Development Report

AsherSportsConsortium Application Report

Object Oriented Programming (CIS4037-N-FJ1-2022)

Student Id : Q2078619 Akshay Kumar

May 7, 2023

Teesside University, Middlesbrough

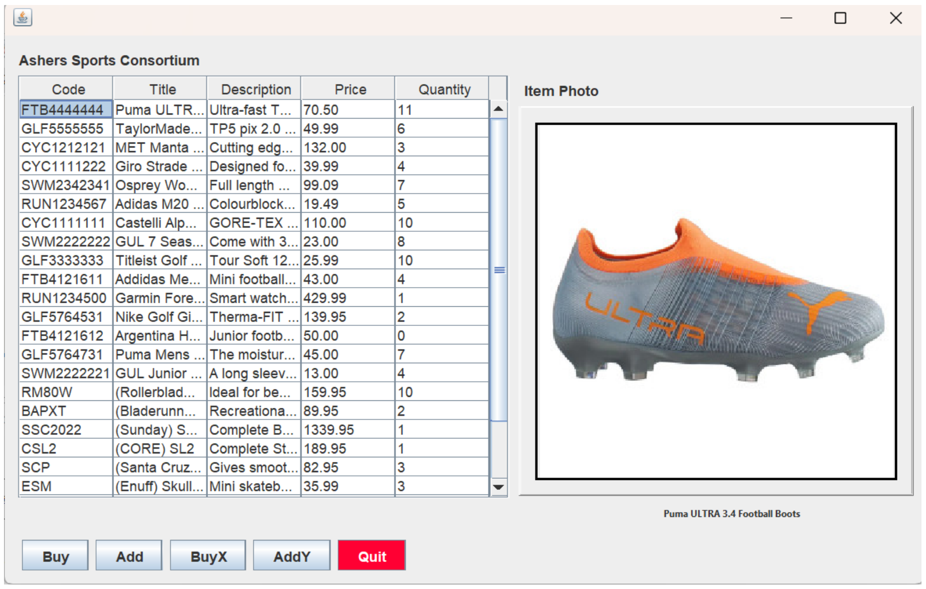
Contents

1 Introduction

In this project, we have to convert the existing console application to GUI application and some GUI feature and merge the TeesideStake System that combines two existing applications, including loading data from multiple input files, updating the GUI, displaying images, and buying multiple items at once. We have also implemented the Adapter design pattern to allow the application to use data from both input files.

2 Discussion of GUI Application

Console applications are easy to implement, but when I had to change my console-based application to a [1] GUI-based application, my main concern was learning Java Swing library as I don't know much about it. It took me a few days to cover its core concepts before I began transferring the Task 3 Code of console application to the GUI application For Task 5. In Element 2, we had to implement the classes mentioned in Element 2 in Task 5 and reuse the most of methods from Task 3. While this task seemed easy, I had to handle the console output message with [2] JOptionPane to display the message on the GUI, and I did changes in the logic flow of console-based application to match the new application requirements. They may also need to include extra functionality like user input validation, error handling. I was new to GUI Application in Java Swing, designing the UI was a challenge. However, I was able to use the pick and drop UI option provided by Java to design the UI interface, including Text, Layout, JTable, Buttons and Lables. Overall, I designed the GUI application with Add, Addx, and Buy, BuyX button. I used the existing Load method to populate the data in the [2] JTable with proper column names. Overall. I designed the whole GUI Application but some features we have not implement in the task 5.


