GROUP ASSIGNMENT

Technology Park Malaysia

CT038-3-2-OODJ

OBJECT-ORIENTED DEVELOPMENT WITH JAVA

APD2F2305CS(DA) , APD2F2305CS(CYB)

HAND OUT DATE : 30 June 2023

HAND IN DATE : 10 September 2023

WEIGHTAGE : 50%

INSTRUCTIONS TO CANDIDATES:

1 Students are advised to underpin their answers with the use of references (cited using the American Psychological Association (APA) Referencing).

2 Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld.

3 Cases of plagiarism will be penalized.

4 Submit the assignment to APU Learning Management System.

5 You must obtain 50% overall to pass this module.

Name	TP Number	Intake Code
Teh Chen Ming	TP068804	APD2F2305CS(DA)
Aidan Chang Wai Yue	TP063411	APU2F2305CS(DA)
Haaerish A/L Kumar	TP063626	APD2F2305CS(CYB)

Abstraction

Sigma SDN BHD (SSB) is a growing wholesaler in Johor Bahru, it is works on distributing groceries, fresh produce, and food products to the whole Malaysia retailers. In order to enhance efficiency of the work rate, we will use Java-based purchase management order system with OOP techniques to work on it. This application will automate the Purchase Order process by starting with Sales Manager Purchase Requisition (PR), then go to Purchase Managers approve PR to generating Purchase Order (PO).

Table of Contents

Table of Figure

1.0 Introduction

Since technology is evolving at an alarming rate it is only logical if we move from traditional ways to start implementing technology in our daily lives. With this our company Sigma SDN BHD has implemented a new system to order items from suppliers and sell them to customers. This system has 3 main roles which are Admin which has access to do everything from editing Purchase Orders to removing items from list. Next is Sales Manager the sales manager will make the Purchase requisition which includes of the items that has to be purchased. That PR will be sent to our last user, who is the Purchase Manager. The Purchase manager will then review the Purchase requisition and if approved by him then the Purchase Order will be generated.

2.0 Use Case Diagram

Use Case Diagram

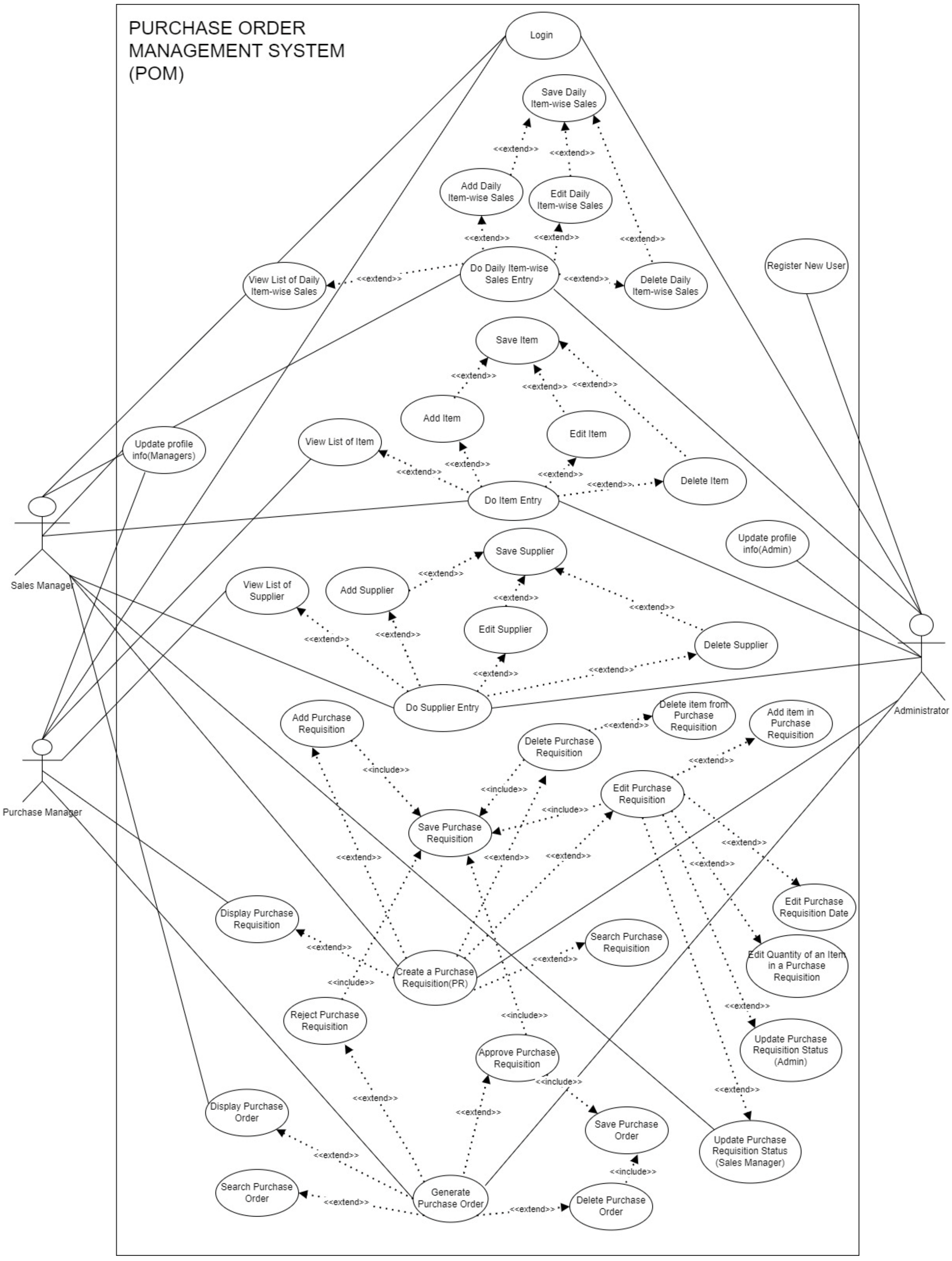

PURCHASE ORDER MANAGEMENT SYSTEM (POM)
Login
Save Daily Item-wise Sales
Add Daily Item-wise Sales
Edit Daily Item-wise Sales
Delete Daily Item-wise Sales
Do Daily Item-wise Sales Entry
View List of Daily Item-wise Sales
Register New User
Save Item
Add Item
Edit Item
Delete Item
Do Item Entry
View List of Item
Update profile info(Managers)
Update profile info(Admin)
Save Supplier
Add Supplier
Edit Supplier
Delete Supplier
View List of Supplier
Do Supplier Entry
Add Purchase Requisition
Delete Purchase Requisition
Delete item from Purchase Requisition
Add item in Purchase Requisition
Edit Purchase Requisition
Save Purchase Requisition
Edit Purchase Requisition Date
Display Purchase Requisition
Search Purchase Requisition
Create a Purchase Requisition(PR)
Edit Quantity of an Item in a Purchase Requisition
Reject Purchase Requisition
Update Purchase Requisition Status (Admin)
Approve Purchase Requisition
Save Purchase Order
Display Purchase Order
Update Purchase Requisition Status (Sales Manager)
Search Purchase Order
Generate Purchase Order
Delete Purchase Order
<<extend>>
<<include>>
Sales Manager
Purchase Manager
Administrator

Figure 1: Case Diagram

Use case specification

Use Case	Login
Brief Description	This allows the user to login into the system.
Actors	Sales Manager, Admin, Purchase Manager
Pre-conditions	The user's credentials must exist in the database.
Post-condition	The user logs into their respective role menus.
Main Flow	1. The user inputs a valid email address. 2. The user inputs a valid password. 3. The user is redirected to the menu of their role.
Alternative Flows	1. If the user inputs the incorrect credentials, they will be prompted to re-enter their credentials again. They are given 3 attempts to do so. 2. If the user fails to login within 3 attempts, they will be redirected back to the main menu. 3. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Register new user
Brief Description	This allows the administrator to register a new user.
Actors	Admin
Pre-conditions	The administrator must be successfully logged in.

Post-condition	The new user is created and their credentials are stored in the text file.
Main Flow	1. The administrator chooses to register a new user. 2. The admin inputs a valid name. 3. The admin inputs a valid password. 4. The admin inputs a valid email address. 5. The admin chooses the user's role. 6. The updated information is saved into the text file.
Alternative Flows	1. If the name is a duplicate name, the admin will need to re-enter the name. 2. If the password does not follow the format, the admin will need to re-enter the name and password. 3. If the name is a duplicate and invalid email, the admin will need to re-enter everything from the start. 4. If system is unsuccessful in saving the updated information, it will say that an error occurred. 5. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Update profile info (Manager)
Brief Description	This allows the user to change their name or password.
Actors	Sales Manager, Purchase Manager
Pre-conditions	The user must be successfully logged in.

Post-condition	The updated information is updated in the text files.
Main Flow	1. The user enters the function to update their personal information. 2. The user chooses whether to update their name or password. 3. If they choose to change their name, they are expected to input a non-duplicate name. 4. If they choose to change their password, they are expected to input a password that follows the format. 5. The updated information is updated in the text file.
Alternative Flows	1. If the user chooses to go back to the menu after entering the function, they can input “3”. 2. If the name is a duplicate name, the user will need to re-enter the name. 3. If the password does not follow the format, the user will need to re-enter the name and password. 4. If system is unsuccessful in saving the updated information, it will say that an error occurred. 5. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Update profile info (Admin)
Brief Description	This allows the administrator to change the user’s role.
Actors	Administrator
Pre-conditions	The administrator must be successfully logged in.
Post-condition	The selected user’s role and ID are changed and the updated information is updated in the text file.
Main Flow	1. The admin enters the function to change the user’s role. 2. The system displays all of the user’s information. 3. The admin enters a valid user ID. 4. The admin selects either “1” to change their role and ID to a Sales Manager, or “2” to change their role and ID to a “Purchase Manager”. 5. The updated information is updated in the text file.
Alternative Flows	1. If the admin enters an invalid user ID, they will be prompted to re-enter a valid user ID. 2. If system is unsuccessful in saving the updated information, it will say that an error occurred. 3. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Do Item Entry
Brief Description	The Item Entry class is used to manage and arrange the item, user can choose to add item, delete item, edit item, view list item or back to their main page.
Actors	Admin, Sales Manager
Pre-conditions	The admin or sales manager should login-in to their account first and choose what function he wants to work on.
Post-condition	The user is navigated to the selected function.
Main Flow	1. First the user chooses Item Entry 2. Then the Item Entry shows different function for user to choose 3. User will be navigated to the selected function
Alternative Flows	-

Use Case	Add Item
Brief Description	It is uses for adding item into the system
Actors	Admin, Sales Manager
Pre-conditions	The user chooses to add item
Post-condition	The item is added into the item database.
Main Flow	1. The user have to key in basic information such as ItemID, item name, quantity, price etc

	2. The system will then confirm whether user want to save the item to the database. 3. If user confirm it will save the item and navigate the user back to Item Entry
Alternative Flows	1. The user not entering exist itemID will be navigate back to the ItemEntry. 2. After adding all the info if user does not confirm to save the action, it will revoke and return to ItemEntry

Use Case	Save Item
Brief Description	It is uses for saving item into the system
Actors	Admin, Sales Manager
Pre-conditions	The user has to finish adding item, deleting item, and editing item and confirm to save the action.
Post-condition	The action that done by the user will be save into the system.
Main Flow	1. The user confirm the to save the action 2. The information is saved to the database
Alternative Flows	1. User does not confirm to save the action, it will revoke and return to ItemEntry

Use Case	Delete item
Brief Description	It is to delete the item from text file.
Actors	Sales Manager, Admin
Pre-conditions	User chooses to delete an item
Post-condition	Item deleted from text file
Main Flow	1. The user first enter the itemID he wants to delete 2. The system will then ask for the confirmation of the deletion 3. If yes, the item is deleted from the text file.
Alternative Flows	1. The user not entering exist itemID will be navigate back to the ItemEntry. 2. After deleting selected item if user does not confirm to save the action, it will revoke and return to ItemEntry

Use Case	Edit Item
Brief Description	It is for editing the item’s description.
Actors	Sales Manager, Admin
Pre-conditions	User chooses to edit the item
Post-condition	The item will be edited
Main Flow	1) The system will display all the item for user to choose and let the user to enter the itemID 2) The system will ask what item’s attribute that user want to change

	3) The attribute will be changed
Alternative Flows	1. The user not entering exist itemID will be navigate back to the ItemEntry. 2. After editing selected item if user does not confirm to save the action, it will revoke and return to ItemEntry

Use Case	View List of Item
Brief Description	This will display all the item in the item text file
Actors	Sales Manager, Admin, Purchase Manager
Pre-conditions	The user chooses to view the item list
Post-condition	Item list is display
Main Flow	1) User chooses to see the item list 2) item list displayed 3) User navigated by to Item Entry
Alternative Flows	-

Use Case	Do Supplier Entry
Brief Description	The Supplier Entry class is used to manage the Suppliers, user can choose to add supplier, delete supplier, edit supplier details, view Supplier list or back to their main page.
Actors	Sales Manager, Admin
Pre-conditions	User needs to login and chose their function.
Post-condition	User is navigated to Selected Function
Main Flow	1. First the user chooses Item Entry 2. Then the Item Entry shows different function for user to choose 3. User will be navigated to the selected function
Alternative Flows	-

Use Case	Add supplier
Brief Description	Add new supplier into text file
Actors	Sales Manager, Admin
Pre-conditions	Decided to add supplier
Post-condition	New supplier is added into text file.
Main Flow	1. The user must add basic information such as Supplier ID, Supplier Name, Supplier contact, Supplier email, Supplier Address.

	2. The system will then confirm whether user want to save the item to the database. 3. If user confirm it will save the item and navigate the user back to Supplier Entry
Alternative Flows	1) The user not entering exist SupplierID will be navigate back to the SupplierEntry. 2) After adding all the info if user does not confirm to save the action, it will revoke and return to Supplier Entry.

Use Case	Save Supplier
Brief Description	Saves the supplier details into the text file
Actors	Sales Manager and Admin
Pre-conditions	User needs to complete all the changes before saving
Post-condition	All details are saved into text file and the system is updated
Main Flow	1) User must choose to save after doing changes. 2) Changes are saved and the text files are updated
Alternative Flows	1) User will choose not to save. 2) Changes will be revoked and not saved into text files.

Use Case	Delete Supplier
Brief Description	This deletes the Supplier from the text file.
Actors	Sales Manager, Admin
Pre-conditions	Must decide to delete Supplier from the Text file
Post-condition	Deleted Supplier will be removed from the Text file
Main Flow	User needs to enter the Supplier ID of the supplier that they want to delete
Alternative Flows	If User enters Supplier ID that is not in the text file, then system will provide an error message.

Use Case	Edit Supplier
Brief Description	Edit Supplier Details
Actors	Sales Manager, Admin
Pre-conditions	Needs to know what the user wants to edit
Post-condition	Edited Supplier details will be entered back into the text file.
Main Flow	1) The user needs to enter the supplier ID of the supplier they are trying to edit. 2) The system will display all the details of the supplier such as: name, number, email, and address. 3) After choosing the attribute they want to change the user needs to enter the new value.

	4) System will display the new list with the updated supplier information. 5) User needs to enter 'Y' to confirm and save the changes.
Alternative Flows	1) If a wrong Supplier ID is entered the system will display an error message and ask the user to renter.

Use Case	View List of Supplier
Brief Description	To see all the registered Supplier
Actors	Sales Manager, Admin, Purchase Manager
Pre-conditions	Wants to view the supplier list
Post-condition	User will know the supplier details.
Main Flow	User needs to choose the option to view the supplier list.
Alternative Flows	1. If system does not display the items list from the text file due to a file reading error, the user must contact the administrator for them to send them the items list as the administrator will have access to the database for it.

Use Case	Do Daily Item-wise Sales Entry
Brief Description	The Daily Item-wise Sales Entry is used to manage the daily sales, user can choose to add Daily Sales, delete Daily Sales, edit Daily Sales details, view Daily Sales list or back to their main page.
Actors	Sales Manager, Admin
Pre-conditions	User needs to login and chose their function.
Post-condition	User is navigated to Selected Function
Main Flow	1. First the user chooses Daily Sales Entry 2. Then the Daily Sales Entry shows different function for user to choose 3. User will be navigated to the selected function
Alternative Flows	-

Use Case	Add Daily Item-wise Sales
Brief Description	Adding daily sales into text file.
Actors	Sales Manager and Admin
Pre-conditions	User needs to know the details of the daily sales that they would like to enter.

Post-condition	Daily item sales are added into the text file.
Main Flow	1) User enters an itemSales ID of their choice but must follow the format which is capital 'IS' followed by 5 digits. 2) Then once the ID is entered the system will display the details of all the available items. 3) The user is required to enter the Item ID of the item that they are adding into the sales and the quantity of the item sold. 4) The system will display the details of that item and the quantity of sales and will ask the user to enter 'Y' to confirm and save. 5) Once confirmed the quantity will be deducted from the stock.
Alternative Flows	1) If invalid Item Sales ID is entered the system will display an error message and ask the user to renter. 2) Id invalid Item ID is entered then the user will be asked to renter the ItemID. 3) If the quantity of the sales of that Item is more than the available stock of that item, then an error message 'NOT ENOUGH STOCKS. TRY AGAIN' will be displayed and the

	user will be asked to reinput the quantity.

Use Case	Save Daily Item-wise Sales
Brief Description	Saves the Daily Item-wise Sales details into the text file
Actors	Sales Manager and Admin
Pre-conditions	User needs to complete all the changes before saving
Post-condition	All details are saved into text file and the system is updated
Main Flow	1) User must choose to save after doing changes. 2) Changes are saved and the text files are updated
Alternative Flows	1) User will choose not to save. 2) Changes will be revoked and not saved into text files.

Use Case	Delete Daily Item-wise Sales
Brief Description	Deletes daily item sales entry from the text file.
Actors	Sales Manager and admin
Pre-conditions	User needs to know what Daily item sales they are going to delete.
Post-condition	Specific Daily item sales is deleted

Main Flow	1) Once user chooses to delete a daily item sale the system will display all the current Item Sales Entries. 2) The user needs to choose the and enter the Item Sales ID of the item sales that needs to be deleted. 3) Then the new item Sales Entry list will be displayed and the quantity of the one that was deleted will be entered back into the quantity of the item. 4) User needs to enter ‘Y’ and confirm the changes.
Alternative Flows	1) If user enters a wrong ID, the system will reject it and ask to renter. 2) User can choose to exit the system instead of deleting by typing ‘exit’

Use Case	Edit Daily Item-wise Sales
Brief Description	Edits information about the daily Item sales entry
Actors	Sales Manager and Admin
Pre-conditions	User needs to know what information they going to change.
Post-condition	New information about the sales entry is updated.
Main Flow	1) Once the user chooses to edit the daily Item sales entry, the system will

	display all the details about the existing Item Sales entry and their corresponding ID's. 2) Then the user must enter the ID of the Sales entry they are trying to change. 3) Once they enter the ID the system will display the attributes that the user can change along with the current details. 4) The Attribute that can be changes are Date and quantity. 5) After the user chooses the attribute, they will have to enter the new value within the correct conditions and format. 6) Lastly the user must enter 'Y' to save the changes
Alternative Flows	1) If invalid ID is entered the system will reject and ask to renter. 2) If user enters a number that doesn't correspond with the attribute the system will reject and provide an error message "INVALID INPUT" and ask the user to renter. 3) If the new value entered by the user is not within the conditions, then the system will reject.

Use Case	View List of Daily Item-wise Sales
Brief Description	See all the existing Daily Item-Wise sales entry
Actors	Sales Manager, Admin
Pre-conditions	Want to see the list of item sales
Post-condition	Can view all the display item sales
Main Flow	1) If user chooses the option to view the list in the menu than the entire list is displayed.
Alternative Flows	1. If system does not display the items list from the text file due to a file reading error, the user must contact the administrator for them to send them the items list as the administrator will have access to the database for it.

Use Case	Create a Purchase Requisition
Brief Description	This function is for the user to create a purchase requisition.
Actors	Sales Manager, Admin
Pre-conditions	User must be logged in and must know which items to be added into the purchase requisition

Post-condition	Purchase Requisition is created, and information is stored in text files
Main Flow	1. User will enter the create purchase requisition function from the Sales Manager menu. 2. System will display the items list to the user. 3. The user will enter the item ID and the quantity to be requested for that item. 4. When user has nothing else to add, they will input “-1”. 5. User will enter a valid date. 6. Purchase requisition information will be saved into text files.
Alternative Flows	1. If system does not display the items list from the text file due to a file reading error, the user must contact the administrator for them to send them the items list as the administrator will have access to the database for it. 2. If the user enters an item ID which does not exist in the items database, the system will say that it is a non-existent item ID and the user will be prompted to re-enter a valid item ID. 3. If the user enters and item ID which has already been added into the same purchase requisition, the system will say that a duplicate item in the same purchase requisition is not allowed

	and the user will be prompted to re-enter a different item ID. 4. If the user adds an invalid quantity such as leaving it blank or entering anything but a number, the system will say that this is not allowed and the user is required to re-enter a valid quantity. 5. If the user enters a data that is invalid, the system will say that the date is invalid and the user will be prompted to re-enter a valid date. 6. If the system fails to save the information into the purchase requisition text files, it will return an error.

Use Case	Add item in purchase Requisition
Brief Description	This lets the user add an item into an existing purchase requisition
Actors	Sales Manager, Admin
Pre-conditions	User must be logged in and must know which items to be added into the purchase requisition
Post-condition	Purchase requisition with updated information is updated in text file and selected item stock also decreases in text file
Main Flow	1. User will enter the update purchase requisition menu.

	2. User will enter the purchase requisition ID that is to be updated. 3. User will select the option to add an item into the purchase requisition. 4. User will enter a valid item ID to be added. 5. User will enter a valid quantity to be added. 6. Purchase requisition information will be updated in the text files.
Alternative Flows	1. If the user does not want to edit the purchase requisition, they can input “6” to go back to the Sales Manager menu. 2. If the user enters an invalid purchase requisition ID, they will be prompted to re-enter a valid ID. 3. If the user enters an invalid item ID, they will be prompted to re-enter a valid ID. 4. If the user enters an invalid quantity for the item, they will be prompted to re-enter a valid quantity. 5. If system is unsuccessful in saving the updated information, it will say that an error occurred. 6. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Save Purchase Requisition

Brief Description	The system will save the purchase requisition into text files.
Actors	Sales Manager, Admin
Pre-conditions	The user must be successfully logged in and there must be something to save/update in the text files.
Post-condition	The updated information is successfully saved in the text files.
Main Flow	1. The system gets the updated information. 2. The system writes the updated information into the text files.
Alternative Flows	1. If system is unsuccessful in saving the updated information, it will say that an error occurred. 2. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Delete Purchase Requisition
Brief Description	This allows the user to delete the whole purchase requisition.
Actors	Sales Manager, Admin

Pre-conditions	User must be successfully logged in and must know which purchase requisition that they would like to delete.
Post-condition	Purchase requisition is successfully deleted and the updated information is updated in the text files.
Main Flow	1. The user will input the option from the Sales Manager menu to delete a PR. 2. The user selects that they would like to delete the whole PR. 3. The user enters "Y" when the system prompts them on their confirmation for their choice. 4. The PR is successfully deleted. 5. The system writes the updated information into the text files.
Alternative Flows	1. If the user does not want to delete the PR when asked how they would like to delete the PR (either by item or the full PR), the user will enter "3" to go back to the Sales Manager menu. 2. If the user does not want to delete the PR when asked for their confirmation, they will input anything but "Y". 3. If the user does not enter a valid PR ID, they will be prompted to re-enter a valid PR ID. 4. If system is unsuccessful in saving the updated information, it will say that an error occurred.

	5. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Delete Item from Purchase Requisition
Brief Description	This allows the user to delete a specific item from a PR.
Actors	Sales Manager, Admin
Pre-conditions	The user must be successfully logged in.
Post-condition	The item is successfully deleted from the PR and the updated information is updated in the text files.
Main Flow	1. The user will input the option from the Sales Manager menu to delete a PR. 2. The user selects that they would like to delete an item from the PR. 3. The user enters a valid PR ID. 4. The system displays the PR and its items. 5. The user enters a valid item ID that exists in the PR and in the item database. 6. The item is successfully deleted from the PR. 7. The system writes the updated information into the text files.
Alternative Flows	1. If the user does not want to delete the PR when asked how they would like

	to delete the PR (either by item or the full PR), the user will enter "3" to go back to the Sales Manager menu. 2. If the user does not enter a valid item ID, they will be prompted to re-enter a valid item ID. 3. If the user does not enter a valid PR ID, they will be prompted to re-enter a valid PR ID. 4. If system is unsuccessful in saving the updated information, it will say that an error occurred. 5. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Edit Purchase Requisition
Brief Description	This allows the user to edit the purchase requisition.
Actors	Sales Manager, Admin
Pre-conditions	The user must be successfully logged in.
Post-condition	The user enters the function on how they would like to edit in the menu.
Main Flow	1. The user enters "6" from the Sales Manager menu to go into this menu. 2. The system displays the PRs to the user. 3. The user inputs a valid PR ID to edit.

	4. The user inputs a valid choice when asked what they would like to edit in the PR.
Alternative Flows	1. If the user enters the edit menu and they want to go back, they can input "6" to go back to the Sales Manager menu. 2. If the user enters an invalid PR ID, they will be prompted to re-enter a valid PR ID. 3. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Edit Purchase Requisition Date
Brief Description	This allows the user to edit the date of a purchase requisition.
Actors	Sales Manager, Admin
Pre-conditions	The user must be successfully logged in.
Post-condition	The updated information is successfully updated in the text files.
Main Flow	1. The user enters "6" from the Sales Manager menu to go into this menu. 2. The system displays the PRs to the user. 3. The user inputs a valid PR ID to edit. 4. The user inputs that they would like to edit the date of said PR. 5. The user enters a valid date.

	6. The updated information is successfully updated in the text files.
Alternative Flows	1. If the user enters the edit menu and they want to go back, they can input "6" to go back to the Sales Manager menu. 2. If the user enters an invalid PR ID, they will be prompted to re-enter a valid PR ID. 3. If the user enters an invalid date, they will be prompted to re-enter a valid date. 4. If the system is unsuccessful in reading the files, it will say an error occurred. 5. If the system is unsuccessful in updating the files, it will say an error occurred.

Use Case	Edit Quantity of an Item in a Purchase Requisition
Brief Description	This allows the user to edit the quantity of an item in a purchase requisition.
Actors	Sales Manager, Admin
Pre-conditions	The user must be successfully logged in.
Post-condition	The updated information is successfully updated in the text files.
Main Flow	1. The user enters "6" from the Sales Manager menu to go into this menu.

	2. The system displays the PRs to the user. 3. The user inputs a valid PR ID to edit. 4. The user inputs that they would like to edit the quantity of the PR. 5. The user enters a valid item ID that exists in both the PR and in the items database. 6. The user enters a valid quantity. 7. The updated information is successfully updated in the text files.
Alternative Flows	1. If the user enters the edit menu and they want to go back, they can input “6” to go back to the Sales Manager menu. 2. If the user enters an invalid PR ID, they will be prompted to re-enter a valid PR ID. 3. If the user enters an invalid item ID, they will be prompted to re-enter a valid item ID. 4. If the user enters an invalid quantity, they will be prompted to re-enter a valid quantity. 5. If the system is unsuccessful in reading the files, it will say an error occurred. 6. If the system is unsuccessful in updating the files, it will say an error occurred.

Use Case	Display Purchase Requisition
Brief Description	This allows the user to view the list of purchase requisitions.
Actors	Sales Manager, Admin, Purchase Manager
Pre-conditions	The user must be successfully logged in.
Post-condition	The correct list is successfully shown to the user.
Main Flow	1. The user enters the function to view the PR, which is "5" in the Sales Manager menu and "3" in the Purchase Manager menu respectively. 2. The system reads the files and displays the correct list to the user.
Alternative Flows	1. If the system is unsuccessful in reading the file, it will say an error occurred.

Use Case	Search a Purchase Requisition
Brief Description	This allows the user to search a specific purchase requisition.
Actors	Sales Manager, Admin
Pre-conditions	The user successfully logs in and knows which PR they would like to search for.
Post-condition	The system successfully finds the PR and displays it to the user.
Main Flow	1. The user enters the function to search for a PR. 2. The user enters a valid PR ID. 3. The system displays the PR to the user.

Alternative Flows	1. If the user enters an invalid PR ID, they will be prompted to re-enter the PR ID. 2. If the system is unsuccessful in reading the file, it will say an error occurred.

Use Case	Update Purchase Requisition Status (Sales Manager)
Brief Description	This allows the Sales Manager to cancel the PR.
Actors	Sales Manager
Pre-conditions	The user must be successfully logged in.
Post-condition	The PR is cancelled.
Main Flow	1. The user enters the edit function. 2. The user enters a valid PR ID. 3. The user enters the function to cancel the PR status. 4. The updated information is successfully updated in the text file.
Alternative Flows	1. If the user enters an invalid PR ID, they will be prompted to re-enter the PR ID. 2. If the user wants to go back from the edit menu, they can input “6” to go back. 3. If the PR status is either “APPROVED” or “REJECTED”, they cannot cancel the PR.

	4. If the system is unsuccessful in reading the files, it will say an error occurred. 5. If the system is unsuccessful in updating the files, it will say an error occurred.

Use Case	Update Purchase Requisition Status (Admin)
Brief Description	This allows the admin to change the status of the PR.
Actors	Admin
Pre-conditions	User must be successfully logged in.
Post-condition	The PR status has been changed accordingly.
Main Flow	1. The admin enters a valid PR ID. 2. The admin enters the function to edit a PR status. 3. The admin chooses which status that they would like to change it to, depending on the current status of the PR. 4. The status of the PR is successfully changed and the updated information is updated in the text files.
Alternative Flows	1. If the admin enters an invalid PR ID, they will be prompted to re-enter a valid PR ID. 2. If the system is unsuccessful in reading the files, it will say an error occurred.

	3. If the system is unsuccessful in updating the files, it will say an error occurred.

Use Case	Generate Purchase Order
Brief Description	This will allow the user to approve a PR and generate a PO from it.
Actors	Purchase Manager, Admin
Pre-conditions	User must be successfully logged in.
Post-condition	PO(s) successfully generated from PR.
Main Flow	1. User will enter the function to approve a PR. 2. User will enter a valid PR ID. 3. PR is approved and converted to PO based on suppliers. 4. Updated information is updated in text files.
Alternative Flows	1. If the user enters an invalid PR ID, they will be prompted to re-enter a valid PR ID again. 2. If the user wishes to not approve the PR, but they have entered the function, they can type “exit” to go back to the PM Menu. 3. If the date of the PR is invalid (less than 1 week from the current date or more than 1 year from the current date), the system will deny the approval.

	4. If the status of the PR is "CANCELLED", the system will deny the approval. 5. If the system is unsuccessful in reading the files, it will say an error occurred. 6. If the system is unsuccessful in updating the files, it will say an error occurred.

Use Case	Save Purchase Order
Brief Description	This function allows the system to save the PO in text files.
Actors	Purchase Manager, Admin
Pre-conditions	There must be either a new PO or something to be changed inside a PO that will be added/updated into the text files.
Post-condition	PO is successfully added/updated in text files.
Main Flow	1. System gets the updated information after user adds/updates a PO. 2. System adds the updated information into a text file.
Alternative Flows	1. If system is unsuccessful in saving the updated information, it will say that an error occurred.

	2. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Delete Purchase Order
Brief Description	This allows the user to delete a PO.
Actors	Purchase Manager, Admin
Pre-conditions	User must be successfully logged in.
Post-condition	PO is successfully deleted and updated information is updated in text files.
Main Flow	1. User enters the delete PO function from the PM Menu. 2. User enters a valid PO ID. 3. User enters “Y” as confirmation in deleting the PO when prompted by the system to confirm choice. 4. Selected PO is deleted. 5. Updated information is updated in text files.
Alternative Flows	1. If the user enters an invalid PO ID, they will be prompted to re-enter a valid PO ID. 2. If the user does not want to delete the PO ID, they can enter anything but “Y” to exit the function. 3. If system is unsuccessful in saving the updated information, it will say that an error occurred.

	4. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Reject Purchase Requisition
Brief Description	This allows the Purchase Manager to reject a PR.
Actors	Purchase Manager, Admin
Pre-conditions	User must be successfully logged in.
Post-condition	PR status is updated to “REJECTED” and updated information is updated in text files.
Main Flow	1. User enters the function to reject a PR. 2. System displays all PR information. 3. User enters a valid PR ID to be rejected. 4. PR status is updated to be “REJECTED”. 5. Updated information is updated in text files.
Alternative Flows	1. If the user enters an invalid PR ID, they will be prompted to re-enter a valid PR ID. 2. If the current status of the PR is “CANCELLED”, the system will deny the rejection.

	5. If system is unsuccessful in saving the updated information, it will say that an error occurred. 3. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Display Purchase Order
Brief Description	This allows the user to view the list of POs.
Actors	Sales Manager, Admin, Purchase Manager
Pre-conditions	User must be successfully logged in.
Post-condition	System successfully lists the POs to the user.
Main Flow	1. User enters the function to view the list of POs. 2. System reads files and prints list of PO to the user.
Alternative Flows	1. If the system is unsuccessful in reading the files, it will say an error occurred.

Use Case	Search Purchase Order
Brief Description	This function allows the user to search for a specific PO.
Actors	Purchase Manager, Admin
Pre-conditions	User must be successfully logged in.
Post-condition	User is able to view the selected PO.

Main Flow	1. User enters the function to search for a PO. 2. User enters a valid PO ID. 3. User is able to view the selected PO and its information.
Alternative Flows	1. If the user enters an invalid PO ID, they will be prompted to re-enter a valid PO ID. 2. If the system is unsuccessful in reading the files, it will say an error occurred.

3.0 Class Diagram

Class Diagram with only class

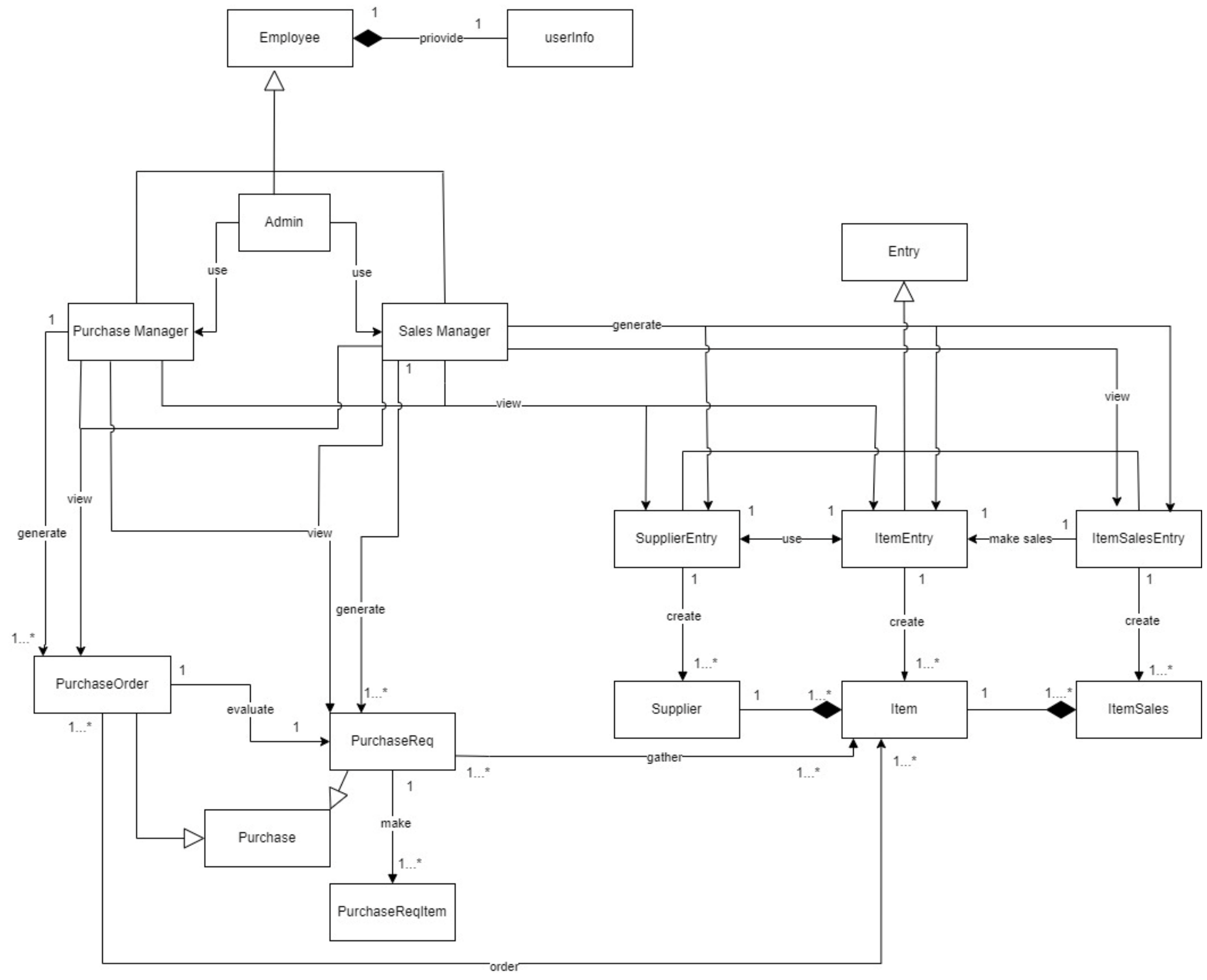


Figure 2: Class Diagram with only class

Full diagram of class diagram

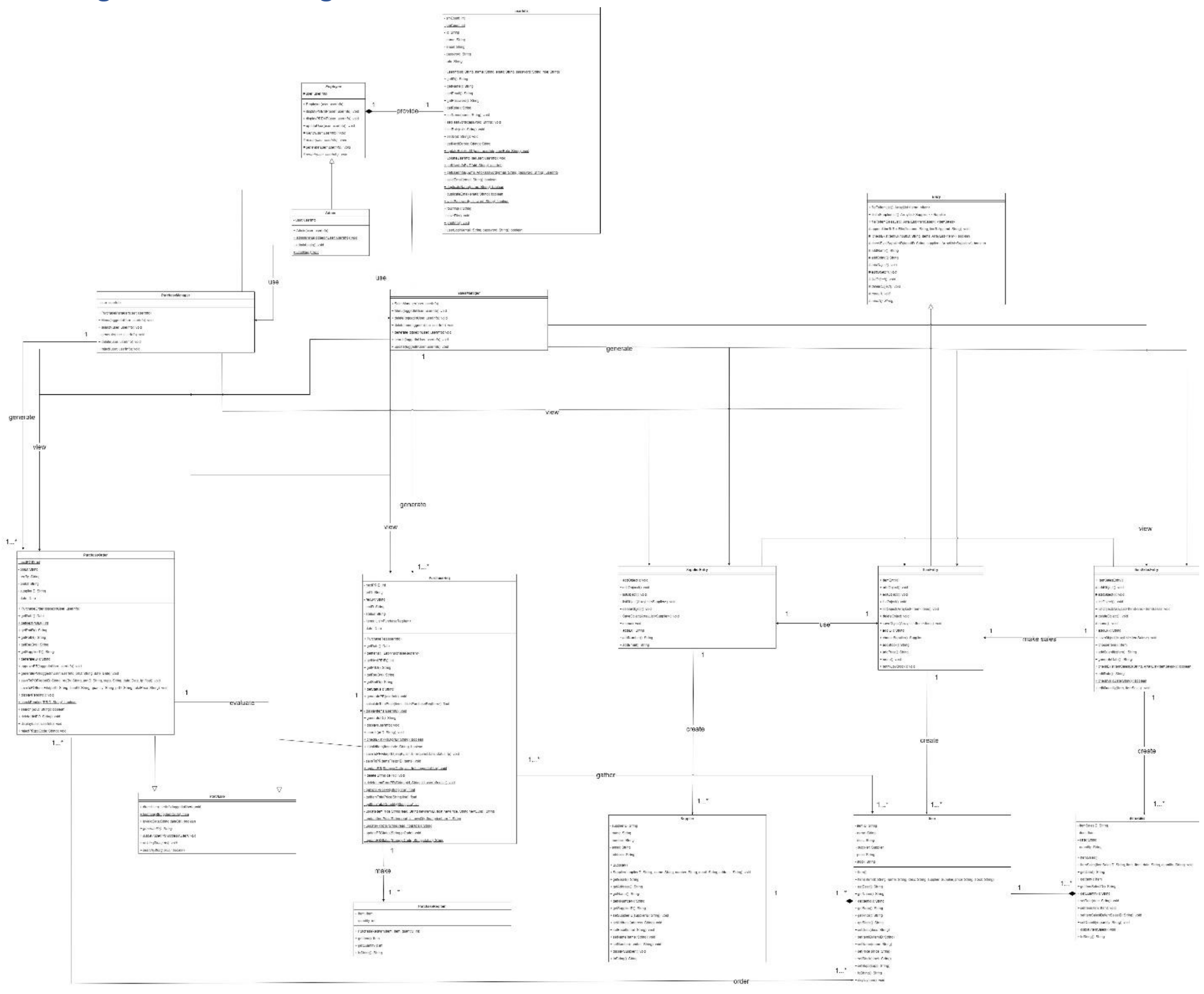
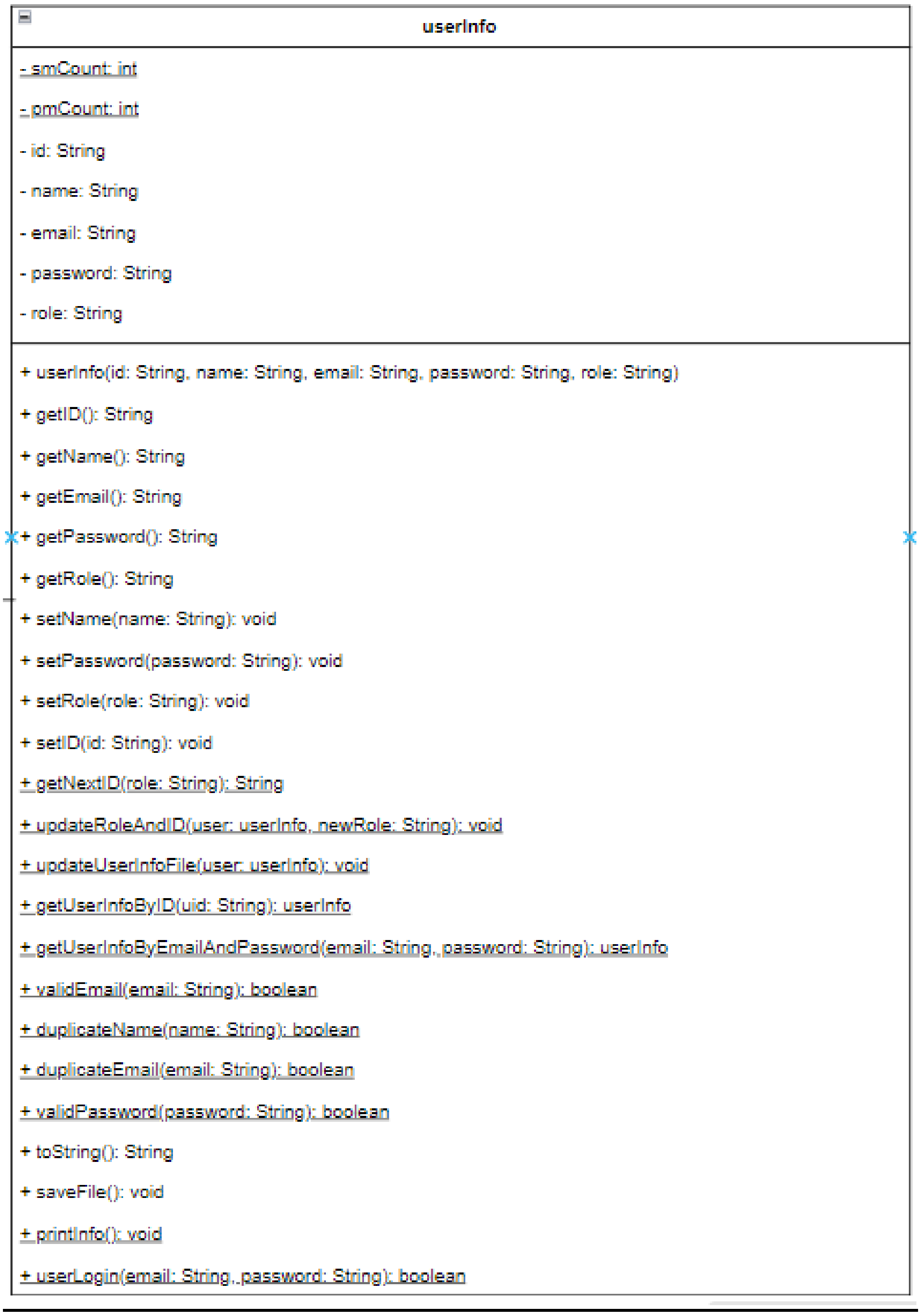


Figure 3: Full class diagram

Table of class diagram



userInfo

- smCount: int
- pmCount: int
- id: String
- name: String
- email: String
- password: String
- role: String

+ userInfo(id: String, name: String, email: String, password: String, role: String)
+ getID(): String
+ getName(): String
+ getEmail(): String
+ getPassword(): String
+ getRole(): String
+ setName(name: String): void
+ setPassword(password: String): void
+ setRole(role: String): void
+ setID(id: String): void
+ getNextID(role: String): String
+ updateRoleAndID(user: userInfo, newRole: String): void
+ updateUserInfoFile(user: userInfo): void
+ getUserInfoByID(uid: String): userInfo
+ getUserInfoByEmailAndPassword(email: String, password: String): userInfo
+ validEmail(email: String): boolean
+ duplicateName(name: String): boolean
+ duplicateEmail(email: String): boolean
+ validPassword(password: String): boolean
+ toString(): String
+ saveFile(): void
+ printInfo(): void
+ userLogin(email: String, password: String): boolean

Figure 4: userInfo class

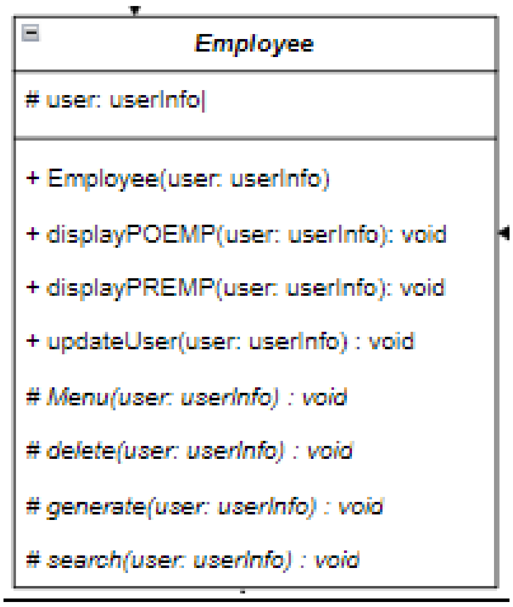

Employee
# user: userInfo
+ Employee(user: userInfo)
+ displayPOEMP(user: userInfo): void
+ displayPREMP(user: userInfo): void
+ updateUser(user: userInfo) : void
# Menu(user: userInfo) : void
# delete(user: userInfo) : void
# generate(user: userInfo) : void
# search(user: userInfo) : void

Figure 5: Employee class

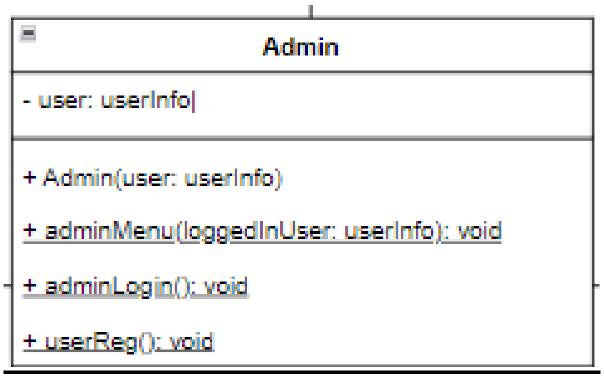


Figure 6: Admin class

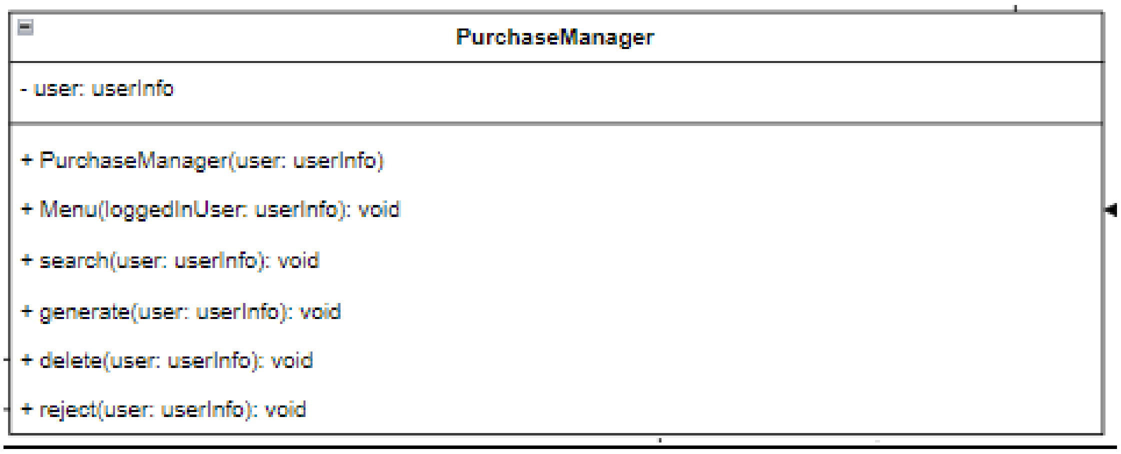
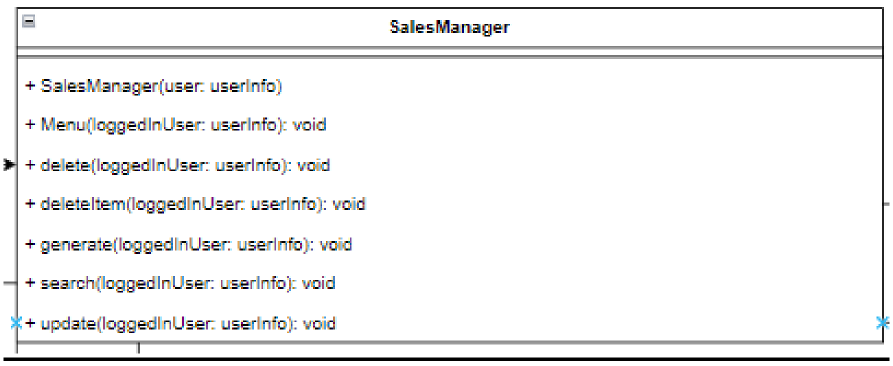

PurchaseManager
- user: userInfo
+ PurchaseManager(user: userInfo)
+ Menu(loggedInUser: userInfo): void
+ search(user: userInfo): void
+ generate(user: userInfo): void
+ delete(user: userInfo): void
+ reject(user: userInfo): void

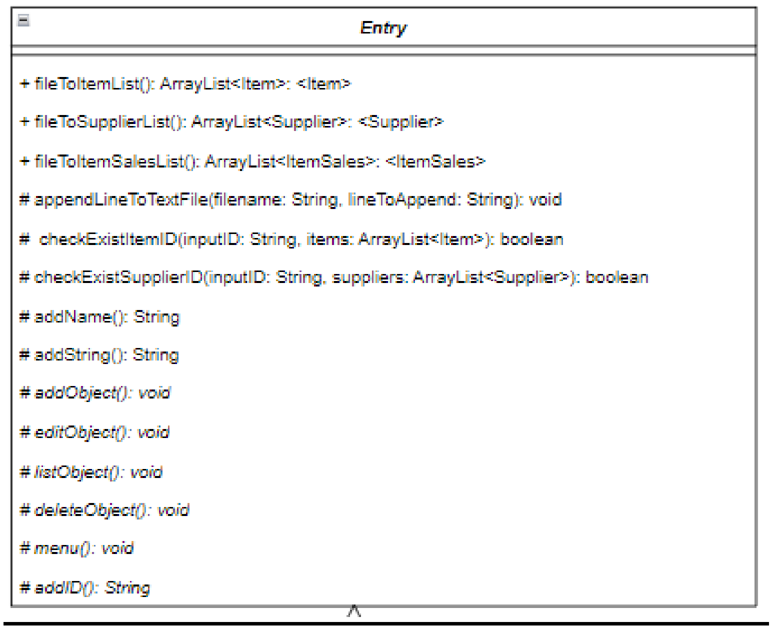
Figure 7: Purchase Manager class



SalesManager

+ SalesManager(user: userInfo)
+ Menu(loggedInUser: userInfo): void
+ delete(loggedInUser: userInfo): void
+ deleteItem(loggedInUser: userInfo): void
+ generate(loggedInUser: userInfo): void
+ search(loggedInUser: userInfo): void
+ update(loggedInUser: userInfo): void

