MEDICATION REMINDER

An

Object-Oriented Programming through Java Course Project Report

in partial fulfilment of the degree

Bachelor of Technology
in
Computer Science & Engineering

By

Name: K.Sai Nath **HTNo:**2103A51560

Name: P.Sai Charith **HTNo:**2103A51098

Name: G.Vamshi Krishna **HTNo:**2103A51554

Under the guidance of

K.Balakrishna

Asst professor , CS&AI

Submitted to

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

SR UNIVERSITY

SR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Object Oriented Programming through Java - Course Project Report entitled " **MEDICATION REMINDER** " is a record of bonafide work carried out by the student **K.Sai Nath, P.Sai Charith, G.Vamshi Krishna** bearing Roll No(s) **2103A51560, 2103A51098, 2103A51554** during the academic year 2022-23 in partial fulfillment of the award of the degree of Bachelor of Technology in Computer Science & Engineering by the SR University , Ananthsagar ,Warangal.

Lab In-charge **Head of the Department**

Table of Contents

1.ABSTRACT

The Medication Reminder System is a Java-based software application designed to assist individuals in managing their medication schedules efficiently. Medication adherence is crucial for maintaining good health, especially for individuals with chronic conditions or complex medication regimens. This project aims to address this issue by providing a user-friendly and customisable solution.The key features of the Medication Reminder System include:

1. **User Profiles:** The system allows users to create and manage their profiles, including personal information, medical history, and a list of prescribed medications.
2. **Medication Database:** A comprehensive database of medications is integrated into the system, making it easy for users to search and add their medications remainders. Users can also specify dosage instructions.
3. **Medication Scheduling:** Users can set up medication schedules, specifying the frequency, dosage, and timing of each medication. The system provides flexibility in setting one-time or recurring reminders.
4. **Alerts and Notifications:** The system sends reminders and notifications to the user at the scheduled times, ensuring they never miss a dose. Notifications can be delivered through various channels, such as SMS, email, or mobile app notifications.

2.OBJECTIVE OF THE PROJECT

The objectives of Medication Reminder Project are to create a software application or system that helps individuals manage their medication schedules effectively. The primary goals and objectives of this project include.

1. **Enhance Medication Adherence:** Improve and increase the adherence of users to their prescribed medication regimens by sending timely reminders.
2. **Improve Health Outcomes:** Ultimately, the project aims to contribute to better health outcomes for users by ensuring that they take their medications as prescribed, which can lead to more effective disease management and improved overall well-being.
3. **Simplify Medication Management:** Simplify the process of managing multiple medications, dosages, and schedules by providing a user-friendly and centralised platform.
4. **Reduce Medication Errors:** Minimise the risk of medication errors, such as double dosing or missing doses, by providing clear and accurate reminders.
5. **Increase User Convenience:** Provide users with a convenient and easy-to-use system for tracking and managing their medications. Customisable reminders and user-friendly interfaces are essential.

3.ELEMENTS USED IN THE PROJECT

1. **JavaFX:** JavaFX is a Java-based framework and platform for building rich and interactive desktop and web applications. JavaFX is designed to provide a modern, visually appealing, and highly functional user interface for Java applications.
2. **Java Swing:** Java Swing is a graphical user interface (GUI) toolkit and library for Java. It provides a set of components and tools for building desktop applications with a graphical user interface.Swing is a lightweight GUI toolkit, meaning it doesn't rely on the platform's native components for rendering.
3. **Java Database Connectivity (JDBC):** Java Database Connectivity (JDBC) is a Java-based API and framework that allows Java applications to interact with relational databases. JDBC provides a standard interface for connecting to, querying, and updating databases, making it an essential tool for database-driven applications.
4. **AWT:** Java AWT (Abstract Window Toolkit) is a set of classes in the Java programming language for creating graphical user interfaces (GUIs). AWT provides a platform-independent way to create windows, dialogs, buttons, menus, and other GUI elements in Java applications.
5. **ActionListener:** ActionListener is an interface used for handling button click events. It defines methods to respond to user interactions with buttons, like adding a new task or habit when a button is clicked.
6. **SQL Commands:** SQL commands are statements used to create and manage a database. They include actions like creating tables, inserting data, updating records, and deleting information, facilitating interaction with the database.
7. **Timer:** The java.util.Timer class in Java is used for scheduling tasks to run at specific times or after predefined time intervals. It provides a simple and straightforward way to set up timers for various purposes.
8. **Text**: The java.text package provides classes and interfaces for handling various text-related operations like SimpleDateFormat.