Ashers Sports Consortium

Code	Title	Description	Price	Quantity
FTB4444444	Puma ULTR...	Ultra-fast T...	70.50	11
GLF5555555	TaylorMade...	TP5 pix 2.0 ...	49.99	6
CYC1212121	MET Manta ...	Cutting edg...	132.00	3
CYC1111222	Giro Strade ...	Designed fo...	39.99	4
SWM2342341	Osprey Wo...	Full length ...	99.09	7
RUN1234567	Adidas M20 ...	Colourblock...	19.49	5
CYC1111111	Castelli Alp...	GORE-TEX ...	110.00	10
SWM2222222	GUL 7 Seas...	Come with 3...	23.00	8
GLF3333333	Titleist Golf ...	Tour Soft 12...	25.99	10
FTB4121611	Addidas Me...	Mini football...	43.00	4
RUN1234500	Garmin Fore...	Smart watch...	429.99	1
GLF5764531	Nike Golf Gi...	Therma-FIT ...	139.95	2
FTB4121612	Argentina H...	Junior footb...	50.00	0
GLF5764731	Puma Mens ...	The moistur...	45.00	7
SWM2222221	GUL Junior ...	A long sleev...	13.00	4
RM80W	(Rollerblad...	Ideal for be...	159.95	10
BAPXT	(Bladerunn...	Recreationa...	89.95	2
SSC2022	(Sunday) S...	Complete B...	1339.95	1
CSL2	(CORE) SL2	Complete St...	189.95	1
SCP	(Santa Cruz...	Gives smoot...	82.95	3
ESM	(Enuff) Skull...	Mini skateb...	35.99	3

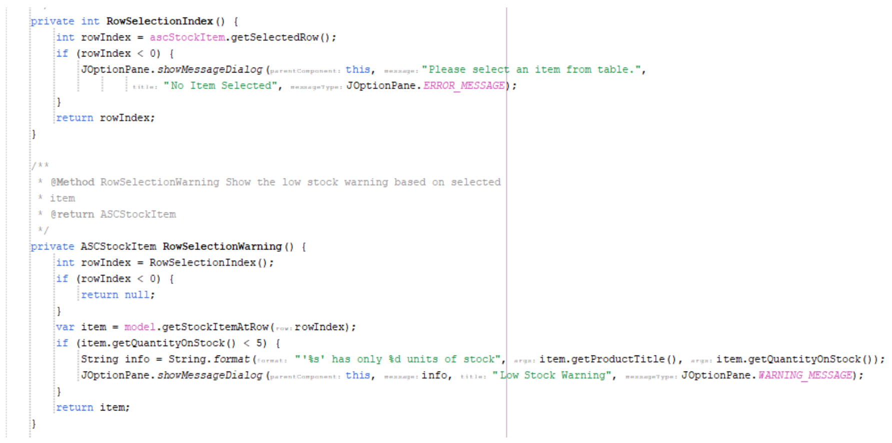
Figure 1: GUI Application

3 Discussion of Enhanced Application

In this Section, I will be discussing the enhancements made to an existing [1]GUI application. Four new features were added to the application to improve its functionality and main focus on any two enhancements.

3.1 Enhancement 1: Displaying Photographs of Stock Items

I will show the addition of photographs of stock items. This feature was added to enhance the GUI of the application and provide users with a better understanding of the products if they are purchasing. To implement the feature of [8] displaying photographs of stock items, the teacher provided the images of each item. These images were stored in the 'photos' folder within the application, and the image title is the Stock Item code. I wrote a [9] service method called 'getPhotoFileName' that returns the file name with custom modifications, such as appending the product code with a .jpg extension. To ensure that the correct image was displayed for each stock item, I implemented a selection listener for the JTable row selection. When user selects the stock item, listener invokes and calls the 'getPhotoFileName' method to retrieve the file name of the corresponding item, and load image and display it on the screen to allow the user to see Image of the stock item they are interested in purchasing.



```java
private int RowSelectionIndex() {
    int rowIndex = ascStockItem.getSelectedRow();
    if (rowIndex < 0) {
        JOptionPane.showMessageDialog(parentComponent: this, message: "Please select an item from table.",
                title: "No Item Selected", messageType: JOptionPane.ERROR_MESSAGE);
    }
    return rowIndex;
}

/**
 * @Method RowSelectionWarning Show the low stock warning based on selected
 * item
 * @return ASCStockItem
 */
private ASCStockItem RowSelectionWarning() {
    int rowIndex = RowSelectionIndex();
    if (rowIndex < 0) {
        return null;
    }
    var item = model.getStockItemAtRow(row: rowIndex);
    if (item.getQuantityOnStock() < 5) {
        String info = String.format(format: "'%s' has only %d units of stock", args: item.getProductTitle(), args: item.getQuantityOnStock());
        JOptionPane.showMessageDialog(parentComponent: this, message: info, title: "Low Stock Warning", messageType: JOptionPane.WARNING_MESSAGE);
    }
    return item;
}
```

