LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS5003NI Data Structure and Specialist Programming

30% Individual Coursework

2023-24 Autumn

Student Name: Rajita maharjan

London Met ID: 22067335

College ID: np01ai4a220052

Assignment Due Date: Friday, January 12, 2024
Assignment Submission Date: Friday, January 12, 2024
Word Count: 2773

I confirm that I understand my coursework needs to be submitted online My Second teacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Table of Contents

Table of Figures

Table of tables

1. Introduction

1.1. About This Coursework.

This given coursework is to create a user-friendly User Interface (UI) focused on skincare information. The main objective of this coursework is to improve the project by creating a user-friendly interface that allows simple access to a complete database of skincare information. With this, the coursework aims to enhance the user interaction and overall experience with skincare products.

1.2. Tools Used in Coursework

1.2.1. Apache NetBeans

Apache NetBeans is a top-level Apache Project dedicated to providing extremely reliable software development products to developers, users, and businesses who rely on NetBeans as the foundation for their products specifically, to enable them to develop these products quickly, efficiently, and easily by using the strengths of the Java platform and other relevant standards in the industry. The Apache NetBeans Platform provides a reliable and flexible application architecture. (Apache NetBeans 20, 2017-2023)

Some advantages of NetBeans:

- The platform allows you to add, delete and change menus, toolbars or keyboard shortcuts through an XML file called a layer file distributed with your application.
- The platform handles managing "windows," which are simply top-level panels within the larger NetBeans frame. Users can reposition, resize, open or close these windows, and the platform retains the state and position of each the next time the application is launched. (Tom Wheeler, 2023)