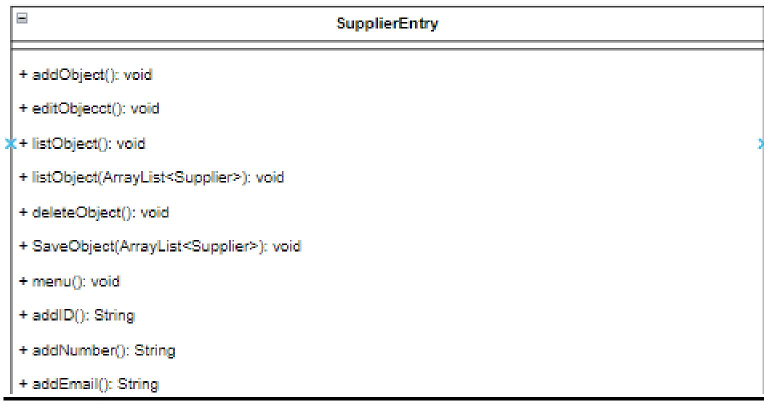
Figure 8: Sales Manager class



Entry

+ fileToItemList(): ArrayList<Item>: <Item>
+ fileToSupplierList(): ArrayList<Supplier>: <Supplier>
+ fileToItemSalesList(): ArrayList<ItemSales>: <ItemSales>
# appendLineToTextFile(filename: String, lineToAppend: String): void
# checkExistItemID(inputID: String, items: ArrayList<Item>): boolean
# checkExistSupplierID(inputID: String, suppliers: ArrayList<Supplier>): boolean
# addName(): String
# addString(): String
addObject(): void
editObject(): void
listObject(): void
deleteObject(): void
menu(): void
addID(): String

Figure 9: Entry class



SupplierEntry

+ addObject(): void
+ editObjecct(): void
+ listObject(): void
+ listObject(ArrayList<Supplier>): void
+ deleteObject(): void
+ SaveObject(ArrayList<Supplier>): void
+ menu(): void
+ addID(): String
+ addNumber(): String
+ addEmail(): String

Figure 10: Supplier Enrty class

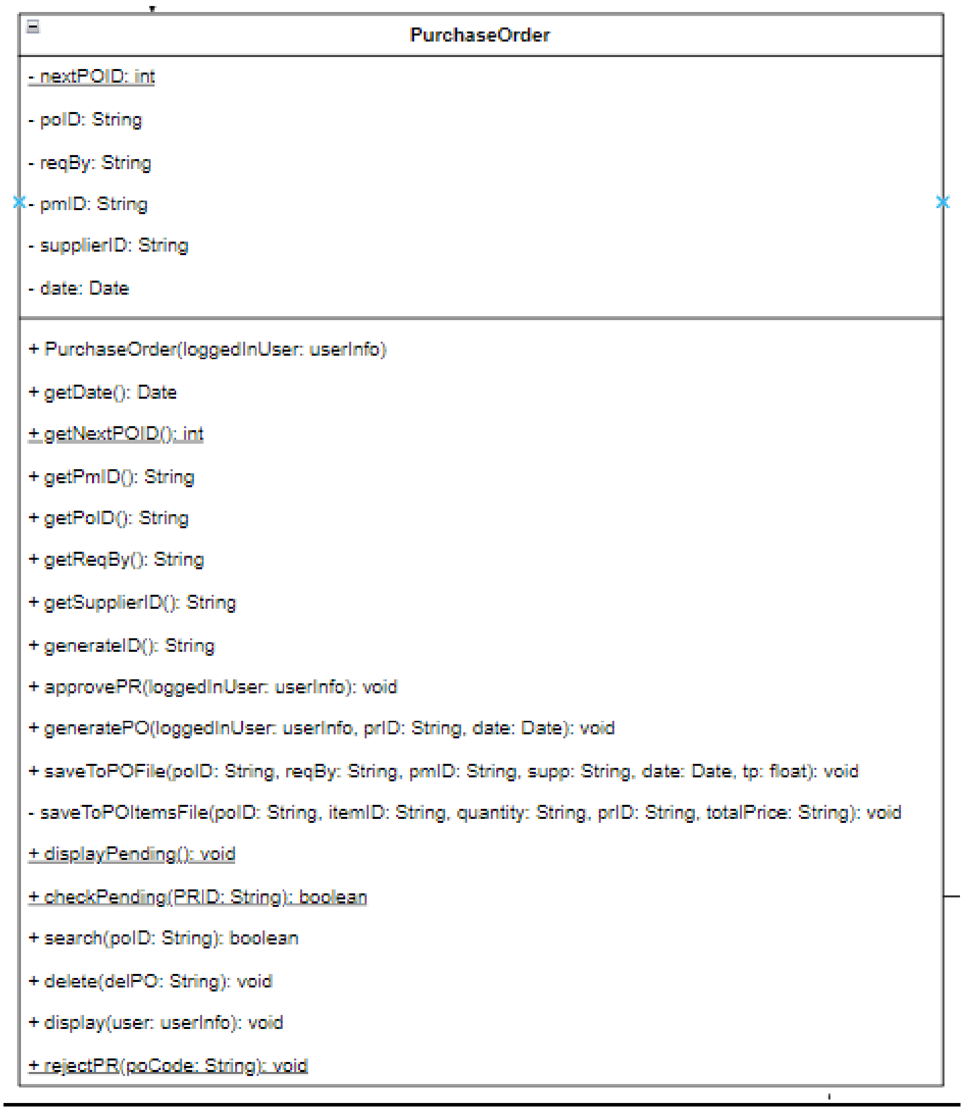

PurchaseOrder
- nextPOID: int
- poID: String
- reqBy: String
- pmID: String
- supplierID: String
- date: Date
+ PurchaseOrder(loggedInUser: userInfo)
+ getDate(): Date
+ getNextPOID(): int
+ getPmID(): String
+ getPoID(): String
+ getReqBy(): String
+ getSupplierID(): String
+ generateID(): String
+ approvePR(loggedInUser: userInfo): void
+ generatePO(loggedInUser: userInfo, prID: String, date: Date): void
+ saveToPOFile(poID: String, reqBy: String, pmID: String, supp: String, date: Date, tp: float): void
- saveToPOItemsFile(poID: String, itemID: String, quantity: String, prID: String, totalPrice: String): void
+ displayPending(): void
+ checkPending(PRID: String): boolean
+ search(poID: String): boolean
+ delete(delPO: String): void
+ display(user: userInfo): void
+ rejectPR(poCode: String): void

Figure 11: Purchase Order class

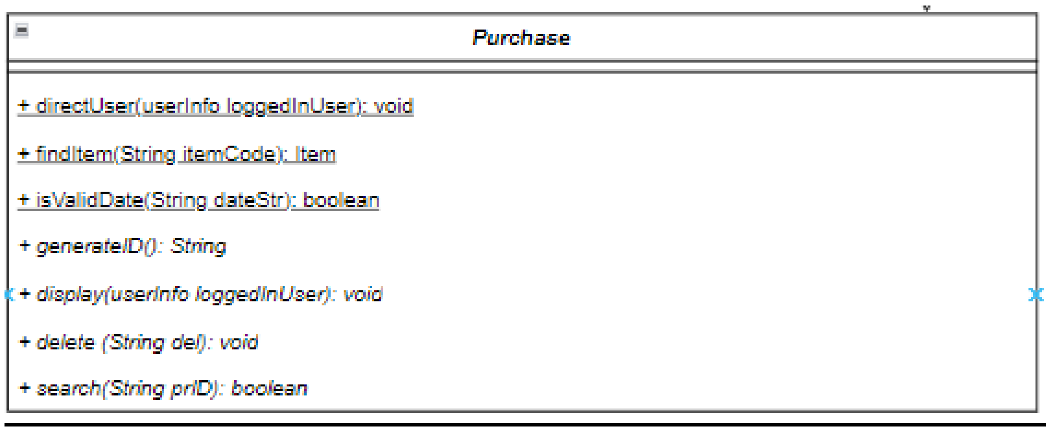
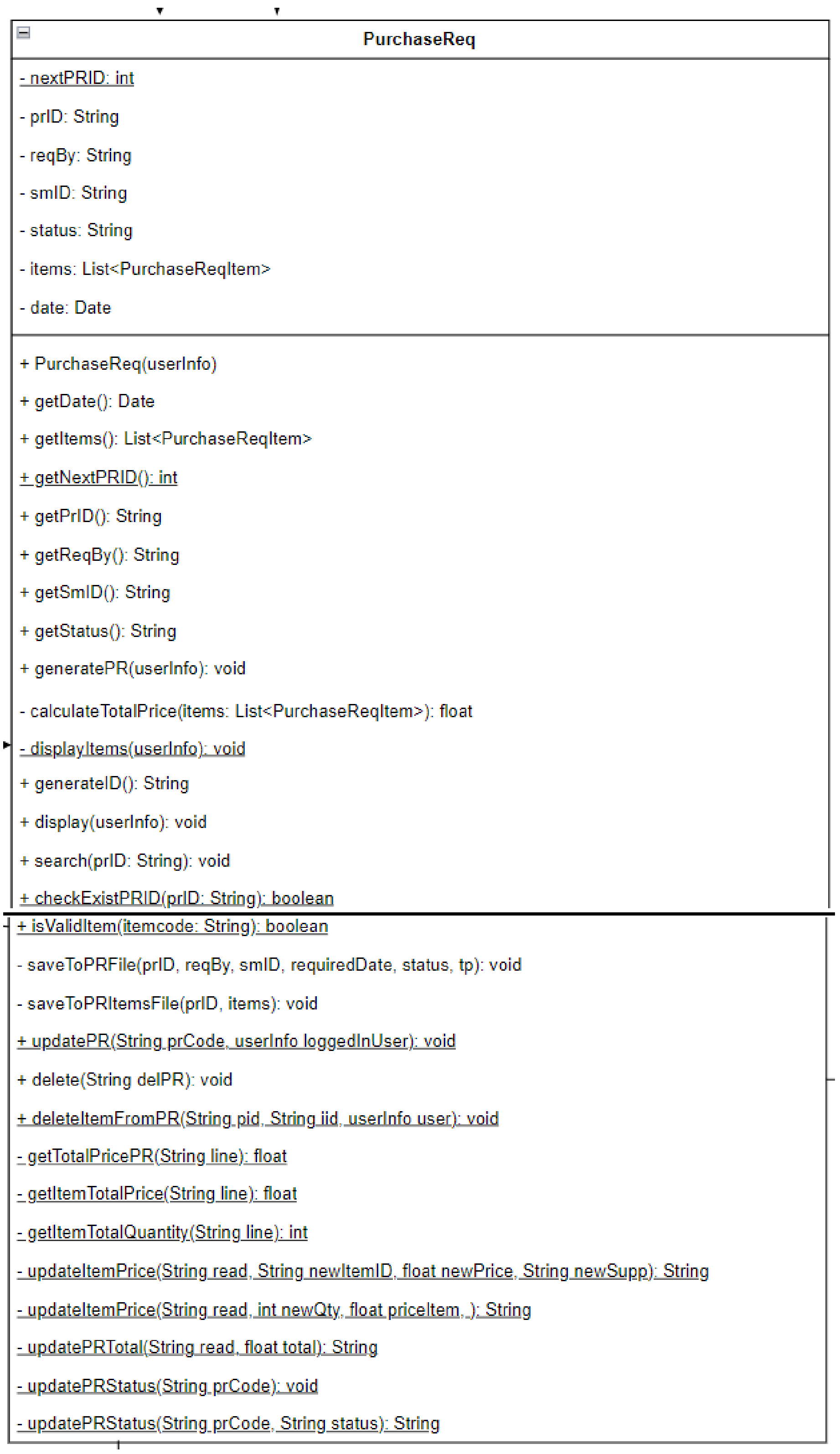

Purchase
+ directUser(userInfo loggedInUser): void
+ findItem(String itemCode): Item
+ isValidDate(String dateStr): boolean
+ generateID(): String
+ display(userInfo loggedInUser): void
+ delete (String del): void
+ search(String prID): boolean

Figure 12: Purchase class



PurchaseReq

- nextPRID: int
- prID: String
- reqBy: String
- smID: String
- status: String
- items: List<PurchaseReqItem>
- date: Date

+ PurchaseReq(userInfo)
+ getDate(): Date
+ getItems(): List<PurchaseReqItem>
+ getNextPRID(): int
+ getPrID(): String
+ getReqBy(): String
+ getSmID(): String
+ getStatus(): String
+ generatePR(userInfo): void
- calculateTotalPrice(items: List<PurchaseReqItem>): float
- displayItems(userInfo): void
+ generateID(): String
+ display(userInfo): void
+ search(prID: String): void
+ checkExistPRID(prID: String): boolean
+ isValidItem(itemcode: String): boolean
- saveToPRFile(prID, reqBy, smID, requiredDate, status, tp): void
- saveToPRItemsFile(prID, items): void
+ updatePR(String prCode, userInfo loggedInUser): void
+ delete(String delPR): void
+ deleteItemFromPR(String pid, String iid, userInfo user): void
- getTotalPricePR(String line): float
- getItemTotalPrice(String line): float
- getItemTotalQuantity(String line): int
- updateItemPrice(String read, String newItemID, float newPrice, String newSupp): String
- updateItemPrice(String read, int newQty, float priceItem,): String
- updatePRTotal(String read, float total): String
- updatePRStatus(String prCode): void
- updatePRStatus(String prCode, String status): String

Figure 13: Purchase Requisition class

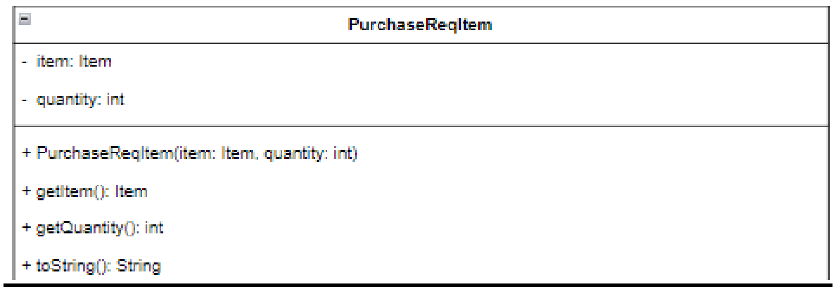


Figure 14: Purchase Requisition Item class

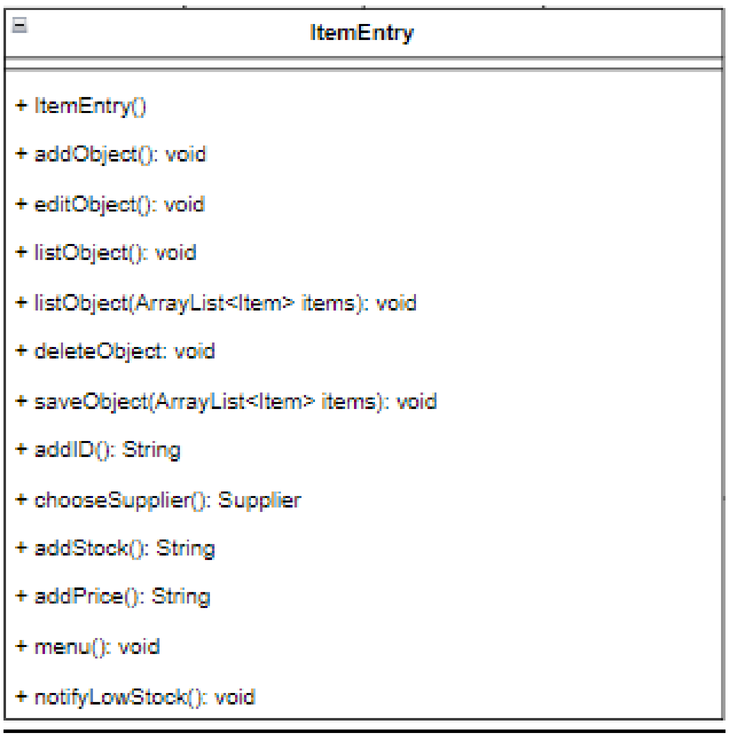
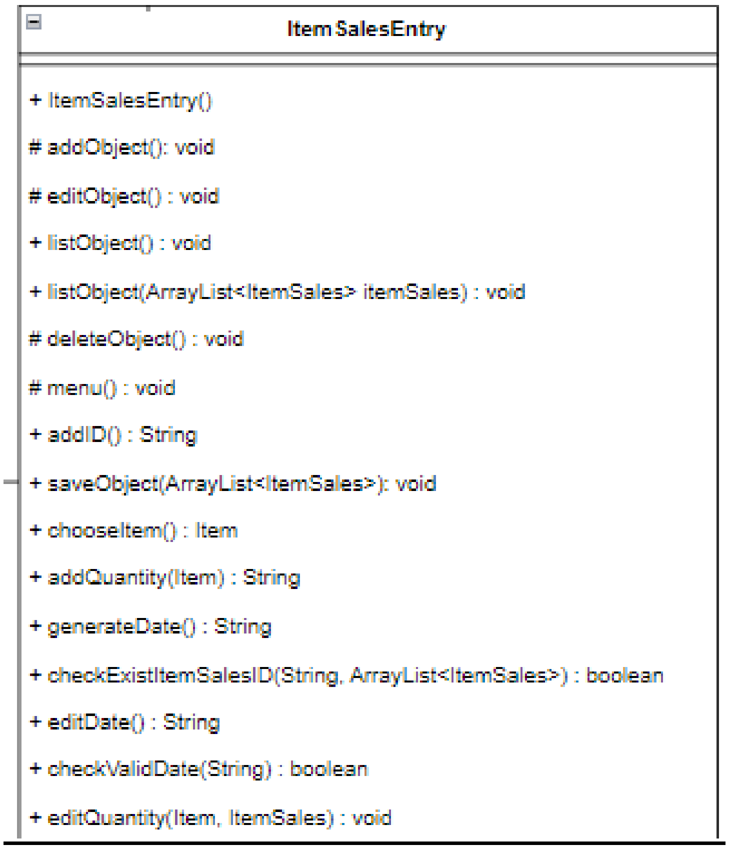


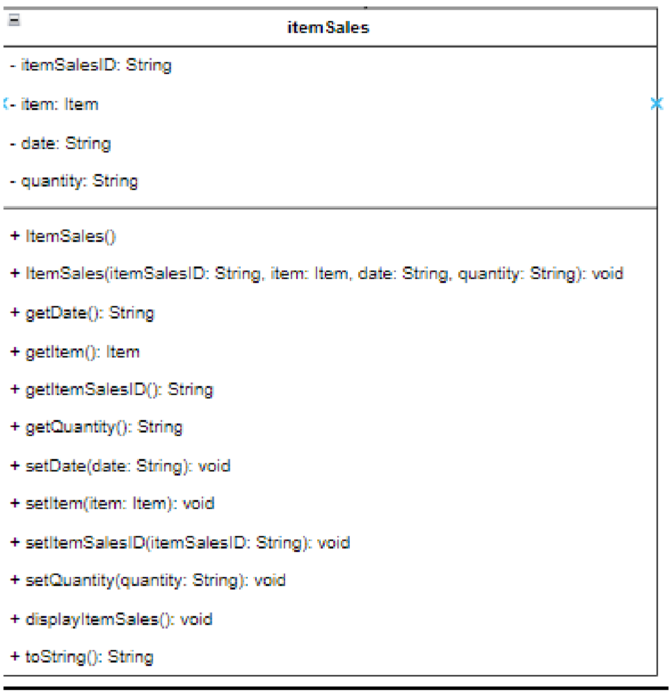
Figure 15: Item Entry class



ItemSalesEntry

+ ItemSalesEntry()
# addObject(): void
# editObject() : void
+ listObject() : void
+ listObject(ArrayList<ItemSales> itemSales) : void
# deleteObject() : void
# menu() : void
+ addID() : String
+ saveObject(ArrayList<ItemSales>): void
+ chooseItem() : Item
+ addQuantity(Item) : String
+ generateDate() : String
+ checkExistItemSalesID(String, ArrayList<ItemSales>) : boolean
+ editDate() : String
+ checkValidDate(String) : boolean
+ editQuantity(Item, ItemSales) : void

Figure 16: Item Sales Emtry class

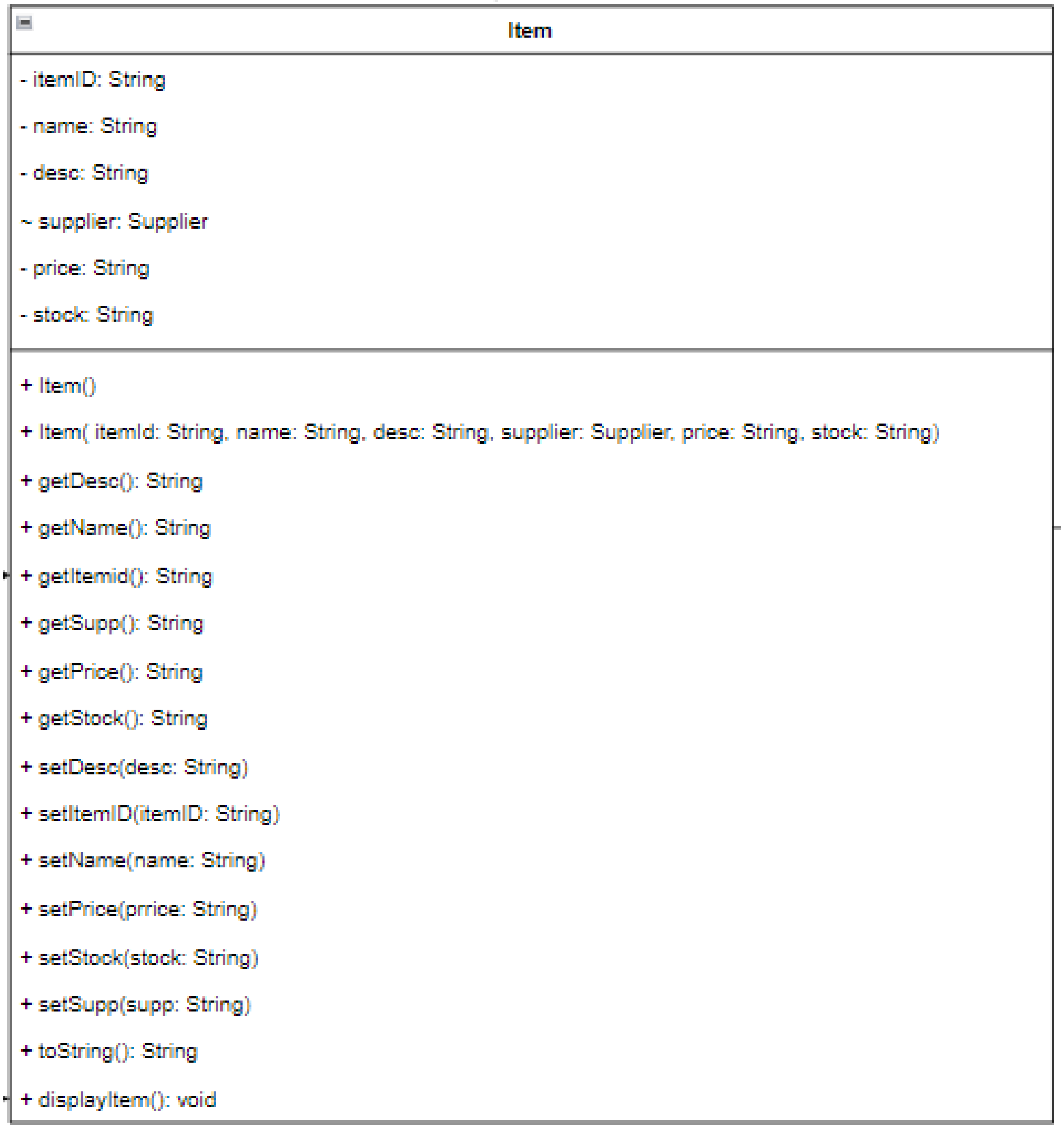


itemSales

- itemSalesID: String
- item: Item
- date: String
- quantity: String

+ ItemSales()
+ ItemSales(itemSalesID: String, item: Item, date: String, quantity: String): void
+ getDate(): String
+ getItem(): Item
+ getItemSalesID(): String
+ getQuantity(): String
+ setDate(date: String): void
+ setItem(item: Item): void
+ setItemSalesID(itemSalesID: String): void
+ setQuantity(quantity: String): void
+ displayItemSales(): void
+ toString(): String

Figure 17: Item Sales class



Item

- itemID: String
- name: String
- desc: String
~ supplier: Supplier
- price: String
- stock: String

+ Item()
+ Item(itemId: String, name: String, desc: String, supplier: Supplier, price: String, stock: String)
+ getDesc(): String
+ getName(): String
+ getItemid(): String
+ getSupp(): String
+ getPrice(): String
+ getStock(): String
+ setDesc(desc: String)
+ setItemID(itemID: String)
+ setName(name: String)
+ setPrice(prrice: String)
+ setStock(stock: String)
+ setSupp(supp: String)
+ toString(): String
+ displayItem(): void

Figure 18: item class

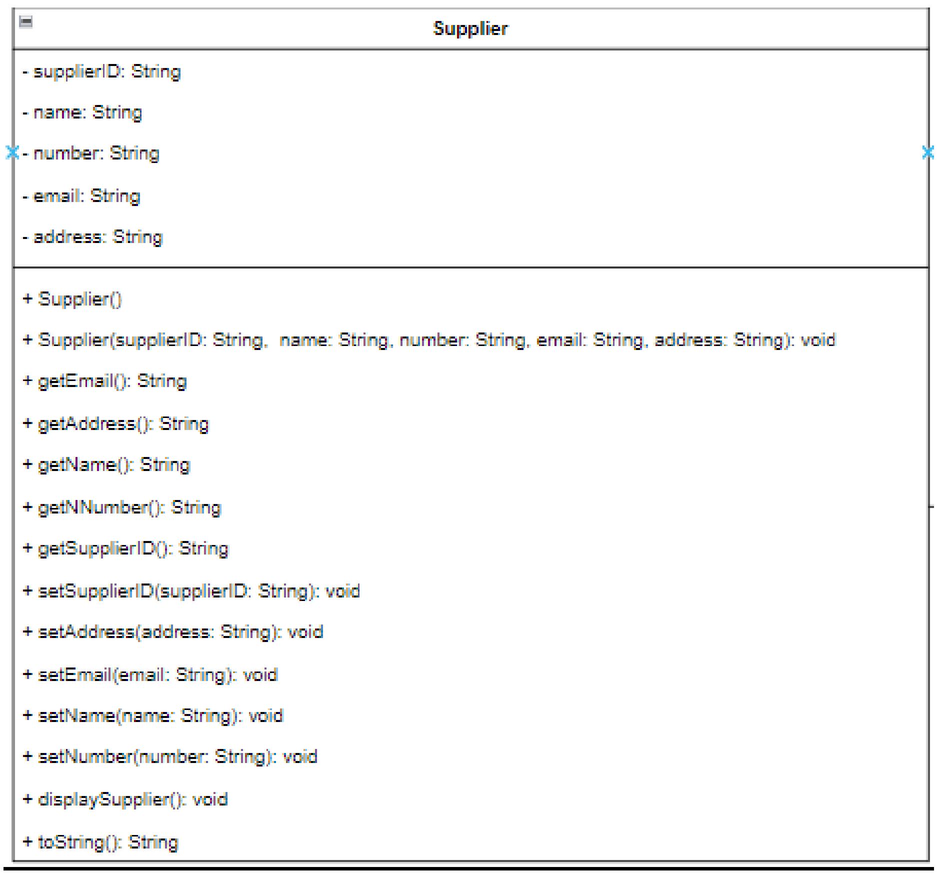
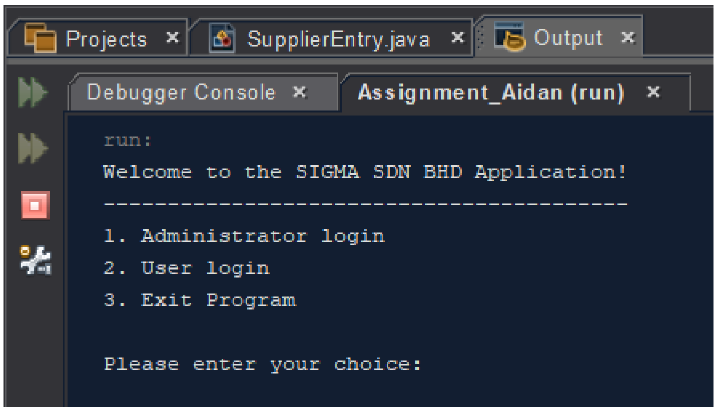

Supplier
- supplierID: String
- name: String
- number: String
- email: String
- address: String
+ Supplier()
+ Supplier(supplierID: String, name: String, number: String, email: String, address: String): void
+ getEmail(): String
+ getAddress(): String
+ getName(): String
+ getNNumber(): String
+ getSupplierID(): String
+ setSupplierID(supplierID: String): void
+ setAddress(address: String): void
+ setEmail(email: String): void
+ setName(name: String): void
+ setNumber(number: String): void
+ displaySupplier(): void
+ toString(): String

Figure 19: Supplier class

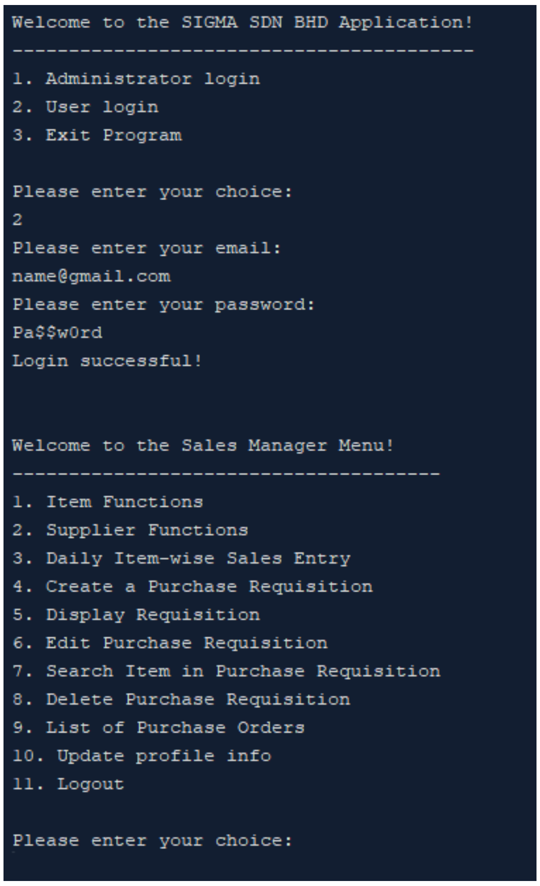
4.0 Screenshots of System Output. (User Guide)



```
run:
Welcome to the SIGMA SDN BHD Application!
-----------------------------------------
1. Administrator login
2. User login
3. Exit Program

Please enter your choice:
```

Figure 20: Main Homepage

This is the main home page for our system where the user will be prompted to choose whether they want to login as administrator or user or exit the system. Let's talk about User login first.



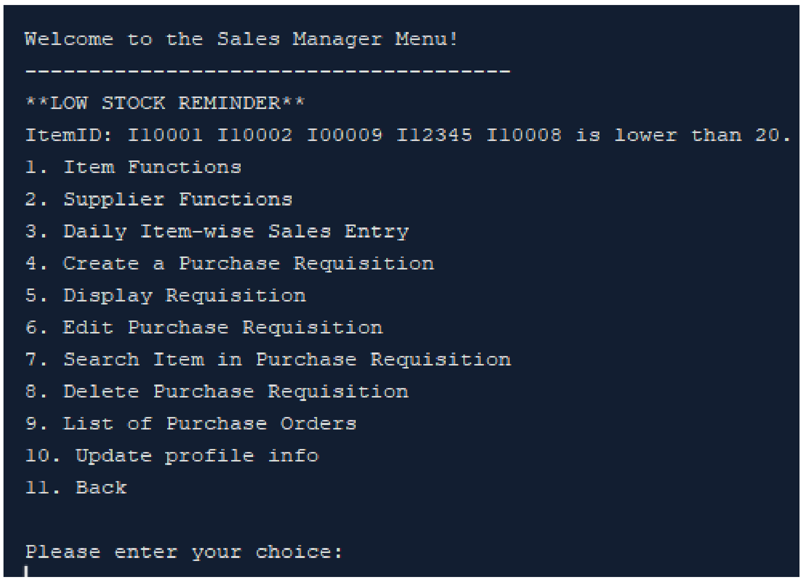
```
Welcome to the SIGMA SDN BHD Application!
-----------------------------------------
1. Administrator login
2. User login
3. Exit Program

Please enter your choice:
2
Please enter your email:
name@gmail.com
Please enter your password:
Pa$$w0rd
Login successful!


Welcome to the Sales Manager Menu!
--------------------------------------
1. Item Functions
2. Supplier Functions
3. Daily Item-wise Sales Entry
4. Create a Purchase Requisition
5. Display Requisition
6. Edit Purchase Requisition
7. Search Item in Purchase Requisition
8. Delete Purchase Requisition
9. List of Purchase Orders
10. Update profile info
11. Logout

Please enter your choice:
```

Figure 21: Sales Manager Menu

If the user choses the user login, then the system will ask the user to enter the email address followed by password. Since there are already pre-registered users in our system, we enter the email address: name@gmail.com and the password: Pa$$w0rd.

Once the user is verified then they will be redirected to the Sales Manager Functions page.

Sales Manager Functions

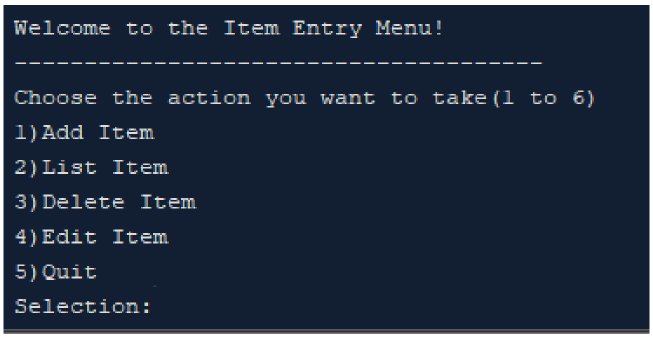


```
Welcome to the Sales Manager Menu!
--------------------------------------
**LOW STOCK REMINDER**
ItemID: I10001 I10002 I00009 I12345 I10008 is lower than 20.
1. Item Functions
2. Supplier Functions
3. Daily Item-wise Sales Entry
4. Create a Purchase Requisition
5. Display Requisition
6. Edit Purchase Requisition
7. Search Item in Purchase Requisition
8. Delete Purchase Requisition
9. List of Purchase Orders
10. Update profile info
11. Back

Please enter your choice:
```

Figure 22: Sales Manager Functions

The sales manager has 10 different functions that they can process. The first function is item function. The Sales Manager will get a notification of all the Items that are lower than 20 which will remind the SM to create PR to restock them.

Item Functions



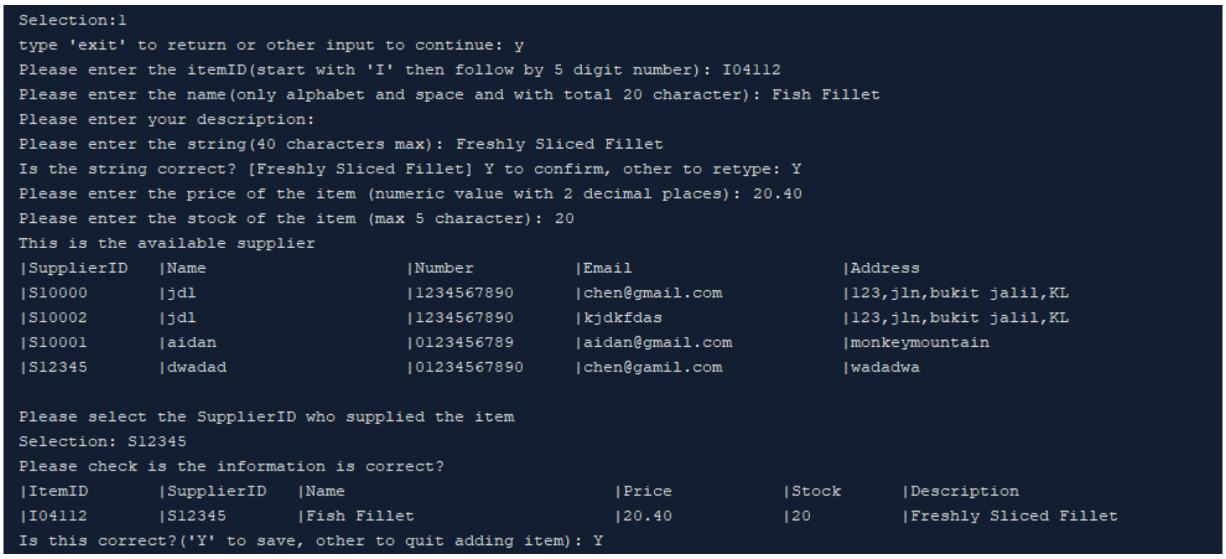
```
Welcome to the Item Entry Menu!
--------------------------------------
Choose the action you want to take(1 to 6)
1)Add Item
2)List Item
3)Delete Item
4)Edit Item
5)Quit
Selection:
```

Figure 23: Item Functions

Inside the Item Functions the Salas Manager can choose to Add item, List item, delete item, edit item, and quit.

Let's Start from the top:

Adding Item

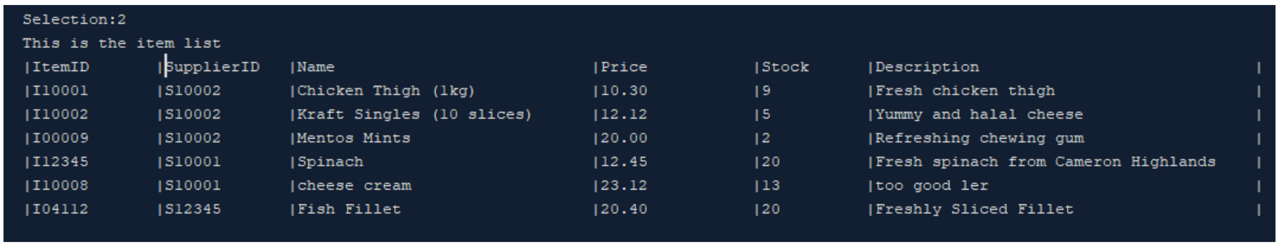


```
Selection:1
type 'exit' to return or other input to continue: y
Please enter the itemID(start with 'I' then follow by 5 digit number): I04112
Please enter the name(only alphabet and space and with total 20 character): Fish Fillet
Please enter your description:
Please enter the string(40 characters max): Freshly Sliced Fillet
Is the string correct? [Freshly Sliced Fillet] Y to confirm, other to retype: Y
Please enter the price of the item (numeric value with 2 decimal places): 20.40
Please enter the stock of the item (max 5 character): 20
This is the available supplier
|SupplierID   |Name                    |Number          |Email                     |Address
|S10000       |jdl                     |1234567890      |chen@gmail.com            |123,jln,bukit jalil,KL
|S10002       |jdl                     |1234567890      |kjdkfdas                  |123,jln,bukit jalil,KL
|S10001       |aidan                   |0123456789      |aidan@gmail.com           |monkeymountain
|S12345       |dwadad                  |01234567890     |chen@gamil.com            |wadadwa

Please select the SupplierID who supplied the item
Selection: S12345
Please check is the information is correct?
|ItemID       |SupplierID   |Name                           |Price           |Stock      |Description
|I04112       |S12345       |Fish Fillet                    |20.40           |20         |Freshly Sliced Fillet
Is this correct?('Y' to save, other to quit adding item): Y
```

Figure 24: Adding Item

As shown in the diagram above once the Adding item function has started the SM will first be prompted to confirm that they are going to add a product by entering any value or they can choose to return to the menu by typing exit. If the SM chooses to proceed with adding the item, then the SM will have to enter the ID for the item that they are trying to add. In the image above the ID 'I04112'. Following that the user is prompted to enter the name of the item. For this example, we have entered 'Fish Fillet'. After entering the name of the item, the SM must enter the description of the item which we have entered ‘Freshly Sliced Fillet’. Following that the SM must choose to confirm by typing ‘Y’ or they can choose to retype by entering any other value. If they choose to confirm then they will be asked to enter the price and stock of the item which we have used ’20.40’ for price and ‘20’ for stock. In addition to this the SM must choose which supplier this is from by entering the corresponding Supplier ID. Finally, the SM will be asked to confirm one last time before adding the item into the list and text file.

Listing Items

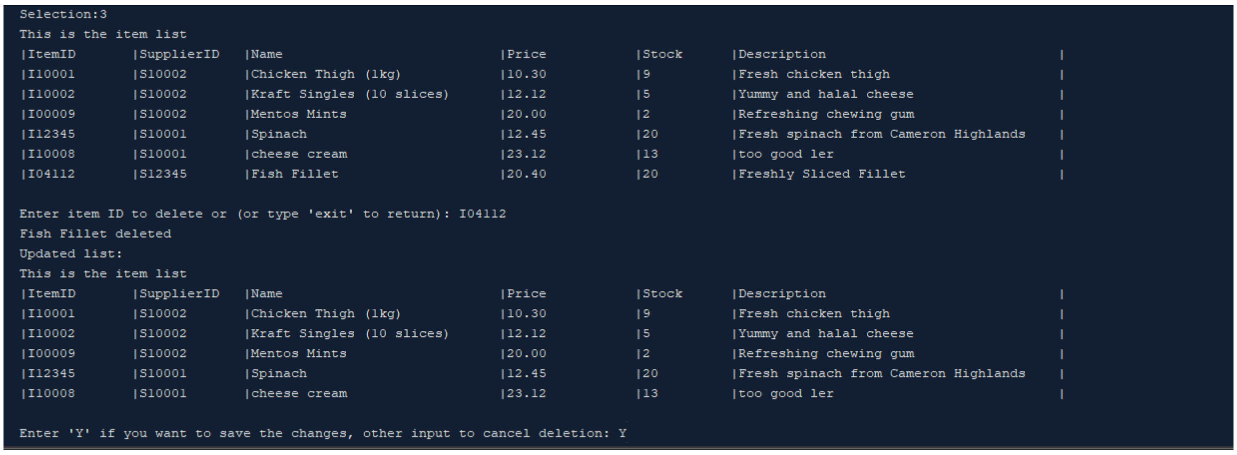


```
Selection:2
This is the item list
|ItemID      |SupplierID   |Name                       |Price          |Stock      |Description                          |
|I10001      |S10002       |Chicken Thigh (1kg)        |10.30          |9          |Fresh chicken thigh                  |
|I10002      |S10002       |Kraft Singles (10 slices)  |12.12          |5          |Yummy and halal cheese               |
|I00009      |S10002       |Mentos Mints               |20.00          |2          |Refreshing chewing gum               |
|I12345      |S10001       |Spinach                    |12.45          |20         |Fresh spinach from Cameron Highlands |
|I10008      |S10001       |cheese cream               |23.12          |13         |too good ler                         |
|I04112      |S12345       |Fish Fillet                |20.40          |20         |Freshly Sliced Fillet                |
```

Figure 25: Listing Items

If the SM chooses to List the Items, then a list of all the items in the text file will be displayed as shows above. As you can see the item we just added above named ‘Fish Fillet’ is in the list too.

Delete Item from List



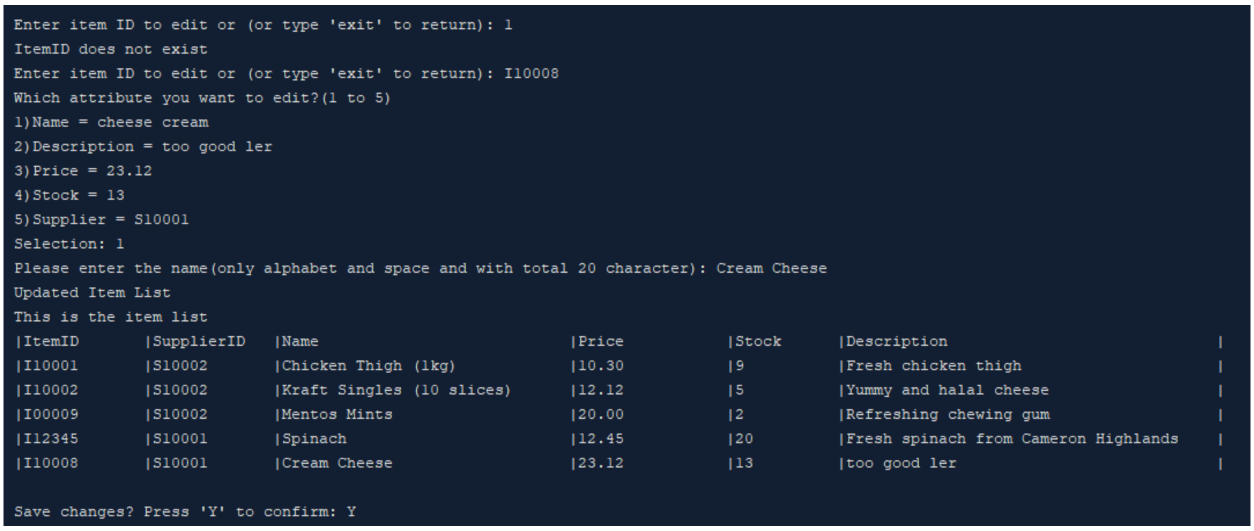
```
Selection:3
This is the item list
|ItemID      |SupplierID   |Name                       |Price          |Stock      |Description                          |
|I10001      |S10002       |Chicken Thigh (1kg)        |10.30          |9          |Fresh chicken thigh                  |
|I10002      |S10002       |Kraft Singles (10 slices)  |12.12          |5          |Yummy and halal cheese               |
|I00009      |S10002       |Mentos Mints               |20.00          |2          |Refreshing chewing gum               |
|I12345      |S10001       |Spinach                    |12.45          |20         |Fresh spinach from Cameron Highlands |
|I10008      |S10001       |cheese cream               |23.12          |13         |too good ler                         |
|I04112      |S12345       |Fish Fillet                |20.40          |20         |Freshly Sliced Fillet                |

Enter item ID to delete or (or type 'exit' to return): I04112
Fish Fillet deleted
Updated list:
This is the item list
|ItemID      |SupplierID   |Name                       |Price          |Stock      |Description                          |
|I10001      |S10002       |Chicken Thigh (1kg)        |10.30          |9          |Fresh chicken thigh                  |
|I10002      |S10002       |Kraft Singles (10 slices)  |12.12          |5          |Yummy and halal cheese               |
|I00009      |S10002       |Mentos Mints               |20.00          |2          |Refreshing chewing gum               |
|I12345      |S10001       |Spinach                    |12.45          |20         |Fresh spinach from Cameron Highlands |
|I10008      |S10001       |cheese cream               |23.12          |13         |too good ler                         |

Enter 'Y' if you want to save the changes, other input to cancel deletion: Y
```

Figure 26: Delete Item

If SM chooses to DELETE an item from the list which is the third option in the Item Functions, then they will be asked to enter the Item ID of that item. In our case we entered ‘I04112’ which is Fish fillet the item that we entered earlier. The chosen item will be displayed and then list without the item will be shown. Lastly a confirmation will be prompted to the user to enter 'Y' if you want to save the changes or type anything to cancel deletion.

Edit Item from List

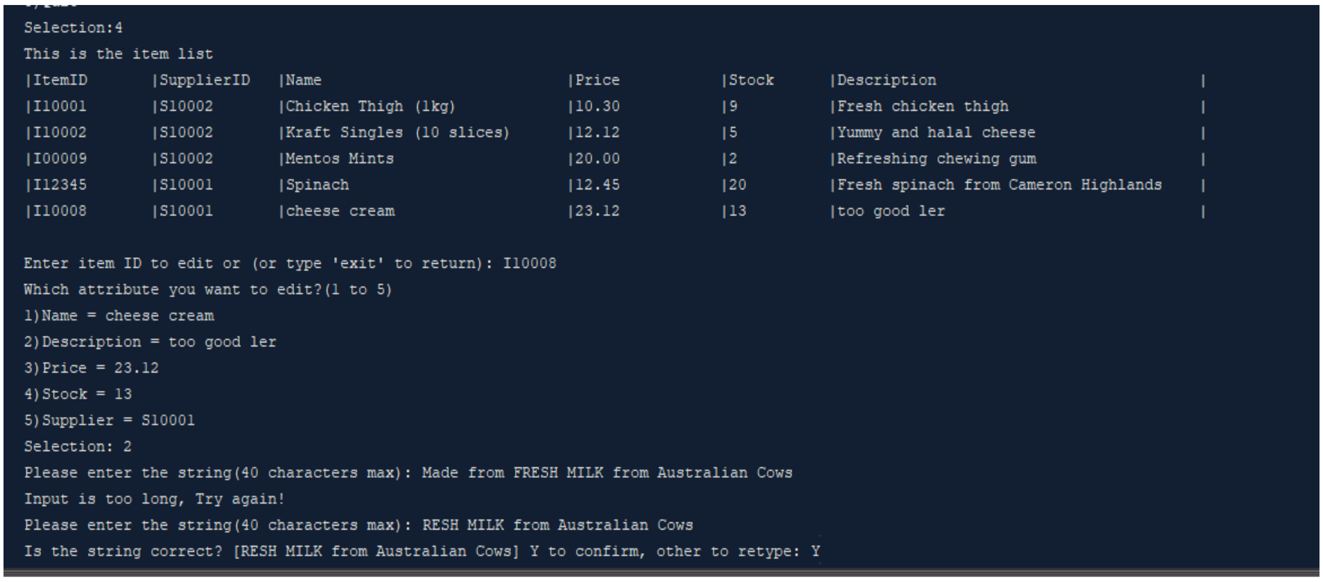
If SM chooses to edit a specific item from the list, then they are prompted to enter item ID of the item that you want to edit. If the item is identified from the text file, then the system will ask the SM to choose which category they would like to edit. The categories are name, description, price, stock, supplier.



```
Enter item ID to edit or (or type 'exit' to return): 1
ItemID does not exist
Enter item ID to edit or (or type 'exit' to return): I10008
Which attribute you want to edit?(1 to 5)
1)Name = cheese cream
2)Description = too good ler
3)Price = 23.12
4)Stock = 13
5)Supplier = S10001
Selection: 1
Please enter the name(only alphabet and space and with total 20 character): Cream Cheese
Updated Item List
This is the item list
|ItemID      |SupplierID   |Name                         |Price          |Stock      |Description                           |
|I10001      |S10002       |Chicken Thigh (1kg)          |10.30          |9          |Fresh chicken thigh                   |
|I10002      |S10002       |Kraft Singles (10 slices)    |12.12          |5          |Yummy and halal cheese                |
|I00009      |S10002       |Mentos Mints                 |20.00          |2          |Refreshing chewing gum                |
|I12345      |S10001       |Spinach                      |12.45          |20         |Fresh spinach from Cameron Highlands  |
|I10008      |S10001       |Cream Cheese                 |23.12          |13         |too good ler                          |

Save changes? Press 'Y' to confirm: Y
```

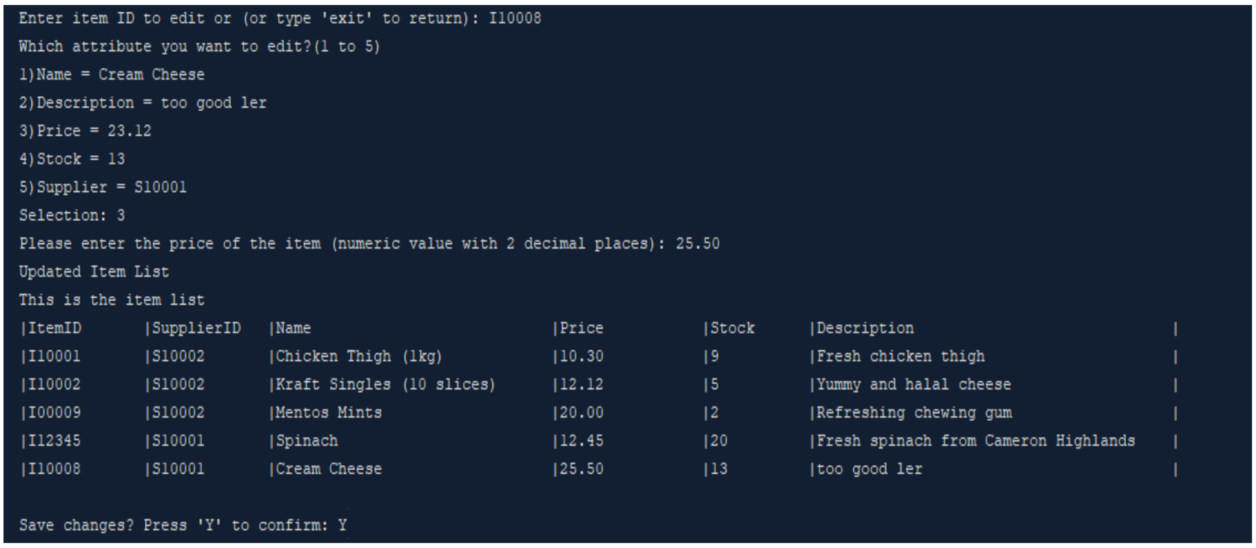
Figure 27: Edit name of item.