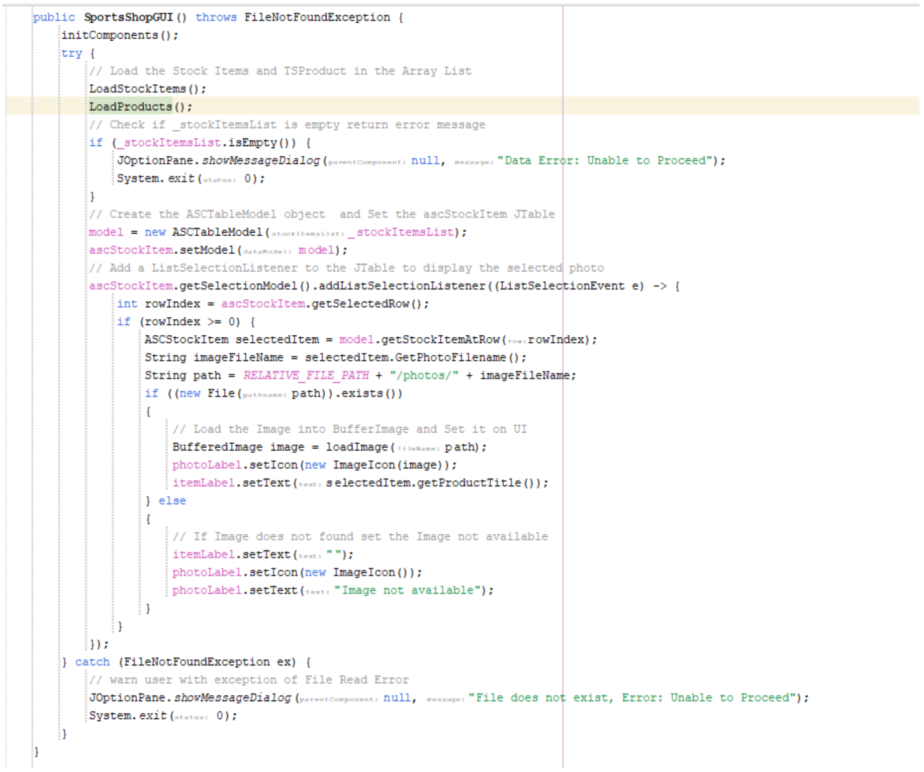
Figure 2: Warning Code

3.2 Enhancement 2: Allowing User to Purchase More Than One Unit

Users can purchase more than one unit of a stock item. It will increase sales by making it easier to buy multiple units of a product for [10] B2B Customer. hence B2C Customer can purchase only one unit of item. To implement it I added a new [6] JOption with JCombobox with Quantity popup the dialog box to for user where he can Select any quantity of the item up to maximum quantity available for specific stock Item and set the value setValueAt() method and fireTableCellUpdated() to update the value in the JTable and Array List. as well as For Add new quantity to stock item, he can select the quatity from 5 to 10 then using the setValueAt() method to set the value and fireTableCellInserted() method to update the Jtable and Array List.

