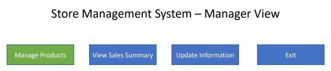
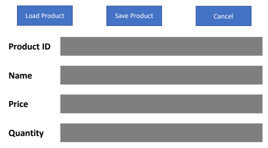
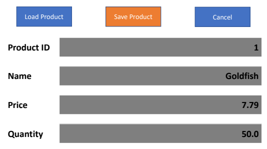
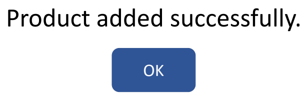
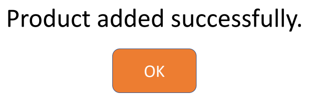
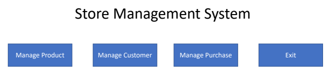
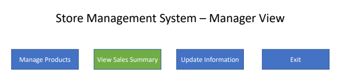
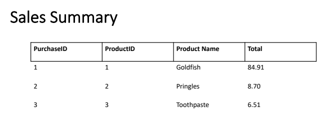
Manager View – Save Product

User saves a product to the database.

Client	System	Server
Click the “Manage Product” button  Store Management System – Manager View Manage Products View Sales Summary Update Information Exit	Display “Manage Product” Screen  Load Product Save Product Cancel Product ID Name Price Quantity	
Fill in the all of the product information and click the “Save Product” button  Product ID 1 Name Goldfish Price 7.79 Quantity 50.0	A message will be displayed showing the action was successful  Product added successfully.	The client sends the server the populated ProductModel as data in a MessageModel with the PUT_PRODUCT code through Gson. The server checks to see if the product id matches a preexisting product id. If it does, then the old product is deleted. The product is then inserted into the database and the server sends back confirmation if the insertion was a success.
Click the “OK” button  Product added successfully.	Display the Home Screen  Store Management System Manage Product Manage Customer Manage Purchase Exit	

Manager View – Sales Summary

User saves a product to the database.

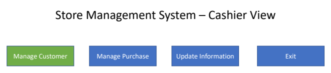
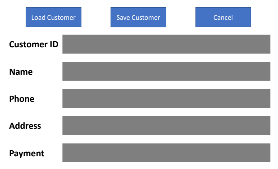
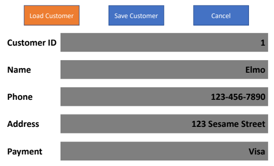
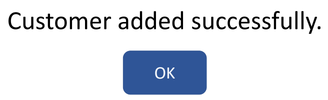
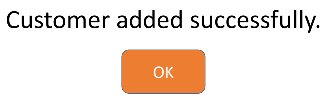
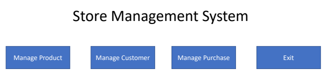
Client	**System**	**Server**
Click the “View Sales Summary” button  Store Management System – Manager View Manage Products View Sales Summary Update Information Exit	Display “Sales Summary” Screen 	The client requests a full list of purchases from the server with the code, GET_FULL_PURCHASE_LIST. The server sends back a PurchaseListModel and is displayed.

Sales Summary

PurchaseID	ProductID	Product Name	Total
1	1	Goldfish	84.91
2	2	Pringles	8.70
3	3	Toothpaste	6.51

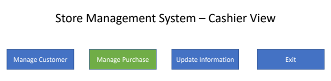
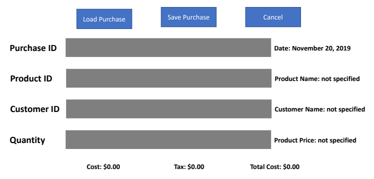
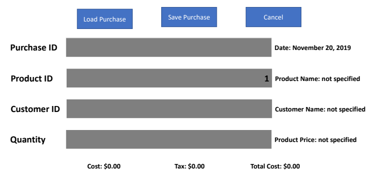
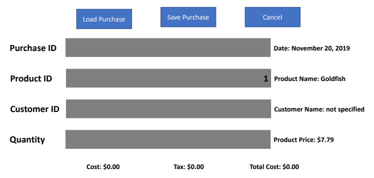
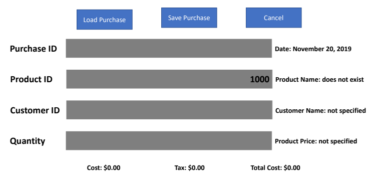
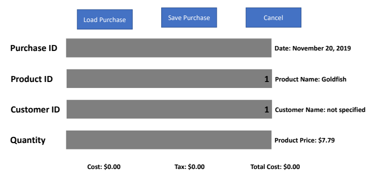
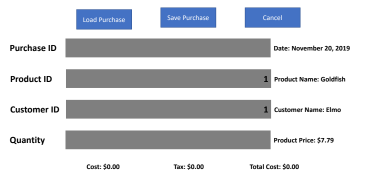
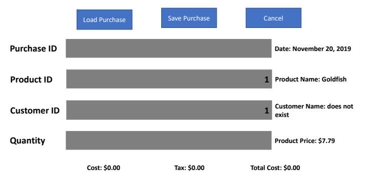
Cashier View – Save Customer

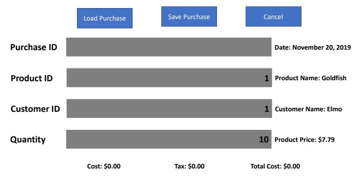
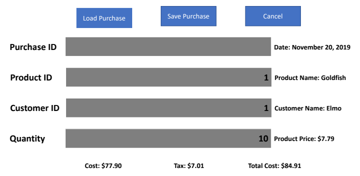
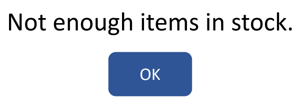
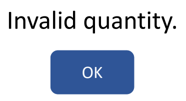
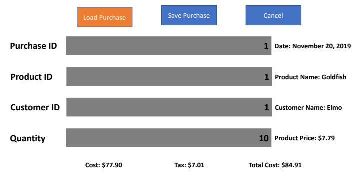
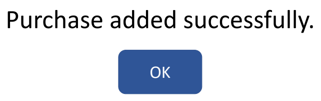
User saves a customer to the database.

Client	**System**	**Server**
Click the "Manage Customer" button  Store Management System – Cashier View Manage Customer Manage Purchase Update Information Exit	Display "Manage Customer" Screen  Load Customer Save Customer Cancel Customer ID Name Phone Address Payment	
Fill in the all of the customer information and click the "Save Customer" button  Customer ID 1 Name Elmo Phone 123-456-7890 Address 123 Sesame Street Payment Visa	A message will be displayed showing the action was successful  Customer added successfully. 	The client sends the server the populated CustomerModel as data in a MessageModel with the PUT_CUSTOMER code through Gson. The server checks to see if the customer id matches a preexisting customer id. If it does, then the old customer is deleted. The customer is then inserted into the database and the server sends back confirmation if the insertion was a success.
Click the "OK" button  Customer added successfully. 	Display the Home Screen  Store Management System Manage Product Manage Customer Manage Purchase Exit	

Cashier View – Save Purchase

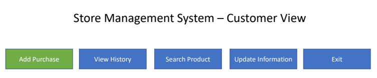
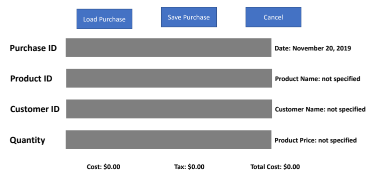
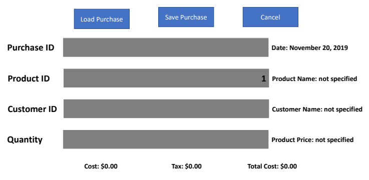
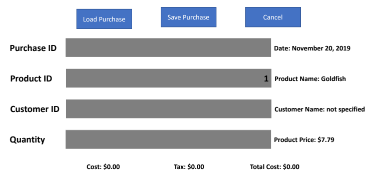
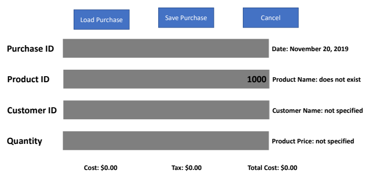
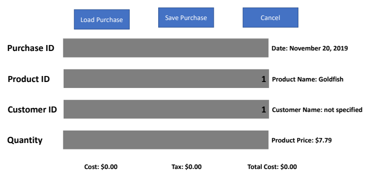
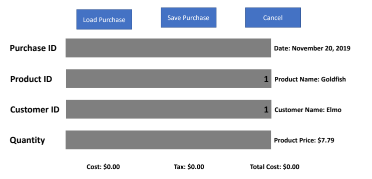
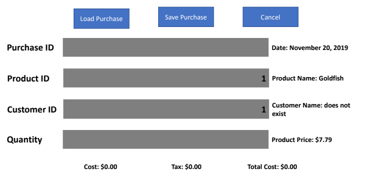
User saves a purchase to the database.

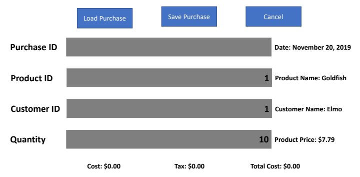
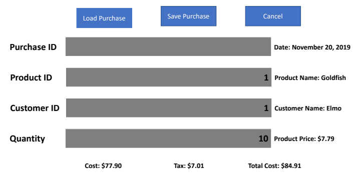
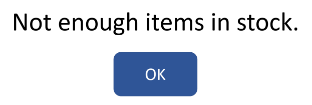
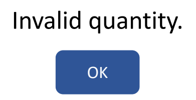
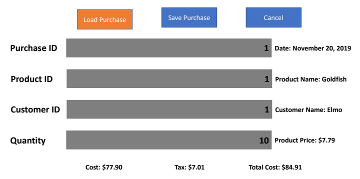
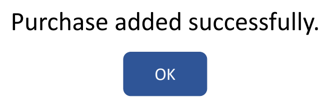
Client	**System**	**Server**
Click the "Manage Purchase" button  Store Management System – Cashier View Manage Customer Manage Purchase Update Information Exit	Display "Manage Purchase" Screen  Purchase ID Date: November 20, 2019 Product ID Product Name: not specified Customer ID Customer Name: not specified Quantity Product Price: not specified Cost: $0.00 Tax: $0.00 Total Cost: $0.00	
Fill in the "Product ID" field  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: not specified Customer ID Customer Name: not specified Quantity Product Price: not specified Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The "Product Name" and "Product Price" labels update  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID Customer Name: not specified Quantity Product Price: $7.79 Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The client sends the server the product id and the server sends back the corresponding product from the database. If the product does not exist, the label will be filled accordingly:  Purchase ID Date: November 20, 2019 Product ID 1000 Product Name: does not exist Customer ID Customer Name: not specified Quantity Product Price: not specified Cost: $0.00 Tax: $0.00 Total Cost: $0.00
Fill in the "Customer ID" field  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: not specified Quantity Product Price: $7.79 Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The "Customer Name" label updates  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity Product Price: $7.79 Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The client sends the server the customer id and the server sends back the corresponding customer from the database. If the product does not exist, the label will be filled accordingly:  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: does not exist Quantity Product Price: $7.79 Cost: $0.00 Tax: $0.00 Total Cost: $0.00

Fill in the “Quantity” field  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity 10 Product Price: $7.79 Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The “Cost”, “Tax”, and “Total Cost” labels update  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity 10 Product Price: $7.79 Cost: $77.90 Tax: $7.01 Total Cost: $84.91 If there are not enough items in stock, a message displays:  Not enough items in stock. If the quantity is invalid:  Invalid quantity. 	
Fill in the “Purchase ID” field and click the “Save Purchase” button  Purchase ID 1 Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity 10 Product Price: $7.79 Cost: $77.90 Tax: $7.01 Total Cost: $84.91	Display the Home Screen  Purchase added successfully. 	The client sends the server the populated PurchaseModel as data in a MessageModel with the PUT_PURCHASE code through Gson. The server checks to see if the purchase id matches a preexisting purchase id. If it does, then the old purchase is deleted. The purchase is then inserted into the database and the server sends back confirmation if the insertion was a success.

Customer View – Add Purchase

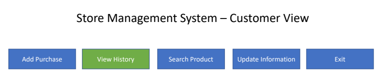
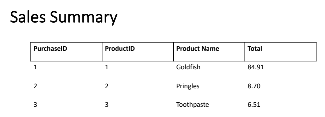
User saves a purchase to the database.

Client	**System**	**Server**
Click the "Manage Purchase" button  Store Management System – Customer View Add Purchase View History Search Product Update Information Exit	Display "Manage Purchase" Screen  Purchase ID: Date: November 20, 2019; Product ID: Product Name: not specified; Customer ID: Customer Name: not specified; Quantity: Product Price: not specified; Cost: $0.00 Tax: $0.00 Total Cost: $0.00	
Fill in the "Product ID" field  Purchase ID: Date: November 20, 2019; Product ID: 1 Product Name: not specified; Customer ID: Customer Name: not specified; Quantity: Product Price: not specified; Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The "Product Name" and "Product Price" labels update  Purchase ID: Date: November 20, 2019; Product ID: 1 Product Name: Goldfish; Customer ID: Customer Name: not specified; Quantity: Product Price: $7.79; Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The client sends the server the product id and the server sends back the corresponding product from the database. If the product does not exist, the label will be filled accordingly:  Purchase ID: Date: November 20, 2019; Product ID: 1000 Product Name: does not exist; Customer ID: Customer Name: not specified; Quantity: Product Price: not specified; Cost: $0.00 Tax: $0.00 Total Cost: $0.00
Fill in the "Customer ID" field  Purchase ID: Date: November 20, 2019; Product ID: 1 Product Name: Goldfish; Customer ID: 1 Customer Name: not specified; Quantity: Product Price: $7.79; Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The "Customer Name" label updates  Purchase ID: Date: November 20, 2019; Product ID: 1 Product Name: Goldfish; Customer ID: 1 Customer Name: Elmo; Quantity: Product Price: $7.79; Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The client sends the server the customer id and the server sends back the corresponding customer from the database. If the product does not exist, the label will be filled accordingly:  Purchase ID: Date: November 20, 2019; Product ID: 1 Product Name: Goldfish; Customer ID: 1 Customer Name: does not exist; Quantity: Product Price: $7.79; Cost: $0.00 Tax: $0.00 Total Cost: $0.00

Fill in the “Quantity” field  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity 10 Product Price: $7.79 Cost: $0.00 Tax: $0.00 Total Cost: $0.00	The “Cost”, “Tax”, and “Total Cost” labels update  Purchase ID Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity 10 Product Price: $7.79 Cost: $77.90 Tax: $7.01 Total Cost: $84.91 If there are not enough items in stock, a message displays:  Not enough items in stock. If the quantity is invalid:  Invalid quantity. 	
Fill in the “Purchase ID” field and click the “Save Purchase” button  Purchase ID 1 Date: November 20, 2019 Product ID 1 Product Name: Goldfish Customer ID 1 Customer Name: Elmo Quantity 10 Product Price: $7.79 Cost: $77.90 Tax: $7.01 Total Cost: $84.91	Display the Home Screen  Purchase added successfully. 	The client sends the server the populated PurchaseModel as data in a MessageModel with the PUT_PURCHASE code through Gson. The server checks to see if the purchase id matches a preexisting purchase id. If it does, then the old purchase is deleted. The purchase is then inserted into the database and the server sends back confirmation if the insertion was a success.

Customer View – Purchase History

User saves a product to the database.

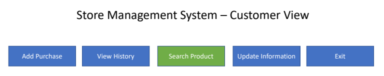
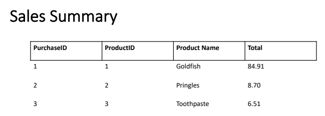
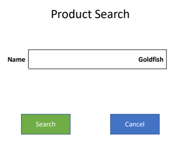
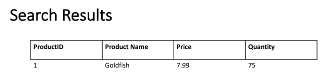
Client	**System**	**Server**
Click the “View Sales Summary” button 	Display “Sales Summary” Screen 	The client requests a full list of purchases from the server with the code, GET_PURCHASE_LIST. The server sends back a PurchaseListModel and is displayed.

Sales Summary

PurchaseID	ProductID	Product Name	Total
1	1	Goldfish	84.91
2	2	Pringles	8.70
3	3	Toothpaste	6.51

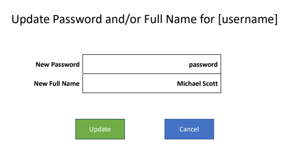
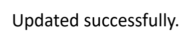
Customer View – Search Product

User saves a product to the database.

Client	**System**	**Server**
Click the “View Sales Summary” button  Store Management System – Customer View Add Purchase View History Search Product Update Information Exit	Display “Product Search” Screen  Sales Summary PurchaseID | ProductID | Product Name | Total 1 | 1 | Goldfish | 84.91 2 | 2 | Pringles | 8.70 3 | 3 | Toothpaste | 6.51	
Fill in the information and click the “Search” button.  Product Search Name: Goldfish 	Display the found product information.  Search Results ProductID | Product Name | Price | Quantity 1 | Goldfish | 7.99 | 75	The client requests a full list of purchases from the server with the code, GET_PRODUCT_LIST. The server sends back a ProductListModel and is displayed.

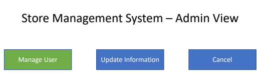
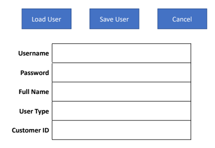
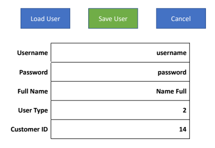
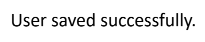
User – Update Information

User saves a product to the database.

Client	System	Server
Fill out the “New Password” and “New Full Name” fields and click the “Update” button.  Update Password and/or Full Name for [username]	Display “Updated Successfully” screen.  Updated successfully.	The client sends the server the UserModel with the PUT_USER code. The user is put in the table with the new updated password or name. The server sends back a confirmation code.

Admin View – Manage User

User saves a customer to the database.

Client	**System**	**Server**
Click the "Manage User" button  Store Management System – Admin View	Display "Manage User" Screen  Load User Save User Cancel Username Password Full Name User Type Customer ID	
Fill in the all of the user information and click the "Save User" button  Username: username Password: password Full Name: Name Full User Type: 2 Customer ID: 14	A message will be displayed showing the action was successful  User saved successfully.	The client sends the server the populated UserModel as data in a MessageModel with the PUT_USER code through Gson. The server checks to see if the username matches a preexisting username. If it does, then the old user is deleted. The user is then inserted into the database and the server sends back confirmation if the insertion was a success.