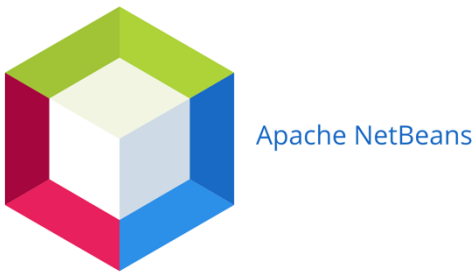

Figure 1: Apache NetBeans

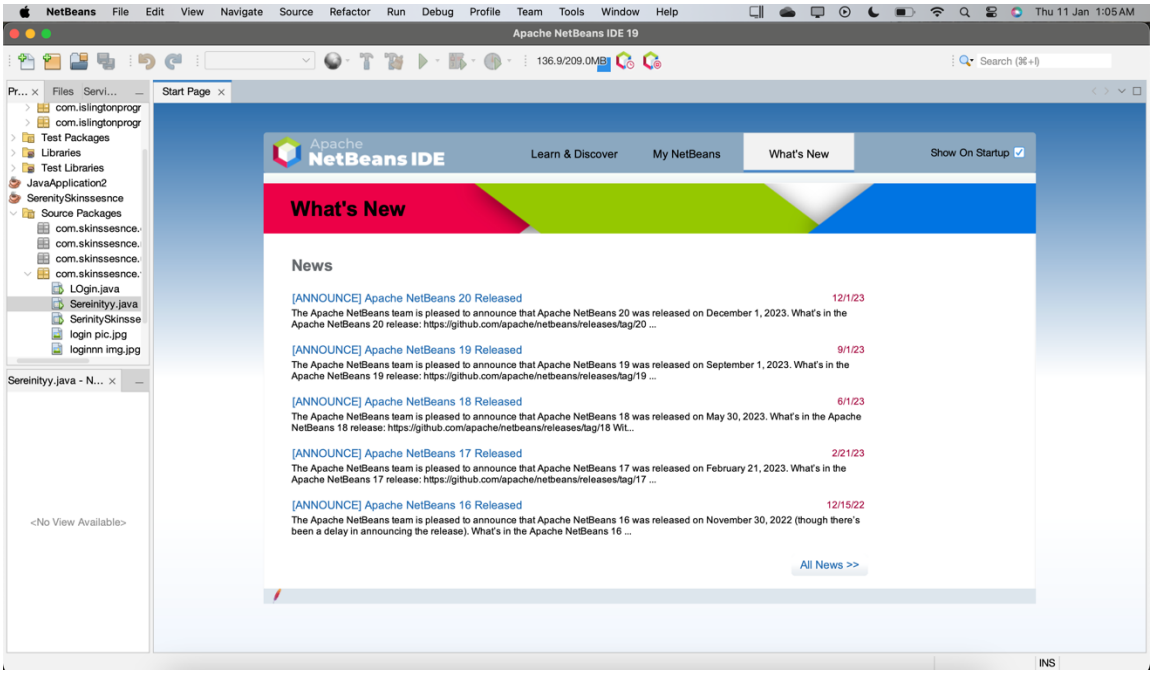

Figure 2

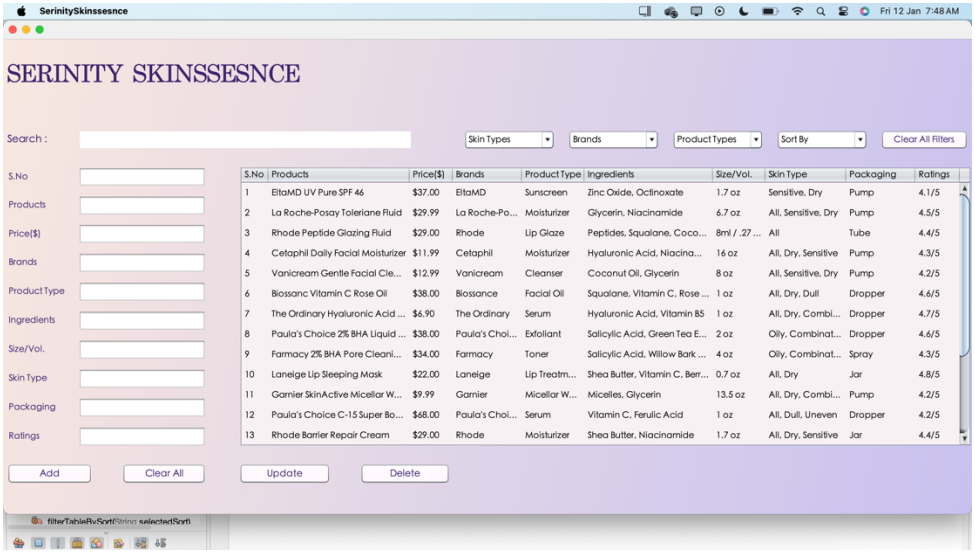

Figure 3

1.2.2. Java

Java is a popular object-oriented programming language and software platform used by billions of devices that includes laptop computers, mobile devices, gaming consoles medical equipment, and many more. Java's principles and grammar are based on the C and C++ programming languages. (Introduction to Java , n.d.)

The main objective of developing the Java programming language is to make it portable, simple, and secure. Apart from that, there are multiple helpful features which helps significantly to the popularity of this language. (Jaiswal, n.d.)

- Java was easy to learn because of its familiar, C-like syntax that was popular at the time of the language's release.
- One of the advantages of Java over other languages is how quickly it adopts new features and responds to the needs of the community.
- Java is completely object-oriented, implementing major OOA&D concepts such as : Inheritance, composition, polymorphism, encapsulation and interfaces.
- Java supports functional programming. (Advantages of Java , 2000 - 2024)

Figure 4 : Java.

2. Algorithms Utilized

2.1. Binary Search Algorithm

Purpose of Binary search algorithm:

The Binary Search algorithm's purpose is to efficiently search through a sorted list of products. Instead of a linear search, which requires reviewing each item repeatedly, binary search uses the list's sorted form to immediately reduce the search space.

The Binary Search algorithm used in this program searches for a specific product inside a pre-sorted list of products, with an in time complexity of O(n), which allows for increased responsiveness, especially for graphical user interfaces. The binarySearch function, which is called when the 'SearchTextFieldKeyReleased' method receives user input, continuously decreases the search space by comparing the search word to the middle element. This process continues until a match is found or the search space has been filled, at that point it outputs true or false. Binary Search was selected for being able to quickly locate things within sorted records, which is suitable for the graphical user interface's requirement for responsive search operations in the 'JTable'.

Pseudocode :

```
PROCEDURE searchTextFieldKeyReleased(evt: KeyEvent):
    // Get the search text from the search JTextField
    searchText := toLowerCase(searchJTextField.getText())
    // Perform binary search on the sorted list
    found := binarySearch(searchText)
    // Update GUI based on the search result
    IF found THEN
        // Item found, update GUI accordingly
        OUTPUT "Item found: " + searchText
    ELSE
        // Item not found, handle as needed'
        OUTPUT "Item not found: " + searchText
    END IF
```

```
END PROCEDURE

FUNCTION binarySearch(searchTerm: String) RETURNS BOOLEAN:

    // Geting the table model from the products JTable

    tblModel := getTableModelFromJTable(productsJTable)

    // Getting the row count

    rowCount := getRowCount(tblModel)

    // Initializingg low and high for binary search
    low := 0

    high := rowCount - 1

    // Binary search loop

    WHILE low <= high DO

        // Calculate mid index

        mid := low + (high - low) / 2

        // Get the column value from the mid index

        columnValue := toLowerCase(tblModel.getValueAt(mid, 1).toString())

        // Compare column value with search term

        result := compareStrings(columnValue, searchTerm)

        IF result = 0 THEN

            // Match found

            RETURN true

        ELSE IF result < 0 THEN

            // Adjust low for the right half

            low := mid + 1

        ELSE

            // Adjust high for the left half

            high := mid – 1
```

```
        END IF
    END WHILE
    // Match not found
    RETURN false
END FUNCTION
```