4 Discussion of Merged Application

To merge data class from the Teesside Skates system with the Enhanced AsherSportsConsortium Application, The Teesside Skates system has a data class named TSProduct and a sample data file named ts_products.txt. task requires that no changes be made to the contents of the input files or original data class files, and that no changes are required to the Teesside Skates System. To allow the ASCSystem to use data from both input files, Next, we use the [3] Adapter Design Pattern to make the TS products behave as if they were ASC stock items. This involves creating a new class that adapts the TSProduct class to the ASCStockItem class, so I need to implement the new class with name [4] ProductToStockItemAdapter class that extends the ASCStockItem class as well as inside the adapter class constructor we initialize the TSProduct class. We begin by copying the ts_products.txt file to the same location as Asher-



```java
public SportsShopGUI() throws FileNotFoundException {
    initComponents();
    try {
        // Load the Stock Items and TSProduct in the Array List
        LoadStockItems();
        LoadProducts();
        // Check if _stockItemsList is empty return error message
        if (_stockItemsList.isEmpty()) {
            JOptionPane.showMessageDialog( parentComponent: null,  message: "Data Error: Unable to Proceed");
            System.exit( status: 0);
        }
        // Create the ASCTableModel object  and Set the ascStockItem JTable
        model = new ASCTableModel( stockItemsList: _stockItemsList);
        ascStockItem.setModel( dataModel: model);
        // Add a ListSelectionListener to the JTable to display the selected photo
        ascStockItem.getSelectionModel().addListSelectionListener((ListSelectionEvent e) -> {
            int rowIndex = ascStockItem.getSelectedRow();
            if (rowIndex >= 0) {
                ASCStockItem selectedItem = model.getStockItemAtRow( row: rowIndex);
                String imageFileName = selectedItem.GetPhotoFilename();
                String path = RELATIVE_FILE_PATH + "/photos/" + imageFileName;
                if ((new File( pathname: path)).exists())
                {
                    // Load the Image into BufferImage and Set it on UI
                    BufferedImage image = loadImage( fileName: path);
                    photoLabel.setIcon(new ImageIcon(image));
                    itemLabel.setText( text: selectedItem.getProductTitle());
                } else
                {
                    // If Image does not found set the Image not available
                    itemLabel.setText( text: " ");
                    photoLabel.setIcon(new ImageIcon());
                    photoLabel.setText( text: "Image not available");
                }
            }
        });
    } catch (FileNotFoundException ex) {
        // warn user with exception of File Read Error
        JOptionPane.showMessageDialog( parentComponent: null,  message: "File does not exist, Error: Unable to Proceed");
        System.exit( status: 0);
    }
}
```

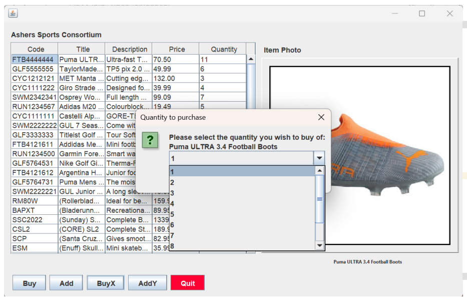
Figure 3: Item selection code

SportsConsortium3.csv within the ASCSystem NetBeans Project folder. We also extract the contents of the ts_pics.zip file to the photo's subfolder. Finally, we modified the SportsShopGUI.java class to add new method load data from TSProduct input files into a single Array List. Creates instances of the adapter class for each TS product. We then add these instances to the existing Array List of ASC stock items. We need to make sure we do not touch any other file functionalities, only the necessary changes in SportsShopGUI.java.

Similarly, if the [8] photographs of TS products are not available to displayed, I put message that "Image not available" Overall, the success of the merged application would depend on the effectiveness of the [3] Adapter Design Pattern implementation and the thoroughness of the testing and debugging process.

5 Discussion of Improvement

The current application could be improved by [7] implementing a login feature for users. This improvement would provide an added layer of security to the application, ensuring that only authorized users can access sensitive data. A login



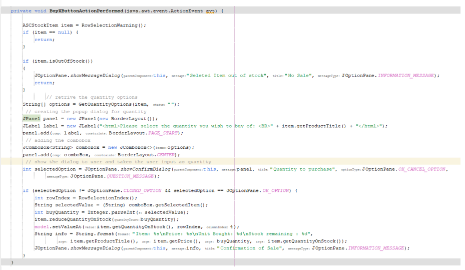
Ashers Sports Consortium

Code	Title	Description	Price	Quantity
FTB4444444	Puma ULTR...	Ultra-fast T...	70.50	11
GLF5555555	TaylorMade...	TP5 pix 2.0 ...	49.99	6
CYC1212121	MET Manta ...	Cutting edg...	132.00	3
CYC1111222	Giro Strade ...	Designed fo...	39.99	4
SWM2342341	Osprey Wo...	Full length ...	99.09	7
RUN1234567	Adidas M20 ...	Colourblock...	19.49	5
CYC1111111	Castelli Alp...	GORE-T		
SWM2222222	GUL 7 Seas...	Come wit		
GLF3333333	Titleist Golf ...	Tour Soft		
FTB4121611	Addidas Me...	Mini footb		
RUN1234500	Garmin Fore...	Smart wa		
GLF5764531	Nike Golf Gi...	Therma-F		
FTB4121612	Argentina H...	Junior foo		
GLF5764731	Puma Mens ...	The mois		
SWM2222221	GUL Junior ...	A long sle		
RM80W	(Rollerblad...	Ideal for be...	159.9	
BAPXT	(Bladerunn...	Recreationa...	89.95	
SSC2022	(Sunday) S...	Complete B...	1339	
CSL2	(CORE) SL2	Complete St...	189.9	
SCP	(Santa Cruz...	Gives smoot...	82.95	
ESM	(Enuff) Skull...	Mini skateb...	35.99	