4.IMPLEMENTATION-CODE

<u>JDBC User Login</u>:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class JDBC_User {

	public static void main(String[] args) {

		try {
			Class.forName("oracle.jdbc.driver.OracleDriver");
Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1098");

			Statement smt=con.createStatement();
			smt.executeUpdate("create table users(username varchar(50) PRIMARY
KEY,password varchar(100))");

			System.out.print("Table Created successfully");
		}
		catch(Exception e)
		{
			System.out.print(e);
		}
	}

}
```

<u>User Login</u>:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.sql.ResultSet;
import javax.swing.border.Border;
import javax.swing.border.EmptyBorder;

public class UserRegistrationLogin2 {
    private JPanel panel;
    private JTextField usernameField;
    private JPasswordField passwordField;

    public UserRegistrationLogin2() {
        panel = new JPanel();
        panel.setLayout(new GridLayout(6, 2));
        panel.setBackground(new Color(173, 216, 230));
        Border roundedBorder = new EmptyBorder(10, 10, 10, 10);
        Font robotoFont = new Font("Roboto", Font.BOLD, 16);

        JLabel welcomeLabel = new JLabel("Welcome to Medical Reminder ");
        welcomeLabel.setFont(new Font("Roboto", Font.BOLD, 18));
        welcomeLabel.setForeground(Color.BLACK);
        panel.add(welcomeLabel);
        panel.add(new JLabel());
        JLabel subtitleLabel = new JLabel("An Application for Reminding You to Take Your Pills On
Time");
        subtitleLabel.setFont(new Font("Roboto", Font.BOLD, 14));
        subtitleLabel.setForeground(Color.BLACK);
        panel.add(subtitleLabel);
        panel.add(new JLabel());

        JLabel usernameLabel = new JLabel("Username:");
        usernameLabel.setFont(robotoFont);
        usernameLabel.setForeground(Color.BLACK);
        panel.add(usernameLabel);
        usernameField = new JTextField();
        usernameField.setFont(robotoFont);
        panel.add(usernameField);

        JLabel passwordLabel = new JLabel("Password:");
        passwordLabel.setFont(robotoFont);
        passwordLabel.setForeground(Color.BLACK);
        panel.add(passwordLabel);
        passwordField = new JPasswordField();
        passwordField.setFont(robotoFont);
        panel.add(passwordField);

        JButton registerButton = new JButton("Register");
        registerButton.setFont(robotoFont);
        registerButton.setBackground(new Color(255, 0, 0));
        registerButton.setForeground(Color.WHITE);
        registerButton.setBorder(roundedBorder);
        panel.add(registerButton);

        JButton loginButton = new JButton("Login");
        loginButton.setFont(robotoFont);
        loginButton.setBackground(new Color(0, 128, 0));
        loginButton.setForeground(Color.WHITE);
        loginButton.setBorder(roundedBorder);
        panel.add(loginButton);
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
            JOptionPane.showMessageDialog(null, "Oracle JDBC driver not found.", "Error",
JOptionPane.ERROR_MESSAGE);
        }

        registerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());

                if (registerUser(username, password)) {
                    JOptionPane.showMessageDialog(null, "Registration successful.", "Success",
JOptionPane.INFORMATION_MESSAGE);
                } else {
                    JOptionPane.showMessageDialog(null, "Registration failed. Username already exists.",
"Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });

        loginButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String username = usernameField.getText();
                String password = new String(passwordField.getPassword());

                if (loginUser(username, password)) {
                    JFrame topFrame = (JFrame) SwingUtilities.getWindowAncestor(panel);
                    topFrame.dispose();

                    new MedicRem(username);
                } else {
                    JOptionPane.showMessageDialog(null, "Login failed. Invalid username or password.",
"Error", JOptionPane.ERROR_MESSAGE);
                }
            }
        });
    }

    private static boolean registerUser(String username, String password) {
        String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:xe";
        String dbUsername = "system";
        String dbPassword = "1098";
        try {
            Connection connection = DriverManager.getConnection(jdbcUrl, dbUsername, dbPassword);
            String insertSQL = "INSERT INTO Users (username, password) VALUES (?, ?)";
            PreparedStatement preparedStatement = connection.prepareStatement(insertSQL);
            preparedStatement.setString(1, username);
            preparedStatement.setString(2, password);

            int rowsAffected = preparedStatement.executeUpdate();
            preparedStatement.close();
            connection.close();
            return rowsAffected > 0;
        } catch (SQLException ex) {
            ex.printStackTrace();
            return false;
        }
    }

    private static boolean loginUser(String username, String password) {
        String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:xe";
        String dbUsername = "system";
        String dbPassword = "1098";
        try {
            Connection connection = DriverManager.getConnection(jdbcUrl, dbUsername, dbPassword);
            String selectSQL = "SELECT username, password FROM Users WHERE username = ? AND
password = ?";
            PreparedStatement preparedStatement = connection.prepareStatement(selectSQL);
            preparedStatement.setString(1, username);
            preparedStatement.setString(2, password);

            ResultSet resultSet = preparedStatement.executeQuery();
            boolean loginSuccessful = resultSet.next();

            resultSet.close();
            preparedStatement.close();
            connection.close();

            return loginSuccessful;
        } catch (SQLException ex) {
            ex.printStackTrace();
            return false;
        }
    }
public JPanel getPanel() {
        return panel;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            UserRegistrationLogin2 login = new UserRegistrationLogin2();
            JFrame frame = new JFrame("Medical Reminder Application");
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(600, 300);
            frame.setLocationRelativeTo(null);
            frame.add(login.getPanel());
            frame.setVisible(true);
        });
    }
}
```

