

**PROJECT REPORT**

**PROJECT TITLE: Palindrome Check**

**PRESENTED BY: Sagar Dhondge**

**INTERN ID: VN-JD-4W289**

Palindrome Checker App Report.

# 1. Introduction

# The Palindrome Checker Application is a simple yet effective tool designed to determine whether a given word, phrase, or number is a palindrome. A palindrome reads the same forwards and backwards, disregarding spaces, punctuation, and capitalization. This application aims to provide immediate feedback on whether the input is a palindrome, enhancing the user's understanding of this interesting linguistic and numerical phenomenon.

# 2. Technologies Used

* **Java**: Core programming language used for the application's development.
* **JavaFX**: Used for creating the graphical user interface (GUI).
* **MySQL**: A relational database management system to store user data and quiz results.
* **Swing (optional)**: Java framework used for creating the GUI.
* **Standard Java Libraries**: Utilized for string manipulation, I/O operations, and user interaction.

# 3. **Objective and Scope**

**Objectives**

* To develop an interactive application that checks if a given input is a palindrome.
* To provide immediate feedback on the palindrome status of the input.
* To handle various types of inputs including words, phrases, and numbers.

**Scope**

* The application will process inputs of different lengths and characters.
* It will offer both command-line and graphical user interface (GUI) interactions.
* Future versions may include more complex input handling and language support.

# 4. System Design

**4.1 Architecture Pattern: MVC (Model-View-Controller)**

* Model: Handles the logic for determining whether an input is a palindrome.
* View: (Optional) GUI components that display the input and result.
* Controller: Manages user input, validates the input, and triggers the palindrome check.

# **4.2 Flow of the Application**

# Accept input from the user.

# Normalize the input by removing spaces, punctuation, and converting it to lowercase.

# Check if the normalized input reads the same forwards and backwards.

# Display the result to the user.

# 5. Implementation

**5.1. User Interface**

* **Display:** It consist of three boxes in which we have to give some input to user to check it is palindrome or not.
* **Main Menu:** Provides options for users to input text or numbers to check the result.
* **Result Display**: Shows whether the input is a palindrome**.**

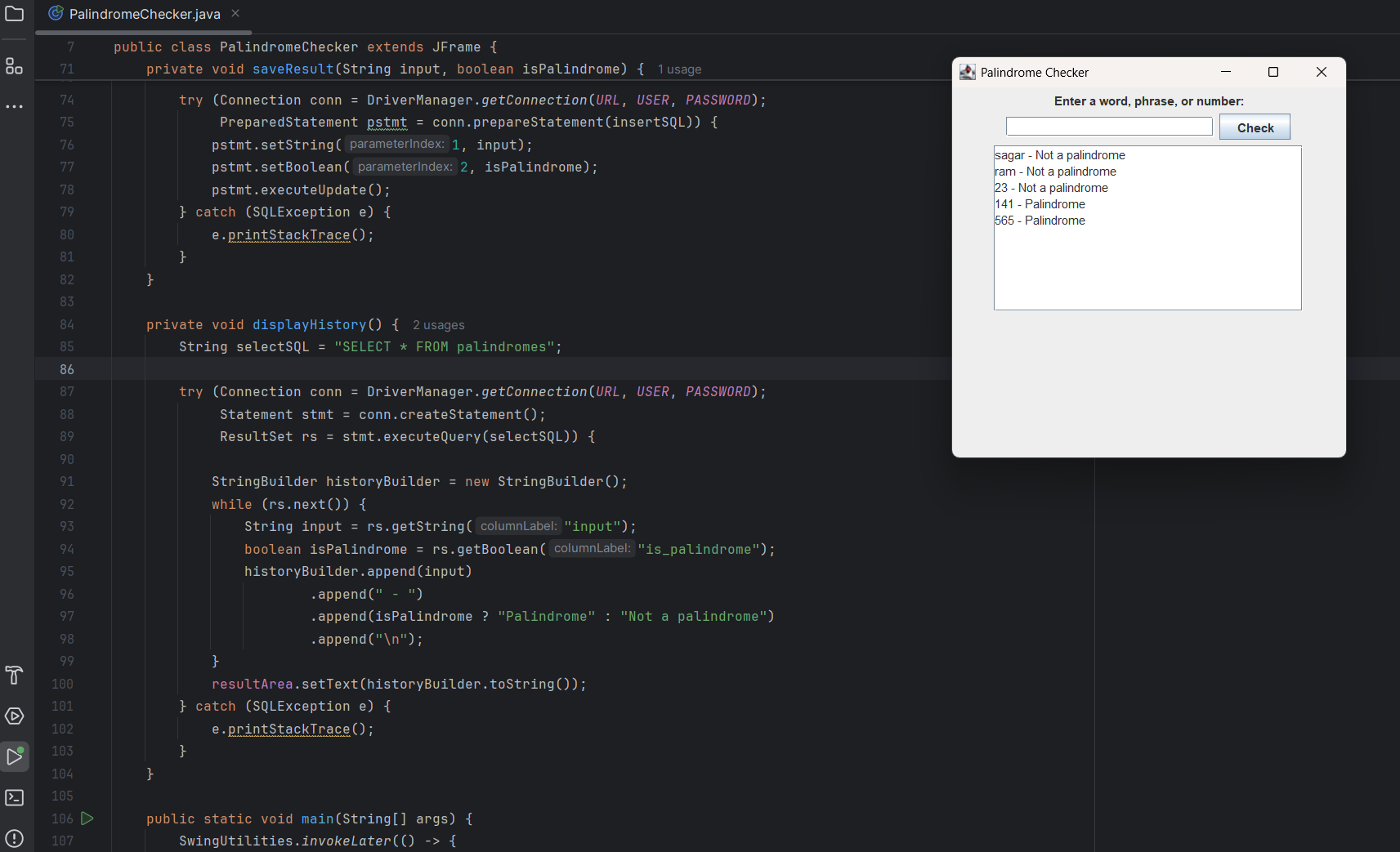
**5.2. Database Integration**

* **Connection**: Uses JDBC (Java Database Connectivity) to connect to the MySQL database.
* **CRUD Operations**: Includes operations to create, read, update the display bar.

# 6. Code.

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.ActionEvent;  
import java.awt.event.ActionListener;  
import java.sql.\*;  
  
public class PalindromeChecker extends JFrame {  
  
 private JTextField inputField;  
 private JTextArea resultArea;  
 private static final String *URL* = "jdbc:mysql://localhost:3306/palindrome";  
 private static final String *USER* = "root";  
 private static final String *PASSWORD* = "Sagar@9075";  
  
 public PalindromeChecker() {  
 setTitle("Palindrome Checker");  
 setSize(400, 400);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setLayout(new FlowLayout());  
  
 JLabel inputLabel = new JLabel("Enter a word, phrase, or number:");  
 inputField = new JTextField(20);  
 JButton checkButton = new JButton("Check");  
 resultArea = new JTextArea(10, 30);  
 resultArea.setEditable(false);  
  
 add(inputLabel);  
 add(inputField);  
 add(checkButton);  
 add(new JScrollPane(resultArea));  
  
 checkButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String input = inputField.getText();  
 if (input != null && !input.isEmpty()) {  
 boolean isPalindrome = checkPalindrome(input);  
 saveResult(input, isPalindrome);  
 displayHistory();  
 inputField.setText("");  
 } else {  
 JOptionPane.*showMessageDialog*(null, "Please enter a valid input.", "Input Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 });  
  
 createTableIfNotExists();  
 displayHistory();  
 }  
  
 private boolean checkPalindrome(String str) {  
 String cleanStr = str.replaceAll("[^a-zA-Z0-9]", "").toLowerCase();  
 String reversedStr = new StringBuilder(cleanStr).reverse().toString();  
 return cleanStr.equals(reversedStr);  
 }  
  
 private void createTableIfNotExists() {  
 String createTableSQL = "CREATE TABLE IF NOT EXISTS palindromes ("  
 + "id INT AUTO\_INCREMENT PRIMARY KEY, "  
 + "input TEXT NOT NULL, "  
 + "is\_palindrome BOOLEAN NOT NULL)";  
  
 try (Connection conn = DriverManager.*getConnection*(*URL*, *USER*, *PASSWORD*);  
 Statement stmt = conn.createStatement()) {  
 stmt.execute(createTableSQL);  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void saveResult(String input, boolean isPalindrome) {  
 String insertSQL = "INSERT INTO palindromes(input, is\_palindrome) VALUES(?, ?)";  
  
 try (Connection conn = DriverManager.*getConnection*(*URL*, *USER*, *PASSWORD*);  
 PreparedStatement pstmt = conn.prepareStatement(insertSQL)) {  
 pstmt.setString(1, input);  
 pstmt.setBoolean(2, isPalindrome);  
 pstmt.executeUpdate();  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void displayHistory() {  
 String selectSQL = "SELECT \* FROM palindromes";  
  
 try (Connection conn = DriverManager.*getConnection*(*URL*, *USER*, *PASSWORD*);  
 Statement stmt = conn.createStatement();  
 ResultSet rs = stmt.executeQuery(selectSQL)) {  
  
 StringBuilder historyBuilder = new StringBuilder();  
 while (rs.next()) {  
 String input = rs.getString("input");  
 boolean isPalindrome = rs.getBoolean("is\_palindrome");  
 historyBuilder.append(input)  
 .append(" - ")  
 .append(isPalindrome ? "Palindrome" : "Not a palindrome")  
 .append("\n");  
 }  
 resultArea.setText(historyBuilder.toString());  
 } catch (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> {  
 PalindromeChecker frame = new PalindromeChecker();  
 frame.setVisible(true);  
 });  
 }  
}

# 7.Result.



# **7. Testing**

**Unit Testing**: Conducted for the isPalindrome method to ensure accurate palindrome checking.

**Integration Testing**: Ensured smooth interaction between the app and the database.

**User Testing**: Feedback from users helped refine the UI and improve the user experience.

**8. Result**

The Palindrome Checker App accurately identifies palindromes and provides immediate user feedback, making it a useful tool for educational and practical purposes.

**8. Conclusion**

The Palindrome Checker App offers a simple yet effective platform for users to explore and understand palindromes. With potential for GUI enhancements and additional features, the app can evolve to provide a more interactive and engaging user experience.