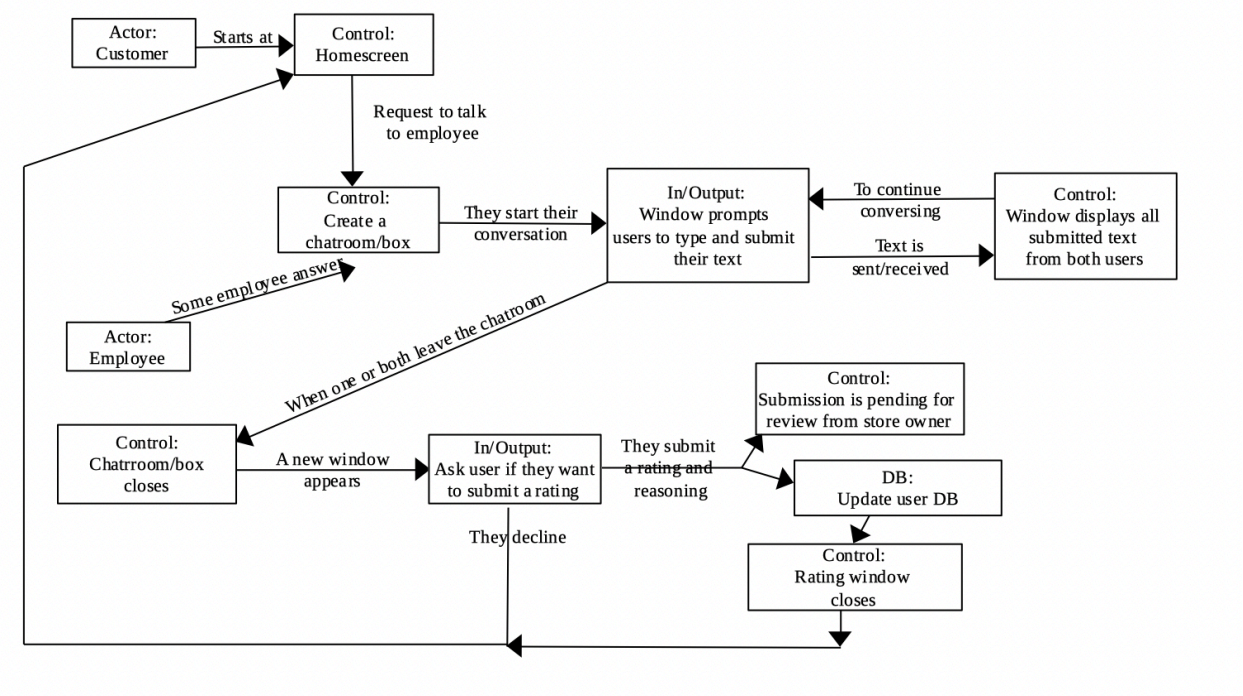
2.6 Customer Communication

Normal Case:

Customers can communicate with employees through some chatbox system. Once they finish their interaction, the either party can rate each other. They can rate each other with either a "compliment" or a "complain". Only customers require some form of justification. The store owner has final decision on how to process with the rating. If accepted, appropriate measures will be taken. If it is a compliment, a "compliment" tally will be added to their account. If it is a complaint, a "warning" tally will be added to their account.

Exception Case:

Either party is not required to rate, so there may be cases where a rating is not submitted. The store owner can also reject the rating to their own discretion.



2.7 Ban/Discount Customers

FYI: ratings are on a 5 star system. A compliment is three 5 starts without any 0 stars, and vice versa for complaints.

Normal Case:

When a customer gets either a "compliment" or "complaint", the system will do the following:

If the customer receives a compliment and now has 3 total, the message will congratulate them and tell them their next purchase has a 10% discount.