JDBC Remainder Page:

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;


public class JDBC_Reminder {


	public static void main(String[] args) {

		try {

			Class.forName("oracle.jdbc.driver.OracleDriver");

			Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1098");


			Statement smt=con.createStatement();

			smt.executeUpdate("create table MedicationReminder(MedicationName
varchar2(255),Dosage varchar2(50),Frequency varchar2(50),ScheduledDateTime timestamp,username
varchar2(50),foreign key (username) references users(username))");


			System.out.print("Table Created successfully");

		}


		catch(Exception e)

		{

			System.out.print(e);

		}


	}


}
```

Remainder Page:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
import javax.swing.border.Border;
import java.util.List;
import java.util.Timer;

public class MedicRem {
    private JFrame frame;
    private JTextField medicationNameField;
    private JTextField dosageField;
    private JTextField frequencyField;
    private JTextField dateTimeField;
    private String loggedInUser;

    private List<MedicationReminder> reminders;

    public MedicRem(String loggedInUser) {
        this.loggedInUser = loggedInUser;
        frame = new JFrame("Medication MedicRem");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(600, 400);
        frame.setLocationRelativeTo(null);

        reminders = new ArrayList<>();

        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(7, 2));
        panel.setBackground(new Color(173, 216, 230));

        Font robotoFont = new Font("Roboto", Font.BOLD, 16);

        JLabel medicationNameLabel = new JLabel("Medication Name:");
        medicationNameLabel.setFont(robotoFont);
        medicationNameLabel.setForeground(Color.RED);
        medicationNameField = new JTextField();
        medicationNameField.setFont(robotoFont);
        panel.add(medicationNameLabel);
        panel.add(medicationNameField);

        JLabel dosageLabel = new JLabel("Dosage:");
        dosageLabel.setFont(robotoFont);
        dosageLabel.setForeground(Color.RED);
        dosageField = new JTextField();
        dosageField.setFont(robotoFont);
        panel.add(dosageLabel);
        panel.add(dosageField);

        JLabel frequencyLabel = new JLabel("Frequency:");
        frequencyLabel.setFont(robotoFont);
        frequencyLabel.setForeground(Color.RED);
        frequencyField = new JTextField();
        frequencyField.setFont(robotoFont);
```

