# Description (MK)

The system we've developed is built upon the stock management system design I created last semester. Our inventory tracking portal is designed to facilitate quick navigation between different inventory tables. We opted for a layout featuring a navigation bar on the left-hand side of the program, allowing users to easily switch between panels. The system has essential components such as stock items, orders, sales pages, and a settings page where users can modify their account password.

We used a card layout for the main panel to display relevant data, utilizing buttons on the navigation bar to switch between panels. In terms of design, we chose a white theme accented with purple. The background across all pages is white, while the navigation bar and JLabel borders on the home page are purple to make them stand out on the white background. In future iterations, I would like to implement a feature within the settings page to enable users to customize the color scheme throughout the program as well as create a dark mode for the system.

To visualize data on the home page, we integrated the JFreeChart library, selecting a line graph to effectively present the total revenue from sales over the past three months. Additionally, we included some basic information on this screen, such as the number of suppliers and items in the database, the total sales, the most recent order, and the best-selling item. Each label is dynamically updated through SQL queries as new information is added to the database.

Each table within the portal displays corresponding data from the database and features options to add, delete, and amend entries, facilitating seamless interaction with the database. For the sales and orders tables we used Jcomboboxs for the user to only be allowed to enter StockIDs which are present in the database to avoid any complications. On all our pages the user must first click on the entry they wish to amend as once pressed the entry gets auto filled with the data from the entry and then the user can change the information.

# Requirements (DM)

|  |  |
| --- | --- |
| Name | Refresh table |
| Requirement | Table dynamically refreshes upon database changes |
| Preconditions | User must have established connection to database |
| Steps | 1. User opens application. 2. Login screen prompts user to enter valid login 3. User navigates to the database they wish to use 4. User deletes, adds, or amends entry 5. Validation occurs to ensure the correct data has been entered 6. Query is put through the database and calls for the program to create a new instance of table class which now includes changes |
| Expected Result | A new table is generated and replaces the old one, thus including changes made by user |

|  |  |
| --- | --- |
| Name | Ensure unique primary keys |
| Requirement | Every entry made has its own unique primary key |
| Preconditions | User must have established connection to database |
| Steps | 1. User opens application. 2. Login screen prompts user to enter valid login 3. User navigates to the database they wish to use 4. User is not given access to create or modify the primary key in each respective table 5. When a new entry is made the new entries primary key is autoincremented 1 higher than the previous entry |
| Expected Result | Every single entry has its own unique primary key and upon creation |