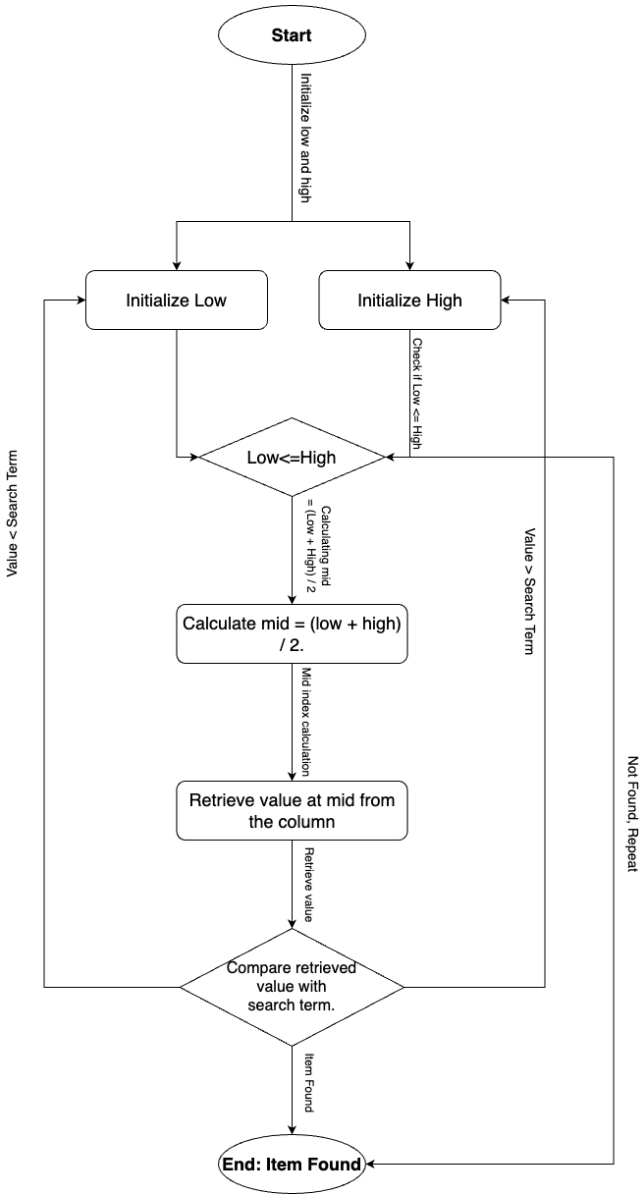

Figure 5: FlowChart of Binary Search algorithm.

3. Class Diagram

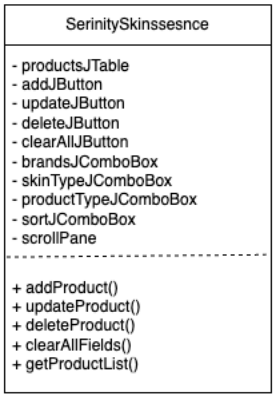

Figure 6: class diagram

4. Method Description

Method1:

searchJTextFieldKeyReleased(java.awt.event.KeyEvent evt):
This method is triggered when keys are released in the search text field.
It updates the search term and performs a binary search on the sorted product list. It then updates the GUI based on the search result.

Method2:

binarySearch(String searchTerm):
Implements the binary search algorithm to find a product based on the search term.
It searches for the search term in the sorted product list and returns true if found, false otherwise.

Method3:

searchJTextFieldActionPerformed(java.awt.event.ActionEvent evt):
Handles actions when the search text field is used.

Method4:

productsJTableMouseClicked(java.awt.event.MouseEvent evt):
Handles mouse click events on the products table.
It retrieves data from the selected row and sets it to various text fields for editing.

Method5:

clearAllJButtonActionPerformed(java.awt.event.ActionEvent evt):
Clears all text fields when the "Clear All" button is clicked.

Method6:

clearAllfiltersJbuttonActionPerformed(java.awt.event.ActionEvent evt):
Clears all filters and resets the table to show all products.
It resets the row filter, refreshes the table, and resets combo boxes for filtering.

Method7:

sortJComboBoxActionPerformed(java.awt.event.ActionEvent evt):
Handles sorting based on the selected item in the sort combo box.
Calls filterTableBySort to sort the table based on the selected criteria.

Method8:

skinTypeJComboBoxActionPerformed(java.awt.event.ActionEvent evt):
Filters the table based on the selected skin type.
Calls filterTableBySkinType to filter the table based on the selected skin type.

Method9:

brandsJComboBoxActionPerformed(java.awt.event.ActionEvent evt):
Filters the table based on the selected brand.
Calls filterTableByBrand to filter the table based on the selected brand.

Method10:

productTypeJComboBoxActionPerformed(java.awt.event.ActionEvent evt):
Filters the table based on the selected product type.
Calls filterTableByProductType to filter the table based on the selected product type.

Method11:

filterTableBySkinType(String selectedSkinType):
Filters the table based on the selected skin type.

Method12:

filterTableBySort(String selectedSort):
Filters the table based on the selected sorting criteria.

Method13:

filterTableByBrand(String selectedBrand):

Filters the table based on the selected brand.

Method14:

filterTableByProductType(String selectedProductType):
Filters the table based on the selected product type.

Method15:

filterTable(int columnIndex, String selectedValue):
Applies a filter to the table based on the selected column index and value.

Method16:

main(String args[]):
The main method that launches the application.
Initializes the UI and sets the look and feel.

5. Test Cases

Test 1: Search for a Product.

Objective	To test search functionality for finding a product.
Action	Entering the name of the product in text field.
Expected Result	The table should display the matching products in the table.
Actual Result	The table shows the products that matches the search term.
Conclusion	The search functionality is successful.

Table 1: Test 1: Search for a Product.