Quantity to purchase

Please select the quantity you wish to buy of:
Puma ULTRA 3.4 Football Boots

Figure 4: AddX items Code



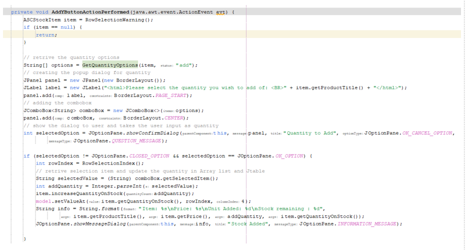
```
private void BuyXButtonActionPerformed(java.awt.event.ActionEvent evt) {

    ASCStockItem item = RowSelectionWarning();
    if (item == null) {
        return;
    }

    if (item.isOutOfStock())
    {
        JOptionPane.showMessageDialog(parentComponent: this, message: "Seleted Item out of stock", title: "No Sale", messageType: JOptionPane.INFORMATION_MESSAGE);
        return;
    }
            // retrive the quantity options
    String[] options = GetQuantityOptions(item, status: "");
     // creating the popup dialog for quantity
    JPanel panel = new JPanel(new BorderLayout());
    JLabel label = new JLabel("<html>Please select the quantity you wish to buy of: <BR>" + item.getProductTitle() + "</html>");
    panel.add(comp: label, constraints: BorderLayout.PAGE_START);
     // adding the combobox
    JComboBox<String> comboBox = new JComboBox<>(items: options);
    panel.add(comp: comboBox, constraints: BorderLayout.CENTER);
     // show the dialog to user and takes the user input as quantity
    int selectedOption = JOptionPane.showConfirmDialog(parentComponent: this, message: panel, title: "Quantity to purchase", optionType: JOptionPane.OK_CANCEL_OPTION,
            messageType: JOptionPane.QUESTION_MESSAGE);

    if (selectedOption != JOptionPane.CLOSED_OPTION && selectedOption == JOptionPane.OK_OPTION) {
        int rowIndex = RowSelectionIndex();
        String selectedValue = (String) comboBox.getSelectedItem();
        int buyQuantity = Integer.parseInt(s: selectedValue);
        item.reduceQuantityOnStock(quantityCount: buyQuantity);
        model.setValueAt(value: item.getQuantityOnStock(), rowIndex, columnIndex: 4);
        String info = String.format(format: "Item: %s\nPrice: %s\nUnit Bought: %d\nStock remaining : %d",
                args: item.getProductTitle(), args: item.getPrice(), args: buyQuantity, args: item.getQuantityOnStock());
        JOptionPane.showMessageDialog(parentComponent: this, message: info, title: "Confirmation of Sale", messageType: JOptionPane.INFORMATION_MESSAGE);
    }
}
```

