Store

CIS-17B

April 30, 2023

Danielle F

# Table of Contents

[Introduction](#_Introduction) .………………………………………........3

[Development Summary](#_Development_Summary) .……………………………......3

[GitHub](#_Reference) ….……….……………………………………...5

[Sample Inputs/Outputs](#_Sample_Outputs:) .…………………………….......5

UML Chart ……………………………………………..9

Flow Chart .…………………………………………….10

Version Summaries ….………………………………10

[Pseudo Code](#_Pseudo_Code) …………………………………………...11

[Reference](#_Reference) .……….…………………………………......13

## 

# **[Introduction](#_Table_of_Contents)**

This program simulates placing an online food order from a restaurant.

# **Objective**

Login as a user and place an online order from a sushi restaurant.

# **Rules**

* A user must have an account to access the Store.
* To sign-up for an account:
  + Name must be at least 2 characters long.
  + Email must be at least eight characters long and must contain an ‘@’ and ‘.’
  + Password must be at least 7 character long, contain a capital letter, one special character (!, $, #), and at least one digit.
  + Must confirm password by retyping it in again.
* User must sign-in to their account correctly to place an order.

## **[Development Summary](#_Table_of_Contents)**

|  |  |
| --- | --- |
| Lines of code  (including h files) | 2,226 |
| Comment lines | 264 |
| Blank lines | 137 |
| Total lines of source file | 1835 |

This project covers chapters one through thirteen in the [textbook](#_Reference). The primary requirement for this project was that everything needed to be in classes. I based this build off my survey project.

My first hurdle came anytime I updated the Admin and User classes in either my group project, Yahtzee or in my survey project, because I had to update it the other two projects.

My next hurdle was how to display the sushi items to the user. This project was built off my survey project. The survey printed each question and saved each answer one at a time, but I wanted all my menu items to be displayed before prompting the user for their selection. I settled on asking the user to select an item number and then prompted for quantity. I saved their choices to an integer array, but the meaning of the values in each element changed. In the survey project, the value in each element represented their answer to the question, answer 1, 2, or 3. For the store, each element represents the quantity the user wants for that item.

The third obstacle was where the user’s invoice should be displayed when they were ready to checkout. At first, I had it printing inside store.cpp, but I moved it to the admin.cpp because it’s the only class that can read the binary file. The user saves their order to their profile and then writes their record to binary. The Admin class confirms the order and displays a checkout confirmation.

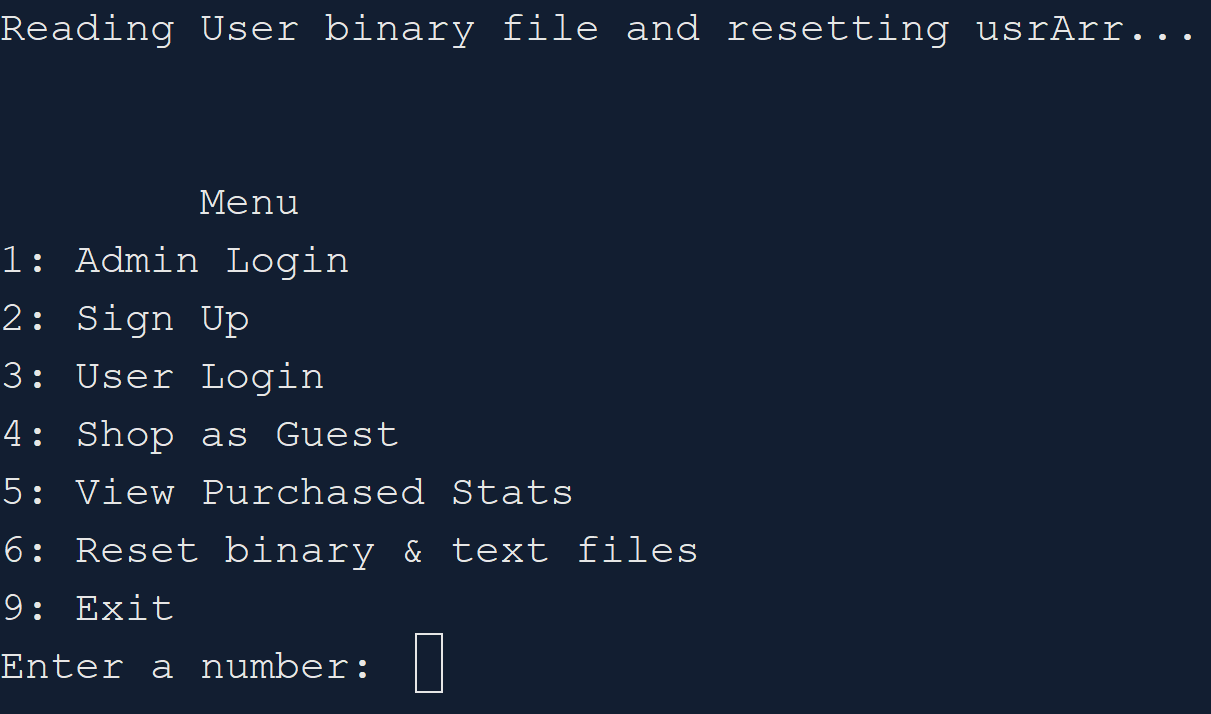
The fourth hurdle came as a result of changing what the values in the cart array represented and how the statistic chart was calculated. I had to add totalItems and cartTotal to the Cart Class to help keep count of the how many times each item had been bought. Admin::printItemStats( ) was simplified because I only have three items. Whereas the survey had three questions and every question had three options to keep count for.