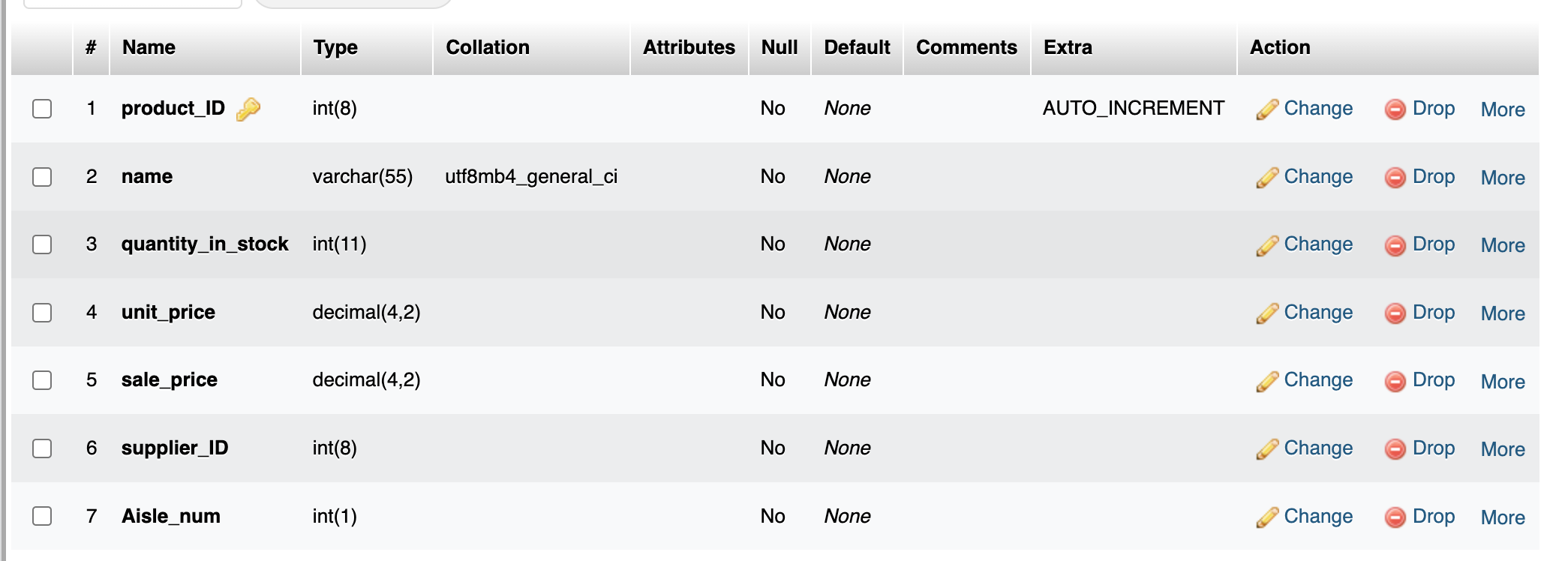
|  |  |
| --- | --- |
| Name | Navigation |
| Requirement | User is able to navigate to each screen |
| Preconditions | User must have established connection to database |
| Steps | 1. User opens application. 2. Login screen prompts user to enter valid login 3. Program creates instance of every single panel but sets all but main menu invisible and navbar 4. When a tab in the navbar is pressed it will make the respective tab visible and the rest invisible |
| Expected Result | User will be able to navigate to each section of the program |

|  |  |
| --- | --- |
| Name | Reset password |
| Requirement | User is able to reset password |
| Preconditions | 1. User must have established connection to database 2. User must know existing password |
| Steps | 1. User opens application. 2. Login screen prompts user to enter valid login 3. User navigates to settings bar 4. User is then prompted with three fields, old password, new password, and match password. 5. When user enters these details a query is made to the database to ensure “old password” field matches password found in Staff table. Then it will check if new and match password are the same. 6. When all validation is performed a query is performed to replace “old password” with “new password. |
| Expected Result | User resets password |