If the SM chooses to edit the name of the item, the chosen then they will be asked to enter the new item name which in the image above is 'Cream Cheese'. Once that is entered the new list is displayed and the user is asked to confirm by typing 'Y'.



```
Selection:4
This is the item list
|ItemID       |SupplierID   |Name                          |Price           |Stock      |Description                          |
|I10001       |S10002       |Chicken Thigh (1kg)           |10.30           |9          |Fresh chicken thigh                  |
|I10002       |S10002       |Kraft Singles (10 slices)     |12.12           |5          |Yummy and halal cheese               |
|I00009       |S10002       |Mentos Mints                  |20.00           |2          |Refreshing chewing gum               |
|I12345       |S10001       |Spinach                       |12.45           |20         |Fresh spinach from Cameron Highlands |
|I10008       |S10001       |cheese cream                  |23.12           |13         |too good ler                         |

Enter item ID to edit or (or type 'exit' to return): I10008
Which attribute you want to edit?(1 to 5)
1)Name = cheese cream
2)Description = too good ler
3)Price = 23.12
4)Stock = 13
5)Supplier = S10001
Selection: 2
Please enter the string(40 characters max): Made from FRESH MILK from Australian Cows
Input is too long, Try again!
Please enter the string(40 characters max): RESH MILK from Australian Cows
Is the string correct? [RESH MILK from Australian Cows] Y to confirm, other to retype: Y
```

Figure 28: Edit Item description

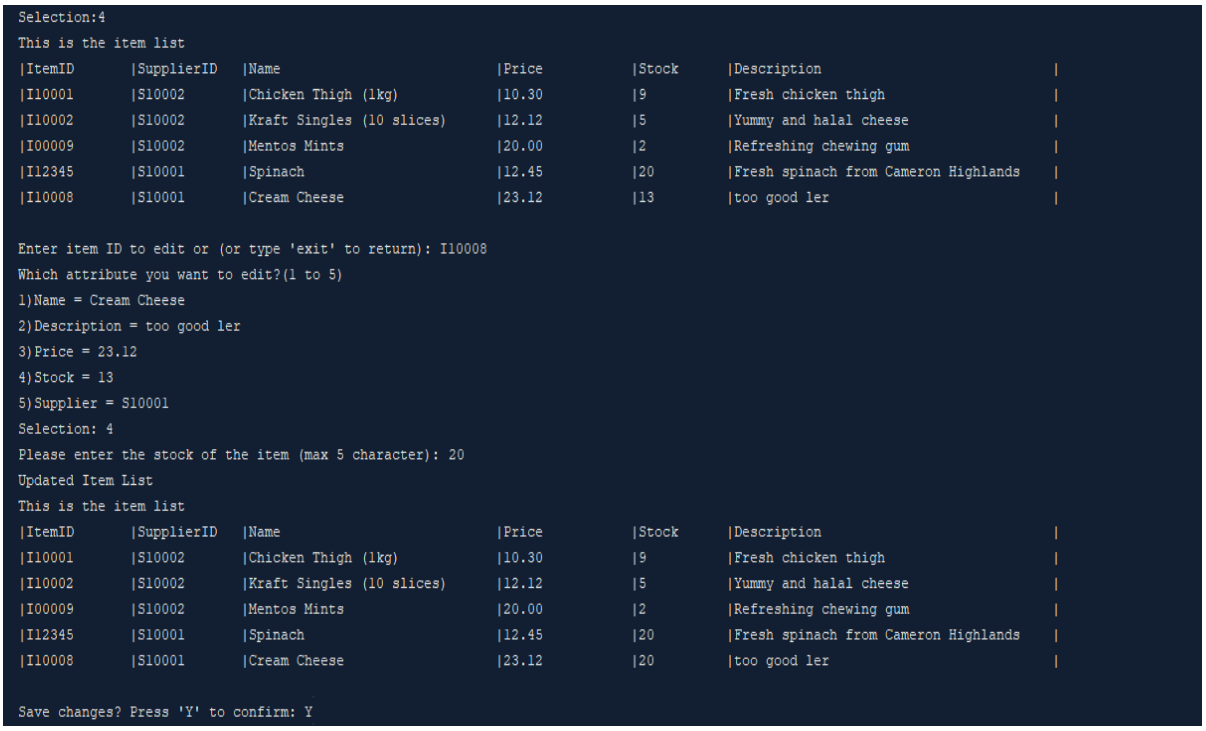


```
Enter item ID to edit or (or type 'exit' to return): I10008
Which attribute you want to edit?(1 to 5)
1)Name = Cream Cheese
2)Description = too good ler
3)Price = 23.12
4)Stock = 13
5)Supplier = S10001
Selection: 3
Please enter the price of the item (numeric value with 2 decimal places): 25.50
Updated Item List
This is the item list
|ItemID       |SupplierID   |Name                          |Price           |Stock      |Description                          |
|I10001       |S10002       |Chicken Thigh (1kg)           |10.30           |9          |Fresh chicken thigh                  |
|I10002       |S10002       |Kraft Singles (10 slices)     |12.12           |5          |Yummy and halal cheese               |
|I00009       |S10002       |Mentos Mints                  |20.00           |2          |Refreshing chewing gum               |
|I12345       |S10001       |Spinach                       |12.45           |20         |Fresh spinach from Cameron Highlands |
|I10008       |S10001       |Cream Cheese                  |25.50           |13         |too good ler                         |

Save changes? Press 'Y' to confirm: Y
```

Figure 29: Edit Item Price

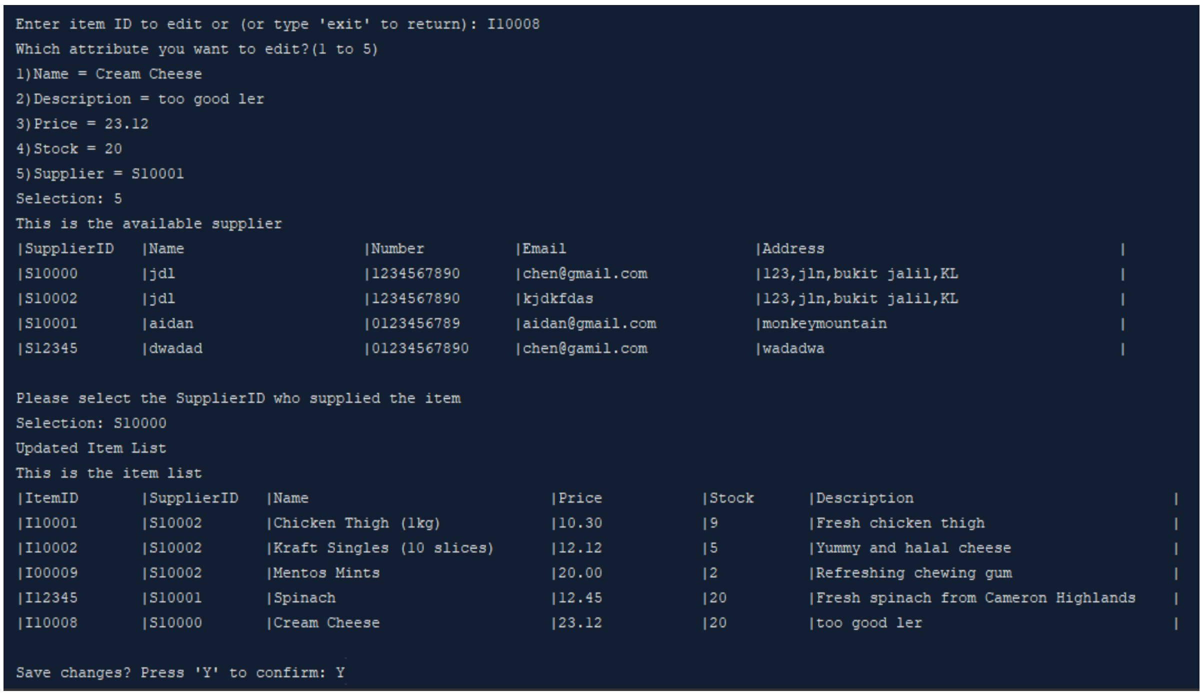
If SM chooses to edit the description of the selected item, Then the SM will be asked to enter a new string for the new description. As shown above, if the description is too long the system will provide an error message saying, “Input is too long, Try again!”. If the input is re-entered and is it under 40 characters, then the SM will be asked to confirm. If the SM is sure about it, then they must enter ‘Y’.

If the SM wants to enter the new price, then they can choose the new price with 2 decimal places and type ‘Y’ to confirm. In our case we entered “25.50”.



```
Selection:4
This is the item list
|ItemID      |SupplierID   |Name                          |Price          |Stock      |Description                           |
|I10001      |S10002       |Chicken Thigh (1kg)           |10.30          |9          |Fresh chicken thigh                   |
|I10002      |S10002       |Kraft Singles (10 slices)     |12.12          |5          |Yummy and halal cheese                |
|I00009      |S10002       |Mentos Mints                  |20.00          |2          |Refreshing chewing gum                |
|I12345      |S10001       |Spinach                       |12.45          |20         |Fresh spinach from Cameron Highlands  |
|I10008      |S10001       |Cream Cheese                  |23.12          |13         |too good ler                          |

Enter item ID to edit or (or type 'exit' to return): I10008
Which attribute you want to edit?(1 to 5)
1)Name = Cream Cheese
2)Description = too good ler
3)Price = 23.12
4)Stock = 13
5)Supplier = S10001
Selection: 4
Please enter the stock of the item (max 5 character): 20
Updated Item List
This is the item list
|ItemID      |SupplierID   |Name                          |Price          |Stock      |Description                           |
|I10001      |S10002       |Chicken Thigh (1kg)           |10.30          |9          |Fresh chicken thigh                   |
|I10002      |S10002       |Kraft Singles (10 slices)     |12.12          |5          |Yummy and halal cheese                |
|I00009      |S10002       |Mentos Mints                  |20.00          |2          |Refreshing chewing gum                |
|I12345      |S10001       |Spinach                       |12.45          |20         |Fresh spinach from Cameron Highlands  |
|I10008      |S10001       |Cream Cheese                  |23.12          |20         |too good ler                          |

Save changes? Press 'Y' to confirm: Y
```

Figure 30: Edit Stock of Item (1)



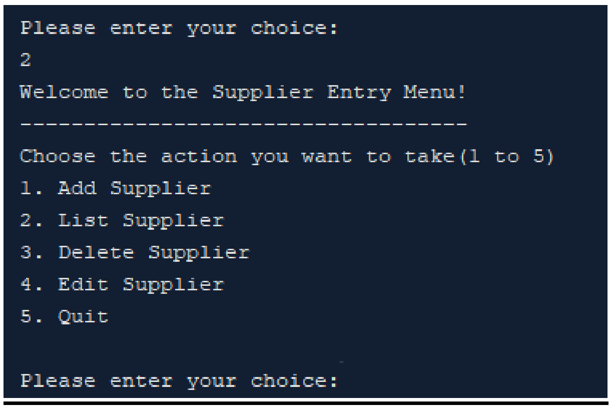
```
Enter item ID to edit or (or type 'exit' to return): I10008
Which attribute you want to edit?(1 to 5)
1)Name = Cream Cheese
2)Description = too good ler
3)Price = 23.12
4)Stock = 20
5)Supplier = S10001
Selection: 5
This is the available supplier
|SupplierID   |Name                     |Number          |Email                      |Address                             |
|S10000       |jdl                      |1234567890      |chen@gmail.com             |123,jln,bukit jalil,KL              |
|S10002       |jdl                      |1234567890      |kjdkfdas                   |123,jln,bukit jalil,KL              |
|S10001       |aidan                    |0123456789      |aidan@gmail.com            |monkeymountain                      |
|S12345       |dwadad                   |01234567890     |chen@gamil.com             |wadadwa                             |

Please select the SupplierID who supplied the item
Selection: S10000
Updated Item List
This is the item list
|ItemID      |SupplierID   |Name                          |Price          |Stock      |Description                           |
|I10001      |S10002       |Chicken Thigh (1kg)           |10.30          |9          |Fresh chicken thigh                   |
|I10002      |S10002       |Kraft Singles (10 slices)     |12.12          |5          |Yummy and halal cheese                |
|I00009      |S10002       |Mentos Mints                  |20.00          |2          |Refreshing chewing gum                |
|I12345      |S10001       |Spinach                       |12.45          |20         |Fresh spinach from Cameron Highlands  |
|I10008      |S10000       |Cream Cheese                  |23.12          |20         |too good ler                          |

Save changes? Press 'Y' to confirm: Y
```

Figure 31: Edit Stock of Item (2)

In the image above, the new stock is asked to be entered which was 20. Then the SM must confirm by typing ‘Y’.

If the SM wants to change the Supplier that supplies the item, the system will display all the available Supplier and then the SM can enter the new supplier ID. Then the SM must enter 'Y' to confirm. After the Item function the SM can enter '5' to quit and go back to the main menu.

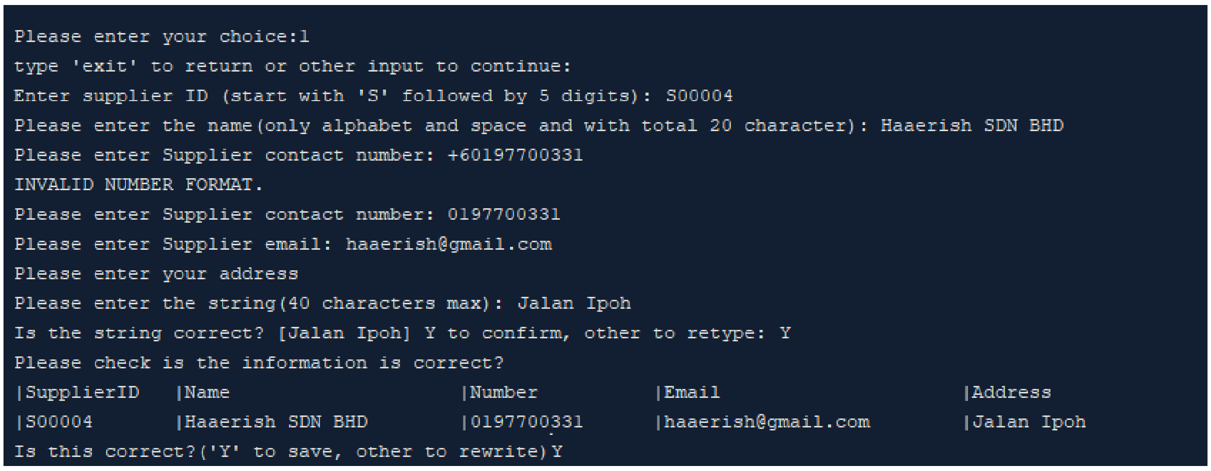
Supplier Functions



```
Please enter your choice:
2
Welcome to the Supplier Entry Menu!
-----------------------------------
Choose the action you want to take(1 to 5)
1. Add Supplier
2. List Supplier
3. Delete Supplier
4. Edit Supplier
5. Quit

Please enter your choice:
```

Figure 32: Supplier Functions

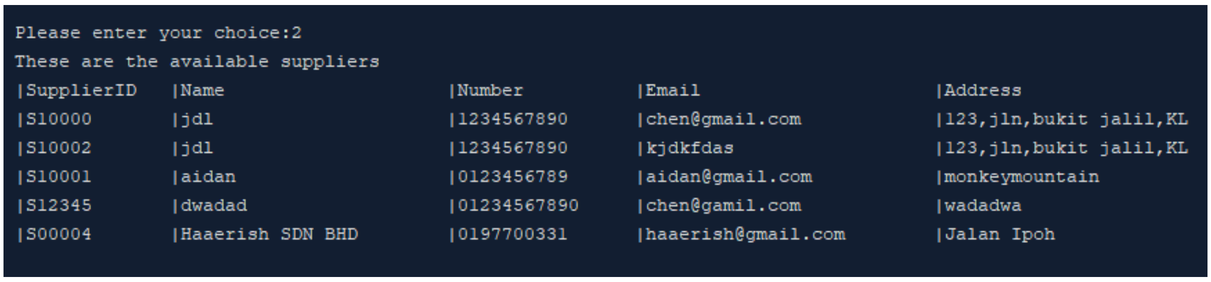
As shown above the SM can choose to Add, List, Delete and Edit supplier if they want to. On top of that they can also quit by typing '5'.



```
Please enter your choice:1
type 'exit' to return or other input to continue:
Enter supplier ID (start with 'S' followed by 5 digits): S00004
Please enter the name(only alphabet and space and with total 20 character): Haaerish SDN BHD
Please enter Supplier contact number: +60197700331
INVALID NUMBER FORMAT.
Please enter Supplier contact number: 0197700331
Please enter Supplier email: haaerish@gmail.com
Please enter your address
Please enter the string(40 characters max): Jalan Ipoh
Is the string correct? [Jalan Ipoh] Y to confirm, other to retype: Y
Please check is the information is correct?
|SupplierID   |Name                   |Number          |Email                      |Address
|S00004       |Haaerish SDN BHD       |0197700331      |haaerish@gmail.com         |Jalan Ipoh
Is this correct?('Y' to save, other to rewrite)Y
```

Figure 33: Adding Supplier

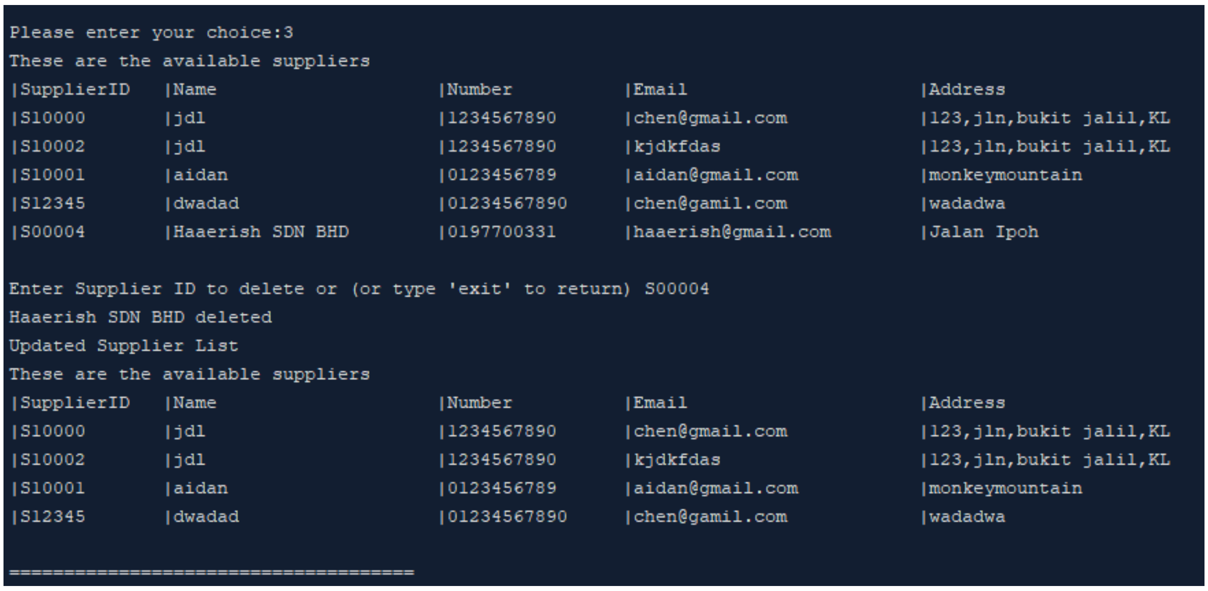
Once the SM chooses to add the supplier then they will be asked to enter the Supplier ID which is the Capital letter 'S' followed by 5 digits. Then the SM must enter the Suppliers: name, contact number, email address and address. Once all the information is entered correctly then the system will display it and the SM can choose to confirm by typing 'Y'.



```
Please enter your choice:2
These are the available suppliers
|SupplierID   |Name                        |Number          |Email                       |Address
|S10000       |jdl                         |1234567890      |chen@gmail.com              |123,jln,bukit jalil,KL
|S10002       |jdl                         |1234567890      |kjdkfdas                    |123,jln,bukit jalil,KL
|S10001       |aidan                       |0123456789      |aidan@gmail.com             |monkeymountain
|S12345       |dwadad                      |01234567890     |chen@gamil.com              |wadadwa
|S00004       |Haaerish SDN BHD            |0197700331      |haaerish@gmail.com          |Jalan Ipoh
```

Figure 34: Listing All Suppliers

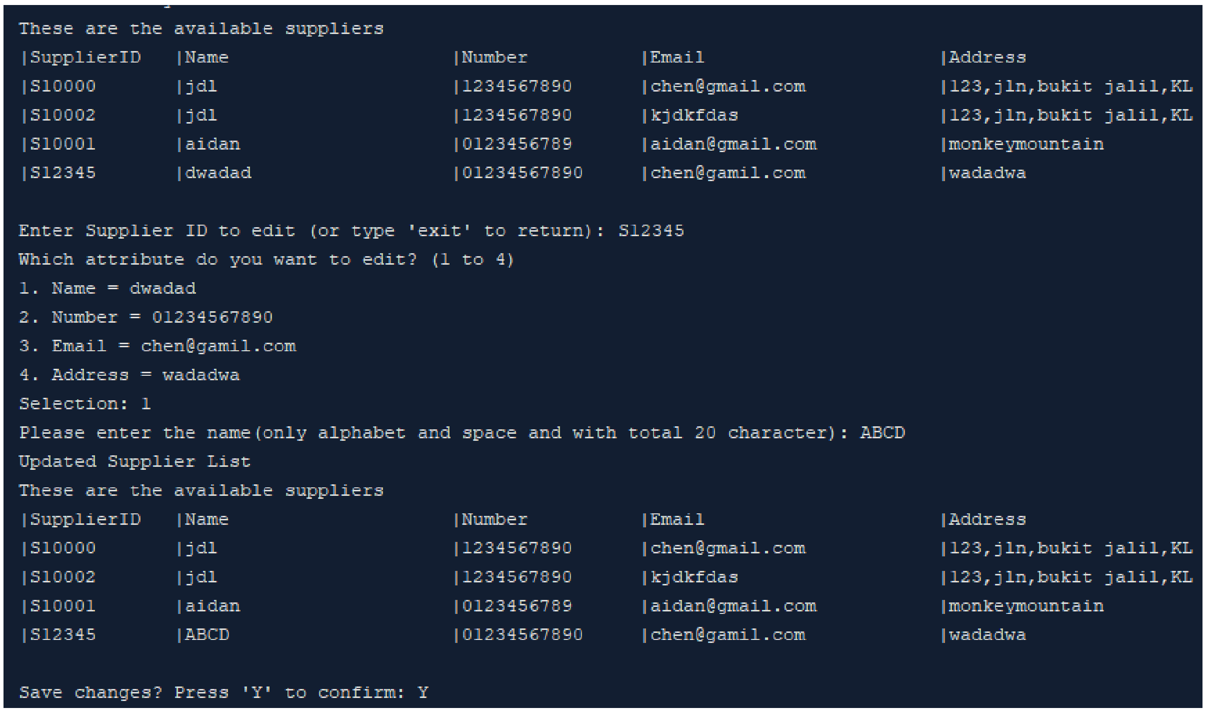
If the SM enters ‘2’ in the menu and chooses to view all the Suppliers, then the system will display all the registered Supplier and the information regarding it.



```
Please enter your choice:3
These are the available suppliers
|SupplierID   |Name                        |Number          |Email                       |Address
|S10000       |jdl                         |1234567890      |chen@gmail.com              |123,jln,bukit jalil,KL
|S10002       |jdl                         |1234567890      |kjdkfdas                    |123,jln,bukit jalil,KL
|S10001       |aidan                       |0123456789      |aidan@gmail.com             |monkeymountain
|S12345       |dwadad                      |01234567890     |chen@gamil.com              |wadadwa
|S00004       |Haaerish SDN BHD            |0197700331      |haaerish@gmail.com          |Jalan Ipoh

Enter Supplier ID to delete or (or type 'exit' to return) S00004
Haaerish SDN BHD deleted
Updated Supplier List
These are the available suppliers
|SupplierID   |Name                        |Number          |Email                       |Address
|S10000       |jdl                         |1234567890      |chen@gmail.com              |123,jln,bukit jalil,KL
|S10002       |jdl                         |1234567890      |kjdkfdas                    |123,jln,bukit jalil,KL
|S10001       |aidan                       |0123456789      |aidan@gmail.com             |monkeymountain
|S12345       |dwadad                      |01234567890     |chen@gamil.com              |wadadwa

=====================================
```

Figure 35: Deleting Supplier

If the SM chooses to delete the supplier, then they must enter the corresponding Supplier ID. If the Supplier ID is in the text file, then the program will remove it and display the new Supplier List.

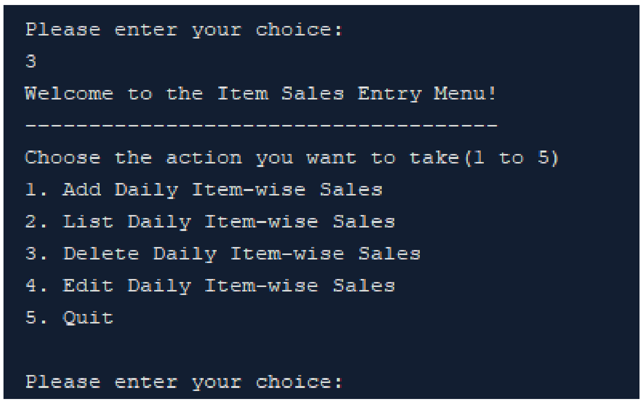


```
These are the available suppliers
|SupplierID   |Name                     |Number         |Email                     |Address
|S10000       |jdl                      |1234567890     |chen@gmail.com            |123,jln,bukit jalil,KL
|S10002       |jdl                      |1234567890     |kjdkfdas                  |123,jln,bukit jalil,KL
|S10001       |aidan                    |0123456789     |aidan@gmail.com           |monkeymountain
|S12345       |dwadad                   |01234567890    |chen@gamil.com            |wadadwa

Enter Supplier ID to edit (or type 'exit' to return): S12345
Which attribute do you want to edit? (1 to 4)
1. Name = dwadad
2. Number = 01234567890
3. Email = chen@gamil.com
4. Address = wadadwa
Selection: 1
Please enter the name(only alphabet and space and with total 20 character): ABCD
Updated Supplier List
These are the available suppliers
|SupplierID   |Name                     |Number         |Email                     |Address
|S10000       |jdl                      |1234567890     |chen@gmail.com            |123,jln,bukit jalil,KL
|S10002       |jdl                      |1234567890     |kjdkfdas                  |123,jln,bukit jalil,KL
|S10001       |aidan                    |0123456789     |aidan@gmail.com           |monkeymountain
|S12345       |ABCD                     |01234567890    |chen@gamil.com            |wadadwa

Save changes? Press 'Y' to confirm: Y
```

Figure 36: Editing Registered Supplier

The SM can choose to edit the Registered supplier. If they wish to do this, then they must enter the Supplier ID of the supplier that they wish to edit. For the example above, we chose the Supplier ID: S12345. Then the SM will be asked what attribute they want to edit. They can choose from: Name, number, email, and address. Once they have chosen the attribute and entered the new value the SM has to type 'Y' to save the changes.

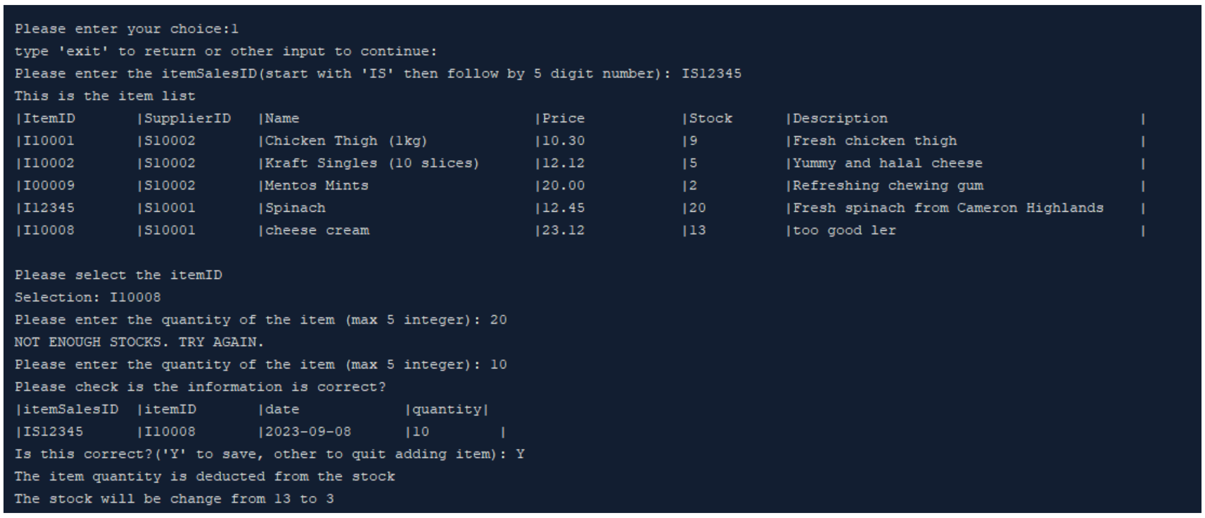
Daily Item-Wise Sales Entry



```
Please enter your choice:
3
Welcome to the Item Sales Entry Menu!
-------------------------------------
Choose the action you want to take(1 to 5)
1. Add Daily Item-wise Sales
2. List Daily Item-wise Sales
3. Delete Daily Item-wise Sales
4. Edit Daily Item-wise Sales
5. Quit

Please enter your choice:
```

Figure 37: Daily Item-Wise Sales Entry Menu

As shown in figure 21, this is the menu of the Daily Item-Wise Sales Entry. The SM can do the following actions: add, list, delete and edit the Daily Item-Wise Sales. The SM can also quit to the main menu by entering '5'.

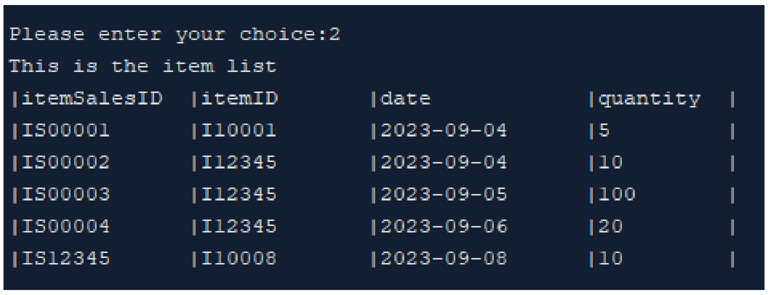


```
Please enter your choice:1
type 'exit' to return or other input to continue:
Please enter the itemSalesID(start with 'IS' then follow by 5 digit number): IS12345
This is the item list
|ItemID       |SupplierID   |Name                           |Price           |Stock      |Description                             |
|I10001       |S10002       |Chicken Thigh (1kg)            |10.30           |9          |Fresh chicken thigh                     |
|I10002       |S10002       |Kraft Singles (10 slices)      |12.12           |5          |Yummy and halal cheese                  |
|I00009       |S10002       |Mentos Mints                   |20.00           |2          |Refreshing chewing gum                  |
|I12345       |S10001       |Spinach                        |12.45           |20         |Fresh spinach from Cameron Highlands    |
|I10008       |S10001       |cheese cream                   |23.12           |13         |too good ler                            |

Please select the itemID
Selection: I10008
Please enter the quantity of the item (max 5 integer): 20
NOT ENOUGH STOCKS. TRY AGAIN.
Please enter the quantity of the item (max 5 integer): 10
Please check is the information is correct?
|itemSalesID  |itemID       |date             |quantity|
|IS12345      |I10008       |2023-09-08       |10        |
Is this correct?('Y' to save, other to quit adding item): Y
The item quantity is deducted from the stock
The stock will be change from 13 to 3
```

Figure 38: Adding Daily Sales

As shown in figure 22, the SM can choose to enter the daily sales. For this the SM must create a new ItemSales ID by starting with 'IS' and followed by 5 digits. For example, I have chosen 'IS12345'. Then the SM must choose the items that was sold by entering the Item ID. Once the Item is chosen the SM needs to enter the quantity sold of that item. As shown above for validation purposes if the SM enters more quantity than the available stock the system will display and error message saying "NOT ENOUGH STOCKS. TRY AGAIN." If the SM enters a value below the stock value, then the details is displayed, and the SM must confirm by typing 'Y'. Once the SM confirms it the system will show to new stock value for that item.

Listing Daily Sales Entry

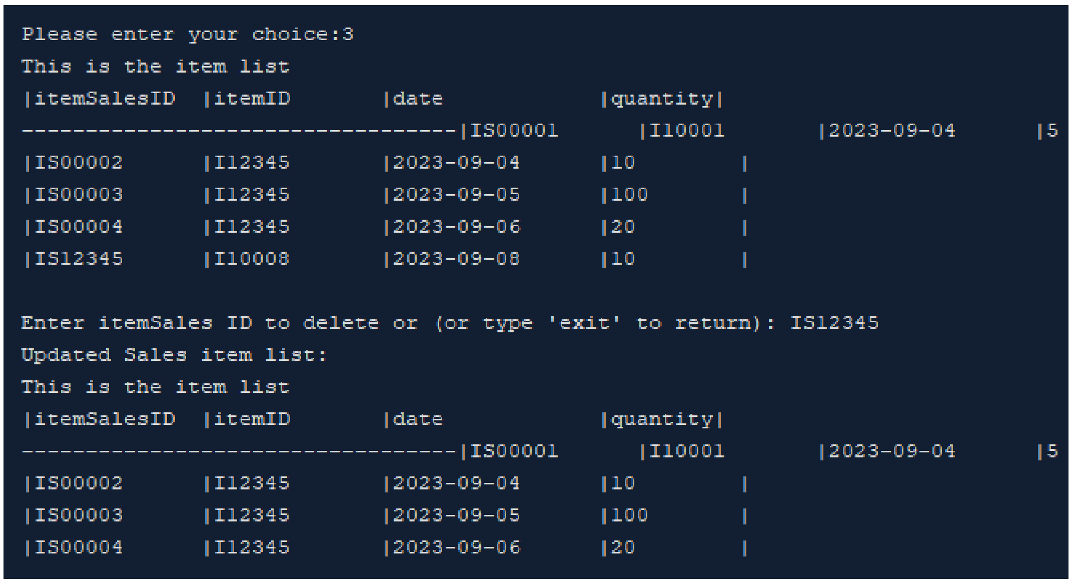


```
Please enter your choice:2
This is the item list
|itemSalesID  |itemID       |date            |quantity  |
|IS00001      |I10001       |2023-09-04      |5         |
|IS00002      |I12345       |2023-09-04      |10        |
|IS00003      |I12345       |2023-09-05      |100       |
|IS00004      |I12345       |2023-09-06      |20        |
|IS12345      |I10008       |2023-09-08      |10        |
```

Figure 39: List of Sales

If the SM types '2' in the menu, then the SM will display the list of Daily Sales as shown above. The sales that we entered in the previous image is now shown here which means it was successfully added.

Deleting Daily Sales Entry

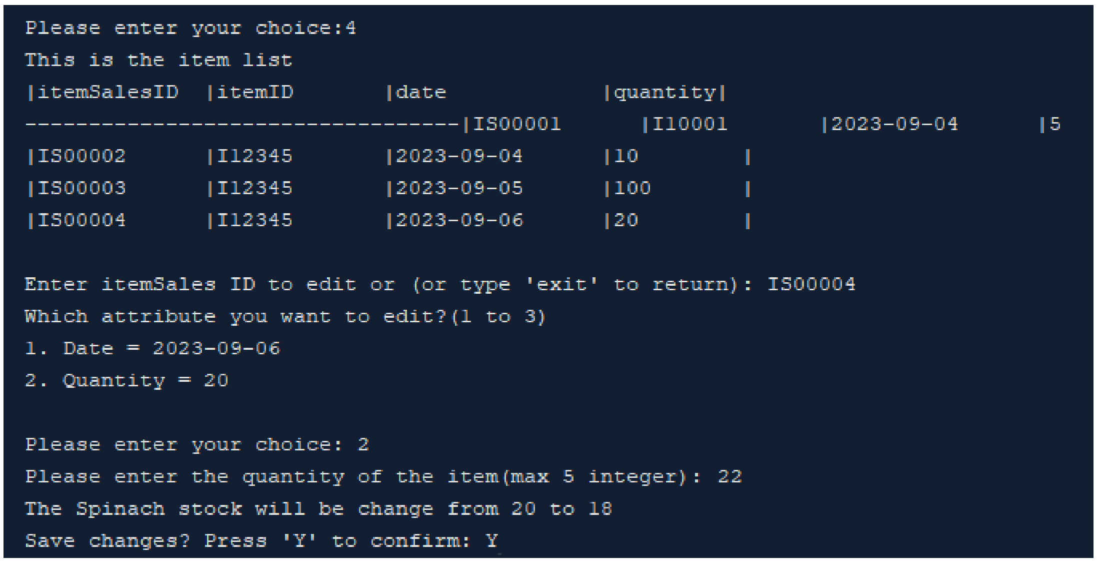


```
Please enter your choice:3
This is the item list
|itemSalesID  |itemID       |date            |quantity|
----------------------------------|IS00001      |I10001       |2023-09-04      |5
|IS00002      |I12345       |2023-09-04      |10        |
|IS00003      |I12345       |2023-09-05      |100       |
|IS00004      |I12345       |2023-09-06      |20        |
|IS12345      |I10008       |2023-09-08      |10        |

Enter itemSales ID to delete or (or type 'exit' to return): IS12345
Updated Sales item list:
This is the item list
|itemSalesID  |itemID       |date            |quantity|
----------------------------------|IS00001      |I10001       |2023-09-04      |5
|IS00002      |I12345       |2023-09-04      |10        |
|IS00003      |I12345       |2023-09-05      |100       |
|IS00004      |I12345       |2023-09-06      |20        |
```

Figure 40: Deleting from daily Item sales entry.

As shown in figure 20, the SM can choose to delete a specific daily item sales entry. This is done by entering the Item Sales ID of the ItemSales that is to be deleted. If the ItemSales ID exists in the text file, then it will be removed from the list. A new list will be displayed showing all the remaining Item sales entry.

Editing Daily Sales Entry



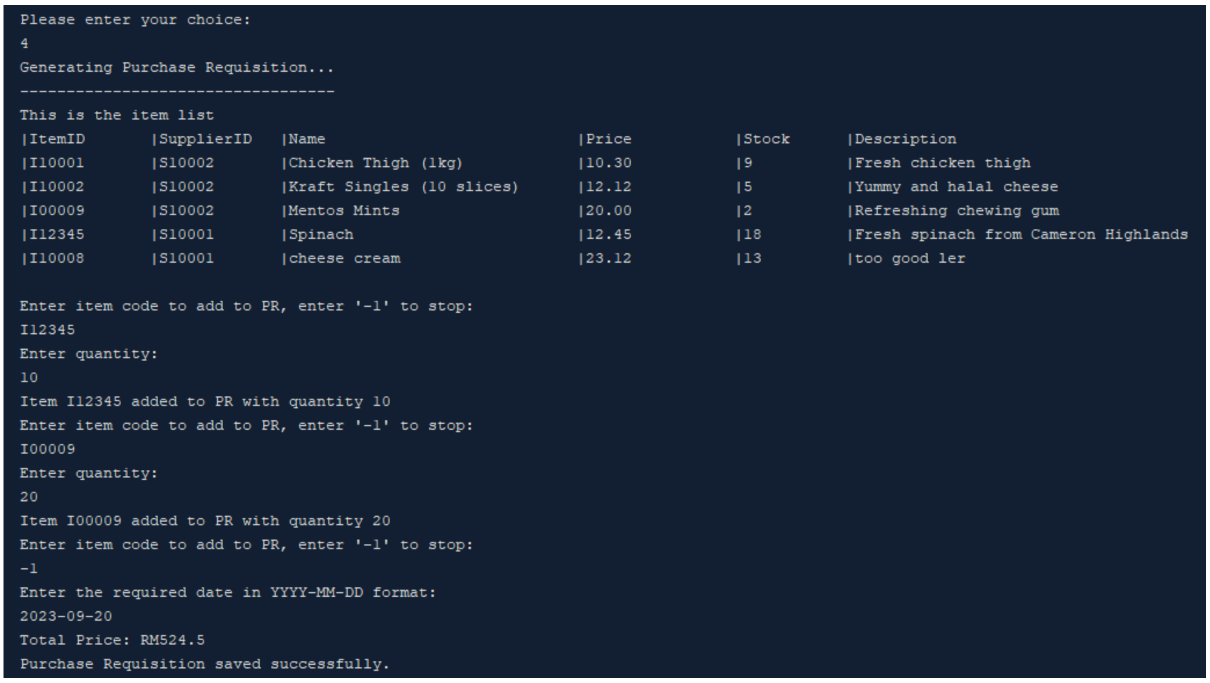
```
Please enter your choice:4
This is the item list
|itemSalesID  |itemID       |date            |quantity|
----------------------------------|IS00001      |I10001       |2023-09-04      |5
|IS00002      |I12345       |2023-09-04      |10        |
|IS00003      |I12345       |2023-09-05      |100       |
|IS00004      |I12345       |2023-09-06      |20        |

Enter itemSales ID to edit or (or type 'exit' to return): IS00004
Which attribute you want to edit?(1 to 3)
1. Date = 2023-09-06
2. Quantity = 20

Please enter your choice: 2
Please enter the quantity of the item(max 5 integer): 22
The Spinach stock will be change from 20 to 18
Save changes? Press 'Y' to confirm: Y
```

Figure 41: Editing from daily sales entry

As shown in the figure above, the other function that the SM can do under the Daily Sales Entry is editing the sales entry. For that the SM needs to enter the ItemSales Id which will then prompt the SM to choose between the changeable attributes which are date and quantity. For the example above I have decided to edit the quantity from 20 to 22. This will also reduce the stock of the item inside the sales which is spinach by 2. Then the SM has to enter ‘Y’ to confirm the changes. This shows there is a functional integration and relationship between the quantity in the Item sales entry and the stock of the item inside.

Purchase Requisition

Creating Purchase Requisition

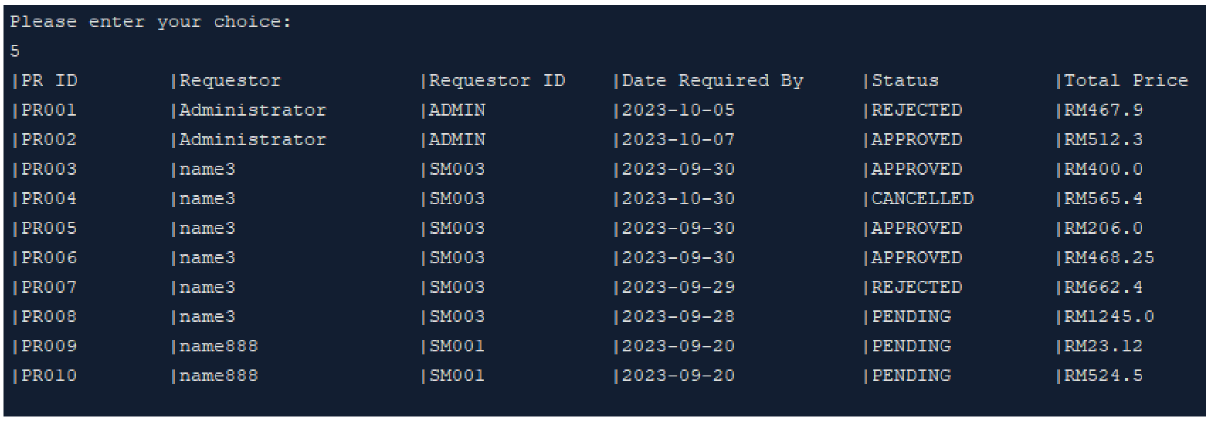


```
Please enter your choice:
4
Generating Purchase Requisition...
----------------------------------
This is the item list
|ItemID      |SupplierID   |Name                           |Price          |Stock       |Description
|I10001      |S10002       |Chicken Thigh (1kg)            |10.30          |9           |Fresh chicken thigh
|I10002      |S10002       |Kraft Singles (10 slices)      |12.12          |5           |Yummy and halal cheese
|I00009      |S10002       |Mentos Mints                   |20.00          |2           |Refreshing chewing gum
|I12345      |S10001       |Spinach                        |12.45          |18          |Fresh spinach from Cameron Highlands
|I10008      |S10001       |cheese cream                   |23.12          |13          |too good ler

Enter item code to add to PR, enter '-1' to stop:
I12345
Enter quantity:
10
Item I12345 added to PR with quantity 10
Enter item code to add to PR, enter '-1' to stop:
I00009
Enter quantity:
20
Item I00009 added to PR with quantity 20
Enter item code to add to PR, enter '-1' to stop:
-1
Enter the required date in YYYY-MM-DD format:
2023-09-20
Total Price: RM524.5
Purchase Requisition saved successfully.
```

Figure 42: Creating Purchase Requisition

SM has a very important role to create a purchase requisition. If SM wants to create a PR, then they will have to enter the ItemID of the item that has to be added and the quantity that needs to be purchased. Once the SM enters all the items, the SM must type '-1' to stop adding item and they will be prompted to enter the required delivery date. The format for the date is in YYYY-MM-DD. Once the date is entered correctly then total cost of buying all the items will be shown. In our case it was RM 524.50.

Display the Purchase Requisitions

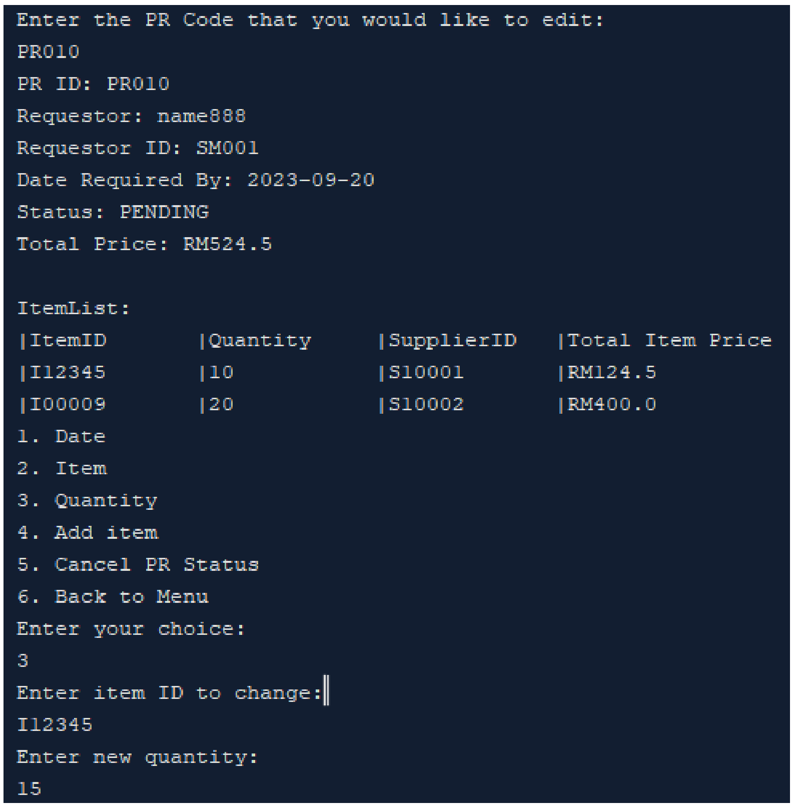


```
Please enter your choice:
5
|PR ID        |Requestor            |Requestor ID    |Date Required By     |Status          |Total Price
|PR001        |Administrator        |ADMIN           |2023-10-05           |REJECTED        |RM467.9
|PR002        |Administrator        |ADMIN           |2023-10-07           |APPROVED        |RM512.3
|PR003        |name3                |SM003           |2023-09-30           |APPROVED        |RM400.0
|PR004        |name3                |SM003           |2023-10-30           |CANCELLED       |RM565.4
|PR005        |name3                |SM003           |2023-09-30           |APPROVED        |RM206.0
|PR006        |name3                |SM003           |2023-09-30           |APPROVED        |RM468.25
|PR007        |name3                |SM003           |2023-09-29           |REJECTED        |RM662.4
|PR008        |name3                |SM003           |2023-09-28           |PENDING         |RM1245.0
|PR009        |name888              |SM001           |2023-09-20           |PENDING         |RM23.12
|PR010        |name888              |SM001           |2023-09-20           |PENDING         |RM524.5
```

Figure 43: Display the Purchase Requisition

If the SM wishes to see all the PR, they can enter '5' in the menu which will display all the PR that is in the system. As you can see above the PR that we just made with the delivery date of 2023-09-20 has already been added.

