



**GROUP ASSIGNMENT
TECHNOLOGY PARK MALAYSIA
CT046-3-2-SDM
OBJECT ORIENTED JAVA PROGRAMMING**

HAND OUT DATE: 1 DECEMBER 2023

HAND IN DATE: 23 FEBRUARY 2024

INTAKE CODE: APD2F2311CS(DA) / APU2F2311CS(DA) / APU2F2311CS

TEAM MEMBERS	TP NUMBER
KERTANAA A/P KUMAR	TP072664
HOOUSHIANEE HARHANGI	TP066080
ABDUL MUHAIMIN AMAN	TP069510
AREEBAH IRFAN MOHAMMED	TP072275

Table of Contents

1.0 INTRODUCTION	4
2.0 PROPOSED SYSTEM	5
3.0 SCOPE	6
4.0 OBJECTIVES	6
5.0 LIMITATIONS	7
6.0 SYSTEM DESIGN.....	8
6.1 USE CASE DIAGRAM (SYSTEM ACTIVITY) – Kertanaa A/P Kumaar	8
6.2 USE CASE DIAGRAM (CUSTOMER)	9
6.3 USE CASE DIAGRAM (MANAGER)	10
6.4 USE CASE DIAGRAM (ADMIN).....	11
6.5 CLASS DIAGRAM (SYSTEM ACTIVITY) – Abdul Muhammin Aman	12
6.5.1 CLASS DIAGRAM DESCRIPTION	13
7.0 CONCEPTS USED.....	15
7.1 ABDUL MUHAMMIN AMAN	15
7.1.0 ENCAPSULATION	15
7.1.1 POLYMORPHISM.....	16
7.1.2 INHERITANCE.....	17
7.2 KERTANAA A\P KUMAAR.....	18
7.2.0 ENCAPSULATION	18
7.2.1 INHERITANCE:	19
7.3 HOUSHIANEE HARHANGI	20
7.3.0 INHERITANCE.....	20
7.3.1 ENCAPSULATION	20
7.3.2 ABSTRACTION.....	21
7.3.3 POLYMORPHISM.....	21
8.0 IMPLEMENTATION.....	22
8.1 ADMINISTRATOR- ABDUL MUHAMMIN AMAN	22
8.2 SALESPERSON - KERTANAA A/P KUMAAR.....	34

8.3 OFFICER – AREEBAH IRFAN MOHAMMED	47
8.4 AUTHORIZATION AND AUTHENTICATION – HOUSHIANEE HARHANGI	53
9.0 ADDITIONAL FEATURES	63
 9.1 ABDUL MUHAIMIN AMAN	63
9.1.0 BACKGROUND IMAGE	63
9.1.1 STYLED BUTTONS	64
9.1.2 GRADIENT COLOUR BACKGROUND	65
9.1.3 TIMER FOR DIALOG BOX.....	66
9.1.3 ALTERNATING ROW COLOURS	67
9.1.4 CENTERING FRAME TO SCREEN.....	68
 10.0 CONCLUSION	69
 11.0 WORKLOAD MATRIX.....	70

1.0 INTRODUCTION

In this project, we will create a window-based system to manage sales orders, invoicing, and reports for Yoyo-Furniture, a furniture firm. The system will be constructed in an object-oriented manner, with numerous objects representing real-world things like as sales orders, invoices, reports, and customers. The system will enable users to execute a variety of tasks, such as processing sales orders, producing invoices, generating reports, managing user accounts, updating and deleting, generating invoices, listing and modifying sales orders and so on.

With a user-friendly interface, the GUI-driven system will enable people to engage with it. The system will be built to handle a number of situations, such as processing sale orders, creating invoices using the data from the sales orders, producing reports that shed light on the sales success of the business, and maintaining user accounts.

We will use polymorphism, inheritance, and encapsulation—aspects of object-oriented programming—to create the system. Because of its modular architecture, the system will be simple to scale and maintain. Additionally, the system will be built with security in mind, incorporating user identification and authorization protocols.

An description of the system's architecture, implementation specifics, and samples of code showcasing its object-oriented capabilities are all included in this report. In addition, we will include images of the user interface and a thorough explanation of all the features that the system will support.

2.0 PROPOSED SYSTEM

A window-based system that mimics the workflow of handling sale orders, invoicing, and reports is the proposed system for Yoyo-Furniture. An object-oriented design methodology will be used to create the system, with different objects standing in for actual real-world things including users, reports, invoices, and sale orders. Users with varying levels of access, including officers, administrators, and salespeople, will be able to handle sale orders and reports through the system.

The officer will be able to use features including creating reports, monitoring the status of sold products, processing sale orders when a transaction is approved or closed, and maintaining personal accounts. In addition to searching and editing sale orders, the officer will be able to submit sales for production, issue sale invoices, and generate reports like the work-done and approved/closed sale reports.

The administrator will be able to perform tasks such as generating reports and handling employee and personal accounts. In addition to creating reports like the work done report and the approved/closed sale report, the administrator will have the ability to manage the profiles of officers and salesperson

The salesperson will be able to manage personal profiles, sale order quotations, and view a list of all personal sales orders. The salesperson will be able to generate, alter, remove, and search sale order quotations, as well as list of all disapproved and authorized personal sales orders.

We shall list the different properties required to describe the entities (sale order, invoice, report, user,) in the solution architecture. Additionally, we will list and include the techniques required for every entity, including report generation, modification, removal, and search. The system will verify data integrity and look for duplicate entries.

To get at the object-oriented design principles, we will logically assume certain functional needs. For permission and authentication reasons, for instance, we will assume that every user has a distinct username and password. Additionally, we will presume that every sale order has a distinct ID in order to avoid duplication and guarantee precise tracking.

3.0 SCOPE

1. For a furniture firm, design and construct a window-based system to manage selling orders, invoicing, and reports.
2. To represent real-world elements like sale orders, invoices, reports, and users, utilize an object-oriented approach.
3. Assign distinct access privileges to officers, administrators, and salespeople.
4. Put into practice features like report generation, modifying, removal, and searching.

4.0 OBJECTIVES

1. Make the management of sale orders, invoicing, and reports easier to use using an intuitive UI.
 2. Use object-oriented design concepts to create representations of real-world objects.
 3. Implementing access permissions and authentication will help to ensure data security and integrity.
 4. Execute features including report generation, editing, deletion, and searching.
- To make sure the system is bug-free and satisfies functional requirements, test it.

5.0 LIMITATIONS

1. The system won't be linked to an actual database or production environment; instead, it will be a simulation.
2. In a real-world scenario, the system might not account for every event that could occur.
3. Because the system is a simulation rather than an application that is ready for production, it can have constraints with regard to performance, scalability, and security.
4. It's probable that not every edge case and error handling situation is covered by the system.
5. It's possible that the system cannot be adjusted or configured to satisfy certain company needs
6. It's possible that the system hasn't been thoroughly tested and has errors or problems that weren't discovered during development.
7. It's possible that not all hardware setups or operating systems will work with the system.
8. Users with impairments might not be able to access the system completely.
9. It's possible that the system isn't completely adjusted for varying screen sizes and resolutions.
10. It's possible that the system does not adhere to all rules and industry norms.
11. It's possible that the system can't manage a lot of data or many users at once.
12. There's a chance that the system won't work with other apps or platforms.
13. It's possible that the system can't manage processes or complicated business rules.
14. It's possible that the system cannot deliver analytics or data in real time.
15. Requirements for translation or internationalization could be beyond the system's capabilities.

6.0 SYSTEM DESIGN

6.1 USE CASE DIAGRAM (SYSTEM ACTIVITY) – Kertanaa A/P

Kumaar

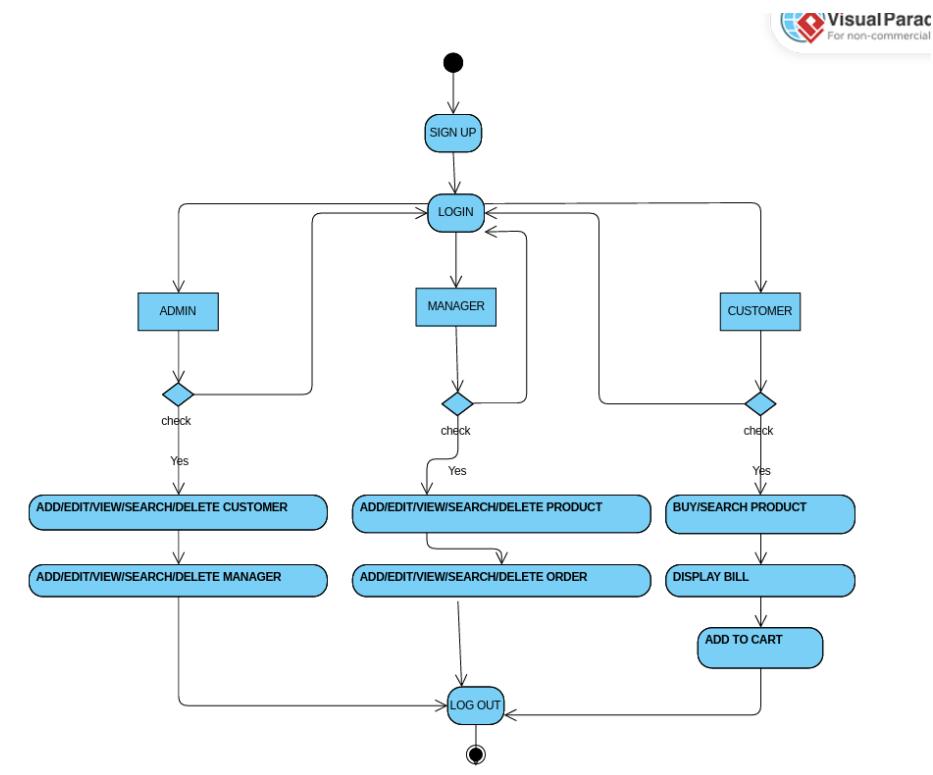


Figure 1 shows the Activity Diagram

System users have to login to the system using valid username and password. Depending on their categories so that they will be able to open their respective pages.

6.2 USE CASE DIAGRAM (CUSTOMER)

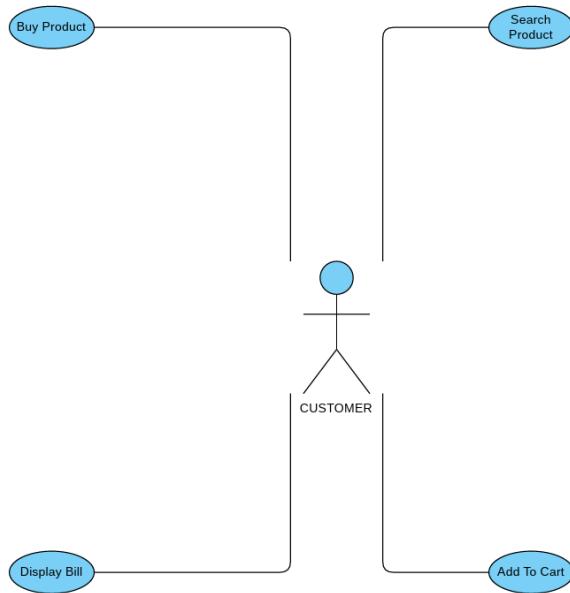


Figure 2 shows the Use Case Diagram of Customer

According to the use case diagram, the Customer role's interactions are mostly focused on product browsing, ordering, and shipping tracking. Customers may go through the products that are offered, choose the things that they want to buy, and then place their purchases. They may also track shipments to keep an eye on the delivery process and examine the status of their orders.

6.3 USE CASE DIAGRAM (MANAGER)

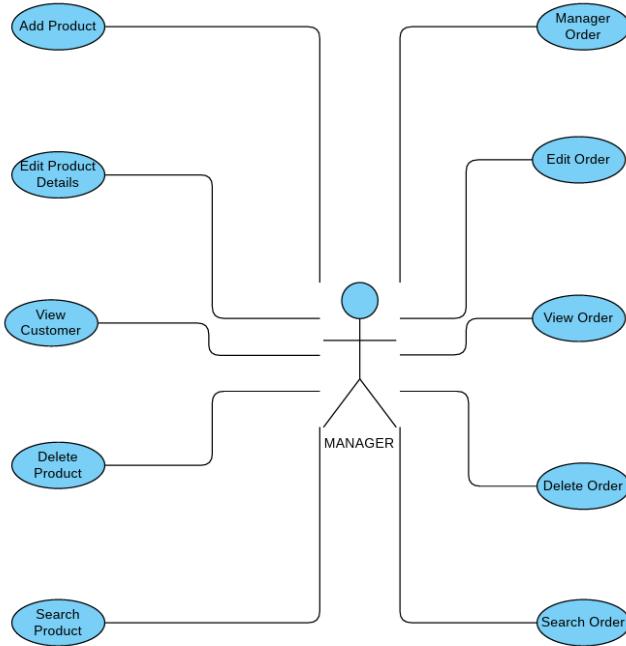


Figure 3 shows the Use Case Diagram of Manager

The Manager job holds more power and is largely responsible for approving or rejecting sales orders. After reviewing outstanding sales orders, managers decide whether to approve or reject them based on several factors, including availability, cost, and client preferences. Furthermore, managers have access to sales reports, which give them insightful knowledge about systemic patterns and overall sales success.

6.4 USE CASE DIAGRAM (ADMIN)

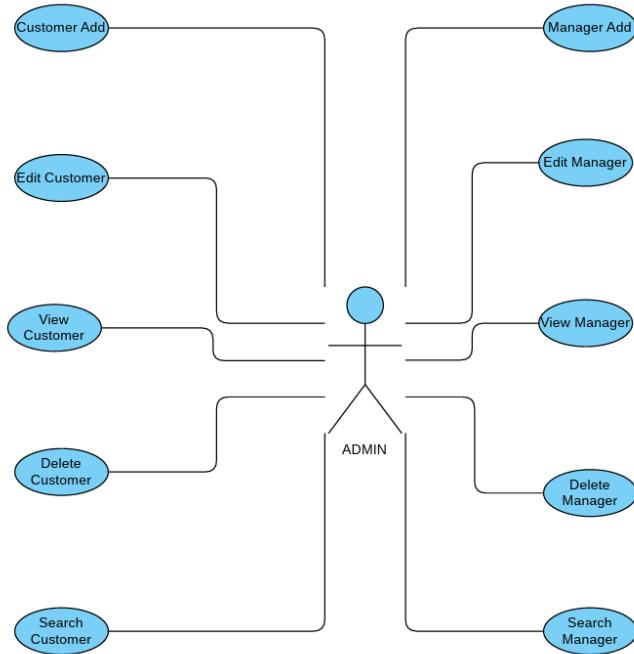
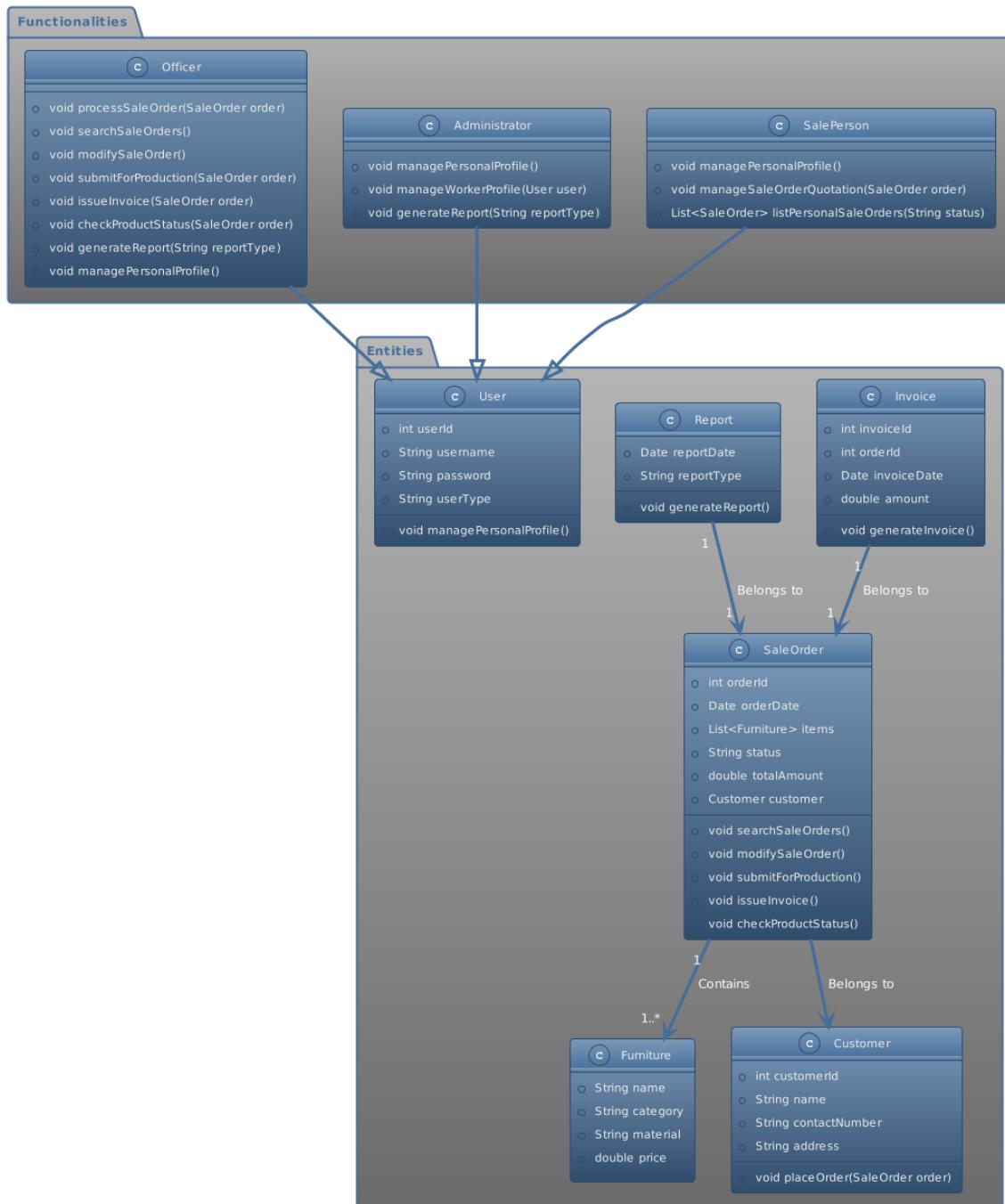


Figure 4 shows the Use Case Diagram of Admin

In contrast, the admin function oversees overseeing and managing the system. Administrators have the power to add, edit, and remove employees from the system, such as officers and salespeople, from their profiles as needed. Moreover, administrators have access to reports and sales data, which gives them the ability to assess system performance and decide after gaining knowledge.