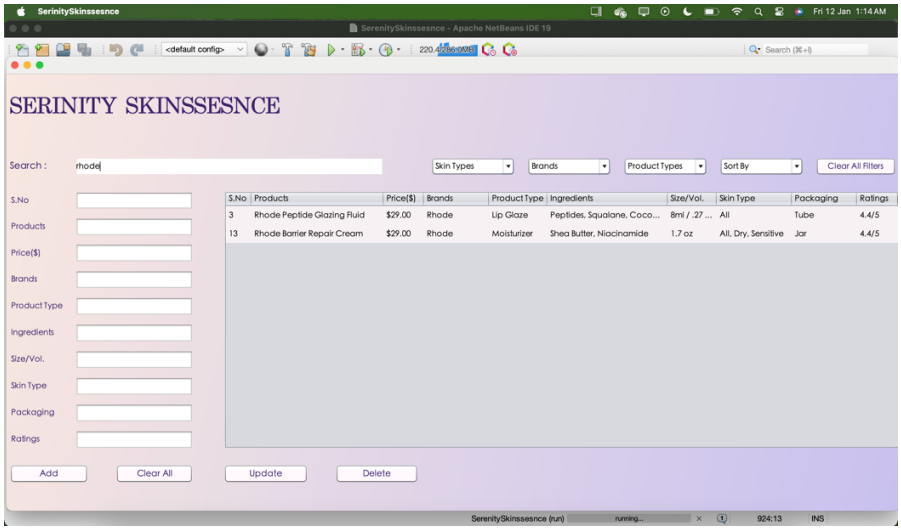

Figure 7: Search for a Product

Test 2: Add New Product to the Jtable.

Objective	To test the functionality of adding the new products.
Action	1. Enter the valid details in the respective text fields. 2. Click on add button to add the product.
Expected Result	The product with the provided details should be added to the table successfully.
Actual Result	The product is successfully added to the table.
Conclusion	The test is sussessful.

Table 2: Test 2: Add New Product to the Jtable.

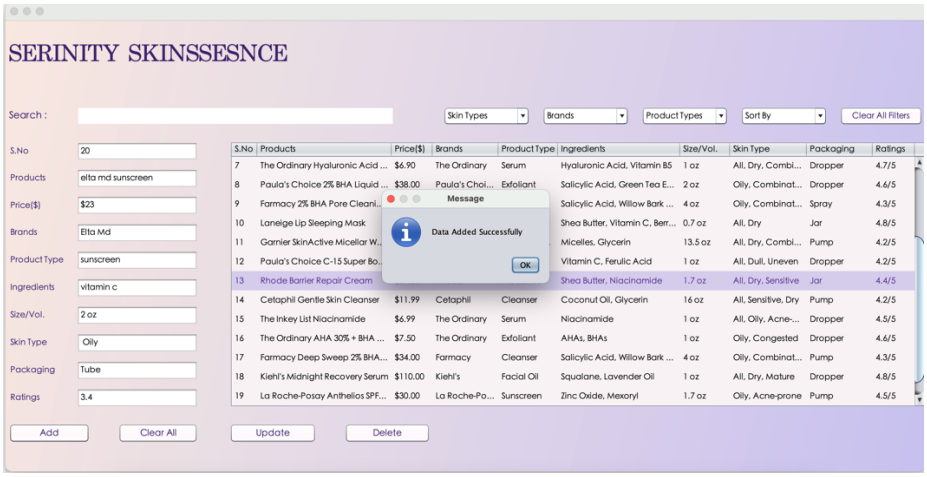

Figure 8: Add New Product to the Jtable.

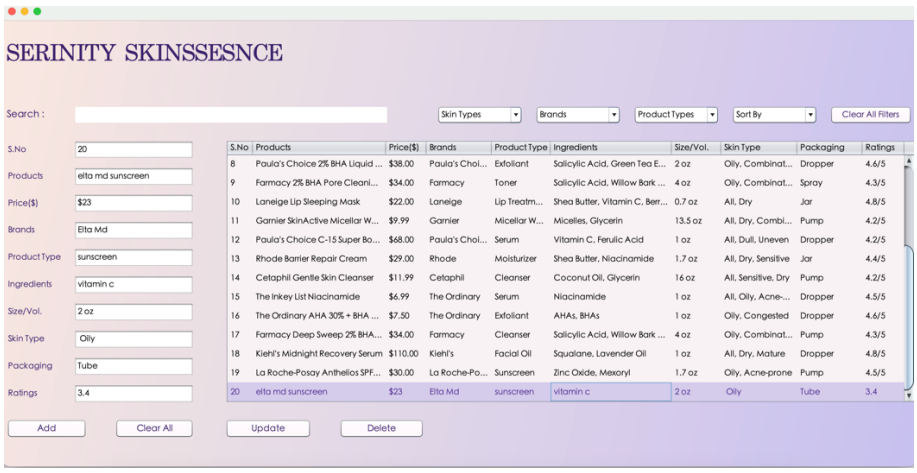

Figure 9: Add New Product to the Jtable{2}.

Test 3: Delete Products from Jtable.

Objective	To test the functionality of deleting a product.
Action	1. Select a product from the table. 2. Click on the "Delete" button.
Expected Result	The selected product should be removed from the table
Actual Result	The selected product is successfully deleted without any errors.
Conclusion	The test is successful.

Table 3: Test 3: Delete Products from Jtable.