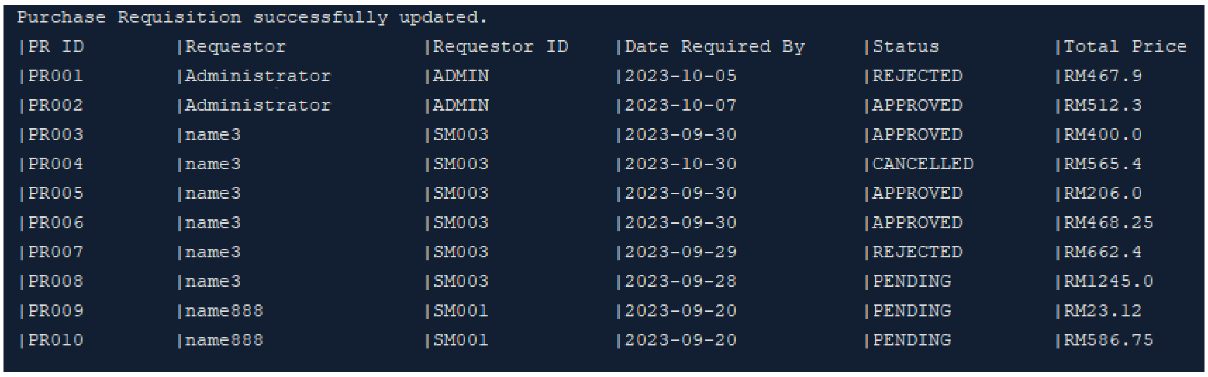
Edit the Purchase Requisitions



```
Enter the PR Code that you would like to edit:
PR010
PR ID: PR010
Requestor: name888
Requestor ID: SM001
Date Required By: 2023-09-20
Status: PENDING
Total Price: RM524.5

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price
|I12345        |10            |S10001        |RM124.5
|I00009        |20            |S10002        |RM400.0
1. Date
2. Item
3. Quantity
4. Add item
5. Cancel PR Status
6. Back to Menu
Enter your choice:
3
Enter item ID to change:
I12345
Enter new quantity:
15
```

Figure 44: Edit the Quantity in the PR

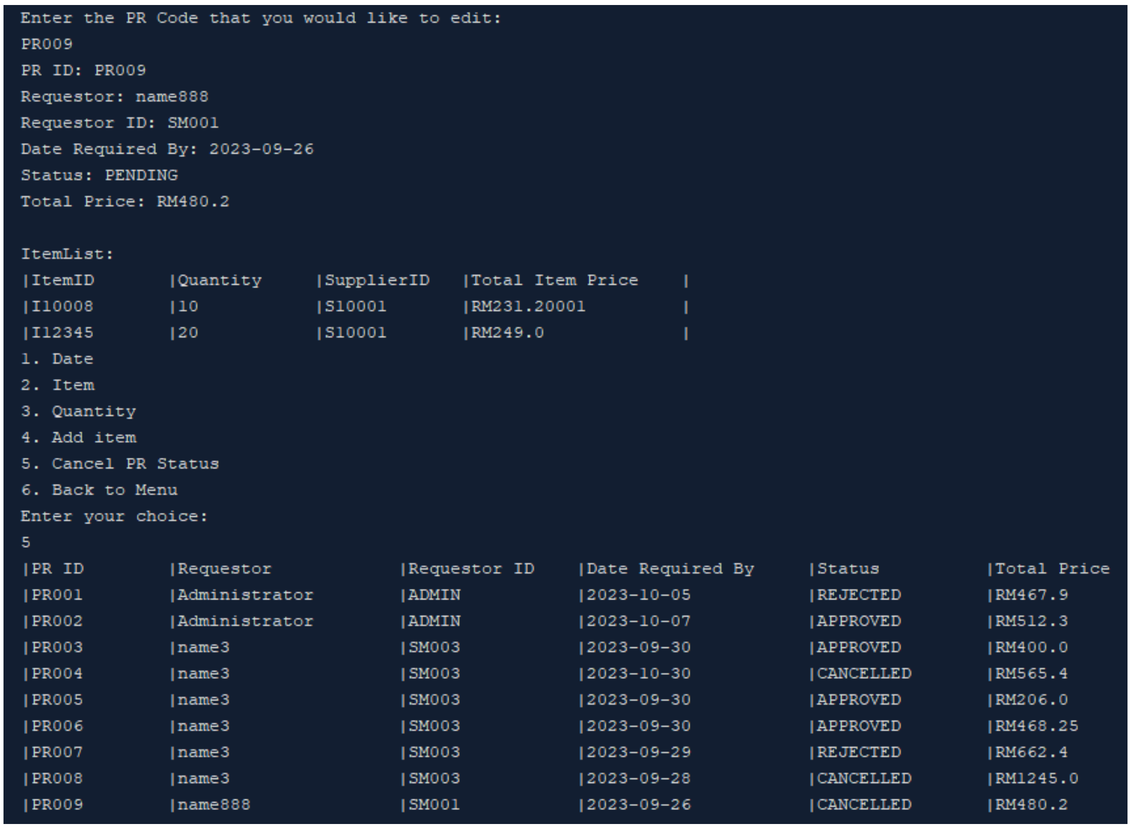
The image above shows changing the quantity of an item inside a PR. SM can do this by entering the PR ID of the PR that they would like to edit. After that the SM can choose to edit the date, item, quantity, add item and cancel the PR status.



```
Purchase Requisition successfully updated.
|PR ID      |Requestor           |Requestor ID    |Date Required By     |Status         |Total Price
|PR001      |Administrator       |ADMIN           |2023-10-05           |REJECTED       |RM467.9
|PR002      |Administrator       |ADMIN           |2023-10-07           |APPROVED       |RM512.3
|PR003      |name3               |SM003           |2023-09-30           |APPROVED       |RM400.0
|PR004      |name3               |SM003           |2023-10-30           |CANCELLED      |RM565.4
|PR005      |name3               |SM003           |2023-09-30           |APPROVED       |RM206.0
|PR006      |name3               |SM003           |2023-09-30           |APPROVED       |RM468.25
|PR007      |name3               |SM003           |2023-09-29           |REJECTED       |RM662.4
|PR008      |name3               |SM003           |2023-09-28           |PENDING        |RM1245.0
|PR009      |name888             |SM001           |2023-09-20           |PENDING        |RM23.12
|PR010      |name888             |SM001           |2023-09-20           |PENDING        |RM586.75
```

Figure 45: Updated PR

As your can see in Figure 29 since the quantity was increased the total prices has also been changed and updated into the list above.

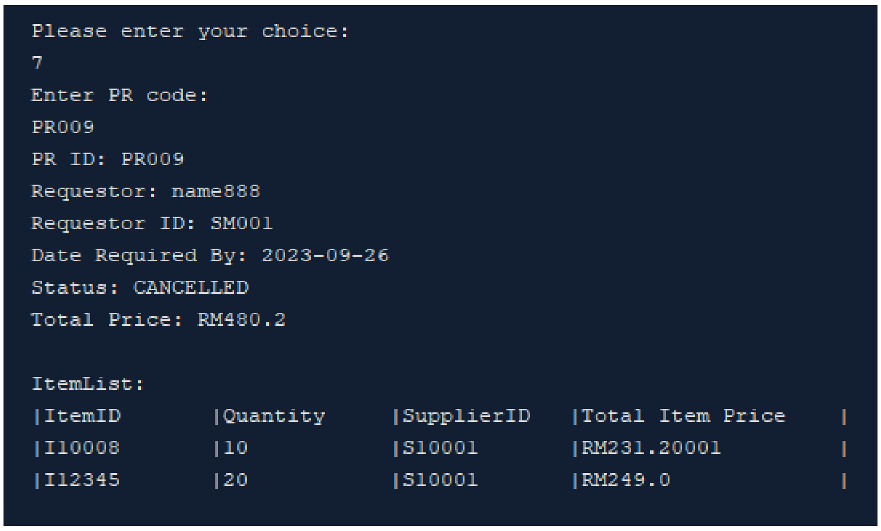


```
Enter the PR Code that you would like to edit:
PR009
PR ID: PR009
Requestor: name888
Requestor ID: SM001
Date Required By: 2023-09-26
Status: PENDING
Total Price: RM480.2

ItemList:
|ItemID       |Quantity     |SupplierID   |Total Item Price    |
|I10008       |10           |S10001       |RM231.20001         |
|I12345       |20           |S10001       |RM249.0             |
1. Date
2. Item
3. Quantity
4. Add item
5. Cancel PR Status
6. Back to Menu
Enter your choice:
5
|PR ID       |Requestor          |Requestor ID     |Date Required By     |Status         |Total Price
|PR001       |Administrator      |ADMIN            |2023-10-05           |REJECTED       |RM467.9
|PR002       |Administrator      |ADMIN            |2023-10-07           |APPROVED       |RM512.3
|PR003       |name3              |SM003            |2023-09-30           |APPROVED       |RM400.0
|PR004       |name3              |SM003            |2023-10-30           |CANCELLED      |RM565.4
|PR005       |name3              |SM003            |2023-09-30           |APPROVED       |RM206.0
|PR006       |name3              |SM003            |2023-09-30           |APPROVED       |RM468.25
|PR007       |name3              |SM003            |2023-09-29           |REJECTED       |RM662.4
|PR008       |name3              |SM003            |2023-09-28           |CANCELLED      |RM1245.0
|PR009       |name888            |SM001            |2023-09-26           |CANCELLED      |RM480.2
```

Figure 46: Cancelling PR Status

On top of being able to edit the details of the PR, the SM is also able to Cancel the PR status of any PR with the Pending status. As shown above PR009 was pending and after using the cancel function the Status is now Cancelled.

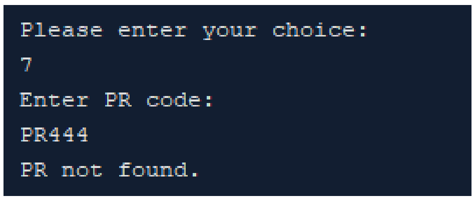
Search the Purchase Requisitions



```
Please enter your choice:
7
Enter PR code:
PR009
PR ID: PR009
Requestor: name888
Requestor ID: SM001
Date Required By: 2023-09-26
Status: CANCELLED
Total Price: RM480.2

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price    |
|I10008        |10            |S10001        |RM231.20001         |
|I12345        |20            |S10001        |RM249.0             |
```

Figure 47: Searching Purchase Requisition

As shown in the diagram above the SM can search the PR for a specific PR with the PR ID. When the ID of an existing PR is entered the program will display all details about the PR including the Status and the Item inside the PR.

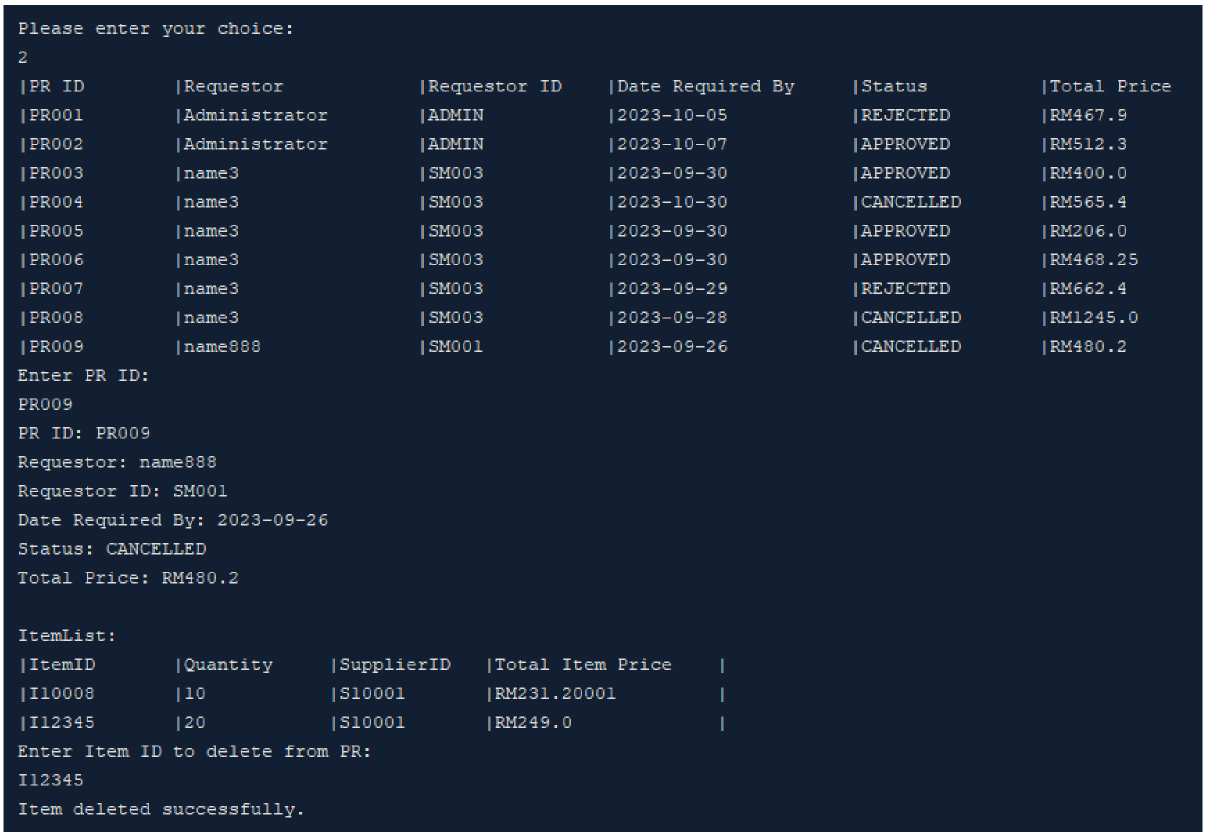


```
Please enter your choice:
7
Enter PR code:
PR444
PR not found.
```

Figure 48: Validation (Wrong PRID)

If the SM enters an invalid PR ID, then the system will provide a separate error message. The message is “PR not found”.

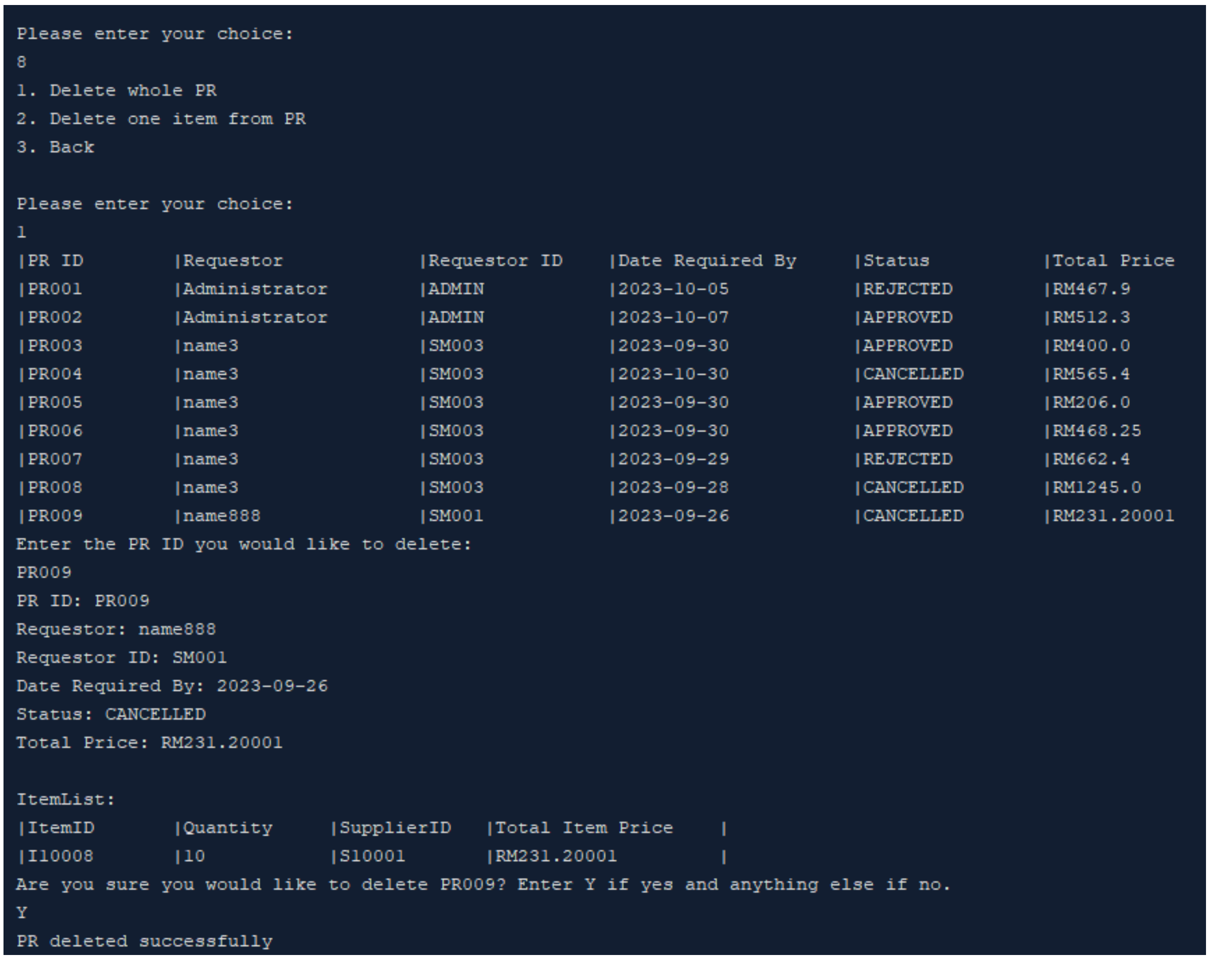
Delete the Purchase Requisitions



```
Please enter your choice:
2
|PR ID        |Requestor            |Requestor ID    |Date Required By    |Status          |Total Price
|PR001        |Administrator        |ADMIN           |2023-10-05          |REJECTED        |RM467.9
|PR002        |Administrator        |ADMIN           |2023-10-07          |APPROVED        |RM512.3
|PR003        |name3                |SM003           |2023-09-30          |APPROVED        |RM400.0
|PR004        |name3                |SM003           |2023-10-30          |CANCELLED       |RM565.4
|PR005        |name3                |SM003           |2023-09-30          |APPROVED        |RM206.0
|PR006        |name3                |SM003           |2023-09-30          |APPROVED        |RM468.25
|PR007        |name3                |SM003           |2023-09-29          |REJECTED        |RM662.4
|PR008        |name3                |SM003           |2023-09-28          |CANCELLED       |RM1245.0
|PR009        |name888              |SM001           |2023-09-26          |CANCELLED       |RM480.2
Enter PR ID:
PR009
PR ID: PR009
Requestor: name888
Requestor ID: SM001
Date Required By: 2023-09-26
Status: CANCELLED
Total Price: RM480.2

ItemList:
|ItemID       |Quantity     |SupplierID    |Total Item Price    |
|I10008       |10           |S10001        |RM231.20001         |
|I12345       |20           |S10001        |RM249.0             |
Enter Item ID to delete from PR:
I12345
Item deleted successfully.
```

Figure 49: Delete singular item from PR.

The SM can choose to either delete the entire PR or a singular item from the PR. Figure 29 shows a singular item being deleted from the PR. This is done by first entering which PR is the SM targeting then the PR is displayed. After that the SM must choose which Item from the PR they want to delete and enter the corresponding Item ID. If the Item ID is found, then it is deleted from the text file and a message is displayed.

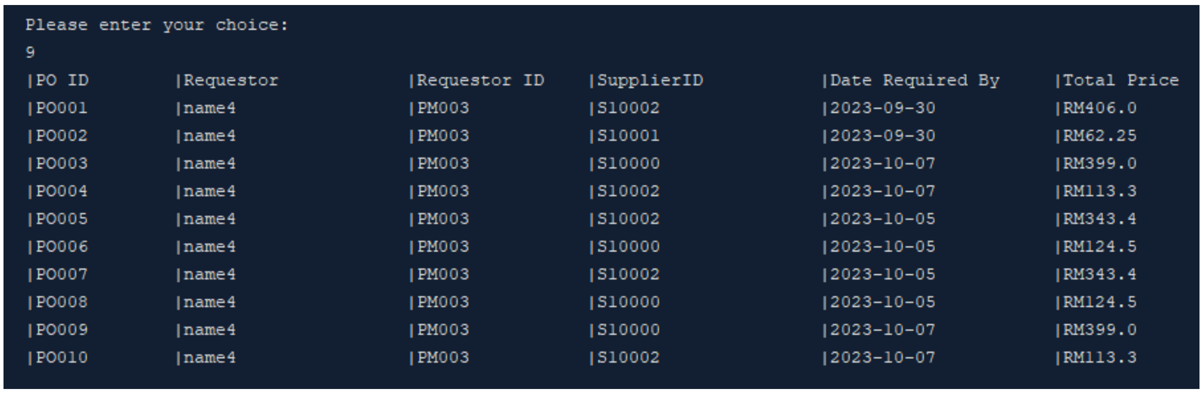


```
Please enter your choice:
8
1. Delete whole PR
2. Delete one item from PR
3. Back

Please enter your choice:
1
|PR ID       |Requestor            |Requestor ID    |Date Required By     |Status          |Total Price
|PR001       |Administrator        |ADMIN           |2023-10-05           |REJECTED        |RM467.9
|PR002       |Administrator        |ADMIN           |2023-10-07           |APPROVED        |RM512.3
|PR003       |name3                |SM003           |2023-09-30           |APPROVED        |RM400.0
|PR004       |name3                |SM003           |2023-10-30           |CANCELLED       |RM565.4
|PR005       |name3                |SM003           |2023-09-30           |APPROVED        |RM206.0
|PR006       |name3                |SM003           |2023-09-30           |APPROVED        |RM468.25
|PR007       |name3                |SM003           |2023-09-29           |REJECTED        |RM662.4
|PR008       |name3                |SM003           |2023-09-28           |CANCELLED       |RM1245.0
|PR009       |name888              |SM001           |2023-09-26           |CANCELLED       |RM231.20001
Enter the PR ID you would like to delete:
PR009
PR ID: PR009
Requestor: name888
Requestor ID: SM001
Date Required By: 2023-09-26
Status: CANCELLED
Total Price: RM231.20001

ItemList:
|ItemID      |Quantity     |SupplierID   |Total Item Price    |
|I10008      |10           |S10001       |RM231.20001         |
Are you sure you would like to delete PR009? Enter Y if yes and anything else if no.
Y
PR deleted successfully
```

Figure 50: Deleting Whole PR

Figure 34 shows the SM deleting the entire PR. This is done by choosing the PR by entering the matching PR ID, this will make the system display the PR details and the items in it. Then the SM needs to confirm by typing ‘Y’.

List of Purchase Orders

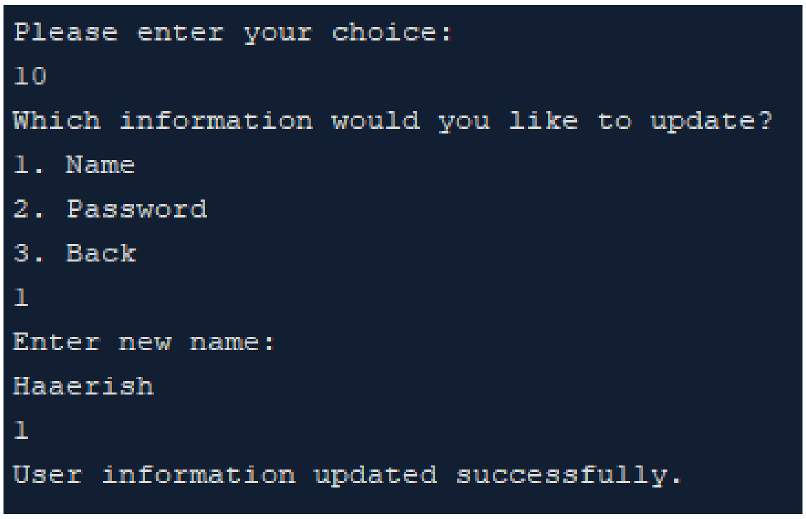


```
Please enter your choice:
9
|PO ID       |Requestor            |Requestor ID    |SupplierID          |Date Required By     |Total Price
|PO001       |name4                |PM003           |S10002              |2023-09-30           |RM406.0
|PO002       |name4                |PM003           |S10001              |2023-09-30           |RM62.25
|PO003       |name4                |PM003           |S10000              |2023-10-07           |RM399.0
|PO004       |name4                |PM003           |S10002              |2023-10-07           |RM113.3
|PO005       |name4                |PM003           |S10002              |2023-10-05           |RM343.4
|PO006       |name4                |PM003           |S10000              |2023-10-05           |RM124.5
|PO007       |name4                |PM003           |S10002              |2023-10-05           |RM343.4
|PO008       |name4                |PM003           |S10000              |2023-10-05           |RM124.5
|PO009       |name4                |PM003           |S10000              |2023-10-07           |RM399.0
|PO010       |name4                |PM003           |S10002              |2023-10-07           |RM113.3
```

Figure 51: List of Purchase Order

If the SM wants, they can also look at all the Purchase Orders that are in the system. This is done by typing ‘9’ in the menu,

Update Profile

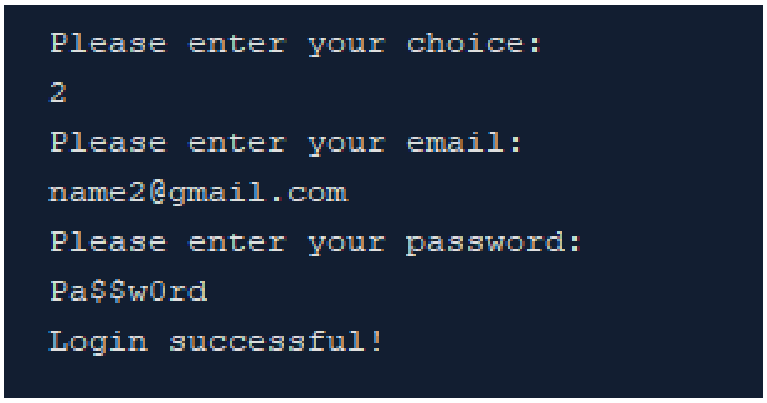


```
Please enter your choice:
10
Which information would you like to update?
1. Name
2. Password
3. Back
1
Enter new name:
Haaerish
1
User information updated successfully.
```

Figure 52: Update Profile

The SM can also change their name or their password as shown above in Figure 36. After all this the SM can also log out of the system by entering ‘11’ in the main menu.

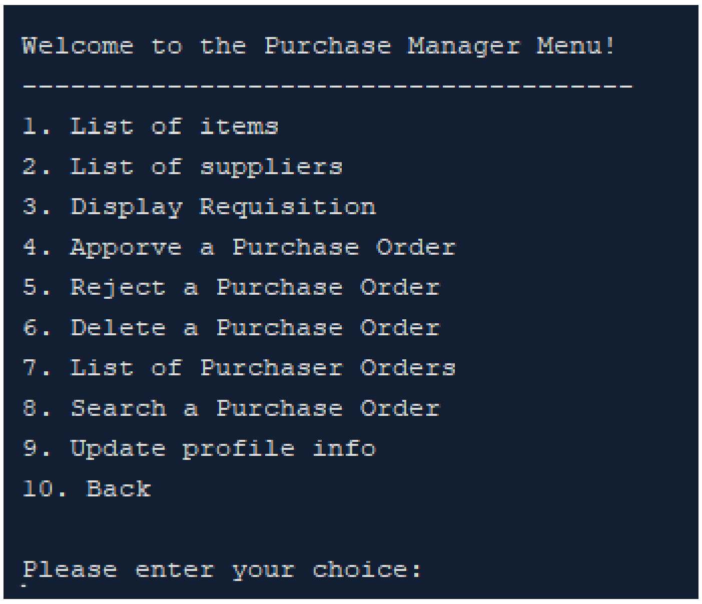
Purchase Manager Functions



```
Please enter your choice:
2
Please enter your email:
name2@gmail.com
Please enter your password:
Pa$$w0rd
Login successful!
```

Figure 53: PM login Info

To Login as Purchase manager the email is name2@gmail.com and the password is Pa$$w0rd.

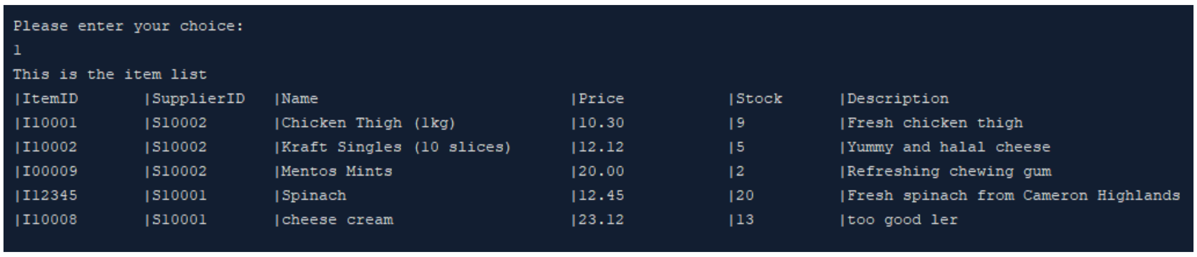


```
Welcome to the Purchase Manager Menu!
--------------------------------------
1. List of items
2. List of suppliers
3. Display Requisition
4. Apporve a Purchase Order
5. Reject a Purchase Order
6. Delete a Purchase Order
7. List of Purchaser Orders
8. Search a Purchase Order
9. Update profile info
10. Back

Please enter your choice:
```

Figure 54: Menu of Purchase Manager

As the Purchase Manager you can perform a total of 9 different task which are: List all the items, List all of the Suppliers, Display all the PR, Approve a PO, Reject a PO, Delete a PO, List of PO, Search a PO, and update Profile info.

List of Items

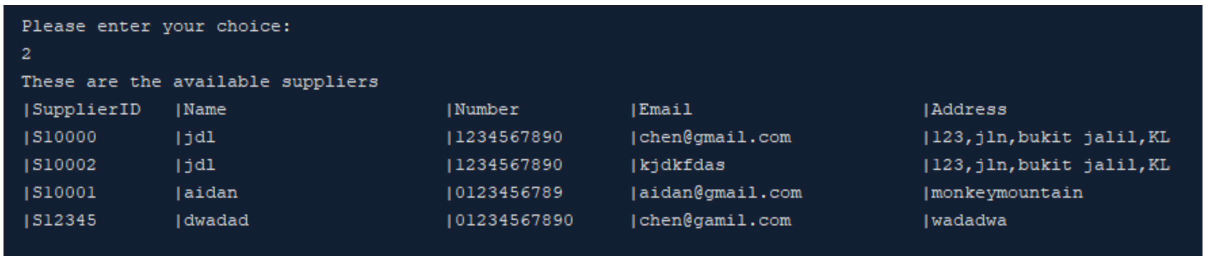


```
Please enter your choice:
1
This is the item list
|ItemID      |SupplierID   |Name                      |Price         |Stock      |Description
|I10001      |S10002       |Chicken Thigh (1kg)       |10.30         |9          |Fresh chicken thigh
|I10002      |S10002       |Kraft Singles (10 slices) |12.12         |5          |Yummy and halal cheese
|I00009      |S10002       |Mentos Mints              |20.00         |2          |Refreshing chewing gum
|I12345      |S10001       |Spinach                   |12.45         |20         |Fresh spinach from Cameron Highlands
|I10008      |S10001       |cheese cream              |23.12         |13         |too good ler
```

Figure 55: List of Items

PM can view the list of Items available and details such as the price and stock by entering ‘1’ in the menu.

List of Suppliers

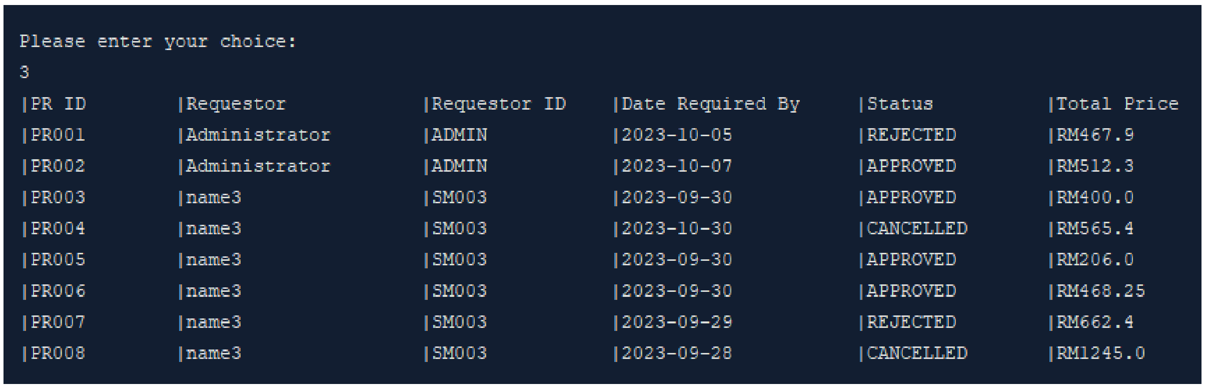


```
Please enter your choice:
2
These are the available suppliers
|SupplierID   |Name                      |Number          |Email                      |Address
|S10000       |jdl                       |1234567890      |chen@gmail.com             |123,jln,bukit jalil,KL
|S10002       |jdl                       |1234567890      |kjdkfdas                   |123,jln,bukit jalil,KL
|S10001       |aidan                     |0123456789      |aidan@gmail.com            |monkeymountain
|S12345       |dwadad                    |01234567890     |chen@gamil.com             |wadadwa
```

Figure 56: List of Supplier

PM can view the list of Suppliers available and details such as the number and email by entering ‘2’ in the menu.

Display Requisition



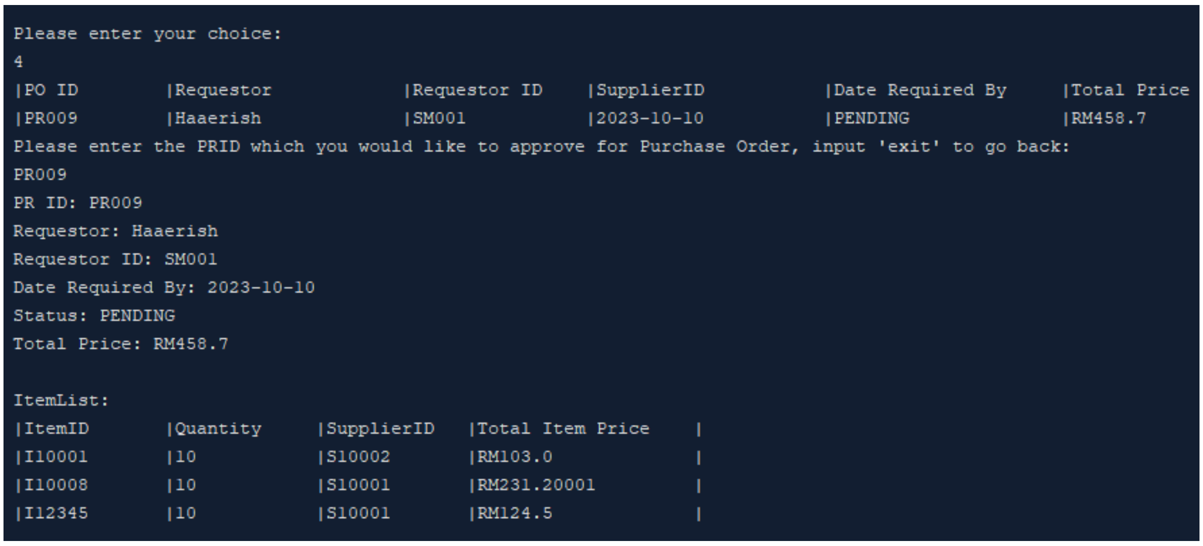
```
Please enter your choice:
3
|PR ID        |Requestor            |Requestor ID     |Date Required By      |Status          |Total Price
|PR001        |Administrator        |ADMIN            |2023-10-05            |REJECTED        |RM467.9
|PR002        |Administrator        |ADMIN            |2023-10-07            |APPROVED        |RM512.3
|PR003        |name3                |SM003            |2023-09-30            |APPROVED        |RM400.0
|PR004        |name3                |SM003            |2023-10-30            |CANCELLED       |RM565.4
|PR005        |name3                |SM003            |2023-09-30            |APPROVED        |RM206.0
|PR006        |name3                |SM003            |2023-09-30            |APPROVED        |RM468.25
|PR007        |name3                |SM003            |2023-09-29            |REJECTED        |RM662.4
|PR008        |name3                |SM003            |2023-09-28            |CANCELLED       |RM1245.0
```

Figure 57: Display all Purchase Requisition

PM can view all the Purchase Requisition and details such as the Date and Status by entering ‘3’ in the menu. This will display all the PR that has been entered into the system even the ones that are rejected and cancelled.

Purchase Requisition

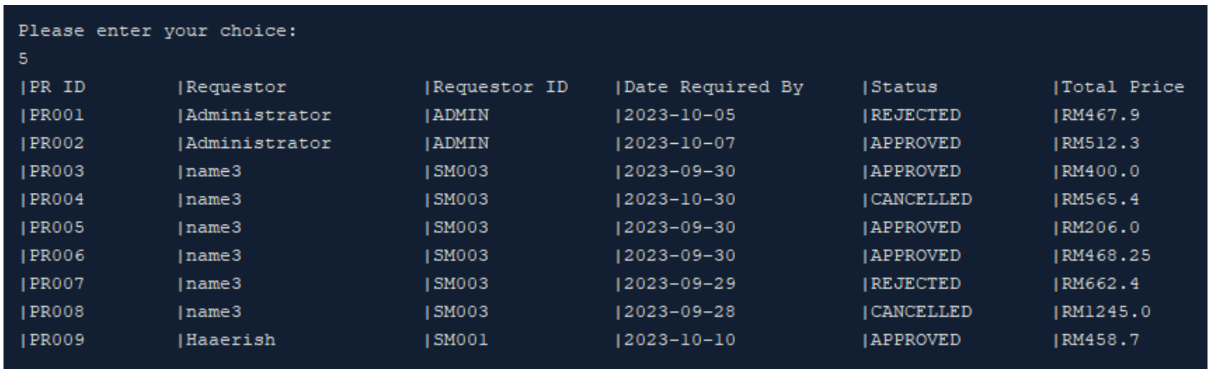
Approve a Purchase Requisition



```
Please enter your choice:
4
|PO ID        |Requestor            |Requestor ID    |SupplierID           |Date Required By     |Total Price
|PR009        |Haaerish             |SM001           |2023-10-10           |PENDING              |RM458.7
Please enter the PRID which you would like to approve for Purchase Order, input 'exit' to go back:
PR009
PR ID: PR009
Requestor: Haaerish
Requestor ID: SM001
Date Required By: 2023-10-10
Status: PENDING
Total Price: RM458.7

ItemList:
|ItemID       |Quantity     |SupplierID   |Total Item Price    |
|I10001       |10           |S10002       |RM103.0             |
|I10008       |10           |S10001       |RM231.20001         |
|I12345       |10           |S10001       |RM124.5             |
```

Figure 58: Approving PR

In figure 58, the PM can approve the Purchase Requisition by entering the Purchase Requisition ID. As for the example above, we have chosen the Purchase Requisition PR009. This will display all the details of that Purchase Requisition and the status of the Purchase Requisition is pending.

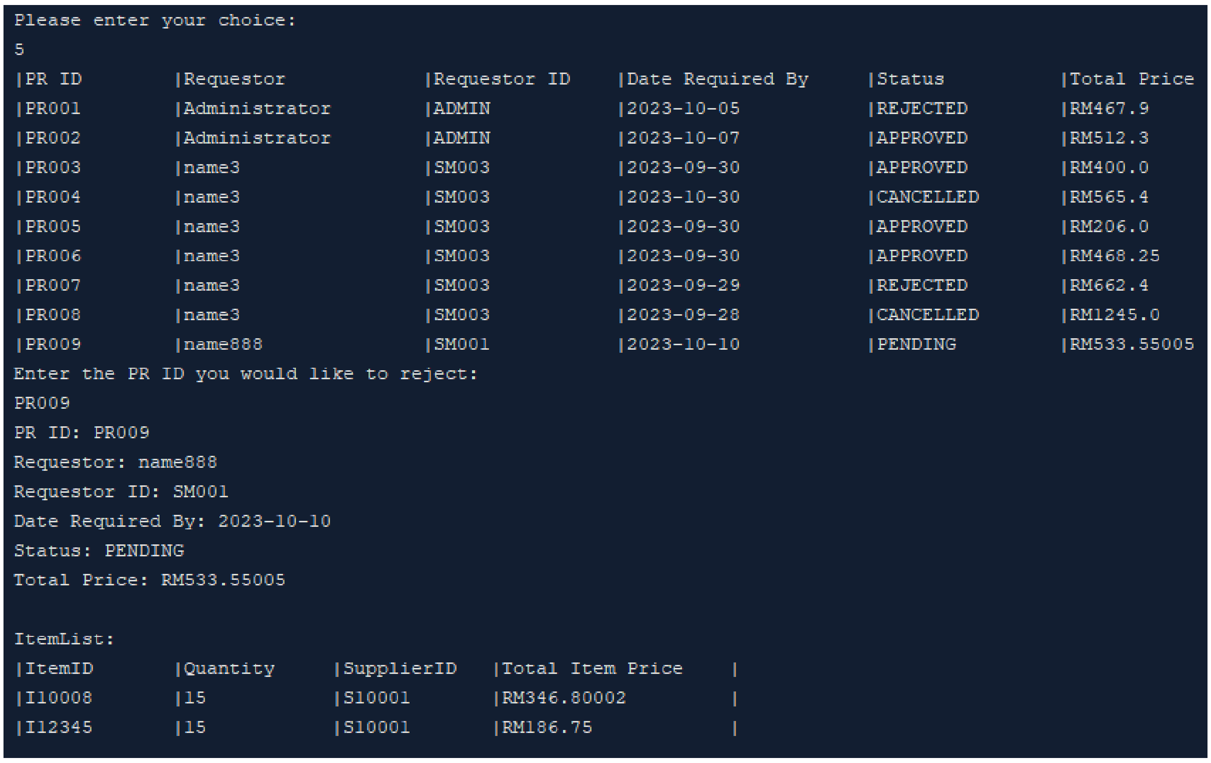


```
Please enter your choice:
5
|PR ID        |Requestor            |Requestor ID    |Date Required By     |Status          |Total Price
|PR001        |Administrator        |ADMIN           |2023-10-05           |REJECTED        |RM467.9
|PR002        |Administrator        |ADMIN           |2023-10-07           |APPROVED        |RM512.3
|PR003        |name3                |SM003           |2023-09-30           |APPROVED        |RM400.0
|PR004        |name3                |SM003           |2023-10-30           |CANCELLED       |RM565.4
|PR005        |name3                |SM003           |2023-09-30           |APPROVED        |RM206.0
|PR006        |name3                |SM003           |2023-09-30           |APPROVED        |RM468.25
|PR007        |name3                |SM003           |2023-09-29           |REJECTED        |RM662.4
|PR008        |name3                |SM003           |2023-09-28           |CANCELLED       |RM1245.0
|PR009        |Haaerish             |SM001           |2023-10-10           |APPROVED        |RM458.7
```

Figure 59: List of PR to show the status.

The figure 59 above shows the status of the Purchase Requisition with the PR ID PR009. As you can see after approving the status has changed from Pending to Approved.

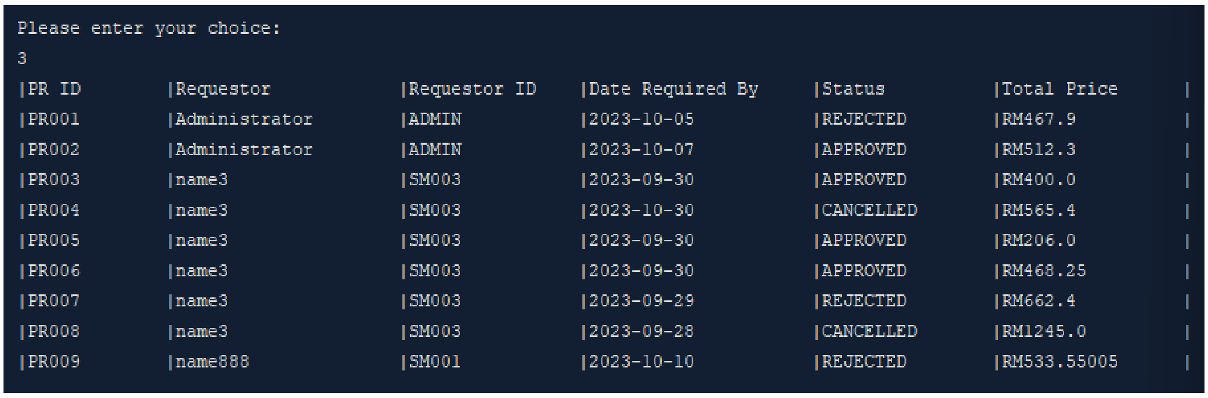
Reject a Purchase Requisition



```
Please enter your choice:
5
|PR ID        |Requestor            |Requestor ID    |Date Required By     |Status          |Total Price
|PR001        |Administrator        |ADMIN           |2023-10-05           |REJECTED        |RM467.9
|PR002        |Administrator        |ADMIN           |2023-10-07           |APPROVED        |RM512.3
|PR003        |name3                |SM003           |2023-09-30           |APPROVED        |RM400.0
|PR004        |name3                |SM003           |2023-10-30           |CANCELLED       |RM565.4
|PR005        |name3                |SM003           |2023-09-30           |APPROVED        |RM206.0
|PR006        |name3                |SM003           |2023-09-30           |APPROVED        |RM468.25
|PR007        |name3                |SM003           |2023-09-29           |REJECTED        |RM662.4
|PR008        |name3                |SM003           |2023-09-28           |CANCELLED       |RM1245.0
|PR009        |name888              |SM001           |2023-10-10           |PENDING         |RM533.55005
Enter the PR ID you would like to reject:
PR009
PR ID: PR009
Requestor: name888
Requestor ID: SM001
Date Required By: 2023-10-10
Status: PENDING
Total Price: RM533.55005

ItemList:
|ItemID       |Quantity     |SupplierID   |Total Item Price    |
|I10008       |15           |S10001       |RM346.80002         |
|I12345       |15           |S10001       |RM186.75            |
```

Figure 60: Reject Purchase Requisition

Same as Approving the PR the PM can also reject the PR. For this the PM must enter the PR ID of the PR that the want to reject. After that the system will display all the details of that Purchase Order.



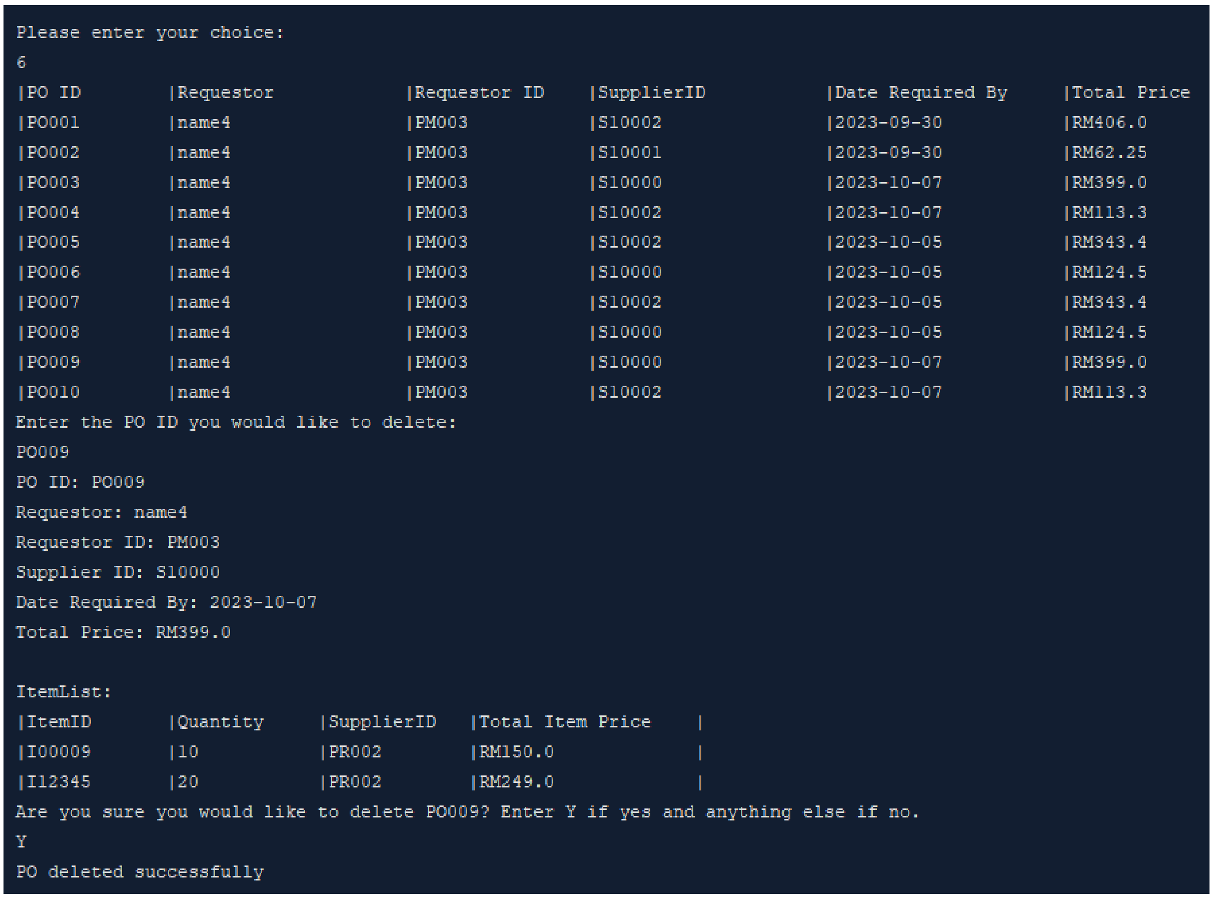
```
Please enter your choice:
3
|PR ID        |Requestor            |Requestor ID    |Date Required By     |Status          |Total Price      |
|PR001        |Administrator        |ADMIN           |2023-10-05           |REJECTED        |RM467.9          |
|PR002        |Administrator        |ADMIN           |2023-10-07           |APPROVED        |RM512.3          |
|PR003        |name3                |SM003           |2023-09-30           |APPROVED        |RM400.0          |
|PR004        |name3                |SM003           |2023-10-30           |CANCELLED       |RM565.4          |
|PR005        |name3                |SM003           |2023-09-30           |APPROVED        |RM206.0          |
|PR006        |name3                |SM003           |2023-09-30           |APPROVED        |RM468.25         |
|PR007        |name3                |SM003           |2023-09-29           |REJECTED        |RM662.4          |
|PR008        |name3                |SM003           |2023-09-28           |CANCELLED       |RM1245.0         |
|PR009        |name888              |SM001           |2023-10-10           |REJECTED        |RM533.55005      |
```

Figure 61: Displaying the PR.

The figure 61 above shows the status of the Purchase Requisition with the PR ID PR009. As you can see after approving the status has changed from Pending to Rejected.

Purchase Order

Delete Purchase Order

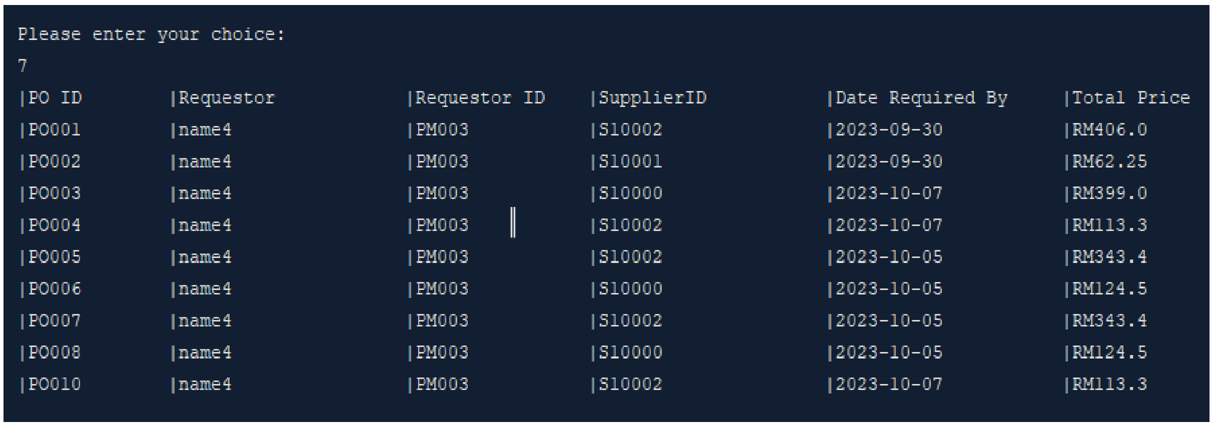


```
Please enter your choice:
6
|PO ID        |Requestor            |Requestor ID     |SupplierID            |Date Required By      |Total Price
|PO001        |name4                |PM003            |S10002                |2023-09-30            |RM406.0
|PO002        |name4                |PM003            |S10001                |2023-09-30            |RM62.25
|PO003        |name4                |PM003            |S10000                |2023-10-07            |RM399.0
|PO004        |name4                |PM003            |S10002                |2023-10-07            |RM113.3
|PO005        |name4                |PM003            |S10002                |2023-10-05            |RM343.4
|PO006        |name4                |PM003            |S10000                |2023-10-05            |RM124.5
|PO007        |name4                |PM003            |S10002                |2023-10-05            |RM343.4
|PO008        |name4                |PM003            |S10000                |2023-10-05            |RM124.5
|PO009        |name4                |PM003            |S10000                |2023-10-07            |RM399.0
|PO010        |name4                |PM003            |S10002                |2023-10-07            |RM113.3
Enter the PO ID you would like to delete:
PO009
PO ID: PO009
Requestor: name4
Requestor ID: PM003
Supplier ID: S10000
Date Required By: 2023-10-07
Total Price: RM399.0

ItemList:
|ItemID       |Quantity     |SupplierID   |Total Item Price    |
|I00009       |10           |PR002        |RM150.0             |
|I12345       |20           |PR002        |RM249.0             |
Are you sure you would like to delete PO009? Enter Y if yes and anything else if no.
Y
PO deleted successfully
```

Figure 62: Delete Purchase Order

Another function that the PM can do is the delete the Purchase Order from the list. For this the PM must enter the PO ID and that will display all the details of the PO and the details of the Items in it. The program will prompt the PM to enter 'Y' if they wish to confirm and delete the chosen PO.

Listing all Purchase Orders

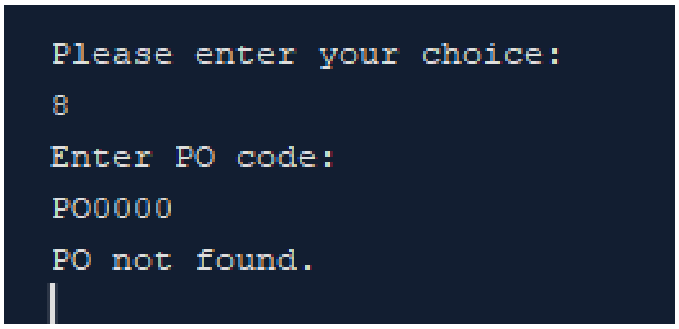


```
Please enter your choice:
7
|PO ID        |Requestor            |Requestor ID   |SupplierID          |Date Required By    |Total Price
|PO001        |name4                |PM003          |S10002              |2023-09-30          |RM406.0
|PO002        |name4                |PM003          |S10001              |2023-09-30          |RM62.25
|PO003        |name4                |PM003          |S10000              |2023-10-07          |RM399.0
|PO004        |name4                |PM003          |S10002              |2023-10-07          |RM113.3
|PO005        |name4                |PM003          |S10002              |2023-10-05          |RM343.4
|PO006        |name4                |PM003          |S10000              |2023-10-05          |RM124.5
|PO007        |name4                |PM003          |S10002              |2023-10-05          |RM343.4
|PO008        |name4                |PM003          |S10000              |2023-10-05          |RM124.5
|PO010        |name4                |PM003          |S10002              |2023-10-07          |RM113.3
```

Figure 63: Listing all PO.

The Purchase manager can view all the Purchase Order that he has generated by approving all the Purchase Requisition. This is the 7th option for the PM.

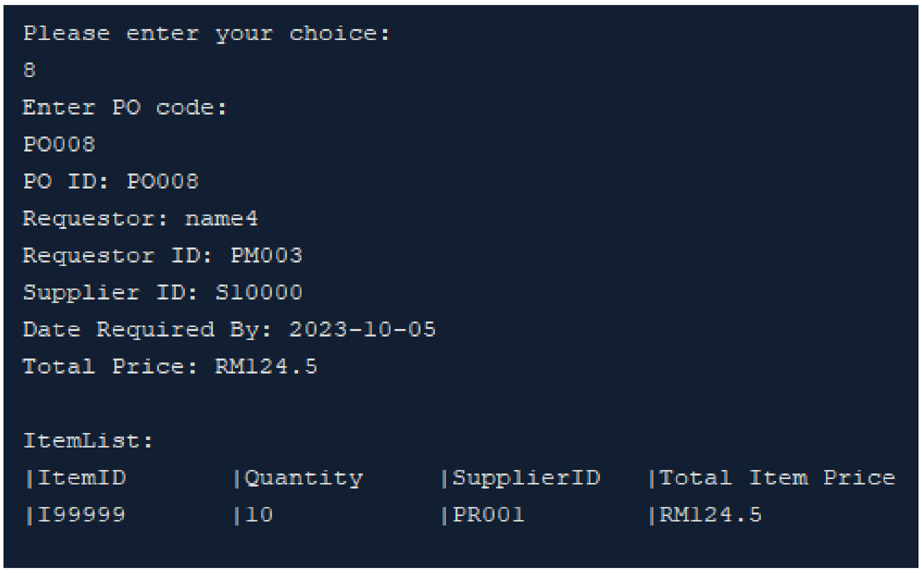
Searching the Purchase Orders



```
Please enter your choice:
8
Enter PO code:
PO0000
PO not found.
```

Figure 64: Searching for wrong PO.

The second last function for the PM is to search the Purchase Orders. This is done by entering the Purchase Order ID. If the ID is not found in the text file an error message will be given out as shown in the image above. If the Id that is entered is valid then the system will display the details of the Purchase Order as shown below and the details of the item in the PO.

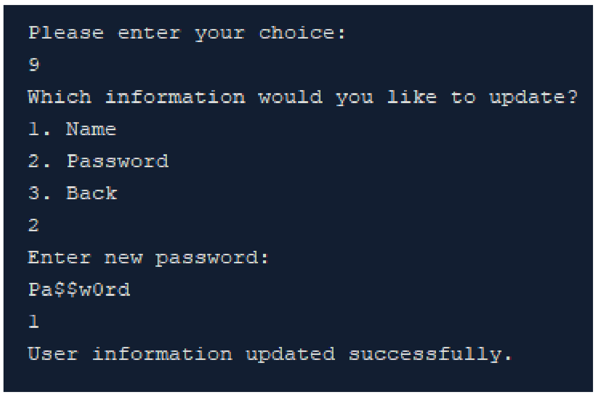


```
Please enter your choice:
8
Enter PO code:
PO008
PO ID: PO008
Requestor: name4
Requestor ID: PM003
Supplier ID: S10000
Date Required By: 2023-10-05
Total Price: RM124.5

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price
|I99999        |10            |PR001         |RM124.5
```

Figure 65: Searching for specific Purchase Order.

If the PO is available in the text file, then the system will detail about the PO and the details of the items inside it.

Update Profile



```
Please enter your choice:
9
Which information would you like to update?
1. Name
2. Password
3. Back
2
Enter new password:
Pa$$w0rd
1
User information updated successfully.
```

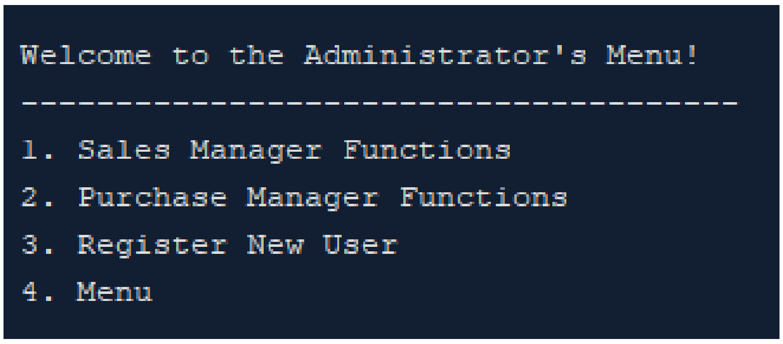
Figure 66: PM Update Profile

As shown in figure 66 the PM can update their profile which is the last function of the PM. The PM can choose to either update their name or their Password which will then be updated into the system.

Admin Functions

To login into the admin role the username is Admin, and the password is Pa$$w0rd.

The admin function mainly consists of 3 different action which are: SM functions, PM function and creating a user.

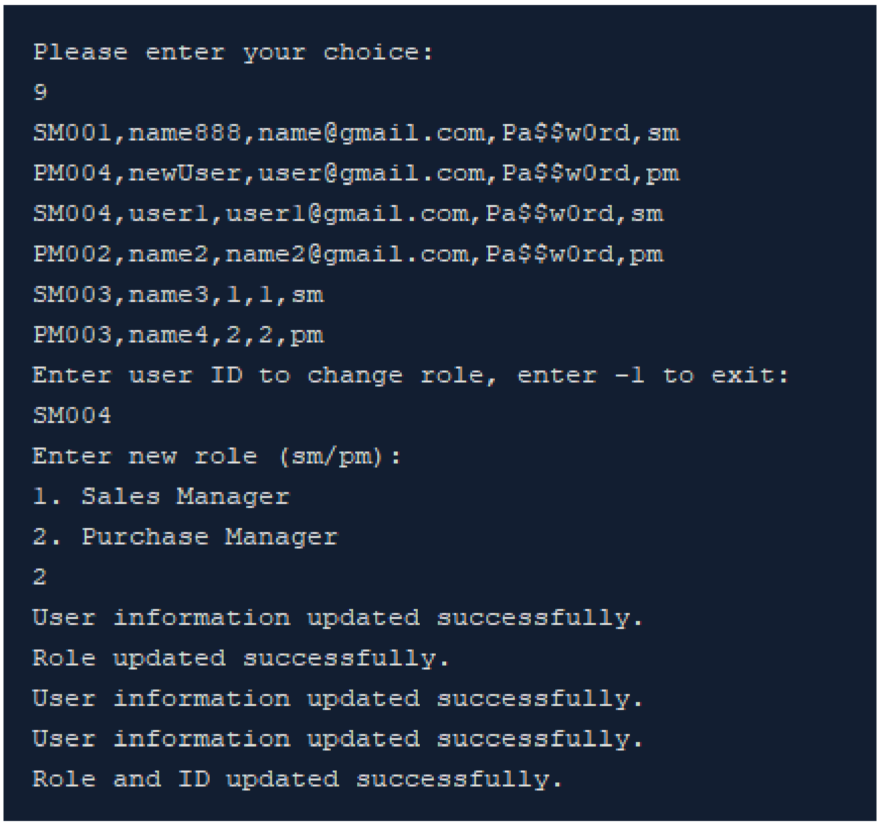


```
Welcome to the Administrator's Menu!
--------------------------------------
1. Sales Manager Functions
2. Purchase Manager Functions
3. Register New User
4. Menu
```

Figure 67: Admin Main Menu

Since we have already covered the SM and the PM features above separately, we won’t be going through them again in the Admin function. Except for some different once such as the Update Profile Info.

Update Profile as Admin.

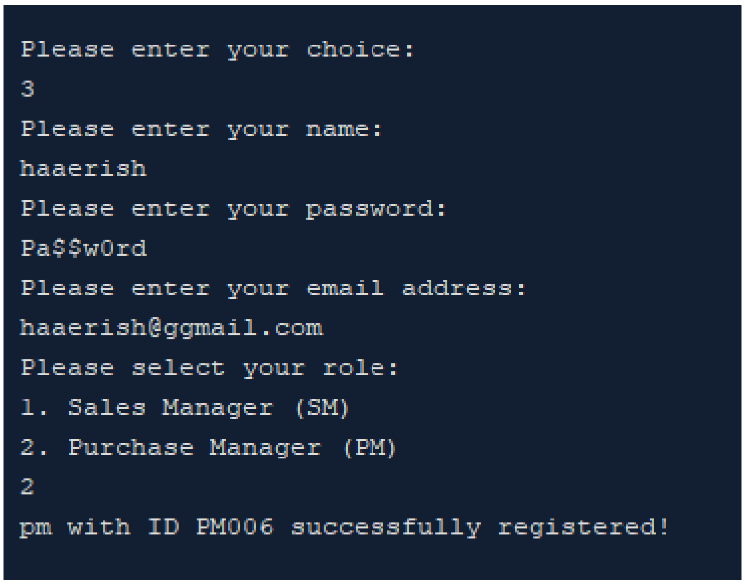


```
Please enter your choice:
9
SM001,name888,name@gmail.com,Pa$$w0rd,sm
PM004,newUser,user@gmail.com,Pa$$w0rd,pm
SM004,user1,user1@gmail.com,Pa$$w0rd,sm
PM002,name2,name2@gmail.com,Pa$$w0rd,pm
SM003,name3,1,1,sm
PM003,name4,2,2,pm
Enter user ID to change role, enter -1 to exit:
SM004
Enter new role (sm/pm):
1. Sales Manager
2. Purchase Manager
2
User information updated successfully.
Role updated successfully.
User information updated successfully.
User information updated successfully.
Role and ID updated successfully.
```

Figure 68: Admin Editing Profile of SM and PM.

The is different form the other user as admin can choose from all the registered users. This is done by entering the user ID then they will have to choose which role they would like to change into. Admin must enter ‘1’ for SM and ‘2’ for PM.

Add user as Admin.

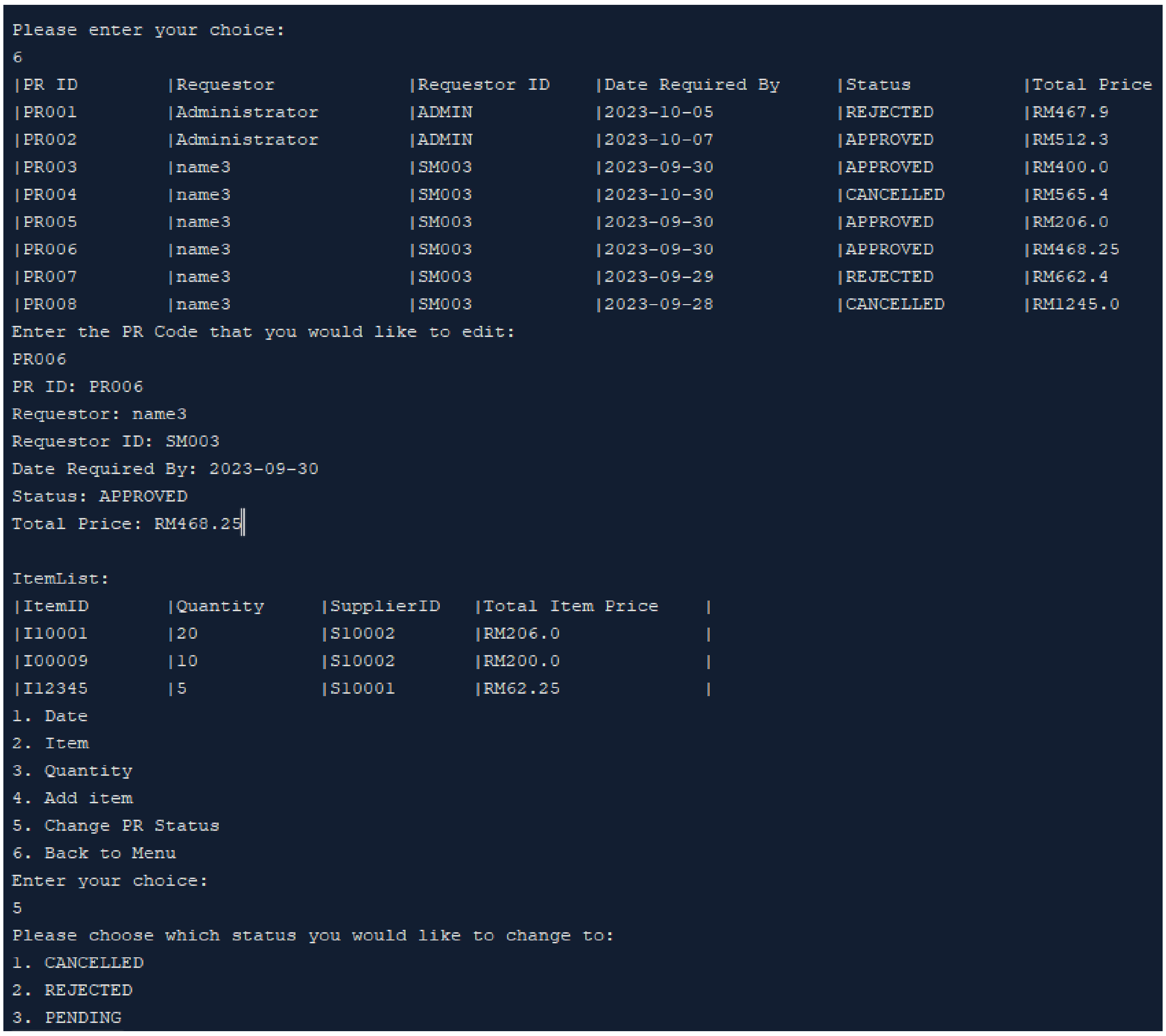


```
Please enter your choice:
3
Please enter your name:
haaerish
Please enter your password:
Pa$$w0rd
Please enter your email address:
haaerish@ggmail.com
Please select your role:
1. Sales Manager (SM)
2. Purchase Manager (PM)
2
pm with ID PM006 successfully registered!
```

Figure 69: Creating new user as admin.

The admin can register a new user as either a sales Manager or as a Purchase Manger. For this the PM must enter the following information: name of user, password of user, email address of user, and the role of the user.

Change PR Status as Admin.



```
Please enter your choice:
6
|PR ID        |Requestor            |Requestor ID    |Date Required By     |Status         |Total Price
|PR001        |Administrator        |ADMIN           |2023-10-05           |REJECTED       |RM467.9
|PR002        |Administrator        |ADMIN           |2023-10-07           |APPROVED       |RM512.3
|PR003        |name3                |SM003           |2023-09-30           |APPROVED       |RM400.0
|PR004        |name3                |SM003           |2023-10-30           |CANCELLED      |RM565.4
|PR005        |name3                |SM003           |2023-09-30           |APPROVED       |RM206.0
|PR006        |name3                |SM003           |2023-09-30           |APPROVED       |RM468.25
|PR007        |name3                |SM003           |2023-09-29           |REJECTED       |RM662.4
|PR008        |name3                |SM003           |2023-09-28           |CANCELLED      |RM1245.0
Enter the PR Code that you would like to edit:
PR006
PR ID: PR006
Requestor: name3
Requestor ID: SM003
Date Required By: 2023-09-30
Status: APPROVED
Total Price: RM468.25

ItemList:
|ItemID       |Quantity     |SupplierID    |Total Item Price    |
|I10001       |20           |S10002        |RM206.0             |
|I00009       |10           |S10002        |RM200.0             |
|I12345       |5            |S10001        |RM62.25             |
1. Date
2. Item
3. Quantity
4. Add item
5. Change PR Status
6. Back to Menu
Enter your choice:
5
Please choose which status you would like to change to:
1. CANCELLED
2. REJECTED
3. PENDING
```

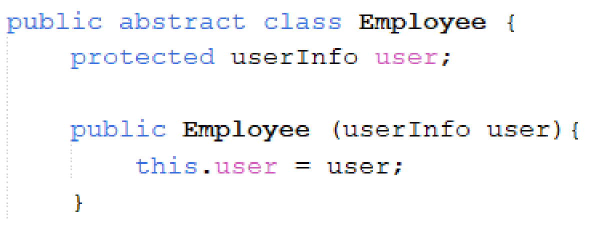
Figure 70: Update PR status as admin.

As shown in figure 70, the admin can choose to change the status of the PR to other status types such as: Cancelled, Rejected, Pending and Approved. Since the PR above already has the status of Approved the system will only allow the admin to change it to the other 3 Status. This is done by typing the PR ID.

5.0 Implementation of Object-Oriented Concept

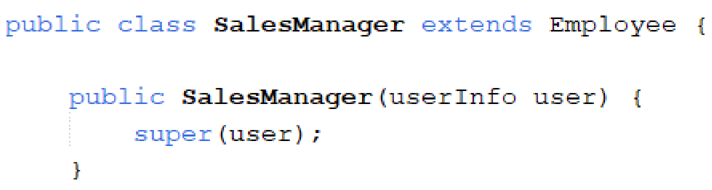
Inheritance

Inheritance in Java is a mechanism through which one object gains access to and inherits all the characteristics and functionalities of a parent object (Inheritance in Java, n.d.). It is an important part of Object-Oriented Programming (OOP) and is implemented using the “extends” keyword. Inheritance is employed when there exists an "is-a" relationship between objects, which is often referred to as a parent-child relationship. The class that inherits attributes and behaviours from another class is called a subclass, while the class from which these features are inherited is known as a superclass, base class, or parent class. This mechanism allows for the creation of new classes that build upon existing ones, and when derived from an existing class, these new classes can utilize the methods and properties inherited from the parent class.



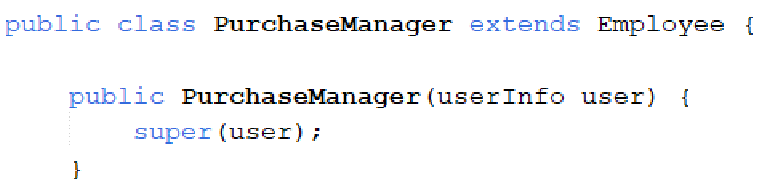
```java
public abstract class Employee {
    protected userInfo user;

    public Employee (userInfo user){
        this.user = user;
    }
```

Figure 71: Employee superclass



```java
public class SalesManager extends Employee {

    public SalesManager(userInfo user) {
        super(user);
    }
```

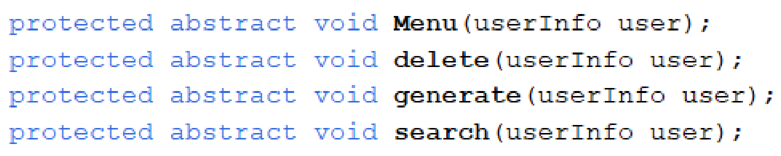
Figure 72: SalesManager subclass extended from Employee superclass



```java
public class PurchaseManager extends Employee {

    public PurchaseManager(userInfo user) {
        super(user);
    }
```

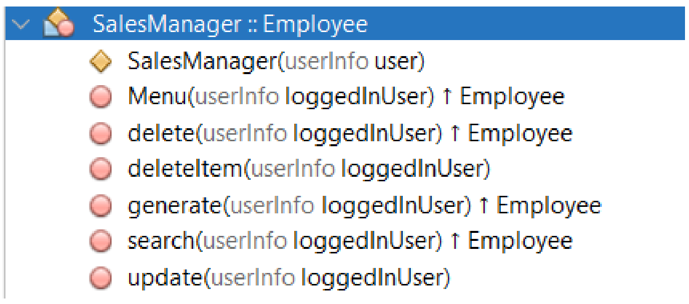
Figure 73: : PurchaseManager subclass extended from Employee superclass

In our case, the Employee class is the superclass, whilst the SalesManager and PurchaseManager classes are the child classes as it inherits attributes, in this case, it’s the userInfo type data, from the Employee superclass using the keyword “super”. When a subclass is created, its constructor has to call the constructor of its parent class. With the super() keyword, which calls the constructor of the parent class. (Oracle, n.d.)



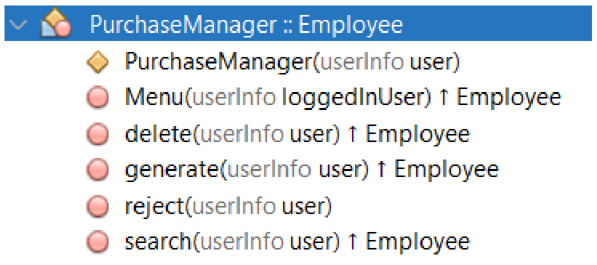
```
protected abstract void Menu(userInfo user);
protected abstract void delete(userInfo user);
protected abstract void generate(userInfo user);
protected abstract void search(userInfo user);
```

Figure 74: Abstract methods in Employee superclass



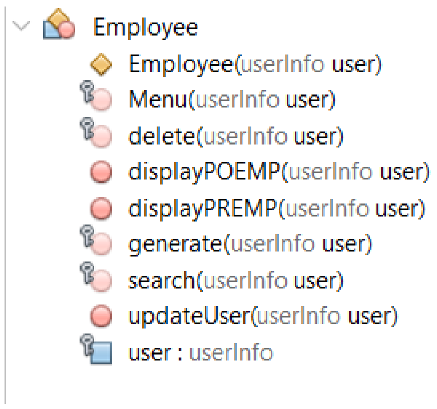
```
SalesManager :: Employee
    SalesManager(userInfo user)
    Menu(userInfo loggedInUser) ↑ Employee
    delete(userInfo loggedInUser) ↑ Employee
    deleteItem(userInfo loggedInUser)
    generate(userInfo loggedInUser) ↑ Employee
    search(userInfo loggedInUser) ↑ Employee
    update(userInfo loggedInUser)
```

Figure 75: SalesManager inherits specific methods from Employee



```
PurchaseManager :: Employee
    PurchaseManager(userInfo user)
    Menu(userInfo loggedInUser) ↑ Employee
    delete(userInfo user) ↑ Employee
    generate(userInfo user) ↑ Employee
    reject(userInfo user)
    search(userInfo user) ↑ Employee
```

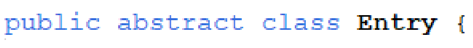
Figure 76: PurchaseManager inherits specific methods from Employee



```
Employee
    Employee(userInfo user)
    Menu(userInfo user)
    delete(userInfo user)
    displayPOEMP(userInfo user)
    displayPREMP(userInfo user)
    generate(userInfo user)
    search(userInfo user)
    updateUser(userInfo user)
    user : userInfo
```

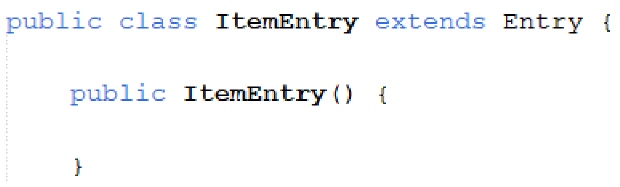
Figure 77: Employee superclass methods

Additionally, the SalesManager and PurchaseManager classes also inherit the abstract methods delete(), generate(), search() and Menu(), from the Employee superclass, as shown in the figures above. displayPOEMP(), displayPREMP(), updateUser() is the concrete method that inherit by the subclasses.



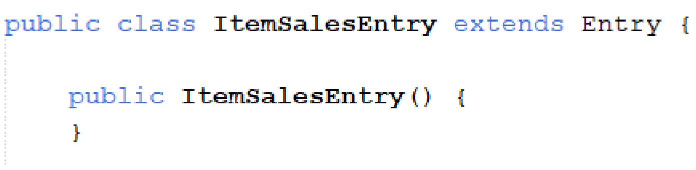
```
public abstract class Entry {
```

Figure 78: Entry superclass



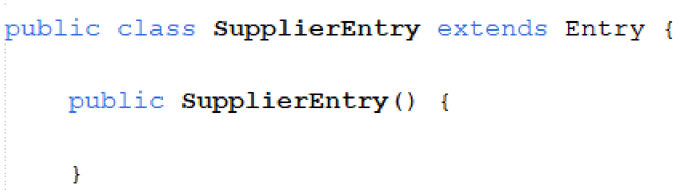
```
public class ItemEntry extends Entry {

    public ItemEntry() {

    }
```

Figure 79: ItemEntry class extended from Entry superclass



```
public class ItemSalesEntry extends Entry {

    public ItemSalesEntry() {
    }
```

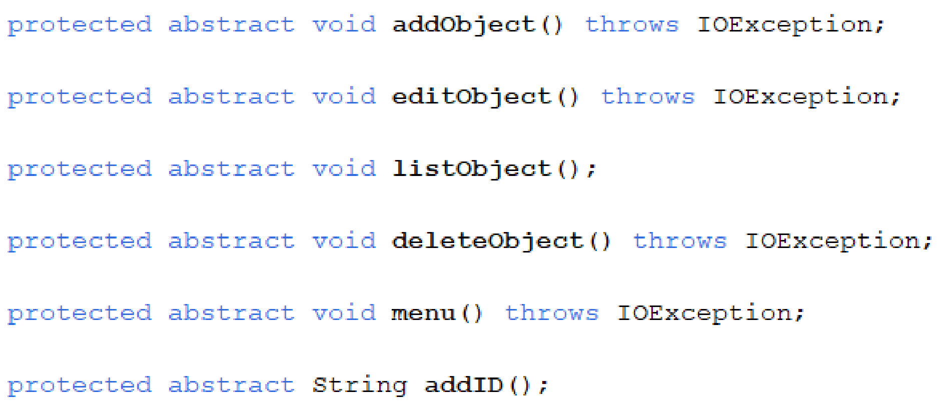
Figure 80: ItemSalesEntry extended from Entry superclass



```
public class SupplierEntry extends Entry {

    public SupplierEntry() {

    }
```

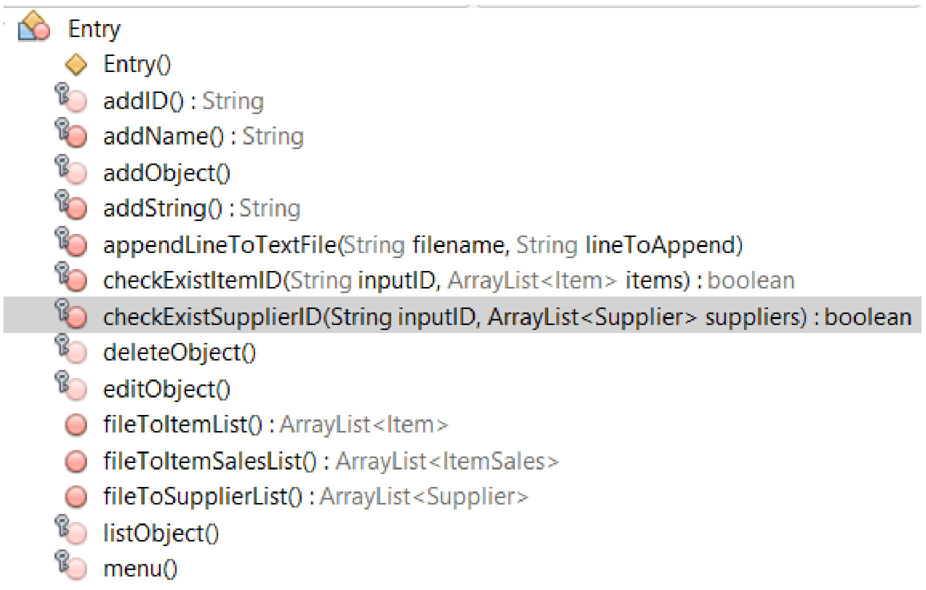
Figure 81: SupplierEntry class extended from Entry superclass

Another implementation of inheritance in our code is with the Entry class. The Entry class acts as the parent class, whilst the ItemEntry, ItemSalesEntry and SupplierEntry classes act as the child classes in this case.



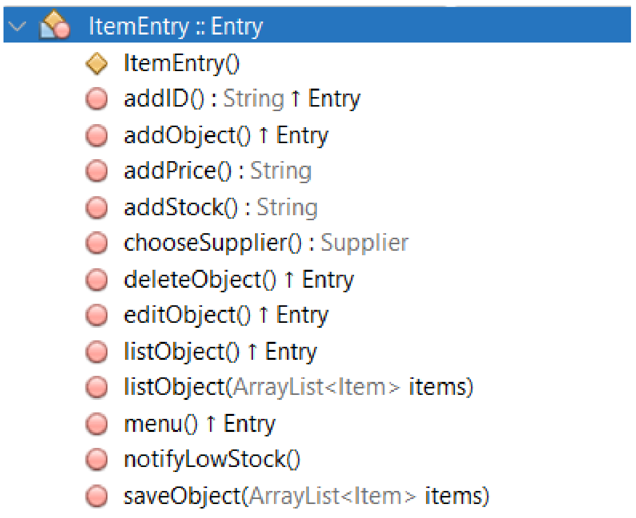
```
protected abstract void addObject() throws IOException;

protected abstract void editObject() throws IOException;

protected abstract void listObject();

protected abstract void deleteObject() throws IOException;

protected abstract void menu() throws IOException;

protected abstract String addID();
```

Figure 82: Methods to be inherited from Entry superclass



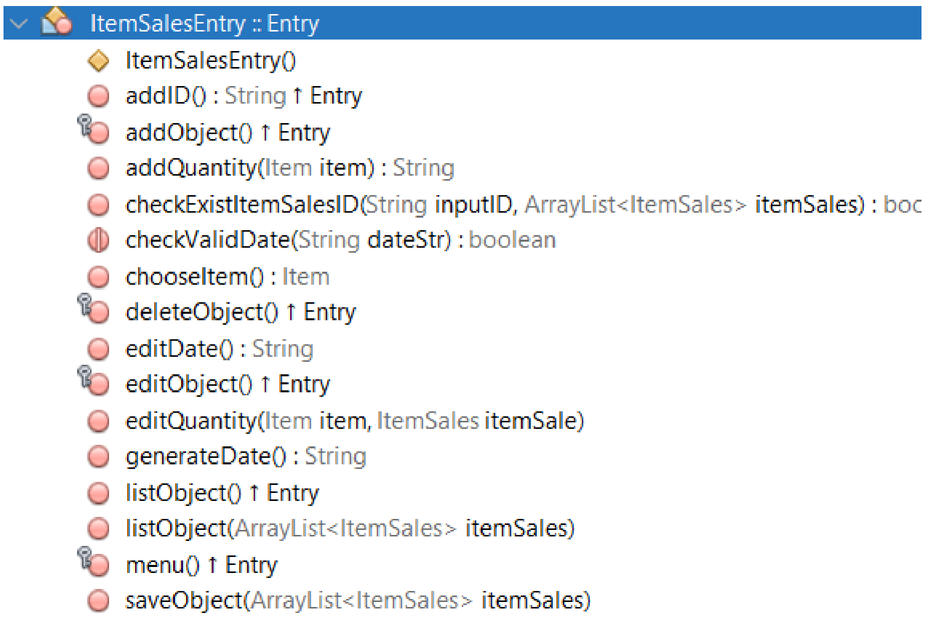
```
Entry
    Entry()
    addID() : String
    addName() : String
    addObject()
    addString() : String
    appendLineToTextFile(String filename, String lineToAppend)
    checkExistItemID(String inputID, ArrayList<Item> items) : boolean
    checkExistSupplierID(String inputID, ArrayList<Supplier> suppliers) : boolean
    deleteObject()
    editObject()
    fileToItemList() : ArrayList<Item>
    fileToItemSalesList() : ArrayList<ItemSales>
    fileToSupplierList() : ArrayList<Supplier>
    listObject()
    menu()
```

Figure 83: Entry superclass methods



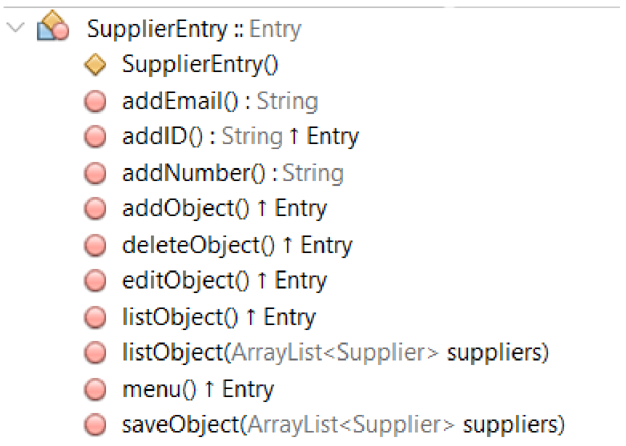
```
ItemEntry :: Entry
    ItemEntry()
    addID() : String ↑ Entry
    addObject() ↑ Entry
    addPrice() : String
    addStock() : String
    chooseSupplier() : Supplier
    deleteObject() ↑ Entry
    editObject() ↑ Entry
    listObject() ↑ Entry
    listObject(ArrayList<Item> items)
    menu() ↑ Entry
    notifyLowStock()
    saveObject(ArrayList<Item> items)
```

Figure 84: ItemEntry inherits specific methods from Entry superclass



```
ItemSalesEntry :: Entry
    ItemSalesEntry()
    addID() : String ↑ Entry
    addObject() ↑ Entry
    addQuantity(Item item) : String
    checkExistItemSalesID(String inputID, ArrayList<ItemSales> itemSales) : boo
    checkValidDate(String dateStr) : boolean
    chooseItem() : Item
    deleteObject() ↑ Entry
    editDate() : String
    editObject() ↑ Entry
    editQuantity(Item item, ItemSales itemSale)
    generateDate() : String
    listObject() ↑ Entry
    listObject(ArrayList<ItemSales> itemSales)
    menu() ↑ Entry
    saveObject(ArrayList<ItemSales> itemSales)
```

Figure 85: ItemSalesEntry class inherits specific methods from Entry superclass



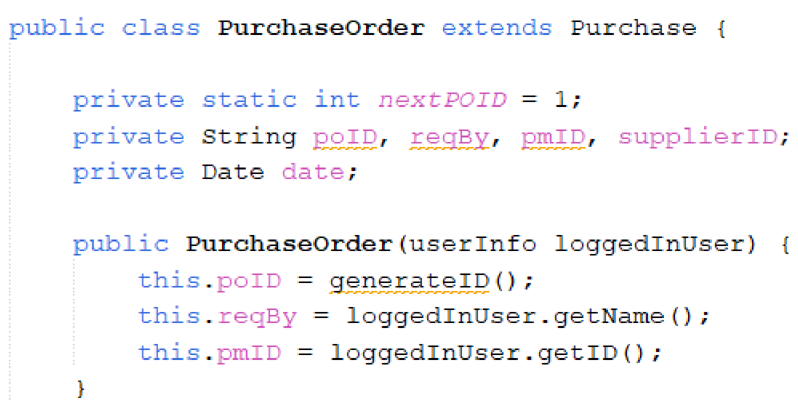
```
SupplierEntry :: Entry
    SupplierEntry()
    addEmail() : String
    addID() : String ↑ Entry
    addNumber() : String
    addObject() ↑ Entry
    deleteObject() ↑ Entry
    editObject() ↑ Entry
    listObject() ↑ Entry
    listObject(ArrayList<Supplier> suppliers)
    menu() ↑ Entry
    saveObject(ArrayList<Supplier> suppliers)
```

Figure 86: SupplierEntry class inherits specific methods from Entry superclass

All of the subclasses inherit the abstract classes such as addID(), addObject(), deleteObject(), editObject(), listObject() and menu() methods from the Entry superclass. Moveover, the concrete classes addName(), addString(), appendLineTOtextFile(), checkExistItemID(), checkExistSupplierID(), fileToItemList(), fileToItemSalesList(), fileToSupplierList() also been inherit by the subclasses. This function is commonly used in three subclasses so in order to reuse it we have put it into the superclass. Also, the entry class also logically related to the three entry classes since they have same common purpose which is entry object into the system.

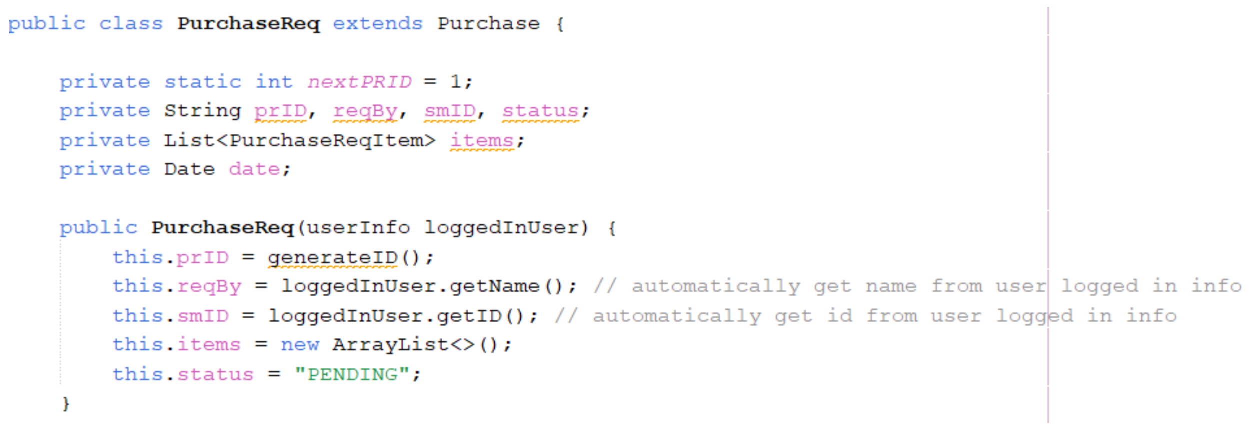
```
public abstract class Purchase {
```

Figure 87: Purchase superclass



```
public class PurchaseOrder extends Purchase {

    private static int nextPOID = 1;
    private String poID, reqBy, pmID, supplierID;
    private Date date;

    public PurchaseOrder(userInfo loggedInUser) {
        this.poID = generateID();
        this.reqBy = loggedInUser.getName();
        this.pmID = loggedInUser.getID();
    }
```

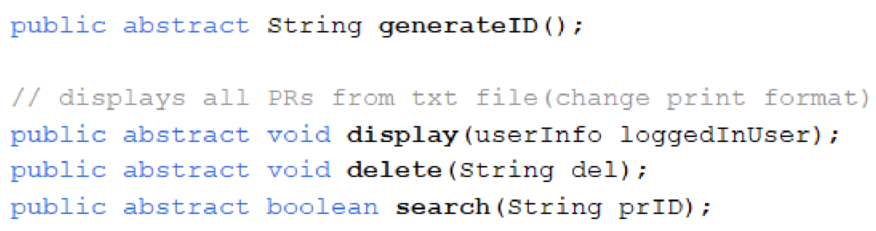
Figure 88: PurchaseOrder extended from Purchase superclass



```
public class PurchaseReq extends Purchase {

    private static int nextPRID = 1;
    private String prID, reqBy, smID, status;
    private List<PurchaseReqItem> items;
    private Date date;

    public PurchaseReq(userInfo loggedInUser) {
        this.prID = generateID();
        this.reqBy = loggedInUser.getName(); // automatically get name from user logged in info
        this.smID = loggedInUser.getID(); // automatically get id from user logged in info
        this.items = new ArrayList<>();
        this.status = "PENDING";
    }
```

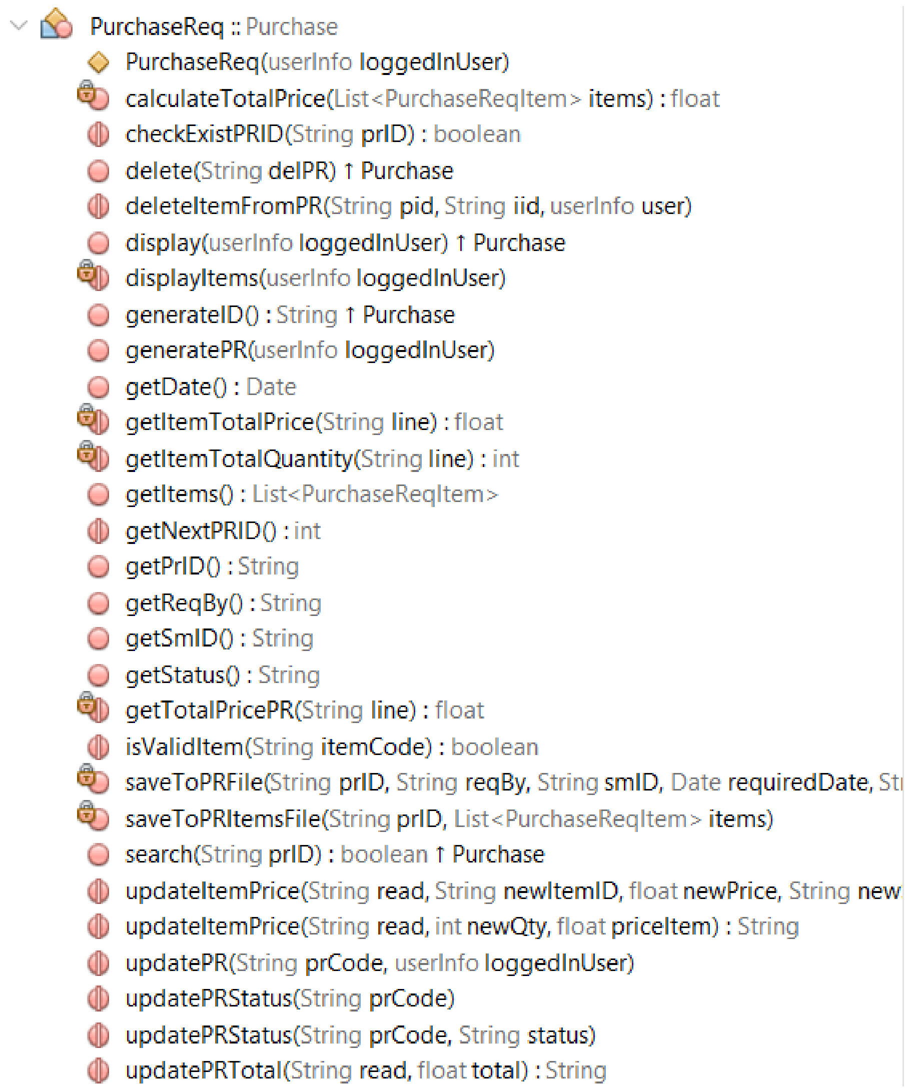
Figure 89: PurchaseReq extended from Purchase superclass

The third and final implementation of inheritance in our code is the Purchase superclass, which extends to the PurchaseReq and PurchaseOrder class for them to act as subclasses.



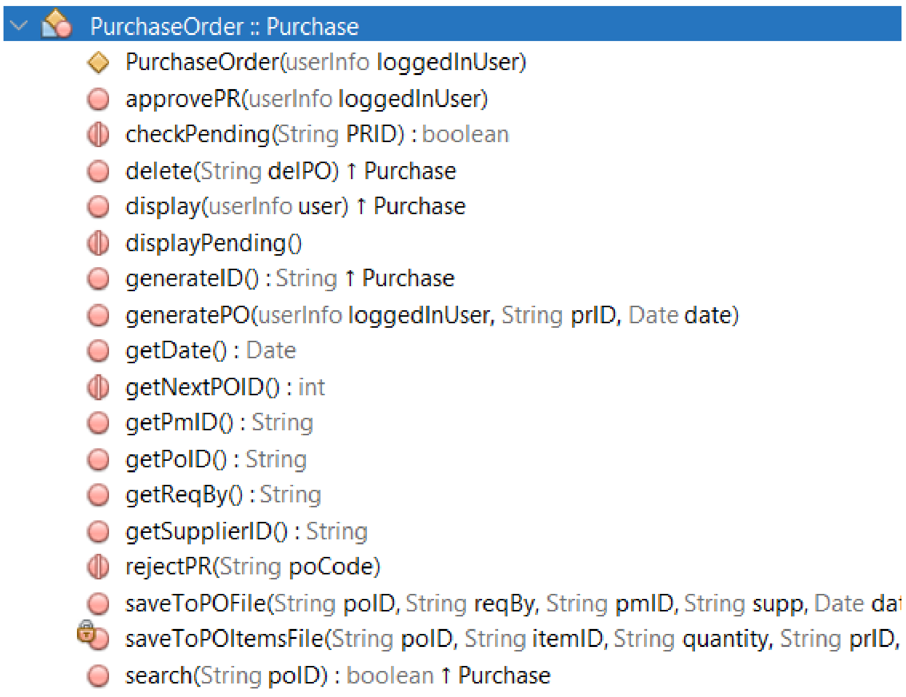
```
public abstract String generateID();

// displays all PRs from txt file(change print format)
public abstract void display(userInfo loggedInUser);
public abstract void delete(String del);
public abstract boolean search(String prID);
```

Figure 90: Methods to be inherited from Purchase superclass



```
PurchaseReq :: Purchase
    PurchaseReq(userInfo loggedInUser)
    calculateTotalPrice(List<PurchaseReqItem> items) : float
    checkExistPRID(String prID) : boolean
    delete(String delPR) ↑ Purchase
    deleteItemFromPR(String pid, String iid, userInfo user)
    display(userInfo loggedInUser) ↑ Purchase
    displayItems(userInfo loggedInUser)
    generateID() : String ↑ Purchase
    generatePR(userInfo loggedInUser)
    getDate() : Date
    getItemTotalPrice(String line) : float
    getItemTotalQuantity(String line) : int
    getItems() : List<PurchaseReqItem>
    getNextPRID() : int
    getPrID() : String
    getReqBy() : String
    getSmID() : String
    getStatus() : String
    getTotalPricePR(String line) : float
    isValidItem(String itemCode) : boolean
    saveToPRFile(String prID, String reqBy, String smID, Date requiredDate, St
    saveToPRItemsFile(String prID, List<PurchaseReqItem> items)
    search(String prID) : boolean ↑ Purchase
    updateItemPrice(String read, String newItemID, float newPrice, String new
    updateItemPrice(String read, int newQty, float priceItem) : String
    updatePR(String prCode, userInfo loggedInUser)
    updatePRStatus(String prCode)
    updatePRStatus(String prCode, String status)
    updatePRTotal(String read, float total) : String
```

Figure 91: Specific methods are inherited from Purchase superclass in PurchaseReq class



PurchaseOrder :: Purchase
PurchaseOrder(userInfo loggedInUser)
approvePR(userInfo loggedInUser)
checkPending(String PRID) : boolean
delete(String delPO) ↑ Purchase
display(userInfo user) ↑ Purchase
displayPending()
generateID() : String ↑ Purchase
generatePO(userInfo loggedInUser, String prID, Date date)
getDate() : Date
getNextPOID() : int
getPmID() : String
getPoID() : String
getReqBy() : String
getSupplierID() : String
rejectPR(String poCode)
saveToPOFile(String poID, String reqBy, String pmID, String supp, Date dat
saveToPOItemsFile(String poID, String itemID, String quantity, String prID,
search(String poID) : boolean ↑ Purchase

Figure 92: Specific methods are inherited from Purchase superclass in PurchaseOrder class

Both of the subclasses inherit the abstract class generateID(), display(), delete() and search() methods from the Purchase class. The findItem(), directUser(), isValidDate() also commonly used in these two classes.

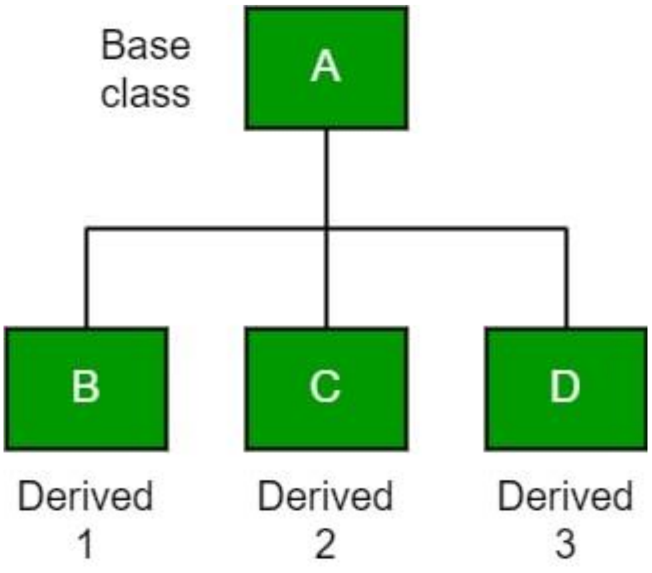


Figure 93: Structure of hierarchal inheritance from GeeksforGeeks.org (GeeksforGeeks.org)

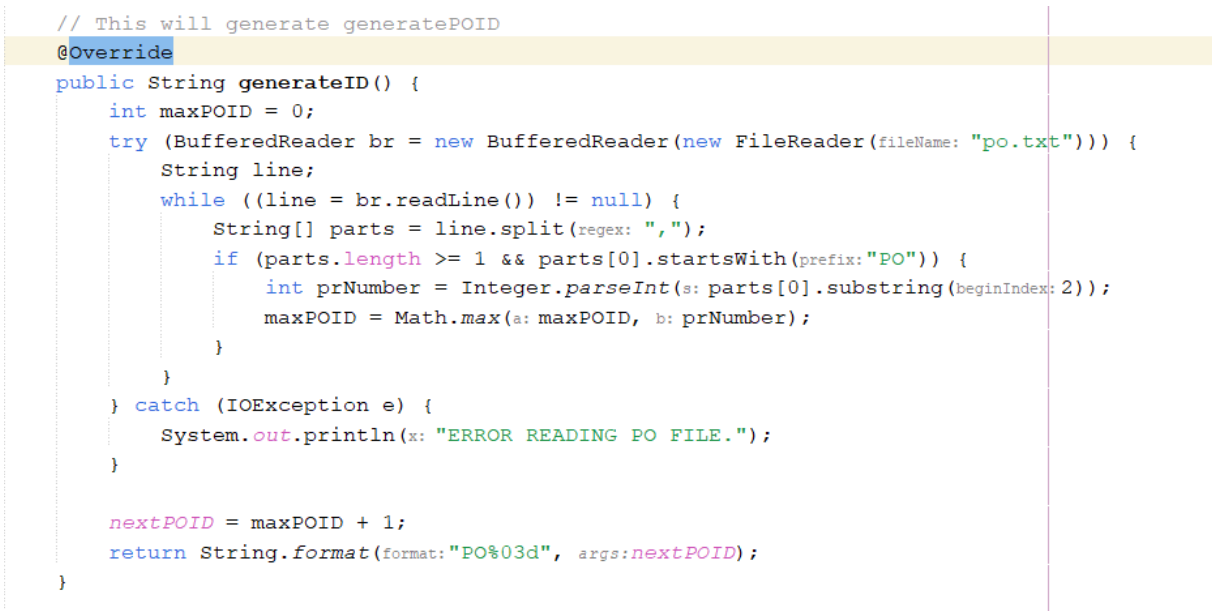
All of the implementations of code inheritance in our program are known as a Hierarchical Inheritance, which is defined as when one class acts as a superclass for more than one subclass (GeeksforGeeks, 2023).

Polymorphism

Polymorphism in Java refers to the capability of objects to take on multiple forms. This concept extends to methods, where methods of the same name can perform distinct operations depending on the specific object they are called upon (Debnath, Diving Deeper into Polymorphism and its Benefits in Java, 2019).

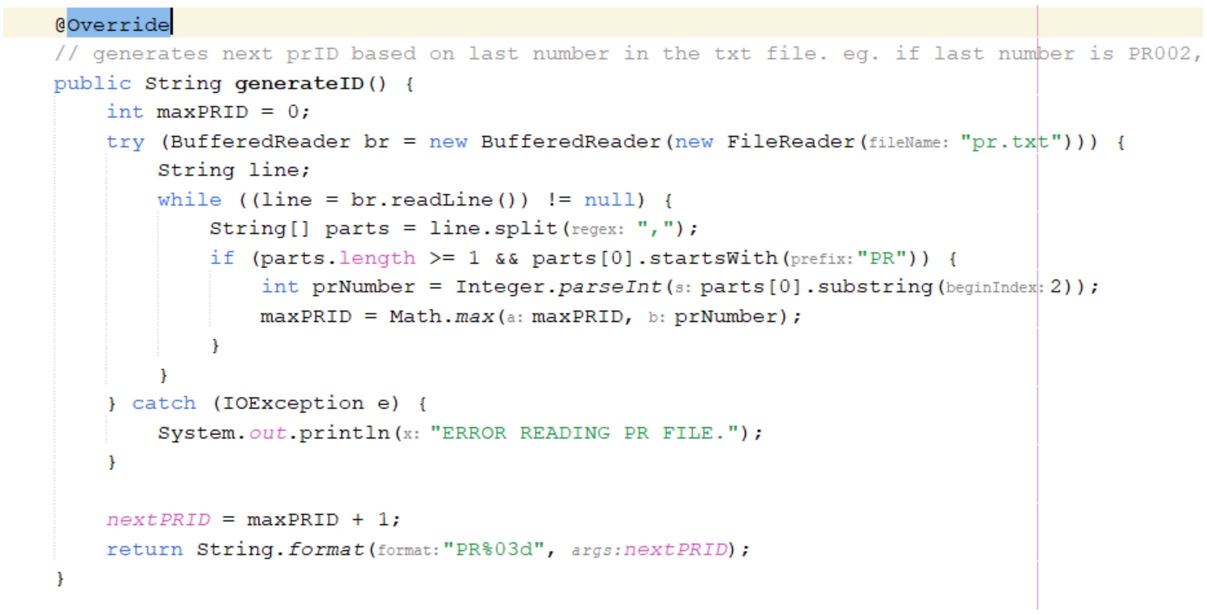
Overriding

In our code, this can be shown by method overriding. This is when the subclass inherits the method from its superclass, then provides its own implementation (Programiz, n.d.). This was achieved by allowing subclasses to inherit methods from their superclass and subsequently replacing or customizing them within the specific subclass using the "@Override" annotation. This practice, referred to as method overriding, is also known as run-time polymorphism (Vadapalli, 2022).



```java
// This will generate generatePOID
@Override
public String generateID() {
    int maxPOID = 0;
    try (BufferedReader br = new BufferedReader(new FileReader(fileName: "po.txt"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] parts = line.split(regex: ",");
            if (parts.length >= 1 && parts[0].startsWith(prefix:"PO")) {
                int prNumber = Integer.parseInt(s: parts[0].substring(beginIndex: 2));
                maxPOID = Math.max(a: maxPOID, b: prNumber);
            }
        }
    } catch (IOException e) {
        System.out.println(x: "ERROR READING PO FILE.");
    }

    nextPOID = maxPOID + 1;
    return String.format(format:"PO%03d", args:nextPOID);
}
```

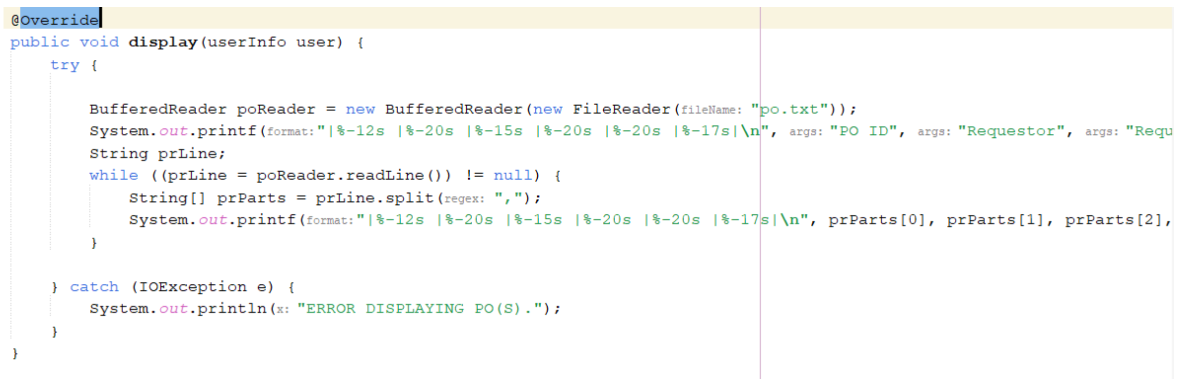
Figure 94: Method overriding in PurchaseOrder class for generateID()



```
@Override
// generates next prID based on last number in the txt file. eg. if last number is PR002,
public String generateID() {
    int maxPRID = 0;
    try (BufferedReader br = new BufferedReader(new FileReader(fileName: "pr.txt"))) {
        String line;
        while ((line = br.readLine()) != null) {
            String[] parts = line.split(regex: ",");
            if (parts.length >= 1 && parts[0].startsWith(prefix:"PR")) {
                int prNumber = Integer.parseInt(s: parts[0].substring(beginIndex: 2));
                maxPRID = Math.max(a: maxPRID, b: prNumber);
            }
        }
    } catch (IOException e) {
        System.out.println(x: "ERROR READING PR FILE.");
    }

    nextPRID = maxPRID + 1;
    return String.format(format:"PR%03d", args:nextPRID);
}
```

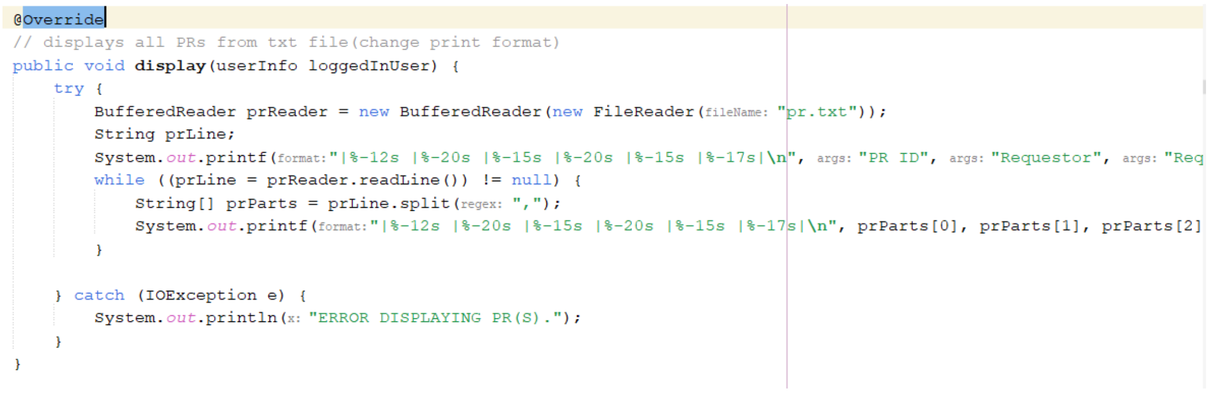
Figure 95: Method overriding in PurchaseReq class for generateID()

In the figures shown above, the first figure depicts the generateID() method in the PurchaseOrder class, which was inherited from the Purchase superclass. However, in the second figure, despite having the same method name and also inheriting it from the Purchase superclass, the generateID () method in the PurchaseReq class has different contents. This means that if the system utilizes the specific class, and if it employs method overriding, then the implementation of that method in that specific class will be used.



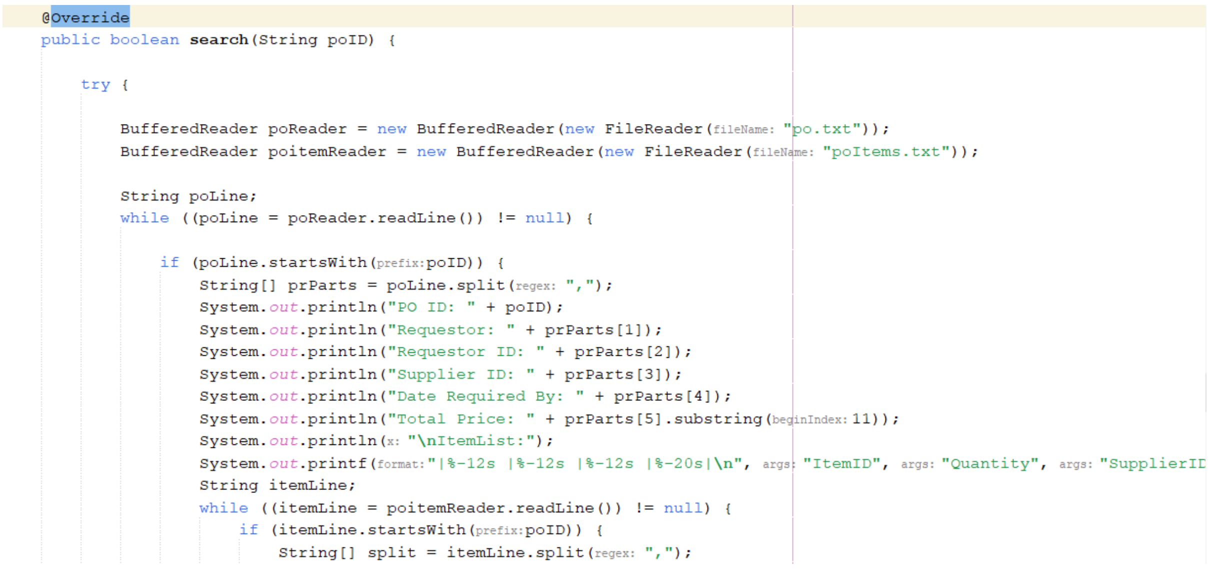
```
@Override
public void display(userInfo user) {
    try {

        BufferedReader poReader = new BufferedReader(new FileReader(fileName: "po.txt"));
        System.out.printf(format:"|%-12s |%-20s |%-15s |%-20s |%-20s |%-17s|\n", args: "PO ID", args: "Requestor", args: "Requ
        String prLine;
        while ((prLine = poReader.readLine()) != null) {
            String[] prParts = prLine.split(regex: ",");
            System.out.printf(format:"|%-12s |%-20s |%-15s |%-20s |%-20s |%-17s|\n", prParts[0], prParts[1], prParts[2],
        }

    } catch (IOException e) {
        System.out.println(x: "ERROR DISPLAYING PO(S).");
    }
}
```

Figure 96: Method overriding in PurchaseOrder class for display()



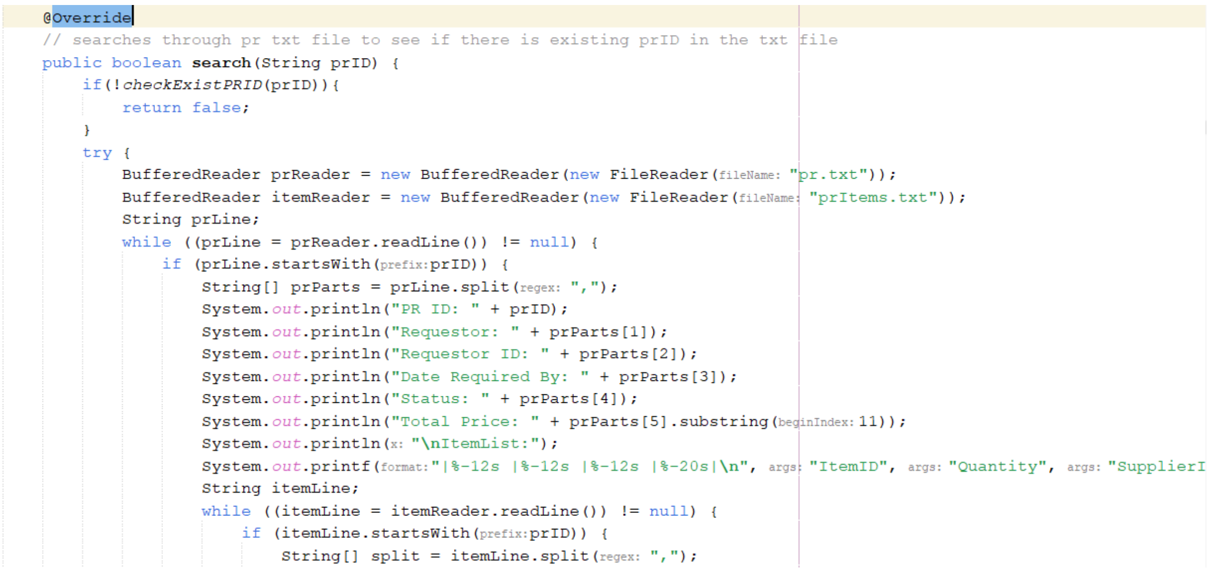
```
@Override
// displays all PRs from txt file(change print format)
public void display(userInfo loggedInUser) {
    try {
        BufferedReader prReader = new BufferedReader(new FileReader(fileName: "pr.txt"));
        String prLine;
        System.out.printf(format:"|%-12s |%-20s |%-15s |%-20s |%-15s |%-17s|\n", args: "PR ID", args: "Requestor", args: "Req
        while ((prLine = prReader.readLine()) != null) {
            String[] prParts = prLine.split(regex: ",");
            System.out.printf(format:"|%-12s |%-20s |%-15s |%-20s |%-15s |%-17s|\n", prParts[0], prParts[1], prParts[2]
        }

    } catch (IOException e) {
        System.out.println(x: "ERROR DISPLAYING PR(S).");
    }
}
```

Figure 97: Method overriding in PurchaseReq class for display()



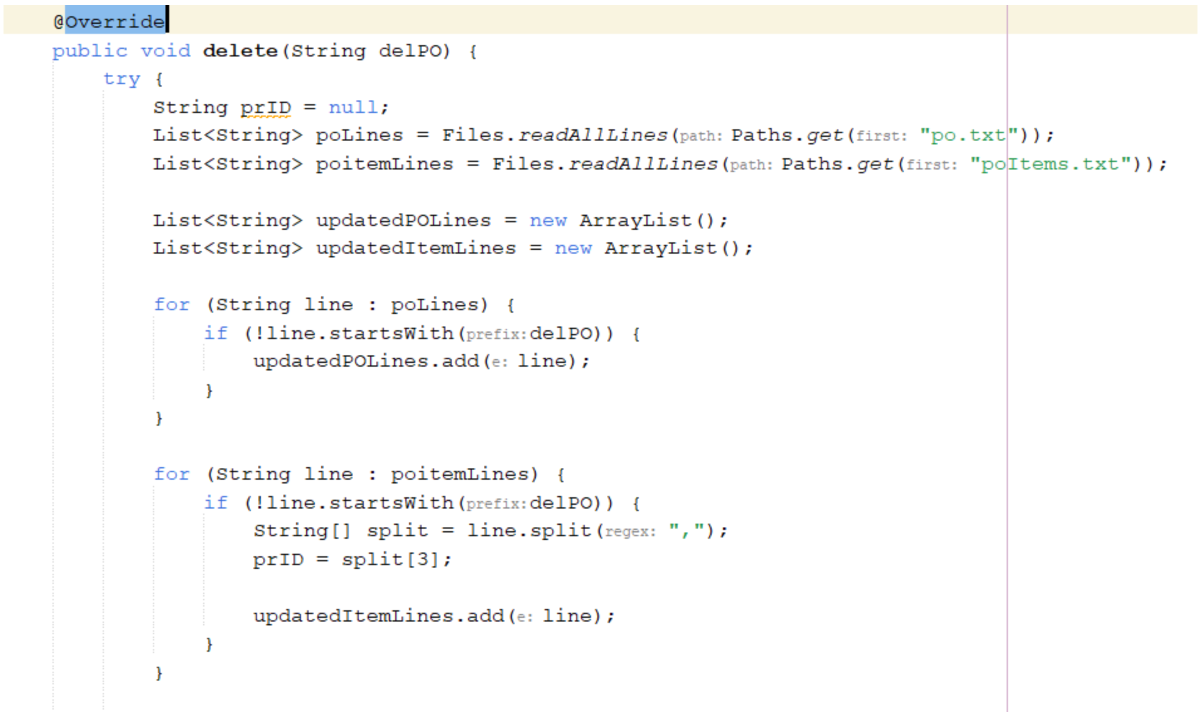
```
@Override
public boolean search(String poID) {

    try {

        BufferedReader poReader = new BufferedReader(new FileReader(fileName: "po.txt"));
        BufferedReader poitemReader = new BufferedReader(new FileReader(fileName: "poItems.txt"));

        String poLine;
        while ((poLine = poReader.readLine()) != null) {

            if (poLine.startsWith(prefix:poID)) {
                String[] prParts = poLine.split(regex: ",");
                System.out.println("PO ID: " + poID);
                System.out.println("Requestor: " + prParts[1]);
                System.out.println("Requestor ID: " + prParts[2]);
                System.out.println("Supplier ID: " + prParts[3]);
                System.out.println("Date Required By: " + prParts[4]);
                System.out.println("Total Price: " + prParts[5].substring(beginIndex: 11));
                System.out.println(x: "\nItemList:");
                System.out.printf(format:"|%-12s |%-12s |%-12s |%-20s|\n", args: "ItemID", args: "Quantity", args: "SupplierID
                String itemLine;
                while ((itemLine = poitemReader.readLine()) != null) {
                    if (itemLine.startsWith(prefix:poID)) {
                        String[] split = itemLine.split(regex: ",");
```

Figure 98: Method overriding in PurchaseOrder class for search()



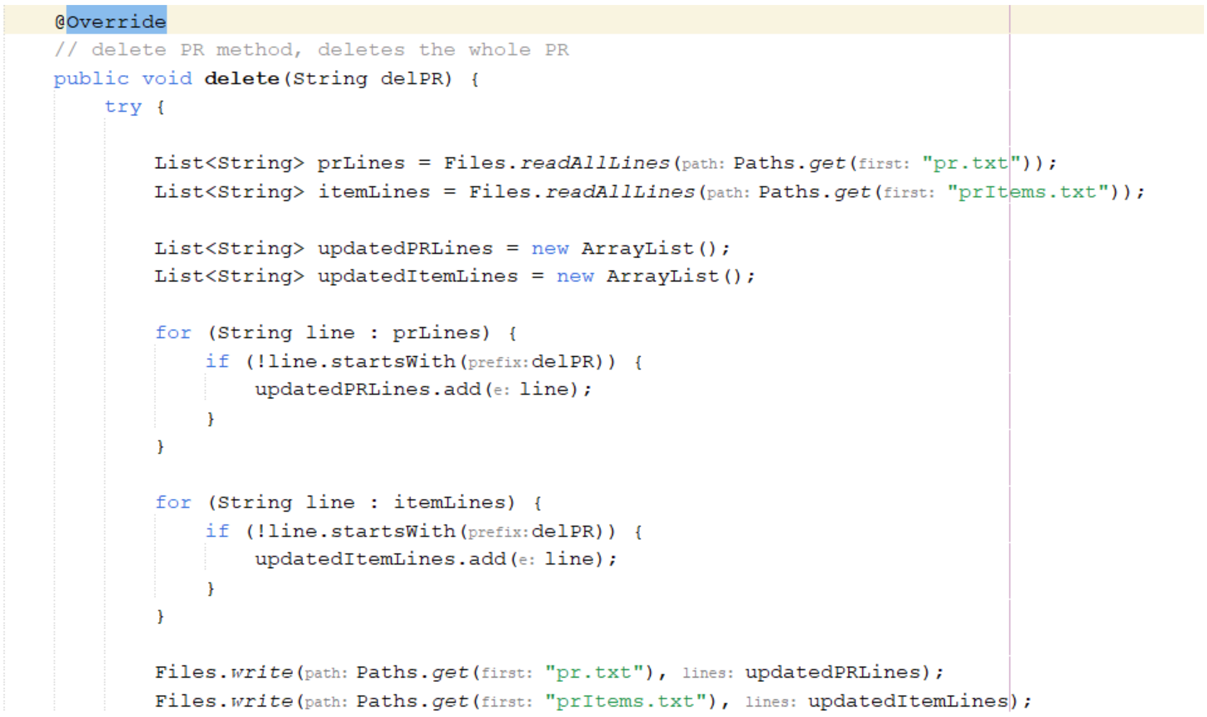
```
@Override
// searches through pr txt file to see if there is existing prID in the txt file
public boolean search(String prID) {
    if(!checkExistPRID(prID)){
        return false;
    }
    try {
        BufferedReader prReader = new BufferedReader(new FileReader(fileName: "pr.txt"));
        BufferedReader itemReader = new BufferedReader(new FileReader(fileName: "prItems.txt"));
        String prLine;
        while ((prLine = prReader.readLine()) != null) {
            if (prLine.startsWith(prefix:prID)) {
                String[] prParts = prLine.split(regex: ",");
                System.out.println("PR ID: " + prID);
                System.out.println("Requestor: " + prParts[1]);
                System.out.println("Requestor ID: " + prParts[2]);
                System.out.println("Date Required By: " + prParts[3]);
                System.out.println("Status: " + prParts[4]);
                System.out.println("Total Price: " + prParts[5].substring(beginIndex: 11));
                System.out.println(x: "\nItemList:");
                System.out.printf(format:"|%-12s |%-12s |%-12s |%-20s|\n", args: "ItemID", args: "Quantity", args: "SupplierI
                String itemLine;
                while ((itemLine = itemReader.readLine()) != null) {
                    if (itemLine.startsWith(prefix:prID)) {
                        String[] split = itemLine.split(regex: ",");
```

Figure 99: Method overriding in PurchaseReq class for search()



```
@Override
public void delete(String delPO) {
    try {
        String prID = null;
        List<String> poLines = Files.readAllLines(path: Paths.get(first: "po.txt"));
        List<String> poitemLines = Files.readAllLines(path: Paths.get(first: "poItems.txt"));

        List<String> updatedPOLines = new ArrayList();
        List<String> updatedItemLines = new ArrayList();

        for (String line : poLines) {
            if (!line.startsWith(prefix:delPO)) {
                updatedPOLines.add(e: line);
            }
        }

        for (String line : poitemLines) {
            if (!line.startsWith(prefix:delPO)) {
                String[] split = line.split(regex: ",");
                prID = split[3];

                updatedItemLines.add(e: line);
            }
        }
```

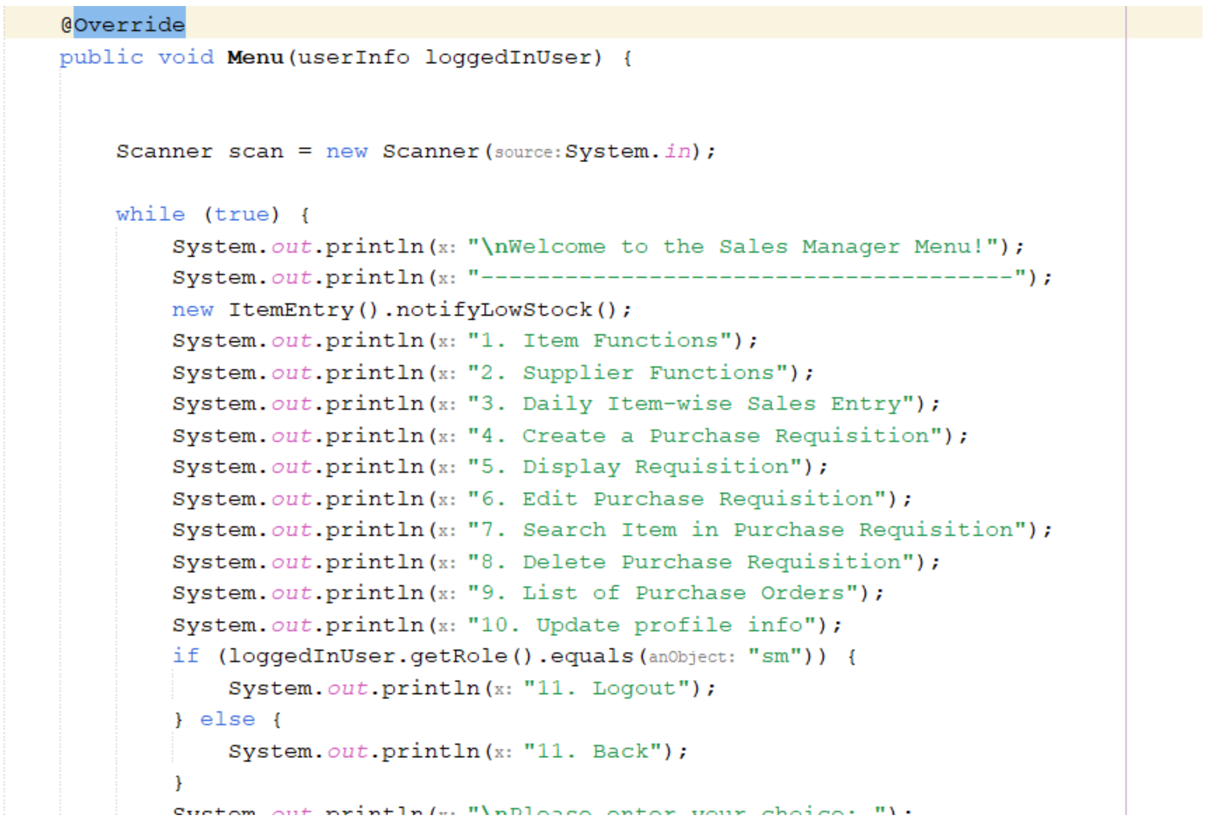
Figure 100: Method overriding in PurchaseOrder class for delete()



```
@Override
// delete PR method, deletes the whole PR
public void delete(String delPR) {
    try {

        List<String> prLines = Files.readAllLines(path: Paths.get(first: "pr.txt"));
        List<String> itemLines = Files.readAllLines(path: Paths.get(first: "prItems.txt"));

        List<String> updatedPRLines = new ArrayList();
        List<String> updatedItemLines = new ArrayList();

        for (String line : prLines) {
            if (!line.startsWith(prefix:delPR)) {
                updatedPRLines.add(e: line);
            }
        }

        for (String line : itemLines) {
            if (!line.startsWith(prefix:delPR)) {
                updatedItemLines.add(e: line);
            }
        }

        Files.write(path: Paths.get(first: "pr.txt"), lines: updatedPRLines);
        Files.write(path: Paths.get(first: "prItems.txt"), lines: updatedItemLines);
```

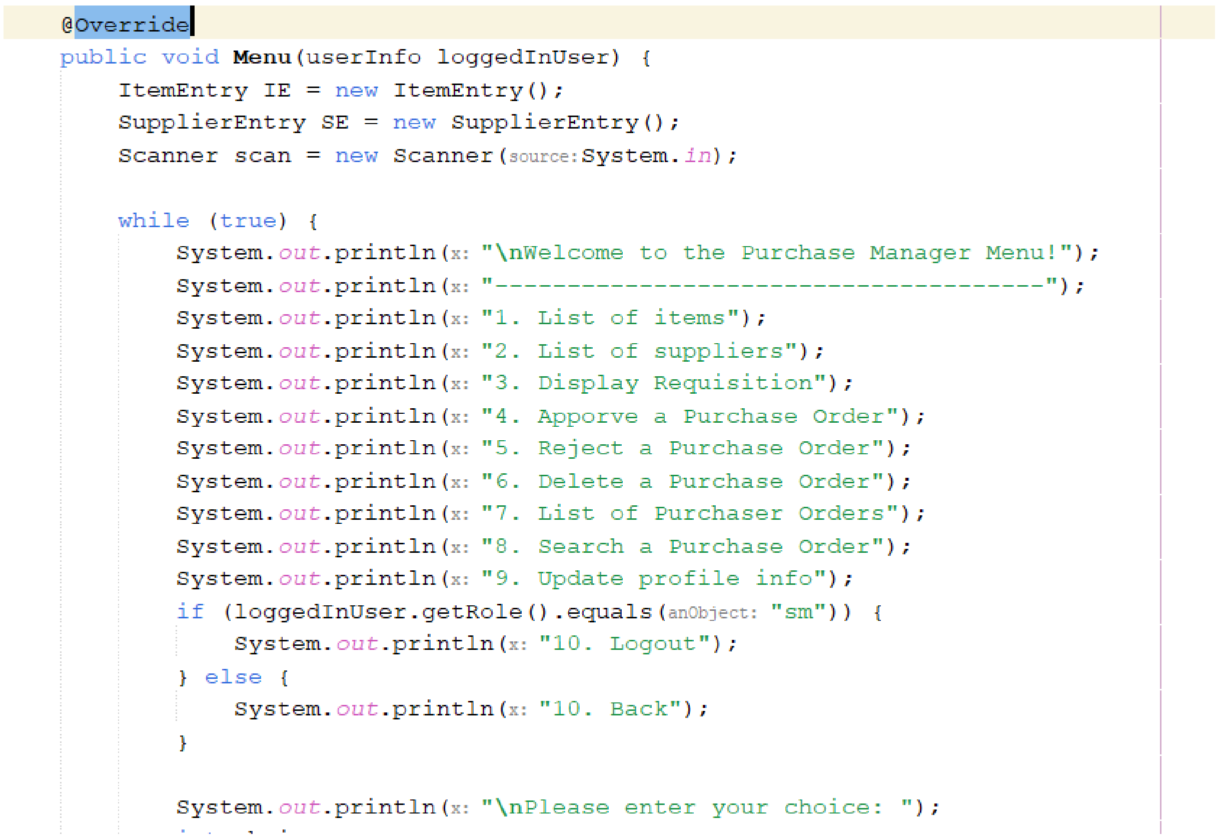
Figure 101: Method overriding in PurchaseReq class for delete()

Other methods in both the PurchaseOrder and PurchaseReq class that were also inherited from the Purchase class such as display() (figures 96 and 97), search() (figures 98 and 99) and delete() (figures 100 and 101) also employ method overriding.



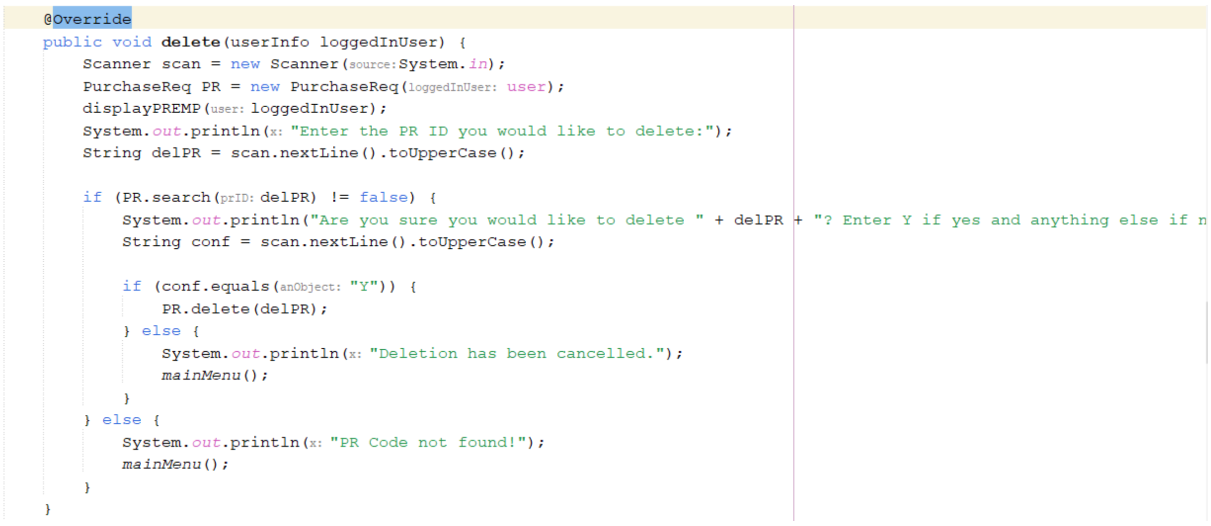
```
@Override
public void Menu(userInfo loggedInUser) {


    Scanner scan = new Scanner(source:System.in);

    while (true) {
        System.out.println(x: "\nWelcome to the Sales Manager Menu!");
        System.out.println(x: "------------------------------------");
        new ItemEntry().notifyLowStock();
        System.out.println(x: "1. Item Functions");
        System.out.println(x: "2. Supplier Functions");
        System.out.println(x: "3. Daily Item-wise Sales Entry");
        System.out.println(x: "4. Create a Purchase Requisition");
        System.out.println(x: "5. Display Requisition");
        System.out.println(x: "6. Edit Purchase Requisition");
        System.out.println(x: "7. Search Item in Purchase Requisition");
        System.out.println(x: "8. Delete Purchase Requisition");
        System.out.println(x: "9. List of Purchase Orders");
        System.out.println(x: "10. Update profile info");
        if (loggedInUser.getRole().equals(anObject: "sm")) {
            System.out.println(x: "11. Logout");
        } else {
            System.out.println(x: "11. Back");
        }
        System.out.println(x: "\nPlease enter your choice: ");
```

Figure 102: Method overriding in SalesManager class for Menu()



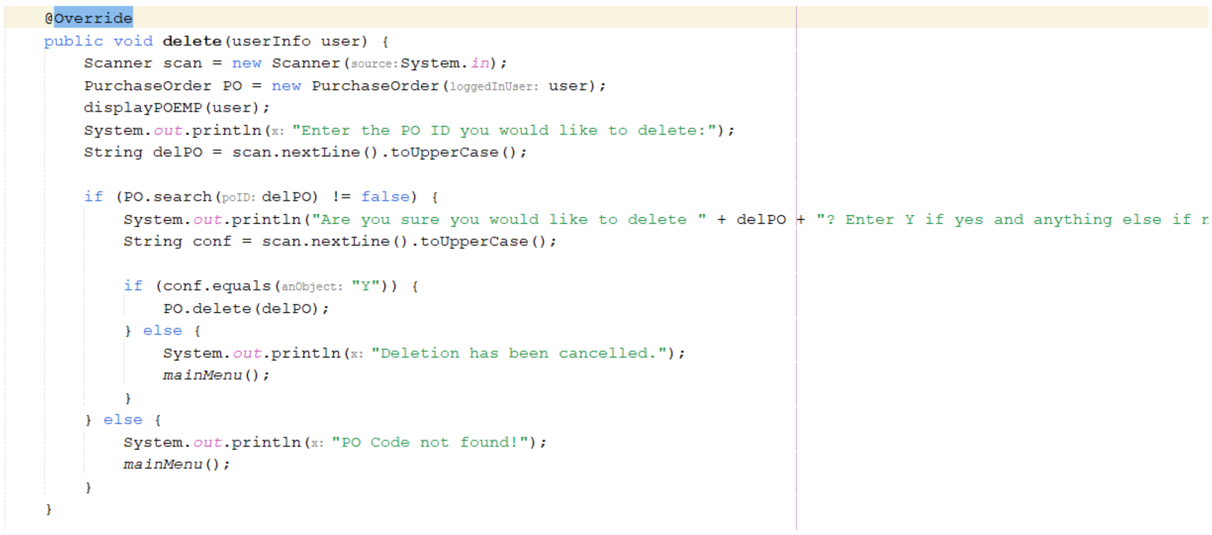
```
@Override
public void Menu(userInfo loggedInUser) {
    ItemEntry IE = new ItemEntry();
    SupplierEntry SE = new SupplierEntry();
    Scanner scan = new Scanner(source:System.in);

    while (true) {
        System.out.println(x: "\nWelcome to the Purchase Manager Menu!");
        System.out.println(x: "--------------------------------------");
        System.out.println(x: "1. List of items");
        System.out.println(x: "2. List of suppliers");
        System.out.println(x: "3. Display Requisition");
        System.out.println(x: "4. Apporve a Purchase Order");
        System.out.println(x: "5. Reject a Purchase Order");
        System.out.println(x: "6. Delete a Purchase Order");
        System.out.println(x: "7. List of Purchaser Orders");
        System.out.println(x: "8. Search a Purchase Order");
        System.out.println(x: "9. Update profile info");
        if (loggedInUser.getRole().equals(anObject: "sm")) {
            System.out.println(x: "10. Logout");
        } else {
            System.out.println(x: "10. Back");
        }

        System.out.println(x: "\nPlease enter your choice: ");
```

Figure 103: Method overriding in PurchaseManager class for Menu()



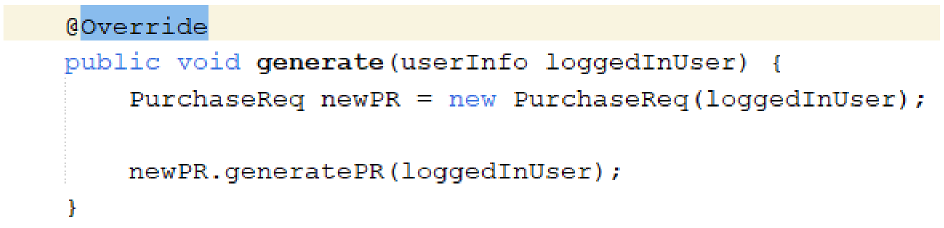
```
@Override
public void delete(userInfo loggedInUser) {
    Scanner scan = new Scanner(source:System.in);
    PurchaseReq PR = new PurchaseReq(loggedInUser: user);
    displayPREMP(user: loggedInUser);
    System.out.println(x: "Enter the PR ID you would like to delete:");
    String delPR = scan.nextLine().toUpperCase();

    if (PR.search(prID: delPR) != false) {
        System.out.println("Are you sure you would like to delete " + delPR + "? Enter Y if yes and anything else if n
        String conf = scan.nextLine().toUpperCase();

        if (conf.equals(anObject: "Y")) {
            PR.delete(delPR);
        } else {
            System.out.println(x: "Deletion has been cancelled.");
            mainMenu();
        }
    } else {
        System.out.println(x: "PR Code not found!");
        mainMenu();
    }
}
```

Figure 104: Method overriding in SalesManager class for delete()



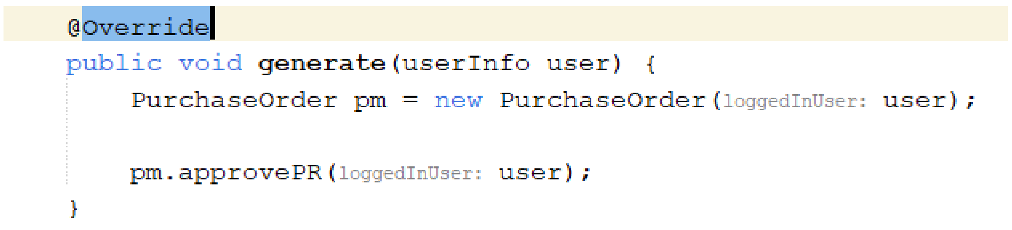
```
@Override
public void delete(userInfo user) {
    Scanner scan = new Scanner(source:System.in);
    PurchaseOrder PO = new PurchaseOrder(loggedInUser: user);
    displayPOEMP(user);
    System.out.println(x: "Enter the PO ID you would like to delete:");
    String delPO = scan.nextLine().toUpperCase();

    if (PO.search(poID: delPO) != false) {
        System.out.println("Are you sure you would like to delete " + delPO + "? Enter Y if yes and anything else if r
        String conf = scan.nextLine().toUpperCase();

        if (conf.equals(anObject: "Y")) {
            PO.delete(delPO);
        } else {
            System.out.println(x: "Deletion has been cancelled.");
            mainMenu();
        }
    } else {
        System.out.println(x: "PO Code not found!");
        mainMenu();
    }
}
```

Figure 105: Method overriding in PurchaseManager class for delete()



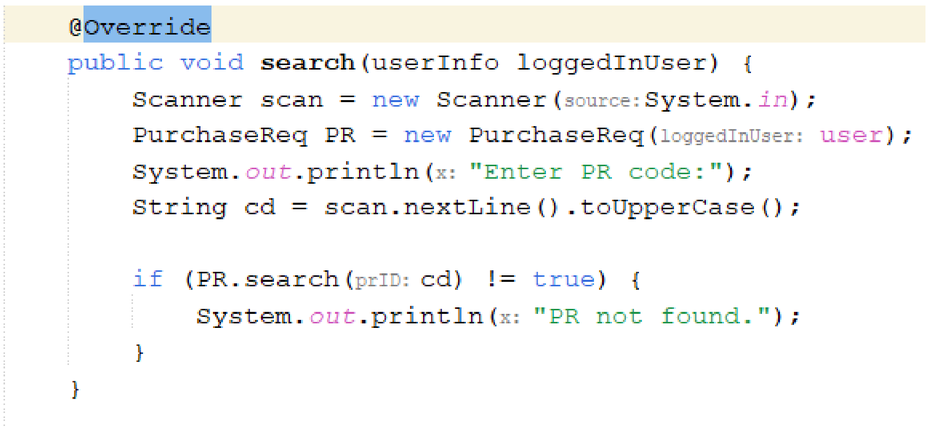
```
@Override
public void generate(userInfo loggedInUser) {
    PurchaseReq newPR = new PurchaseReq(loggedInUser);

    newPR.generatePR(loggedInUser);
}
```

Figure 106: Method overriding in SalesManager class for generate()



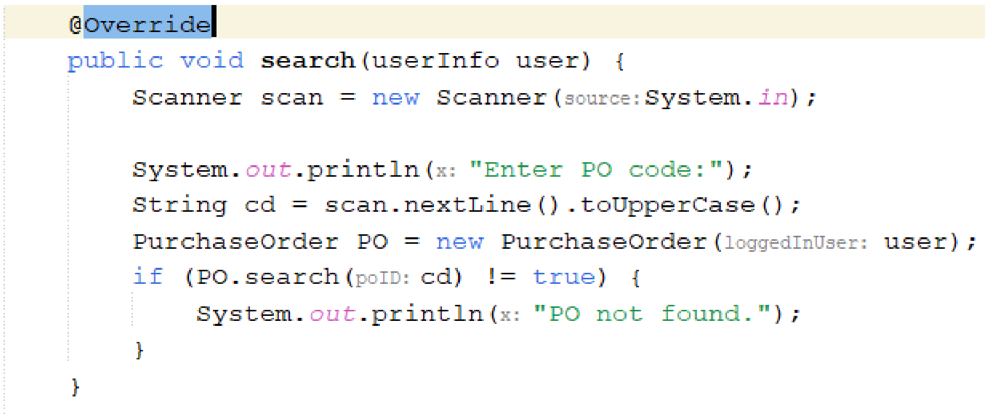
```
    @Override
    public void generate(userInfo user) {
        PurchaseOrder pm = new PurchaseOrder(loggedInUser: user);

        pm.approvePR(loggedInUser: user);
    }
```

Figure 107: Method overriding in PurchaseManager class for generate()



```
    @Override
    public void search(userInfo loggedInUser) {
        Scanner scan = new Scanner(source:System.in);
        PurchaseReq PR = new PurchaseReq(loggedInUser: user);
        System.out.println(x: "Enter PR code:");
        String cd = scan.nextLine().toUpperCase();

        if (PR.search(prID: cd) != true) {
            System.out.println(x: "PR not found.");
        }
    }
```

Figure 108: Method overriding in SalesManager class for search()



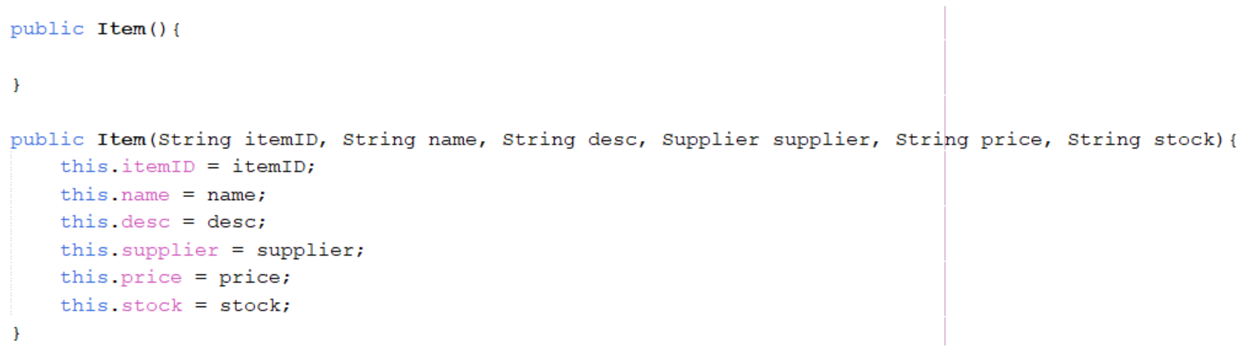
```
    @Override
    public void search(userInfo user) {
        Scanner scan = new Scanner(source:System.in);

        System.out.println(x: "Enter PO code:");
        String cd = scan.nextLine().toUpperCase();
        PurchaseOrder PO = new PurchaseOrder(loggedInUser: user);
        if (PO.search(poID: cd) != true) {
            System.out.println(x: "PO not found.");
        }
    }
```

Figure 109: Method overriding in PurchaseManager class for search()

The same applies to the SalesManager and PurchaseManager classes, whereby they both inherit the Menu() (figures 102 and 103), delete() (figures 104 and 105), generate() (figures 106 and 107) and search() (figures 108 and 109) methods from the Employee class.

Overloading

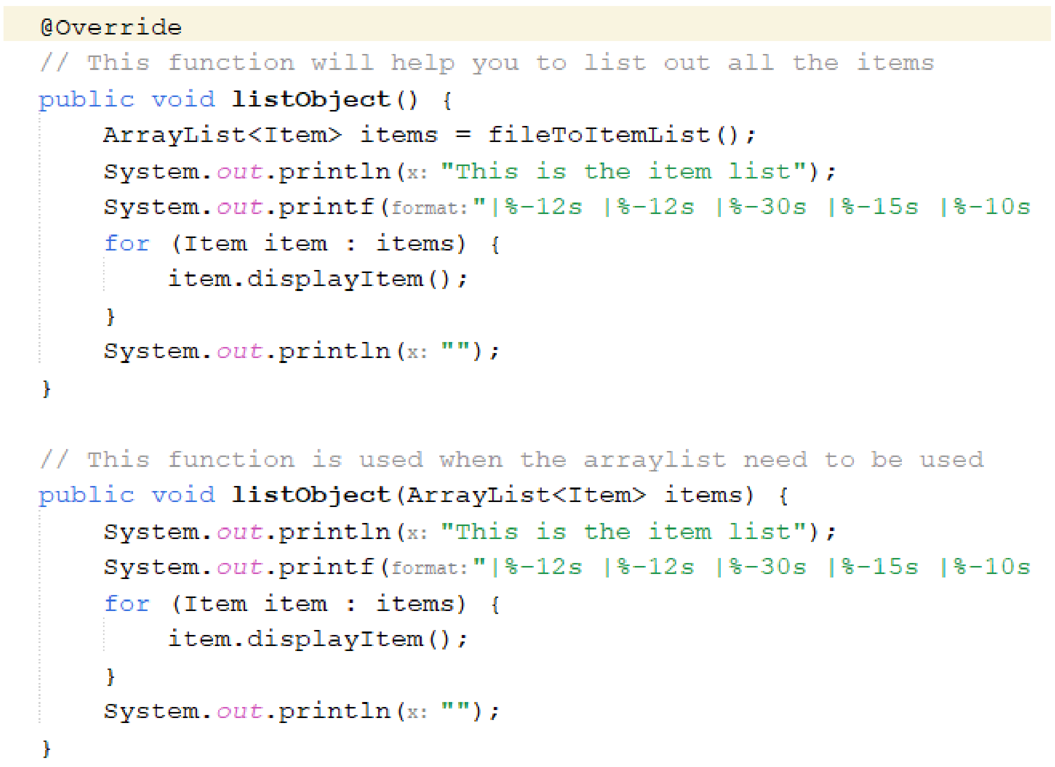
Overloading is the next polymorphism concept utilised by our system. Overloading is a Java feature in which a class has multiple methods with the same name but different parameters. It can generate multiple methods with the same name within the same class, but each method operates in a different way. When a class contains multiple methods with the same name, we refer to them as overloaded methods. (Great Learning Team, 2023)



```
public Item(){

}

public Item(String itemID, String name, String desc, Supplier supplier, String price, String stock){
    this.itemID = itemID;
    this.name = name;
    this.desc = desc;
    this.supplier = supplier;
    this.price = price;
    this.stock = stock;
}
```

Figure 110: Method overloading in Item class

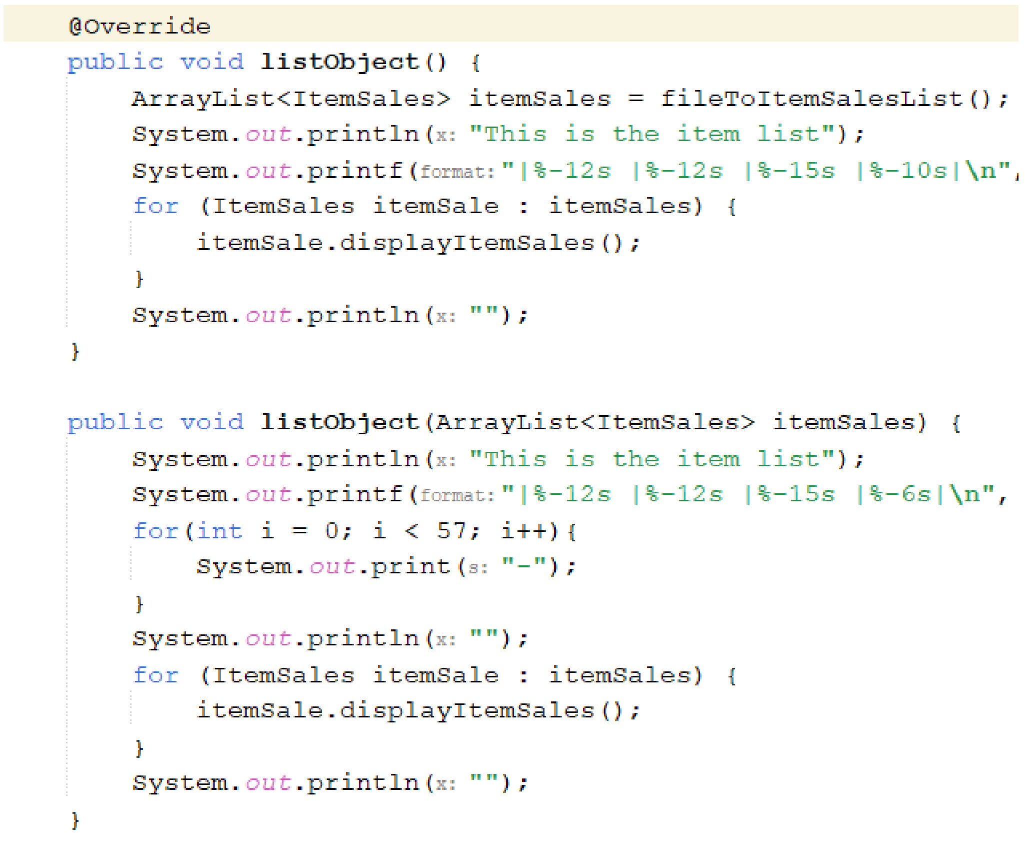
This is the overloading on constructer class. There is two same name construct class but different parameters. It is necessary to have an empty construct because we need to create an object to use its function. The constructer class with parameter is used to make an object. So even there have the same name but they have different functions.



```java
@Override
// This function will help you to list out all the items
public void listObject() {
    ArrayList<Item> items = fileToItemList();
    System.out.println(x: "This is the item list");
    System.out.printf(format:"|%-12s |%-12s |%-30s |%-15s |%-10s
    for (Item item : items) {
        item.displayItem();
    }
    System.out.println(x: "");
}

// This function is used when the arraylist need to be used
public void listObject(ArrayList<Item> items) {
    System.out.println(x: "This is the item list");
    System.out.printf(format:"|%-12s |%-12s |%-30s |%-15s |%-10s
    for (Item item : items) {
        item.displayItem();
    }
    System.out.println(x: "");
}
```

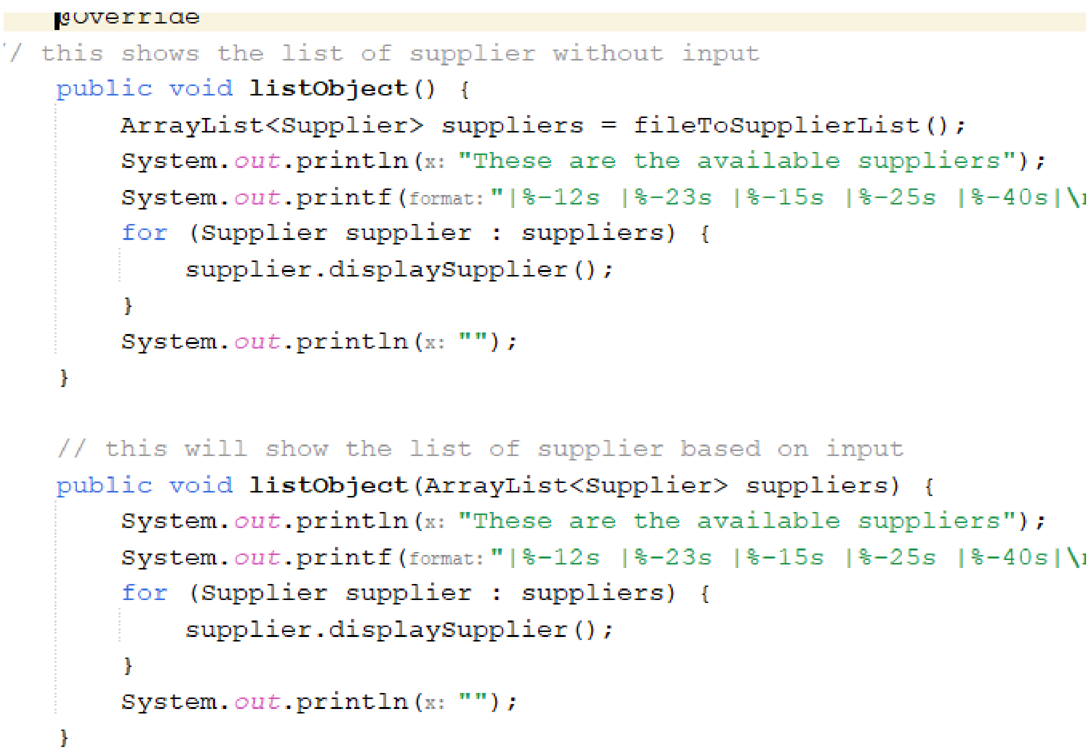
Figure 111: Method overloading in listObject() in Item class

,



```java
@Override
public void listObject() {
    ArrayList<ItemSales> itemSales = fileToItemSalesList();
    System.out.println(x: "This is the item list");
    System.out.printf(format:"|%-12s |%-12s |%-15s |%-10s|\n",
    for (ItemSales itemSale : itemSales) {
        itemSale.displayItemSales();
    }
    System.out.println(x: "");
}

public void listObject(ArrayList<ItemSales> itemSales) {
    System.out.println(x: "This is the item list");
    System.out.printf(format:"|%-12s |%-12s |%-15s |%-6s|\n",
    for(int i = 0; i < 57; i++){
        System.out.print(s: "-");
    }
    System.out.println(x: "");
    for (ItemSales itemSale : itemSales) {
        itemSale.displayItemSales();
    }
    System.out.println(x: "");
}
```

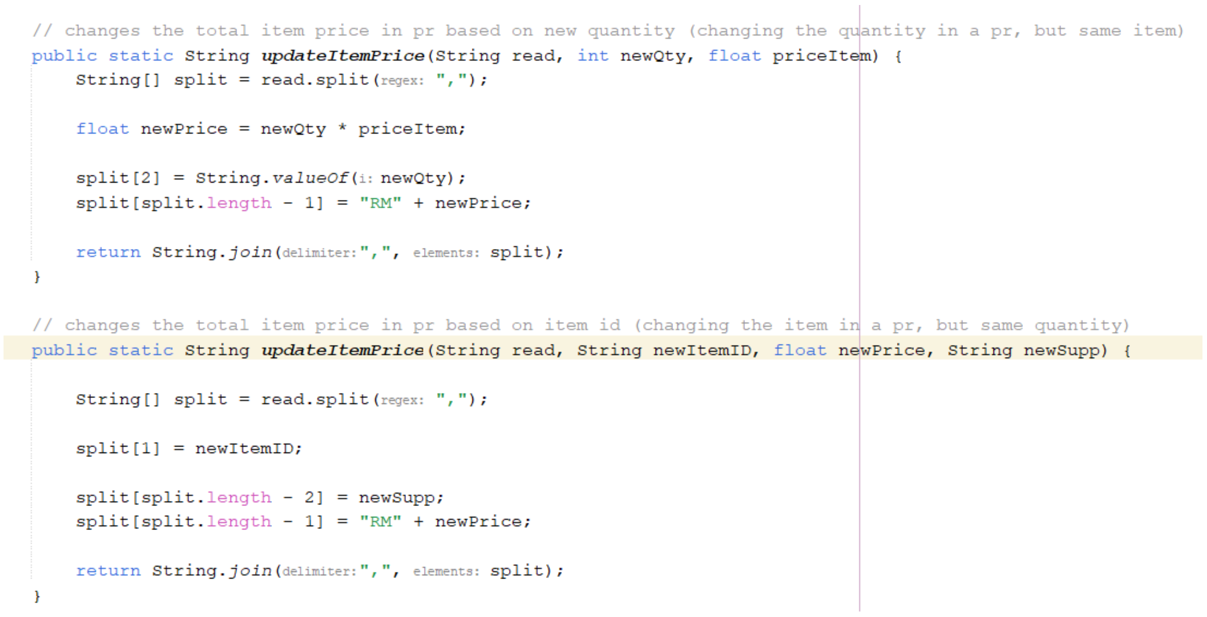
Figure 112: Method overloading in listObject() in ItemSales class



```
    @Override
/ this shows the list of supplier without input
    public void listObject() {
        ArrayList<Supplier> suppliers = fileToSupplierList();
        System.out.println(x: "These are the available suppliers");
        System.out.printf(format: "|%-12s |%-23s |%-15s |%-25s |%-40s|\
        for (Supplier supplier : suppliers) {
            supplier.displaySupplier();
        }
        System.out.println(x: "");
    }

    // this will show the list of supplier based on input
    public void listObject(ArrayList<Supplier> suppliers) {
        System.out.println(x: "These are the available suppliers");
        System.out.printf(format: "|%-12s |%-23s |%-15s |%-25s |%-40s|\
        for (Supplier supplier : suppliers) {
            supplier.displaySupplier();
        }
        System.out.println(x: "");
    }
```

Figure 113: Method overloading in listObject() in Supplier class

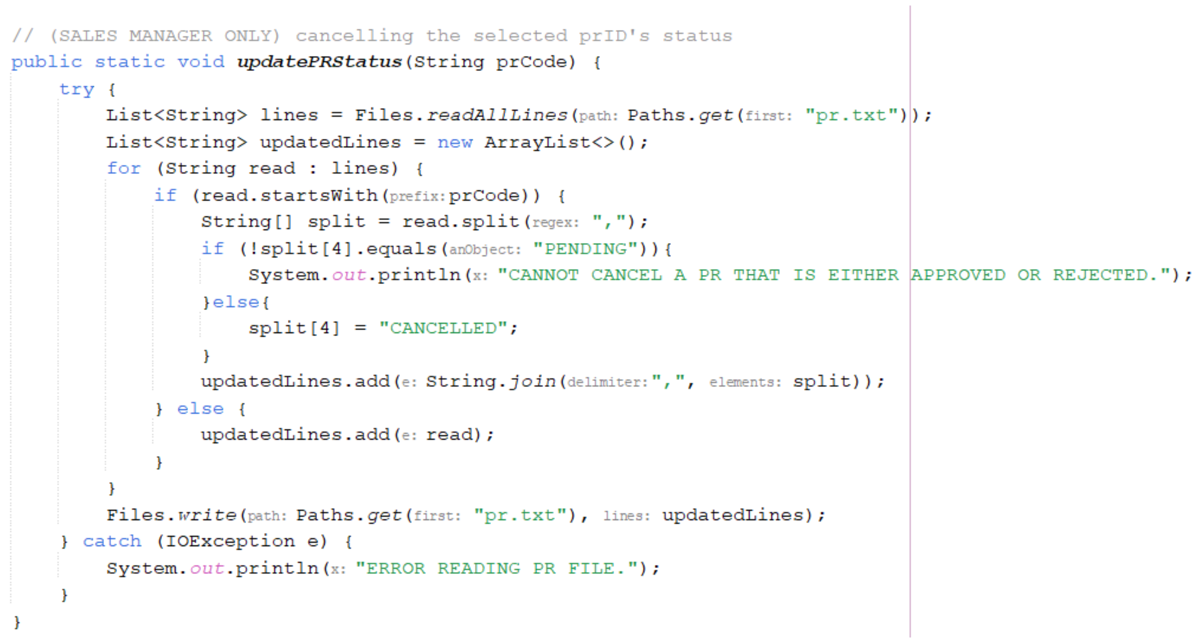
These three listObject used polymorphism because they have the same purpose which is let the user to view the data in the file. The different between them is one will directly take from the text file whereas the other will take from the ArrayList<> we input into it.



```
// changes the total item price in pr based on new quantity (changing the quantity in a pr, but same item)
public static String updateItemPrice(String read, int newQty, float priceItem) {
    String[] split = read.split(regex: ",");

    float newPrice = newQty * priceItem;

    split[2] = String.valueOf(i: newQty);
    split[split.length - 1] = "RM" + newPrice;

    return String.join(delimiter:",", elements: split);
}

// changes the total item price in pr based on item id (changing the item in a pr, but same quantity)
public static String updateItemPrice(String read, String newItemID, float newPrice, String newSupp) {

    String[] split = read.split(regex: ",");

    split[1] = newItemID;

    split[split.length - 2] = newSupp;
    split[split.length - 1] = "RM" + newPrice;

    return String.join(delimiter:",", elements: split);
}
```

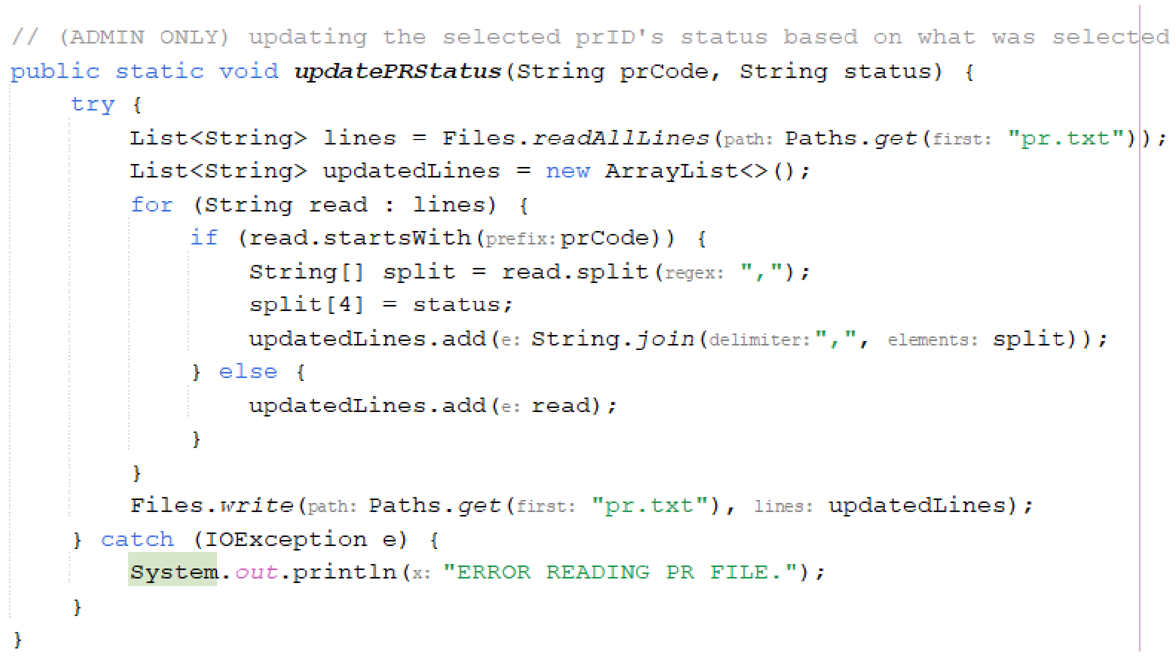
Figure 114: Method overloading in updateItemPrice() in PurchaseReq class

For these two same name functions, they have different content and purpose which one is changing quantity but the item in pr is keep and other is change item in pr, but quantity keep.



```
// (SALES MANAGER ONLY) cancelling the selected prID's status
public static void updatePRStatus(String prCode) {
    try {
        List<String> lines = Files.readAllLines(path: Paths.get(first: "pr.txt"));
        List<String> updatedLines = new ArrayList<>();
        for (String read : lines) {
            if (read.startsWith(prefix:prCode)) {
                String[] split = read.split(regex: ",");
                if (!split[4].equals(anObject: "PENDING")){
                    System.out.println(x: "CANNOT CANCEL A PR THAT IS EITHER APPROVED OR REJECTED.");
                }else{
                    split[4] = "CANCELLED";
                }
                updatedLines.add(e: String.join(delimiter:",", elements: split));
            } else {
                updatedLines.add(e: read);
            }
        }
        Files.write(path: Paths.get(first: "pr.txt"), lines: updatedLines);
    } catch (IOException e) {
        System.out.println(x: "ERROR READING PR FILE.");
    }
}
```

Figure 115: Method overloading in updatePRStatus() in PurchaseReq class (1)



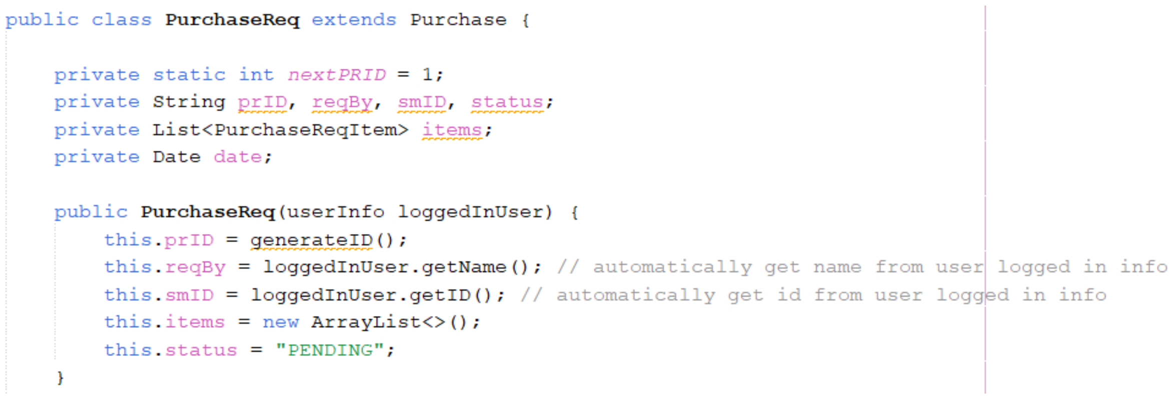
```
// (ADMIN ONLY) updating the selected prID's status based on what was selected
public static void updatePRStatus(String prCode, String status) {
    try {
        List<String> lines = Files.readAllLines(path: Paths.get(first: "pr.txt"));
        List<String> updatedLines = new ArrayList<>();
        for (String read : lines) {
            if (read.startsWith(prefix:prCode)) {
                String[] split = read.split(regex: ",");
                split[4] = status;
                updatedLines.add(e: String.join(delimiter:",", elements: split));
            } else {
                updatedLines.add(e: read);
            }
        }
        Files.write(path: Paths.get(first: "pr.txt"), lines: updatedLines);
    } catch (IOException e) {
        System.out.println(x: "ERROR READING PR FILE.");
    }
}
```

Figure 116: Method overloading in updatePRStatus() in PurchaseReq class (2)

Since the admin and sales manager have different level of permission so the updatePRstatus should also be different. The admin can update the PRID’s based on what he or she wants, whereas the sales manager can only cancel the purchase requisition.

Encapsulation

In Java, encapsulation is characterized by the practice of bundling variables and methods together into a unified unit (S, 2023). Typically, encapsulation is implemented by using a private field, making it accessible only to members of the same class. This is accomplished by declaring variables as private and offering getter methods for accessing the variables and setter methods for modifying them. This way, the internal state of an object is controlled and protected from unauthorized access or modification (Tech Target Contributor, 2016).



```
public class PurchaseReq extends Purchase {

    private static int nextPRID = 1;
    private String prID, reqBy, smID, status;
    private List<PurchaseReqItem> items;
    private Date date;

    public PurchaseReq(userInfo loggedInUser) {
        this.prID = generateID();
        this.reqBy = loggedInUser.getName(); // automatically get name from user logged in info
        this.smID = loggedInUser.getID(); // automatically get id from user logged in info
        this.items = new ArrayList<>();
        this.status = "PENDING";
    }
```

Figure 117: Encapsulation in PurchaseReq class

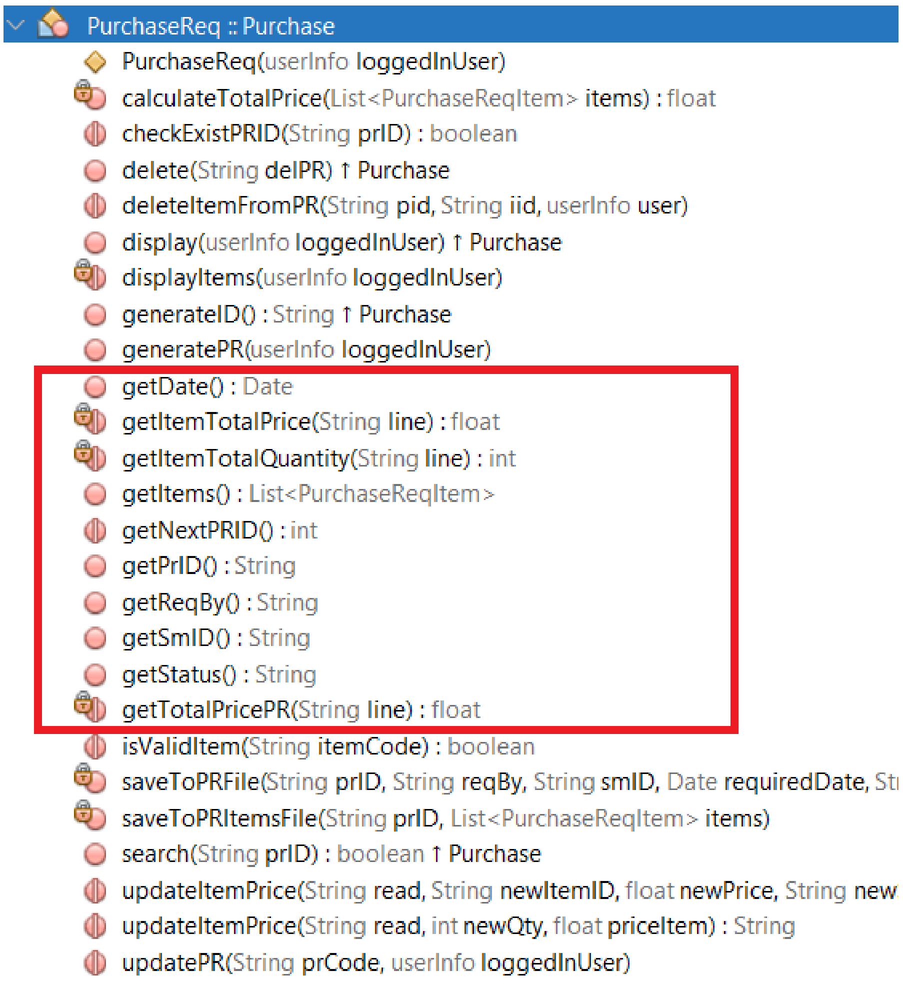
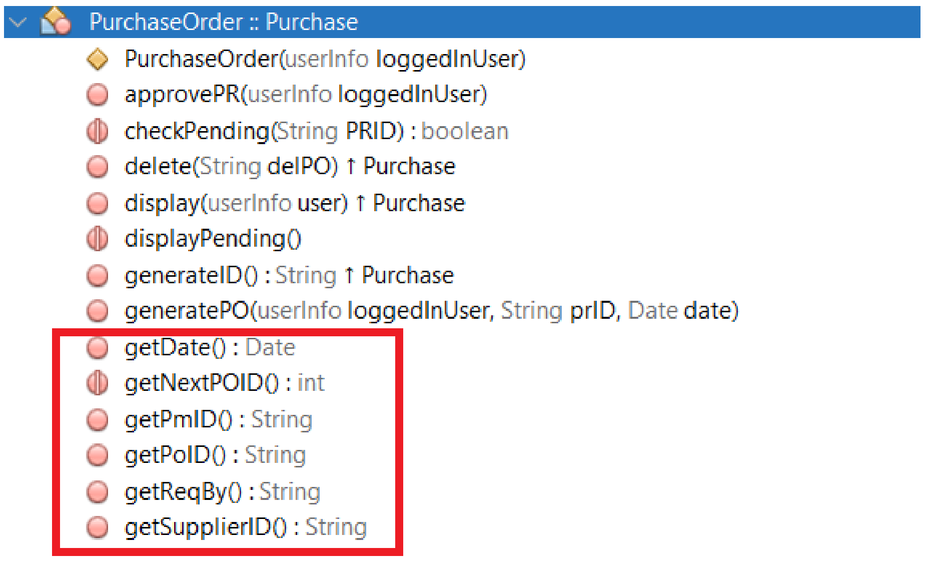

```
PurchaseReq :: Purchase
    PurchaseReq(userInfo loggedInUser)
    calculateTotalPrice(List<PurchaseReqItem> items) : float
    checkExistPRID(String prID) : boolean
    delete(String delPR) ↑ Purchase
    deleteItemFromPR(String pid, String iid, userInfo user)
    display(userInfo loggedInUser) ↑ Purchase
    displayItems(userInfo loggedInUser)
    generateID() : String ↑ Purchase
    generatePR(userInfo loggedInUser)
    getDate() : Date
    getItemTotalPrice(String line) : float
    getItemTotalQuantity(String line) : int
    getItems() : List<PurchaseReqItem>
    getNextPRID() : int
    getPrID() : String
    getReqBy() : String
    getSmID() : String
    getStatus() : String
    getTotalPricePR(String line) : float
    isValidItem(String itemCode) : boolean
    saveToPRFile(String prID, String reqBy, String smID, Date requiredDate, Stı
    saveToPRItemsFile(String prID, List<PurchaseReqItem> items)
    search(String prID) : boolean ↑ Purchase
    updateItemPrice(String read, String newItemID, float newPrice, String new
    updateItemPrice(String read, int newQty, float priceItem) : String
    updatePR(String prCode, userInfo loggedInUser)
```

Figure 118: Getter methods in PurchaseReq class

The figure above shows one implementation of abstraction in our code. In the PurchaseReq class, the variables nextPRID, prID, reqBy, smID, status, items and date are set as private attributes. Getter and setter methods are also implemented in the code, with the getter methods being shown in the box with red borders. Other examples of encapsulation in the program code are going to be shown below, with the getter methods marked in boxes with red borders and setter methods marked in boxes with blue borders.

```
public class PurchaseOrder extends Purchase {

    private static int nextPOID = 1;
    private String poID, reqBy, pmID, supplierID;
    private Date date;

    public PurchaseOrder(userInfo loggedInUser) {
        this.poID = generateID();
        this.reqBy = loggedInUser.getName();
        this.pmID = loggedInUser.getID();
    }
```

Figure 119: Encapsulation in PurchaseOrder class

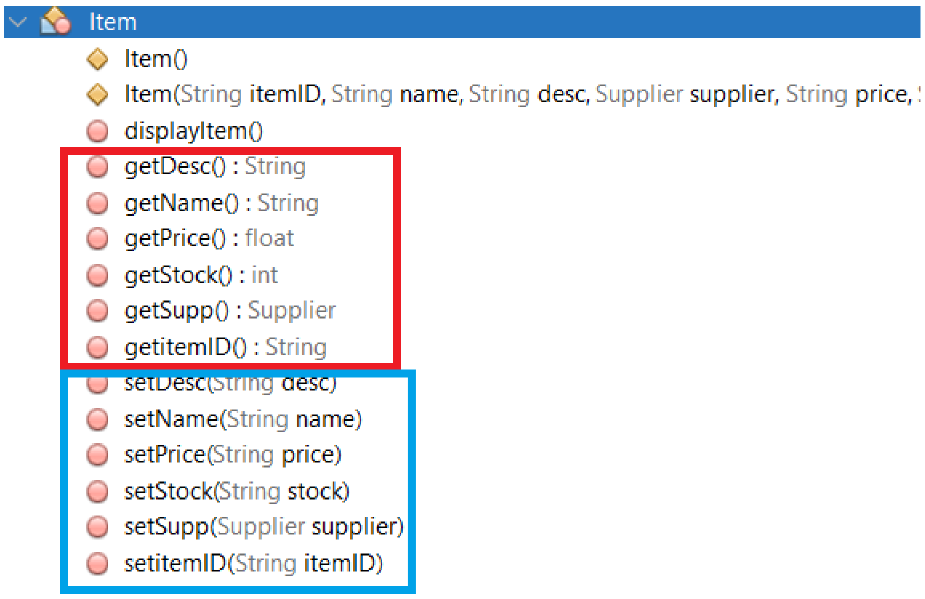


```
PurchaseOrder :: Purchase
    PurchaseOrder(userInfo loggedInUser)
    approvePR(userInfo loggedInUser)
    checkPending(String PRID) : boolean
    delete(String delPO) ↑ Purchase
    display(userInfo user) ↑ Purchase
    displayPending()
    generateID() : String ↑ Purchase
    generatePO(userInfo loggedInUser, String prID, Date date)
    getDate() : Date
    getNextPOID() : int
    getPmID() : String
    getPoID() : String
    getReqBy() : String
    getSupplierID() : String
```

Figure 120: Getter methods in PurchaseOrder class

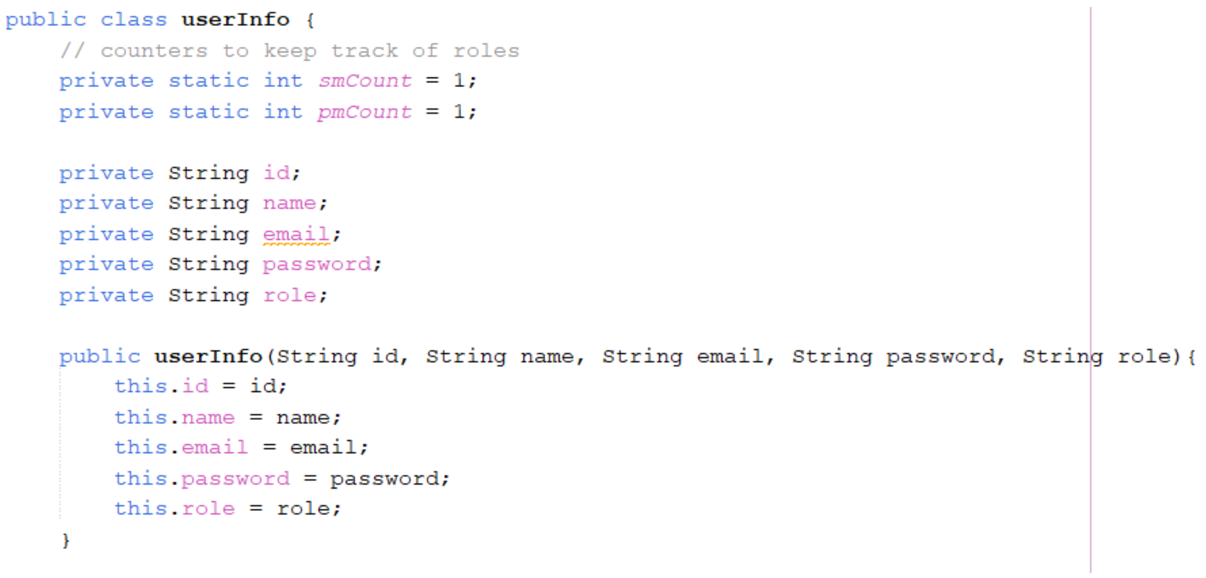
```
public class Item {
    private String itemID;
    private String name;
    private String desc;
    Supplier supplier;
    private String price;
    private String stock;

    public Item(){

    }
```

Figure 121: Encapsulation in Item class



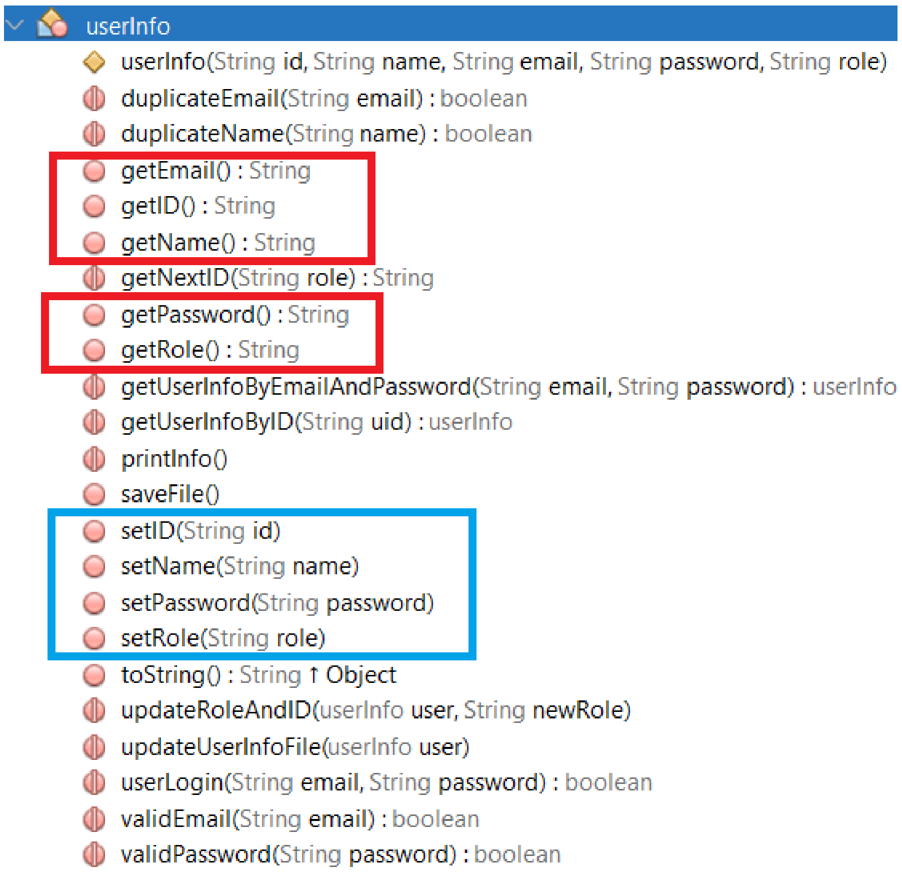
```
Item
    Item()
    Item(String itemID, String name, String desc, Supplier supplier, String price,
    displayItem()
    getDesc() : String
    getName() : String
    getPrice() : float
    getStock() : int
    getSupp() : Supplier
    getitemID() : String
    setDesc(String desc)
    setName(String name)
    setPrice(String price)
    setStock(String stock)
    setSupp(Supplier supplier)
    setitemID(String itemID)
```

Figure 122: Getter and setter methods in Item class



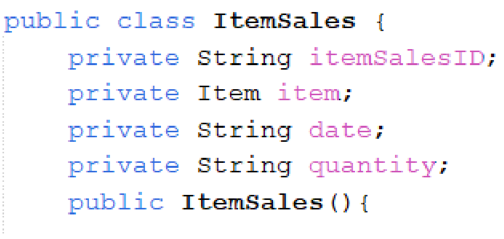
```java
public class userInfo {
    // counters to keep track of roles
    private static int smCount = 1;
    private static int pmCount = 1;

    private String id;
    private String name;
    private String email;
    private String password;
    private String role;

    public userInfo(String id, String name, String email, String password, String role){
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
    }
```

Figure 123: Encapsulation in userInfo class



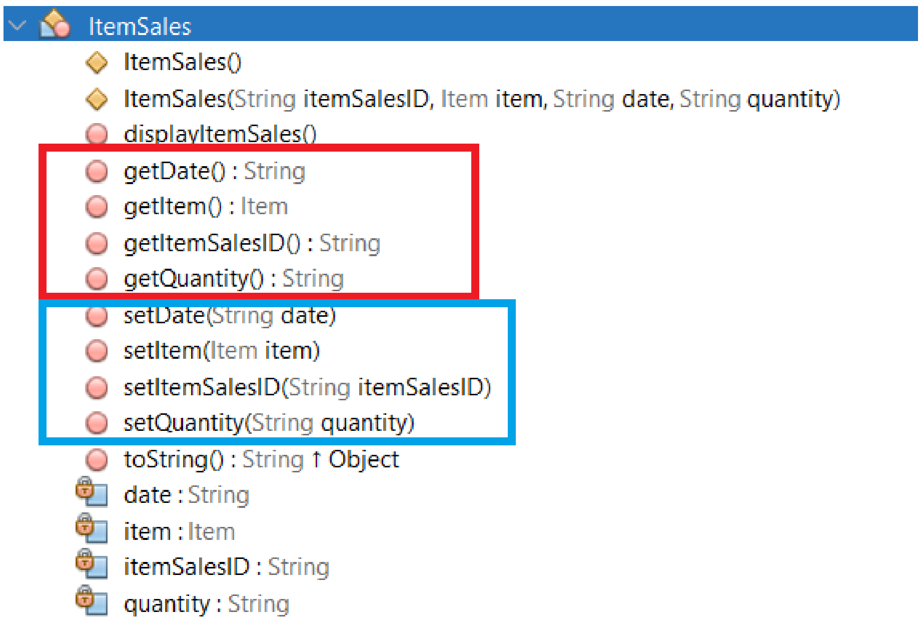
```
userInfo
    userInfo(String id, String name, String email, String password, String role)
    duplicateEmail(String email) : boolean
    duplicateName(String name) : boolean
    getEmail() : String
    getID() : String
    getName() : String
    getNextID(String role) : String
    getPassword() : String
    getRole() : String
    getUserInfoByEmailAndPassword(String email, String password) : userInfo
    getUserInfoByID(String uid) : userInfo
    printInfo()
    saveFile()
    setID(String id)
    setName(String name)
    setPassword(String password)
    setRole(String role)
    toString() : String ↑ Object
    updateRoleAndID(userInfo user, String newRole)
    updateUserInfoFile(userInfo user)
    userLogin(String email, String password) : boolean
    validEmail(String email) : boolean
    validPassword(String password) : boolean
```

Figure 124: Getter and setter methods in userInfo class



```java
public class ItemSales {
    private String itemSalesID;
    private Item item;
    private String date;
    private String quantity;
    public ItemSales(){
```

Figure 125: Encapsulation in ItemSales class

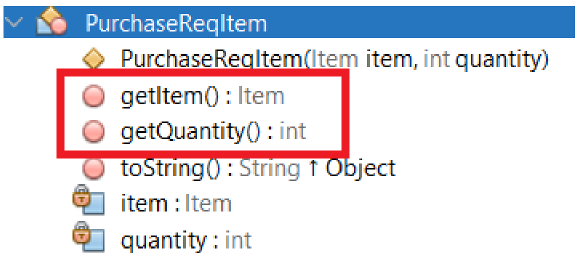


```
ItemSales
    ItemSales()
    ItemSales(String itemSalesID, Item item, String date, String quantity)
    displayItemSales()
    getDate() : String
    getItem() : Item
    getItemSalesID() : String
    getQuantity() : String
    setDate(String date)
    setItem(Item item)
    setItemSalesID(String itemSalesID)
    setQuantity(String quantity)
    toString() : String ↑ Object
    date : String
    item : Item
    itemSalesID : String
    quantity : String
```

Figure 126: Getter and setter methods in ItemSales class

```java
public class PurchaseReqItem{
    private Item item;
    private int quantity;
```

Figure 127: Encapsulation in PurchaseReqItem class



```
PurchaseReqItem
    PurchaseReqItem(Item item, int quantity)
    getItem() : Item
    getQuantity() : int
    toString() : String ↑ Object
    item : Item
    quantity : int
```

Figure 128: Getter methods in PurchaseReqItem class

```java
public class Supplier {
    private String supplierID;
    private String name;
    private String number;
    private String email;
    private String address;
    public Supplier(){
    }
```

Figure 129: Encapsulation in Supplier class

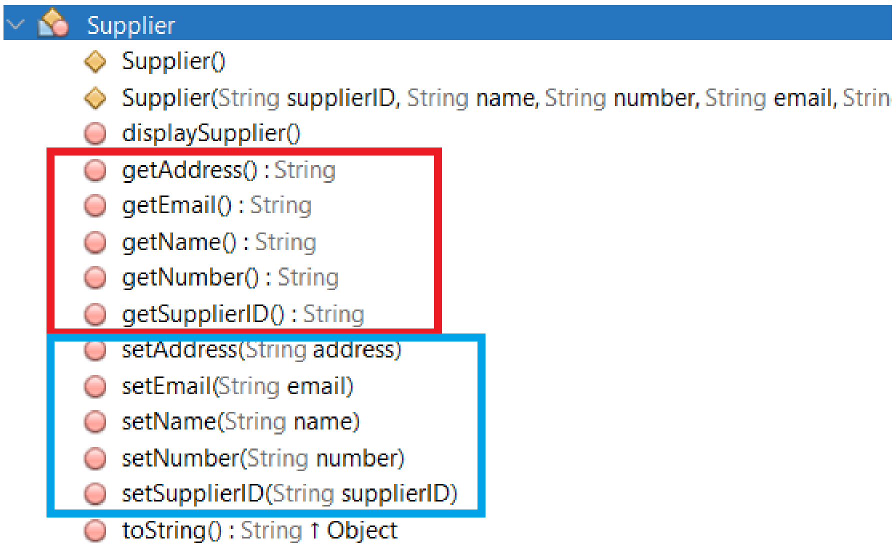
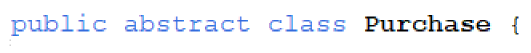

Supplier
Supplier()
Supplier(String supplierID, String name, String number, String email, Strin
displaySupplier()
getAddress() : String
getEmail() : String
getName() : String
getNumber() : String
getSupplierID() : String
setAddress(String address)
setEmail(String email)
setName(String name)
setNumber(String number)
setSupplierID(String supplierID)
toString() : String ↑ Object

Figure 130: Getter and setter methods in Supplier class

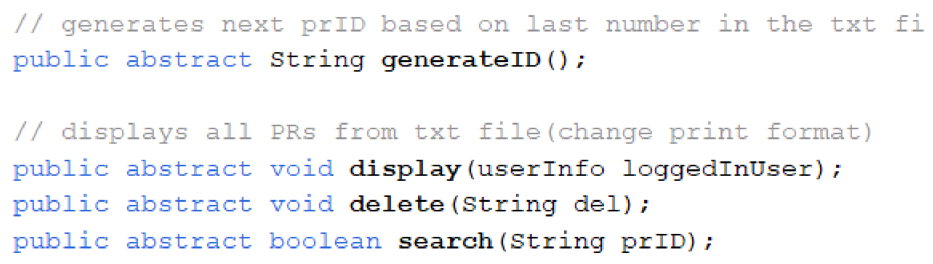
Abstraction

In Java, abstraction is a concept that involves concealing unnecessary details and revealing only the pertinent and essential information (miglani, n.d.). This can be accomplished by creating an abstract class and defining abstract methods within it using the "abstract" keyword. Subsequently, subclasses are derived from this abstract class. These subclasses are then responsible for providing their own implementations of the abstract methods outlined in the abstract class. This approach effectively combines both inheritance and polymorphism. Abstraction, in this context, often involves creating templates for other classes and leveraging polymorphism for flexible usage.



```java
public abstract class Purchase {
```

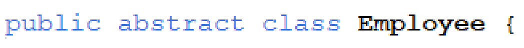
Figure 131: Abstract Purchase class



```java
// generates next prID based on last number in the txt fi
public abstract String generateID();

// displays all PRs from txt file(change print format)
public abstract void display(userInfo loggedInUser);
public abstract void delete(String del);
public abstract boolean search(String prID);
```

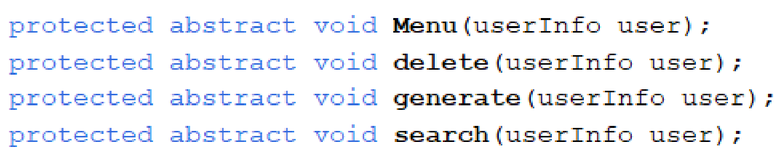
Figure 132: Abstract methods in Purchase class

In the figures above, the Purchase superclass is an abstract superclass, and it contains abstract methods that are to be inherited and polymorphed by the PurchaseReq and PurchaseOrder classes.



```java
public abstract class Employee {
```

Figure 133: Abstract class Employee

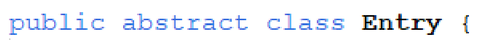


```java
protected abstract void Menu(userInfo user);
protected abstract void delete(userInfo user);
protected abstract void generate(userInfo user);
protected abstract void search(userInfo user);
```

Figure 134: Abstract methods in Employee class

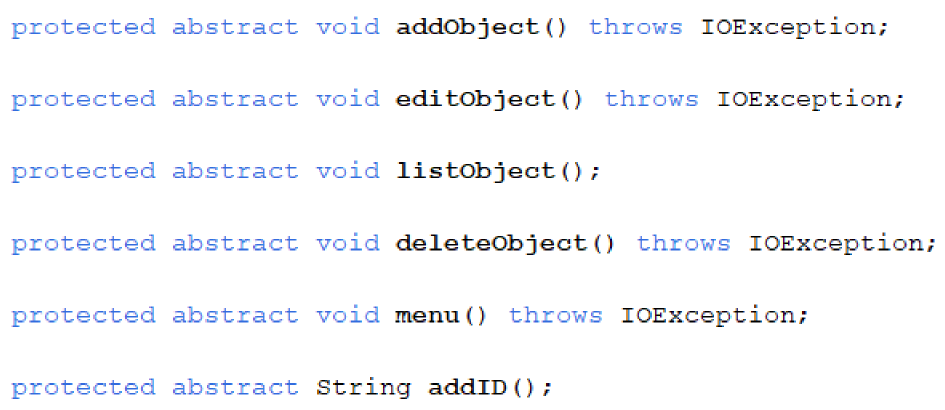
The same can be said for the Employee superclass, whereby it is defined as an abstract class and contains abstract methods to be inherited and polymorphed by the SalesManager and

PurchaseManager classes. When these two subclasses inherit this abstract class they might share the same name but different content inside.



```java
public abstract class Entry {
```

Figure 135: Abstract Entry class



```java
    protected abstract void addObject() throws IOException;

    protected abstract void editObject() throws IOException;

    protected abstract void listObject();

    protected abstract void deleteObject() throws IOException;

    protected abstract void menu() throws IOException;

    protected abstract String addID();
```

Figure 136: Abstract methods in Entry class

The last abstract class is Entry class the class the class contain addObject(), editObject(), listObject(), deleteObject(), menu(), addID(). The Entry class will be inherited by the itemEntry class, supplierEntry class, itemSalesEntry class. The reason we use abstraction is because they have common functions with the same name but different functions. So, for easy recognition we use abstraction and abstraction can also remind use what this class should consist of to prevent missing.

Modularity

In Java, modularity pertains to the approach of decomposing an intricate system into more manageable components. It's a method aimed at simplifying the design and upkeep of a software product by dividing it into distinct, manageable parts (Debnath, Modularity in Java: Java 9 Modularity versus Prior Versions, 2017). Modularity is crucial because it plays a significant role in reducing complexity and enhancing the comprehensibility and maintainability of code.

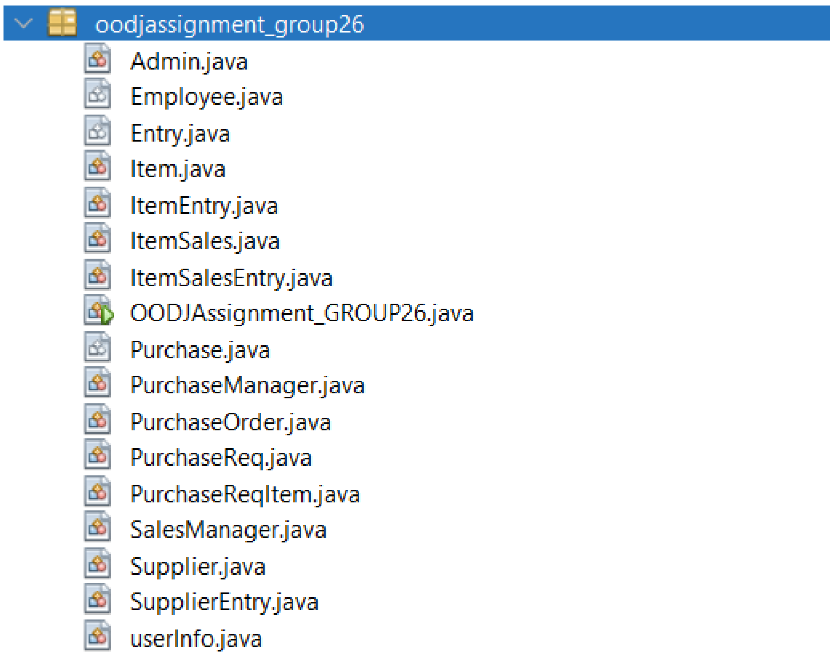

oodjassignment_group26
- Admin.java
- Employee.java
- Entry.java
- Item.java
- ItemEntry.java
- ItemSales.java
- ItemSalesEntry.java
- OODJAssignment_GROUP26.java
- Purchase.java
- PurchaseManager.java
- PurchaseOrder.java
- PurchaseReq.java
- PurchaseReqItem.java
- SalesManager.java
- Supplier.java
- SupplierEntry.java
- userInfo.java

Figure 137: Modularity in program

Modularity was implemented in the system by breaking it down into classes with specific purposes. For example, the userInfo class would house the user's information, including getter and setter methods for the user credentials, and the ability to modify the user's information. PurchaseReq contains all of the purchase requisition functions, including but not limited to generating, deleting, editing aspects of, searching, and saving a purchase requisition. Instead of adding the itemEntry and item together into one class we spilt it into two classes as item is an item and entry is an action. This can help us more specific the system and easier to debug or review when we need to reuse or refactor the program. The same goes for supplier and supplierEntry and itemSales and itemSalesEntry.

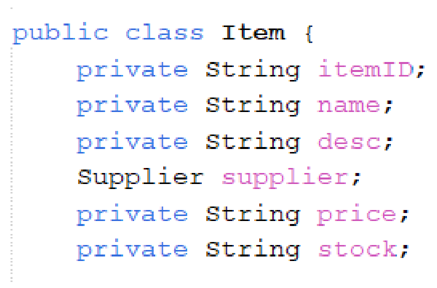
```
package oodjassignment_group26;
```

Figure 138: Main class package

Additionally, the main class is contained in a package for added modularity.

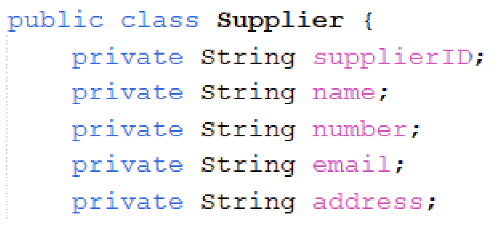
Composition

Composition is one of the fundamental principles in object-oriented programming. It defines a class that instance variables contain references to one or more objects of other classes. This permits the modelling of a 'has-a' relationship between objects. The dependent object will be meaningless if the independent object is not there. (Janssen, 2023)



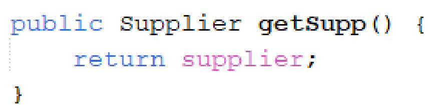
```java
public class Item {
    private String itemID;
    private String name;
    private String desc;
    Supplier supplier;
    private String price;
    private String stock;
```

Figure 139: Item class



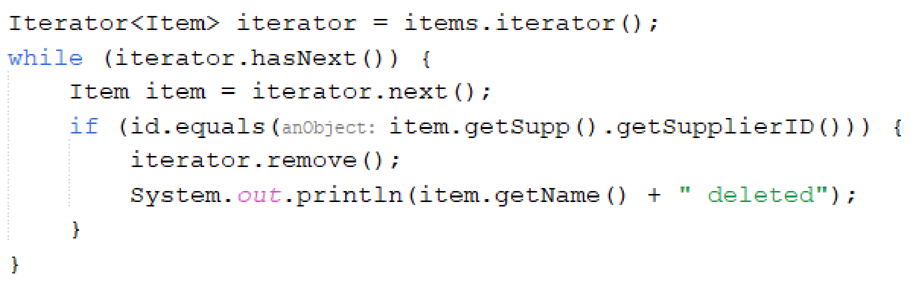
```java
public class Supplier {
    private String supplierID;
    private String name;
    private String number;
    private String email;
    private String address;
```

Figure 140: Supplier class



```java
public Supplier getSupp() {
    return supplier;
}
```

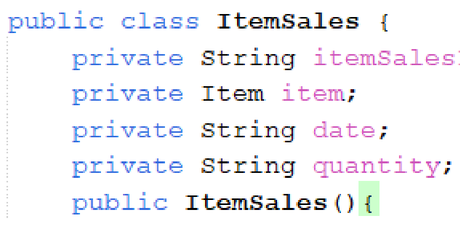
Figure 141: Function getSupp needs Supplier class



```java
Iterator<Item> iterator = items.iterator();
while (iterator.hasNext()) {
    Item item = iterator.next();
    if (id.equals(anObject: item.getSupp().getSupplierID())) {
        iterator.remove();
        System.out.println(item.getName() + " deleted");
    }
}
```

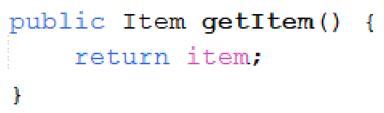
Figure 142: Method in Item class needs Supplier

From this site we can see that when we are deleting the supplier, we will also delete the item supplied by the item, so we need to get the supplier in the item. Then we will call the ID of the supplier to check all the item with the supplier's name, is the item is possessed by the supplier then it will be deleted. The composition is used between the item and supplier class as every item must be supplied by one supplier and all the item supplied by the supplier will be gone if the supplier is deleted.



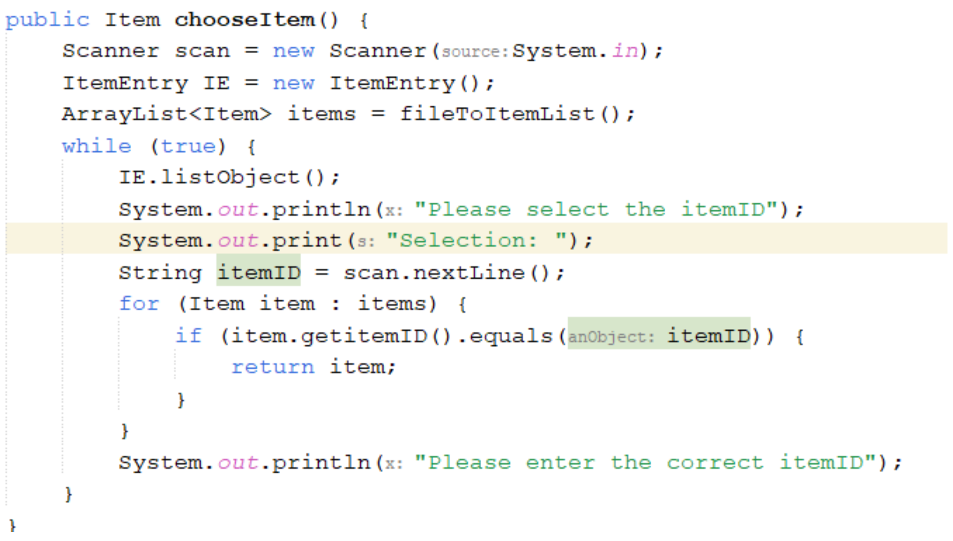
```
public class ItemSales {
    private String itemSalesID;
    private Item item;
    private String date;
    private String quantity;
    public ItemSales(){
```

Figure 143: ItemSales class



```
public Item getItem() {
    return item;
}
```

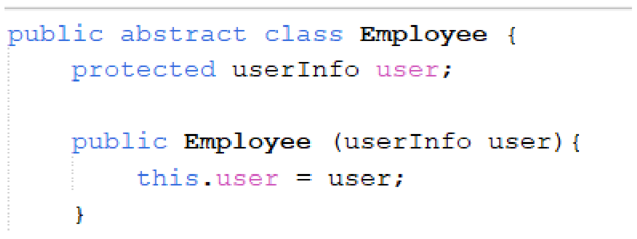
Figure 144: getItem() in ItemSales needs Item class



```
public Item chooseItem() {
    Scanner scan = new Scanner(source:System.in);
    ItemEntry IE = new ItemEntry();
    ArrayList<Item> items = fileToItemList();
    while (true) {
        IE.listObject();
        System.out.println(x: "Please select the itemID");
        System.out.print(s: "Selection: ");
        String itemID = scan.nextLine();
        for (Item item : items) {
            if (item.getitemID().equals(anObject: itemID)) {
                return item;
            }
        }
        System.out.println(x: "Please enter the correct itemID");
    }
}
```

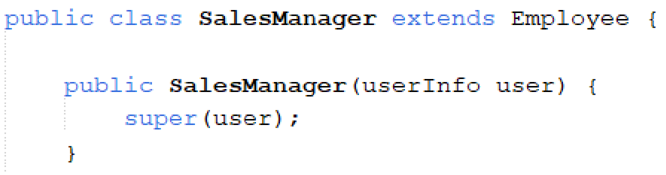
Figure 145: Method in ItemSales() class needs Item class

Same goes to the itemSales and item, the itemSales must have an item, so they have composition relationship. But when the item is deleted, the item sales will still be remained as it will keep as a record for future use. The delete of item sales will also not affect the item class as the item is not dependent on itemSales.



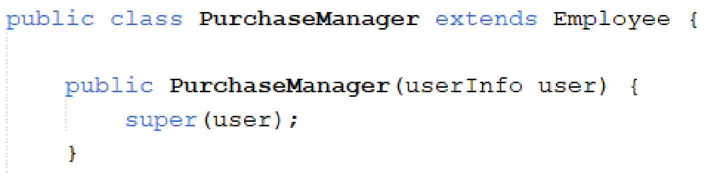
```java
public abstract class Employee {
    protected userInfo user;

    public Employee (userInfo user){
        this.user = user;
    }
```

Figure 146: Employee class



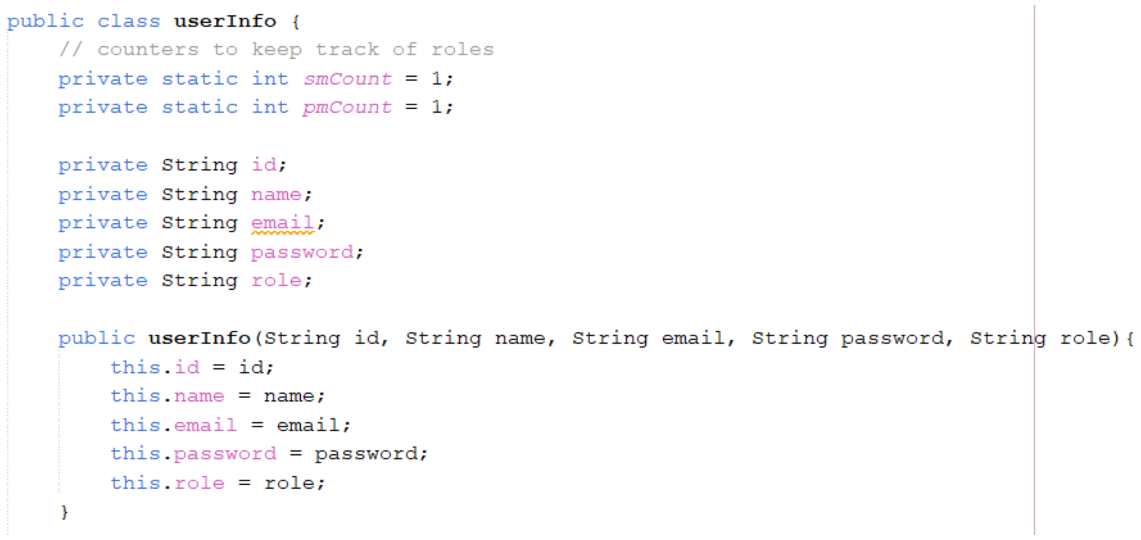
```java
public class SalesManager extends Employee {

    public SalesManager(userInfo user) {
        super(user);
    }
```

Figure 147: SalesManager class



```java
public class PurchaseManager extends Employee {

    public PurchaseManager(userInfo user) {
        super(user);
    }
```

Figure 148: PurchaseManager class



```java
public class userInfo {
    // counters to keep track of roles
    private static int smCount = 1;
    private static int pmCount = 1;

    private String id;
    private String name;
    private String email;
    private String password;
    private String role;

    public userInfo(String id, String name, String email, String password, String role){
        this.id = id;
        this.name = name;
        this.email = email;
        this.password = password;
        this.role = role;
    }
```

Figure 149: The Employee, SalesManager and PurchaseManager classes above needs userInfo class

Another occurrence of composition in the program code is between the Employee and userInfo class, where the Employee relies on the userInfo class. This means that if the userInfo class was to be deleted, the Employee class and its subclasses such as SalesManager and PurchaseManager will be deleted as well as it needs the userInfo class to exist. However, if the Employee class were to be deleted, the userInfo class will not be affected at all.

6.0 Limitations of the System

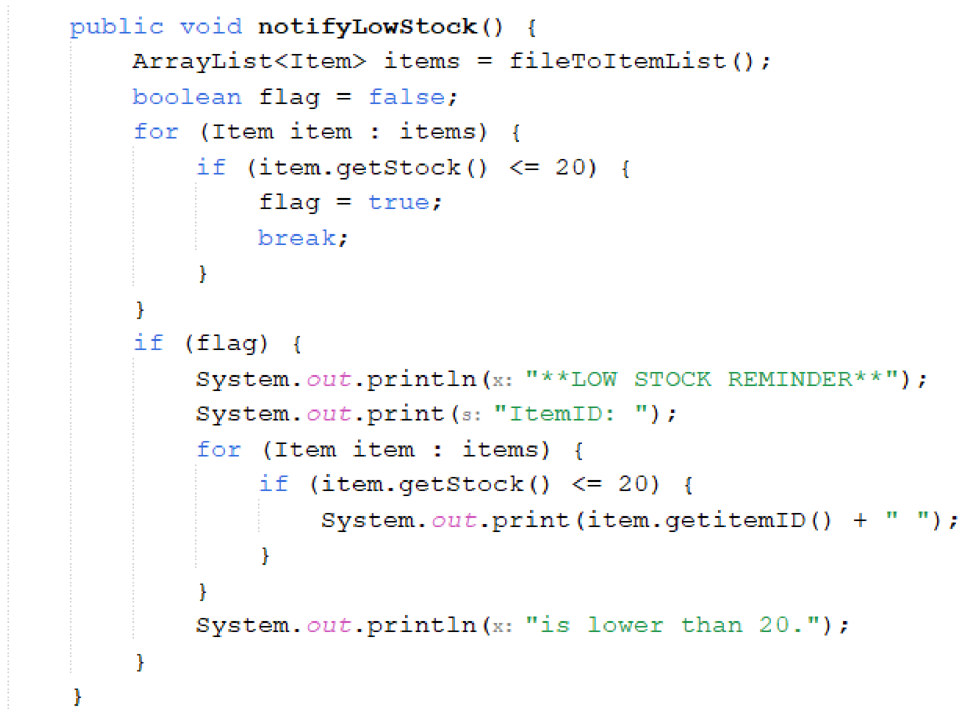
In this system, we only allow each item supplied by one supplier as we believe that each supplier supplied different quality or identity of that item even if they are the same item. For example, one supplier supplies frozen chicken meat whereas another supplies fresh chicken meat; they are both inherently chicken meat but with difference. We believe that separating the item is better for item recognition.

The itemID, supplierID and itemSales ID is not allowed to edit as they are believed to be unique, we allow user to set at the beginning but not allow them to change after it. This is because they act as foreign key in other file so simply changing might cause error.

The users are also not allowed to change their email addresses as this is a primary form of validating the security of a user. We can't validate the user's identity by their name and passwords alone, but we can do so by checking their email addresses. Additionally, it's risky to change their email addresses as notifications will be sent to the new email addresses; so imagine a scenario whereby the user changes their email address frequently, this may result in a loss of communication and notifications if they were to try to find old notifications in their old email addresses. By keeping their email addresses the same, the system can ensure that the users will be able to consistently receive notifications without any worry of missed notifications.

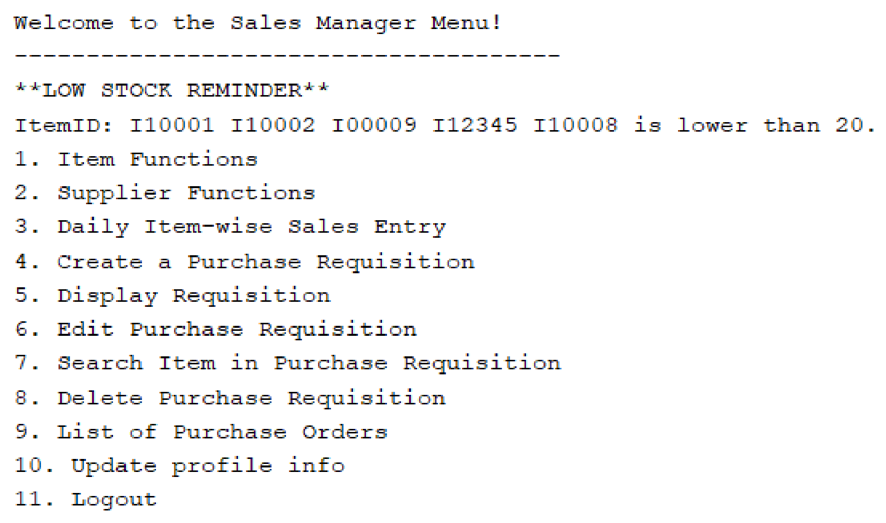
7.0 Additional features

Low Stock Notification



```
public void notifyLowStock() {
    ArrayList<Item> items = fileToItemList();
    boolean flag = false;
    for (Item item : items) {
        if (item.getStock() <= 20) {
            flag = true;
            break;
        }
    }
    if (flag) {
        System.out.println(x: "**LOW STOCK REMINDER**");
        System.out.print(s: "ItemID: ");
        for (Item item : items) {
            if (item.getStock() <= 20) {
                System.out.print(item.getitemID() + " ");
            }
        }
        System.out.println(x: "is lower than 20.");
    }
}
```

Figure 150: Low stock notification

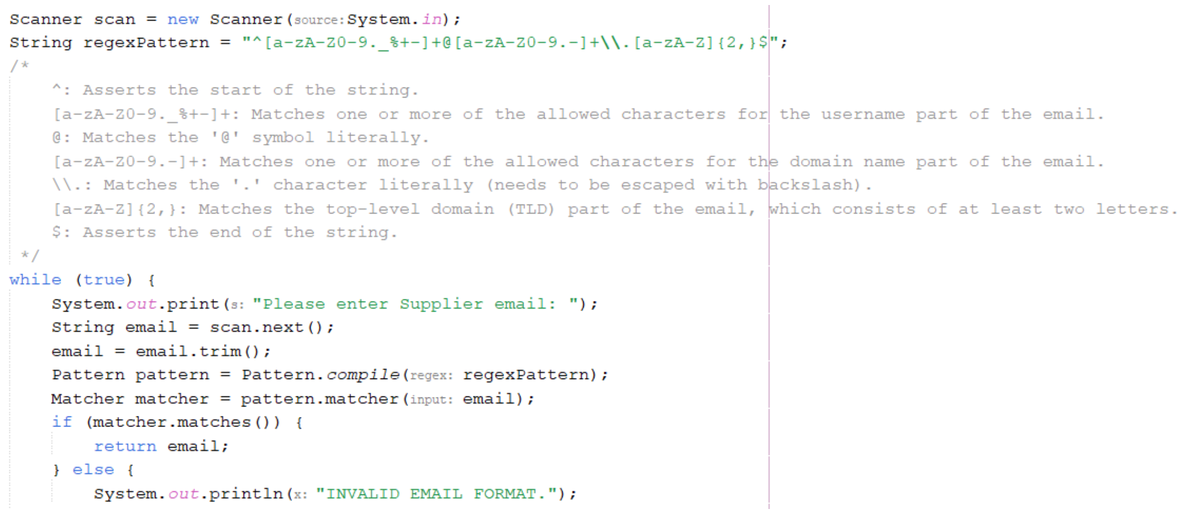


```
Welcome to the Sales Manager Menu!
--------------------------------------
**LOW STOCK REMINDER**
ItemID: I10001 I10002 I00009 I12345 I10008 is lower than 20.
1. Item Functions
2. Supplier Functions
3. Daily Item-wise Sales Entry
4. Create a Purchase Requisition
5. Display Requisition
6. Edit Purchase Requisition
7. Search Item in Purchase Requisition
8. Delete Purchase Requisition
9. List of Purchase Orders
10. Update profile info
11. Logout
```

Figure 151: Output of low stock notification

This function is to notify the sales manager to create a PR of an item if the stock of said item is lower than 20. It will appear on every sales manager menu page. As a distributor they can set whatever amount they want, this is just a example of the low stock notification. We think this is necessary because it can alert the Sales Manager to refill the item.

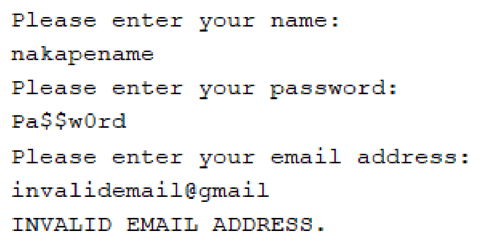
Regrex Validation



```
Scanner scan = new Scanner(source:System.in);
String regexPattern = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}$";
/*
    ^: Asserts the start of the string.
    [a-zA-Z0-9._%+-]+: Matches one or more of the allowed characters for the username part of the email.
    @: Matches the '@' symbol literally.
    [a-zA-Z0-9.-]+: Matches one or more of the allowed characters for the domain name part of the email.
    \\.: Matches the '.' character literally (needs to be escaped with backslash).
    [a-zA-Z]{2,}: Matches the top-level domain (TLD) part of the email, which consists of at least two letters.
    $: Asserts the end of the string.
 */
while (true) {
    System.out.print(s: "Please enter Supplier email: ");
    String email = scan.next();
    email = email.trim();
    Pattern pattern = Pattern.compile(regex: regexPattern);
    Matcher matcher = pattern.matcher(input: email);
    if (matcher.matches()) {
        return email;
    } else {
        System.out.println(x: "INVALID EMAIL FORMAT.");
```

Figure 152 : Example of regrex.

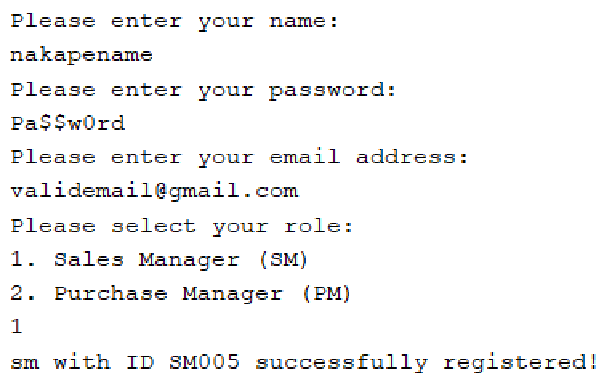
Regrex is used in this system to validate the input of the user. It can specific the string enter by the user and restrict any invalid input. An example of this being used is in the email validation system, where it checks the format of the email to see if it is valid or not. A valid format would consist of an "@" and ending with a ".com".



```
Please enter your name:
nakapename
Please enter your password:
Pa$$w0rd
Please enter your email address:
invalidemail@gmail
INVALID EMAIL ADDRESS.
```

Figure 153: Example of an invalid email format.

As shown in the figure above, it rejects an invalid email format.

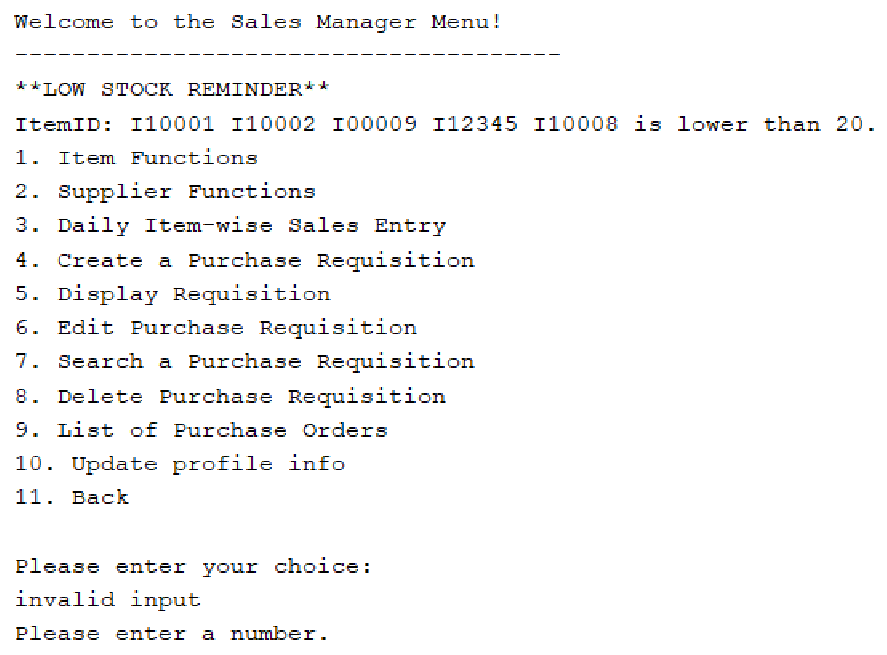


```
Please enter your name:
nakapename
Please enter your password:
Pa$$w0rd
Please enter your email address:
validemail@gmail.com
Please select your role:
1. Sales Manager (SM)
2. Purchase Manager (PM)
1
sm with ID SM005 successfully registered!
```

Figure 154: A valid email format is accepted.

As shown in the figure above contrary to the last image, the system accepts this email format.

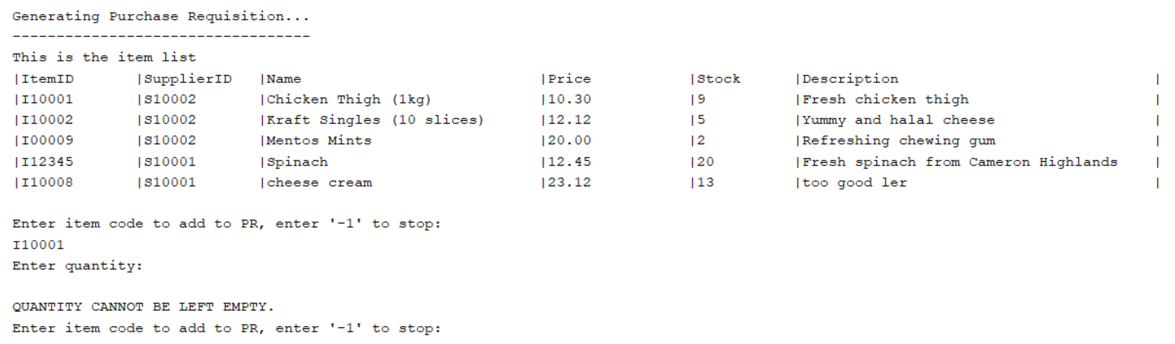
Detection of invalid inputs



```
Welcome to the Sales Manager Menu!
--------------------------------------
**LOW STOCK REMINDER**
ItemID: I10001 I10002 I00009 I12345 I10008 is lower than 20.
1. Item Functions
2. Supplier Functions
3. Daily Item-wise Sales Entry
4. Create a Purchase Requisition
5. Display Requisition
6. Edit Purchase Requisition
7. Search a Purchase Requisition
8. Delete Purchase Requisition
9. List of Purchase Orders
10. Update profile info
11. Back

Please enter your choice:
invalid input
Please enter a number.
```

Figure 155: Entering an invalid input in the menu.

The system also detects any invalid inputs, such as when the user enters a character, string, invalid number or a mixture of all. For example, in the menu of users, if the user enters such an input, the system will reject it.

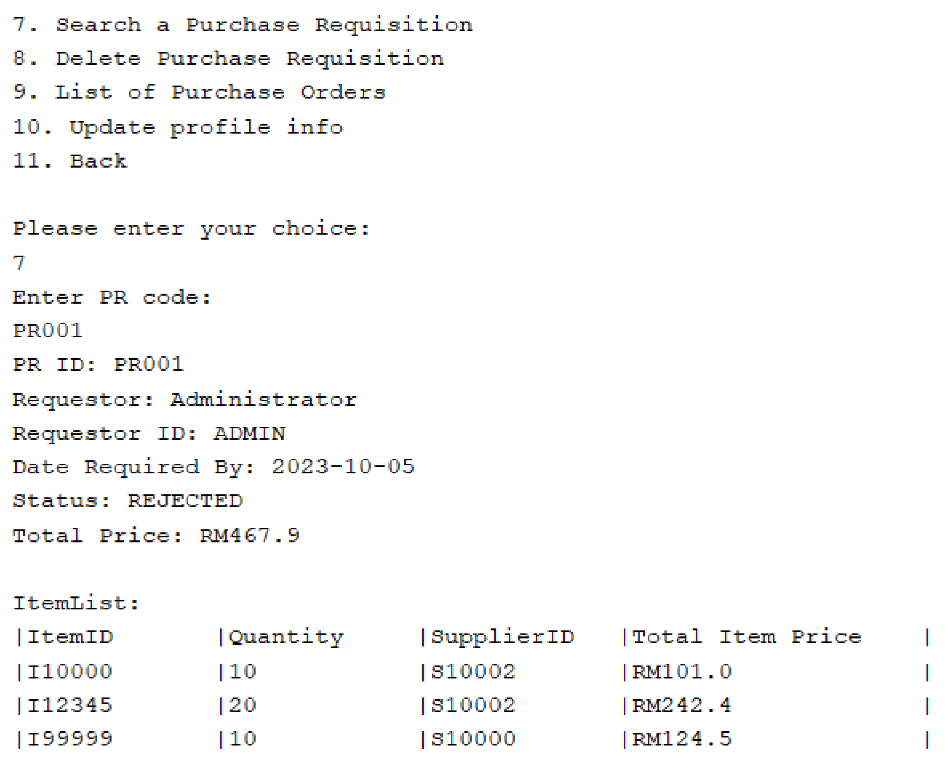


```
Generating Purchase Requisition...
---------------------------------
This is the item list
|ItemID      |SupplierID   |Name                         |Price          |Stock       |Description                          |
|I10001      |S10002       |Chicken Thigh (1kg)          |10.30          |9           |Fresh chicken thigh                  |
|I10002      |S10002       |Kraft Singles (10 slices)    |12.12          |5           |Yummy and halal cheese               |
|I00009      |S10002       |Mentos Mints                 |20.00          |2           |Refreshing chewing gum               |
|I12345      |S10001       |Spinach                      |12.45          |20          |Fresh spinach from Cameron Highlands |
|I10008      |S10001       |cheese cream                 |23.12          |13          |too good ler                         |

Enter item code to add to PR, enter '-1' to stop:
I10001
Enter quantity:

QUANTITY CANNOT BE LEFT EMPTY.
Enter item code to add to PR, enter '-1' to stop:
```

Figure 156: Leaving the input as blank.

This also applies to if the user leaves the quantity as empty or as any of the aforementioned invalid inputs.

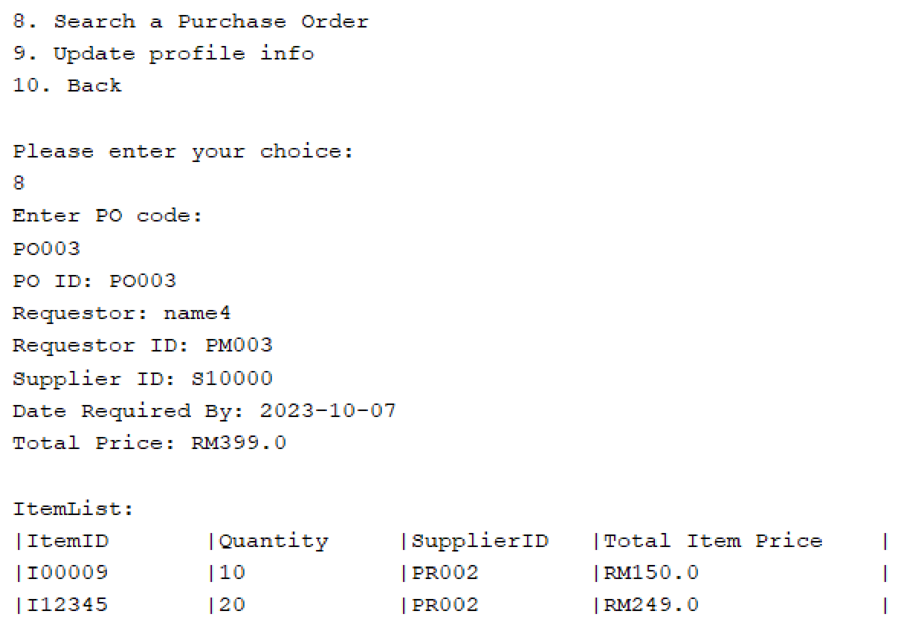
Search Function



```
7. Search a Purchase Requisition
8. Delete Purchase Requisition
9. List of Purchase Orders
10. Update profile info
11. Back

Please enter your choice:
7
Enter PR code:
PR001
PR ID: PR001
Requestor: Administrator
Requestor ID: ADMIN
Date Required By: 2023-10-05
Status: REJECTED
Total Price: RM467.9

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price    |
|I10000        |10            |S10002        |RM101.0             |
|I12345        |20            |S10002        |RM242.4             |
|I99999        |10            |S10000        |RM124.5             |
```

Figure 157: Search function for PR.

Moreover, the system allows the user to search for a specific PR by their ID, so long as it is valid.

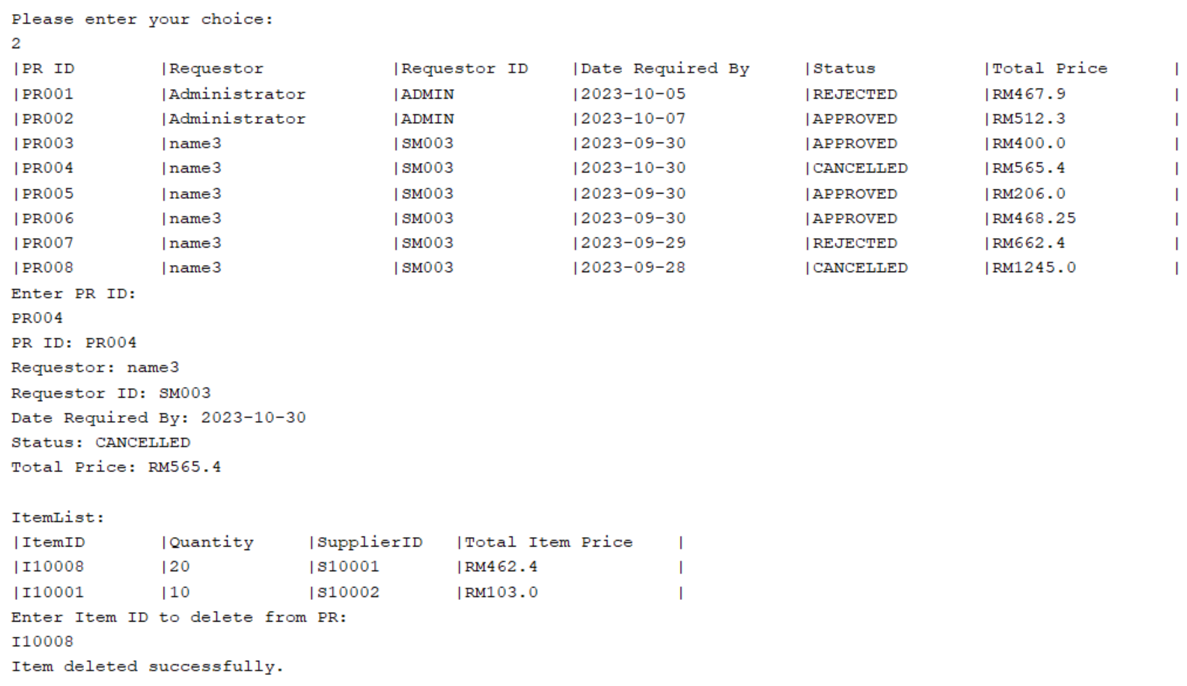


```
8. Search a Purchase Order
9. Update profile info
10. Back

Please enter your choice:
8
Enter PO code:
PO003
PO ID: PO003
Requestor: name4
Requestor ID: PM003
Supplier ID: S10000
Date Required By: 2023-10-07
Total Price: RM399.0

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price    |
|I00009        |10            |PR002         |RM150.0             |
|I12345        |20            |PR002         |RM249.0             |
```

Figure 158: Search function for PO.

This also applies to the PO.

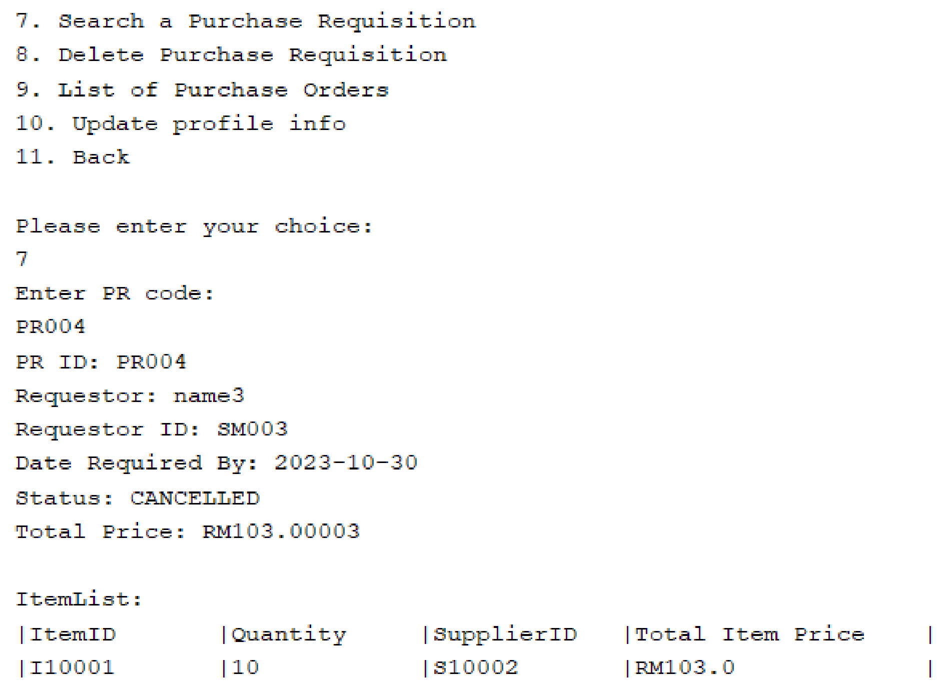
Delete Item from PR



```
Please enter your choice:
2
|PR ID        |Requestor            |Requestor ID    |Date Required By     |Status         |Total Price      |
|PR001        |Administrator        |ADMIN           |2023-10-05           |REJECTED       |RM467.9          |
|PR002        |Administrator        |ADMIN           |2023-10-07           |APPROVED       |RM512.3          |
|PR003        |name3                |SM003           |2023-09-30           |APPROVED       |RM400.0          |
|PR004        |name3                |SM003           |2023-10-30           |CANCELLED      |RM565.4          |
|PR005        |name3                |SM003           |2023-09-30           |APPROVED       |RM206.0          |
|PR006        |name3                |SM003           |2023-09-30           |APPROVED       |RM468.25         |
|PR007        |name3                |SM003           |2023-09-29           |REJECTED       |RM662.4          |
|PR008        |name3                |SM003           |2023-09-28           |CANCELLED      |RM1245.0         |
Enter PR ID:
PR004
PR ID: PR004
Requestor: name3
Requestor ID: SM003
Date Required By: 2023-10-30
Status: CANCELLED
Total Price: RM565.4

ItemList:
|ItemID       |Quantity     |SupplierID   |Total Item Price    |
|I10008       |20           |S10001       |RM462.4             |
|I10001       |10           |S10002       |RM103.0             |
Enter Item ID to delete from PR:
I10008
Item deleted successfully.
```

Figure 159: Deleting an item from PR004.

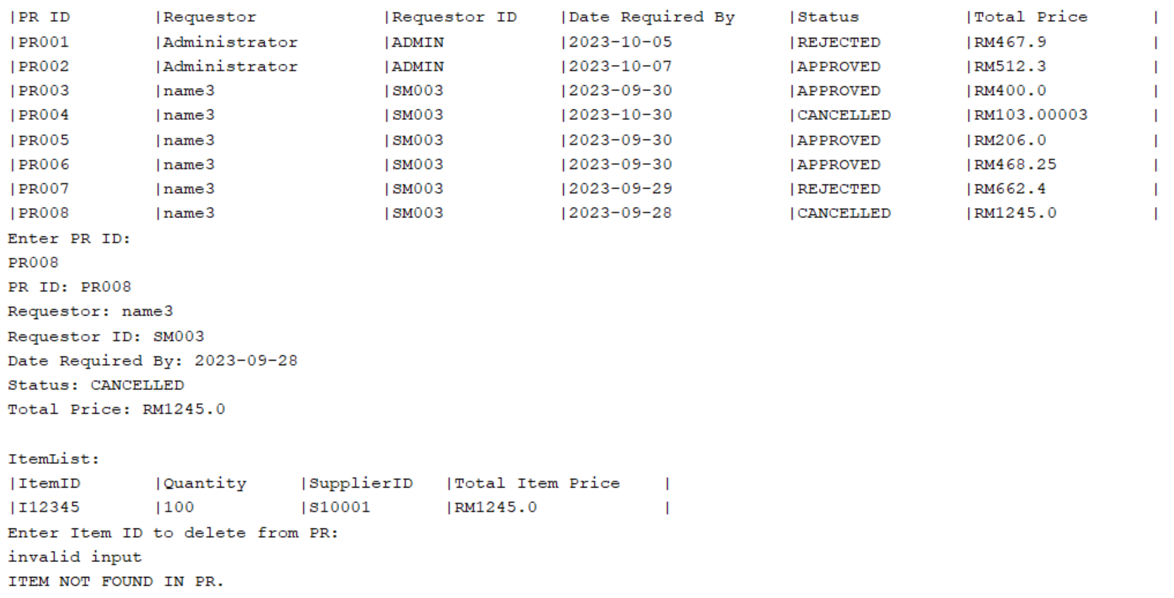
The system also allows the user to delete a specific item from the PR, so long as it exists in the selected PR.



```
7. Search a Purchase Requisition
8. Delete Purchase Requisition
9. List of Purchase Orders
10. Update profile info
11. Back

Please enter your choice:
7
Enter PR code:
PR004
PR ID: PR004
Requestor: name3
Requestor ID: SM003
Date Required By: 2023-10-30
Status: CANCELLED
Total Price: RM103.00003

ItemList:
|ItemID       |Quantity     |SupplierID   |Total Item Price    |
|I10001       |10           |S10002       |RM103.0             |
```

Figure 160: After deleting the item from the PR, the item price is updated.

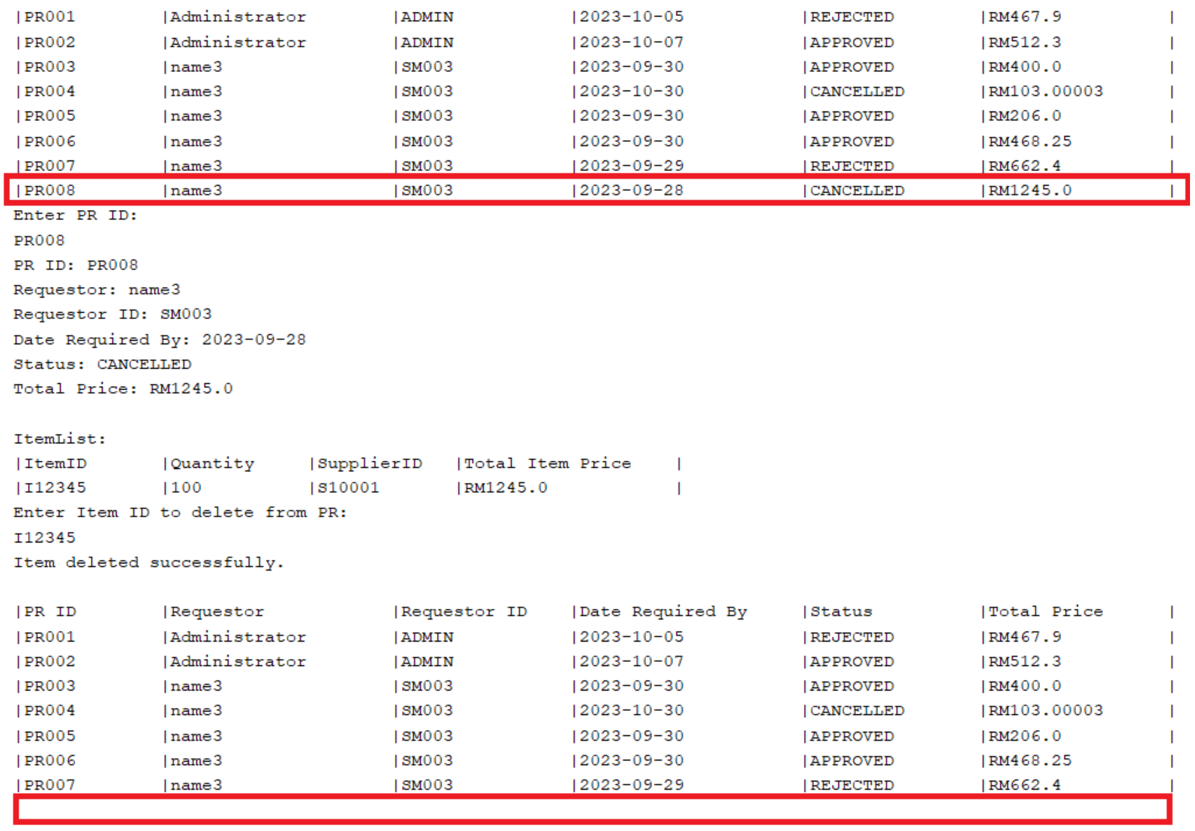
After deleting the item, the total price of the PR will also be updated accordingly.



```
|PR ID         |Requestor             |Requestor ID     |Date Required By      |Status          |Total Price        |
|PR001         |Administrator         |ADMIN            |2023-10-05            |REJECTED        |RM467.9            |
|PR002         |Administrator         |ADMIN            |2023-10-07            |APPROVED        |RM512.3            |
|PR003         |name3                 |SM003            |2023-09-30            |APPROVED        |RM400.0            |
|PR004         |name3                 |SM003            |2023-10-30            |CANCELLED       |RM103.00003        |
|PR005         |name3                 |SM003            |2023-09-30            |APPROVED        |RM206.0            |
|PR006         |name3                 |SM003            |2023-09-30            |APPROVED        |RM468.25           |
|PR007         |name3                 |SM003            |2023-09-29            |REJECTED        |RM662.4            |
|PR008         |name3                 |SM003            |2023-09-28            |CANCELLED       |RM1245.0           |
Enter PR ID:
PR008
PR ID: PR008
Requestor: name3
Requestor ID: SM003
Date Required By: 2023-09-28
Status: CANCELLED
Total Price: RM1245.0

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price    |
|I12345        |100           |S10001        |RM1245.0            |
Enter Item ID to delete from PR:
invalid input
ITEM NOT FOUND IN PR.
```

Figure 161: The system rejects the input as the item ID does not exist in the PR.

The system will also reject the input if the item ID is not found in the PR.

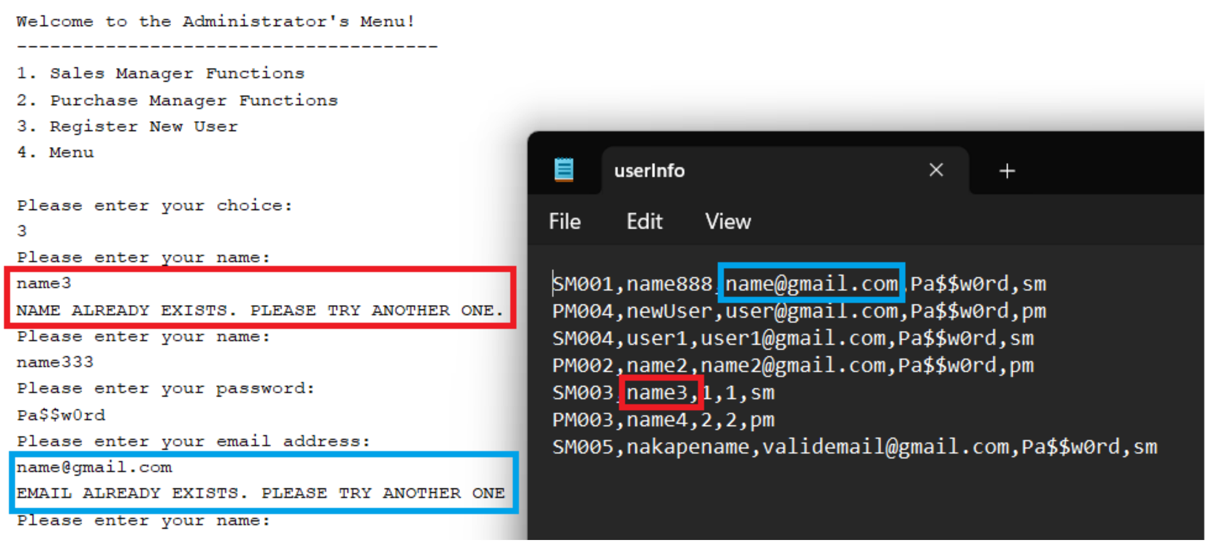


```
|PR001         |Administrator         |ADMIN            |2023-10-05            |REJECTED        |RM467.9            |
|PR002         |Administrator         |ADMIN            |2023-10-07            |APPROVED        |RM512.3            |
|PR003         |name3                 |SM003            |2023-09-30            |APPROVED        |RM400.0            |
|PR004         |name3                 |SM003            |2023-10-30            |CANCELLED       |RM103.00003        |
|PR005         |name3                 |SM003            |2023-09-30            |APPROVED        |RM206.0            |
|PR006         |name3                 |SM003            |2023-09-30            |APPROVED        |RM468.25           |
|PR007         |name3                 |SM003            |2023-09-29            |REJECTED        |RM662.4            |
|PR008         |name3                 |SM003            |2023-09-28            |CANCELLED       |RM1245.0           |
Enter PR ID:
PR008
PR ID: PR008
Requestor: name3
Requestor ID: SM003
Date Required By: 2023-09-28
Status: CANCELLED
Total Price: RM1245.0

ItemList:
|ItemID        |Quantity      |SupplierID    |Total Item Price    |
|I12345        |100           |S10001        |RM1245.0            |
Enter Item ID to delete from PR:
I12345
Item deleted successfully.

|PR ID         |Requestor             |Requestor ID     |Date Required By      |Status          |Total Price        |
|PR001         |Administrator         |ADMIN            |2023-10-05            |REJECTED        |RM467.9            |
|PR002         |Administrator         |ADMIN            |2023-10-07            |APPROVED        |RM512.3            |
|PR003         |name3                 |SM003            |2023-09-30            |APPROVED        |RM400.0            |
|PR004         |name3                 |SM003            |2023-10-30            |CANCELLED       |RM103.00003        |
|PR005         |name3                 |SM003            |2023-09-30            |APPROVED        |RM206.0            |
|PR006         |name3                 |SM003            |2023-09-30            |APPROVED        |RM468.25           |
|PR007         |name3                 |SM003            |2023-09-29            |REJECTED        |RM662.4            |
```

Figure 162: Deleting a PR with one item deletes the whole PR as there is no more use for it.

If the PR contains one item, and the item from that PR is deleted, the whole PR will also be deleted. In the figure above, we delete PR008, which contained one item only. As shown in the boxes with the red borders, PR008 is gone from the list as it is completely deleted as per the logic.

Detection of duplicate names and emails

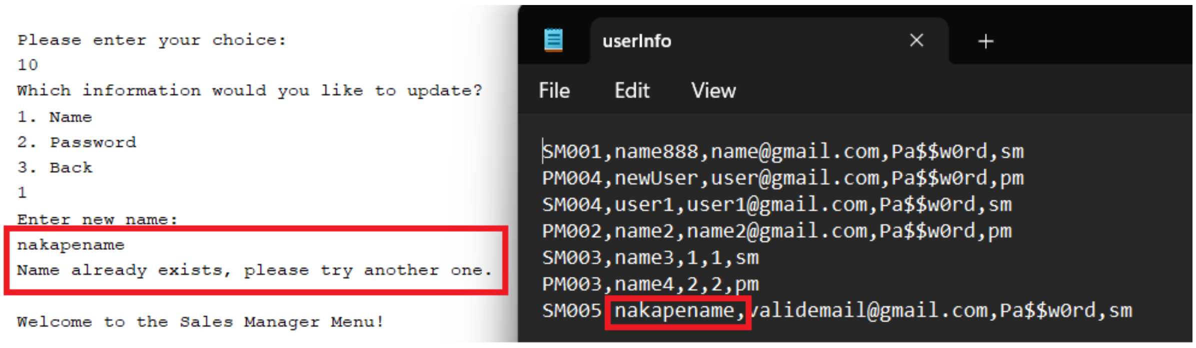


```
Welcome to the Administrator's Menu!
-------------------------------------
1. Sales Manager Functions
2. Purchase Manager Functions
3. Register New User
4. Menu

Please enter your choice:
3
Please enter your name:
name3
NAME ALREADY EXISTS. PLEASE TRY ANOTHER ONE.
Please enter your name:
name333
Please enter your password:
Pa$$w0rd
Please enter your email address:
name@gmail.com
EMAIL ALREADY EXISTS. PLEASE TRY ANOTHER ONE
Please enter your name:
```

```
SM001,name888,name@gmail.com,Pa$$w0rd,sm
PM004,newUser,user@gmail.com,Pa$$w0rd,pm
SM004,user1,user1@gmail.com,Pa$$w0rd,sm
PM002,name2,name2@gmail.com,Pa$$w0rd,pm
SM003,name3,1,1,sm
PM003,name4,2,2,pm
SM005,nakapename,validemail@gmail.com,Pa$$w0rd,sm
```

Figure 163: System will not allow for duplicate names and emails when registering for new users.

Additionally, the system will not allow for duplicate names and emails, as highlighted in the boxes with the red border and blue border respectively.



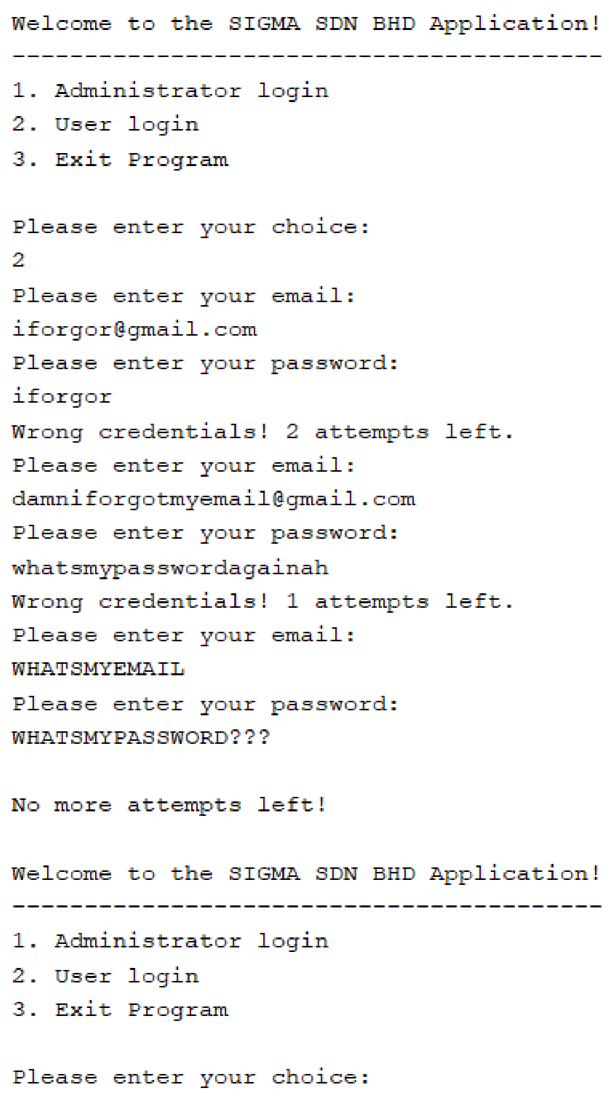
```
Please enter your choice:
10
Which information would you like to update?
1. Name
2. Password
3. Back
1
Enter new name:
nakapename
Name already exists, please try another one.

Welcome to the Sales Manager Menu!
```

```
SM001,name888,name@gmail.com,Pa$$w0rd,sm
PM004,newUser,user@gmail.com,Pa$$w0rd,pm
SM004,user1,user1@gmail.com,Pa$$w0rd,sm
PM002,name2,name2@gmail.com,Pa$$w0rd,pm
SM003,name3,1,1,sm
PM003,name4,2,2,pm
SM005,nakapename,validemail@gmail.com,Pa$$w0rd,sm
```

Figure 164: The system also does not allow for duplicate names when updating user information.

The same logic is applied when the user attempts to change their name into a name that already exists in the database.

Three-attempt verification system



```
Welcome to the SIGMA SDN BHD Application!
-----------------------------------------
1. Administrator login
2. User login
3. Exit Program

Please enter your choice:
2
Please enter your email:
iforgor@gmail.com
Please enter your password:
iforgor
Wrong credentials! 2 attempts left.
Please enter your email:
damniforgotmyemail@gmail.com
Please enter your password:
whatsmypasswordagainah
Wrong credentials! 1 attempts left.
Please enter your email:
WHATSMYEMAIL
Please enter your password:
WHATSMYPASSWORD???

No more attempts left!

Welcome to the SIGMA SDN BHD Application!
-----------------------------------------
1. Administrator login
2. User login
3. Exit Program

Please enter your choice:
```

Figure 165: The user is given three chances to login with their correct credentials.

The system also employs a three-attempt verification system when the user attempts to login to their account. This means that the user is given three attempts to login with the correct credentials, if they don’t meet login after those three chances, they will be redirected back to the main menu.

8.0 References

Debnath, M. (2017, October 9). *Modularity in Java: Java 9 Modularity versus Prior Versions*. Retrieved from developer.com: https://www.developer.com/design/modularity-in-java-java-9-modularity-versus-prior-versions/

Debnath, M. (2019, March 19). *Diving Deeper into Polymorphism and its Benefits in Java*. Retrieved from developer.com: https://www.developer.com/design/diving-deeper-into-polymorphism-and-its-benefits-in-java/

GeeksforGeeks. (2023, April 17). *Inheritance in Java*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/inheritance-in-java/

GeeksforGeeks.org. (n.d.). Hierarchal Inheritance. GeeksforGeeks.org.

Inheritance in Java. (n.d.). Retrieved from Javatpoint: https://www.javatpoint.com/inheritance-in-java

miglani, g. (n.d.). *Abstraction in Java*. Retrieved from GeeksforGeeks: https://www.geeksforgeeks.org/abstraction-in-java-2/

Programiz. (n.d.). *Java Polymorphism*. Retrieved from Programiz: https://www.programiz.com/java-programming/polymorphism

S, R. A. (2023, May 18). *What is Encapsulation in Java and How to Implement It*. Retrieved from simplilearn: https://www.simplilearn.com/tutorials/java-tutorial/java-encapsulation

Tech Target Contributor. (2016, July). *encapsulation in Java*. Retrieved from TheServerSide: https://www.theserverside.com/definition/encapsulation-in-Java

Vadapalli, P. (2022, January 29). *Runtime Polymorphism in Java with Examples*. Retrieved from upgrad: https://www.upgrad.com/blog/runtime-polymorphism-java-examples/