Figure 5: BuyX items Code



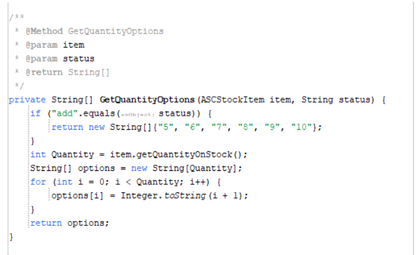
```
private void AddYButtonActionPerformed(java.awt.event.ActionEvent evt) {
    ASCStockItem item = RowSelectionWarning();
    if (item == null) {
        return;
    }

    // retrive the quantity options
    String[] options = GetQuantityOptions(item, status: "add");
    // creating the popup dialog for quantity
    JPanel panel = new JPanel(new BorderLayout());
    JLabel label = new JLabel("<html>Please select the quantity you wish to add of: <BR>" + item.getProductTitle() + "</html>");
    panel.add(comp: label, constraints: BorderLayout.PAGE_START);
    // adding the combobox
    JComboBox<String> comboBox = new JComboBox<>(items: options);
    panel.add(comp: comboBox, constraints: BorderLayout.CENTER);
    // show the dialog to user and takes the user input as quantity
    int selectedOption = JOptionPane.showConfirmDialog(parentComponent: this, message: panel, title: "Quantity to Add", optionType: JOptionPane.OK_CANCEL_OPTION,
            messageType: JOptionPane.QUESTION_MESSAGE);

    if (selectedOption != JOptionPane.CLOSED_OPTION && selectedOption == JOptionPane.OK_OPTION) {
        int rowIndex = RowSelectionIndex();
        // retrive selection item and update the quantity in Array list and Jtable
        String selectedValue = (String) comboBox.getSelectedItem();
        int addQuantity = Integer.parseInt(s: selectedValue);
        item.increaseQuantityOnStock(quantityCount: addQuantity);
        model.setValueAt(value: item.getQuantityOnStock(), rowIndex, columnIndex: 4);
        String info = String.format(format: "Item: %s\nPrice: %s\nUnit Added: %d\nStock remaining : %d",
                args: item.getProductTitle(), args: item.getPrice(), args: addQuantity, args: item.getQuantityOnStock());
        JOptionPane.showMessageDialog(parentComponent: this, message: info, title: "Stock Added", messageType: JOptionPane.INFORMATION_MESSAGE);

    }
```

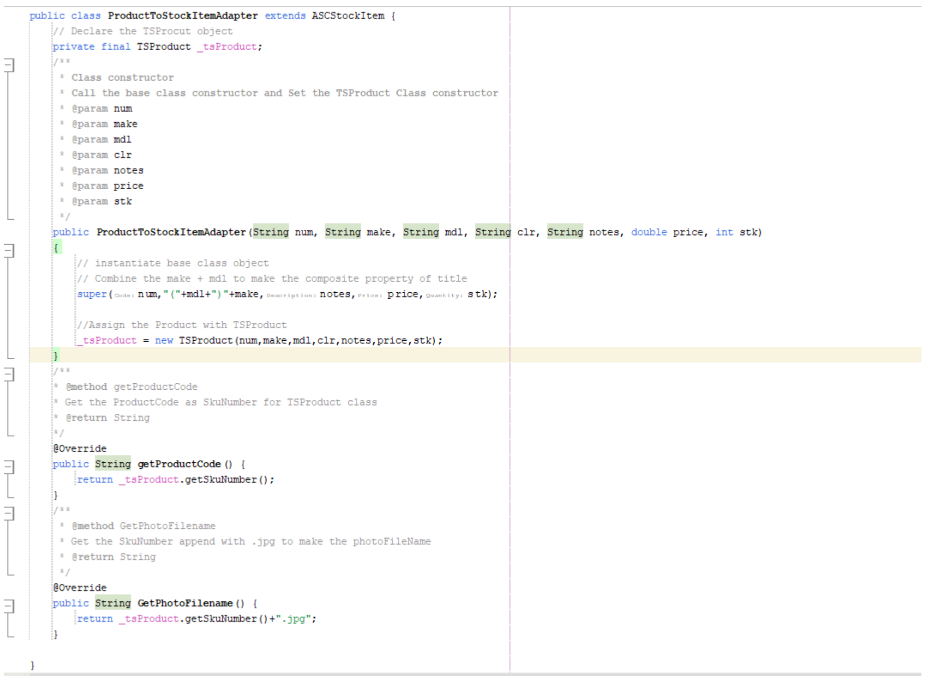
Figure 6: AddX items Code



```
/**
 * @Method GetQuantityOptions
 * @param item
 * @param status
 * @return String[]
 */
private String[] GetQuantityOptions(ASCStockItem item, String status) {
    if ("add".equals(anObject: status)) {
        return new String[]{"5", "6", "7", "8", "9", "10"};
    }
    int Quantity = item.getQuantityOnStock();
    String[] options = new String[Quantity];
    for (int i = 0; i < Quantity; i++) {
        options[i] = Integer.toString(i + 1);
    }
    return options;
}
```