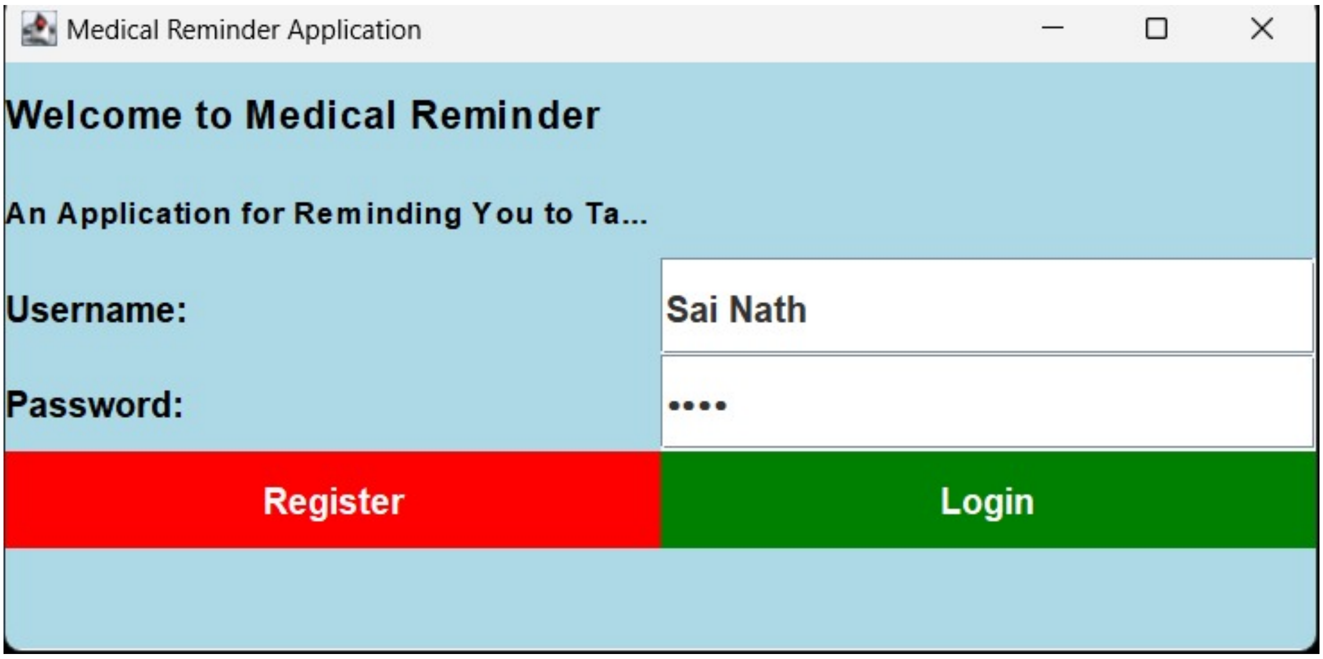
```
panel.add(frequencyLabel);
panel.add(frequencyField);

JLabel dateTimeLabel = new JLabel("Scheduled Date and Time (MM/dd/yyyy hh:mm a):");
dateTimeLabel.setFont(robotoFont);
dateTimeLabel.setForeground(Color.RED);
dateTimeField = new JTextField();
dateTimeField.setFont(robotoFont);
panel.add(dateTimeLabel);
panel.add(dateTimeField);

JButton addReminderButton = new JButton("Add Reminder");
addReminderButton.setFont(robotoFont);
addReminderButton.setBackground(new Color(255, 69, 0));
addReminderButton.setForeground(Color.WHITE);
panel.add(addReminderButton);

JButton activeRemindersButton = new JButton("Active Reminders");
activeRemindersButton.setFont(new Font("Roboto", Font.BOLD, 14));
activeRemindersButton.setBackground(new Color(0, 128, 0));
activeRemindersButton.setForeground(Color.WHITE);
panel.add(activeRemindersButton);

int borderRadius = 15;
addReminderButton.setBorder(new RoundedBorder(borderRadius));
activeRemindersButton.setBorder(new RoundedBorder(borderRadius));

addReminderButton.addActionListener(new ActionListener() {
   @Override
   public void actionPerformed(ActionEvent e) {
      submitMedicRem();
```

```
        }
    });

    activeRemindersButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            showActiveReminders();
        }
    });

    frame.add(panel);
    frame.setVisible(true);
}

private void submitMedicRem() {
    String medicationName = medicationNameField.getText();
    String dosage = dosageField.getText();
    String frequency = frequencyField.getText();
    String inputDateTime = dateTimeField.getText();

    String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:XE";
    String username = "system";
    String password = "1098";

    Connection connection = null;

    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection = DriverManager.getConnection(jdbcUrl, username, password);
```

```
            String insertSQL = "INSERT INTO MedicationReminder (Username, MedicationName,
Dosage, Frequency, ScheduledDateTime) " +
                    "VALUES (?, ?, ?, ?, TO_TIMESTAMP(?, 'MM/DD/YYYY HH24:MI'))";

            PreparedStatement preparedStatement = connection.prepareStatement(insertSQL);
            preparedStatement.setString(1, loggedInUser);
            preparedStatement.setString(2, medicationName);
            preparedStatement.setString(3, dosage);
            preparedStatement.setString(4, frequency);
            String formattedDateTime = convertTo24HourFormat(inputDateTime);
            preparedStatement.setString(5, formattedDateTime);

            int rowsAffected = preparedStatement.executeUpdate();
            if (rowsAffected > 0) {
                MedicationReminder reminder = new MedicationReminder(formattedDateTime,
medicationName, dosage, frequency);
                scheduleReminder(reminder);
                reminders.add(reminder);

                showCustomMessageBox("Medication Reminder Added", "Medicine Name: " +
medicationName + "\nDosage: " + dosage + "\nFrequency: " + frequency + "\nTime: " +
formattedDateTime, "success");
            } else {
                showCustomMessageBox("Failed to Add Reminder", "Please check your input.", "error");
            }
        } catch (ClassNotFoundException | SQLException ex) {
            ex.printStackTrace();
            showCustomMessageBox("Database Error", "Error: " + ex.getMessage(), "error");
        } finally {
            try {
                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }

    private void showActiveReminders() {
        StringBuilder reminderText = new StringBuilder("Active Reminders:\n\n");
        for (MedicationReminder reminder : reminders) {
            reminderText.append("Medication Name:
").append(reminder.getMedicationName()).append("\n");
            reminderText.append("Dosage: ").append(reminder.getDosage()).append("\n");
            reminderText.append("Frequency: ").append(reminder.getFrequency()).append("\n");
            reminderText.append("Scheduled Time:
").append(reminder.getScheduledDateTime()).append("\n\n");
        }

        if (reminders.isEmpty()) {
            reminderText.append("No active reminders.");
        }

        showCustomMessageBox("Active Reminders", reminderText.toString(), "info");
    }

    private void scheduleReminder(MedicationReminder reminder) {
        try {
            SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy HH:mm");
            Date scheduledDateTime = sdf.parse(reminder.getScheduledDateTime());

            Date currentTime = new Date();
            if (scheduledDateTime.before(currentTime)) {
                showCustomMessageBox("The time for your medication has already passed. You cannot set
a past time.", "Reminder Alert", "error");
                reminders.remove(reminder);
            } else {
                Timer timer = new Timer();
                timer.schedule(new TimerTask() {
                    @Override
                    public void run() {
                        showReminder(reminder.getMedicationName(), reminder.getDosage(),
reminder.getFrequency());
                        reminders.remove(reminder);
                    }
                }, scheduledDateTime);
            }
        } catch (ParseException e) {
            e.printStackTrace();
            showCustomMessageBox("Invalid date and time format. Please use MM/dd/yyyy hh:mm AM/
PM.", "Error", "error");
        }
    }

    private void showReminder(String medicationName, String dosage, String frequency) {
        showCustomMessageBox("Take Your Medicine",
                "Time to take your medication:\n" +
                        "Medication Name: " + medicationName + "\n" +
                        "Dosage: " + dosage + "\n" +
                        "Frequency: " + frequency,
                "info");
    }

    private String convertTo24HourFormat(String inputDateTime) {
        try {
            SimpleDateFormat inputFormat = new SimpleDateFormat("MM/dd/yyyy hh:mm a");
            Date date = inputFormat.parse(inputDateTime);

            Calendar calendar = Calendar.getInstance();
