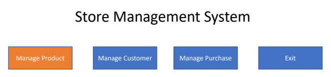
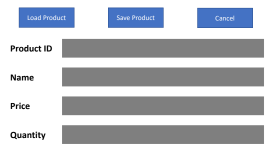
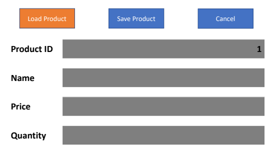
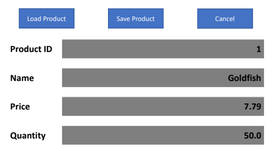
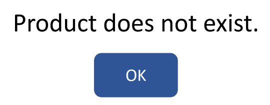
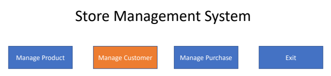
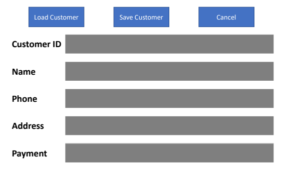
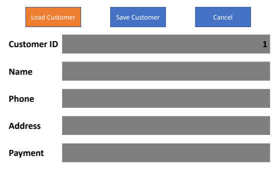
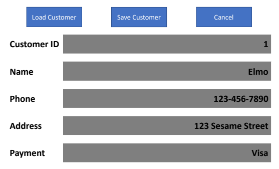
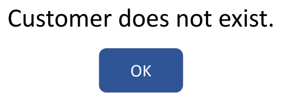
Manage Product – Load

User loads an existing product's information from the database.

Client	**System**	**Server**
Click the "Manage Product" button  Store Management System Manage Product Manage Customer Manage Purchase Exit	Display "Manage Product" Screen  Load Product Save Product Cancel Product ID Name Price Quantity	
Fill in the "Product ID" field and click the "Load Product" button  Product ID 1 Name Price Quantity	Display the loaded product information in the corresponding fields  Product ID 1 Name Goldfish Price 7.79 Quantity 50.0	The client sends the server a ProductModel with the product id populated as data in a MessageModel with the GET_PRODUCT code and the server sends back the corresponding product from the database. If the product does not exist, this error message will be displayed:  Product does not exist.

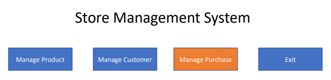
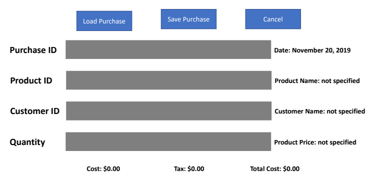
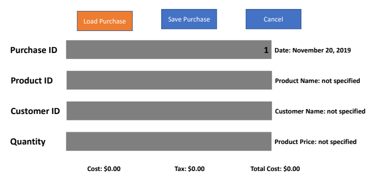
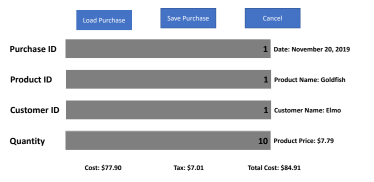
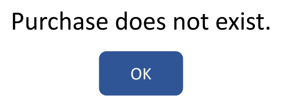
Manage Customer – Load

User loads an existing customer’s information from the database.

Client	**System**	**Server**
Click the “Manage Customer” button  Store Management System Manage Product Manage Customer Manage Purchase Exit	Display “Manage Customer” Screen  Load Customer Save Customer Cancel Customer ID Name Phone Address Payment	
Fill in the “Customer ID” field and click the “Load Customer” button  Customer ID: 1 Name: Phone: Address: Payment:	Display the loaded customer information in the corresponding fields  Customer ID: 1 Name: Elmo Phone: 123-456-7890 Address: 123 Sesame Street Payment: Visa	The client sends the server a CustomerModel with the customer id populated as data in a MessageModel with the GET_CUSTOMER code and the server sends back the corresponding customer from the database. If the customer does not exist, this error message will be displayed:  Customer does not exist.

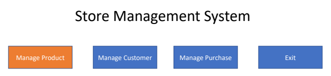
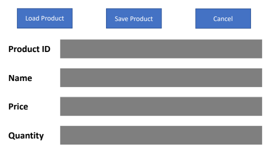
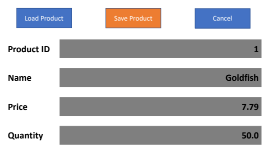
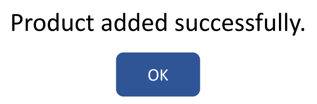
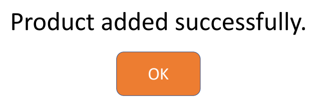
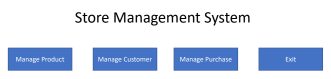
Manage Purchase – Load

User loads an existing purchase’s information from the database.

Client	**System**	**Server**
Click the “Manage Purchase” button  Store Management System Manage Product Manage Customer Manage Purchase 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 “Purchase ID” field and click the “Load Purchase” button  Purchase ID 1 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	Display the loaded purchase information in the corresponding fields and populate the label information.  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	The client sends the server a PurchaseModel with the purchase id populated as data in a MessageModel with the GET_PURCHASE code and the server sends back the corresponding purchase from the database. If the purchase does not exist, this error message will be displayed:  Purchase does not exist.

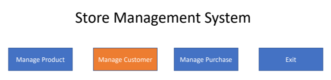
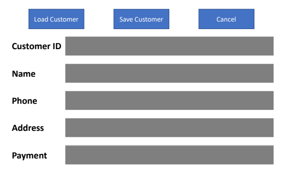
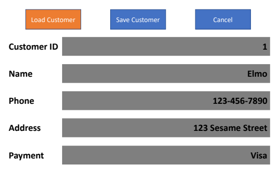
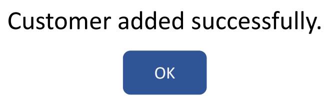
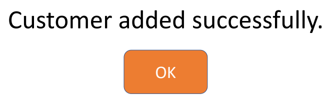
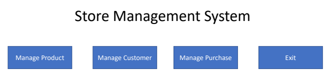
Manage Product – Save

User saves a product to the database.

Client	**System**	**Server**
Click the “Manage Product” button  Store Management System Manage Product Manage Customer Manage Purchase 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	

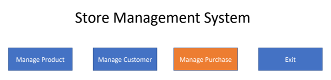
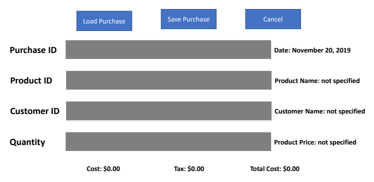
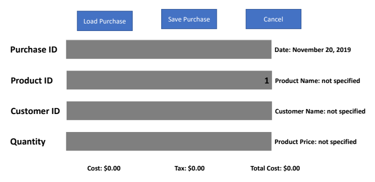
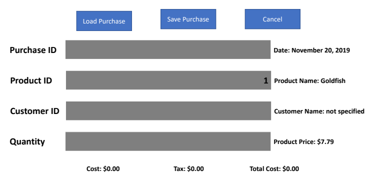
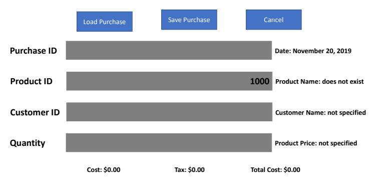
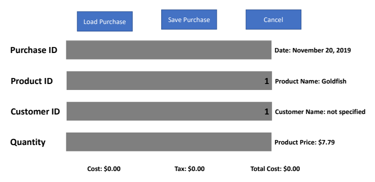
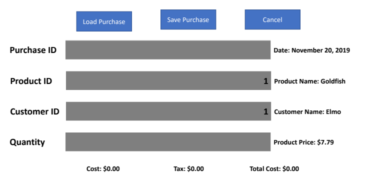
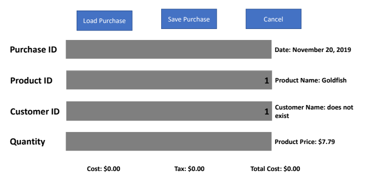
Manage Customer – Save

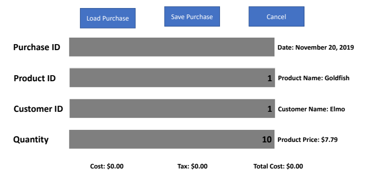
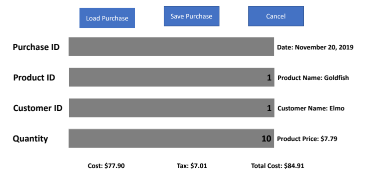
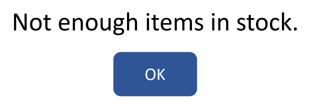
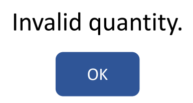
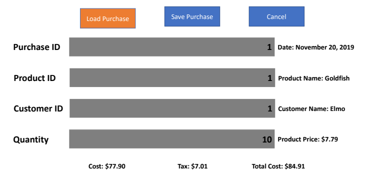
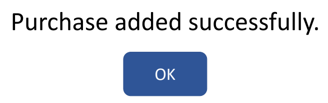
User saves a customer to the database.

Client	**System**	**Server**
Click the "Manage Customer" button  Store Management System Manage Product Manage Customer Manage Purchase 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	

Manage Purchase – Save

User saves a purchase to the database.

Client	System	Server
Click the "Manage Purchase" button  Store Management System Manage Product · Manage Customer · Manage Purchase · Exit	Display "Manage Purchase" Screen  Load Purchase · Save Purchase · Cancel 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  Load Purchase · Save Purchase · Cancel 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  Load Purchase · Save Purchase · Cancel 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:  Load Purchase · Save Purchase · Cancel 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  Load Purchase · Save Purchase · Cancel 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  Load Purchase · Save Purchase · Cancel 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:  Load Purchase · Save Purchase · Cancel 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.