Figure 7: Options code

screen would be added to the application, requiring users to enter a username and password to gain access to its features. Once logged in, the user would be granted access to the features of the application. Additionally, a [7] database would need to be created to store user login credentials securely. To implement this feature, several changes would be required in the application. Firstly, the user interface would need to be modified to include a login screen. This screen would prompt the user to enter their login credentials. Secondly, a database would need to be created to store user login credentials. User login information would be encrypted and stored securely. The application would then verify the user's login credentials with the database to determine if they have access to the features. The key code required to implement this feature would be a new method for verifying user login credentials and the user would be granted access to the application's features, and if not, they would receive an error message. To test the improved application, a series of test cases would need to be developed to ensure that the login feature works correctly. These test cases could include entering incorrect login credentials, entering correct login credentials, and attempting to access the application without logging in. Furthermore, The goal is to provide a seamless user experience by ensuring that all images load quickly. The Buy in Bulk Quantity feature could also be improved by allowing users to enter a specific quantity if available, and if not, the system should return an error message with a maximum limit of quantity. In the add in Bulk Quantity feature, we should allow users to add less than 5 quantities too, rather



```java
public class ProductToStockItemAdapter extends ASCStockItem {
    // Declare the TSProcut object
    private final TSProduct _tsProduct;
    /**
     * Class constructor
     * Call the base class constructor and Set the TSProduct Class constructor
     * @param num
     * @param make
     * @param mdl
     * @param clr
     * @param notes
     * @param price
     * @param stk
     */
    public ProductToStockItemAdapter(String num, String make, String mdl, String clr, String notes, double price, int stk)
    {
        // instantiate base class object
        // Combine the make + mdl to make the composite property of title
        super( Code: num,"("+mdl+")"+make, Description: notes, Price: price, Quantity: stk);

        //Assign the Product with TSProduct
        _tsProduct = new TSProduct(num,make,mdl,clr,notes,price,stk);
    }
    /**
    * @method getProductCode
    * Get the ProductCode as SkuNumber for TSProduct class
    * @return String
    */
    @Override
    public String getProductCode() {
        return _tsProduct.getSkuNumber();
    }
    /**
     * @method GetPhotoFilename
     * Get the SkuNumber append with .jpg to make the photoFileName
     * @return String
     */
    @Override
    public String GetPhotoFilename() {
        return _tsProduct.getSkuNumber()+".jpg";
    }

}
```

Figure 8: Adapter class code

than only greater than or equal to 5. Another improvement could be the integration of payment gateways such as [11] PayPal or Braintree, to allow users to make secure payments for their purchases. In conclusion, the implementation of a login feature, optimization of image loading speed, improvement of Bulk Quantity features, and payment gateway integration would improve the [12] application's robustness and user experience. These improvements would result in increased sales and improved customer satisfaction, providing a secure and seamless shopping experience for users.

6 References

1. Java Swing - `https://www.javatpoint.com/java-swing`
2. Introduction to Java Swing - `https://www.geeksforgeeks.org/introduction-to-java-swing`
3. Adapter Design Pattern - `https://www.tutorialspoint.com/design_pattern/adapter_pattern.htm`
4. Adapter Design Pattern in Java - `https://www.javadevjournal.com/java-design-patterns/adapter-design-pattern`
5. Java Swing JTable - `https://www.geeksforgeeks.org/java-swing-jtable/`
6. Java Swing JComboBox Examples - `https://www.geeksforgeeks.org/java-swing-jcombobox-examples/`
7. Login with database - `https://www.javaguides.net/2019/07/login-application-using-java-swin`
   `html/`
8. Image Hanlding in GUI - `https://netbeans.apache.org/kb/docs/java/gui-image-display.html`
9. Service Methods in Java - `https://docs.oracle.com/javaee/5/tutorial/doc/bnafv.html`
10. B2B and B2C - `https://dzone.com/articles/b2b-vs-b2c-mobile-apps-how-to-know-the-differe`
11. Payment Integration - `https://www.codejava.net/coding/how-to-integrate-paypal-payment-int`
12. Application Robustness - `https://stackoverflow.com/questions/283141/best-practices-for-robustness`