            calendar.setTime(date);

            SimpleDateFormat outputFormat = new SimpleDateFormat("MM/dd/yyyy HH:mm");
            return outputFormat.format(calendar.getTime());
        } catch (ParseException e) {
            showCustomMessageBox("Invalid Date and Time Format", "Please use MM/dd/yyyy hh:mm
AM/PM.", "error");
            return "";
        }
    }

    private void showCustomMessageBox(String title, String message, String messageType) {
        JPanel panel = new JPanel();
        panel.setLayout(new BoxLayout(panel, BoxLayout.PAGE_AXIS));

        JLabel titleLabel = new JLabel(title);
        titleLabel.setFont(new Font("Roboto", Font.BOLD, 18));
        JLabel titleLabel2 = new JLabel("");

        JTextArea textArea = new JTextArea(message);
        textArea.setFont(new Font("Roboto", Font.PLAIN, 14));
        textArea.setWrapStyleWord(true);
        textArea.setLineWrap(true);
        textArea.setOpaque(false);
        textArea.setEditable(false);

        panel.add(titleLabel);
        panel.add(titleLabel2);
        panel.add(textArea);

        if (messageType.equals("success")) {
            titleLabel.setForeground(Color.GREEN);
        } else if (messageType.equals("error")) {
            titleLabel.setForeground(Color.RED);
        } else if (messageType.equals("info")) {
            titleLabel.setForeground(new Color(0, 102, 204));
        }

        ImageIcon alertIcon = new ImageIcon("alert_icon.png");
        int optionType = JOptionPane.DEFAULT_OPTION;
        int messageTypeCode = JOptionPane.PLAIN_MESSAGE;
        JOptionPane.showOptionDialog(null, panel, "Message", optionType, messageTypeCode,
alertIcon, new Object[]{"OK"}, null);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new MedicRem("Your_Logged_In_Username"));
    }
}

class MedicationReminder {
    private String scheduledDateTime;
    private String medicationName;
    private String dosage;
    private String frequency;

    public MedicationReminder(String scheduledDateTime, String medicationName, String dosage,
String frequency) {
        this.scheduledDateTime = scheduledDateTime;
        this.medicationName = medicationName;
        this.dosage = dosage;
        this.frequency = frequency;
    }

    public String getScheduledDateTime() {
        return scheduledDateTime;
    }

    public String getMedicationName() {
        return medicationName;
    }

    public String getDosage() {
        return dosage;
    }

    public String getFrequency() {
        return frequency;
    }
}

class RoundedBorder implements Border {
    private int radius;

    public RoundedBorder(int radius) {
        this.radius = radius;
    }

    public void paintBorder(Component c, Graphics g, int x, int y, int width, int height) {
        g.drawRoundRect(x, y, width - 1, height - 1, radius, radius);
    }

    public Insets getBorderInsets(Component c) {
        return new Insets(this.radius + 1, this.radius + 1, this.radius + 2, this.radius + 1);
    }

    public boolean isBorderOpaque() {
        return true;
    }
}
```