6.5 CLASS DIAGRAM (SYSTEM ACTIVITY) – Abdul Muhammin Aman



6.5.1 CLASS DIAGRAM DESCRIPTION

Entity/Functionality	Attributes	Brief Description
Officer	<ul style="list-style-type: none"> - userID
- username
- password
- userType (officer)
 	Represents an officer with access to specific functionalities, such as processing sale orders, generating reports, and managing personal profiles.
Administrator	<ul style="list-style-type: none"> - userID
- username
- password
- userType (administrator)
 	Represents an administrator with access to managing personal profiles, worker profiles, and generating reports.
Customer	<ul style="list-style-type: none"> - customerID
- name
- contactNumber
- address
 	Represents a customer, with attributes for customer ID, name, contact number, and address.
Furniture	<ul style="list-style-type: none"> - name
- category
- material
- price
 	Represents a furniture product, with attributes for name, category, material, and price.
Report	<ul style="list-style-type: none"> - reportDate
- reportType
 	Represents a report, with attributes for report date and report type.
Sales Invoice	<ul style="list-style-type: none"> - invoiceID
- invoiceDate
- amount
- order
 	Represents a sales invoice, with attributes for invoice ID, invoice date, amount, and order.

Entity/Functionality	Attributes	Brief Description
processSaleOrder(SaleOrder order)	- order 	A functionality for officers to process a sale order upon sale approval or closed sale.
generateReport(String reportType)	- reportType 	A functionality for officers and administrators to generate reports based on report type.
managePersonalProfile()	- user 	A functionality for users to manage their personal profile.
placeOrder(SaleOrder order)	- order - customer 	A functionality for customers to place an order by providing the order and customer details.

7.0 CONCEPTS USED

7.1 ABDUL MUHAIMIN AMAN

7.1.0 ENCAPSULATION

```
1 package oodjaman;
2
3+ import javax.imageio.ImageIO;[]
10
11 public class oodjaman2 extends JFrame {
12
13     private AdministratorProfile administratorProfile;
14
15     private JTextField emailField;
16     private JTextField phoneNumberField;
17     private JTextField addressField;
18     private JButton saveButton;
19     private JButton goBackButton;
20
21+     public oodjaman2() {
22         administratorProfile = AdministratorProfileManager.readProfile();
23
24         setTitle("Administrator Profile");
25         setSize(400, 300);
26         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27
28         createUI();
29     }
--
```

Encapsulation is implemented in the oodjaman2 class using private methods (createUI(), saveChanges(), createStyledButton(String text)), private inner class (BackgroundPanel), and private fields (administratorProfile, emailField, phoneNumberField, addressField, saveButton, goBackButton). These aspects are protected from external classes by being marked as private, which limits access to the class itself and hides the internal state and implementation details. By enabling limited access to the class's functionality and encouraging information concealment, this encapsulation helps to create a more modular and manageable construction.

7.1.1 POLYMORPHISM

```
54
55     jTable reportTable = new JTable(tableModel) {
56         @Override
57         public Component prepareRenderer(TableCellRenderer renderer, int row, int column) {
58             Component comp = super.prepareRenderer(renderer, row, column);
59
60             // Handle selection state
61             if (isRowSelected(row)) {
62                 comp.setBackground(new Color(70, 130, 180)); // Light blue for selected row
63             } else {
64                 comp.setBackground(row % 2 == 0 ? Color.WHITE : new Color(255, 255, 255)); // Alternating colors
65             }
66
67             return comp;
68         }
69     };
70 }
```

This code snippet overrides the `prepareRenderer` function to adjust table cell rendering based on selection status and row index. This is an example of polymorphism, as the overridden method offers a customized implementation inside the context of `ClosedSaleReport`.

7.1.2 INHERITANCE

```

// Custom JPanel with background image
private class BackgroundPanel extends JPanel {
    private Image backgroundImage;

    public BackgroundPanel(String imagePath) {
        try {
            backgroundImage = ImageIO.read(new File(imagePath));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (backgroundImage != null) {
            g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
        }
    }
}

// Create a blue gradient background for JOptionPane
private JPanel createBlueGradientBackground() {
    JPanel panel = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            super.paintComponent(g);
            Graphics2D g2d = (Graphics2D) g;
            int w = getWidth();
            int h = getHeight();
            Color color1 = new Color(70, 130, 180); // Light blue
            Color color2 = new Color(30, 60, 90); // Dark blue
            GradientPaint gp = new GradientPaint(0, 0, color1, 0, h, color2);
            g2d.setPaint(gp);
            g2d.fillRect(0, 0, w, h);
        }
    };
    return panel;
}
}

```

Java inheritance is demonstrated by the `BackgroundPanel` class and the `createBlueGradientBackground` function, which both extend or generate a subclass of the `JPanel` class. The custom panels offers extra or altered functionality while inheriting the `JPanel` class's attributes and behaviours.

7.2 KERTANAA A\P KUMAAR

7.2.0 ENCAPSULATION

```
public class Manage_Personal_Profile_Page extends JFrame {  
    private static final long serialVersionUID = 1L;  
    private JPanel contentPane;  
    private JTextField textField_Name;  
    private JTextField textField_Position;  
    private JTextField textField_Email;  
    private JTextField textField_PhoneNumber;  
    private JTextField textField_Address;
```

From the code shown above, all the variables have been declared with a private access method which promotes encapsulation that hide all the data from being modified by non-relevant objects, as opposed to structured programming in that data will act as global data without any privilege access method to control the access. Furthermore, for above variables to be accessed, mutator and access method must be implemented.

7.2.1 INHERITANCE:

```

public Manage_Personal_Profile_Page() {
    setTitle("Manage Personal Profile Page");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setBounds(100, 100, 501, 393);
    contentPane = new JPanel();
    contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
    setContentPane(contentPane);
    contentPane.setLayout(null);

    JLabel lblNewLabel_Name = new JLabel("Name:");
    lblNewLabel_Name.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
    lblNewLabel_Name.setBounds(145, 14, 87, 24);
    contentPane.add(lblNewLabel_Name);

    textField_Name = new JTextField();
    textField_Name.setBounds(233, 18, 117, 20);
    contentPane.add(textField_Name);
    textField_Name.setColumns(10);

    JLabel lblNewLabel_Position = new JLabel("Position:");
    lblNewLabel_Position.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
    lblNewLabel_Position.setBounds(145, 49, 87, 24);
    contentPane.add(lblNewLabel_Position);

    textField_Position = new JTextField();
    textField_Position.setBounds(233, 53, 117, 20);
    contentPane.add(textField_Position);
    textField_Position.setColumns(10);

    JLabel lblNewLabel_Email = new JLabel("Email:");
    lblNewLabel_Email.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
    lblNewLabel_Email.setBounds(145, 84, 87, 24);
    contentPane.add(lblNewLabel_Email);

    textField_Email = new JTextField();
    textField_Email.setBounds(233, 88, 200, 20);
    contentPane.add(textField_Email);

    JLabel lblNewLabel_PhoneNumber = new JLabel("Phone Number:");
    lblNewLabel_PhoneNumber.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
    lblNewLabel_PhoneNumber.setBounds(85, 119, 136, 24);
    contentPane.add(lblNewLabel_PhoneNumber);

    textField_PhoneNumber = new JTextField();
    textField_PhoneNumber.setBounds(233, 123, 117, 20);
    contentPane.add(textField_PhoneNumber);
    textField_PhoneNumber.setColumns(10);

    JLabel lblNewLabel_Address = new JLabel("Address:");
    lblNewLabel_Address.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
    lblNewLabel_Address.setBounds(145, 154, 87, 24);
    contentPane.add(lblNewLabel_Address);

    textField_Address = new JTextField();
    textField_Address.setBounds(233, 158, 200, 20);
    contentPane.add(textField_Address);
    textField_Address.setColumns(10);

    // Save Button
    JButton btnSave = new JButton("Save");
    btnSave.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            saveProfile();
            // Add code to display success message
            JLabel successLabel = new JLabel("Profile saved successfully!");
            successLabel.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
            successLabel.setBounds(180, 250, 200, 30);
            contentPane.add(successLabel);
            contentPane.revalidate();
            contentPane.repaint();
        }
    });
    btnSave.setBounds(180, 200, 100, 30);
    contentPane.add(btnSave);
}
}

```

The code supplied uses inheritance, where the `JFrame` class is extended by the `Manage_Personal_Profile_Page` class. {Manage_Personal_Profile_Page} may become a graphical window by inheriting all of `JFrame`'s methods and properties thanks to this inheritance. Several Swing components, such as JLabel, JTextField, and JButton, are added to the window's content pane inside the Manage_Personal_Profile_Page constructor. These components inherit the JFrame methods for component placement and addition. Furthermore, event handling is implemented, whereby the `JButton` class, which is derived from `JFrame`, provides an action listener to control the click event of the "Save" button.

7.3 HOUSHIANEE HARHANGI

7.3.0 INHERITANCE

```
public Administrator_Account(String username) {  
    initComponents();  
    user = username;  
}
```

The inherited methods of JFrame are responsible for displaying, sizing, and closing windows. This is a prerequisite for all GUI software.

It is inherited from JFrame the capability to add GUI components such as icons, panels, and labels. Thus, modular user interface construction is possible through the addition of components to the frame in order to form a unified structure.

By specifying a username in the constructor, a logical user-centric interface is demonstrated. This design decision demonstrates an enhanced comprehension of the usability principles of object-oriented programming (OOP) software. By customizing the user experience, the username exemplifies how inheritance can significantly enhance functionality.

7.3.1 ENCAPSULATION

```
private String user;
```

Access to private user variables is restricted to the Administrator_Account class.

Encapsulation serves as a safeguard against unauthorized access to user data. Possibly additional methods (not fully illustrated in the provided excerpts), including getters, setters, and business logic methods, as well as the class's constructor, regulate the access and modification of this data.

Encapsulating data in this manner regulates the administration of application states, thereby enhancing data integrity and security.

7.3.2 ABSTRACTION

```
private void initComponents() {  
    ...  
}
```

The `initComponents` function simplifies the configuration and initialization of GUI components. This method exemplifies the concept of abstraction by concealing from the application logic the complex GUI setup elements, such as generating components, designating their properties, and adding them to the frame. Users of the `Administrator_Account` class are not required to comprehend GUI assembly or component initialization in order to utilize the class.

7.3.3 POLYMORPHISM

```
private void initComponents() {  
    ...  
}
```

The Java.awt file. The polymorphic management of component subclasses, such as `JLabel` and `JButton`, is possible with Swing container classes.

Polymorphism permits adaptable administration of GUI components. Enhancing scalability and adaptability, it integrates the interactions of the `Administrator_GenerateReport` class with various component types. Although they may vary, the process of incorporating various `jPanel1` components remains consistent.

For report generation, the `initComponents` method facilitates the process of initializing and configuring GUI components.

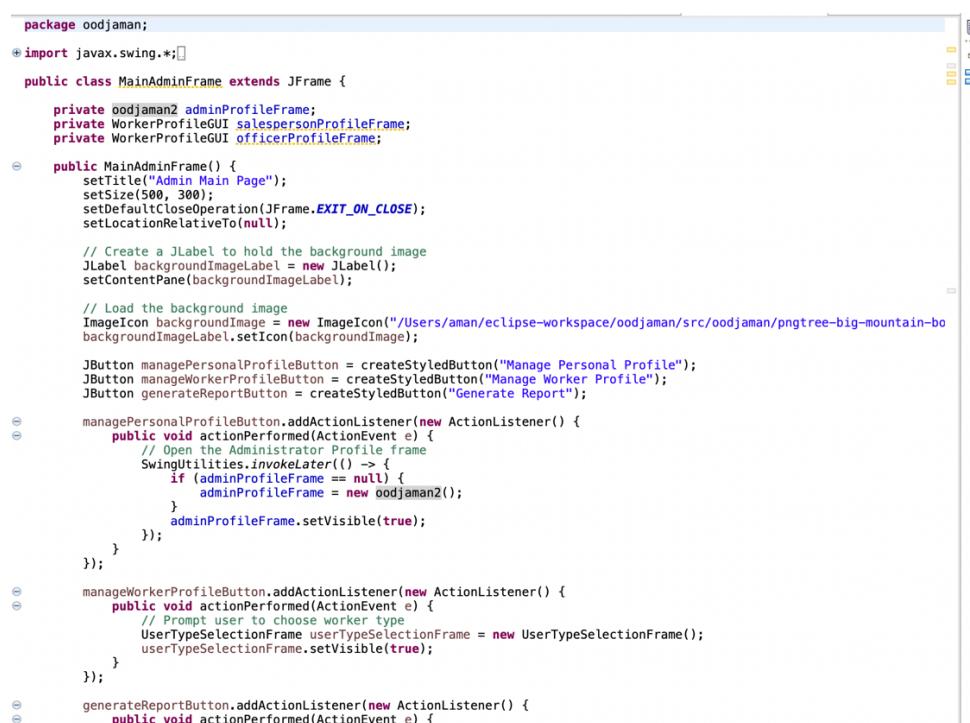
8.0 IMPLEMENTATION

8.1 ADMINISTRATOR- ABDUL MUHAIMIN AMAN

CODE:

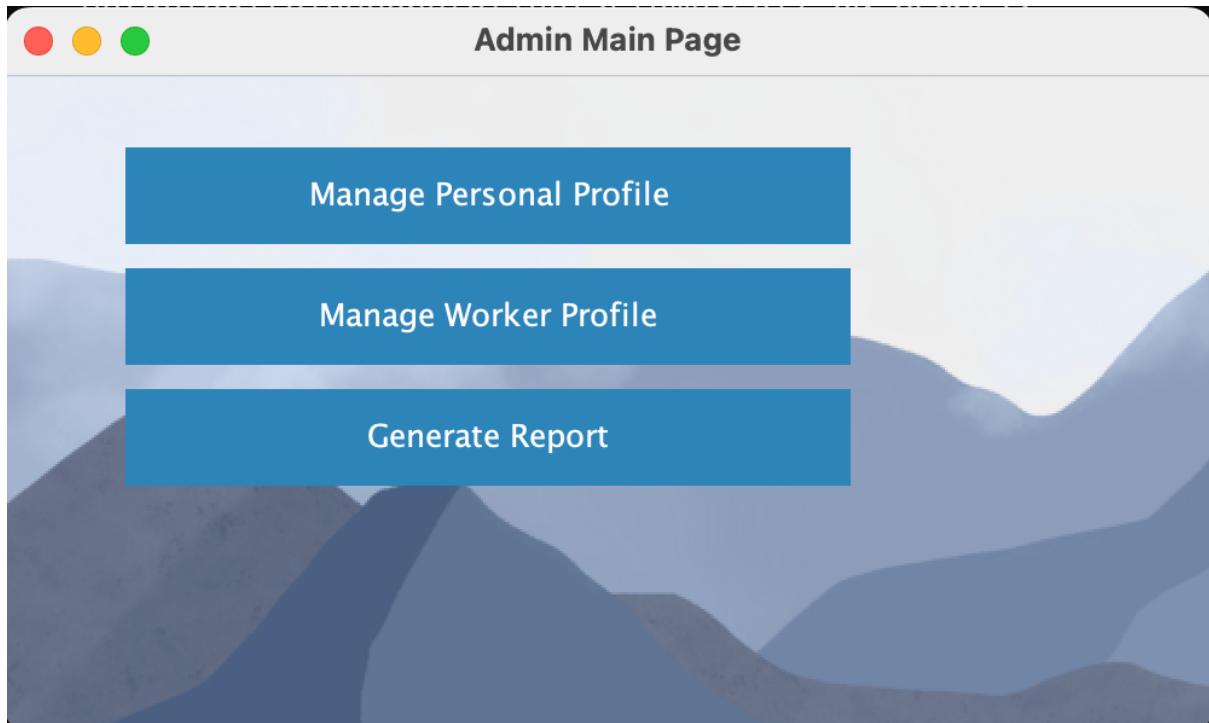
```

52@     generateReportButton.addActionListener(new ActionListener() {
53@         public void actionPerformed(ActionEvent e) {
54@             // Open the SalesReportMain frame
55@             SwingUtilities.invokeLater(() -> {
56@                 new SalesReportMain().setVisible(true);
57@             });
58@         });
59@     });
60@ 
61@     // Set layout manager to null for the content pane
62@     setLayout(null);
63@ 
64@     // Set bounds for the buttons and background image
65@     managePersonalProfileButton.setBounds(50, 30, 300, 40);
66@     manageWorkerProfileButton.setBounds(50, 80, 300, 40);
67@     generateReportButton.setBounds(50, 130, 300, 40);
68@     backgroundImageLabel.setBounds(0, 0, getWidth(), getHeight());
69@ 
70@     // Add buttons to the content pane
71@     add(managePersonalProfileButton);
72@     add(manageWorkerProfileButton);
73@     add(generateReportButton);
74@ }
75@ 
76@ private JButton createStyledButton(String text) {
77@     JButton button = new JButton(text);
78@     button.setBackground(new Color(70, 130, 180)); // Light blue
79@     button.setForeground(Color.WHITE);
80@     button.setFocusPainted(false);
81@     button.setBorderPainted(false);
82@     button.setOpaque(true);
83@     return button;
84@ }
85@ 
86@ public static void main(String[] args) {
87@     SwingUtilities.invokeLater(() -> {
88@         MainAdminFrame mainAdminFrame = new MainAdminFrame();
89@         mainAdminFrame.setVisible(true);
90@     });
91@ }
92@ }
93@ }
94@ 
```



```

package oodjaman;
import javax.swing.*;
public class MainAdminFrame extends JFrame {
    private oodjaman2 adminProfileFrame;
    private WorkerProfileGU salespersonProfileFrame;
    private WorkerProfileGU officerProfileFrame;
    public MainAdminFrame() {
        setTitle("Admin Main Page");
        setSize(500, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        // Create a JLabel to hold the background image
        JLabel backgroundImageLabel = new JLabel();
        setContentPane(backgroundImageLabel);
        // Load the background image
        ImageIcon backgroundImage = new ImageIcon("/Users/aman/eclipse-workspace/oodjaman/src/oodjaman/pngtree-big-mountain-bo
        backgroundImageLabel.setIcon(backgroundImage);
        JButton managePersonalProfileButton = createStyledButton("Manage Personal Profile");
        JButton manageWorkerProfileButton = createStyledButton("Manage Worker Profile");
        JButton generateReportButton = createStyledButton("Generate Report");
        managePersonalProfileButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Open the Admin Profile frame
                SwingUtilities.invokeLater(() -> {
                    if (adminProfileFrame == null) {
                        adminProfileFrame = new oodjaman2();
                    }
                    adminProfileFrame.setVisible(true);
                });
            }
        });
        manageWorkerProfileButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Prompt user to choose worker type
                UserTypeSelectionFrame userTypeSelectionFrame = new UserTypeSelectionFrame();
                userTypeSelectionFrame.setVisible(true);
            }
        });
        generateReportButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
```

GUI:**DESCRIPTION:**

A Java Swing JFrame that expands the capabilities of (GUI) is the MainAdminFrame class. The application's entry point is the public static void main function. The frame's position, size, closure operation, and title are all set inside the constructor. It uses an ImageIcon to load the background picture and configures styled buttons with action listeners. The content pane is set using the setContentPane function, and layout management is turned off by using setLayout(null), which permits component placement by hand. Lastly, the MainAdminFrame object is created and made visible by the main function, which then starts the GUI.

CODE:

```

1 package oodjaman;
2
3 import javax.imageio.ImageIO;
4
5 public class oodjaman2 extends JFrame {
6
7     private AdministratorProfile administratorProfile;
8
9     private JTextField emailField;
10    private JTextField phoneNumberField;
11    private JTextField addressField;
12    private JButton saveButton;
13    private JButton goBackButton;
14
15    public oodjaman2() {
16        administratorProfile = AdministratorProfileManager.readProfile();
17
18        setTitle("Administrator Profile");
19        setSize(400, 300);
20        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
21
22        createUI();
23    }
24
25    private void createUI() {
26        JPanel mainPanel = new BackgroundPanel("/Users/aman/eclipse-workspace/oodjaman/src/oodjaman/pngtree-big-mountain-bord");
27        mainPanel.setLayout(new GridLayout(6, 2, 10, 10));
28
29        mainPanel.add(new JLabel("Name:"));
30        mainPanel.add(new JTextField(administratorProfile.getName())).setEnabled(false);
31        mainPanel.add(new JLabel("Position:"));
32        mainPanel.add(new JTextField(administratorProfile.getPosition())).setEnabled(false);
33        mainPanel.add(new JLabel("Email:"));
34        emailField = new JTextField(administratorProfile.getEmail());
35        mainPanel.add(emailField);
36        mainPanel.add(new JLabel("Phone Number:"));
37        phoneNumberField = new JTextField(administratorProfile.getPhoneNumber());
38        mainPanel.add(phoneNumberField);
39        mainPanel.add(new JLabel("Address:"));
40        addressField = new JTextField(administratorProfile.getAddress());
41        mainPanel.add(addressField);
42
43        saveButton = createStyledButton("Save");
44        saveButton.addActionListener(new ActionListener() {
45            @Override
46            public void actionPerformed(ActionEvent e) {
47                saveChanges();
48            }
49        });
50
51        // Create "Go Back" button
52        goBackButton = createStyledButton("Back");
53        goBackButton.addActionListener(new ActionListener() {
54            @Override
55            public void actionPerformed(ActionEvent e) {
56                dispose(); // Close the current frame
57            }
58        });
59
60        mainPanel.add(saveButton);
61        mainPanel.add(goBackButton);
62        setLocationRelativeTo(null);
63
64        getContentPane().add(mainPanel, BorderLayout.CENTER);
65    }
66
67    private void saveChanges() {
68        administratorProfile.setEmail(emailField.getText());
69        administratorProfile.setPhoneNumber(phoneNumberField.getText());
70        administratorProfile.setAddress(addressField.getText());
71
72        AdministratorProfileManager.writeProfile(administratorProfile);
73
74        // Create a custom JDialog with a blue gradient background
75        JDialog dialog = new JDialog(this, "Success", true);
76        dialog.setContentPane(createBlueGradientBackground());
77        dialog.setSize(300, 100);
78        dialog.setLocationRelativeTo(this);
79
80        // Add a label to the dialog
81        JLabel label = new JLabel("Changes saved successfully.");
82        label.setForeground(Color.WHITE);
83        dialog.add(label, BorderLayout.CENTER);
84
85        // Set a timer to close the dialog after 2 seconds (2000 milliseconds)
86        Timer timer = new Timer(2000, new ActionListener() {
87            @Override
88            public void actionPerformed(ActionEvent e) {
89                dialog.dispose();
90            }
91        });
92        timer.setRepeats(false); // Only run the timer once
93        timer.start();
94
95        dialog.setVisible(true);
96    }
97
98    private JButton createStyledButton(String text) {
99        JButton button = new JButton(text);
100
101    }
102
103
104    private JButton createStyledButton(String text) {
105        JButton button = new JButton(text);
106
107    }

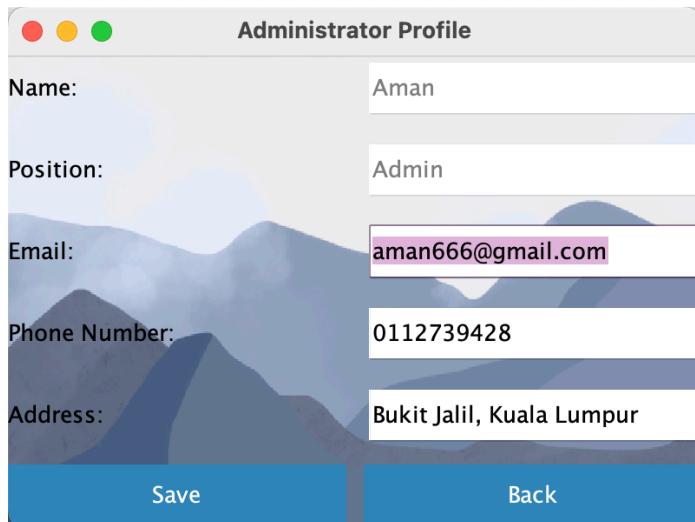
```

```

03
04  private JButton createStyledButton(String text) {
05      JButton button = new JButton(text);
06      button.setBackground(new Color(70, 130, 180)); // Light blue
07      button.setForeground(Color.WHITE);
08      button.setFocusPainted(false);
09      button.setBorderPainted(false);
10      button.setOpaque(true);
11      return button;
12  }
13
14  public static void main(String[] args) {
15      SwingUtilities.invokeLater(new Runnable() {
16          @Override
17          public void run() {
18              new oodjaman2().setVisible(true);
19          }
20      });
21  }
22
23  // Custom JPanel with background image
24  private class BackgroundPanel extends JPanel {
25      private Image backgroundImage;
26
27      public BackgroundPanel(String imagePath) {
28          try {
29              backgroundImage = ImageIO.read(new File(imagePath));
30          } catch (IOException e) {
31              e.printStackTrace();
32          }
33      }
34
35      @Override
36      protected void paintComponent(Graphics g) {
37          super.paintComponent(g);
38          if (backgroundImage != null) {
39              g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
40          }
41      }
42  }
43  private JPanel createBlueGradientBackground() {
44      JPanel panel = new JPanel() {
45          @Override
46          protected void paintComponent(Graphics g) {
47              super.paintComponent(g);
48              Graphics2D g2d = (Graphics2D) g;
49              int w = getWidth();
50              int h = getHeight();
51              Color color1 = new Color(70, 130, 180); // Light blue
52              Color color2 = new Color(30, 60, 90); // Dark blue

```

GUI:



DESCRIPTION:

A Swing `JFrame` that manages the Administrator Profile is the `oodjaman2` class. It has areas where you may examine and edit details. The `saveChanges` method modifies the profile, opens a feedback box with a backdrop gradient of blue, and closes on its own after two seconds. The user interface (UI) makes use of Swing components, and the buttons `({saveButton}` and `goBackButton{}` have been given a unique light blue style with white text to make them visually appealing.

CODE:

```

1 package oodjaman;
2
3@import javax.imageio.ImageIO;[]
10
11 public class WorkerProfileGUI extends JFrame {
12
13     private WorkerProfile workerProfile;
14     private JButton goBackButton;
15
16     private JTextField emailField;
17     private JTextField phoneNumberField;
18     private JTextField addressField;
19
20     private String workerType; // "SALESPERSON" or "OFFICER"
21
22@    public WorkerProfileGUI(String workerType, String filePath) {
23        this.workerType = workerType;
24        workerProfile = WorkerProfileManager.readWorkerProfile(workerType, filePath);
25
26        setTitle(workerType + " Profile");
27        setSize(400, 300);
28        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
29        setLocationRelativeTo(null);
30        createUI();
31    }
32
33@    private void createUI() {
34        JPanel mainPanel = new BackgroundPanel("/Users/aman/eclipse-workspace/oodjaman/src/oodjaman/pngtree-big-mountain-bord
35        mainPanel.setLayout(new GridLayout(6, 2, 10, 10));
36
37        mainPanel.add(new JLabel("Name:"));
38        mainPanel.add(new JTextField(workerProfile.getName()).setEnabled(false));
39        mainPanel.add(new JLabel("Position:"));
40        mainPanel.add(new JTextField(workerProfile.getPosition()).setEnabled(false));
41        mainPanel.add(new JLabel("Email:"));
42        emailField = new JTextField(workerProfile.getEmail());
43        mainPanel.add(emailField);
44        mainPanel.add(new JLabel("Phone Number:"));
45        phoneNumberField = new JTextField(workerProfile.getPhoneNumber());
46        mainPanel.add(phoneNumberField);
47        mainPanel.add(new JLabel("Address:"));
48        addressField = new JTextField(workerProfile.getAddress());
49        mainPanel.add(addressField);
50
51        JButton saveButton = createStyledButton("Save");
52@        saveButton.addActionListener(new ActionListener() {
53@            @Override
54            public void actionPerformed(ActionEvent e) {
55                saveChanges();
56            }
57        });
58
59        // Create "Go Back" button
60        goBackButton = createStyledButton("Go Back");
61        goBackButton.addActionListener(new ActionListener() {
62@            @Override
63            public void actionPerformed(ActionEvent e) {
64                dispose(); // Close the current frame
65            }
66        });
67
68        mainPanel.add(saveButton);
69        mainPanel.add(goBackButton);
70
71        getContentPane().add(mainPanel, BorderLayout.CENTER);
72    }
73
74@    private void saveChanges() {
75        workerProfile.setEmail(emailField.getText());
76        workerProfile.setPhoneNumber(phoneNumberField.getText());
77        workerProfile.setAddress(addressField.getText());
78
79        WorkerProfileManager.writeWorkerProfile(workerProfile, workerType);
80
81        // Create a custom JDialog with a blue gradient background
82        JDialog dialog = new JDialog(this, "Success", true);
83        dialog.setContentPane(createBlueGradientBackground());
84        dialog.setSize(300, 100);
85        dialog.setLocationRelativeTo(this);
86
87        // Add a label to the dialog
88        JLabel label = new JLabel("Changes saved successfully.");
89        label.setForeground(Color.WHITE);
90        dialog.add(label, BorderLayout.CENTER);
91
92        // Set a timer to close the dialog after 2 seconds (2000 milliseconds)
93        Timer timer = new Timer(2000, new ActionListener() {
94@            @Override
95            public void actionPerformed(ActionEvent e) {
96                dialog.dispose();
97            }
98        });
99        timer.setRepeats(false); // Only run the timer once
100        timer.start();
101
102        dialog.setVisible(true);
103    }
104
105    private JButton createStyledButton(String text) {
106        JButton button = new JButton(text);
107        button.setBackground(new Color(70, 130, 180)); // Light blue
108        button.setForeground(Color.WHITE);
109    }

```

```

106  private JButton createStyledButton(String text) {
107      JButton button = new JButton(text);
108      button.setBackground(new Color(70, 130, 180)); // Light blue
109      button.setForeground(Color.WHITE);
110      button.setFocusPainted(false);
111      button.setOpaque(true);
112      return button;
113  }
114
115
116  public static void main(String[] args) {
117      SwingUtilities.invokeLater(new Runnable() {
118          @Override
119          public void run() {
120              // Open UserTypeSelectionFrame to choose worker type
121              UserTypeSelectionFrame userTypeSelectionFrame = new UserTypeSelectionFrame();
122              userTypeSelectionFrame.setVisible(true);
123          }
124      });
125  }
126
127
128
129
130  // Custom JPanel with background image
131  private class BackgroundPanel extends JPanel {
132      private Image backgroundImage;
133
134      public BackgroundPanel(String imagePath) {
135          try {
136              backgroundImage = ImageIO.read(new File(imagePath));
137          } catch (IOException e) {
138              e.printStackTrace();
139          }
140      }
141
142      @Override
143      protected void paintComponent(Graphics g) {
144          super.paintComponent(g);
145          if (backgroundImage != null) {
146              g.drawImage(backgroundImage, 0, 0, getWidth(), getHeight(), this);
147          }
148      }
149  }
150
151  // Create a blue gradient background for JOptionPane
152  private JPanel createBlueGradientBackground() {
153      JPanel panel = new JPanel() {
154          @Override
155          protected void paintComponent(Graphics g) {
156              super.paintComponent(g); ...
157      }
158  }

```

GUI:

Name:	Areebah
Position:	Officer
Email:	areebah777@gmail.com
Phone Number:	015678342
Address:	Bukit Jalil, Kuala Lumpur, Mal.

[Save](#) [Go Back](#)

DESCRIPTION:

The WorkerProfileGUI class uses Swing to provide a graphical user interface for controlling worker profiles. It extends JFrame, uses numerous Swing components (such as JPanel and JButton), and handles events via ActionListener. It has layout control, file I/O for worker profiles, and a customizable BackgroundPanel for background pictures. The GUI allows users to change and save worker information while receiving visible feedback via a success dialog.

CODE:

```

1 package oodjaman;
2
3 import javax.swing.*;
4
5 public class SalesReportMain extends JFrame {
6
7     private JButton goBackButton;
8
9     public SalesReportMain() {
10         setTitle("Sales Report Generator");
11         setSize(400, 200);
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         setLocationRelativeTo(null);
14
15         // Set up gradient background
16         setContentPane(new JPanel() {
17             @Override
18             protected void paintComponent(Graphics g) {
19                 Graphics2D g2d = (Graphics2D) g;
20                 Color color1 = new Color(135, 206, 250); // Light blue
21                 Color color2 = new Color(255, 255, 255); // White
22                 GradientPaint gp = new GradientPaint(0, 0, color1, getWidth(), getHeight(), color2);
23                 g2d.setPaint(gp);
24                 g2d.fillRect(0, 0, getWidth(), getHeight());
25             }
26         });
27
28         JButton closedSaleReportButton = createStyledButton("Closed Sale Report");
29         JButton workDoneReportButton = createStyledButton("Work Done Report");
30         goBackButton = createStyledButton("Go Back");
31
32         closedSaleReportButton.addActionListener(new ActionListener() {
33             public void actionPerformed(ActionEvent e) {
34                 // Open Closed Sale Report frame
35                 SwingUtilities.invokeLater(() -> {
36                     new ClosedSaleReport("/Users/aman/Downloads/zip (1)/SalesQuotation.txt").setVisible(true);
37                     dispose(); // Close the main frame
38                 });
39             }
40         });
41
42         workDoneReportButton.addActionListener(new ActionListener() {
43             public void actionPerformed(ActionEvent e) {
44                 // Open Work Done Report frame
45                 SwingUtilities.invokeLater(() -> {
46                     new WorkdoneReport("/Users/aman/Downloads/zip (1)/SalesQuotation.txt").setVisible(true);
47                     dispose(); // Close the main frame
48                 });
49             }
50         });
51
52         goBackButton.addActionListener(new ActionListener() {
53             @Override
54             public void actionPerformed(ActionEvent e) {
55                 SwingUtilities.invokeLater(() -> new MainAdminFrame().setVisible(true));
56                 dispose(); // Close the current frame
57             }
58         });
59
60         // Layout setup
61
62         GroupLayout layout = new GroupLayout(getContentPane());
63         getContentPane().setLayout(layout);
64         layout.setAutoCreateGaps(true);
65         layout.setAutoCreateContainerGaps(true);
66
67         layout.setHorizontalGroup(layout.createSequentialGroup()
68             .addHorizontalGroup(layout.createParallelGroup(GroupLayout.Alignment.LEADING)
69                 .addGroup(layout.createSequentialGroup()
70                     .addComponent(closedSaleReportButton)
71                     .addComponent(workDoneReportButton)
72                     .addComponent(goBackButton))
73                 .addGroup(layout.createSequentialGroup()
74                     .addComponent(closedSaleReportButton)
75                     .addComponent(workDoneReportButton)
76                     .addComponent(goBackButton))
77             )
78         );
79
80         layout.setVerticalGroup(layout.createSequentialGroup()
81             .addGroup(layout.createParallelGroup(GroupLayout.Alignment.BASELINE)
82                 .addComponent(closedSaleReportButton)
83                 .addComponent(workDoneReportButton)
84                 .addComponent(goBackButton))
85         );
86
87         private JButton createStyledButton(String text) {
88             JButton button = new JButton(text);
89             button.setBackground(new Color(70, 130, 180)); // Light blue
90             button.setForeground(Color.WHITE);
91             button.setFocusPainted(false);
92             button.setBorderPainted(false);
93             button.setOpaque(true);
94         }
95
96     }
97 }

```

GUI:**DESCRIPTION:**

The 'SalesReportMain' class manages its layout using GroupLayout, and the 'createStyledButton' function is used to make styled buttons. The buttons ("Closed Sale Report," "Work Done Report," and "Go Back") have matching listeners, which use SwingUtilities for thread safety. When clicked, the "Closed Sale Report" and "Work Done Report" buttons open other frames ('ClosedSaleReport' and 'WorkDoneReport', respectively), while the "Go Back" button returns to the 'MainAdminFrame'. The design attempts to provide a visually appealing and responsive sales report generation interface.

CODE:

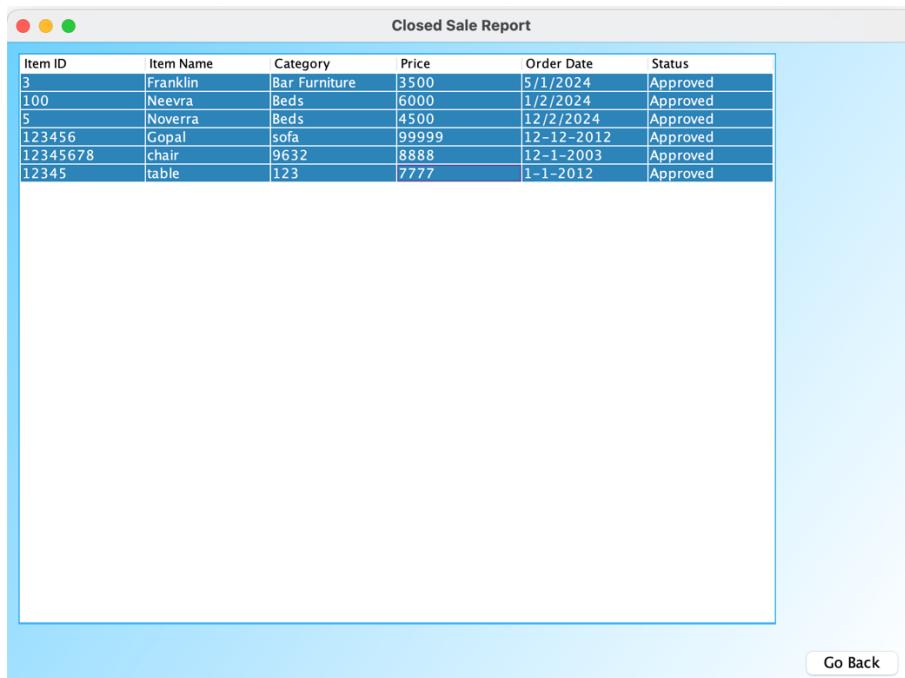
```

1 package oodjaman;
2
3 import javax.swing.*;
4
5 public class ClosedSaleReport extends JFrame {
6
7     private JButton goBackButton;
8
9     public ClosedSaleReport(String filePath) {
10         setTitle("Closed Sale Report");
11         setSize(800, 600);
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         // Set up gradient background
15         setContentPane(new JPanel() {
16             @Override
17             protected void paintComponent(Graphics g) {
18                 Graphics2D g2d = (Graphics2D) g;
19                 Color color1 = new Color(135, 206, 250); // Light orange
20                 Color color2 = new Color(255, 255, 255); // Coral
21                 GradientPaint gp = new GradientPaint(0, 0, color1, getWidth(), getHeight(), color2);
22                 g2d.setPaint(gp);
23                 g2d.fillRect(0, 0, getWidth(), getHeight());
24             }
25         });
26
27         List<String[]> approvedSales = getApprovedSales(filePath);
28
29         // Create table model
30         String[] columnNames = {"Item ID", "Item Name", "Category", "Price", "Order Date", "Status"};
31         DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0) {
32             @Override
33             public Class<?> getColumnClass(int column) {
34                 return getValueAt(0, column).getClass();
35             }
36         };
37
38         // Add approved sales to table model
39         for (String[] sale : approvedSales) {
40             tableModel.addRow(sale);
41         }
42
43         JTable reportTable = new JTable(tableModel) {
44             @Override
45             public Component prepareRenderer(TableCellRenderer renderer, int row, int column) {
46                 Component comp = super.prepareRenderer(renderer, row, column);
47
48                 // Handle selection state
49                 if (isRowSelected(row)) {
50                     comp.setBackground(new Color(70, 130, 180)); // Light blue for selected row
51                 }
52             }
53
54             reportTable.setFillsViewportHeight(false);
55             JScrollPane scrollPane = new JScrollPane(reportTable);
56
57             // Create "Go Back" button
58             goBackButton = new JButton("Go Back");
59             goBackButton.addActionListener(new ActionListener() {
60                 @Override
61                 public void actionPerformed(ActionEvent e) {
62                     dispose(); // Close the current frame
63                     SwingUtilities.invokeLater(() -> new SalesReportMain().setVisible(true));
64                 }
65             });
66
67             // Use GridBagLayout to center the table and add the "Go Back" button
68             getContentPane().setLayout(new GridBagLayout());
69             GridBagConstraints constraints = new GridBagConstraints();
70             constraints.gridx = 0;
71             constraints.gridy = 0;
72             constraints.weightx = 1.0;
73             constraints.weighty = 1.0;
74             constraints.fill = GridBagConstraints.BOTH;
75             constraints.insets = new Insets(10, 10, 10, 10);
76             getContentPane().add(scrollPane, constraints);
77
78             constraints.gridx = 1;
79             constraints.gridy = 1;
80             constraints.weightx = 0;
81             constraints.weighty = 0;
82             constraints.fill = GridBagConstraints.NONE;
83             constraints.anchor = GridBagConstraints.PAGE_END;
84             constraints.insets = new Insets(10, 10, 10, 10);
85             getContentPane().add(goBackButton, constraints);
86
87             setLocationRelativeTo(null); // Center the frame on the screen
88             setVisible(true);
89         }
90
91         private List<String[]> getApprovedSales(String filePath) {
92             List<String[]> approvedSales = new ArrayList<>();
93
94             try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
95                 String line;
96
97                 while ((line = reader.readLine()) != null) {
98                     String[] parts = line.split("\t"); // Assuming tab-separated values
99
100                    if (parts.length >= 6) {
101                        String status = parts[5].trim().toLowerCase();
102
103                        // Check if the sale is approved
104                        if (status.equals("approved")) {
105                            approvedSales.add(parts);
106                        }
107                    }
108                }
109            } catch (IOException e) {
110                e.printStackTrace();
111            }
112        }
113    }
114}
```

```

109     private List<String[]> getApprovedSales(String filePath) {
110         List<String[]> approvedSales = new ArrayList<>();
111
112         try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
113             String line;
114
115             while ((line = reader.readLine()) != null) {
116                 String[] parts = line.split("\t"); // Assuming tab-separated values
117
118                 if (parts.length >= 6) {
119                     String status = parts[5].trim().toLowerCase();
120
121                     // Check if the sale is approved
122                     if (status.equals("approved")) {
123                         approvedSales.add(parts);
124                     }
125                 }
126             }
127         } catch (IOException e) {
128             e.printStackTrace(); // Handle the exception appropriately
129         }
130
131         return approvedSales;
132     }
133
134     public static void main(String[] args) {
135         SwingUtilities.invokeLater(() -> {
136             new ClosedSaleReport("/Users/aman/Downloads/zip (1)/SalesQuotation.txt");
137         });
138     }
139 }

```

GUI:**DESCRIPTION:**

The ClosedSaleReport class uses Swing for its GUI, which includes a JTable displaying authorized sales data and a gradient backdrop. By using SwingUtilities.invokeLater in the main function guarantees thread safety. The class changes the JTable dynamically after reading sales data from a file and removing authorized sales. The "Go Back" button returns to the sales report's main frame with ease. The program uses a GridBagLayout to keep its layout tidy and aesthetically pleasing. In order to mitigate any concurrent concerns, it is imperative that invokeLater be used to ensure that Swing components are generated and changed on the EDT.

CODE:

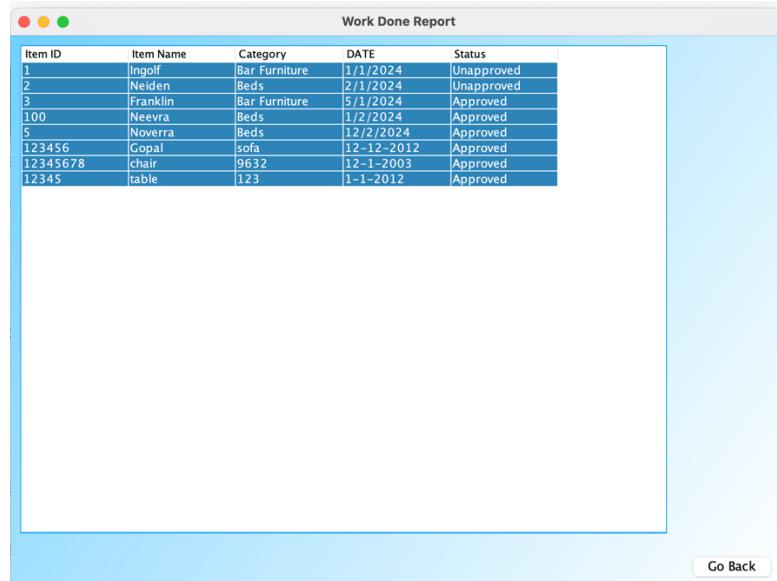
```

1 package oodjaman;
2
3 import javax.swing.*;
4
5 public class WorkDoneReport extends JFrame {
6
7     private JButton goBackButton;
8
9     public WorkDoneReport(String filePath) {
10         setTitle("Work Done Report");
11         setSize(800, 600);
12         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13
14         // Set up gradient background
15         setContentPane(new JPanel() {
16             @Override
17             protected void paintComponent(Graphics g) {
18                 Graphics2D g2d = (Graphics2D) g;
19                 Color color1 = new Color(135, 206, 250); // Light blue
20                 Color color2 = new Color(255, 255, 255); // White
21                 GradientPaint gp = new GradientPaint(0, 0, color1, getWidth(), getHeight(), color2);
22                 g2d.setPaint(gp);
23                 g2d.fillRect(0, 0, getWidth(), getHeight());
24             }
25         });
26
27         List<String[]> salesData = getSalesData(filePath);
28
29         // Create table model
30         String[] columnNames = {"Item ID", "Item Name", "Category", "Price", "DATE", "Status"};
31         DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0) {
32             @Override
33             public Class<?> getColumnClass(int column) {
34                 return getValueAt(0, column).getClass();
35             }
36         };
37
38         // Add sales data to table model
39         for (String[] sale : salesData) {
40             tableModel.addRow(sale);
41         }
42
43         JTable reportTable = new JTable(tableModel) {
44             @Override
45             public Component prepareRenderer(TableCellRenderer renderer, int row, int column) {
46                 Component comp = super.prepareRenderer(renderer, row, column);
47
48                 // Handle selection state
49                 if (isRowSelected(row)) {
50                     comp.setBackground(new Color(70, 130, 180)); // Light blue for selected row
51                 } else {
52                     comp.setBackground(new Color(255, 255, 255));
53                 }
54
55                 reportTable.setFillsViewportHeight(false);
56                 JScrollPane scrollPane = new JScrollPane(reportTable);
57
58                 // Create "Go Back" button
59                 goBackButton = new JButton("Go Back");
60                 goBackButton.addActionListener(new ActionListener() {
61                     @Override
62                     public void actionPerformed(ActionEvent e) {
63                         dispose(); // Close the current frame
64                         SwingUtilities.invokeLater(() -> new SalesReportMain().setVisible(true));
65                     }
66                 });
67
68                 // Use GridBagLayout to center the table and add the "Go Back" button
69                 getContentPane().setLayout(new GridBagLayout());
70                 GridBagConstraints constraints = new GridBagConstraints();
71                 constraints.gridx = 0;
72                 constraints.gridy = 0;
73                 constraints.weightx = 1.0;
74                 constraints.weighty = 1.0;
75                 constraints.fill = GridBagConstraints.BOTH;
76                 constraints.insets = new Insets(10, 10, 10, 10);
77                 getContentPane().add(scrollPane, constraints);
78
79                 constraints.gridx = 1;
80                 constraints.gridy = 1;
81                 constraints.weightx = 0;
82                 constraints.weighty = 0;
83                 constraints.fill = GridBagConstraints.NONE;
84                 constraints.anchor = GridBagConstraints.PAGE_END;
85                 constraints.insets = new Insets(10, 10, 10, 10);
86                 getContentPane().add(goBackButton, constraints);
87
88                 setLocationRelativeTo(null); // Center the frame on the screen
89                 setVisible(true);
90             }
91         };
92
93         private List<String[]> getSalesData(String filePath) {
94             List<String[]> salesData = new ArrayList<>();
95
96             try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
97                 String line;
98                 boolean isFirstLine = true;
99
100                 while ((line = reader.readLine()) != null) {
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
735
736
737
737
738
738
739
739
740
741
742
743
744
745
745
746
747
747
748
748
749
749
750
751
752
753
753
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
```

```

.04      }
.05  }
.06
.07@  private List<String[]> getSalesData(String filePath) {
.08      List<String[]> salesData = new ArrayList<>();
.09
.10     try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
.11         String line;
.12         boolean isFirstLine = true;
.13
.14         while ((line = reader.readLine()) != null) {
.15             if (isFirstLine) {
.16                 isFirstLine = false;
.17                 continue; // Skip the first line (column names)
.18             }
.19
.20             String[] parts = line.split("\t"); // Assuming tab-separated values
.21
.22             if (parts.length >= 6) {
.23                 // Include every column
.24                 String[] saleDetails = Arrays.copyOfRange(parts, 0, 6);
.25                 salesData.add(saleDetails);
.26             }
.27         }
.28     } catch (IOException e) {
.29         e.printStackTrace(); // Handle the exception appropriately
.30     }
.31
.32     return salesData;
.33 }
.34
.35@ public static void main(String[] args) {
.36     SwingUtilities.invokeLater(() -> {
.37         new WorkDoneReport("/Users/aman/Downloads/zip (1)/SalesQuotation.txt");
.38     });
.39 }
.40 }
.41

```

GUI:**DESCRIPTION:**

Swing is used in the `WorkDoneReport` class to provide a modern GUI with a gradient backdrop and a dynamic `JTable` for sales data. Using `SwingUtilities.invokeLater`, thread safety is ensured. The "Go Back" button seamlessly returns to the main sales report, and the parsing algorithm filters and shows sales information. A balanced visual layout is orchestrated by the `GridBagLayout`.

8.2 SALESPERSON - KERTANAA A/P KUMAAR

CODE:

```

package group_Assignment_Java;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JTable;
import javax.swing.JButton;
import javax.swing.table.DefaultTableModel;
import javax.swing.JScrollPane;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.FileReader;
import java.awt.event.ActionEvent;

public class List_All_Personal_Sales_Orders_Page extends JFrame {

    JPanel contentPane;
    private JTable table;
    private static final String sales_quotation_txt_file ="C:\\\\Users\\\\kerta\\\\OneDrive\\\\Documents\\\\OODJ\\\\Salesquotation.txt";
    JButton btn_BacktoPreviousPage;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    List_All_Personal_Sales_Orders_Page frame = new List_All_Personal_Sales_Orders_Page();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public List_All_Personal_Sales_Orders_Page() {

        List_All_Personal_Sales_Orders_Page() {
            setTitle("List All Personal Sales Orders Page");
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            setBounds(100, 100, 756, 485);
            contentPane = new JPanel();
            contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
            setContentPane(contentPane);
            contentPane.setLayout(null);

            JScrollPane scrollPane = new JScrollPane();
            scrollPane.setBounds(96, 78, 549, 253);
            contentPane.add(scrollPane);

            table = new JTable();
            scrollPane.setViewportView(table);
            table.setModel(new DefaultTableModel(
                new Object[][]{
                    {},
                    new String[]{ "Item ID", "Item Name", "Category", "Price", "Order Date", "Status" }
                },
                new Class[] { Integer.class, String.class, String.class, Integer.class, String.class, String.class }
            ));
            public Class getColumnClass(int columnIndex) {
                return columnTypes[columnIndex];
            }
        });

        JButton btnNewButton = new JButton("List all Personal Orders");
        btnNewButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                updateTableFromFile();
            }
        });
        btnNewButton.setBounds(276, 343, 182, 51);
        contentPane.add(btnNewButton);

        JButton btnBacktoPreviousPage = new JButton("Back To Previous Page");
        btnBacktoPreviousPage.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                // Use tab as the delimiter
                String delimiter = "\\t";

                // Split the line into columns using the delimiter
                String[] columns = line.split(delimiter);

                // Add a new row to the table model
                model.addRow(columns);
            }
        });
        catch (Exception ex) {
            ex.printStackTrace();
            // Handle the exception (e.g., show an error message)
        }
    }

}

}

```

GUI:

The figure shows a screenshot of a Windows application window titled "List All Personal Sales Orders Page". The window has a standard title bar with minimize, maximize, and close buttons. Below the title bar is a toolbar with a "Back To Previous Page" button. The main area of the window contains a table with the following data:

Item ID	Item Name	Category	Price	Order Date	Status
1	Ingolf	Bar Furniture	2300	1/1/2024	Unapproved
2	Neiden	Beds	5300	2/1/2024	Unapproved
3	Franklin	Bar Furniture	3500	5/1/2024	Approved
100	Neevra	Beds	6000	1/2/2024	Approved
5	Noverra	Beds	4500	12/2/2024	Approved
123456	Gopal	sofa	99999	12-12-2012	approve
12345678	chair	9632	8888	12-1-2003	approve
12345	table	123	7777	1-1-2012	approve

At the bottom of the window is a button labeled "List all Personal Orders".

The figure above shows the listing of all the orders. It contains of Item ID, Item Name, Category, Price, Order Date and Status.

CODE:

```

package group_Assignment_Java;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.JButton;
import java.awt.Font;
import javax.swing.SwingConstantsConstants;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class Salesperson_Page extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    JPanel contentPane;

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Salesperson_Page frame = new Salesperson_Page();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public Salesperson_Page() {
        setTitle("Salesperson Page");
    }

    public Salesperson_Page() {
        setTitle("Salesperson Page");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 522, 392);
        setLocationRelativeTo(null);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel = new JLabel("What would you like to do?");
        lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);
        lblNewLabel.setFont(new Font("Tw Cen MT", Font.BOLD, 20));
        lblNewLabel.setBounds(48, 35, 420, 30);
        contentPane.add(lblNewLabel);

        JButton btn_Manage_Profile = new JButton("Manage Profile");
        btn_Manage_Profile.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Manage_Personal_Profile_Page profilePage = new Manage_Personal_Profile_Page();
                profilePage.setVisible(true);
            }
        });
        btn_Manage_Profile.setBounds(158, 104, 195, 52);
        contentPane.add(btn_Manage_Profile);

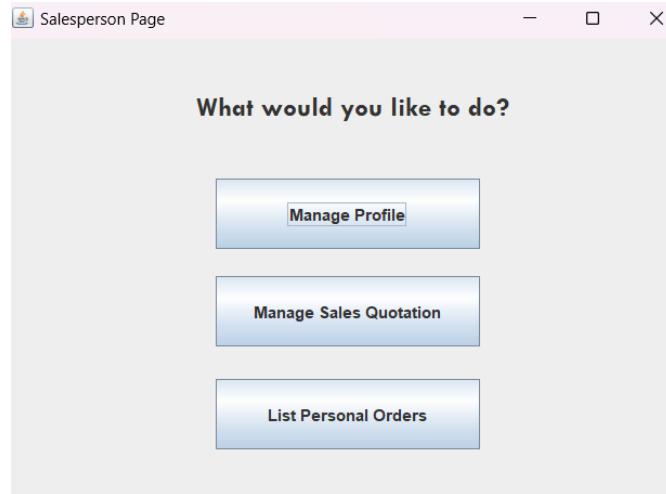
        JButton btn_Manage_Sales_Quotation = new JButton("Manage Sales Quotation");
        btn_Manage_Sales_Quotation.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                Manage_Sales_Order_Quotation_Page salesQuotationPage = new Manage_Sales_Order_Quotation_Page();
                salesQuotationPage.setVisible(true);
            }
        });
        btn_Manage_Sales_Quotation.setBounds(158, 176, 195, 52);
        contentPane.add(btn_Manage_Sales_Quotation);

        JButton btn_List_All_Personal_Orders = new JButton("List Personal Orders");
        btn_List_All_Personal_Orders.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                List_All_Personal_Sales_Orders_Page personalSalesOrdersPage = new List_All_Personal_Sales_Orders_Page();
                personalSalesOrdersPage.setVisible(true);
            }
        });
    }
}

```

```
});  
btn_Manage_Sales_Quotation.setBounds(158, 176, 195, 52);  
contentPane.add(btn_Manage_Sales_Quotation);  
  
JButton btn_List_All_Personal_Orders = new JButton("List Personal Orders");  
btn_List_All_Personal_Orders.addActionListener(new ActionListener() {  
    public void actionPerformed(ActionEvent e) {  
        List_All_Personal_Sales_Orders_Page personalSalesOrdersPage = new List_All_Personal_Sales_Orders_Page();  
        personalSalesOrdersPage.setVisible(true);  
    }  
});  
btn_List_All_Personal_Orders.setBounds(158, 252, 195, 52);  
contentPane.add(btn_List_All_Personal_Orders);  
}  
}
```

GUI:



The figure above shows the main page of a salesperson. The page contains of Manage Profile, Manage Sales Quotation and Listings of Personal Orders.

CODE:

```


import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
public class Manage_Personal_Profile_Page extends JFrame {
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField textField_Name;
    private JTextField textField_Position;
    private JTextField textField_Email;
    private JTextField textField_PhoneNumber;
    private JTextField textField_Address;
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Manage_Personal_Profile_Page frame = new Manage_Personal_Profile_Page();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }
    public Manage_Personal_Profile_Page() {
        setTitle("Manage Personal Profile Page");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 501, 393);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblNewLabel_Name = new JLabel("Name:");
        lblNewLabel_Name.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
        lblNewLabel_Name.setBounds(145, 14, 87, 24);
        contentPane.add(lblNewLabel_Name);

        textField_Name = new JTextField();
        textField_Name.setBounds(233, 18, 117, 20);
        contentPane.add(textField_Name);
        textField_Name.setColumns(10);

        JLabel lblNewLabel_Position = new JLabel("Position:");
        lblNewLabel_Position.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
        lblNewLabel_Position.setBounds(145, 49, 87, 24);
        contentPane.add(lblNewLabel_Position);

        textField_Position = new JTextField();
        textField_Position.setBounds(233, 53, 117, 20);
        contentPane.add(textField_Position);
        textField_Position.setColumns(10);

        JLabel lblNewLabel_Email = new JLabel("Email:");
        lblNewLabel_Email.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
        lblNewLabel_Email.setBounds(145, 84, 87, 24);
        contentPane.add(lblNewLabel_Email);

        textField_Email = new JTextField();
        textField_Email.setBounds(233, 88, 200, 20);
        contentPane.add(textField_Email);
        textField_Email.setColumns(10);

        JLabel lblNewLabel_PhoneNumber = new JLabel("Phone Number:");
        lblNewLabel_PhoneNumber.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
        lblNewLabel_PhoneNumber.setBounds(65, 119, 136, 24);
        contentPane.add(lblNewLabel_PhoneNumber);

        textField_PhoneNumber = new JTextField();
        textField_PhoneNumber.setBounds(233, 123, 117, 20);
        contentPane.add(textField_PhoneNumber);
        textField_PhoneNumber.setColumns(10);
    }
}


```

```

setContentPane(contentPane);
contentPane.setLayout(null);

JLabel lblNewLabel_Name = new JLabel("Name:");
lblNewLabel_Name.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
lblNewLabel_Name.setBounds(145, 14, 87, 24);
contentPane.add(lblNewLabel_Name);

textField_Name = new JTextField();
textField_Name.setBounds(233, 18, 117, 20);
contentPane.add(textField_Name);
textField_Name.setColumns(10);

JLabel lblNewLabel_Position = new JLabel("Position:");
lblNewLabel_Position.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
lblNewLabel_Position.setBounds(145, 49, 87, 24);
contentPane.add(lblNewLabel_Position);

textField_Position = new JTextField();
textField_Position.setBounds(233, 53, 117, 20);
contentPane.add(textField_Position);
textField_Position.setColumns(10);

JLabel lblNewLabel_Email = new JLabel("Email:");
lblNewLabel_Email.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
lblNewLabel_Email.setBounds(145, 84, 87, 24);
contentPane.add(lblNewLabel_Email);

textField_Email = new JTextField();
textField_Email.setBounds(233, 88, 200, 20);
contentPane.add(textField_Email);
textField_Email.setColumns(10);

JLabel lblNewLabel_PhoneNumber = new JLabel("Phone Number:");
lblNewLabel_PhoneNumber.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
lblNewLabel_PhoneNumber.setBounds(85, 119, 136, 24);
contentPane.add(lblNewLabel_PhoneNumber);

textField_PhoneNumber = new JTextField();
textField_PhoneNumber.setBounds(233, 123, 117, 20);
contentPane.add(textField_PhoneNumber);
textField_PhoneNumber.setColumns(10);

JLabel lblNewLabel_Address = new JLabel("Address:");
lblNewLabel_Address.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
lblNewLabel_Address.setBounds(145, 154, 87, 24);
contentPane.add(lblNewLabel_Address);

textField_Address = new JTextField();
textField_Address.setBounds(233, 158, 200, 20);
contentPane.add(textField_Address);
textField_Address.setColumns(10);

// Save Button
 JButton btnSave = new JButton("Save");
btnSave.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveProfile();
        // Add code to display success message
        JLabel successLabel = new JLabel("Profile saved successfully!");
        successLabel.setFont(new Font("TIMES NEW ROMAN", Font.BOLD, 14));
        successLabel.setBounds(180, 250, 200, 30);
        contentPane.add(successLabel);
        contentPane.revalidate();
        contentPane.repaint();
    }
});
btnSave.setBounds(180, 200, 100, 30);
contentPane.add(btnSave);
}

// Method to save profile data
private void saveProfile() {
    String name = textField_Name.getText();
    String position = textField_Position.getText();
    String email = textField_Email.getText();
    String phoneNumber = textField_PhoneNumber.getText();
    String address = textField_Address.getText();

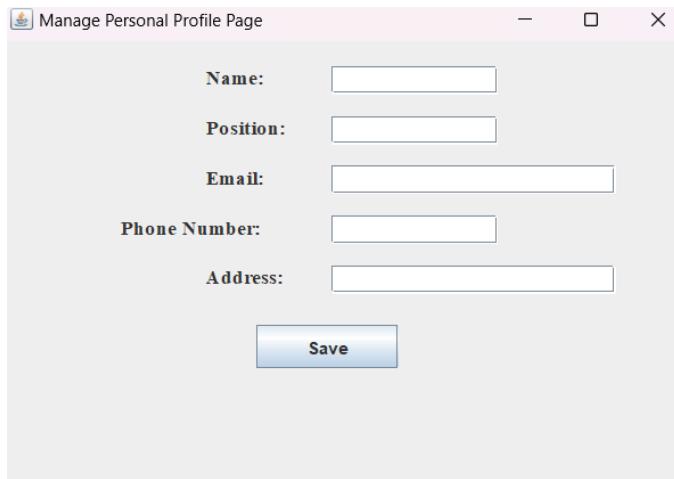
    // Write profile data to a text file
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("profile.txt"))) {
        writer.write("Name: " + name);
        writer.newLine();
        writer.write("Position: " + position);
        writer.newLine();
    }
}

```

```
// Method to save profile data
private void saveProfile() {
    String name = textField_Name.getText();
    String position = textField_Position.getText();
    String email = textField_Email.getText();
    String phoneNumber = textField_PhoneNumber.getText();
    String address = textField_Address.getText();

    // Write profile data to a text file
    try (BufferedWriter writer = new BufferedWriter(new FileWriter("profile.txt"))) {
        writer.write("Name: " + name);
        writer.newLine();
        writer.write("Position: " + position);
        writer.newLine();
        writer.write("Email: " + email);
        writer.newLine();
        writer.write("Phone Number: " + phoneNumber);
        writer.newLine();
        writer.write("Address: " + address);
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}
```

GUI:



The figure above shows the Personal Profile page. The page includes Name, Position, Email, Phone Number and Address.

CODE:

```

package group_Assignment_Java;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
import javax.swing.JScrollPane;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import java.awt.Font;
import javax.swing.JTextField;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.HashSet;
import java.util.Set;
import java.awt.event.ActionEvent;

public class Manage_Sales_Order_Quotation_Page extends JFrame {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTable table;
    private JTextField textField_ItemID;
    private JTextField textField_ItemName;
    private JTextField textField_Category;
    private JTextField textField_OrderDate;
    private JTextField textField_Status;
    private JTextField textField_Price;
    private static final String sales_quotation_txt_file = "C:\\\\Users\\\\kerta\\\\OneDrive\\\\Documents\\\\OODJ\\\\SalesQuotation1.txt";

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {
                    Manage_Sales_Order_Quotation_Page frame = new Manage_Sales_Order_Quotation_Page();
                    frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

    /**
     * Create the frame.
     */
    public Manage_Sales_Order_Quotation_Page() {
        setTitle("Manage Sales Order Quotation Page");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(100, 100, 907, 548);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        JScrollPane scrollPane = new JScrollPane();
        scrollPane.setBounds(38, 11, 818, 294);
        contentPane.add(scrollPane);

        table = new JTable();
        scrollPane.setViewportView(table);
        loadDataFromFile();

        table.setModel(new DefaultTableModel(
            new Object[][] {
                {},
                new String[] {
                    "Item ID", "Item Name", "Category", "Price", "Order Date", "Status"
                }
            }
        );
    }
}

```

```

) {
    /**
     */
    private static final long serialVersionUID = 1L;
    Class[] columnTypes = new Class[] {
        Integer.class, String.class, String.class, Integer.class, String.class, String.class
    };
    public Class getColumnClass(int columnIndex) {
        return columnTypes[columnIndex];
    }
};

JLabel lblNewLabel = new JLabel("Item ID:");
lblNewLabel.setFont(new Font("Tw Cen MT", Font.BOLD, 17));
lblNewLabel.setBounds(26, 337, 108, 14);
contentPane.add(lblNewLabel);

JLabel lblItemName = new JLabel("Item Name:");
lblItemName.setFont(new Font("Tw Cen MT", Font.BOLD, 17));
lblItemName.setBounds(26, 374, 108, 24);
contentPane.add(lblItemName);

JLabel lblCategory = new JLabel("Category:");
lblCategory.setFont(new Font("Tw Cen MT", Font.BOLD, 17));
lblCategory.setBounds(26, 412, 108, 17);
contentPane.add(lblCategory);

JLabel lblPrice = new JLabel("Price:");
lblPrice.setFont(new Font("Tw Cen MT", Font.BOLD, 17));
lblPrice.setBounds(295, 337, 108, 14);
contentPane.add(lblPrice);

JLabel lblOrderDate = new JLabel("Order Date:");
lblOrderDate.setFont(new Font("Tw Cen MT", Font.BOLD, 17));
lblOrderDate.setBounds(295, 377, 108, 19);
contentPane.add(lblOrderDate);

JLabel lblNewLabel_1 = new JLabel("Status:");
lblNewLabel_1.setFont(new Font("Tw Cen MT", Font.BOLD, 17));
lblNewLabel_1.setBounds(295, 415, 108, 14);
contentPane.add(lblNewLabel_1);

textField_ItemID = new JTextField();
textField_ItemID.setBounds(132, 336, 125, 20);
contentPane.add(textField_ItemID);
textField_ItemID.setColumns(10);

textField_ItemName = new JTextField();
textField_ItemName.setColumns(10);
textField_ItemName.setBounds(132, 378, 125, 20);
contentPane.add(textField_ItemName);

textField_Category = new JTextField();
textField_Category.setColumns(10);
textField_Category.setBounds(132, 412, 125, 20);
contentPane.add(textField_Category);

textField_OrderDate = new JTextField();
textField_OrderDate.setColumns(10);
textField_OrderDate.setBounds(389, 378, 125, 20);
contentPane.add(textField_OrderDate);

textField_Status = new JTextField();
textField_Status.setColumns(10);
textField_Status.setBounds(389, 414, 125, 20);
contentPane.add(textField_Status);

textField_Price = new JTextField();
textField_Price.setColumns(10);
textField_Price.setBounds(389, 336, 125, 20);
contentPane.add(textField_Price);

```

```

JButton btn_Add = new JButton("Add");
btn_Add.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if(validateFields()) {
            int itemID = Integer.parseInt(textField_ItemID.getText());
            String itemName = textField_ItemName.getText();
            String category = textField_Category.getText();
            int price = Integer.parseInt(textField_Price.getText());
            String orderDate = textField_OrderDate.getText();
            String status = textField_Status.getText();

            DefaultTableModel model = (DefaultTableModel) table.getModel();
            model.addRow(new Object[]{itemID, itemName, category, price, orderDate, status});
            textField_ItemID.setText("");
            textField_ItemName.setText("");
            textField_Category.setText("");
            textField_Price.setText("");
            textField_OrderDate.setText("");
            textField_Status.setText("");
        }
        else {
            JOptionPane.showMessageDialog(null, "Please fill in all fields", "Validation Error", JOptionPane.ERROR_MESSAGE);
        }
    }
});
btn_Add.setBounds(600, 328, 100, 37);
contentPane.add(btn_Add);

JButton btn_Delete = new JButton("Delete");
btn_Delete.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Get the selected row index
        int selectedRowIndex = table.getSelectedRow();

        // Check if a row is selected
        if (selectedRowIndex != -1) {
            // Get the model of the table
            DefaultTableModel model = (DefaultTableModel) table.getModel();

            // Remove the selected row
            model.removeRow(selectedRowIndex);
        }
    }
});
btn_Delete.setBounds(731, 328, 100, 37);
contentPane.add(btn_Delete);

JButton btn_Update = new JButton("Update");
btn_Update.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        // Get the selected row index
        int selectedRowIndex = table.getSelectedRow();

        // Check if a row is selected
        if (selectedRowIndex != -1) {
            // Get the model of the table
            DefaultTableModel model = (DefaultTableModel) table.getModel();

            // Update the selected row with new values
            for (int columnIndex = 0; columnIndex < model.getColumnCount(); columnIndex++) {
                String newValue = getTextFieldValue(columnIndex);
                if (!newValue.isEmpty()) {
                    model.setValueAt(newValue, selectedRowIndex, columnIndex);
                }
            }

            // Clear the text fields after the update
            clearTextFields();
        }
    }
});

```

```

// Helper method to get the value from the corresponding text field
private String getTextFieldValue(int columnIndex) {
    switch (columnIndex) {
        case 0:
            return textField_ItemID.getText();
        case 1:
            return textField_ItemName.getText();
        case 2:
            return textField_Category.getText();
        case 3:
            return textField_Price.getText();
        case 4:
            return textField_OrderDate.getText();
        case 5:
            return textField_Status.getText();
        default:
            return ""; // Handle additional columns if needed
    }
}

// Helper method to clear all text fields
private void clearTextFields() {
    textField_ItemID.setText("");
    textField_ItemName.setText("");
    textField_Category.setText("");
    textField_Price.setText("");
    textField_OrderDate.setText("");
    textField_Status.setText("");
}
});

btn_Update.setBounds(600, 404, 100, 37);
contentPane.add(btn_Update);

JButton btn_Save = new JButton("Save");
btn_Save.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        saveDataToFile();
    }
});

btn_Save.setBounds(731, 404, 100, 37);
contentPane.add(btn_Save);

JButton btn_BacktoPreviousPage = new JButton("Back To Previous Page");
btn_BacktoPreviousPage.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Salesperson_Page salespersonPage = new Salesperson_Page();
        salespersonPage.setVisible(true);
        dispose();
    }
});
btn_BacktoPreviousPage.setBounds(27, 455, 182, 37);
contentPane.add(btn_BacktoPreviousPage);
}

private void saveDataToFile() {
    try {
        DefaultTableModel model = (DefaultTableModel) table.getModel();
        int rowCount = model.getRowCount();

        BufferedWriter writer = new BufferedWriter(new FileWriter(sales_quotation_txt_file, true));

        // Write column headers
        if (salesQuotationFileIsEmpty()) {
            for (int i = 0; i < model.getColumnCount(); i++) {
                writer.write(model.getColumnName(i));
                if (i < model.getColumnCount() - 1) {
                    writer.write("\t");
                } else {
                    writer.write("\n");
                }
            }
        }
    }
}

```

```

// Write data
if (rowCount > 0) {
    int lastRow = rowCount - 1;
    String lastRowString = new StringBuilder();
    for (int col = 0; col < model.getColumnCount(); col++) {
        lastRowString.append(model.getValueAt(lastRow, col).toString());
        if (col < model.getColumnCount() - 1) {
            lastRowString.append("\t");
        } else {
            lastRowString.append("\n");
        }
    }
}

Set<String> existingRows = readExistingRows();
if (!existingRows.contains(lastRowString.toString())) {
    // Write data only if it doesn't already exist
    writer.write(lastRowString.toString());
}
}

writer.close();
 JOptionPane.showMessageDialog(null, "Data successfully added to " + txt_file);
} catch (IOException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null, "Error saving data to file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}
}

private void loadDataFromFile() {
try (BufferedReader reader = new BufferedReader(new FileReader(sales_quotation_txt_file))) {
    DefaultTableModel model = (DefaultTableModel) table.getModel();
    String line;

    // Read column headers
    if ((line = reader.readLine()) != null) {
        String[] columnHeaders = line.split("\t");
        model.setColumnIdentifiers(columnHeaders);
    }

    // Read data
    while ((line = reader.readLine()) != null) {
        String[] rowData = line.split("\t");
        model.addRow(rowData);
    }
}

catch (IOException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null, "Error loading data from file: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
}
}

private boolean validateFields() {
    // Validate that all fields are not empty
    return !textField_itemID.getText().isEmpty()
        && !textField_itemname.getText().isEmpty()
        && !textField_qty.getText().isEmpty()
        && !textField_Price.getText().isEmpty()
        && !textField_OrderDate.getText().isEmpty()
        && !textField_Status.getText().isEmpty();
}

private boolean salesQuotationFileIsEmpty() throws IOException {
    BufferedReader reader = new BufferedReader(new FileReader(sales_quotation_txt_file));
    boolean isEmpty = reader.readLine() == null; // Check if the first line is null, indicating an empty file
    reader.close();
    return isEmpty;
}

private Set<String> readExistingRows() throws IOException {
    Set<String> existingRows = new HashSet<>();
    BufferedReader reader = new BufferedReader(new FileReader(sales_quotation_txt_file));
    String line;
    while ((line = reader.readLine()) != null) {
        existingRows.add(line);
    }
    reader.close();
    return existingRows;
}
}

```

GUI:

The figure above shows the page to manage sales order quotation. The page above shows the field to put in the details.

8.3 OFFICER – AREEBAH IRFAN MOHAMMED

CODE:

Officer class:

```

21* public String getUsername() {
22     return username;
23 }
24 |
25 // Method to manage personal profile
26* public void manageProfile(String username, String password) {
27     this.username = username;
28     this.password = password;
29     System.out.println("Profile updated successfully!");
30 }
31
32 // Method to process sale order upon sale approval/ closed sale
33* public void processSaleOrder(SaleOrder saleOrder) {
34     if (!saleOrders.containsKey(saleOrder.getOrderId())) {
35         saleOrders.put(saleOrder.getOrderId(), saleOrder);
36         System.out.println("Sale order processed successfully!");
37     } else {
38         System.out.println("Sale order already processed!");
39     }
40 }
41
42 // Method to search sale orders
43* public SaleOrder searchSaleOrder(String orderId) {
44     return saleOrders.get(orderId);
45 }

```

```

1 package group_Assignment_Java;
2
3 import java.util.HashMap;
4 import java.util.List;
5 import java.util.Map;
6
7 //Officer class representing a user with access to system functionalities
8 @SuppressWarnings({ "unused" })
9 class Officer {
10     private String username;
11     @SuppressWarnings("unused")
12     private String password;
13     private Map<String, SaleOrder> saleOrders;
14
15* public Officer(String username, String password) {
16     this.username = username;
17     this.password = password;
18     this.saleOrders = new HashMap<>();
19 }
20
21* public String getUsername() {
22     return username;
23 }
24
25 // Method to manage personal profile
26* public void manageProfile(String username, String password) {
27     this.username = username;
28     this.password = password;

```

```

47 // Method to create a new sale order
48* public void createSaleOrder(SaleOrder saleOrder) {
49     saleOrders.put(saleOrder.getOrderId(), saleOrder);
50     System.out.println("Sale order created successfully!");
51 }
52
53 // Method to modify a sale order
54* public void modifySaleOrder(SaleOrder saleOrder, List<Product> newProducts) {
55     saleOrder.getProducts1().clear();
56     List<Product> products = (List<Product>) saleOrder.getProducts();
57     List<Product> products2 = products;
58     List<Product> products1 = products2;
59     List<Product> list = products1;
60     boolean all = list.addAll(newProducts);
61     System.out.println("Sale order modified successfully!");
62 }
63
64 // Method to remove a sale order
65* public void removeSaleOrder(SaleOrder saleOrder) {
66     saleOrders.remove(saleOrder.getOrderId());
67     System.out.println("Sale order removed successfully!");
68 }

```

```

70 // Method to submit sales for production and issue sale invoice
71 public void submitForProduction(SaleOrder saleOrder) {
72     saleOrder.approveSale();
73     saleOrder.closeSale();
74     System.out.println("Sale submitted for production and invoice issued successfully!");
75 }
76
77 // Method to check sale product status
78 public String checkProductStatus(SaleOrder saleOrder) {
79     // Logic to check product status
80     return "Product status: In Progress";
81 }
82
83 // Method to generate reports
84 public Report generateReport(String reportType) {
85     String reportContent = "";
86     // Logic to generate report content based on report type
87     switch (reportType) {
88         case "workDone":
89             reportContent = "Work done report content...";
90             break;
91         case "approvedClosedSale":
92             reportContent = "Approved/closed sale report content...";
93             break;
94         default:
95             reportContent = "Invalid report type!";
96     }
97     return new Report(reportType, reportContent);

```

The 'Officer' class in the provided code serves as a representation of a user within a system, specifically focusing on their role in managing sale orders and generating reports. The class uses encapsulation to store and manipulate relevant data, such as the officer's username and password, as private instance variables. The 'saleOrders' variable, implemented as a 'Map', allows efficient storage and retrieval of sale orders based on their unique order IDs. The class provides several methods to interact with sale orders. The 'processSaleOrder' method adds a sale order to the 'saleOrders' map if it doesn't already exist, and the 'createSaleOrder' method adds a new sale order to the map. Conversely, the 'removeSaleOrder' method removes a sale order from the map. The 'modifySaleOrder' method enables the modification of a sale order by replacing its existing list of products with a new list. Other methods focus on different aspects of sale order management. The 'searchSaleOrder' method allows searching for a specific sale order based on its order ID. The 'submitForProduction' method initiates the process of approving and closing a sale order, while the 'checkProductStatus' method is intended to provide information about the status of the products within a sale order. Additionally, the 'Officer' class includes functionality to manage personal profiles. The 'manageProfile' method allows an officer to update their username and password. Lastly, the 'generateReport' method generates reports based on a specified report type. The method uses a 'switch' statement to determine the content of the report based on the provided type. Currently, two report types, "workDone" and "approvedClosedSale," are supported, each with its corresponding report content. If an invalid report type is provided, a default message is assigned.

Product class:

```
1 package group_Assignment_Java;
2
3
4 //Product class representing furniture items
5 public class Product {
6     private int id;
7     private String name;
8     private String category;
9     private double price;
10
11    public Product(int id, String name, String category, double price) {
12        this.id = id;
13        this.name = name;
14        this.category = category;
15        this.price = price;
16    }
17
18    public int getId() {
19        return id;
20    }
21
22    public String getName() {
23        return name;
24    }
25
26    public String getCategory() {
27        return category;
28    }
29
30    public double getPrice() {
31        return price;
32    }
33}
34
```

The provided code defines a 'Product' class representing furniture items. This class encapsulates the properties of a product, including its ID, name, category, and price. The class has a constructor that takes these properties as parameters and assigns them to the corresponding instance variables. Additionally, the class provides getter methods for each property, allowing other classes to retrieve the values of the product's ID, name, category, and price. Overall, the 'Product' class serves as a blueprint for creating instances of furniture products and accessing their information when needed.

Invoice class:

```

1 package group_Assignment_Java;
2
3 //Invoice class representing an invoice for a sale order
4 class Invoice {
5     private String invoiceID;
6     private SaleOrder saleOrder;
7
8     public Invoice(String invoiceID, SaleOrder saleOrder) {
9         this.invoiceID = invoiceID;
10        this.saleOrder = saleOrder;
11    }
12
13    public String getInvoiceID() {
14        return invoiceID;
15    }
16
17    public SaleOrder getSaleOrder() {
18        return saleOrder;
19    }
20}
21

```

The provided code introduces the 'Invoice' class, representing an invoice for a sale order. This class encapsulates the properties of an invoice, including the invoice ID and the corresponding sale order. The 'Invoice' class has a constructor that takes the invoice ID and the associated sale order as parameters and initializes the respective instance variables. It also provides getter methods to retrieve the invoice ID and the sale order. The 'Invoice' class serves as a blueprint for creating invoice objects that store information about a specific sale order, enabling easy access to the invoice details when needed.

Report class:

```

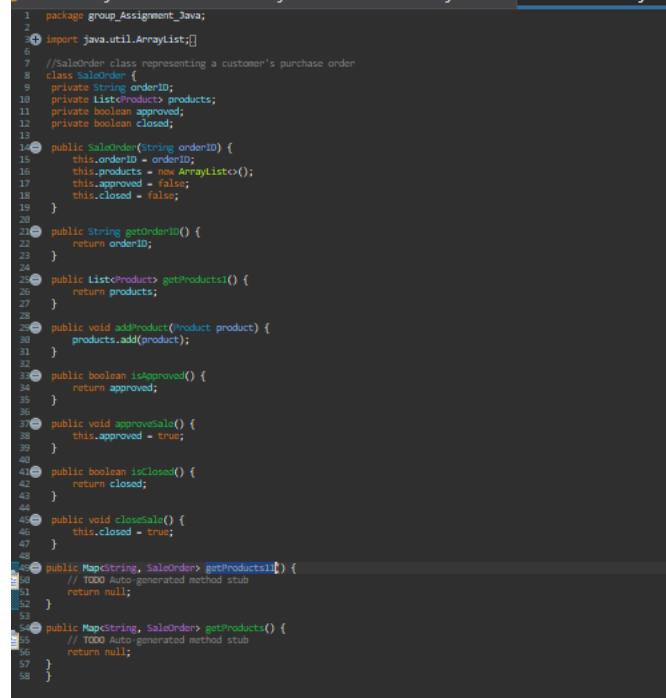
1 package group_Assignment_Java;
2
3 //Report class representing various types of reports
4 class Report {
5     private String type;
6     private String content;
7
8     public Report(String type, String content) {
9         this.type = type;
10        this.content = content;
11    }
12
13    public String getType() {
14        return type;
15    }
16
17    public String getContent() {
18        return content;
19    }
20}
21

```

The provided code introduces the 'Report' class, which represents various types of reports. This class encapsulates two properties: the type of the report and its corresponding content. The 'Report' class includes a constructor that takes the report type and content as parameters and initializes the respective instance variables. It also provides getter methods to retrieve the

report type and content. The 'Report' class serves as a blueprint for creating report objects with specific types and content, allowing easy access to these details when generating or working with reports in a system.

SaleOrder class:

A screenshot of a Java code editor showing the SaleOrder.java file. The code defines a SaleOrder class with properties for order ID, products, approval status, and closure status. It includes methods for getting and setting these properties, adding products, and changing approval and closure status. There are also two auto-generated methods for getProducts() and getProducts().

```
1 package group_Assignment_Java;
2
3+ import java.util.ArrayList;[]
4
5 //SaleOrder class representing a customer's purchase order
6 class SaleOrder {
7     private String orderId;
8     private List<Product> products;
9     private boolean approved;
10    private boolean closed;
11
12    public SaleOrder(String orderId) {
13        this.orderId = orderId;
14        this.products = new ArrayList<>();
15        this.approved = false;
16        this.closed = false;
17    }
18
19    public String getOrderId() {
20        return orderId;
21    }
22
23    public List<Product> getProducts() {
24        return products;
25    }
26
27    public void addProduct(Product product) {
28        products.add(product);
29    }
30
31    public boolean isApproved() {
32        return approved;
33    }
34
35    public void approveSale() {
36        this.approved = true;
37    }
38
39    public boolean isClosed() {
40        return closed;
41    }
42
43    public void closeSale() {
44        this.closed = true;
45    }
46
47    public Map<String, SaleOrder> getProducts() {
48        // TODO Auto-generated method stub
49        return null;
50    }
51
52    public Map<String, SaleOrder> getProducts() {
53        // TODO Auto-generated method stub
54        return null;
55    }
56
57 }
```

The provided code presents the 'SaleOrder' class, representing a customer's purchase order. This class encapsulates the properties of a sale order, including the order ID, a list of products, and the status of approval and closure. The 'SaleOrder' class features a constructor that takes the order ID as a parameter and initializes the instance variables. It also provides getter methods to retrieve the order ID, the list of products, and the status of approval and closure. Additionally, there are methods to add a product to the order, approve the sale, and close the sale. The 'SaleOrder' class serves as a blueprint for creating sale order objects, allowing easy access to the order details and enabling operations such as adding products, approving the sale, and closing the sale within a system.

YoyoFurnitureGUI1 class:

```

1 package group_Assignment_Java;
2
3 import java.awt.event.ActionEvent;
4 import java.awt.event.ActionListener;
5
6 import javax.swing.JFrame;
7
8 // GUI class for Yoyo Furniture System
9 public class YoyoFurnitureGUI1 extends JFrame {
10
11     /**
12      *
13      */
14     private static final long serialVersionUID = 1L;
15     // GUI components
16     private JButton manageProfileButton;
17     private JButton processSaleOrderButton;
18     private JButton searchSaleOrderButton;
19     private JButton createSaleOrderButton;
20     private JButton modifySaleOrderButton;
21     private JButton removeSaleOrderButton;
22     private JButton submitForProductionButton;
23     private JButton checkProductStatusButton;
24     private JButton generateReportButton;
25     private YoyoFurnitureGUI1 frame;
26
27     public void YoyoFurnitureGUI1() {
28         // Set up the frame
29         setTitle("Yoyo Furniture System");
30         setSize(400, 300);
31         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
32         setLayout(new GridLayout(3, 3));
33
34         // Initialize GUI components
35         manageProfileButton = new JButton();
36         processSaleOrderButton = new JButton();
37         searchSaleOrderButton = new JButton();
38         createSaleOrderButton = new JButton();
39         modifySaleOrderButton = new JButton();
40         removeSaleOrderButton = new JButton();
41         submitForProductionButton = new JButton();
42         checkProductStatusButton = new JButton();
43         generateReportButton = new JButton();
44

```

The provided code introduces the `YoyoFurnitureGUI1` class, which represents a graphical user interface (GUI) for the Yoyo Furniture System. This class extends the `JFrame` class from the Swing library, enabling the creation of a window-based interface. The GUI consists of several buttons, each representing a specific functionality within the furniture system. These buttons include options such as managing profiles, processing sale orders, searching sale orders, creating sale orders, modifying sale orders, removing sale orders, submitting orders for production, checking product status, and generating reports. Each button is associated with an action listener, implemented as an anonymous inner class. When a button is clicked, the corresponding action is triggered, and the associated code block is executed. This allows for the customization of functionality and the implementation of specific logic for each button. The GUI components are added to the frame using the `add()` method,

8.4 AUTHORIZATION AND AUTHENTICATION – HOUSHIANEE HARHANGI

Administrator_Account

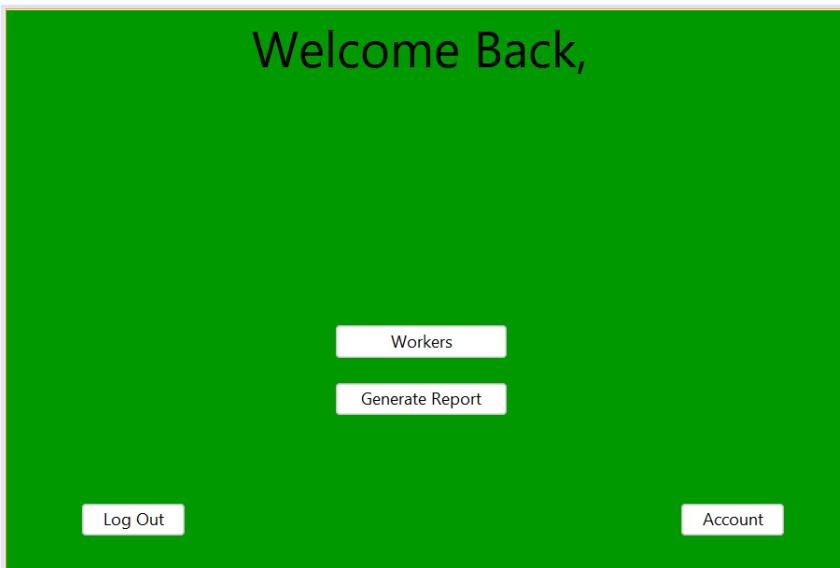
```

5 package javaapplication2;
6
7 /**
8 * @author Cody
9 */
10
11 public class AdministratorPage extends javax.swing.JFrame {
12     String username;
13
14     /**
15      * Creates new form Trainer
16     */
17     public AdministratorPage() {
18         initComponents();
19     }
20
21     AdministratorPage(String text){
22         this();
23         username = text;
24         jlabel1.setText("Welcome Back, " + username);
25     }
26     @SuppressWarnings("unchecked")
27     Generated Code
28
29     private void logOutActionPerformed(java.awt.event.ActionEvent evt) {
30         Login_done l = new Login_done(2);
31         l.show();
32         dispose();
33     }
34
35     private void editClassActionPerformed(java.awt.event.ActionEvent evt) {
36         Administrator_Workers ec = new Administrator_Workers(username);
37         ec.show();
38         dispose();
39     }
40
41     private void generateReportActionPerformed(java.awt.event.ActionEvent evt) {
42
43 javaapplication2.AdministratorPage >
```

```

32
33     private void generateReportActionPerformed(java.awt.event.ActionEvent evt) {
34         Administrator_GenerateReport ogr = new Administrator_GenerateReport(username);
35         ogr.show();
36         dispose();
37     }
38
39     private void accountActionPerformed(java.awt.event.ActionEvent evt) {
40         Administrator_Account oa = new Administrator_Account(username);
41         oa.show();
42         dispose();
43     }
44
45     /**
46      * @param args the command line arguments
47     */
48     public static void main(String args[]) {
49         /* Set the Nimbus look and feel */
50         // Look and feel setting code (optional)
51         //<<editor-fold>
52         //<<editor-fold>
53         //<<editor-fold>
54         //<<editor-fold>
55         //<<editor-fold>
56         //<<editor-fold>
57         //<<editor-fold>
58         //<<editor-fold>
59         //<<editor-fold>
60         //<<editor-fold>
61         //<<editor-fold>
62         //<<editor-fold>
63         //<<editor-fold>
64         //<<editor-fold>
65         //<<editor-fold>
66
67         /* Create and display the form */
68         java.awt.EventQueue.invokeLater(new Runnable() {
69             public void run() {
70                 new AdministratorPage().setVisible(true);
71             }
72         });
73     }
74 Javaapplication2.AdministratorPage >
```

```
191 [     |     |     |
192 |     |     });
193 }
194
195 // Variables declaration - do not modify
196 private javax.swing.JButton account;
197 private javax.swing.JButton editClass;
198 private javax.swing.JButton generateReport;
199 private javax.swing.JLabel jLabel1;
200 private javax.swing.JPanel jPanel1;
201 private javax.swing.JButton logout;
202 // End of variables declaration
203 }
204 }
```

GUI:

It appears that the class constructor receives the username variable, which enables the jLabel1 component to render a customised welcome message. Following "Welcome Back" is the alias on this label.

When the Workers button (editClass) is clicked, the editClassActionPerformed method is called, which closes the graphical user interface and loads the Administrator_Workers page with the user's username.

The generateReport action is initiated when the generateReport button (generateReport) is clicked. This action terminates the current window and launches Administrator_GenerateReport with the specified username in order to generate the report.

To manage the administrator's account settings, this button invokes accountActionPerformed to open the Administrator_Account window and close the present one.

The action listener for the logout icon (logOut) terminates the current window and initiates the Login_done process to log the user out.

This method launches the application, configures the graphical user interface, and renders the AdministratorPage.

Landing_done

CODE;

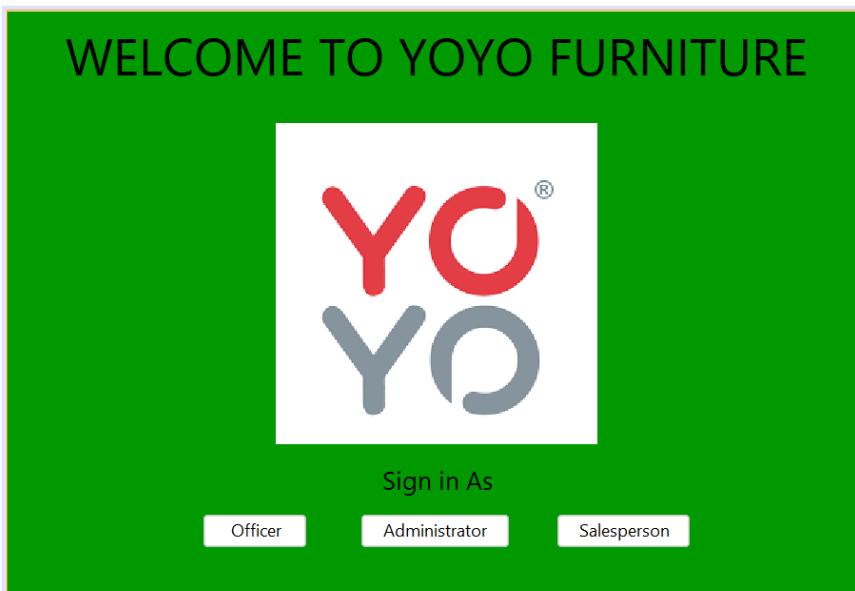
```
5  package javaapplication2;
6
7  /**
8   * 
9   * @author Cody
10  */
11 public class Landing_done extends javax.swing.JFrame {
12
13 /**
14  * Creates new form Login
15  */
16 public Landing_done() {
17     initComponents();
18 }
19 void signIn(int i){
20     Login_done l = new Login_done(i);
21     l.show();
22     dispose();
23 }
24 /**
25  * This method is called from within the constructor to initialize the form.
26  * WARNING: Do NOT modify this code. The content of this method is always
27  * regenerated by the Form Editor.
28  */
29 @SuppressWarnings("unchecked")
30 Generated Code
31
32 /**
33  * officerSignInActionPerformed(java.awt.event.ActionEvent evt)
34  * 
35  */
36 private void officerSignInActionPerformed(java.awt.event.ActionEvent evt) {
37     signIn(1);
38 }
39
40 /**
41  * administratorSignInActionPerformed(java.awt.event.ActionEvent evt)
42  * 
43  */
44 private void administratorSignInActionPerformed(java.awt.event.ActionEvent evt) {
45     signIn(2);
46 }
47
48 /**
49  * salespersonSignInActionPerformed(java.awt.event.ActionEvent evt)
50  * 
51  */
52 private void salespersonSignInActionPerformed(java.awt.event.ActionEvent evt) {
53     signIn(3);
54 }
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
978
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163

```

```

128     signIn(2);
129 }
130 
131     private void salespersonSignInActionPerformed(java.awt.event.ActionEvent evt) {
132         signIn(3);
133     }
134 
135     /**
136      * @param args the command line arguments
137      */
138     public static void main(String args[]) {
139         /* Set the Nimbus look and feel */
140         // Look and feel setting code (optional)
141         //</editor-fold>
142         //</editor-fold>
143         //</editor-fold>
144 
145         /* Create and display the form */
146         java.awt.EventQueue.invokeLater(new Runnable() {
147             public void run() {
148                 new Landing_done().setVisible(true);
149             }
150         });
151     }
152 
153     // Variables declaration - do not modify
154     private javax.swing.JButton administratorSignIn;
155     private javax.swing.JLabel jLabel1;
156     private javax.swing.JLabel jLabel2;
157     private javax.swing.JLabel jLabel3;
158     private javax.swing.JPanel jPanel1;
159     private javax.swing.JButton officerSignIn;
160     private javax.swing.JButton salespersonSignIn;
161     // End of variables declaration
162 }
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183

```

GUI:

GUI windows, including `Landing_done`, extend `JFrame`. Without displaying their contents, the `initComponents()` method initialises components. It appears that the `signIn()` method generates a new `Login_done` window populated with an integer value that corresponds to the user role (e.g., officer, administrator, merchant).

The `officerSalesperson`, `administratorSignInActionPerformed`, and `SignInActionPerformedDifferent` integers are passed to `signIn(int)` via the `SignInActionPerformed` method in order to denote unique user categories.

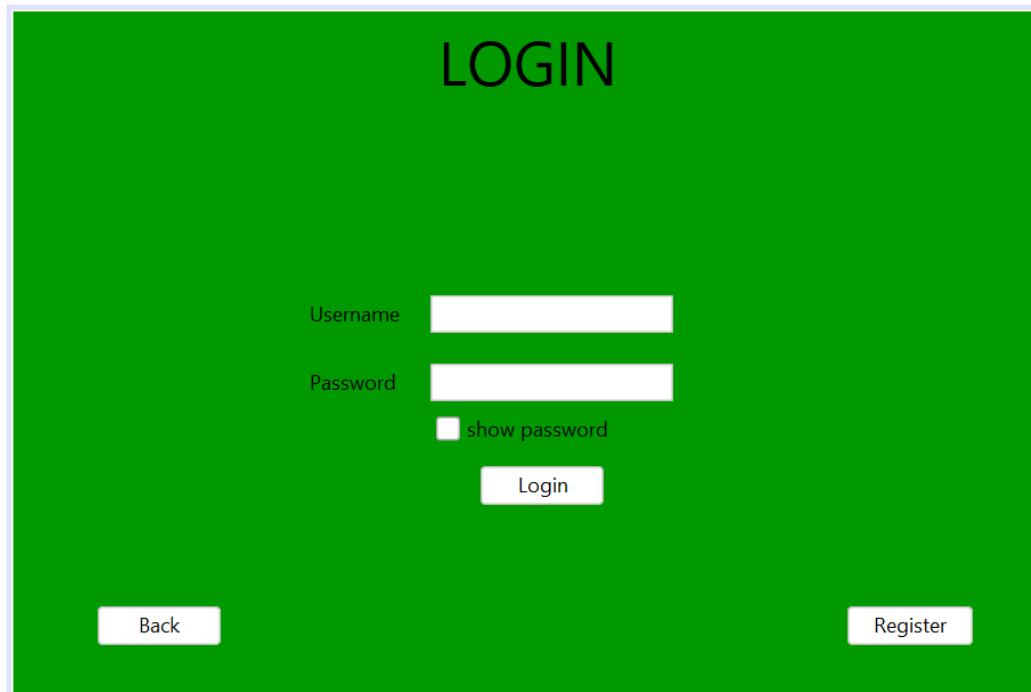
Main establishes the aesthetic and functional characteristics of the application and launches Landing_done. Using this technique, the GUI application is launched.

Login_done

CODE:

```
196
197     if (fileUsername.equals(username.getText()) && filePassword.equals(password.getText())) {
198         JOptionPane.showMessageDialog(null, "Successfully Log In");
199         switch (condition){
200             case 1:
201                 OfficerPage op = new OfficerPage(username.getText());
202                 op.show();
203                 dispose();
204                 status = false;
205                 break;
206             case 2:
207                 AdministratorPage ap = new AdministratorPage(username.getText());
208                 ap.show();
209                 dispose();
210                 status = false;
211                 break;
212             case 3:
213                 SalespersonPage sp = new SalespersonPage(username.getText());
214                 sp.show();
215                 dispose();
216                 status = false;
217                 break;
218         }
219     }
220     if(status){
221         JOptionPane.showMessageDialog(null, "Invalid Credentials, please try again");
222     }
223 } catch (FileNotFoundException ex) {
224     Logger.getLogger(Register.class.getName()).log(Level.SEVERE, null, ex);
225 } catch (IOException ex) {
226     Logger.getLogger(Register.class.getName()).log(Level.SEVERE, null, ex);
227 }
228 }
229
230 }
231
232 private void backActionPerformed(java.awt.event.ActionEvent evt) {
233     Landing_done landing = new Landing_done();
234     landing.show();
235     dispose();
236 }
237
238
239 private void registerActionPerformed(java.awt.event.ActionEvent evt) {
240     Register register = new Register();
241     register.show();
242     dispose();
243 }
244
245 /**
246 * @param args the command line arguments
247 */
248 public static void main(String args[]) {
249     /* Set the Nimbus look and feel */
250     Look and feel setting code (optional)
251     //</editor-fold>
252     //</editor-fold>
253     //</editor-fold>
254     //</editor-fold>
255     //</editor-fold>
256     //</editor-fold>
257     //</editor-fold>
258     //</editor-fold>
259     //</editor-fold>
260     //</editor-fold>
261     //</editor-fold>
262     //</editor-fold>
263     //</editor-fold>
264     //</editor-fold>
265
266     /* Create and display the form */
267     java.awt.EventQueue.invokeLater(new Runnable() {
268         public void run() {
269             new Login_done(1).setVisible(true);
270         }
271     });
272 }
273
274
275 // Variables declaration - do not modify
276 private javax.swing.JButton back;
```

```
  ...
292 }
293
294 // Variables declaration - do not modify
295 private javax.swing.JButton back;
296 private javax.swing.JLabel jLabel1;
297 private javax.swing.JLabel jLabel2;
298 private javax.swing.JPanel jPanel1;
299 private javax.swing.JButton login;
300 private javax.swing.JPasswordField password;
301 private javax.swing.JButton register;
302 private javax.swing.JCheckBox show;
303 private javax.swing.JLabel title;
304 private javax.swing.JTextField username;
305 // End of variables declaration
306 }
307 }
```

GUI:

ACCOUNT_LOGIN

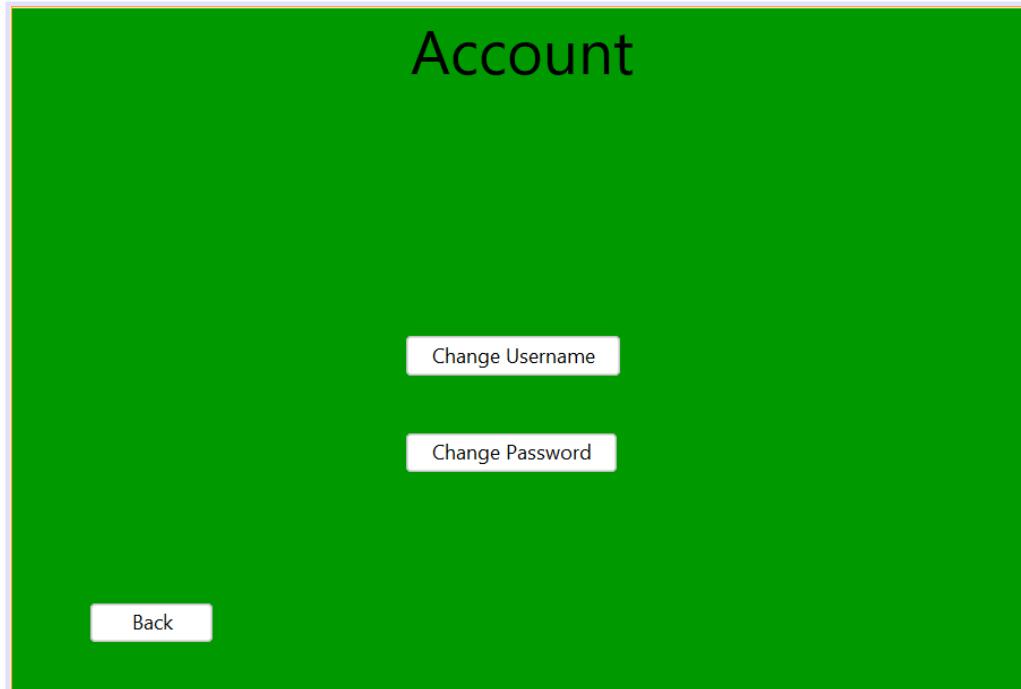
CODE:

```

5   package javaapplication2;
6
7   /**
8    * 
9    * @author Cody
10   */
11  public class Officer_Account extends javax.swing.JFrame {
12      String username;
13
14      /**
15       * Creates new form Officer_Account
16       */
17      public Officer_Account(String user) {
18          initComponents();
19          username = user;
20      }
21
22      /**
23       * This method is called from within the constructor to initialize the form.
24       * WARNING: Do NOT modify this code. The content of this method is always
25       * regenerated by the Form Editor.
26      */
27      @SuppressWarnings("unchecked")
28      // Generated Code
29
30      private void changeUsernameActionPerformed(java.awt.event.ActionEvent evt) {
31          // TODO add your handling code here:
32      }
33
34      private void changePasswordActionPerformed(java.awt.event.ActionEvent evt) {
35          // TODO add your handling code here:
36      }
37
38      private void backActionPerformed(java.awt.event.ActionEvent evt) {
39          OfficerPage op = new OfficerPage(username);
40          op.show();
41          dispose();
42      }
43
44  }
45  javaapplication2.Officer_Account >
```

```

120      ...
121      dispose();
122  }
123
124  /**
125   * @param args the command line arguments
126   */
127  public static void main(String args[]) {
128      /* Set the Nimbus look and feel */
129      // Look and feel setting code (optional)
130
131      /* Create and display the form */
132      java.awt.EventQueue.invokeLater(new Runnable() {
133          private String user;
134          public void run() {
135              new Officer_Account(user).setVisible(true);
136          }
137      });
138
139      // Variables declaration - do not modify
140      private javax.swing.JButton back;
141      private javax.swing.JButton changePassword;
142      private javax.swing.JButton changeUsername;
143      private javax.swing.JLabel jLabel1;
144      private javax.swing.JPanel jPanel1;
145      // End of variables declaration
146  }
147
```

GUI:

Page of Login GUI: To log in, enter your username and password. By analysing text files for user credentials, the `Login_done` class authorises access to user pages according to the user's role (e.g., officer, administrator, salesperson).

Landing Page GUI: Users are redirected to their role-specific landing page upon logging in. Users can view sales history, submit sales, verify product status, generate reports, manage accounts, or log out via the icons that appear on this page.

9.0 ADDITIONAL FEATURES

9.1 ABDUL MUHAIMIN AMAN

9.1.0 BACKGROUND IMAGE

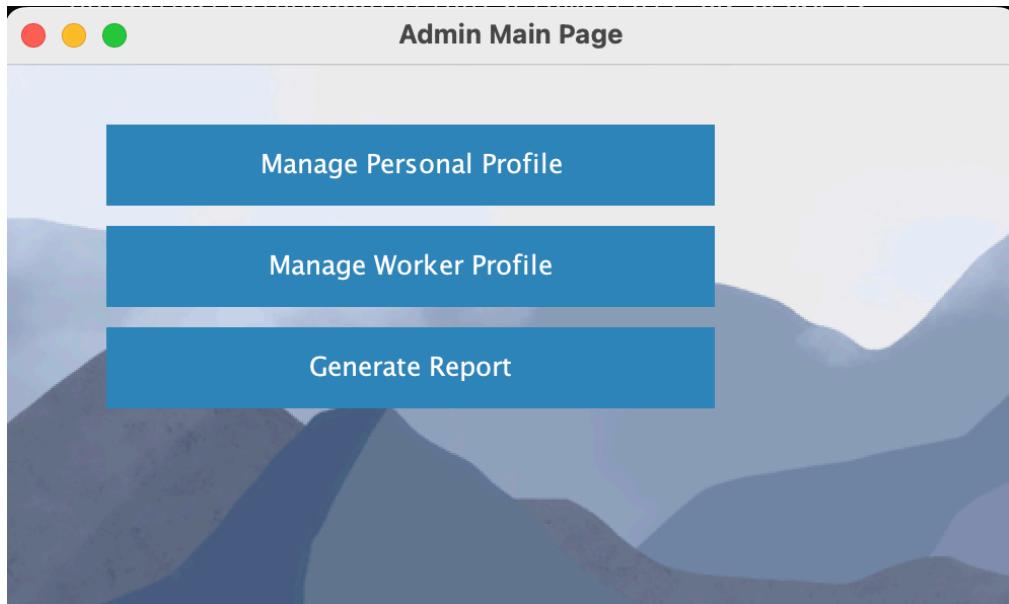
CODE:

```
// Create a JLabel to hold the background image
JLabel backgroundImageLabel = new JLabel();
setContentPane(backgroundImageLabel);

// Load the background image
ImageIcon backgroundImage = new ImageIcon("/Users/aman/eclipse-workspace/oodjaman/src/oodjaman/pngtree-big-mountain-bo
backgroundImageLabel.setIcon(backgroundImage);

// Set bounds for the buttons and background image
managePersonalProfileButton.setBounds(50, 30, 300, 40);
manageWorkerProfileButton.setBounds(50, 80, 300, 40);
generateReportButton.setBounds(50, 130, 300, 40);
backgroundImageLabel.setBounds(0, 0, getWidth(), getHeight());
```

GUI:



9.1.1 STYLED BUTTONS

CODE:

```

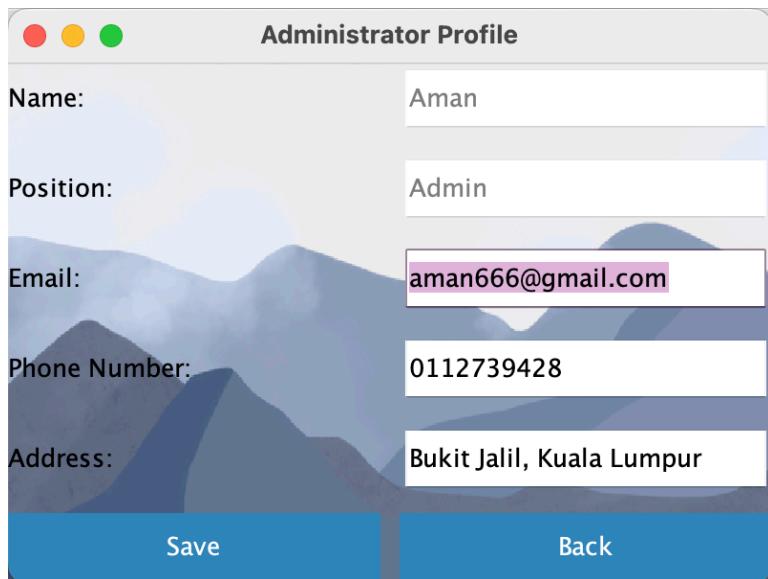
saveButton = createStyledButton("Save");
saveButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        saveChanges();
    }
});

// Create "Go Back" button
goBackButton = createStyledButton("Back");
goBackButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        dispose(); // Close the current frame
    }
});

private JButton createStyledButton(String text) {
    JButton button = new JButton(text);
    button.setBackground(new Color(70, 130, 180)); // Light blue
    button.setForeground(Color.WHITE);
    button.setFocusPainted(false);
    button.setBorderPainted(false);
    button.setOpaque(true);
    return button;
}

```

GUI:



9.1.2 GRADIENT COLOUR BACKGROUND

CODE:

```
// Set up gradient background
setContentPane(new JPanel() {
    @Override
    protected void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        Color color1 = new Color(135, 206, 250); // Light blue
        Color color2 = new Color(255, 255, 255); // White
        GradientPaint gp = new GradientPaint(0, 0, color1, getWidth(), getHeight(), color2);
        g2d.setPaint(gp);
        g2d.fillRect(0, 0, getWidth(), getHeight());
    }
});
```

GUI:

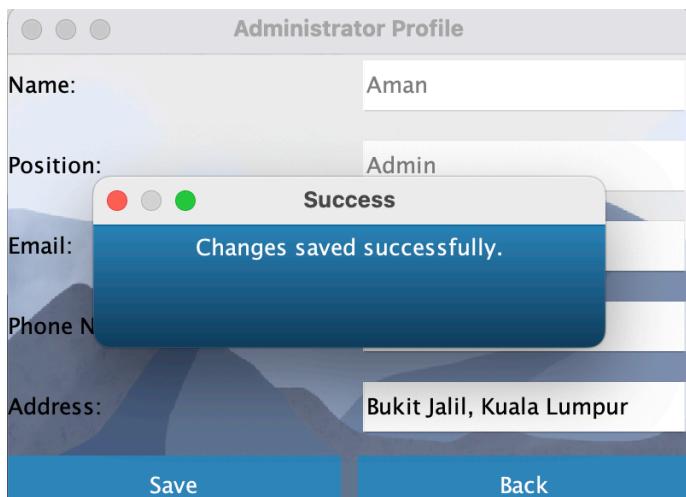


9.1.3 TIMER FOR DIALOG BOX

CODE:

```
// Set a timer to close the dialog after 2 seconds (2000 milliseconds)
Timer timer = new Timer(2000, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        dialog.dispose();
    }
});
timer.setRepeats(false); // Only run the timer once
timer.start();
dialog.setVisible(true);
}
```

GUI:



9.1.3 ALTERNATING ROW COLOURS

CODE:

```
// Handle selection state
if (isRowSelected(row)) {
    comp.setBackground(new Color(70, 130, 180)); // Light blue for selected row
} else {
    comp.setBackground(row % 2 == 0 ? Color.WHITE : new Color(255, 255, 255)); // Alternating row colors
}

return comp;
};
```

GUI:

Closed Sale Report

Item ID	Item Name	Category	Price	Order Date	Status
3	Franklin	Bar Furniture	3500	5/1/2024	Approved
100	Neevra	Beds	6000	1/2/2024	Approved
5	Noverra	Beds	4500	12/2/2024	Approved
123456	Gopal	sofa	99999	12-12-2012	Approved
12345678	chair	9632	8888	12-1-2003	Approved
12345	table	123	7777	1-1-2012	Approved

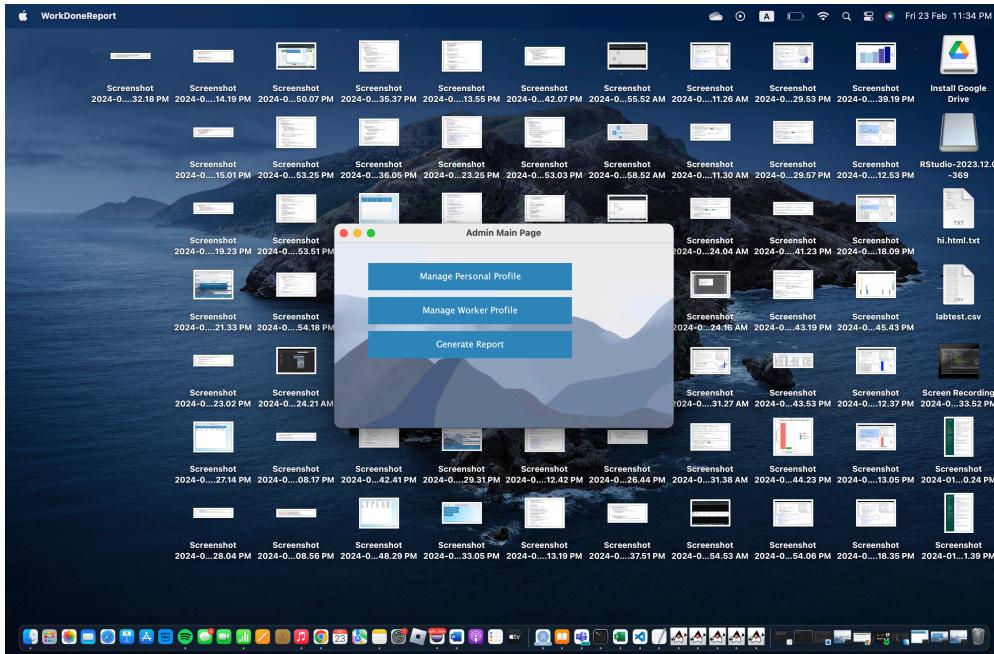
[Go Back](#)

9.1.4 CENTERING FRAME TO SCREEN

CODE:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
setLocationRelativeTo(null);
```

GUI:



10.0 CONCLUSION

To sum up, the Yoyo-Furniture Sales Management System provides a comprehensive approach to addressing the difficulties encountered by the furniture enterprise in overseeing its sales activities. The system offers a user-friendly interface and strong functionality for simplifying sale order processing, invoice production, report generation, and product administration by using contemporary technology and object-oriented design concepts. The system reduces possible security concerns by providing officers, administrators, and salespeople with safe access control using authentication and authorization processes. All things considered, the Yoyo-Furniture Sales Management System is expected to improve productivity, accuracy, and efficiency in duties linked to sales, which will eventually help the firm succeed and expand in the highly competitive industry.

11.0 WORKLOAD MATRIX

NAME	TP NUMBER	TASKS
ABDUL MUHAIMIN AMAN	TP069510	<ul style="list-style-type: none"> • Administrator: · Manage personal profile · Manage worker profile (saleperson and officer) · Generate report (Type: work done report, approved/closed sale report) • Introduction • Table of content • Objective • Scope • Limitations • Class Diagram • Concepts used • Implementation • Additional features • Compiling Documentation • Workload Matrix
KERTANAA A/P KUMAR	TP072664	<ul style="list-style-type: none"> • Salesperson: · Manage personal profile · Manage sale order quotation (create/modify/remove/search) · List all personal sale orders (unapproved/approved) • Use case diagram • Concepts used • Implementation • Documentation

		<ul style="list-style-type: none"> • Conclusion • Workload Matrix
HOOUSHIANEE HARHANGI	TP066080	<ul style="list-style-type: none"> • GUI interface • Concepts used • Authentication and Authorisation • Implementation
AREEBAH IRFAN MOHAMMED	TP072275	<ul style="list-style-type: none"> • Officer: · Manage personal profile · Process sale order upon sale approval/ closed sale · Submit sales for production · Check sale product status for work done or in progress status · Generate report (Type: work done report, approved/closed sale report • Implementation