# Database

## Stock Items / Inventory

### Structure



### Populated Data

A screenshot of a computer

Description automatically generated

## Employees

### Structure

A screenshot of a computer

Description automatically generated

### Populated

A screenshot of a computer

Description automatically generated

## Sales

### Structure

A screenshot of a computer

Description automatically generated

### Populated

A screenshot of a computer

Description automatically generated

## Orders

### Structure

A screenshot of a computer

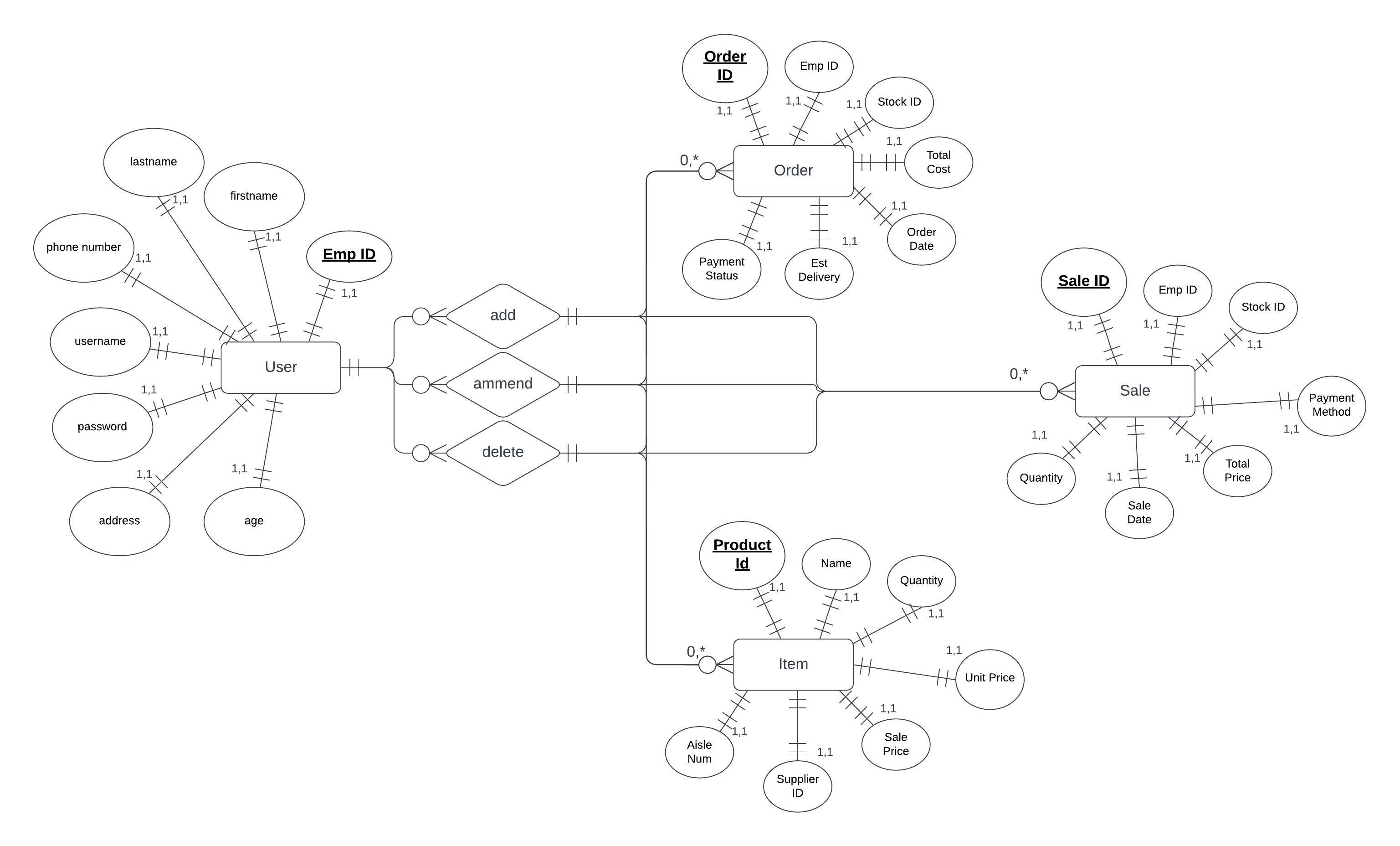
Description automatically generated

### Orders Populated

A screenshot of a computer

Description automatically generated

# ER diagram(D)



# Interesting source code snippets

A screen shot of a computer program

Description automatically generatedmade a for loop for connecting to the database as we both used different software for hosting the database. The loop try’s to connected to each of the ports in the array to see if they are valid.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer code

Description automatically generated

Used for loops to create a method of dynamically creating a SQL query depending on the requirements and then checking whether the value that you wish to enter into the database is of the correct type before it is entered.

