UOW MALAYSIA KDU PENANG UNIVERSITY COLLEGE

PART OF THE UNIVERSITY OF WOLLONGONG AUSTRALIA GLOBAL NETWORK

ASSIGNMENT COVER PAGE

Programme	**Course Code and Title**
Diploma in Computer Studies / Diploma in Information Technology	DJP2264 Java Programming
Student's name / student's id	**Lecturer's name**
• 0205096 THOR WEN ZHENG • 0204677 LIM ZHE YUAN • 0205034 NAVEENRAAJ A/L P THINARTHAN • 0205430 TAN PENG HENG	Tan Phit Huan

Date issued	**Submission Deadline**	**Indicative Weighting**
Week 1 -02/06/2021	Week 9 – 30/07/2021	20%
Assignment 2 title	Car Record Management system	

This assessment assesses the following course learning outcomes

# as in Course Guide	**UOWM KDU Penang University College Learning Outcome**
CLO3	Build a Java GUI application with events handling application

Student's declaration

I certify that the work submitted for this assignment is my own and research sources are fully acknowledged.

Student's signature: ZHE YUAN

Submission Date: 30/7/2021

Table of Contents

MAIN REPORT

COMPLETE PROGRAM SOURCE CODE

- **Car.java**

```
public class Car {
    private String plateNumber;
    private String brand;
    private String model;
    private String type;
    private String colour;
    private String status;
    private double price;

    public Car() {
        plateNumber = "";
        brand = "";
        model = "";
        type = "";
        colour = "";
        status = "";
        price = 0.00;
        price = 0;
    }

    public Car(String pn, String br, String mo, String ty, String co, String st,
double pr) {
        plateNumber = pn;
        brand = br;
        model = mo;
        type = ty;
        colour = co;
        status = st;
        price = pr;
    }

    public void setPlateNumber(String pn) {
        plateNumber = pn;
    }

    public void setBrand(String br) {
        brand = br;
    }
```

```java
    public void setModel(String mo) {
        model = mo;
    }

    public void setType(String ty) {
        type = ty;
    }

    public void setColour(String co) {
        colour = co;
    }

    public void setStatus(String st) {
        status = st;
    }

    public void setPrice(double pr) {
        price = pr;
    }

    public String getPlateNumber() {
        return plateNumber;
    }

    public String getBrand() {
        return brand;
    }

    public String getModel() {
        return model;
    }

    public String getType() {
        return type;
    }

    public String getColour() {
        return colour;
    }

    public String getStatus() {
        return status;
    }

    public double getPrice() {
```

```
        return price;
    }

    public String toString() {
        return "Plate number: " + plateNumber
                + "\nBrand: " + brand
                + "\nModel: " + model
                + "\nType: " + type
                + "\nColour: " + colour
                + "\nStatus: " + status
                + "\nPrice: " + price;
    }
}
```

- **Login.java**

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.Arrays;

public class Login extends JPanel implements ActionListener {

    // String identifiers for main deck cards associated with this function
    private final static String MAIN = "Main Menu Panel";

    // -------------------- "LOGIN" PANEL COMPONENTS -------------------- //
    private JPanel panelSystemTitle = new JPanel(new GridBagLayout());
    private JPanel panelInput = new JPanel(new GridLayout(3, 1));
    private JPanel panelUsername = new JPanel(new FlowLayout(FlowLayout.CENTER));
    private JPanel panelPassword = new JPanel(new FlowLayout(FlowLayout.CENTER));
    private JPanel panelOptions = new JPanel(new BorderLayout());
    private JPanel panelButtons = new JPanel(new FlowLayout(FlowLayout.CENTER));
    private JLabel lblSystemTitle = new JLabel("Car Record Management System", JL
abel.CENTER);
    private JLabel lblUsername = new JLabel("USERNAME");
    private JLabel lblPassword = new JLabel("PASSWORD");
    private JTextField txtUsername = new JTextField(22);
    private JPasswordField passwordField = new JPasswordField(22);
    private JCheckBox showPassword = new JCheckBox("Show password");
    private JButton btnClear = new JButton("CLEAR");
    private JButton btnLogin = new JButton("LOGIN");

    // Function variables
```

```java
    private static final String USERNAME = "HelloWorld";
    private static final char[] PASSWORD = {'1', '2', '3'};

    public Login() {
        makePanel();
    }

    public void makePanel() {
        setLayout(new BorderLayout());

        // Make system title (lblSystemTitle)
        lblSystemTitle.setFont(new Font("Helvetica", Font.BOLD, 20));
        panelSystemTitle.setPreferredSize(new Dimension(0, 70));
        panelSystemTitle.add(lblSystemTitle);

        // Make input panel (panelInput)
        txtUsername.setPreferredSize(new Dimension(0, 30));
        txtUsername.addActionListener(this);
        panelUsername.add(lblUsername);
        panelUsername.add(txtUsername);
        passwordField.setPreferredSize(new Dimension(0, 30));
        passwordField.addActionListener(this);
        panelPassword.add(lblPassword);
        panelPassword.add(passwordField);
        showPassword.setFont(new Font("Sans-serif", Font.PLAIN, 11));
        showPassword.setHorizontalAlignment(SwingConstants.CENTER);
        showPassword.setBorder(new EmptyBorder(0, 0, 0, 54));
        showPassword.addActionListener(this);
        panelOptions.add(showPassword, BorderLayout.NORTH);
        panelInput.add(panelUsername);
        panelInput.add(panelPassword);
        panelInput.add(panelOptions);

        // Make button panel (panelButtons)
        btnLogin.addActionListener(this);
        btnClear.addActionListener(this);
        panelButtons.setPreferredSize(new Dimension(0, 70));
        panelButtons.add(btnClear);
        panelButtons.add(btnLogin);

        // Add all sub-panels into parent panel
        add(panelSystemTitle, BorderLayout.NORTH);
        add(panelInput, BorderLayout.CENTER);
        add(panelButtons, BorderLayout.SOUTH);
    }
```

```java
    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        CardLayout mainCardLayout = (CardLayout) Main.panelContainer.getLayout();
        Window window = (Window) this.getTopLevelAncestor();

        String username = txtUsername.getText();
        char[] password = passwordField.getPassword();

        if (source == txtUsername || source == passwordField || source == btnLogi
n) {
            // Move user to the main menu if username and password are correct; e
lse show error messages
            if (username.equals(USERNAME) && Arrays.equals(password, PASSWORD)) {
                JOptionPane.showMessageDialog(null, "You have successfully logged
 in.",
                        "Login successful", JOptionPane.INFORMATION_MESSAGE);
                mainCardLayout.show(Main.panelContainer, MAIN);
                window.setSize(Main.WIDTH, Main.HEIGHT);
            } else if (username.trim().isEmpty() || password.length == 0) {
                JOptionPane.showMessageDialog(null, "Please ensure that there are
 no empty input fields.",
                        "Empty input field detected", JOptionPane.ERROR_MESSAGE);
            } else if (!username.equals(USERNAME) || !Arrays.equals(password, PAS
SWORD)) {
                JOptionPane.showMessageDialog(null, "Invalid username or password
.",
                        "Login Failed", JOptionPane.ERROR_MESSAGE);
            }
        } else if (source == showPassword) {
            // Show the password characters if "Show Password" is checked
            if (showPassword.isSelected())
                passwordField.setEchoChar((char) 0); // reveals characters
            else
                passwordField.setEchoChar('\u2022'); // \u2022: unicode char "BUL
LET"
        } else if (source == btnClear) {
            // Clear all input fields
            txtUsername.setText("");
            passwordField.setText("");
        }
    }

}
```

- **Main.java**

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class Main extends JFrame implements ActionListener {

    // Main variables
    private ArrayList<Car> carArrayList = new ArrayList<Car>();
    // Static constant variables
    static final int WIDTH = 700;
    static final int HEIGHT = 550;
    static final Dimension HEADER_SIZE = new Dimension(0, 70);
    // String identifiers for every card(panel objects of main functions)
    private static final String LOGIN = "Login Panel";
    private static final String MAIN = "Main Menu Panel";
    private static final String CREATE = "Create Record Panel";
    private static final String DELETE = "Delete Record Panel";
    private static final String EDIT = "Edit Record Panel";
    private static final String SEARCH = "Search Record Panel";
    private static final String DISPLAY = "Display All Records Panel";
    // Panel objects representing the GUI pages of the main system functions
    private Login panelLogin = new Login();
    private CreateRecord panelCreate = new CreateRecord(carArrayList);
    private DeleteRecord panelDelete = new DeleteRecord(carArrayList);
    private EditRecord panelEdit = new EditRecord(carArrayList);
    private SearchRecord panelSearch = new SearchRecord(carArrayList);
    private DisplayRecords panelDisplay = new DisplayRecords(carArrayList);

    // -------------------- "MAIN" FRAME COMPONENTS -------------------- //
    static JPanel panelContainer = new JPanel(new CardLayout());
    private JPanel panelMain = new JPanel(new BorderLayout());
    private JPanel panelHeader = new JPanel(new GridBagLayout());
    private JPanel panelButtons = new JPanel();
    private JLabel lblMainMenu = new JLabel("MAIN MENU");
    private JButton btnCreate = new JButton("Create Record");
    private JButton btnDelete = new JButton("Delete Record");
    private JButton btnEdit = new JButton("Edit Record");
    private JButton btnSearch = new JButton("Search Record");
    private JButton btnDisplay = new JButton("Display All Records");
    private Dimension sizeBtn = new Dimension(300, 50);
    private Dimension marginBtn = new Dimension(0, 20);
```

```java
    public Main() {
        super("Car Record Management System");
        makeFrame();
        showFrame();
    }

    public void makeFrame() {
        // Make header bar (panelHeader)
        lblMainMenu.setForeground(Color.WHITE);
        lblMainMenu.setFont(new Font("Helvetica", Font.BOLD, 20));
        panelHeader.setBackground(Color.DARK_GRAY);
        panelHeader.setPreferredSize(HEADER_SIZE);
        panelHeader.add(lblMainMenu);

        // Make buttons panel (panelButtons)
        panelButtons.setLayout(new BoxLayout(panelButtons, BoxLayout.Y_AXIS));
        btnCreate.setMaximumSize(sizeBtn);
        btnDelete.setMaximumSize(sizeBtn);
        btnSearch.setMaximumSize(sizeBtn);
        btnEdit.setMaximumSize(sizeBtn);
        btnDisplay.setMaximumSize(sizeBtn);
        btnCreate.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnDelete.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnEdit.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnSearch.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnDisplay.setAlignmentX(Component.CENTER_ALIGNMENT);
        btnCreate.addActionListener(this);
        btnDelete.addActionListener(this);
        btnEdit.addActionListener(this);
        btnSearch.addActionListener(this);
        btnDisplay.addActionListener(this);
        panelButtons.add(Box.createRigidArea(marginBtn));
        panelButtons.add(btnCreate);
        panelButtons.add(Box.createRigidArea(marginBtn));
        panelButtons.add(btnDelete);
        panelButtons.add(Box.createRigidArea(marginBtn));
        panelButtons.add(btnEdit);
        panelButtons.add(Box.createRigidArea(marginBtn));
        panelButtons.add(btnSearch);
        panelButtons.add(Box.createRigidArea(marginBtn));
        panelButtons.add(btnDisplay);

        // Make main menu panel (panelMain)
        panelMain.add(panelHeader, BorderLayout.PAGE_START);
        panelMain.add(panelButtons, BorderLayout.CENTER);
```

```java
        // Add all function panels to card deck (panelContainer)
        panelContainer.add(panelLogin, LOGIN);
        panelContainer.add(panelMain, MAIN);
        panelContainer.add(panelCreate, CREATE);
        panelContainer.add(panelDelete, DELETE);
        panelContainer.add(panelEdit, EDIT);
        panelContainer.add(panelSearch, SEARCH);
        panelContainer.add(panelDisplay, DISPLAY);

        // Add card deck to frame
        add(panelContainer);
    }

    public void showFrame() {
        setResizable(false);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        // Set 500 x 300 for login panel
        setSize(500, 300);
        setVisible(true);
    }

    public static void main(String[] args) {
        Main frame = new Main();

        // Exit confirmation dialog box
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                int response = JOptionPane.showConfirmDialog(frame,
                        "        ARE YOU SURE?", "Exit System",
                        JOptionPane.YES_NO_OPTION);
                if (response == JOptionPane.YES_OPTION)
                    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                else if (response == JOptionPane.NO_OPTION)
                    frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
            }
        });
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        CardLayout cardLayout = (CardLayout) panelContainer.getLayout();
```

```
        // Show the panel (card) of the selected function
        if (source == btnCreate) {
            cardLayout.show(panelContainer, CREATE);
            this.setSize(WIDTH, HEIGHT + 100);
        } else if (source == btnDelete) {
            cardLayout.show(panelContainer, DELETE);
            this.setSize(500, 250);
        } else if (source == btnEdit) {
            cardLayout.show(panelContainer, EDIT);
            this.setSize(500, 250);
        } else if (source == btnSearch) {
            cardLayout.show(panelContainer, SEARCH);
            this.setSize(500, 250);
        } else if (source == btnDisplay) {
            panelDisplay.updateTable(false);
            // Only move to display panel if there are existing records
            if (panelDisplay.tableIsEmpty()) {
                JOptionPane.showMessageDialog(null, "There are currently no recor
ds in the system.",
                        "Function Not Available", JOptionPane.INFORMATION_MESSAGE
);
            } else {
                cardLayout.show(panelContainer, DISPLAY);
                this.setSize(WIDTH + 50, HEIGHT);
            }
        }
    }

}
```

- **CreateRecord.java**

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class CreateRecord extends JPanel implements ActionListener, KeyListen
er {

    // String identifiers for main deck cards associated with this function
    private final static String MAIN = "Main Menu Panel";
```

```
    // -------------------- "CREATE RECORD" PANEL COMPONENTS ---------------
---- //
    private JPanel panelHeader = new JPanel(new FlowLayout(FlowLayout.LEFT, 2
0, 10));
    private JPanel panelTitle = new JPanel(new GridLayout(2, 1));
    private JPanel panelInput = new JPanel(new GridLayout(4, 2));
    private JPanel panelPlateNumber = new JPanel(new GridLayout(3, 1));
    private JPanel panelBrand = new JPanel(new GridLayout(3, 1));
    private JPanel panelModel = new JPanel(new GridLayout(3, 1));
    private JPanel panelType = new JPanel(new GridLayout(3, 1));
    private JPanel panelColour = new JPanel(new GridLayout(3, 1));
    private JPanel panelStatus = new JPanel(new GridLayout(3, 1));
    private JPanel panelPrice = new JPanel(new GridLayout(3, 1));
    private JPanel panelButtons = new JPanel(new FlowLayout(FlowLayout.CENTER
, 170, 0));
    private JLabel lblTitle = new JLabel("CREATE RECORD", JLabel.CENTER);
    private JLabel lblDescription = new JLabel("Enter the details of the new
car record.", JLabel.CENTER);
    private JLabel lblPlateNumber = new JLabel("Plate Number");
    private JLabel lblBrand = new JLabel("Brand");
    private JLabel lblModel = new JLabel("Model");
    private JLabel lblType = new JLabel("Type");
    private JLabel lblColour = new JLabel("Colour");
    private JLabel lblStatus = new JLabel("Status");
    private JLabel lblPrice = new JLabel("Price");
    private JLabel lblErrorPlateNumber = new JLabel();
    private JLabel lblErrorBrand = new JLabel();
    private JLabel lblErrorModel = new JLabel();
    private JLabel lblErrorType = new JLabel();
    private JLabel lblErrorColour = new JLabel();
    private JLabel lblErrorStatus = new JLabel();
    private JLabel lblErrorPrice = new JLabel();
    private JTextField txtPlateNumber = new JTextField(10);
    private JTextField txtBrand = new JTextField(10);
    private JTextField txtModel = new JTextField(10);
    private JTextField txtType = new JTextField(10);
    private JTextField txtColour = new JTextField(10);
    private JTextField txtStatus = new JTextField(10);
    private JTextField txtPrice = new JTextField(10);
    private JButton btnBack = new JButton("< BACK");
    private JButton btnClear = new JButton("CLEAR");
    private JButton btnCreate = new JButton("CREATE");
    private Font fontError = new Font("Sans-serif", Font.PLAIN, 11);
    private Dimension sizeBackBtn = new Dimension(80, 35);
    private Dimension sizeActionBtn = new Dimension(130, 40);
```

```java
// Function variables
ArrayList<Car> carArrayList;

public CreateRecord(ArrayList<Car> cars) {
    carArrayList = cars;
    makePanel();
}

public void makePanel() {
    setLayout(new BorderLayout());

    // Make header panel (panelHeader)
    btnBack.setPreferredSize(sizeBackBtn);
    btnBack.addActionListener(this);
    lblTitle.setFont(new Font("Helvetica", Font.BOLD, 16));
    lblTitle.setForeground(Color.WHITE);
    lblDescription.setFont(new Font("Helvetica", Font.PLAIN, 11));
    lblDescription.setForeground(Color.WHITE);
    panelTitle.setBorder(new EmptyBorder(0, Main.WIDTH / 6, 0, 0));
    panelTitle.setBackground(Color.DARK_GRAY);
    panelTitle.add(lblTitle);
    panelTitle.add(lblDescription);
    panelHeader.setBackground(Color.DARK_GRAY);
    panelHeader.add(btnBack);
    panelHeader.add(panelTitle);

    // Make input panel (panelInput)
    GridLayout inputGridLayout = (GridLayout) panelInput.getLayout();
    inputGridLayout.setHgap(50);

    lblErrorPlateNumber.setFont(fontError);
    lblErrorPlateNumber.setForeground(Color.RED);
    txtPlateNumber.addKeyListener(this);
    panelPlateNumber.add(lblPlateNumber);
    panelPlateNumber.add(txtPlateNumber);
    panelPlateNumber.add(lblErrorPlateNumber);

    lblErrorBrand.setFont(fontError);
    lblErrorBrand.setForeground(Color.RED);
    txtBrand.addKeyListener(this);
    panelBrand.add(lblBrand);
    panelBrand.add(txtBrand);
    panelBrand.add(lblErrorBrand);
```

```
        lblErrorModel.setFont(fontError);
        lblErrorModel.setForeground(Color.RED);
        txtModel.addKeyListener(this);
        panelModel.add(lblModel);
        panelModel.add(txtModel);
        panelModel.add(lblErrorModel);

        lblErrorType.setFont(fontError);
        lblErrorType.setForeground(Color.RED);
        txtType.addKeyListener(this);
        panelType.add(lblType);
        panelType.add(txtType);
        panelType.add(lblErrorType);

        lblErrorColour.setFont(fontError);
        lblErrorColour.setForeground(Color.RED);
        txtColour.addKeyListener(this);
        panelColour.add(lblColour);
        panelColour.add(txtColour);
        panelColour.add(lblErrorColour);

        lblErrorStatus.setFont(fontError);
        lblErrorStatus.setForeground(Color.RED);
        txtStatus.addKeyListener(this);
        panelStatus.add(lblStatus);
        panelStatus.add(txtStatus);
        panelStatus.add(lblErrorStatus);

        lblErrorPrice.setFont(fontError);
        lblErrorPrice.setForeground(Color.RED);
        txtPrice.addKeyListener(this);
        panelPrice.add(lblPrice);
        panelPrice.add(txtPrice);
        panelPrice.add(lblErrorPrice);

        panelInput.add(panelPlateNumber);
        panelInput.add(panelBrand);
        panelInput.add(panelModel);
        panelInput.add(panelType);
        panelInput.add(panelColour);
        panelInput.add(panelStatus);
        panelInput.add(panelPrice);
        panelInput.setBorder(new EmptyBorder(0, Main.WIDTH / 6, 0, Main.WIDTH
/ 6));
```

```
        // Make buttons panel (panelButtons)
        btnClear.setPreferredSize(sizeActionBtn);
        btnCreate.setPreferredSize(sizeActionBtn);
        btnClear.addActionListener(this);
        btnCreate.addActionListener(this);
        panelButtons.setPreferredSize(new Dimension(0, 100));
        panelButtons.add(btnClear);
        panelButtons.add(btnCreate);

        // Add all sub-panels into parent panel (this)
        add(panelHeader, BorderLayout.NORTH);
        add(panelInput, BorderLayout.CENTER);
        add(panelButtons, BorderLayout.SOUTH);
    }

    @Override
    public void actionPerformed(ActionEvent actionEvent) {
        Object source = actionEvent.getSource();
        CardLayout mainCardLayout = (CardLayout) Main.panelContainer.getLayou
t();
        Window window = (Window) this.getTopLevelAncestor();

        // Event Handling
        if (source == btnBack) {
            // Prompt user to confirm if they want to return to menu
            int response = JOptionPane.showConfirmDialog(null, "Are you sure
you want to return to the menu?\n" +
                    "All data that you have entered will be discarded.", "Bac
k To Menu", JOptionPane.YES_NO_OPTION);

            // Return to menu if response is YES; else cancel
            if (response == JOptionPane.YES_OPTION) {
                // Clear all input fields
                clearInputFields();

                // Return to menu
                mainCardLayout.show(Main.panelContainer, MAIN);
                window.setSize(Main.WIDTH, Main.HEIGHT);
            }
        } else if (source == btnClear) {
            // Clear all input fields
            clearInputFields();
        } else if (source == btnCreate) {
            boolean hasError = false;
            boolean isNumeric = false;
```

```
            trimAllInput();
            // Get input from all input fields
            String plateNumber = txtPlateNumber.getText();
            String brand = txtBrand.getText();
            String model = txtModel.getText();
            String type = txtType.getText();
            String colour = txtColour.getText();
            String status = txtStatus.getText();
            String strPrice = txtPrice.getText();
            double dblPrice = 0;

            // - - - - - Validate all input - - - - - //
            // Additional validation for plate number
            if (!isValid(plateNumber, 12, lblErrorPlateNumber)) {
                hasError = true;
            } else {
                // Check for duplicate plate number
                for (int i = 0; i < carArrayList.size(); i++) {
                    if (carArrayList.get(i).getPlateNumber().compareToIgnoreC
ase(plateNumber) == 0) {
                        lblErrorPlateNumber.setText("This plate number alread
y exists.");
                        hasError = true;
                    }
                }
            }
            // This here is not redundancy, it's necessary because of the way
 isValid is designed
            if (!isValid(brand, 16, lblErrorBrand))
                hasError = true;
            if (!isValid(model, 16, lblErrorModel))
                hasError = true;
            if (!isValid(type, 16, lblErrorType))
                hasError = true;
            if (!isValid(colour, 16, lblErrorColour))
                hasError = true;
            if (!isValid(status, 16, lblErrorStatus))
                hasError = true;
            // Additional validation for price input
            if (!isValid(strPrice, 9, lblErrorPrice)) {
                hasError = true;
            } else {
                // Attempt to parse string input into double
                try {
```

```
                    dblPrice = Double.parseDouble(strPrice);
                    isNumeric = true;
                } catch (NumberFormatException e) {
                    isNumeric = false;
                    lblErrorPrice.setText("Must be numeric value only.");
                }
            }

            // If all input pass their respective checks, add new record; els
e notify user of errors
            if (!hasError && isNumeric) {
                // Prompt user for confirmation
                int response = JOptionPane.showConfirmDialog(null, "Confirm N
ew Car Record",
                        "Create Record", JOptionPane.YES_NO_OPTION);

                // Add new car record if user clicks "YES"
                if (response == JOptionPane.YES_NO_OPTION) {
                    Car newCarObj = new Car(plateNumber, brand, model, type,
colour, status, dblPrice);
                    carArrayList.add(newCarObj);

                    // Notify user
                    JOptionPane.showMessageDialog(null, "New car record added
 successfully!",
                            "Record Creation Successful", JOptionPane.INFORMA
TION_MESSAGE);

                    // Return to menu
                    clearInputFields();
                    mainCardLayout.show(Main.panelContainer, MAIN);
                    window.setSize(Main.WIDTH, Main.HEIGHT);
                }
            } else {
                JOptionPane.showMessageDialog(null, "There are invalid inputs
, please check all the data.",
                        "Invalid input detected", JOptionPane.ERROR_MESSAGE);
            }

        }
    }

    @Override
    public void keyPressed(KeyEvent keyEvent) { /* not used */ }
```

```java
    @Override
    public void keyTyped(KeyEvent keyEvent) { /* not used */ }

    @Override
    public void keyReleased(KeyEvent keyEvent) { // For basic real-
time error-checking
        Object source = keyEvent.getSource();

        // Return values of isValid will not be used here
        if (source == txtPlateNumber)
            isValid(txtPlateNumber.getText().trim(), 12, lblErrorPlateNumber)
;
        else if (source == txtBrand)
            isValid(txtBrand.getText().trim(), 16, lblErrorBrand);
        else if (source == txtModel)
            isValid(txtModel.getText().trim(), 16, lblErrorModel);
        else if (source == txtType)
            isValid(txtType.getText().trim(), 16, lblErrorType);
        else if (source == txtColour)
            isValid(txtColour.getText().trim(), 16, lblErrorColour);
        else if (source == txtStatus)
            isValid(txtStatus.getText().trim(), 16, lblErrorStatus);
        else if (source == txtPrice)
            isValid(txtPrice.getText().trim(), 9, lblErrorPrice);
    }

    // Helper method for validating input
    public boolean isValid(String attribute, int limit, JLabel lblError) {
        // Validate input for each condition and set appropriate error messag
es accordingly
        if (attribute.isBlank()) {
            lblError.setText("Cannot be empty.");
            return false;
        } else if (attribute.length() > limit) {
            lblError.setText("Cannot be more than " + limit + " characters.")
;
            return false;
        } else {
            lblError.setText("");
            // Return true if input passes all checks
            return true;
        }
    }

    // Helper method for trimming input
```

```java
    public void trimAllInput() {
        txtPlateNumber.setText(txtPlateNumber.getText().trim());
        txtBrand.setText(txtBrand.getText().trim());
        txtModel.setText(txtModel.getText().trim());
        txtType.setText(txtType.getText().trim());
        txtColour.setText(txtColour.getText().trim());
        txtStatus.setText(txtStatus.getText().trim());
        txtPrice.setText(txtPrice.getText().trim());
    }

    // Helper method for clearing all input fields and their error messages
    public void clearInputFields() {
        txtPlateNumber.setText("");
        txtBrand.setText("");
        txtModel.setText("");
        txtType.setText("");
        txtColour.setText("");
        txtStatus.setText("");
        txtPrice.setText("");
        lblErrorPlateNumber.setText("");
        lblErrorBrand.setText("");
        lblErrorModel.setText("");
        lblErrorType.setText("");
        lblErrorColour.setText("");
        lblErrorStatus.setText("");
        lblErrorPrice.setText("");
    }

}
```

- **DeleteRecord.java**

```java
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class DeleteRecord extends JPanel implements ActionListener {

    // String identifiers for main deck cards associated with this function
    private final static String MAIN = "Main Menu Panel";
    // String identifiers for cards of this panel's card deck
    private final static String DELCARD1 = "DEL1 Input Panel";
    private final static String DELCARD2 = "DEL2 Confirmation Panel";
```

```
    // Cards for this function
    private JPanel panelDelCard1 = new JPanel(new BorderLayout());
    private JPanel panelDelCard2 = new JPanel(new BorderLayout());

    // -------------------- DELCARD1 PANEL COMPONENTS -------------------- //
    private JPanel panelHeader = new JPanel(new FlowLayout(FlowLayout.LEFT, 2
0, 10));
    private JPanel panelTitle = new JPanel(new GridLayout(2, 1));
    private JPanel panelInput = new JPanel(new GridLayout(2, 1));
    private JPanel panelButtons1 = new JPanel(new FlowLayout(FlowLayout.CENTE
R, 52, 15));
    private JPanel panelMargin1 = new JPanel();
    private JPanel panelMargin2 = new JPanel();
    private JLabel lblTitle = new JLabel("DELETE RECORD", JLabel.CENTER);
    private JLabel lblDescription = new JLabel("Enter the Plate Number of the
 car record to delete.", JLabel.CENTER);
    private JLabel lblPlateNumber1 = new JLabel("Plate Number");
    private JTextField txtPlateNumber1 = new JTextField(10);
    private JButton btnBack = new JButton("< BACK");
    private JButton btnClear = new JButton("CLEAR");
    private JButton btnContinue = new JButton("CONTINUE");
    private Dimension sizeBackBtn = new Dimension(80, 35);
    // -------------------- DELCARD2 PANEL COMPONENTS -------------------- //
    private JPanel panelFound = new JPanel(new FlowLayout(FlowLayout.LEFT, 40
, 20));
    private JPanel panelContent = new JPanel(new GridLayout(1, 2));
    private JPanel panelLeftCol = new JPanel(new GridLayout(4, 1));
    private JPanel panelRightCol = new JPanel(new GridLayout(4, 1));
    private JPanel panelPlateNumber = new JPanel(new GridLayout(2, 1));
    private JPanel panelBrand = new JPanel(new GridLayout(2, 1));
    private JPanel panelModel = new JPanel(new GridLayout(2, 1));
    private JPanel panelType = new JPanel(new GridLayout(2, 1));
    private JPanel panelColour = new JPanel(new GridLayout(2, 1));
    private JPanel panelStatus = new JPanel(new GridLayout(2, 1));
    private JPanel panelPrice = new JPanel(new GridLayout(2, 1));
    private JPanel panelButtons2 = new JPanel(new FlowLayout(FlowLayout.RIGHT
, 45, 20));
    private JLabel lblFound = new JLabel("MATCHING CAR RECORD FOUND");
    private JLabel lblPlateNumber2 = new JLabel("Plate Number");
    private JLabel lblBrand = new JLabel("Brand");
    private JLabel lblModel = new JLabel("Model");
    private JLabel lblType = new JLabel("Type");
    private JLabel lblColour = new JLabel("Colour");
    private JLabel lblStatus = new JLabel("Status");
    private JLabel lblPrice = new JLabel("Price");
```

```
    private JTextField txtPlateNumber2 = new JTextField(10);
    private JTextField txtBrand = new JTextField(10);
    private JTextField txtModel = new JTextField(10);
    private JTextField txtType = new JTextField(10);
    private JTextField txtColour = new JTextField(10);
    private JTextField txtStatus = new JTextField(10);
    private JTextField txtPrice = new JTextField(10);
    private JButton btnCancel = new JButton("CANCEL");
    private JButton btnDelete = new JButton("DELETE");

    // Function variables
    private ArrayList<Car> carArrayList;
    private int recordIndex;

    public DeleteRecord(ArrayList<Car> cars) {
        carArrayList = cars;
        makePanel();
    }

    public void makePanel() {
        setLayout(new CardLayout());

        // -------------------- CARD 1 (DEL1 INPUT PANEL) -------------------
- //
        // Make header bar (panelHeader)
        btnBack.setPreferredSize(sizeBackBtn);
        btnBack.addActionListener(this);
        lblTitle.setFont(new Font("Helvetica", Font.BOLD, 16));
        lblTitle.setForeground(Color.WHITE);
        lblDescription.setForeground(Color.WHITE);
        lblDescription.setFont(new Font("Helvetica", Font.PLAIN, 11));
        panelTitle.setBackground(Color.DARK_GRAY);
        panelTitle.add(lblTitle);
        panelTitle.add(lblDescription);
        panelHeader.setBackground(Color.DARK_GRAY);
        panelHeader.setPreferredSize(new Dimension(0, 60));
        panelHeader.add(btnBack);
        panelHeader.add(panelTitle);

        // Make margin panels (panelMargin#)
        panelMargin1.setPreferredSize(new Dimension(120, 0));
        panelMargin2.setPreferredSize(new Dimension(120, 0));

        // Make input panel (panelInput)
        txtPlateNumber1.addActionListener(this);
```

```
        panelInput.add(lblPlateNumber1);
        panelInput.add(txtPlateNumber1);

        // Make buttons panel (panelButtons)
        btnClear.setPreferredSize(btnContinue.getPreferredSize());
        btnClear.addActionListener(this);
        btnContinue.addActionListener(this);
        panelButtons1.setPreferredSize(new Dimension(0, 70));
        panelButtons1.add(btnClear);
        panelButtons1.add(btnContinue);

        // Add sub-panels into DEL1 card
        panelDelCard1.add(panelHeader, BorderLayout.NORTH);
        panelDelCard1.add(panelMargin1, BorderLayout.WEST);
        panelDelCard1.add(panelInput, BorderLayout.CENTER);
        panelDelCard1.add(panelMargin2, BorderLayout.EAST);
        panelDelCard1.add(panelButtons1, BorderLayout.SOUTH);

        // -------------------- CARD 2 (DEL2 CONFIRMATION PANEL) ------------
-------- //
        // Make title panel (panelFound)
        lblFound.setFont(new Font("Helvetica", Font.BOLD, 16));
        panelFound.setPreferredSize(Main.HEADER_SIZE);
        panelFound.add(lblFound);

        // Make left column panel (panelLeftCol)
        txtPlateNumber2.setEditable(false);
        txtBrand.setEditable(false);
        txtModel.setEditable(false);
        txtType.setEditable(false);
        panelPlateNumber.add(lblPlateNumber2);
        panelPlateNumber.add(txtPlateNumber2);
        panelBrand.add(lblBrand);
        panelBrand.add(txtBrand);
        panelModel.add(lblModel);
        panelModel.add(txtModel);
        panelType.add(lblType);
        panelType.add(txtType);
        panelLeftCol.add(panelPlateNumber);
        panelLeftCol.add(panelBrand);
        panelLeftCol.add(panelModel);
        panelLeftCol.add(panelType);
        panelLeftCol.setBorder(new EmptyBorder(0, 40, 0, 45));

        // Make right column panel (panelRightCol)
```

```
        txtColour.setEditable(false);
        txtStatus.setEditable(false);
        txtPrice.setEditable(false);
        panelColour.add(lblColour);
        panelColour.add(txtColour);
        panelStatus.add(lblStatus);
        panelStatus.add(txtStatus);
        panelPrice.add(lblPrice);
        panelPrice.add(txtPrice);
        panelRightCol.add(panelColour);
        panelRightCol.add(panelStatus);
        panelRightCol.add(panelPrice);
        panelRightCol.setBorder(new EmptyBorder(0, 40, 0, 45));

        // Make content panel (panelContent)
        panelContent.add(panelLeftCol);
        panelContent.add(panelRightCol);

        // Make buttons2 panel (panelButtons2)
        btnCancel.addActionListener(this);
        btnDelete.addActionListener(this);
        panelButtons2.setPreferredSize(Main.HEADER_SIZE);
        panelButtons2.add(btnCancel);
        panelButtons2.add(btnDelete);

        // Add sub-panels into DEL2 card
        panelDelCard2.add(panelFound, BorderLayout.NORTH);
        panelDelCard2.add(panelContent, BorderLayout.CENTER);
        panelDelCard2.add(panelButtons2, BorderLayout.SOUTH);

        // ----- Add both cards into card deck ----- //
        add(panelDelCard1, DELCARD1);
        add(panelDelCard2, DELCARD2);
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        CardLayout mainCardLayout = (CardLayout) Main.panelContainer.getLayou
t();
        CardLayout delCardLayout = (CardLayout) this.getLayout();

        Window window = (Window) this.getTopLevelAncestor(); // sauce: https:
//stackoverflow.com/questions/25346547/display-different-panel-size-in-
cardlayout
```

```
        // Event Handling for CARD 1 (DEL1 INPUT PANEL)
        if (source == btnBack) {
            // return to main menu
            txtPlateNumber1.setText("");
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        } else if (source == btnClear) {
            // clear plate number input
            txtPlateNumber1.setText("");
        } else if (source == txtPlateNumber1 || source == btnContinue) {
            String plateNumber = txtPlateNumber1.getText();
            int matchedIndex = 0;
            boolean isFound = false;

            // Check if there is a record with entered plateNumber
            for (int i = 0;i < carArrayList.size(); i++) {
                if (carArrayList.get(i).getPlateNumber().compareToIgnoreCase(
plateNumber) == 0) {
                    matchedIndex = i;
                    isFound = true;
                    break;
                }
            }

            // Display error message if not found; else proceed to delete con
firmation screen
            if (!isFound) {
                JOptionPane.showMessageDialog(null, "No car record with the e
ntered Plate Number was found.",
                        "Invalid Plate Number", JOptionPane.ERROR_MESSAGE);
            } else {
                // show delete confirmation screen
                delCardLayout.show(this, DELCARD2);
                window.setSize(600, 400);

                // populate the disabled text fields
                txtPlateNumber2.setText(carArrayList.get(matchedIndex).getPla
teNumber());
                txtBrand.setText(carArrayList.get(matchedIndex).getBrand());
                txtModel.setText(carArrayList.get(matchedIndex).getModel());
                txtType.setText(carArrayList.get(matchedIndex).getType());
                txtColour.setText(carArrayList.get(matchedIndex).getColour())
;
```

```
                txtStatus.setText(carArrayList.get(matchedIndex).getStatus())
;
                txtPrice.setText(String.format("RM %.2f", carArrayList.get(ma
tchedIndex).getPrice()));

                // store the matched index for use in DEL2 screen
                recordIndex = matchedIndex;
            }
        }

        // Event Handling for CARD 2 (DEL2 CONFIRMATION PANEL)
        if (source == btnCancel) {
            txtPlateNumber1.setText("");
            // Return to main menu
            JOptionPane.showMessageDialog(null, "Deletion process cancelled.
Returning to menu.",
                    "Deletion Cancelled", JOptionPane.OK_OPTION);
            delCardLayout.show(this, DELCARD1);
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        } else if (source == btnDelete) {
            // prompt user to confirm deletion
            int response = JOptionPane.showConfirmDialog(null, "        ARE Y
OU SURE?",
                    "Delete Confirmation", JOptionPane.YES_NO_OPTION);
            // process response
            if (response == JOptionPane.YES_OPTION) {
                JOptionPane.showMessageDialog(null, "Car record with Plate Nu
mber [" +
                                carArrayList.get(recordIndex).getPlateNumber(
) + "] has been deleted.",
                        "Deletion Successful", JOptionPane.INFORMATION_MESSAG
E);
                carArrayList.remove(recordIndex);
                delCardLayout.show(this, DELCARD1);
                mainCardLayout.show(Main.panelContainer, MAIN);
                window.setSize(Main.WIDTH, Main.HEIGHT);
            }

            txtPlateNumber1.setText("");
        }
    }

}
```

- **EditRecord.java**

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class EditRecord extends JPanel implements ActionListener {

    // String identifiers for main deck cards associated with this function
    private final static String MAIN = "Main Menu Panel";
    // String identifiers for cards of this panel's card deck
    private final static String EDITCARD1 = "EDIT1 Input Panel";
    private final static String EDITCARD2 = "EDIT2 Record Info Panel";
    // Cards for this function
    private JPanel panelEditCard1 = new JPanel(new BorderLayout());
    private JPanel panelEditCard2 = new JPanel(new BorderLayout());

    // -------------------- EDITCARD1 PANEL COMPONENTS -------------------- //
    private JPanel panelHeader = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 1
0));
    private JPanel panelTitle = new JPanel(new GridLayout(2, 1));
    private JPanel panelInput = new JPanel(new GridLayout(2, 1));
    private JPanel panelButtons1 = new JPanel(new FlowLayout(FlowLayout.CENTER, 5
2, 15));
    private JPanel panelMargin1 = new JPanel();
    private JPanel panelMargin2 = new JPanel();
    private JLabel lblTitle = new JLabel("EDIT RECORD", JLabel.CENTER);
    private JLabel lblDescription = new JLabel("Enter the Plate Number of the car
 record to edit.", JLabel.CENTER);
    private JLabel lblPlateNumber1 = new JLabel("Plate Number");
    private JTextField txtPlateNumber1 = new JTextField(10);
    private JButton btnBack = new JButton("< BACK");
    private JButton btnClear = new JButton("CLEAR");
    private JButton btnContinue = new JButton("CONTINUE");
    private Dimension sizeBackBtn = new Dimension(80, 35);
    // -------------------- EDITCARD2 PANEL COMPONENTS -------------------- //
    private JPanel panelFound = new JPanel(new FlowLayout(FlowLayout.LEFT, 40, 20
));
    private JPanel panelContent = new JPanel(new GridLayout(1, 2));
    private JPanel panelLeftCol = new JPanel(new GridLayout(4, 1));
    private JPanel panelRightCol = new JPanel(new GridLayout(4, 1));
    private JPanel panelPlateNumber = new JPanel(new GridLayout(2, 1));
    private JPanel panelBrand = new JPanel(new GridLayout(2, 1));
```

```
    private JPanel panelModel = new JPanel(new GridLayout(2, 1));
    private JPanel panelType = new JPanel(new GridLayout(2, 1));
    private JPanel panelColour = new JPanel(new GridLayout(2, 1));
    private JPanel panelStatus = new JPanel(new GridLayout(2, 1));
    private JPanel panelPrice = new JPanel(new GridLayout(2, 1));
    private JPanel editPlateNumber = new JPanel(new FlowLayout(FlowLayout.LEFT, 0
, 0));
    private JPanel editBrand = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
    private JPanel editModel = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
    private JPanel editType = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
    private JPanel editColour = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0))
;
    private JPanel editStatus = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0))
;
    private JPanel editPrice = new JPanel(new FlowLayout(FlowLayout.LEFT, 0, 0));
    private JPanel panelButtons2 = new JPanel(new FlowLayout(FlowLayout.RIGHT, 55
, 20));
    private JLabel lblFound = new JLabel("MATCHING CAR RECORD FOUND");
    private JLabel lblPlateNumber2 = new JLabel("Plate Number");
    private JLabel lblBrand = new JLabel("Brand");
    private JLabel lblModel = new JLabel("Model");
    private JLabel lblType = new JLabel("Type");
    private JLabel lblColour = new JLabel("Colour");
    private JLabel lblStatus = new JLabel("Status");
    private JLabel lblPrice = new JLabel("Price");
    private final static int sizeTextField = 13;
    private JTextField txtPlateNumber2 = new JTextField(sizeTextField);
    private JTextField txtBrand = new JTextField(sizeTextField);
    private JTextField txtModel = new JTextField(sizeTextField);
    private JTextField txtType = new JTextField(sizeTextField);
    private JTextField txtColour = new JTextField(sizeTextField);
    private JTextField txtStatus = new JTextField(sizeTextField);
    private JTextField txtPrice = new JTextField(sizeTextField);
    private JButton btnPlateNumber = new JButton("EDIT");
    private JButton btnBrand = new JButton("EDIT");
    private JButton btnModel = new JButton("EDIT");
    private JButton btnType = new JButton("EDIT");
    private JButton btnColour = new JButton("EDIT");
    private JButton btnStatus = new JButton("EDIT");
    private JButton btnPrice = new JButton("EDIT");
    private JButton btnDone = new JButton("DONE");

    // Function variables
    private ArrayList<Car> carArrayList;
    private int recordIndex;
```

```
    private final String INPUT_CANCEL = "THE PROCESS WAS CANCELLED";

    public EditRecord(ArrayList<Car> cars) {
        carArrayList = cars;
        makePanel();
    }

    public void makePanel() {
        setLayout(new CardLayout());

        // -------------------- CARD 1 (EDIT1 INPUT PANEL) -------------------
- //
        // Make header bar (panelHeader)
        btnBack.setPreferredSize(sizeBackBtn);
        btnBack.addActionListener(this);
        lblTitle.setFont(new Font("Helvetica", Font.BOLD, 16));
        lblTitle.setForeground(Color.WHITE);
        lblDescription.setForeground(Color.WHITE);
        lblDescription.setFont(new Font("Helvetica", Font.PLAIN, 11));
        panelTitle.setBackground(Color.DARK_GRAY);
        panelTitle.add(lblTitle);
        panelTitle.add(lblDescription);
        panelHeader.setBackground(Color.DARK_GRAY);
        panelHeader.setPreferredSize(new Dimension(0, 60));
        panelHeader.add(btnBack);
        panelHeader.add(panelTitle);

        // Make margin panels (panelMargin#)
        panelMargin1.setPreferredSize(new Dimension(120, 0));
        panelMargin2.setPreferredSize(new Dimension(120, 0));

        // Make input panel (panelInput)
        txtPlateNumber1.addActionListener(this);
        panelInput.add(lblPlateNumber1);
        panelInput.add(txtPlateNumber1);

        // Make buttons panel (panelButtons)
        btnClear.setPreferredSize(btnContinue.getPreferredSize());
        btnClear.addActionListener(this);
        btnContinue.addActionListener(this);
        panelButtons1.setPreferredSize(new Dimension(0, 70));
        panelButtons1.add(btnClear);
        panelButtons1.add(btnContinue);

        // Add sub-panels into EDIT1 card
```

```
        panelEditCard1.add(panelHeader, BorderLayout.NORTH);
        panelEditCard1.add(panelMargin1, BorderLayout.WEST);
        panelEditCard1.add(panelInput, BorderLayout.CENTER);
        panelEditCard1.add(panelMargin2, BorderLayout.EAST);
        panelEditCard1.add(panelButtons1, BorderLayout.SOUTH);

        // -------------------- CARD 2 (EDIT2 RECORD INFO PANEL) ----------------
---- //
        // Make title panel (panelFound)
        lblFound.setFont(new Font("Helvetica", Font.BOLD, 16));
        panelFound.setPreferredSize(Main.HEADER_SIZE);
        panelFound.add(lblFound);

        // Make left column panel (panelLeftCol)
        txtPlateNumber2.setEditable(false);
        txtPlateNumber2.setMargin(new Insets(3, 0, 3, 0));
        btnPlateNumber.addActionListener(this);
        editPlateNumber.add(txtPlateNumber2);
        editPlateNumber.add(btnPlateNumber);
        panelPlateNumber.add(lblPlateNumber2);
        panelPlateNumber.add(editPlateNumber);

        txtBrand.setEditable(false);
        txtBrand.setMargin(new Insets(3, 0, 3, 0));
        btnBrand.addActionListener(this);
        editBrand.add(txtBrand);
        editBrand.add(btnBrand);
        panelBrand.add(lblBrand);
        panelBrand.add(editBrand);

        txtModel.setEditable(false);
        txtModel.setMargin(new Insets(3, 0, 3, 0));
        btnModel.addActionListener(this);
        editModel.add(txtModel);
        editModel.add(btnModel);
        panelModel.add(lblModel);
        panelModel.add(editModel);

        txtType.setEditable(false);
        txtType.setMargin(new Insets(3, 0, 3, 0));
        btnType.addActionListener(this);
        editType.add(txtType);
        editType.add(btnType);
        panelType.add(lblType);
        panelType.add(editType);
```

```
panelLeftCol.add(panelPlateNumber);
panelLeftCol.add(panelBrand);
panelLeftCol.add(panelModel);
panelLeftCol.add(panelType);
panelLeftCol.setBorder(new EmptyBorder(0, 40, 0, 45));

// Make right column panel (panelRightCol)
txtColour.setEditable(false);
txtColour.setMargin(new Insets(3, 0, 3, 0));
btnColour.addActionListener(this);
editColour.add(txtColour);
editColour.add(btnColour);
panelColour.add(lblColour);
panelColour.add(editColour);

txtStatus.setEditable(false);
txtStatus.setMargin(new Insets(3, 0, 3, 0));
btnStatus.addActionListener(this);
editStatus.add(txtStatus);
editStatus.add(btnStatus);
panelStatus.add(lblStatus);
panelStatus.add(editStatus);

txtPrice.setEditable(false);
txtPrice.setMargin(new Insets(3, 0, 3, 0));
btnPrice.addActionListener(this);
editPrice.add(txtPrice);
editPrice.add(btnPrice);
panelPrice.add(lblPrice);
panelPrice.add(editPrice);

panelRightCol.add(panelColour);
panelRightCol.add(panelStatus);
panelRightCol.add(panelPrice);
panelRightCol.setBorder(new EmptyBorder(0, 40, 0, 45));

// Make content panel (panelContent)
panelContent.add(panelLeftCol);
panelContent.add(panelRightCol);

// Make buttons2 panel (panelButtons2)
btnDone.addActionListener(this);
panelButtons2.setPreferredSize(Main.HEADER_SIZE);
panelButtons2.add(btnDone);
```

```
        // Add sub-panels into DEL2 card
        panelEditCard2.add(panelFound, BorderLayout.NORTH);
        panelEditCard2.add(panelContent, BorderLayout.CENTER);
        panelEditCard2.add(panelButtons2, BorderLayout.SOUTH);

        // ----- Add both cards into card deck ----- //
        add(panelEditCard1, EDITCARD1);
        add(panelEditCard2, EDITCARD2);
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        CardLayout mainCardLayout = (CardLayout) Main.panelContainer.getLayout();
        CardLayout editCardLayout = (CardLayout) this.getLayout();

        Window window = (Window) this.getTopLevelAncestor(); // sauce: https://st
ackoverflow.com/questions/25346547/display-different-panel-size-in-cardlayout

        // Event Handling for CARD 1 (EDIT1 INPUT PANEL)
        if (source == btnBack) {
            // return to main menu
            txtPlateNumber1.setText("");
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        } else if (source == btnClear) {
            // clear plate number input
            txtPlateNumber1.setText("");
        } else if (source == txtPlateNumber1 || source == btnContinue) {
            String plateNumber = txtPlateNumber1.getText();
            int matchedIndex = 0;
            boolean isFound = false;

            // Check if there is a record with entered plateNumber
            for (int i = 0;i < carArrayList.size(); i++) {
                if (carArrayList.get(i).getPlateNumber().compareToIgnoreCase(plat
eNumber) == 0) {
                    matchedIndex = i;
                    isFound = true;
                    break;
                }
            }
```

```
            // Display error message if not found; else proceed to edit record sc
reen
            if (!isFound) {
                JOptionPane.showMessageDialog(null, "No car record with the enter
ed Plate Number was found.",
                        "Invalid Plate Number", JOptionPane.ERROR_MESSAGE);
            } else {
                // show edit record screen
                editCardLayout.show(this, EDITCARD2);
                window.setSize(600, 430);

                // populate the disabled text fields
                txtPlateNumber2.setText(carArrayList.get(matchedIndex).getPlateNu
mber());
                txtBrand.setText(carArrayList.get(matchedIndex).getBrand());
                txtModel.setText(carArrayList.get(matchedIndex).getModel());
                txtType.setText(carArrayList.get(matchedIndex).getType());
                txtColour.setText(carArrayList.get(matchedIndex).getColour());
                txtStatus.setText(carArrayList.get(matchedIndex).getStatus());
                txtPrice.setText(String.format("RM %.2f", carArrayList.get(matche
dIndex).getPrice()));

                // store the matched index for use in EDIT2 screen
                recordIndex = matchedIndex;
            }
        }

        // Event Handling for CARD 2 (EDIT2 PANEL)
        String input;
        if (source == btnPlateNumber) {
            // Get input for Plate Number
            input = getInput("Plate Number", 12);

            // Update Plate Number data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                boolean isUnique = true;

                // Check for duplicate
                for (int i = 0; i < carArrayList.size(); i++) {
                    if (carArrayList.get(i).getPlateNumber().compareToIgnoreCase(
input) == 0) {
                        isUnique = false;
                        break;
                    }
                }
```

```
                // Display error message if duplicate plate number exists; else u
pdate plate number
                if (!isUnique) {
                    JOptionPane.showMessageDialog(null, "The Plate Number you ent
ered already exists in the system.\n" +
                            "                                Please try again.",
                            "Duplicate Plate Number detected", JOptionPane.ERROR_
MESSAGE);
                } else {
                    carArrayList.get(recordIndex).setPlateNumber(input);
                    txtPlateNumber2.setText(input);
                }
            }
        } else if (source == btnBrand) {
            // Get input for Brand
            input = getInput("Brand", 16);

            // Update Brand data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                carArrayList.get(recordIndex).setBrand(input);
                txtBrand.setText(input);
            }
        } else if (source == btnModel) {
            // Get input for Model
            input = getInput("Model", 16);

            // Update Model data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                carArrayList.get(recordIndex).setModel(input);
                txtModel.setText(input);
            }
        } else if (source == btnType) {
            // Get input for Type
            input = getInput("Type", 16);

            // Update Type data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                carArrayList.get(recordIndex).setType(input);
                txtType.setText(input);
            }
        } else if (source == btnColour) {
            // Get input for Colour
            input = getInput("Colour", 16);
```

```
            // Update Colour data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                carArrayList.get(recordIndex).setColour(input);
                txtColour.setText(input);
            }
        } else if (source == btnStatus) {
            // Get input for Status
            input = getInput("Status", 16);

            // Update Status data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                carArrayList.get(recordIndex).setStatus(input);
                txtStatus.setText(input);
            }
        } else if (source == btnPrice) {
            boolean isNumeric = false;
            double price = 0;

            // Get input for Price
            do {
                input = getInput("Price", 9);

                if (input.equals(INPUT_CANCEL))
                    break;

                try {
                    price = Double.parseDouble(input);
                    isNumeric = true;
                } catch (NumberFormatException e) {
                    JOptionPane.showMessageDialog(null, "Please enter a numeric v
alue for Price.",
                            "Invalid input", JOptionPane.ERROR_MESSAGE);
                    isNumeric = false;
                }
            } while (!isNumeric);

            // Update Price data if user clicked "OK"
            if (!input.equals(INPUT_CANCEL)) {
                carArrayList.get(recordIndex).setPrice(price);
                txtPrice.setText(String.format("RM %.2f", price));
            }

        } else if (source == btnDone) {
            txtPlateNumber1.setText("");
            editCardLayout.show(this, EDITCARD1);
```

```
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        }

    }

    // Helper method for displaying input dialog and handling input to edit recor
d
    public String getInput(String attribute, int limit) {
        boolean hasError = false;
        String response;

        do {
            response = JOptionPane.showInputDialog(null, "Enter new " + attribute
,
                    "Edit " + attribute, JOptionPane.QUESTION_MESSAGE);
            if (response == null) {
                return INPUT_CANCEL;
            } else if (response.isBlank()) {
                JOptionPane.showMessageDialog(null, "Input cannot be empty.",
                        "Empty input detected", JOptionPane.OK_OPTION);
                hasError = true;
            } else if (response.trim().length() > limit) {
                JOptionPane.showMessageDialog(null, attribute + " input cannot be
 more than " + limit + " characters.",
                        "Input limit exceeded", JOptionPane.OK_OPTION);
                hasError = true;
            } else {
                hasError = false;
            }
        } while (hasError);

        return response.trim();
    }

}
```

- **SearchRecord.java**

```
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
```

```
public class SearchRecord extends JPanel implements ActionListener {

    // String identifiers for main deck cards associated with this function
    private final static String MAIN = "Main Menu Panel";
    // String identifiers for cards of this panel's card deck
    private final static String SEARCHCARD1 = "SEARCH1 Input Panel";
    private final static String SEARCHCARD2 = "SEARCH2 Record Info Panel";
    // Cards for this function
    private JPanel panelSearchCard1 = new JPanel(new BorderLayout());
    private JPanel panelSearchCard2 = new JPanel(new BorderLayout());

    // -------------------- SEARCHCARD1 PANEL COMPONENTS --------------------//
    private JPanel panelHeader = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 1
0));
    private JPanel panelTitle = new JPanel(new GridLayout(2, 1));
    private JPanel panelInput = new JPanel(new GridLayout(2, 1));
    private JPanel panelButtons1 = new JPanel(new FlowLayout(FlowLayout.CENTER, 7
5, 15));
    private JPanel panelMargin1 = new JPanel();
    private JPanel panelMargin2 = new JPanel();
    private JLabel lblTitle = new JLabel("SEARCH RECORD", JLabel.CENTER);
    private JLabel lblDescription = new JLabel("Enter the Plate Number of the car
 record to search.", JLabel.CENTER);
    private JLabel lblPlateNumber1 = new JLabel("Plate Number");
    private JTextField txtPlateNumber1 = new JTextField(10);
    private JButton btnBack = new JButton("< BACK");
    private JButton btnClear = new JButton("CLEAR");
    private JButton btnSearch = new JButton("SEARCH");
    private Dimension sizeBackBtn = new Dimension(80, 35);
    // -------------------- SEARCHCARD2 PANEL COMPONENTS --------------------//
    private JPanel panelFound = new JPanel(new FlowLayout(FlowLayout.LEFT, 40, 20
));
    private JPanel panelContent = new JPanel(new GridLayout(1, 2));
    private JPanel panelLeftCol = new JPanel(new GridLayout(4, 1));
    private JPanel panelRightCol = new JPanel(new GridLayout(4, 1));
    private JPanel panelPlateNumber = new JPanel(new GridLayout(2, 1));
    private JPanel panelBrand = new JPanel(new GridLayout(2, 1));
    private JPanel panelModel = new JPanel(new GridLayout(2, 1));
    private JPanel panelType = new JPanel(new GridLayout(2, 1));
    private JPanel panelColour = new JPanel(new GridLayout(2, 1));
    private JPanel panelStatus = new JPanel(new GridLayout(2, 1));
    private JPanel panelPrice = new JPanel(new GridLayout(2, 1));
    private JPanel panelButtons2 = new JPanel(new FlowLayout(FlowLayout.RIGHT, 45
, 20));
    private JLabel lblFound = new JLabel("MATCHING CAR RECORD FOUND");
```

```
    private JLabel lblPlateNumber2 = new JLabel("Plate Number");
    private JLabel lblBrand = new JLabel("Brand");
    private JLabel lblModel = new JLabel("Model");
    private JLabel lblType = new JLabel("Type");
    private JLabel lblColour = new JLabel("Colour");
    private JLabel lblStatus = new JLabel("Status");
    private JLabel lblPrice = new JLabel("Price");
    private JTextField txtPlateNumber2 = new JTextField(10);
    private JTextField txtBrand = new JTextField(10);
    private JTextField txtModel = new JTextField(10);
    private JTextField txtType = new JTextField(10);
    private JTextField txtColour = new JTextField(10);
    private JTextField txtStatus = new JTextField(10);
    private JTextField txtPrice = new JTextField(10);
    private JButton btnSearchAgain = new JButton("SEARCH AGAIN");
    private JButton btnBackToMenu = new JButton("BACK TO MENU");

    // Function variables
    private ArrayList<Car> carArrayList;

    public SearchRecord(ArrayList<Car> cars) {
        carArrayList = cars;
        makePanel();
    }

    public void makePanel() {
        setLayout(new CardLayout());

        // -------------------- CARD 1 (SEARCH1 INPUT PANEL) -------------------
- //
        // Make header bar (panelHeader)
        btnBack.setPreferredSize(sizeBackBtn);
        btnBack.addActionListener(this);
        lblTitle.setFont(new Font("Helvetica", Font.BOLD, 16));
        lblTitle.setForeground(Color.WHITE);
        lblDescription.setFont(new Font("Helvetica", Font.PLAIN, 11));
        lblDescription.setForeground(Color.WHITE);
        panelTitle.setBackground(Color.DARK_GRAY);
        panelTitle.add(lblTitle);
        panelTitle.add(lblDescription);
        panelHeader.setBackground(Color.DARK_GRAY);
        panelHeader.add(btnBack);
        panelHeader.add(panelTitle);

        // Make margin panels (panelMargin#)
```

```
        panelMargin1.setPreferredSize(new Dimension(120, 0));
        panelMargin2.setPreferredSize(new Dimension(120, 0));

        // Make input panel (panelInput)
        txtPlateNumber1.addActionListener(this);
        panelInput.add(lblPlateNumber1);
        panelInput.add(txtPlateNumber1);

        // Make buttons panel (panelButtons)
        btnClear.setPreferredSize(btnSearch.getPreferredSize());
        btnClear.addActionListener(this);
        btnSearch.addActionListener(this);
        panelButtons1.setPreferredSize(new Dimension(0, 70));
        panelButtons1.add(btnClear);
        panelButtons1.add(btnSearch);

        // Add sub-panels into SEARCH1 card
        panelSearchCard1.add(panelHeader, BorderLayout.NORTH);
        panelSearchCard1.add(panelMargin1, BorderLayout.WEST);
        panelSearchCard1.add(panelInput, BorderLayout.CENTER);
        panelSearchCard1.add(panelMargin2, BorderLayout.EAST);
        panelSearchCard1.add(panelButtons1, BorderLayout.SOUTH);

        // -------------------- CARD 2 (SEARCH2 RECORD INFO PANEL) --------------
------ //
        // Make title panel (panelFound)
        lblFound.setFont(new Font("Helvetica", Font.BOLD, 16));
        panelFound.setPreferredSize(Main.HEADER_SIZE);
        panelFound.add(lblFound);

        // Make left column panel (panelLeftCol)
        txtPlateNumber2.setEditable(false);
        txtBrand.setEditable(false);
        txtModel.setEditable(false);
        txtType.setEditable(false);
        panelPlateNumber.add(lblPlateNumber2);
        panelPlateNumber.add(txtPlateNumber2);
        panelBrand.add(lblBrand);
        panelBrand.add(txtBrand);
        panelModel.add(lblModel);
        panelModel.add(txtModel);
        panelType.add(lblType);
        panelType.add(txtType);
        panelLeftCol.add(panelPlateNumber);
        panelLeftCol.add(panelBrand);
```

```java
        panelLeftCol.add(panelModel);
        panelLeftCol.add(panelType);
        panelLeftCol.setBorder(new EmptyBorder(0, 40, 0, 45));

        // Make right column panel (panelRightCol)
        txtColour.setEditable(false);
        txtStatus.setEditable(false);
        txtPrice.setEditable(false);
        panelColour.add(lblColour);
        panelColour.add(txtColour);
        panelStatus.add(lblStatus);
        panelStatus.add(txtStatus);
        panelPrice.add(lblPrice);
        panelPrice.add(txtPrice);
        panelRightCol.add(panelColour);
        panelRightCol.add(panelStatus);
        panelRightCol.add(panelPrice);
        panelRightCol.setBorder(new EmptyBorder(0, 40, 0, 45));

        // Make content panel (panelContent)
        panelContent.add(panelLeftCol);
        panelContent.add(panelRightCol);

        // Make buttons2 panel (panelButtons2)
        btnSearchAgain.addActionListener(this);
        btnBackToMenu.addActionListener(this);
        panelButtons2.setPreferredSize(Main.HEADER_SIZE);
        panelButtons2.add(btnSearchAgain);
        panelButtons2.add(btnBackToMenu);

        // Add sub-panels into SEARCH2 card
        panelSearchCard2.add(panelFound, BorderLayout.NORTH);
        panelSearchCard2.add(panelContent, BorderLayout.CENTER);
        panelSearchCard2.add(panelButtons2, BorderLayout.SOUTH);

        // ----- Add both cards into card deck ----- //
        add(panelSearchCard1, SEARCHCARD1);
        add(panelSearchCard2, SEARCHCARD2);
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        CardLayout mainCardLayout = (CardLayout) Main.panelContainer.getLayout();
        CardLayout searchCardLayout = (CardLayout) this.getLayout();
```

```
        Window window = (Window) this.getTopLevelAncestor();

        // Event Handling for CARD 1 (SEARCH1 INPUT PANEL)
        if (source == btnBack) {
            txtPlateNumber1.setText("");
            // return to main menu
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        } else if (source == btnClear) {
            // clear plate number input
            txtPlateNumber1.setText("");
        } else if (source == txtPlateNumber1 || source == btnSearch) {
            String plateNumber = txtPlateNumber1.getText();
            int matchedIndex = 0;
            boolean isFound = false;

            // Check if there is a record with entered plateNumber
            for (int i = 0; i < carArrayList.size(); i++) {
                if (carArrayList.get(i).getPlateNumber().compareToIgnoreCase(plat
eNumber) == 0) {
                    matchedIndex = i;
                    isFound = true;
                    break;
                }
            }

            // Display error message if not found; else proceed to next screen
            if (!isFound) {
                JOptionPane.showMessageDialog(null, "No car record with the enter
ed Plate Number was found.",
                        "Invalid Plate Number", JOptionPane.OK_OPTION);
            } else {
                // show next screen
                searchCardLayout.show(this, SEARCHCARD2);
                window.setSize(600, 400);

                // populate the disabled text fields
                txtPlateNumber2.setText(carArrayList.get(matchedIndex).getPlateNu
mber());
                txtBrand.setText(carArrayList.get(matchedIndex).getBrand());
                txtModel.setText(carArrayList.get(matchedIndex).getModel());
                txtType.setText(carArrayList.get(matchedIndex).getType());
                txtColour.setText(carArrayList.get(matchedIndex).getColour());
                txtStatus.setText(carArrayList.get(matchedIndex).getStatus());
```

```
                txtPrice.setText(String.format("RM %.2f", carArrayList.get(matche
dIndex).getPrice()));
            }
        }

        // Event Handling for CARD 2 (SEARCH2 RECORD INFO PANEL)
        if (source == btnSearchAgain) {
            txtPlateNumber1.setText("");
            searchCardLayout.show(this, SEARCHCARD1);
            window.setSize(500, 250);
        } else if (source == btnBackToMenu) {
            txtPlateNumber1.setText("");
            searchCardLayout.show(this, SEARCHCARD1);
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        }
    }

}
```

- **DisplayRecords.java**

```
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.border.EmptyBorder;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;

public class DisplayRecords extends JPanel implements ActionListener, KeyListener
 {

    // String identifiers for main deck cards associated with this panel
    private static final String MAIN = "Main Menu Panel";

    // Column names to be inserted in table model
    private final String[] columnNames = {"No.", "Plate Number", "Brand", "Model"
, "Type", "Colour", "Status", "Price (RM)"};

    // -------------------- "DISPLAY RECORDS" PANEL COMPONENTS ------------------
--//
    private JPanel panelHeader = new JPanel(new FlowLayout(FlowLayout.LEFT, 20, 1
0));
    private JPanel panelTitle = new JPanel(new GridLayout(2, 1));
    private JPanel panelContent = new JPanel(new BorderLayout());
```

```
    private JPanel panelSearchBar = new JPanel(new FlowLayout(FlowLayout.RIGHT, 5
, 10));
    private JPanel panelMargin1 = new JPanel();
    private JPanel panelMargin2 = new JPanel();
    private JPanel panelMargin3 = new JPanel();
    private JTextField searchBox = new JTextField(10);
    private DefaultTableModel tableModel = new DefaultTableModel(columnNames, 0)
{
        @Override // sets all cells to be not editable
        public boolean isCellEditable(int row, int column) {
            return false;
        }
    };
    private JTable recordTable = new JTable(tableModel);
    private JScrollPane scrollTable = new JScrollPane(recordTable, JScrollPane.VE
RTICAL_SCROLLBAR_AS_NEEDED, JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);
    private JLabel lblTitle = new JLabel("DISPLAY RECORDS", JLabel.CENTER);
    private JLabel lblDescription = new JLabel("Search for specific keyword using
 the search box to filter table data.", JLabel.CENTER);
    private JLabel lblSearch = new JLabel("SEARCH:");
    private JButton btnBack = new JButton("< BACK");
    private JButton btnClear = new JButton("CLEAR");
    private Dimension sizeBackBtn = new Dimension(80, 35);

    // Function variables
    private ArrayList<Car> carArrayList;

    public DisplayRecords(ArrayList<Car> cars) {
        carArrayList = cars;
        makePanel();
    }

    public void makePanel() {
        setLayout(new BorderLayout());

        // Make header panel (panelHeader)
        btnBack.setPreferredSize(sizeBackBtn);
        btnBack.addActionListener(this);
        lblTitle.setFont(new Font("Helvetica", Font.BOLD, 16));
        lblTitle.setForeground(Color.WHITE);
        lblDescription.setFont(new Font("Helvetica", Font.PLAIN, 11));
        lblDescription.setForeground(Color.WHITE);
        panelTitle.setBorder(new EmptyBorder(0, 80, 0, 0));
        panelTitle.setBackground(Color.DARK_GRAY);
        panelTitle.add(lblTitle);
```

```
        panelTitle.add(lblDescription);
        panelHeader.setBackground(Color.DARK_GRAY);
        panelHeader.add(btnBack);
        panelHeader.add(panelTitle);

        // Make search bar (panelSearchBar)
        searchBox.setPreferredSize(new Dimension(150, 27));
        searchBox.addKeyListener(this);
        btnClear.addActionListener(this);
        panelSearchBar.setBorder(new EmptyBorder(0, 10, 0, 0));
        panelSearchBar.add(lblSearch);
        panelSearchBar.add(searchBox);
        panelSearchBar.add(btnClear);

        // Make scrollable table (scrollTable)
        recordTable.getColumnModel().getColumn(0).setPreferredWidth(8); // Number
ing col.
        recordTable.getColumnModel().getColumn(1).setPreferredWidth(100); // Plat
e Number col.
        recordTable.getTableHeader().setReorderingAllowed(false);

        // Add search bar and scroll pane
        panelContent.add(panelSearchBar, BorderLayout.NORTH);
        panelContent.add(scrollTable, BorderLayout.CENTER);

        // Make margin panels (panelMargin#)
        panelMargin1.setPreferredSize(new Dimension(40, 0));
        panelMargin2.setPreferredSize(new Dimension(40, 0));
        panelMargin3.setPreferredSize(new Dimension(0, 40));

        // Add sub-panels into parent panel (this)
        add(panelHeader, BorderLayout.NORTH);
        add(panelContent, BorderLayout.CENTER);
        add(panelMargin1, BorderLayout.EAST);
        add(panelMargin2, BorderLayout.WEST);
        add(panelMargin3, BorderLayout.SOUTH);
    }

    @Override
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();

        CardLayout mainCardLayout = (CardLayout) Main.panelContainer.getLayout();
        Window window = (Window) this.getTopLevelAncestor();
```

```java
        if (source == btnBack) {
            searchBox.setText("");
            // Back to main menu
            mainCardLayout.show(Main.panelContainer, MAIN);
            window.setSize(Main.WIDTH, Main.HEIGHT);
        } else if (source == btnClear) {
            // Clear search box
            searchBox.setText("");
            updateTable(false);
        }
    }

    @Override
    public void keyPressed(KeyEvent keyEvent) {/* not used */}

    @Override
    public void keyTyped(KeyEvent keyEvent) {/* not used */}

    @Override
    public void keyReleased(KeyEvent keyEvent) { // For filtering table data
        Object source = keyEvent.getSource();

        if (source == searchBox)
            updateTable(true);
    }

    // Keeps data in the table up-to-date
    public void updateTable(boolean withKeyword) {
        // Clear all rows
        tableModel.setRowCount(0);

        // Determine source of function call
        if (!withKeyword) {
            // Get data from every car object to be stored in rows
            for (int i = 0; i < carArrayList.size(); i++) {
                Object[] data = {
                        i+1,
                        carArrayList.get(i).getPlateNumber(),
                        carArrayList.get(i).getBrand(),
                        carArrayList.get(i).getModel(),
                        carArrayList.get(i).getType(),
                        carArrayList.get(i).getColour(),
                        carArrayList.get(i).getStatus(),
                        String.format("%.2f", carArrayList.get(i).getPrice())
                };
```

```
                tableModel.addRow(data);
            }
        } else {
            // Search for keyword
            String keyword = searchBox.getText().toUpperCase();

            for (int i = 0; i < carArrayList.size(); i++) {
                // If any column of a record contains the keyword
                if (carArrayList.get(i).getPlateNumber().toUpperCase().contains(k
eyword) ||
                    carArrayList.get(i).getBrand().toUpperCase().contains(keyword
) ||
                    carArrayList.get(i).getModel().toUpperCase().contains(keyword
) ||
                    carArrayList.get(i).getType().toUpperCase().contains(keyword)
 ||
                    carArrayList.get(i).getColour().toUpperCase().contains(keywor
d) ||
                    carArrayList.get(i).getStatus().toUpperCase().contains(keywor
d) ||
                    String.valueOf(carArrayList.get(i).getPrice()).toUpperCase().
contains(keyword)) {

                    Object[] data = {
                            i+1,
                            carArrayList.get(i).getPlateNumber(),
                            carArrayList.get(i).getBrand(),
                            carArrayList.get(i).getModel(),
                            carArrayList.get(i).getType(),
                            carArrayList.get(i).getColour(),
                            carArrayList.get(i).getStatus(),
                            String.format("%.2f", carArrayList.get(i).getPrice())
                    };

                    tableModel.addRow(data);
                }
            }
        }
    }

    // For checking if table is empty
    public boolean tableIsEmpty() {
        if (tableModel.getRowCount() == 0)
            return true;
```

```
        else
            return false;
    }

}
```

DESCRIPTION OF THE PROGRAM

- **Login screen**

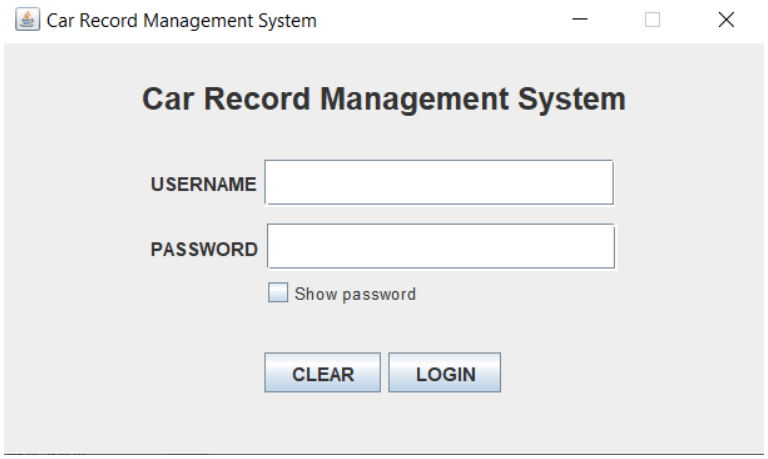


Figure 1.0: System login screen

When the system starts, it will display a login screen. Users will need to enter the correct details to gain access, which is **HelloWorld** for username and **123** for password. If the user leaves the input fields empty, the system will display an error message to let them know they need to fill them. If the user enters the incorrect login details, the system displays an error message that says their username or password is invalid.

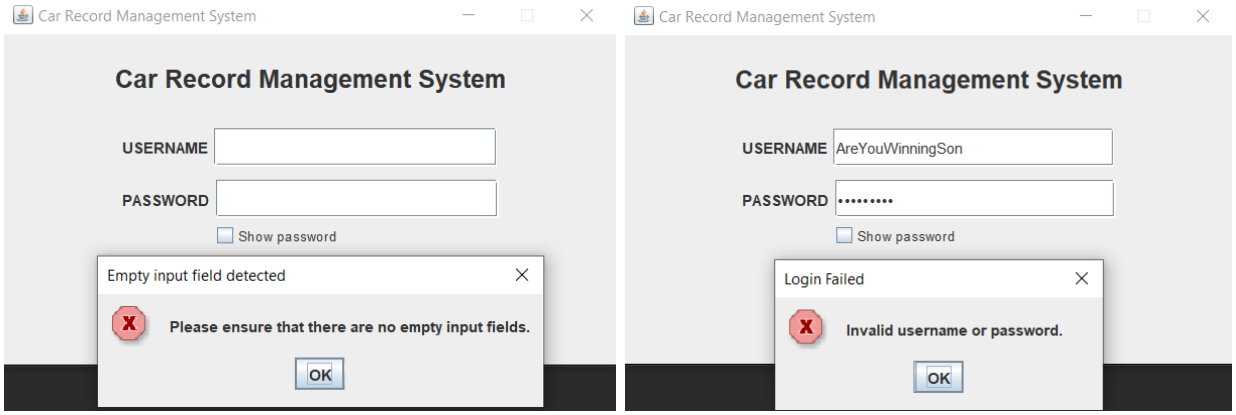


Figure 1.1: Error messages for invalid inputs

The ‘Show password’ checkbox in the login screen allows the user to reveal their password in plain text. Giving people the option to view their password allows them to easily check if they’ve correctly typed what they intended to type. It also allows users to type their password quickly and accurately while also reducing the chances of the user encountering an error due to mistyping something.

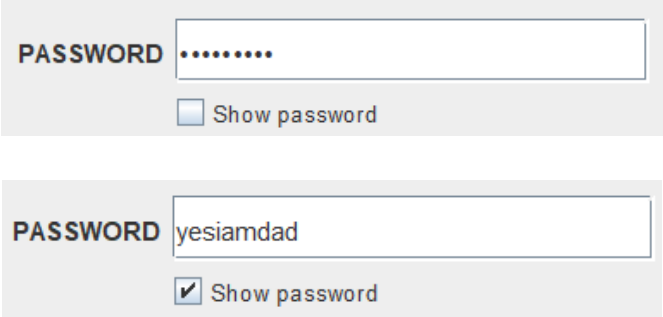


Figure 1.2: Usage of ‘Show password’

If the user enters the correct login details, the system notifies that they have successfully logged in and redirects them to the system's main menu.

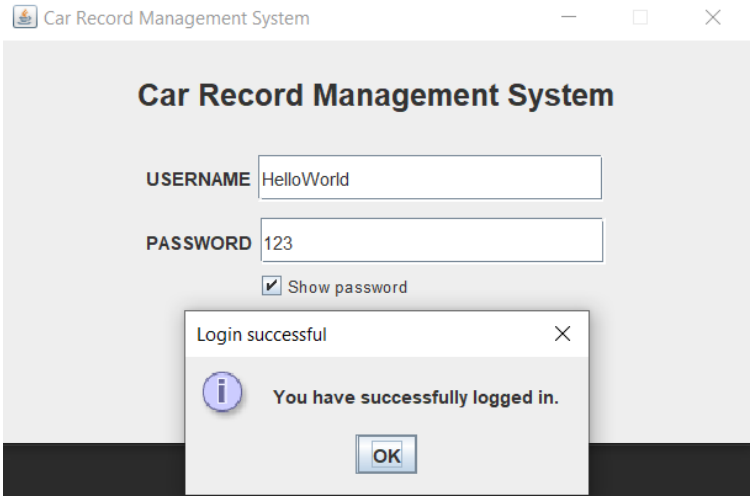


Figure 1.3: System notifies login successful

- **Main Menu**

The main menu of the system consists of 5 buttons that users can select from. Each of the buttons allows the user to access the main functions of the system, which include “Create Record”, “Delete Record”, “Edit Record”, “Search Record”, and “Display All Records”.

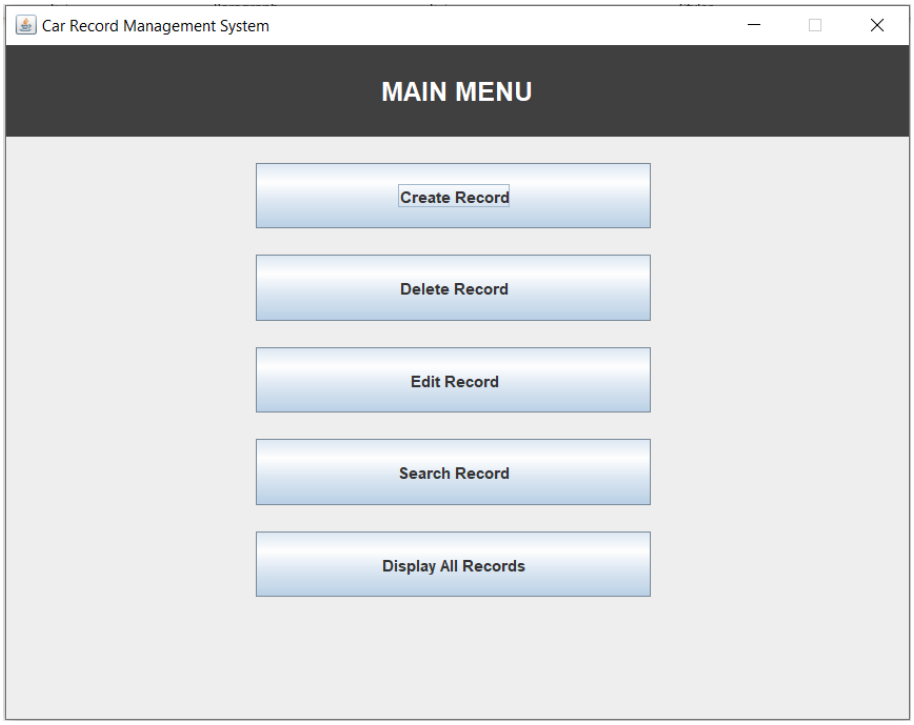


Figure 2.0: Main menu of the system

- **Create Record**

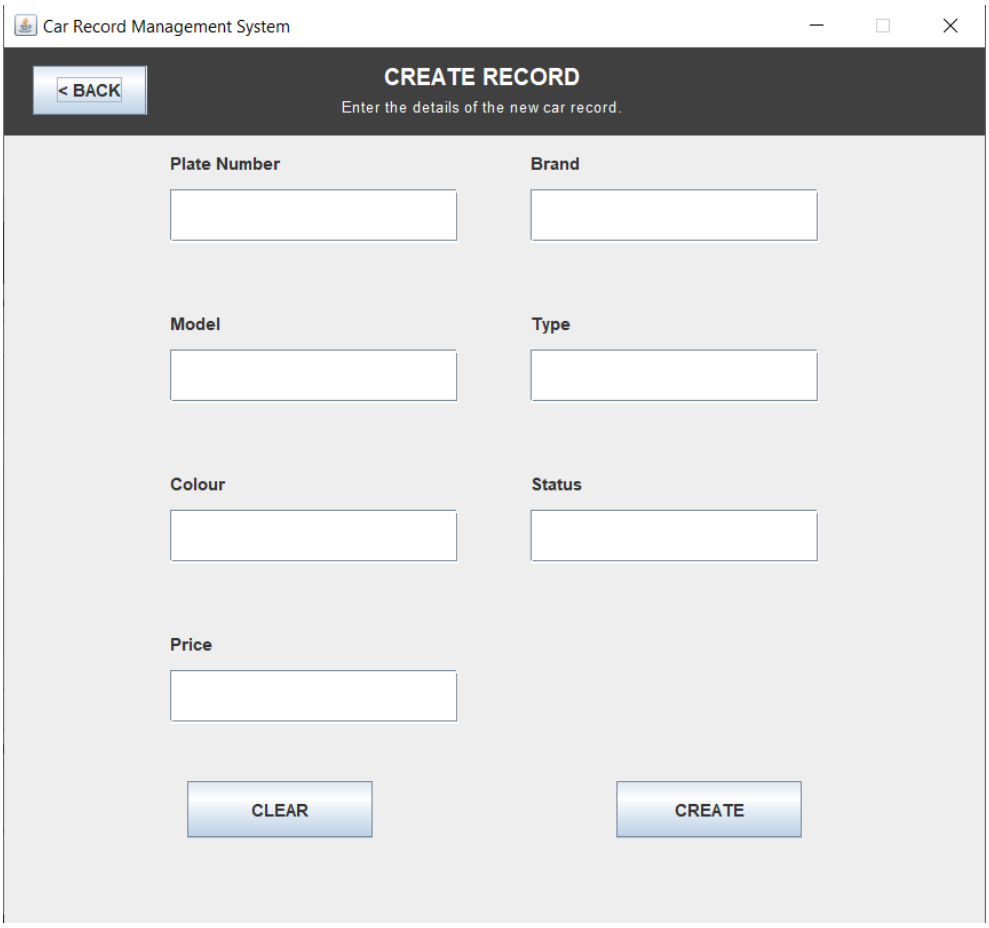


Figure 3.0: Create Record screen

When the 'Create Record' button in the main menu is clicked, the window content changes, and the 'Create Record' form is displayed. The 'Create Record' function allows users to create new car records by entering their details in the input fields and adding them into the system.

There are some constraints when users input values. First, all of the input fields must have a value and cannot be left empty. Next, the new car record's plate number must be unique to differentiate other records currently in the system. The new record's price must also be floating numbers and must not contain any non-integer characters. Each input field must comply with set character limits: At most, 12 characters are allowed for plate numbers, 9 characters for prices, and 16 characters for other input fields.

Corresponsive error messages are displayed instantaneously under the input fields with invalid input while the users enter their input. Certain error messages only show when the system performs thorough checking, such as checking the uniqueness of plate numbers and the data type of the value of the price field, which is performed when the user clicks on the 'CREATE' button.

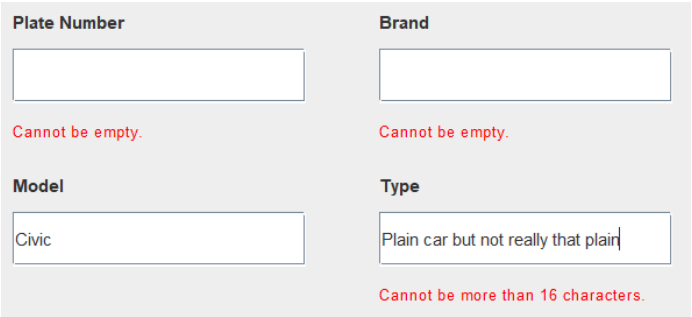


Figure 3.1: Examples of error messages shown under input fields

If the user tries to submit the form when it still contains errors, the system will deny the submission and notify the user to recheck their inputs.

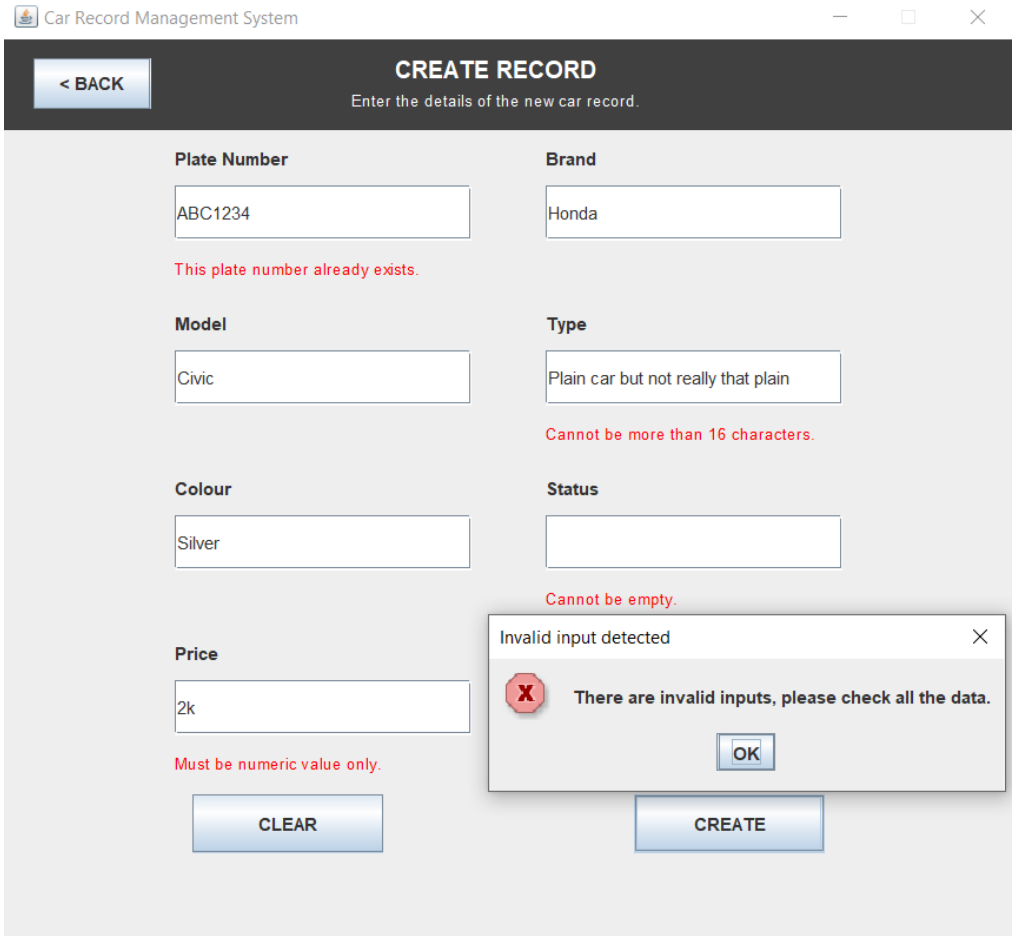


Figure 3.2: System denies submission if there are invalid input fields

If there are no errors, the system confirms with the user if they want to create the new car record. If the user clicks 'Yes', the system tells the user that the record has been created successfully and added to the system. If the user clicks 'No', the system will close the confirmation dialog and let the user recheck their inputs.

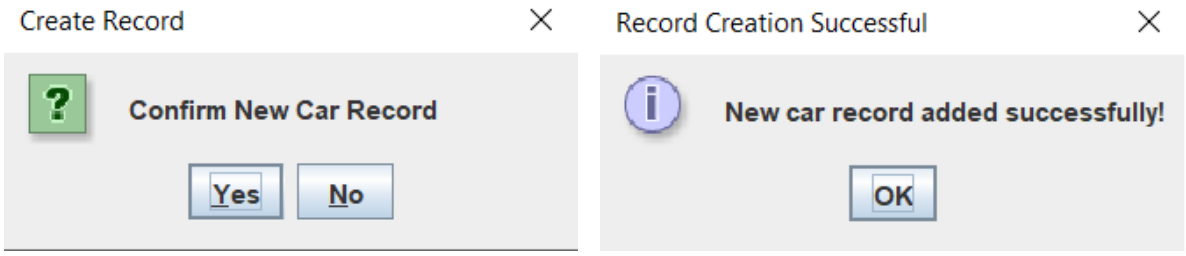
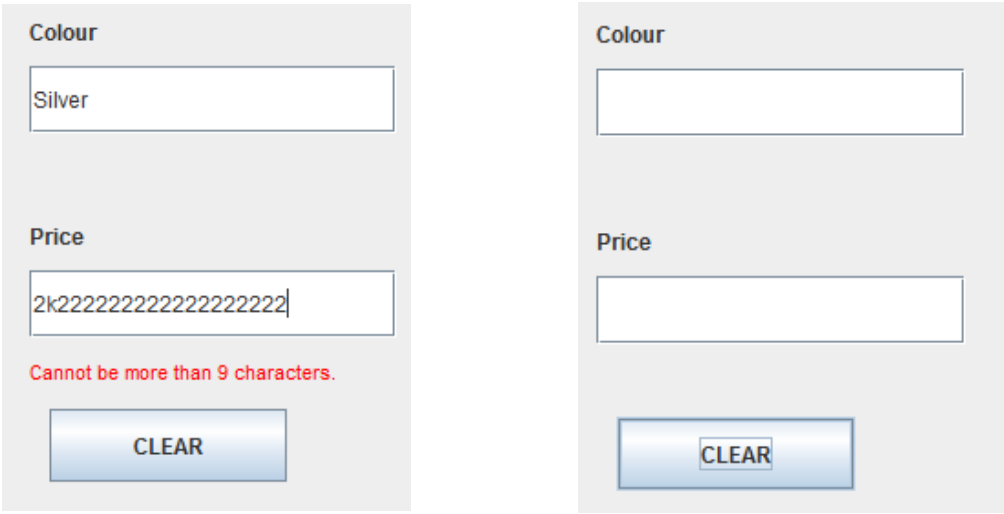


Figure 3.3: Confirmation dialog for creating a record **(left)** and system output if user clicks 'Yes' **(right)**

There are also 'Back' and 'Clear' buttons. The 'Clear' button is used to quickly empty all inputs fields in the form, while the 'Back' button redirects the user back to the main menu. When the 'Back' button is pressed, the system confirms with the user if they want to go back to the main menu and tells them that all data that they have entered will be lost if they do. If they want to proceed, the system discards the data entered and returns the user to the main menu, or else the system will cancel the redirecting process.



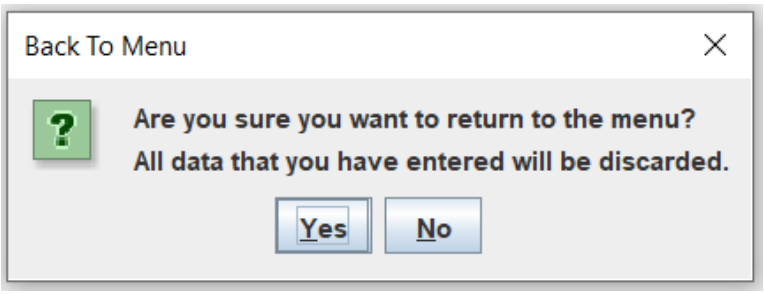


Figure 3.4: Usage of 'Clear' button **(top left and right)** and confirmation dialog that appears when users click the 'Back' button **(bottom)**

- **Delete Record**

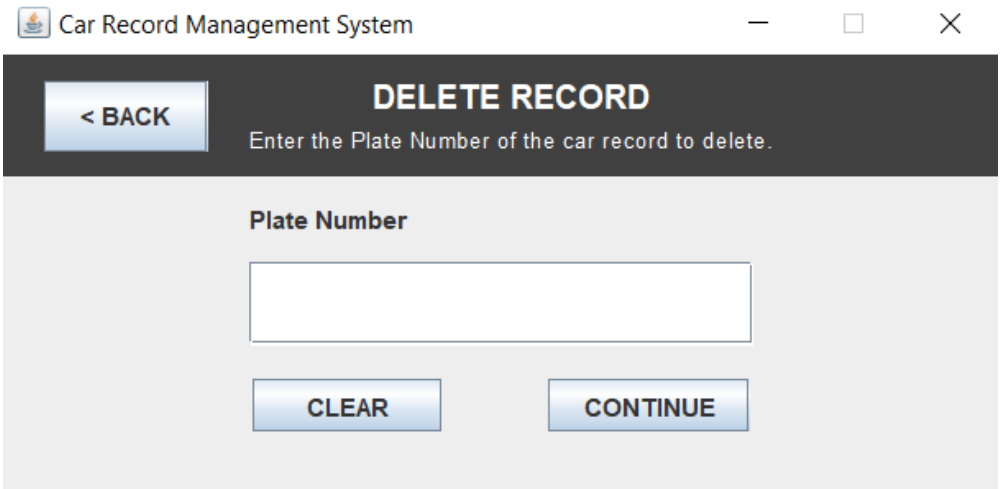


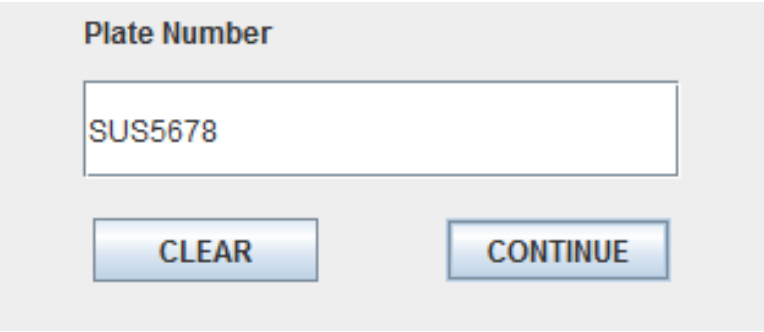
Figure 4.0: Delete Record screen

When the 'Delete Record' button in the main menu is clicked, the window content changes, and the 'Delete Record' screen is displayed. The 'Delete Record' function allows users to delete existing car records in the system that are identified using their plate number.

If the user enters a plate number that does not exist, the system displays an error message saying that no car record with the entered plate number is found within the system.

Figure 4.1: Error message for searching a non-existent plate number

Otherwise, the system displays the details of the car record with the matching plate number that the user entered.



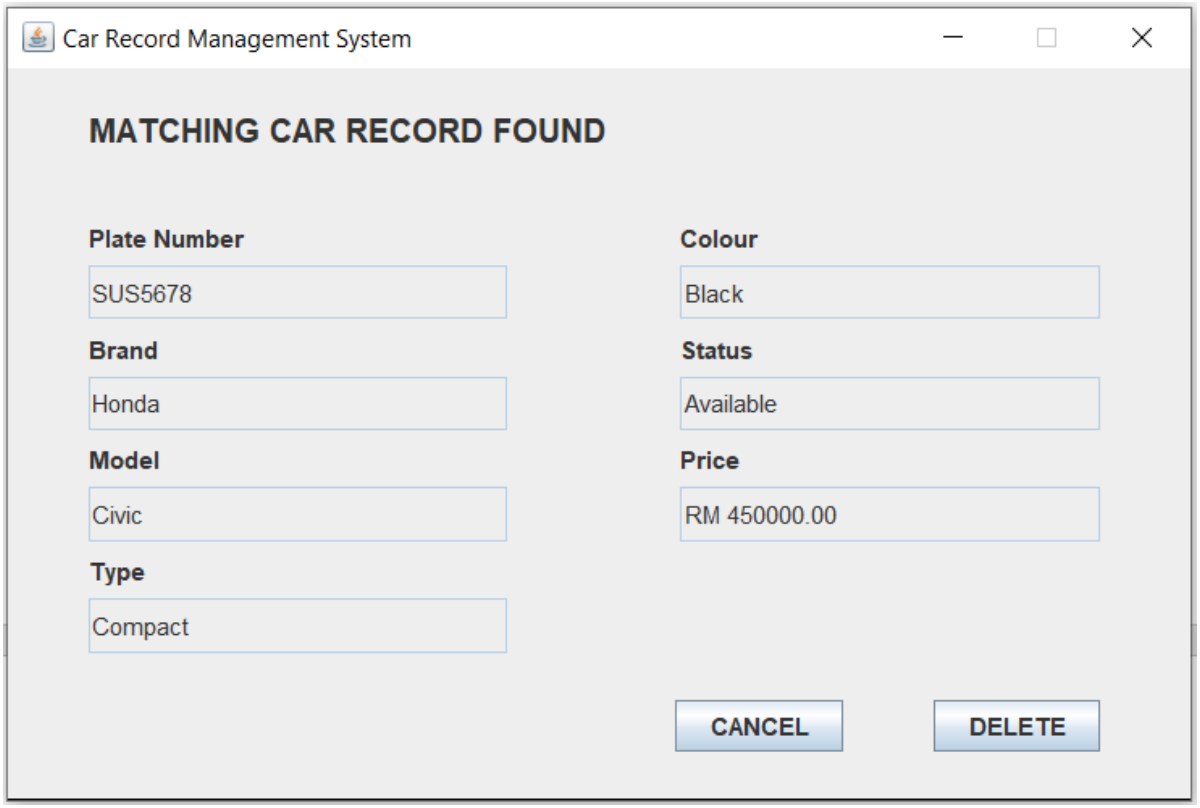


Figure 4.2: Screen output for searching a valid plate number in 'Delete Record'

When the user clicks the 'Delete' button, the system confirms with the user if they want to delete the record. If the user clicks 'Yes', the system notifies the user that the record with the entered plate number has been deleted successfully, and the user is returned to the main menu. Otherwise, the delete process is denied, and the user is allowed to check the details of the car record again.

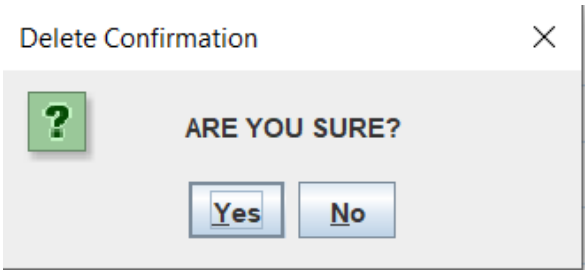


Figure 4.3: Confirmation dialog for deleting a record

Figure 4.4: System output if user clicks 'Yes'

If the user presses the 'Cancel' button, the system displays a message saying that the deletion process is cancelled, and the user is returned to the main menu.

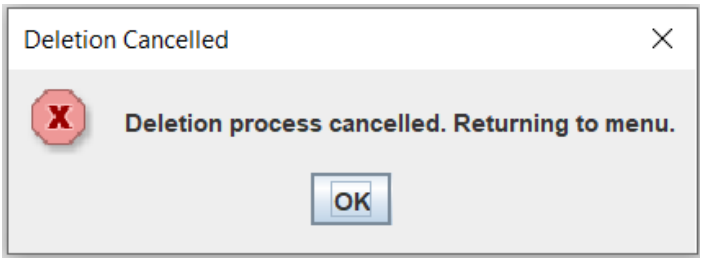


Figure 4.5: System output if user clicks 'No'

- **Edit Record**

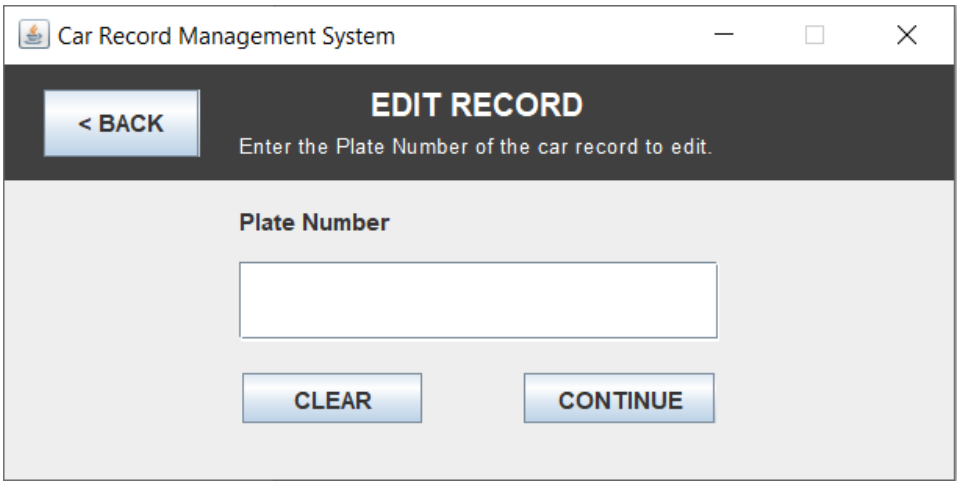


Figure 5.0: Edit Record screen

When the 'Edit Record' button in the main menu is clicked, the window content changes, and the 'Edit Record' screen is displayed. The 'Edit Record' function allows users to edit details of existing car records in the system that are identified using their plate number.

If the user enters a plate number that does not exist, the system displays an error message notifying that no car record with the entered plate number is.

Figure 5.1: Error message for searching a non-existent plate number

Otherwise, the system displays the details of the car record with the matching plate number that the user entered.

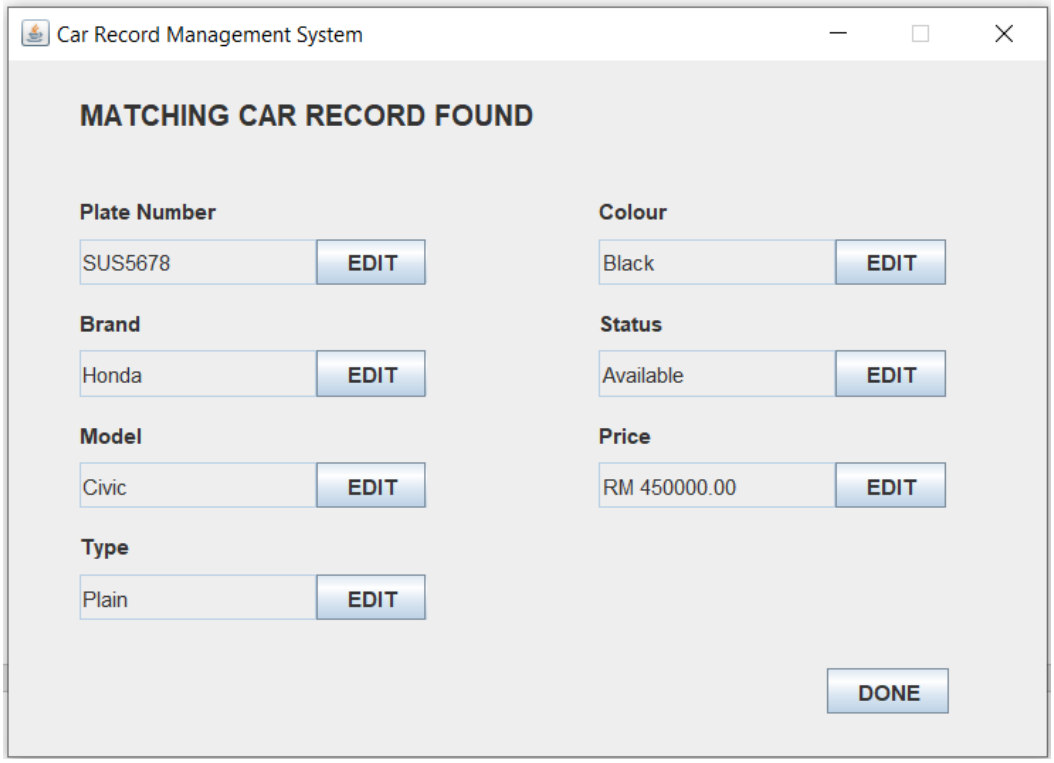


Figure 5.2: Screen output for searching a valid plate number in 'Edit Record'

On the screen, as seen in the image above, there are 'EDIT' buttons next to every car record detail. When the user presses any of the 'EDIT' buttons, the system displays an input dialog and prompts the user to enter new data for the selected record detail. For example, when the user clicks on 'EDIT' in the 'Price' field, the system prompts the user to enter new price data.

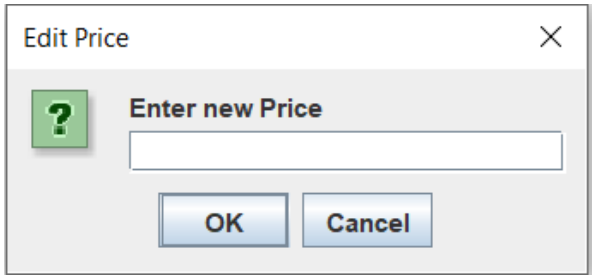


Figure 5.3: System prompts user for the new car price

After entering new data, the user may press the 'OK' button to confirm and apply the data change. Then, the system will update the data within the edited data field. Otherwise, the user may press 'Cancel' to cancel the editing process and close the input dialog.

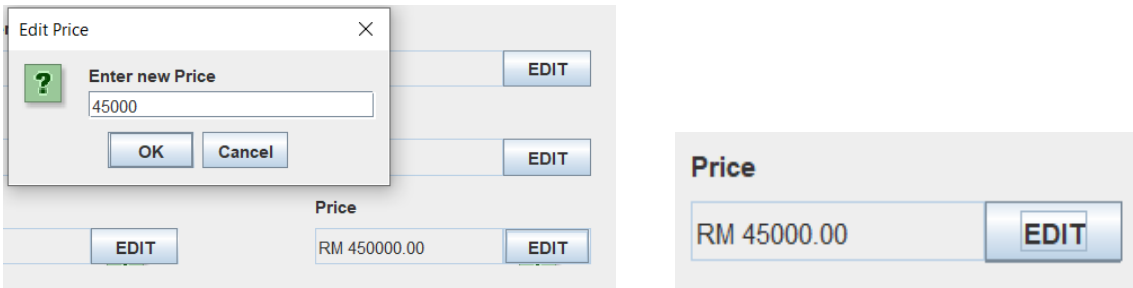


Figure 5.4: System instantly updates new values on the screen

When receiving new data, the system will also check for invalid input, prevent invalid input from changing the current data, and display appropriate error messages to notify the user of the specific errors. If there are errors, the user is prompted to enter another input.

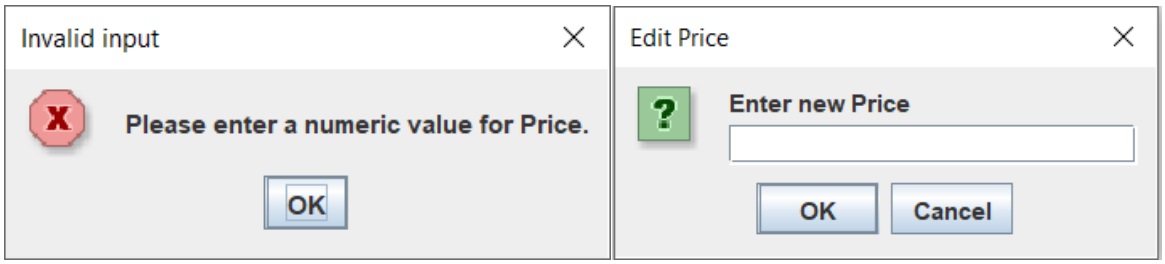


Figure 5.5: Example of error message for invalid price value **(left)**;
System prompts user for new price value again **(right)**

- **Search Record**

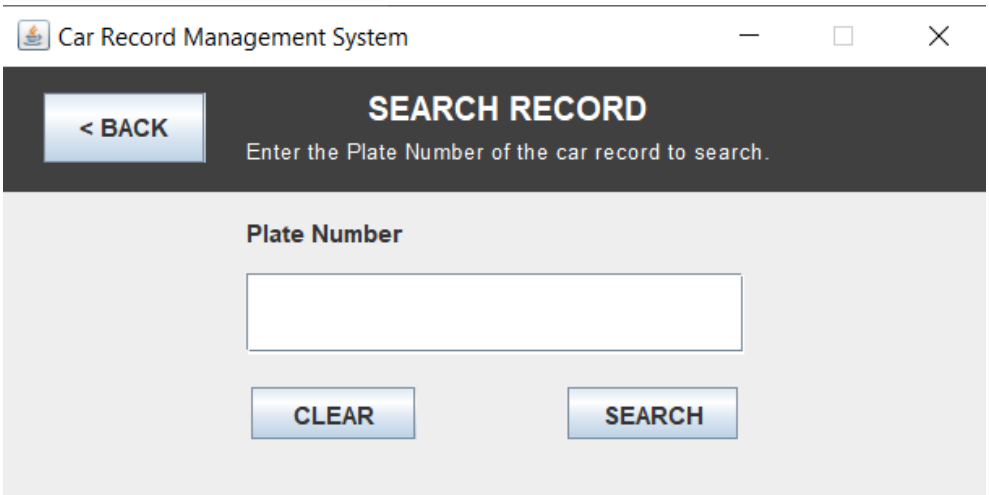


Figure 6.0: Search Record screen

When the 'Search Record' button in the main menu is clicked, the window content changes, and the 'Search Record' screen is displayed. The 'Search Record' function allows users to search and view the details of existing car records in the system that are identified using their plate number.

If the user enters a plate number that does not exist, the system displays an error message saying that no car record with the entered plate number is found.

Figure 6.1: Error message for searching a non-existent plate number

Otherwise, the system displays the details of the car record with the matching plate number that the user entered.

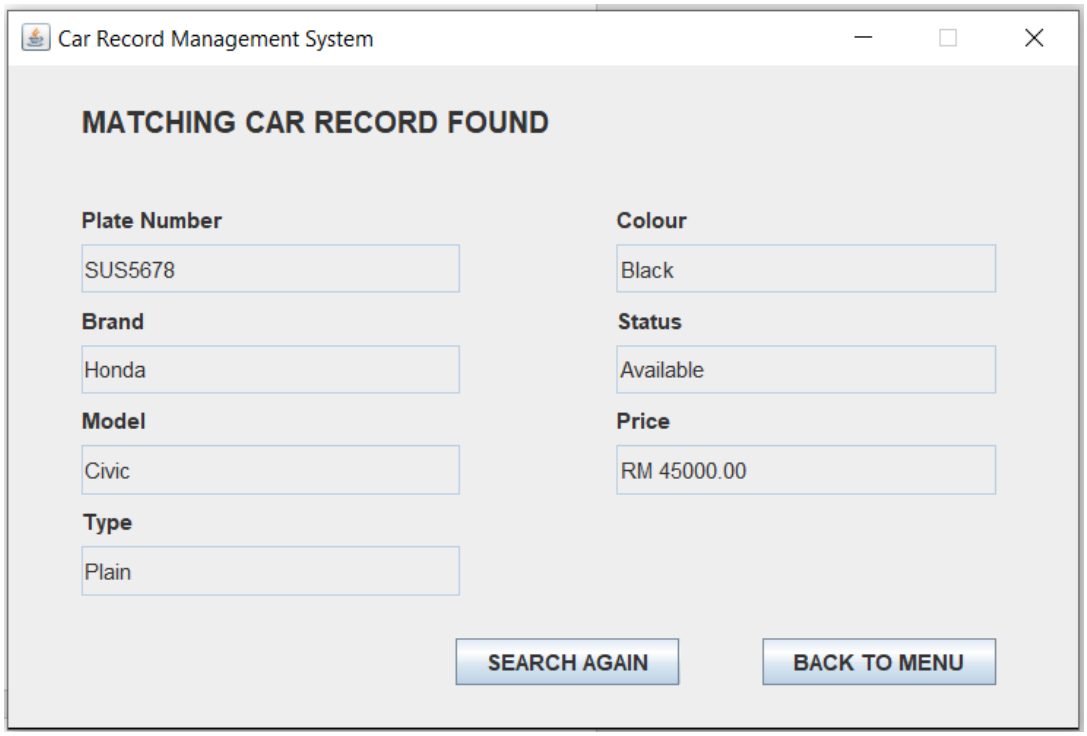
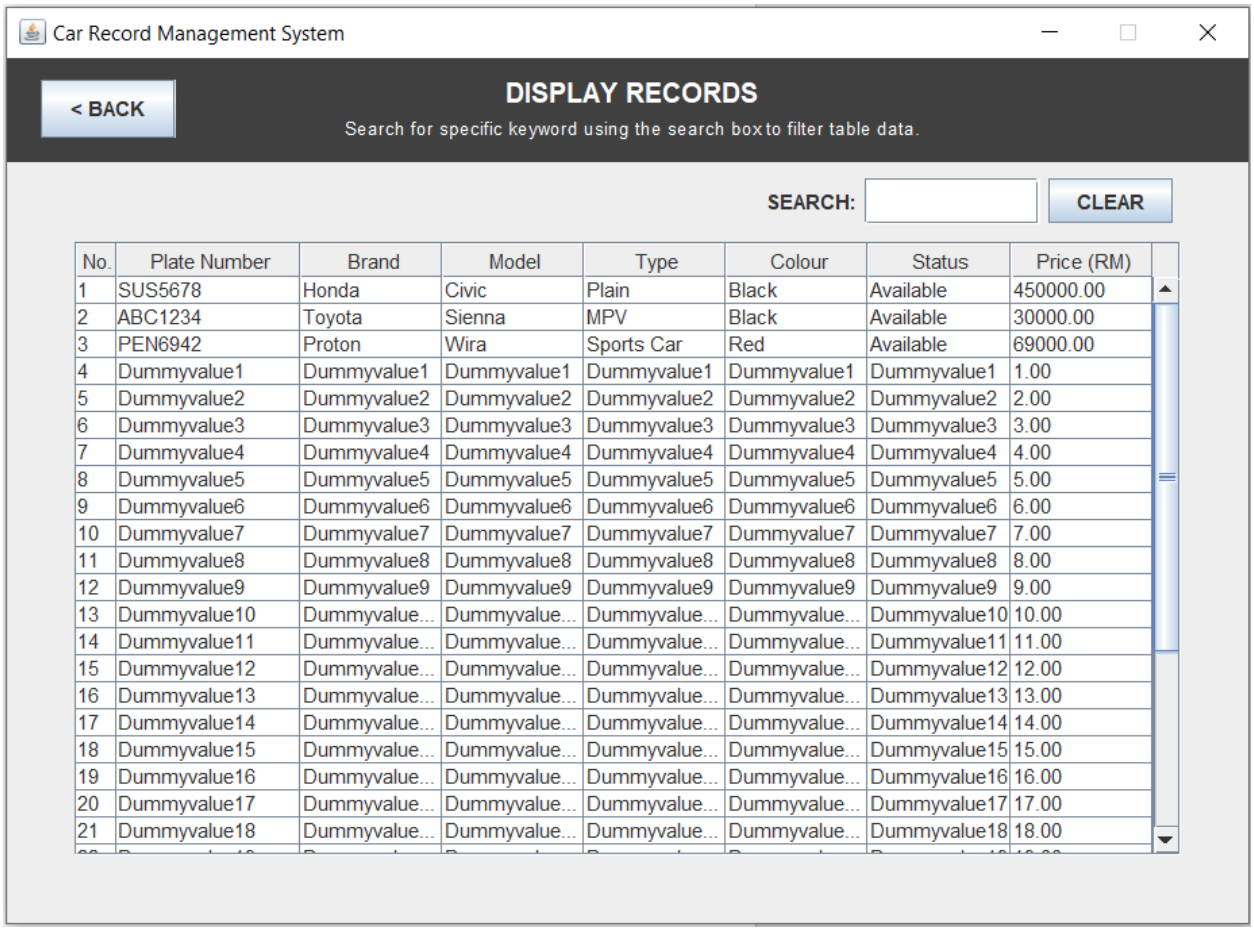


Figure 6.2: Screen output for searching a valid plate number in 'Search Record'

If the user presses the 'SEARCH AGAIN' button, the system will redirect the user to the previous page, where they are prompted to search for a plate number again. If the user presses the 'BACK TO MENU' button, the system returns the user to the main menu.

- **Display Records**



No.	Plate Number	Brand	Model	Type	Colour	Status	Price (RM)
1	SUS5678	Honda	Civic	Plain	Black	Available	450000.00
2	ABC1234	Toyota	Sienna	MPV	Black	Available	30000.00
3	PEN6942	Proton	Wira	Sports Car	Red	Available	69000.00
4	Dummyvalue1	Dummyvalue1	Dummyvalue1	Dummyvalue1	Dummyvalue1	Dummyvalue1	1.00
5	Dummyvalue2	Dummyvalue2	Dummyvalue2	Dummyvalue2	Dummyvalue2	Dummyvalue2	2.00
6	Dummyvalue3	Dummyvalue3	Dummyvalue3	Dummyvalue3	Dummyvalue3	Dummyvalue3	3.00
7	Dummyvalue4	Dummyvalue4	Dummyvalue4	Dummyvalue4	Dummyvalue4	Dummyvalue4	4.00
8	Dummyvalue5	Dummyvalue5	Dummyvalue5	Dummyvalue5	Dummyvalue5	Dummyvalue5	5.00
9	Dummyvalue6	Dummyvalue6	Dummyvalue6	Dummyvalue6	Dummyvalue6	Dummyvalue6	6.00
10	Dummyvalue7	Dummyvalue7	Dummyvalue7	Dummyvalue7	Dummyvalue7	Dummyvalue7	7.00
11	Dummyvalue8	Dummyvalue8	Dummyvalue8	Dummyvalue8	Dummyvalue8	Dummyvalue8	8.00
12	Dummyvalue9	Dummyvalue9	Dummyvalue9	Dummyvalue9	Dummyvalue9	Dummyvalue9	9.00
13	Dummyvalue10	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue10	10.00
14	Dummyvalue11	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue11	11.00
15	Dummyvalue12	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue12	12.00
16	Dummyvalue13	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue13	13.00
17	Dummyvalue14	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue14	14.00
18	Dummyvalue15	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue15	15.00
19	Dummyvalue16	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue16	16.00
20	Dummyvalue17	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue17	17.00
21	Dummyvalue18	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue...	Dummyvalue18	18.00

Figure 7.0: Display Records screen

When the ‘Display All Records’ button in the main menu is clicked, the window content changes and the ‘Display Records’ screen is displayed. The ‘Display Records’ function allows users to view the details of all car records that have been added to the system.

The car records are inserted into a table that becomes scrollable if the number of records causes the table to overflow. Users can also use the search box provided above the table to filter the car records displayed on the screen by searching a specific keyword. Any records that contain the entered keyword in any of their columns will be displayed.

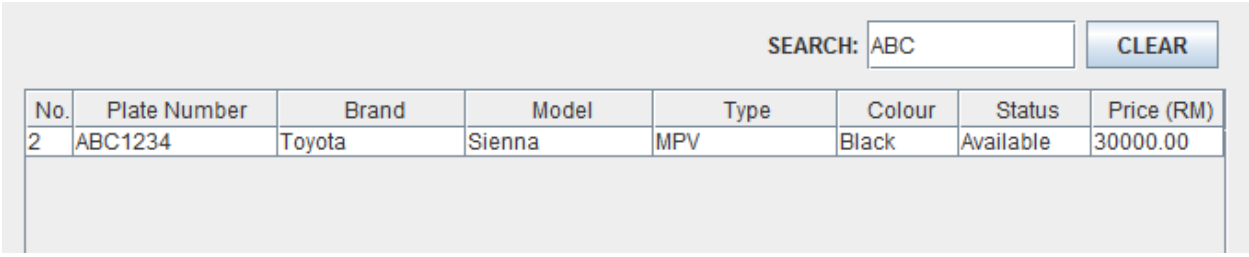


Figure 7.1: System filters table records using the specific keyword

The table columns are also resizable, so users can drag the left or right borders of the columns to view overflowing characters that were previously hidden.

Figure 7.2: Resizable table columns that can be utilized to show hidden characters

- **Exit**

Users may click on the close button at the far right of the window's title bar to exit the system. When users click on it, the system displays a confirmation dialog and prompts them to confirm if they want to exit the system. If the user clicks 'Yes', the system ends; if the user clicks 'No', the system will not end, and it redirects the user to the main menu.

Figure 8.0: Default close button in window title bar

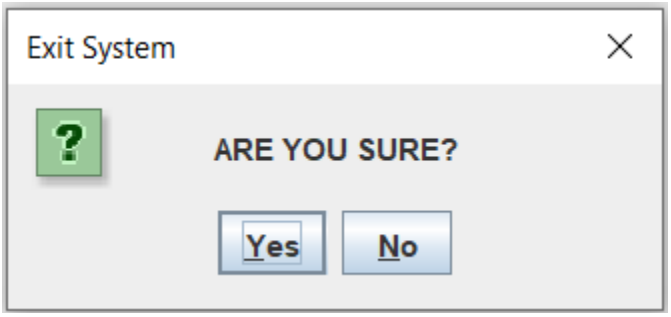


Figure 8.1: Confirmation dialog for exiting the program

BIBLIOGRAPHY

Oracle (2021a) *How to Make Dialogs*. Available from https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html [accessed 13 July 2021].

Oracle (2021b) *How to Use CardLayout*. Available from https://docs.oracle.com/javase/tutorial/uiswing/layout/card.html [accessed 13 July 2021].

Oracle (2021c) *How to Use Password Fields*. Available from https://docs.oracle.com/javase/tutorial/uiswing/components/passwordfield.html [accessed 23 July 2021].

Oracle (2021d) *How to Use Scroll Panes*. Available from https://docs.oracle.com/javase/tutorial/uiswing/components/scrollpane.html [accessed 19 July 2021].

Oracle (2021e) *How to Use Tables*. Available from https://docs.oracle.com/javase/tutorial/uiswing/components/table.html [accessed 19 July 2021].

Oracle (2021f) *How to Write a Key Listener*. Available from https://docs.oracle.com/javase/tutorial/uiswing/events/keylistener.html [accessed 23 July 2021].

Oracle (2021g) *JOptionPane*. Available from https://docs.oracle.com/javase/7/docs/api/javax/swing/JOptionPane.html [accessed 13 July 2021].

StackOverflow (2013a) *Load arrayList data into JTable*. Available from https://stackoverflow.com/questions/20526917/load-arraylist-data-into-jtable [accessed 20 July 2021].

StackOverflow (2013b) *Putting JLabel on top of component*. Available from https://stackoverflow.com/questions/20565782/putting-jlabel-on-top-of-component [accessed 20 July 2021].

StackOverflow (2015) *java unable to set size for textfield*. Available from https://stackoverflow.com/questions/30680673/java-unable-to-set-size-for-textfield [accessed 20 July 2021].

StackOverflow (2016) *How do I put offsets between the JPanels and the JFrame?* Available from https://stackoverflow.com/questions/40976625/how-do-i-put-offsets-between-the-jpanels-and-the-jframe [accessed 20 July 2021].

StackOverflow (2020) *Is it possible to have a java swing border only on the top side?* Available from https://stackoverflow.com/questions/2174319/is-it-possible-to-have-a-java-swing-border-only-on-the-top-side [accessed 20 July 2021].