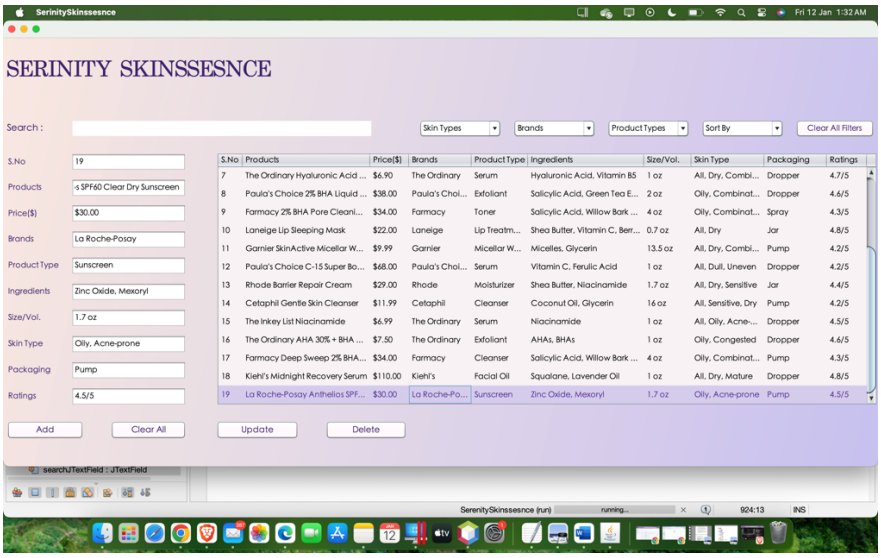

Figure 10: Delete Products from Jtable.

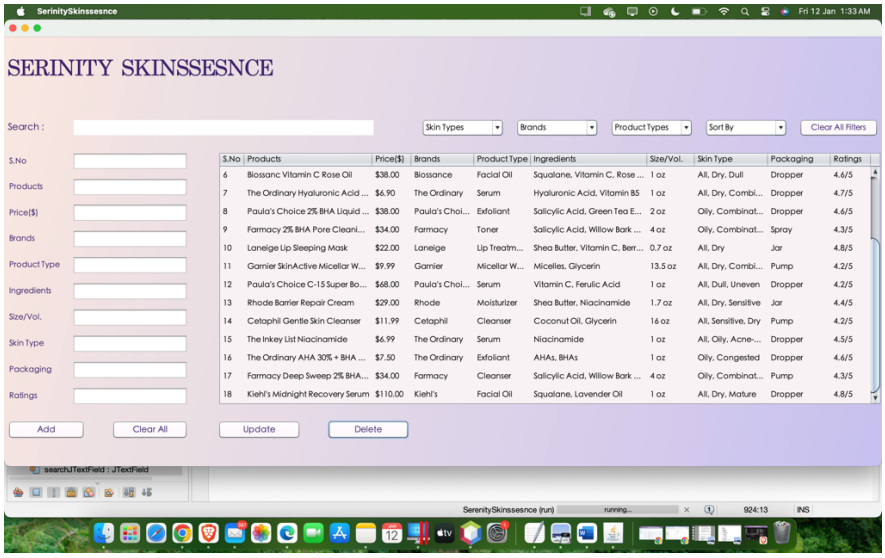

Figure 11: Delete Products from Jtable{2}.

Test 4: Update Product Details.

Objective	To test the functionality of updating product details.
Action	1. Select a product from the table. 2. Modify some details (e.g., change the product name or price). 3. Click on the "Update" button.
Expected Result	The selected product's details should be updated with the new information.
Actual Result	The selected product's details are successfully updated.
Conclusion	The test is successful.

Table 4: Test 4: Update Product Details.

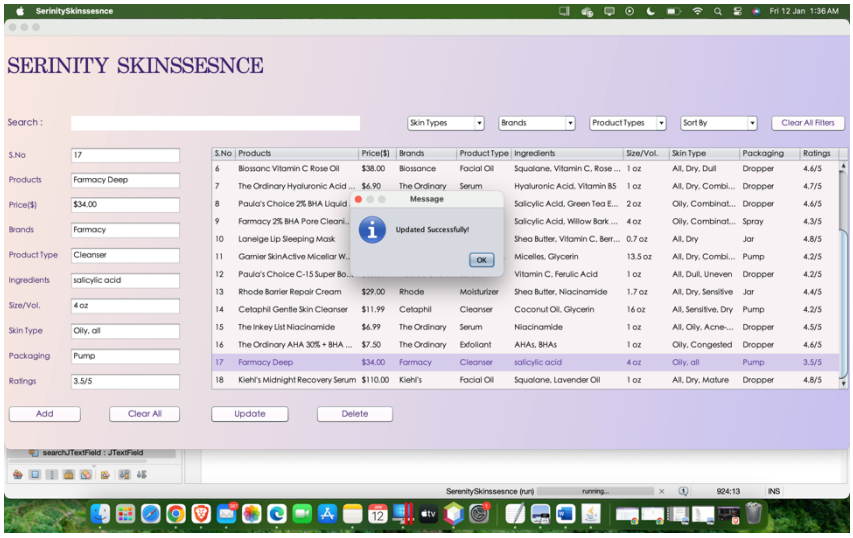

Figure 12: Update Product Details.

Test 5: Search for Non-existent Product.

Objective	To test the binary search functionality when searching for a non-existent product.
Action	1. Enter a random name or a name that doesn't exist in the product list in the search field. 2. Press Enter or wait for automatic search.
Expected Result	The search should return no results, indicating that the product is not found.
Actual Result	The search shows no results in the table.
Conclusion	The test is successful.