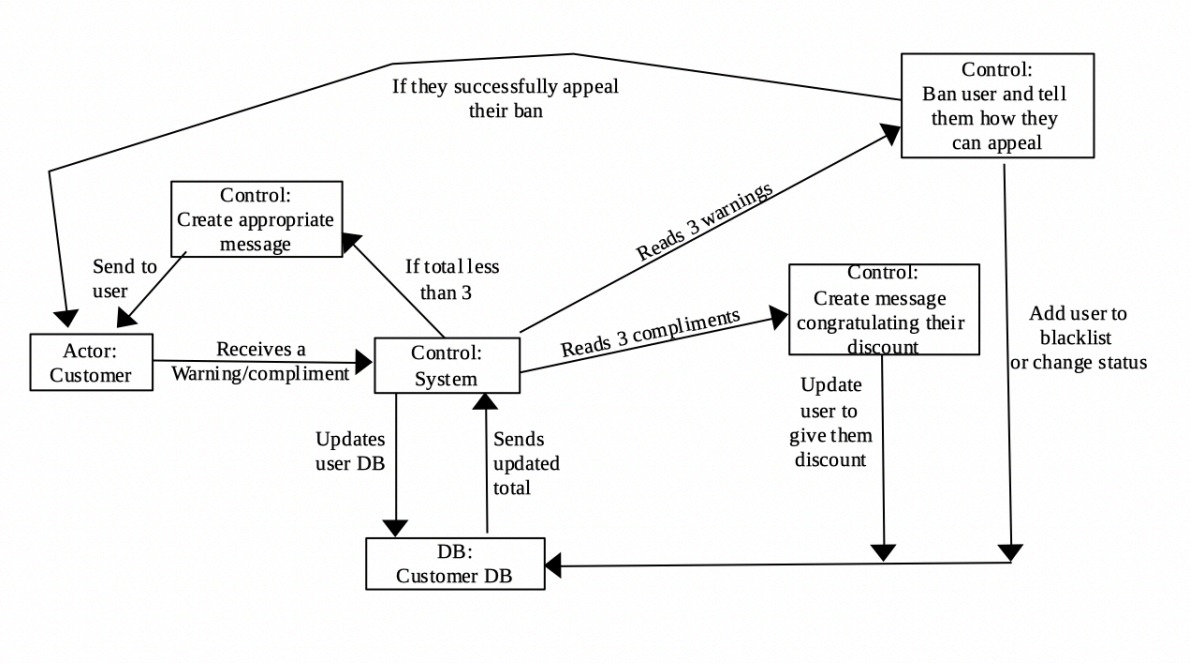
If the customer receives a compliment, but has less than 3 compliment, the message will tell the user how many compliments they have and what will happen when they get 3.

If the customer receives a complaint and now has 3 warnings, the message will tell the customer that they are banned from the website, but they can appeal it in person.

If the customer receives a complaint, but has less than 3 warnings, the message will tell the user how many warnings they have, what will happen when they have 3 warnings, and how to appeal the ban.

Exception Case:

If the customer successfully appeals to their ban, their warning total goes back a number from 0-2.



2.8 Request Account

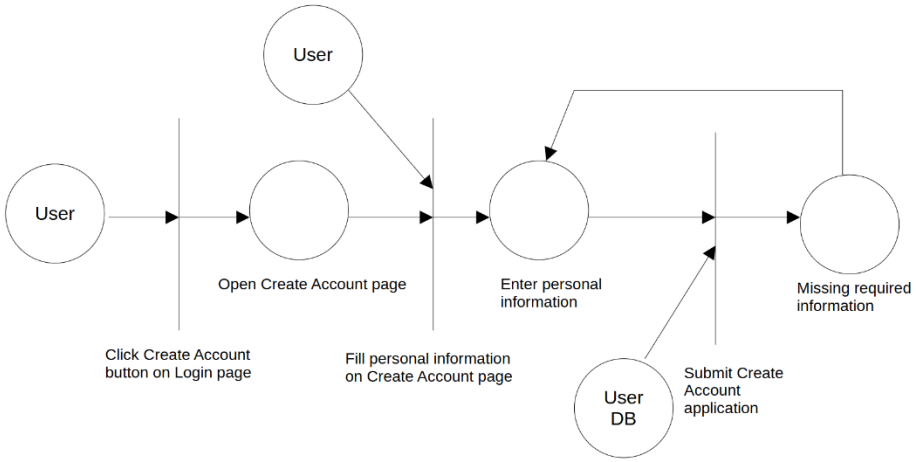
Normal case:

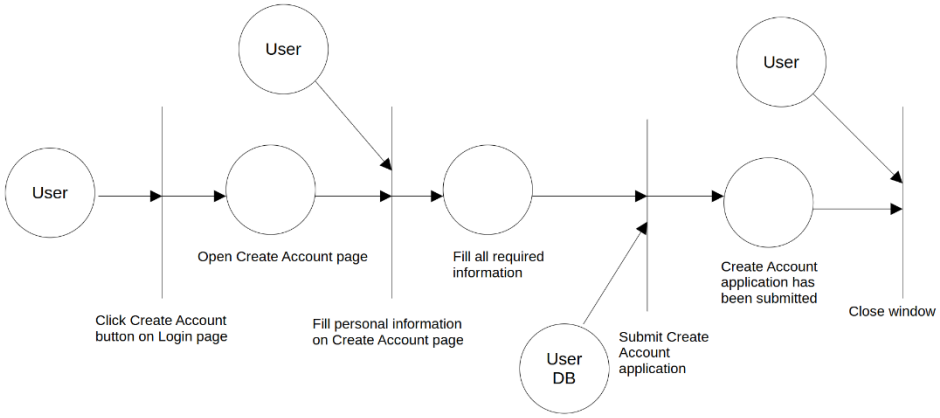
Visitors who want to buy products need to become a customer first. On the login page there will be a button to create an account. When the user clicks it, they will go to the Create Account page. The user needs to fill in required personal information and then submit an application. This application is reviewed by an employee.

Exceptional case:

When the user submits the application, if the user fills in a user email address that already exists in the customer database or employee database, it will be rejected outright, and the user will be reminded of the existence of an account with this email address.

If the user does not fill out the required information completely, the application cannot be processed.





2.9 Purchase

Normal case:

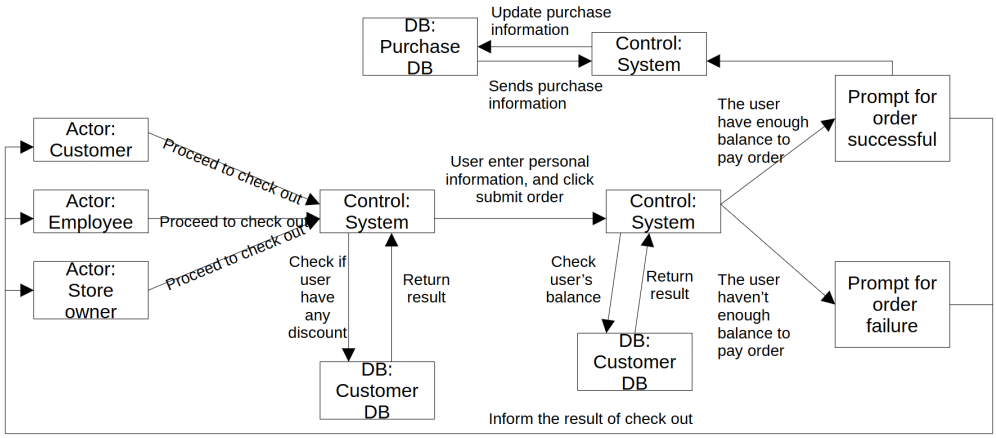
The purchase consists of the cart to checkout section.

Customers and employees can purchase products on the website. The discounts owned by the user will be displayed on the checkout page and the user can choose whether to apply the discount on this transaction or not. On the checkout page, the user needs to fill in required personal details. Then, the user can submit the order.

Exceptional case:

If the user's account balance is less than the total amount of the products purchased by the user for this transaction, the order cannot be submitted. The system should prompt the user to transfer funds.

If the user has multiple discounts, only one of them can be used. Discounts cannot be stacked.



2.10 Adding/removing/promoting/demoting employees

Normal case:

Store owners can add, remove, promote and demote employees.

Employees are classified as assistant, junior, senior, and manager.