5.RESULT SCREEN

User Registration and Login


Medical Reminder Application

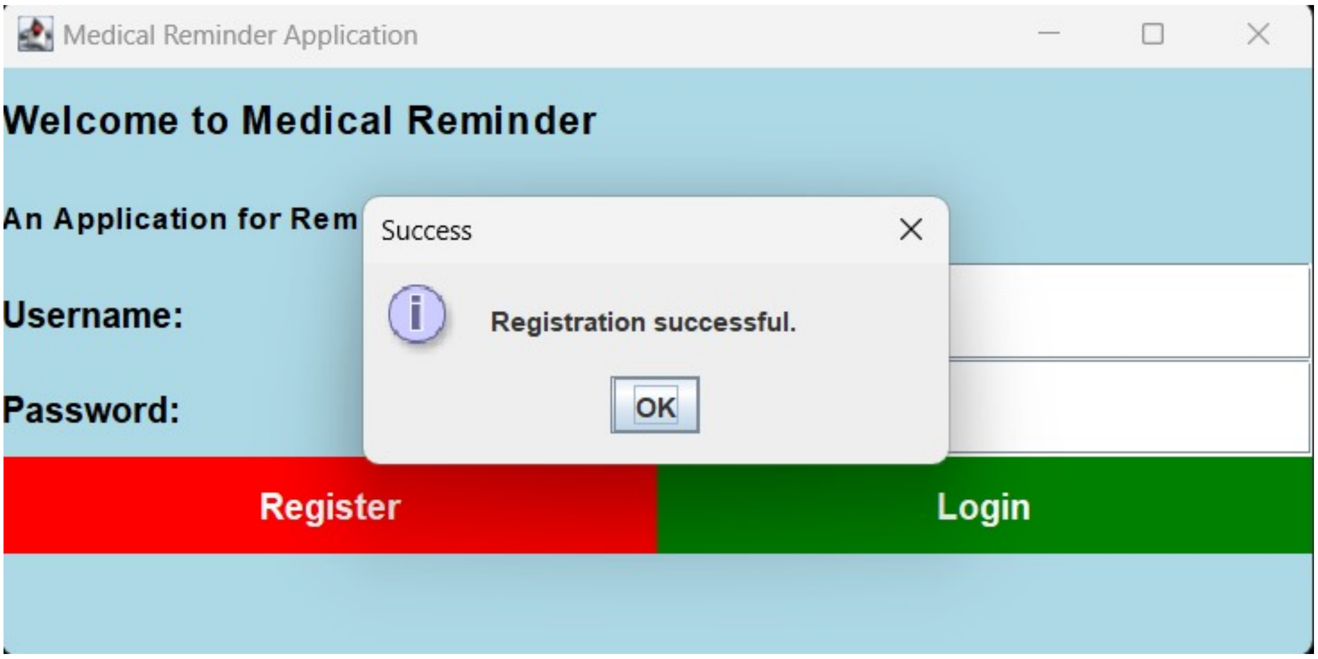
Welcome to Medical Reminder

An Application for Reminding You to Ta...

Username: Sai Nath

Password: ••••

Successful Registration


Medical Reminder Application

Welcome to Medical Reminder

An Application for Rem

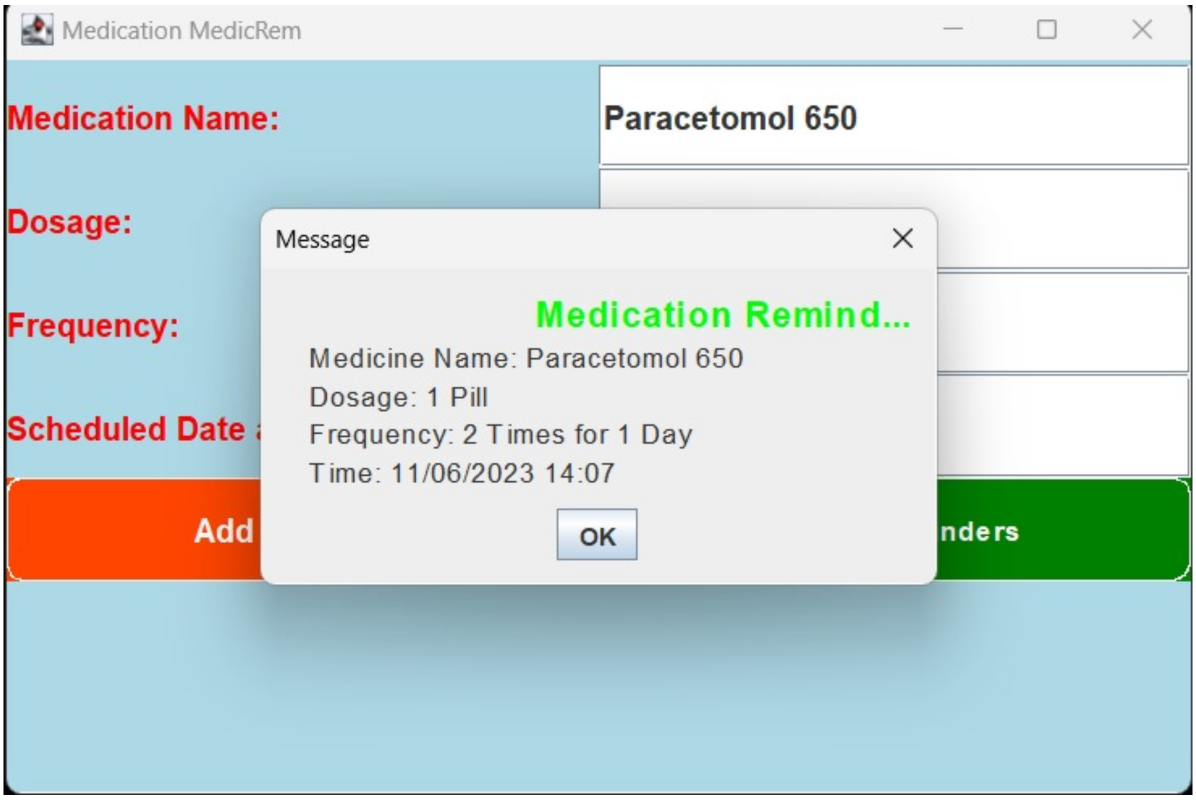
Username:

Password:

Success

Registration successful.

