## Title

Retail Store Management System

## Members

- David Miller
  - Manage Stock System (Backend)
  - Sell Item/Manage Invoice System (Backend)
- Nazmul Rabbi
  - Employee Login System (Backend + Frontend)
  - Manage Stock System (Frontend)
  - Sell Item/Manage Invoice System (Frontend)

## Introduction

The Retail Store Management is an application that aims to simplify the day to day operations of a retail store. This application is inspired by the retail management software used in Best Buy retail stores all around the country.

## Front End/User Interface Design

The application frontend design is divided into two parts, the login window, and the main interface window. The login window allows an employee to securely get access to the retail store management software and keeps nonemployees out. Once the employee gets through the login interface they are then redirected to the main application interface. The main interface is dynamic so that means everything happens in one window/interface. In the spirit of multitasking, the left-hand side of the main interface is dedicated to the employee to manage stock and the right-hand side is dedicated to selling item and managing invoice. The employee can log out of the application once they are done using it from the main window and it redirects to the login screen thus ending the cycle of the application.

## Back End Design

The main class in the back end is our Inventory class. This is the class that performs most of the operations on the database and stores all of the information. It has two important members named invoices and stock which store each invoice and item in the database respectively. We decided to use a map for both of these as searching for a particular item using a stl map is O(log n) which is better than just using a list or vector and having O(n). We store each item in an instance of the Item class. This class is a basic wrapper class that stores data for each item such as brand, category, name, sell price, buy price and stock count as well as setters and getters for each. The invoice class is a little more complex. It has a vector which stores all of the items on that particular invoice (vector because easier to iterate over and invoices will almost always have very few different items on it, especially for retail). It has other members including the filename/id and the customer/supplier name. The Invoice class, besides storing all of the needed data for one invoice, can output the data into its given filename in the invoices folder. The Inventory class uses the methods these two classes provide to perform all of the operations including looking up an item, editing an item's attributes, printing an invoice to the terminal,reading in and writing the item database and all previously generated invoices, generating an invoice, removing items from stock that are being sold as well as being able to handle errors (along with the interface program) a user might encounter such as an invalid item/invoice being chosen, reading an invoice twice, buying or updating an item that doesn't exist, etc. We decided to not have the invoice class extend the inventory class as we originally planned because we realized that the invoice class would inherit a lot of methods it would never use and it would still need to define most if not all of the methods it would actually need. So our class hierarchy is the inventory class has_many of both the item and invoice class.

The employee login system works in the backend by reading in a database file that includes the username and password of all users. The data gathered is then converted to a Stl list and is compared against the user input to verify user authenticity.

**Features**

- Employee Login System
    - Allows Employees to log in and out of the system
    - User access control system which allows only allows an employee to interact with the system
    - Tracks which employee has logged in

- Manage Stock System
    - Lookup Stock
        - Allows to check if an item is available in stock
    - Add Stock
        - Allows an item to be added to the stock
    - Update Stock
        - Allows an item on the stock to be update
    - Delete Stock
        - Allows an item on the stock to be deleted

- Sell Item/Manage Invoice System
    - Sell Item
        - Allows an item from the stock to be sold to a customer
        - Invoice is automatically generated after a sale
    - Customer Invoice
        - Allows to look up an invoice

    - Delete Invoice
        - Allows to delete an invoice

    - Shipment Invoice
        - Allows to receive incoming shipment invoice

**Testing**

The login system was tested with a database that held over 50,000 datasets. The stock data was tested with over 10,000 datasets. The query for any data for the login and stock is returned to the program within a second.

**Status**

All the features have been successfully implemented as planned during the first week of the term project. Due to time constraints, no other features were added to the project.

**Conclusion**

We have reached our required objectives within the original deadline of Friday the 8th. We are more than satisfied with our final version of the application.