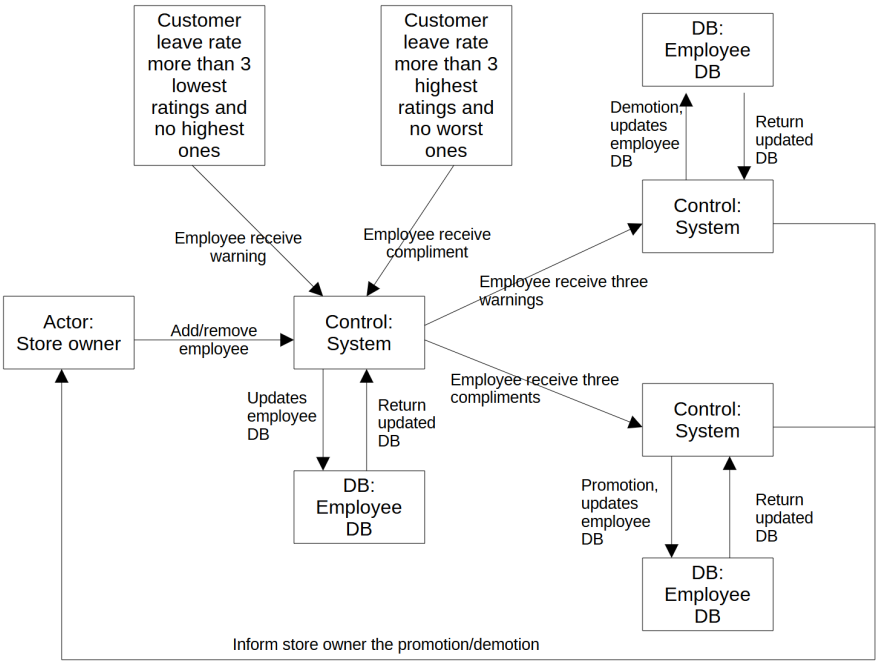
The promotion or demotion of an employee is determined by the warnings or compliments he/she receives. An employee who receives three warnings will be demoted, and an employee who receives three compliments will be promoted.

If an employee is demoted twice, he/she will be removed from the system.

Exceptional case:

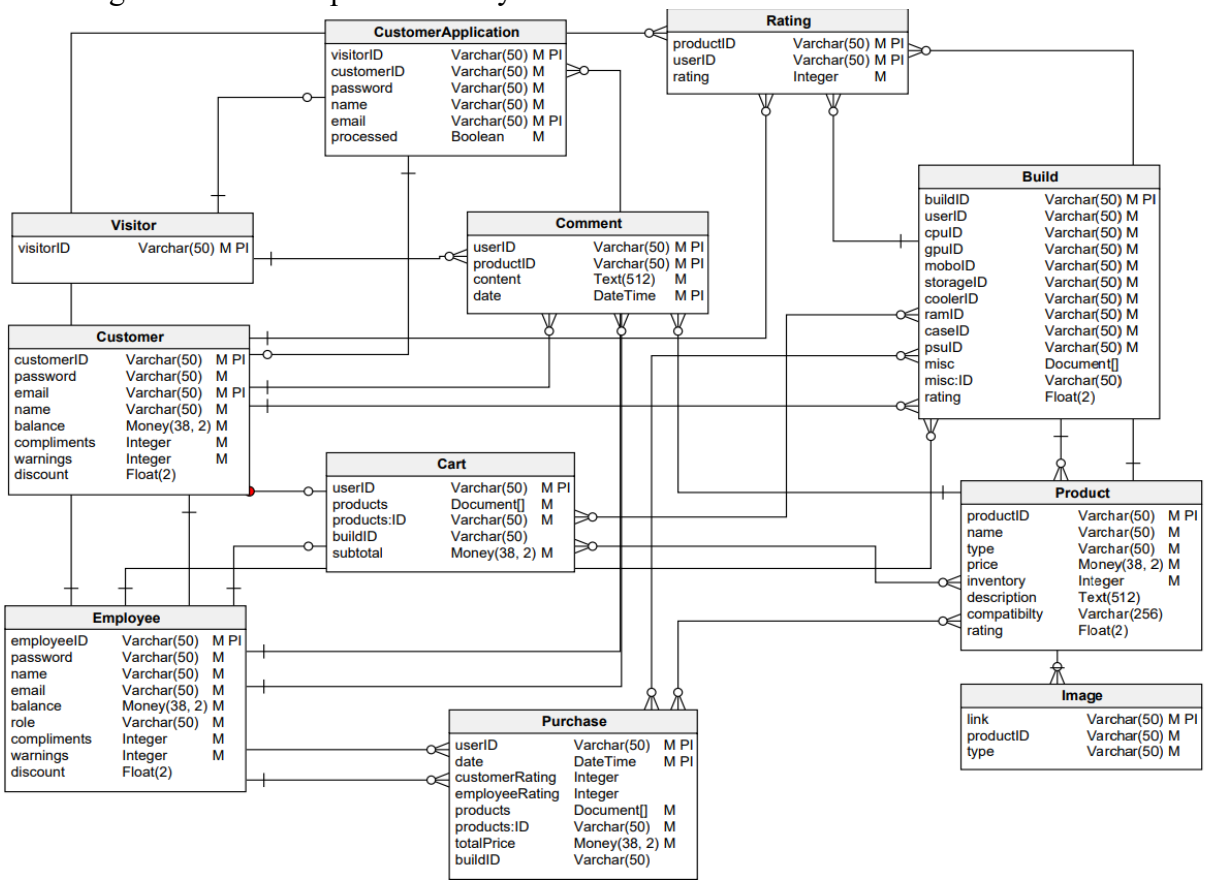
For assistants, they will be removed from the system after receiving one demotion.

Employees can be promoted to manager at most, after that they will no longer get promoted if he/she receives three compliments again.


Customer leave rate more than 3 lowest ratings and no highest ones
Customer leave rate more than 3 highest ratings and no worst ones
DB: Employee DB
Demotion, updates employee DB
Return updated DB
Control: System
Employee receive warning
Employee receive compliment
Employee receive three warnings
Actor: Store owner
Add/remove employee
Control: System
Updates employee DB
Return updated DB
Employee receive three compliments
Control: System
DB: Employee DB
Promotion, updates employee DB
Return updated DB
DB: Employee DB
Inform store owner the promotion/demotion

3. Entity-Relationship Diagram

An Entity Relationship Diagram is a type of diagram that lets you see how different entities (e.g. people, customers, or other objects) relate to each other in an application or a database. Below is the ER diagram of our Computer Store System.



4. Detailed design

4.1 Visitor

- **Assign visitorID**
  Input: Visitor table
  Output: A new visitorID number in sequence saved to browser session and Visitor table
  Description: When a user opens the app, he will be prompted to accept the cookies, using JWT Web Token which is a package manager. The browsing session of each visitor will be saved on the browser along with some of his information that includes a visitorID, the visitor will then be free to browse and leave comments under certain items available on the app and also be able to retrieve and keep their information.
- **New CustomerApplication**
  Input: Customer application form from browser of visitor

Output: New row in CustomerApplication table with relevant information
Description: Form includes user-chosen customerID (i.e. username (unique)), starred-out password, full legal name, and valid email address. When the form is submitted, a new row is added to the CustomerApplication table with this information. customerID must be unique (primary key). If it is not, the user is prompted to choose a new ID. Password is stored using a password hashing function (bycrypt node package manager). An email is sent to the user letting them know the application has been received AFTER the row is successfully added to the table.

- **New Comment**
  Input: userID (employeeID, customerID, or visitorID), productID or buildID, text of comment
  Output: New row in Comment table
  Description: Users may comment on the page of a specific product or build. Submitting the form with their comment calls this function to add a new row to the Comment table. Comment text should be run through the language filter before being saved to the database. A timestamp of when the comment was submitted serves as part of the compound key for this table as users may submit multiple comments on the same product.
- **Start Instant Chat**
  Input: visitorID (from session), user input
  Output: instant messaging window
  Description: At the bottom of the window there should be a button to start an instant chat with an employee. Clicking the button should open a chat window where both end user and employee can type freely and view one another's messages instantly.

4.2 Customer

- **Log in**
  Input: userID (customerID or employeeID), password
  Output: userID (customerID or employeeID) to be used by browser session
  Description: Users type in their log in details and hit submit. The function queries the Employee table first to find a row with a matching employeeID. If none exists, the function queries the Customer table to find a row with a matching customerID. If it does exist, the plaintext password is hashed and compared with the password stored in the relevant Employee or Customer table. If it does not match, the user is again prompted. If there is a match, the function returns the userID to be used by the browser session and the user is brought to the relevant homepage.
- **New Comment**
  Input: userID (employeeID, customerID, or visitorID), productID or buildID, text of comment
  Output: New row in Comment table
  Description: Users may comment on the page of a specific product or build. Submitting the form with their comment calls this function to add a new row to the Comment table.

Comment text should be run through the language filter before being saved to the database. A timestamp of when the comment was submitted serves as part of the compound key for this table as users may submit multiple comments on the same product.

- **Add balance**
  Input: userID from browser session (customerID or employeeID), dollar amount
  Output: update row in Customer or Employee
  Description: Given that a user is logged in, query the Customer or Employee table to find the user's current balance. Add the given dollar amount to this balance to find the new balance and use this value to update the balance attribute.
- **Add to cart**
  Input: userID (from session), productID (or buildID)
  Output: new or updated row in Cart table
  Description: Users may only have one cart. Query Cart table to see if a cart already exists for the userID (either employeeID or customerID). If one exists, update the row to add the productID or buildID for the desired product to the array of items representing the cart. If no row exists, first create one and then do the same. Update the subtotal attribute to reflect the new subtotal by querying Product to obtain the price of the new product and summing it with the existing subtotal.
- **Remove from cart**
  Input: userID (from session), productID (or buildID)
  Output: update row in Cart table
  Description: Query Cart table to see if a cart already exists for the given userID (it should since the option to delete from cart only exists if a cart is being displayed to the user). Remove the productID or buildID of the desired item from the array of IDs representing the user's cart. Update the subtotal attribute to reflect the new subtotal by querying Product to obtain the price of the removed product and subtracting it from the existing subtotal.
- **Display cart**
  Input: userID (from session)
  Output: product details for each product (or build) in cart
  Description: Query Cart table to see if a cart exists for the given userID. If none, display a helpful error message. If one exists, display the name, price, and image for each item in the user's cart. The image will require using the getImage method.
- **Check out (make purchase)**
  Input: userID (employeeID or customerID) with non-empty cart and sufficient balance
  Output: updated row in either Employee or Customer, updated row in Cart, receipt displayed on page and sent to user email
  Description: Query Cart table to ensure that userID has a row with at least one value in the productID list. Calculate grand total of cart based on subtotal, tax + shipping, and discount. Query Employee or Customer table to find the user's balance and compare it with the grand total. If there is not enough in the balance to cover the cost, display a helpful error message and end function. If there is enough, display a confirmation to the

user (yes or no). If no is selected, return the user to the cart and display a message. If yes, subtract the grand total from the user's balance and update the user's row in the correct table to reflect this. Next, create a receipt that displays the name and price of each product along with the subtotal, tax, shipping, and grand total of the purchase. Display this receipt to the user and also send it to the email address of the user by querying the appropriate table to find the email attribute. Remove all productIDs and buildIDs from the row in the Cart table that has the userID of the user as its primary key. If the buyer is a customer, update the appropriate row in the Customer table to have a discount attribute with value 1 (remove future discount if one was applied to this purchase). Query the Product table and, for each productID that was purchased, decrement the inventory attribute by 1. Finally, query the Purchase table and create a new row with all of the information necessary including a timestamp.

- **Rate Product (or build)**
  Input: userID (from session), productID (or buildID), rating (1-5)
  Output: new or updated row in Rating table, updated row in Product table
  Description: Query Rating table using userID and productID. If a row exists, update the rating attribute to the new rating using the given value. If no row exists, create a new row with the userID, productID, and rating. Next, query the Rating table and average all of the rating attributes with a matching productID to find the new average rating for the product. Query the Product table and update the rating attribute of the product with this new value.
- **Start Instant Chat**
  Input: customerID (from session), user input
  Output: instant messaging window
  Description: At the bottom of the window there should be a button to start an instant chat with an employee. Clicking the button should open a chat window where both end user and employee can type freely and view one another's messages instantly.

4.3 Employee

- **All methods of Customer are accessible by Employee. Use employeeID instead of customerID.**
- **Recommend New Build**
  Input: employeeID, productIDs of required components: CPU (Central Processing Unit), GPU (Graphics Processing Unit), mobo (motherboard), storage, RAM, case, PSU (not required: cooler, misc.)
  Output: buildID, new row in Build table
  Description: Form on employee-only page to be filled out with product IDs of required components. When submitted, generates a buildID by querying the Build table, finding the highest buildID, and incrementing by 1. Calls checkCompatibility to ensure that all parts function with one another. Queries the Build table again to add a new row with the required information along with the employeeID of the employee creating the build.

- **Review Customer Applications**
  Input: employeeID (from session), CustomerApplication table
  Output: potential removed rows in CustomerApplication, potential new rows in Customer
  Description: Query the CustomerApplication table and present all information to the user (up to 5 rows with option to view next 5 rows). Next to each application (represented by a row) include 2 buttons for approve and deny. If "approve" is clicked, query the Customer table and add a new row with the customer's information before deleting the row in the CustomerApplication table and refreshing the page to no longer display that application. A brief email should also be sent to the user's email address to let them know their application has been approved. If "deny" is clicked, query the CustomerApplication and delete the appropriate row before sending a brief email to let them know their application has been denied. Denying an application will also send a similar email to the store owner.
- **Reply Instant Chat**
  Input: employeeID, user input
  Output: instant messaging window
  Description: When an employee is logged in and a visitor or customer starts an instant chat, a notification should appear in the employee's window. Clicking the notification should open a chat window where both end user and employee can type freely and view one another's messages instantly.

4.4 Owner

- **All methods of Customer AND Employee are accessible by Owner. Use employeeID instead of customerID when relevant.**
- **Add New Employee**
  Input: employee details (employeeID, password, name, email, role)
  Output: new row in Employee table
  Description: Upon form submission, query the Employee table and add a new row with the given information. Include password hashing for the password and ensure that the chosen employeeID is unique before adding. Discount should be set to 0.9.
- **Apply Compliments/Warnings**
  Input: userID of party receiving compliment/warning (customerID or employeeID), number of compliments/warnings
  Output: updated row in Customer or Employee table
  Description: Query the correct table (Customer or Employee) and increment either the compliment or warning attribute by the number provided by the owner.
- **Add New Product**
  Input: product information (name, type, price, description, compatibility, inventory, images)
  Output: new row in Product table
  Description: Generate a productID by querying the Product table, finding the highest productID, and incrementing by 1. Upon form submission, query the Product table and add a new row including the necessary information (name, type, price, description,

compatibility, inventory). Then, query the Image table and add rows for each image along with the correct productID that was generated.

4.5 Product

- **Display Product**
  Input: productID
  Output: all information related to product including name, price, type, inventory, rating, compatibility, description, and all images of product
  Description: Query Product table to find matching productID. If none exists, display a helpful message. If it exists, display all information related to the product including name, price, type, inventory, compatibility, and description. Average rating can be determined by summing all ratings in the array and dividing by the count. Query Image table with productID to obtain all links with images related to the product.
- **Display Build**
  Input: buildID
  Output: summary of all products in build
  Description: Query the Build table to obtain a list of all productIDs in the build. For each product, query the Product table and retrieve the name and price as well as query the Image table to retrieve the primary image. Display the resultant build summary to the user.

4.6 System

- **Filter Language**
  Input: user-submitted text
  Output: the same text but with blacklisted words starred out
  Description: The function iterates through the given text and replaces any and all blacklisted words with stars (*). The list of blacklisted words is stored as a .txt file.
- **Get Image**
  Input: productID (or buildID), type (of image)
  Output: web link with image of desired product
  Description: Query the Image table using the given productID and the type of image (e.g. primary, closeup, box, etc.) to return the link of a web-hosted image. Called by displayProduct, displayCart
- **Check Compatibility**
  Input: list of productIDs
  Output: productIDs that are incompatible with one another, nothing if all good
  Description: Query the Product table and compare the strings stored in the compatibility attribute of all products in the list. If there are any conflicts, display a warning to the user. A list of conflicts is stored in a .txt file.

- **Assign Compliment/Warning**
  Input: userID (employeeID or customerID), Rating table
  Output: potential for updated rows in Customer or Employee
  Description: Query the Rating table looking for matching userID. Sum the number of 1-star ratings and separately sum the number of 5-star ratings. If there are no 1-star ratings, divide the number of 5-star ratings by 3 (round down) and query the Customer or Employee table, updating the compliment attribute with this value. If there are no 5-star ratings, divide the number of 1-star ratings by 3 (round down) and query the Customer or Employee table, updating the warning attribute with this value.
- **Determine Customer Discount**
  Input: customerID
  Output: boolean (1 means 10% discount on next purchase)
  Description: Query the Customer table and retrieve the number of compliments. Take this number %3 (modulus operator). If the result is 0, query the Customer table again and update the discount attribute to be 0.9. This ensures that every 3 compliments that a customer receives, they gain a 10% discount.
- **Determine Promotion/Demotion**
  Input: employeeID
  Output: potential update row in Employee table
  Description: Query the Employee table with the given employeeID and retrieve both the compliments and warnings attributes. Divide each by 3 (round down). The employee receives a promotion for each of the first numbers, denoted by incrementing the "role" attribute. The employee also receives a demotion for each of the second numbers, denoted by decrementing the "role" attribute. If the number of demotions exceeds 2, the employee will be fired, sending an email to the store owner notifying them of this occurrence.

5. System screens:

The following figures show a few of the main screens of our system.

Figure 1: Login Page

Figure 2: Product Page

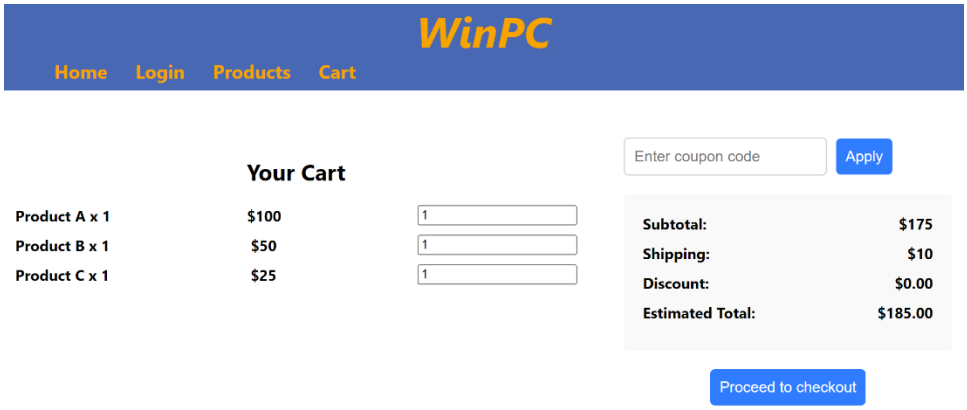


Figure 3: Cart Page

6. Minutes of Group Meeting

Meeting Dates	Discussion
2/25/2023	Discussion of spec requirements
3/24/2023	Discussion of Phase I Report
4/21/2023	Discussion of Phase II Report
4/23/2023	Discussion of diagrams
4/24/2023	Discussion of system's GUI

7. Supporting Information

The Computer Store System described above is being developed by the WinPC team – Oumar Barry, Ian Niles Frost, Zhi Gao, Max Sehaumpai and Richard Yeung

GitHub link: https://github.com/OumB2021/Computer-Store