## [**GitHub Repository**](#_Table_of_Contents)

#### <https://github.com/koa2019/e-store>

#### **Sample Inputs:** 3, homer@simp.com, Homer!23, 1, 2, 3

#### **[Sample Outputs](#_Table_of_Contents):**



Text

Description automatically generated

Text

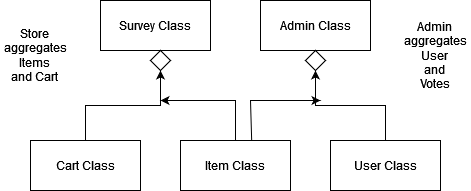
Description automatically generated

Text

Description automatically generated

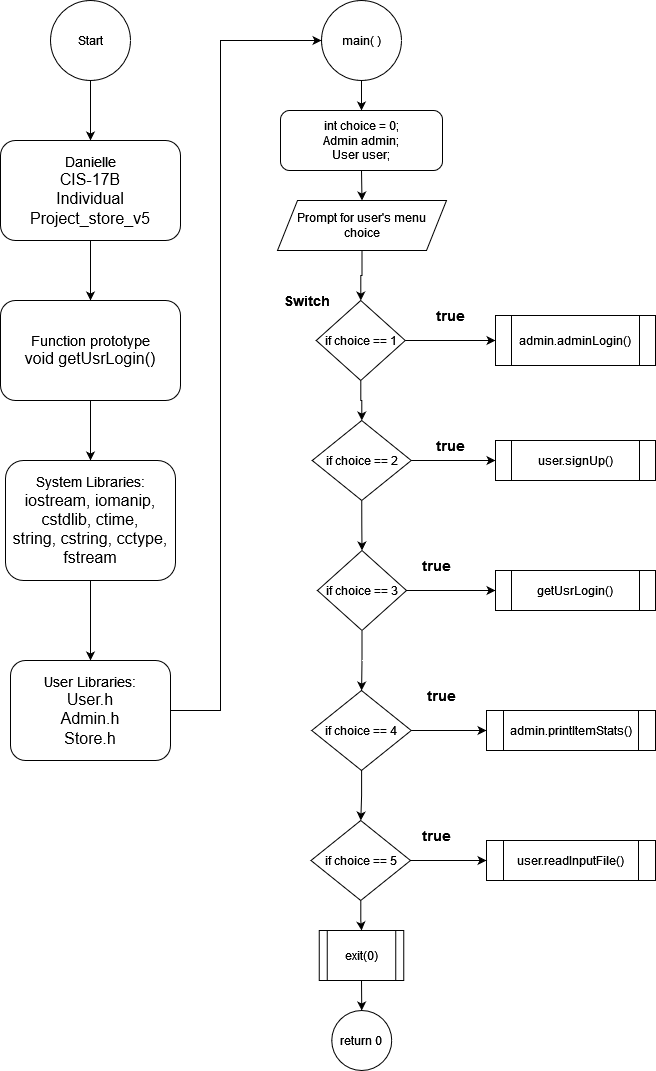
## **[UML Chart](#_Table_of_Contents)**

The full version of this chart is in the store>docs>charts folder.



## [**Flow Chart**](#_Table_of_Contents)

The full version of this flowchart is in the store>docs>charts folder.



### **[Version](#_Table_of_Contents) Summaries**

#### **Version 1.1**

* This is based off of yahtzee\_v24\_moveDice2SC.

#### **Version 1.3**

* Merged yahtzee\_v28.2's Admin & User Classes with store\_v1.2

#### **Version 2**

* Is a copy of survey project v6.
* Changed Question class to Item class, Votes class to Item class.
* Added Item:: float price and nItems = 3 instead of 4. Because I needed a float.
* Added Cart::cartTotal
* Prints total and total number of items in Survey::start()

#### **Version 3**

* Made Survey::Item item[] \*\*item;
* Printed checkout totals in Survey::start()

#### **Version 4**

* Calculated & printed every items purchase history in Admin::setItemHist()

**Version 5**

* Refractored Survey class to Store.
* copy2Usr() saved cart[] elements to user2.
* Added getCheckout() in Admin. It prints the user's order.
* Fixed BUG: Cart::cartTotal not adjusting total when item quantity is updated in store, by creating temp integer array to temporarily saves the user's order until they're ready to checkout.
* Added Cart::totalItems.
* Set each player's totalItems and cartTotal after binary file is read. Inside of Admin::setItemStats().
* Prints percentages in printItemStats().

### [**Pseudo Code**](#_Table_of_Contents)

1. **Parent: Admin class**
   1. Dilemma. Option 1: User class will be the parent and then the Admin and Player classes will inherit User?
      1. Admin & Player both use User class’s variables & functions, but how will the Player ask Admin to test their hiScore variable?
   2. Dilemma. Option 2: Admin is the parent, list wrtBinary() as private and then User inherits Admin?
      1. Admin doesn’t need hiScore when it creates a new object for itself, but it needs to be able to reassign hiScore.
   3. Admin object:
      1. Private members:
         1. Static int num records / Doesn’t hold value between runs /
         2. unique ID
         3. name
         4. password
         5. email
         6. hiScore ? Should this be in User or Admin?
         7. readBinary()
         8. checkHiScore()
         9. setHiScore()
   4. Read inputs for 1 Admin record:
      1. Confirm inputs
      2. Save 1 admin record to object
      3. Write the object to text and binary files.
      4. Print message if successful or not
   5. ONLY Admin is allowed to read binary files
   6. Admin login()
      1. Read first record in Admin’s binary file
      2. Save it to an object
      3. Test login
      4. Print message if login was correct or not
      5. If login is correct, then allow them access to admin only functions
   7. Read User’s binary file:
      1. User login()
         1. Find a record by email
         2. Save it to an object
         3. Test login
         4. Print message if login was correct or not
         5. If login is correct, then allow them to view their profile and play game
      2. Be able to delete/edit a record or member’s of record
         1. Accept an ID or record as an object AND score from User, test if it’s bigger than their current hiSCore and update it accordingly
2. **Child: User class**
   1. Create User Class. User and game classes will have to be combined at some point.
   2. User object:
      1. Private members:
         1. Record #
         2. name
         3. password
         4. email
         5. voteSiz
         6. votes[ ]
         7. dice array or vector
   3. Sign up for a new account. Read in name, email, password
      1. Confirm user inputs before saving their info to object
   4. Write and append each User to a binary file
      1. ONLY has permission to write to binary file
   5. Write and append each User’s to text file
      1. ONLY has permission to write to text file
   6. Play game as guest or as logged in user
   7. When they win they should send their ID or their record as an object AND their current score to Admin, so then Admin can test if it’s bigger than the hiScore saved to their record
3. **Store**
   1. Loop through each question.
      1. Print 1 question.
      2. Print 3 possible answers.
      3. Save each answer in Items array.
      4. Rewrite items array in binary file and text file.
      5. Confirm user’s profile is updated by reprinting their profile.
      6. Print checkout with user’s items listed and their total.
      7. Print a chart of total times for each item has been purchased.

## [**Reference**](#_Table_of_Contents)

1. Gaddis, Tony. *Starting out with C++: From Control Structures through Objects*. 9th ed., Pearson, 2018.
2. Lehr, Mark. “2023\_Spring\_CSC\_CIS\_17B · ml1150258/2023\_spring\_csc\_cis\_17b.” GitHub, 2023,<https://github.com/ml1150258/2023_Spring_CIS_CSC_17B>.