Table 5: Test 5: Search for Non-existent Product

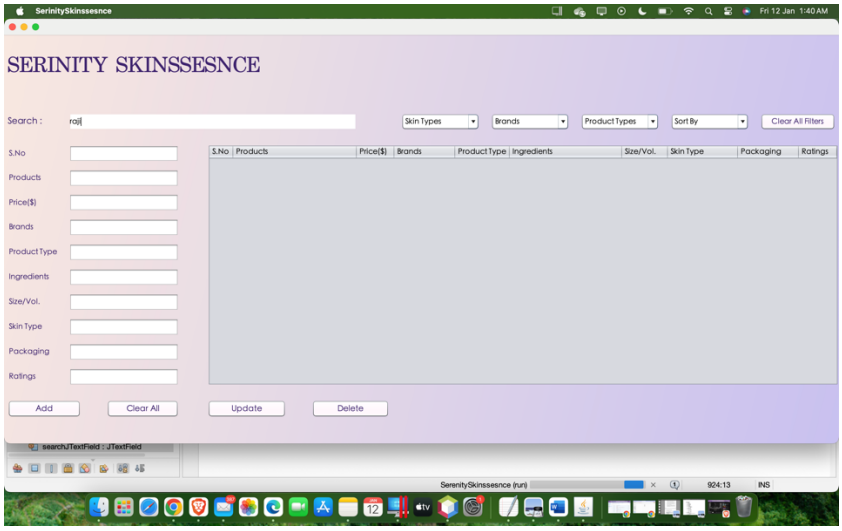

Figure 13: Search for Non-existent Product.

Test 6: Clearing All Input Fields.

Objective	To test the functionality of clearing all input fields.
Action	1. Enter random or valid details in all input fields. 2. Click on the "Clear All" button.
Expected Result	All input fields should be cleared, and no information should be present in any field.
Actual Result	All input fields are successfully cleared.
Conclusion	The test is successful.

Table 6: Test 6: Clearing All Input Fields

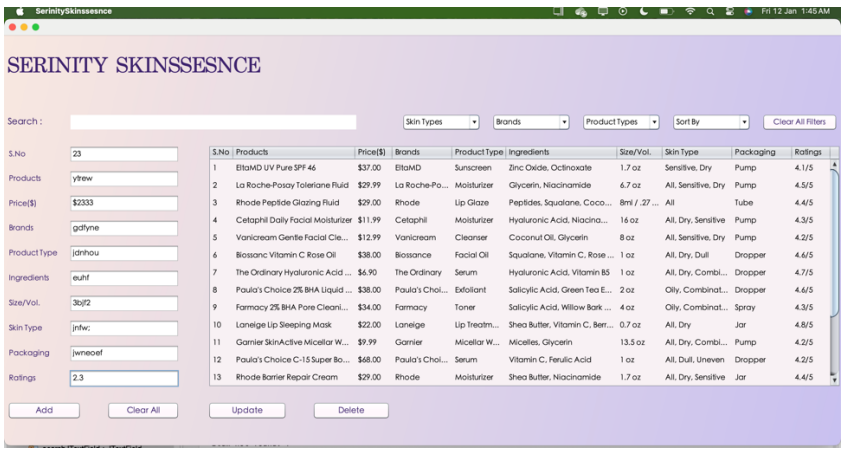

Figure 14: Clearing All Input Fields

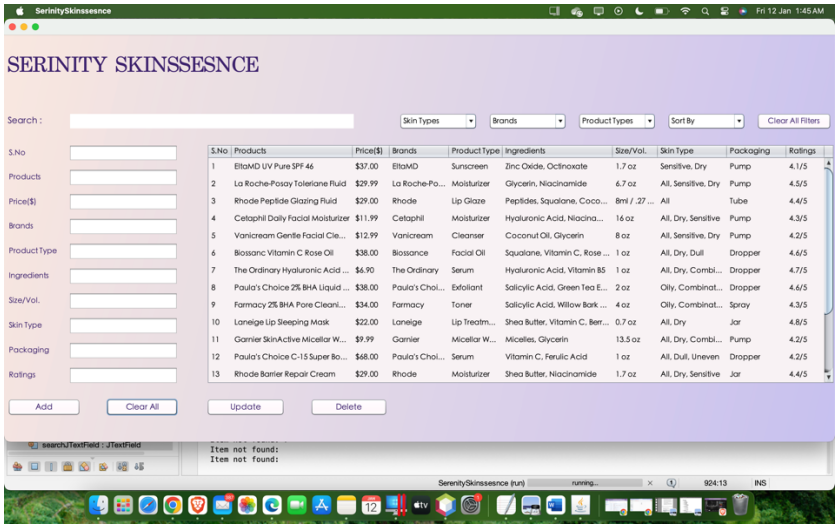

Figure 15: Clearing All Input Fields{2}.

Test 7: Filter by Skin Type, Brands, Product Type.

Objective	To test that users can filter products by skin type, brands or product type.
Action	1. Select a specific option from the skin type, brands or product type dropdown. 2. Check if the table displays only products suitable the selected option.
Expected Result	The table should show only products designed for the selected option.
Actual Result	The table displays products according to the selected skin type.
Conclusion	The test is successful.

Table 7: Test 7: Filter by Skin Type, Brands, Product Type

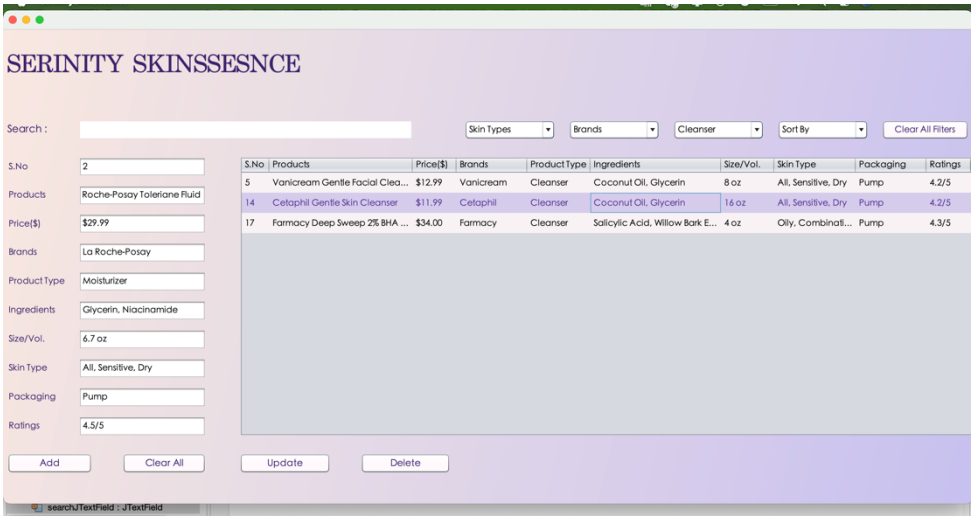

Figure 16: Filter by Skin Type, Brands, Product Type.

6. Development Process

- **Tools and Techniques Implemented**

  Programming Language and frameworks:
  Java is used as the main giving a stable and platform-independent foundation.. The Swing framework was used for Graphical User Interface (GUI) development, as it provides an entire set of components to build interactive and visually appealing interfaces.

  Development environment and Tools:
  The tool used for the coursework, managing projects and coding was NetBeans, an Integrated Development Environment (IDE) with quite a lot of features. It was an effective platform because of its user-friendly interface and strong support for Java development.

  Data Structures and Algorithms:
  For effective data organization and manipulation, key data structures like arrays, lists, and mapping were used. To improve search functionality, the binary search algorithm which is known for its efficiency in sorted datasets was also implemented in the coursework.

- **Problems Faced**
  During the project, I faced some problems and worked hard to solve them.
  Some of the problems were:
  Logical error:
  Incorrect filtering/sorting:
  Logical error are the bugs in the code that results to incorrect behaviour.

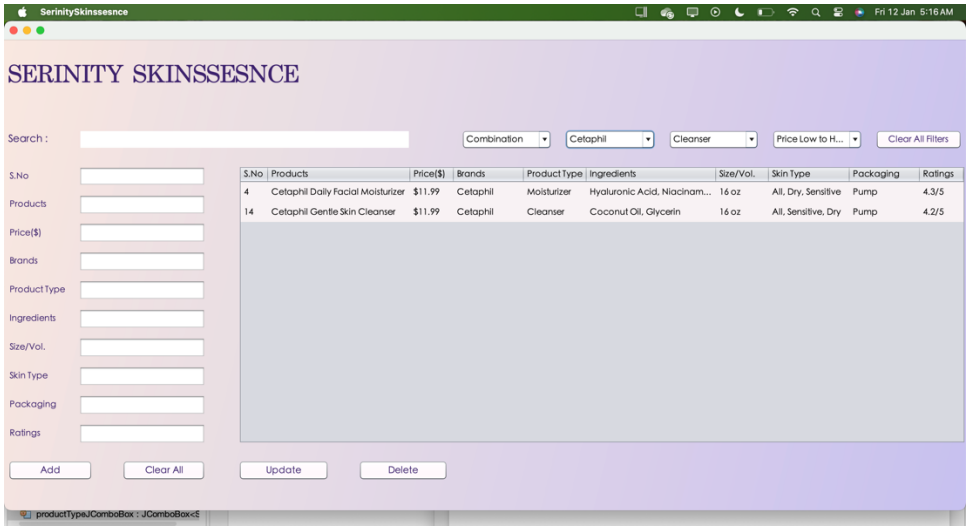

Figure 17: Logical error(Incorrect filtering)

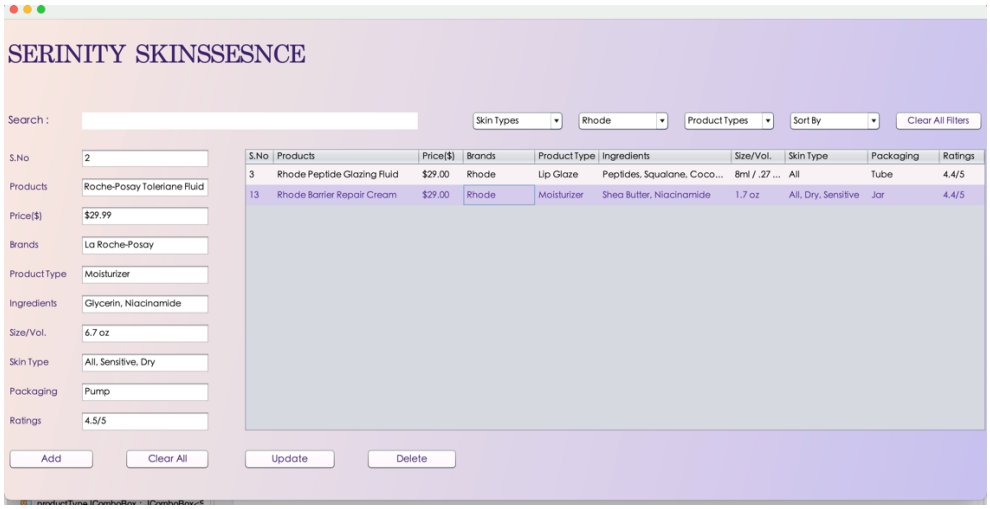

Figure 18: Logical Error(2)

Performance Issues:

Performance issues arises due to inefficient algorithms or poor code optimization. It caused slow response times.

Code Readability and organization:
Improper code organization made it harder to understand the code which resulted in too many confusions.

Insufficient Comments:
It made me more difficult for me to understand the function and goal of various sections of the code when there are insufficient comments inside the code.

```
        }
    }

    private void addJButtonActionPerformed(java.awt.event.ActionEvent evt) {

        if(numJTextField.getText().equals("")||productJTextField.getText().equals("")||priceJTextField.getText().equalsIgnoreCase(
            JOptionPane.showMessageDialog(this, "Please enter all fields!");

        }else {

            String data [] = {numJTextField.getText(),productJTextField.getText(),priceJTextField.getText(),brandsJTextField.getTe

            DefaultTableModel tblModel = (DefaultTableModel)productsJTable.getModel();
            tblModel.addRow(data);


            JOptionPane.showMessageDialog(this, "Data Added Successfully");
            numJTextField.setText("");
            productJTextField.setText("");
            priceJTextField.setText("");
            brandsJTextField.setText("");
            productTypeJTextField.setText("");
            ingredientsJTextField.setText("");
            sizeJTextField.setText("");
            skintypeJTextField.setText("");
            packagingJTextField.setText("");
            ratingsJTextField.setText("");

            Font font = new Font("Heiti TC", Font.BOLD, 13);
            addJButton.setFont(font);
        }
    }
```

Figure 19: problems Faced(Lack Of comments)

- **Challenges**

  During the coursework, I faced a few challenges that helped me grow as a programmer. Although it was challenging sometimes, learning new tools and technologies particularly for UI design and database integration taught me a lot. There were alot of issues.

  Online research and necessary files provided by module leader helped me a lot. It helped me solve specific technical issues. Time managing was also a continuous challenge so I learned to organize my time and work better.

  I learned from these challenges that how important it is to maintain your strength and redefine as project requirements change. Constant testing and correction of errors showed to me the importance of paying close attention to detail. This experience showed me that effective development requires not only skills but also self research.

7. Conclusion

I successfully completed the coursework, I developed the “Serinity Skin Essence” system which meets the requirement of a skincare product management system.

This Java-based GUI application includes major functions such as adding, updating, deleting, searching, and sorting skin care products in a table. The implementation included a binary search algorithm for efficient searching, as well as multiple filtering options for sorting and searching based on different categories.

The coursework provided a practical understanding of Java programming, including GUI development, event handling, and data manipulation. The completion of the 'Serinity Skin Essence' system showed my effective coding abilities, project management skills, and ability to meet deadlines.
The practical understanding I gained while creating a real-world Java application was important for my growth as a developer.

In conclusion, this coursework provided an in-depth experience that improved my skills in Java development, UI design, database integration, and project management. It focused on the value of detailed testing, suitable coding techniques, and continuous learning in software development. The challenges that were faced during the project helped personal growth and ability to solve problems.

8. References

(Minh, 2012 - 2023)

(Code project, 2014)

(Java Swing | JComboBox with examples , n.d.)

(Binary Search – Data Structure and Algorithm Tutorials , n.d.)

(Filter JTable using a JcomboBox, 2024)

(Tom Wheeler, 2023) (Minh, 2012 - 2023) (Java Swing | JComboBox with examples , n.d.)

(Jaiswal, n.d.)

(Introduction to Java , n.d.)

9. Bibliography

Apache NetBeans 20. (2017-2023). From Apache NetBeans 20: https://netbeans.apache.org/front/main/about/

Tom Wheeler, O. S. (2023). *GETTING STARTED WITH THE NETBEANS PLATFORM* . From Object Computing: https://objectcomputing.com/resources/publications/sett/september-2005-getting-started-with-the-netbeans-platform

Jaiswal, S. (n.d.). *Features of Java* . From javaTpoint: https://www.javatpoint.com/features-of-java#:~:text=The%20primary%20objective%20of%20Java,also%20known%20as%20Java%20buzzwords.

Introduction to Java . (n.d.). From geeksforgeeks: https://www.geeksforgeeks.org/introduction-to-java/

Advantages of Java . (2000 - 2024). From theserverside: https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-Advantages-Benefits-Fast-Performance-Simple-Open-Typed-Features-Streams

Minh, N. H. (2012 - 2023, July 6). *JComboBox basic tutorial and examples*. From CodeJava.net: https://www.codejava.net/java-se/swing/jcombobox-basic-tutorial-and-examples

(2014, august 11). From Code project: https://www.codeproject.com/Questions/805595/Here-I-Add-A-Dialogue-Box-To-The-Frame-Now-I-Want

Java Swing | JComboBox with examples . (n.d.). From geeks for geeks: https://www.geeksforgeeks.org/java-swing-jcombobox-examples/

Binary Search – Data Structure and Algorithm Tutorials . (n.d.). From geeks for geeks: https://www.geeksforgeeks.org/binary-search/

Filter JTable using a JcomboBox. (2024). From stack overflow: https://stackoverflow.com/questions/37190269/filter-jtable-using-a-jcombobox