A computer screen shot of a program code

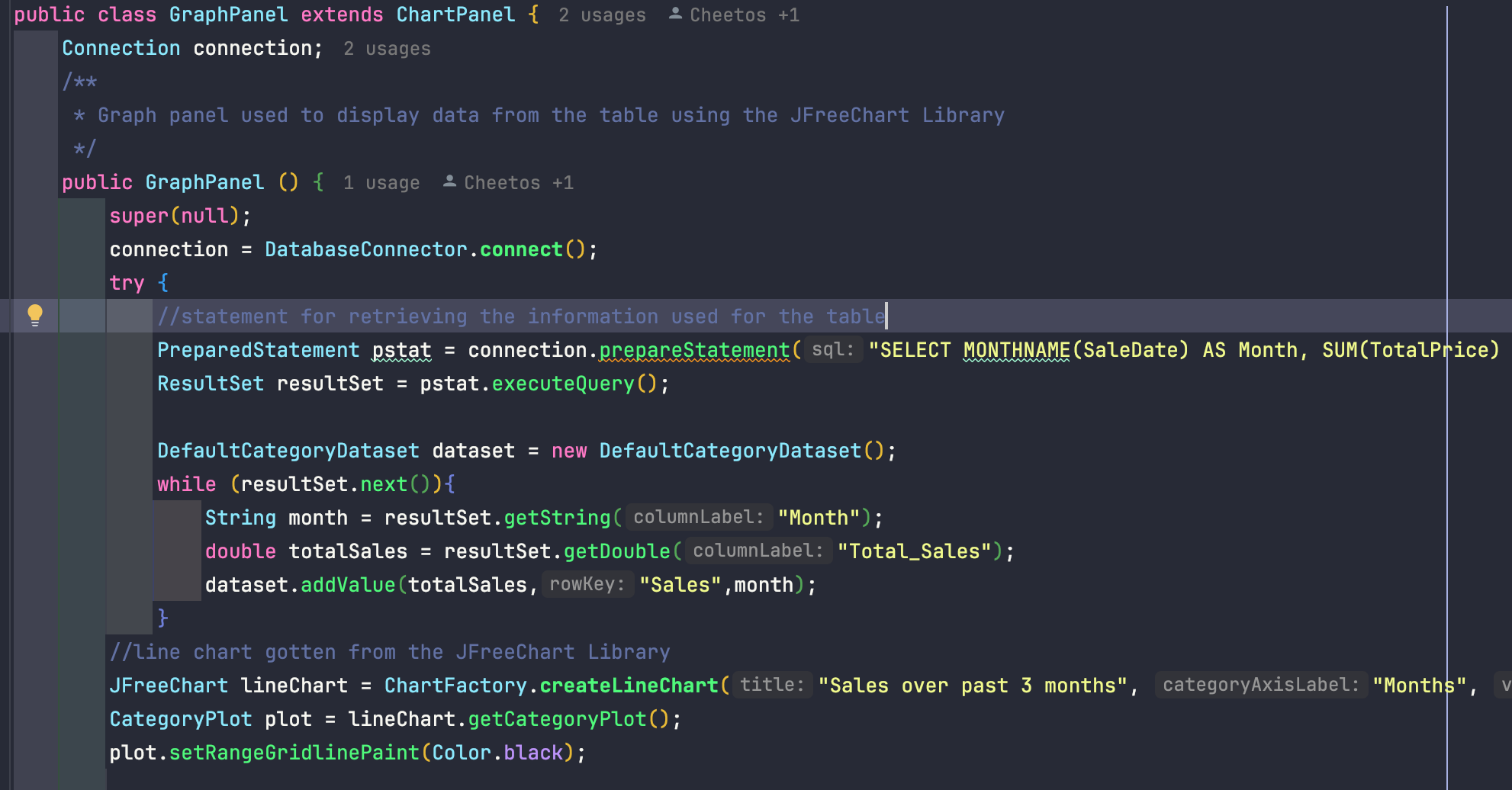
Description automatically generated

A method made for hashing the password using the MD5 method before saving the password into the database for protection. Uses a built-in encryption method from the messageDigest class. . Then a password is turned into an array of bytes which gets connected by a stringBuilder.

A computer screen shot of a program code

Description automatically generated

Created this file which extends from the JTextField class to create a JTextField that only takes in numbers and created another for only Letters. The way this works is it overrides the default model for the JTextField which is a file which takes arguments from the PlainDocument class. The insertString method is overridden from the PlainDocument class to only take Letters into the string. If the character is a letter, it will print it onto the JTextField field otherwise nothing will happen.



We used a Library For charts called JFreeChart. It hosts a collection of different graphs that can be used for displaying data. We settled for the line chart which displays on the home panel a chart showing the sales of items over the last 3 months.

# Tests

### Junit Tests for loginLogic

A screenshot of a computer

Description automatically generated

### Unit Test for Deleting From Database

A screenshot of a computer program

Description automatically generated

### Unit Tests for Querying the Database for data

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated