Server side:

1. Transaction DB table creation and Product DB table modification (database)
2. Transaction exist check API (server)
3. Create the rest controller for the Transaction.
4. Make API that takes in Transaction ID object and return TransactionINFO object when found, throw exception otherwise. This task is uses for return transaction.
5. Transaction id class only contains a transaction id field.
6. TransactionINFO class contain transaction id, amount, transection type field.
7. Transaction class, TransactionEntity class, and createTransaction API (server)
8. getAllActiveProduct API (server)

Client side:

\*NOTE: 1.a.i.1) product view functionality can be the same as 2.a.i) product view functionality

1. StartTransactionActivity
   1. List all products
      1. On click 🡪Invoke product view
         1. Change cart quantity functionality
2. Create Shopping cart activity page UI – showing products
   1. Invoke Product View on click button (switch views)
      1. Change cart Quantity functionality (edit text)
   2. Checkout button – sends transaction information to server.
3. Return vs new sale page
   1. Return page
      1. Send Receipt ID check to server
   2. If successful go to return transaction page
4. Checkout page (finalize), if ok 🡪 Summary Page
5. Summary page (with receipt NO.)
6. Changes to ProductViewActivity to allow add/remove product from cart (edit text).

TASKS

\*\*\*\*\*These are approximate descriptions of the tasks that need to be implemented. Whoever works on each task will need to add appropriate content/alertdialogs/more\_Stuff and communicate with the other team members and decide on how the pieced fit together\*\*\*\*\*

1. Create a TransactionTransition class to be used for passing the following information between the activities. It should probably have the following attributes:
   1. An arraylist of products contained in the cart and their associated information
   2. Double type variable: Total price of cart
   3. Integer: Employee\_ID
   4. Integer for the reference transaction ID if the transaction is a return.
   5. Boolean variable for Sale or return EX: ‘0’=sale, ‘1’=return
2. Modify the HomeScreenActivity
   1. Add two buttons: one for sale and one for return
      1. The return button will create an Async task to query the server to see if the transaction ID entered within the alertDialog popup is a SALE in the database.
         1. If the entered transaction ID is valid, it should add it to the TransactionTransition object and change the transaction type to return. It should also pass this transactionTransition object to the Start Transation screen and go to this screen.
         2. If the entered transaction ID is not valid, perhaps display an alert dialog and tell the user to enter a valid one
      2. If the sale button is selected, you should create a transactionTransition object and pass it to the start transation screen
3. Create the Start transaction screen
   1. List the products and their information
   2. Include a cancel button to return to the homescreen
   3. Include a checkout button to go to the summary screen
   4. Include the running total for the cart
   5. Include an add item button that transitions to the product view screen
4. Modify the product view screen
   1. We are going to use the product view activity to select items to add to the cart.
   2. You should include a add button next to each item.
      1. When this button is pressed, it will transition to the change quantity screen.
5. Create a change quantity screen that has:
   1. An edit text to enter quantity
   2. A confirm button to update the quantity of the object
   3. Update the quantity of the product that was selected in the productView within the TransactionTransition object. Change to the start activity screen when you hit the confirm button
6. Create a summary screen that serves as the receipt for the transaction
   1. It should have the total amount, the transaction ID, and other info as you see fit
   2. It should have a okay button that moves to the homeScreenActivity.
7. Create the service and all of the classes/methods associated with checking the entered transaction ID for returns and getting the transaction information from the server. Includes TransactionID, total amount, etc
8. Create the service and all of the classes for sending the transaction information to the server for storage when the transaction completes. You should get the transaction ID (generated on the server) from server so that It can be displayed on the summary screen.