Medication Remainder



Medication MedicRem

Medication Name: Paracetomol 650

Dosage:

Frequency:

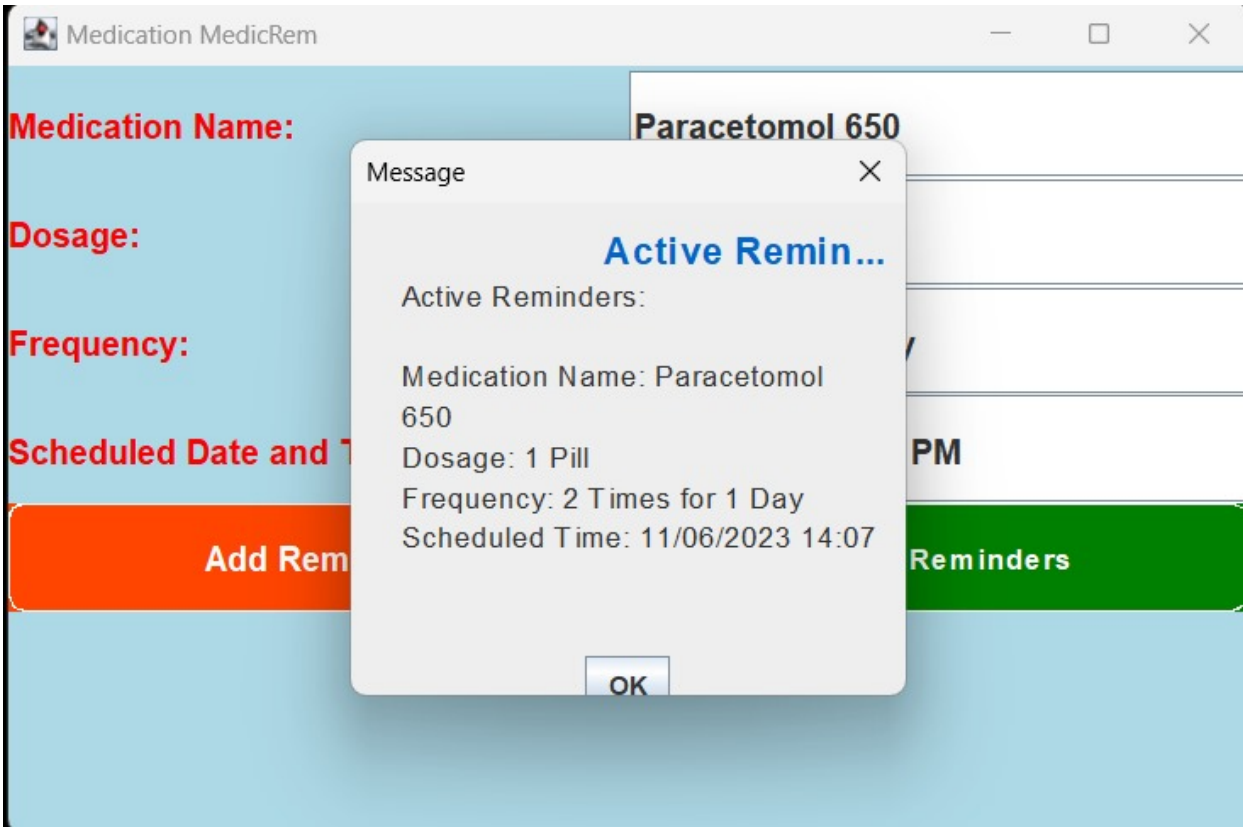
Scheduled Date

Message

Medication Remind...

Medicine Name: Paracetomol 650
Dosage: 1 Pill
Frequency: 2 Times for 1 Day
Time: 11/06/2023 14:07

Active Remainders



Medication MedicRem

Medication Name: Paracetomol 650

Dosage:

Frequency:

Scheduled Date and PM

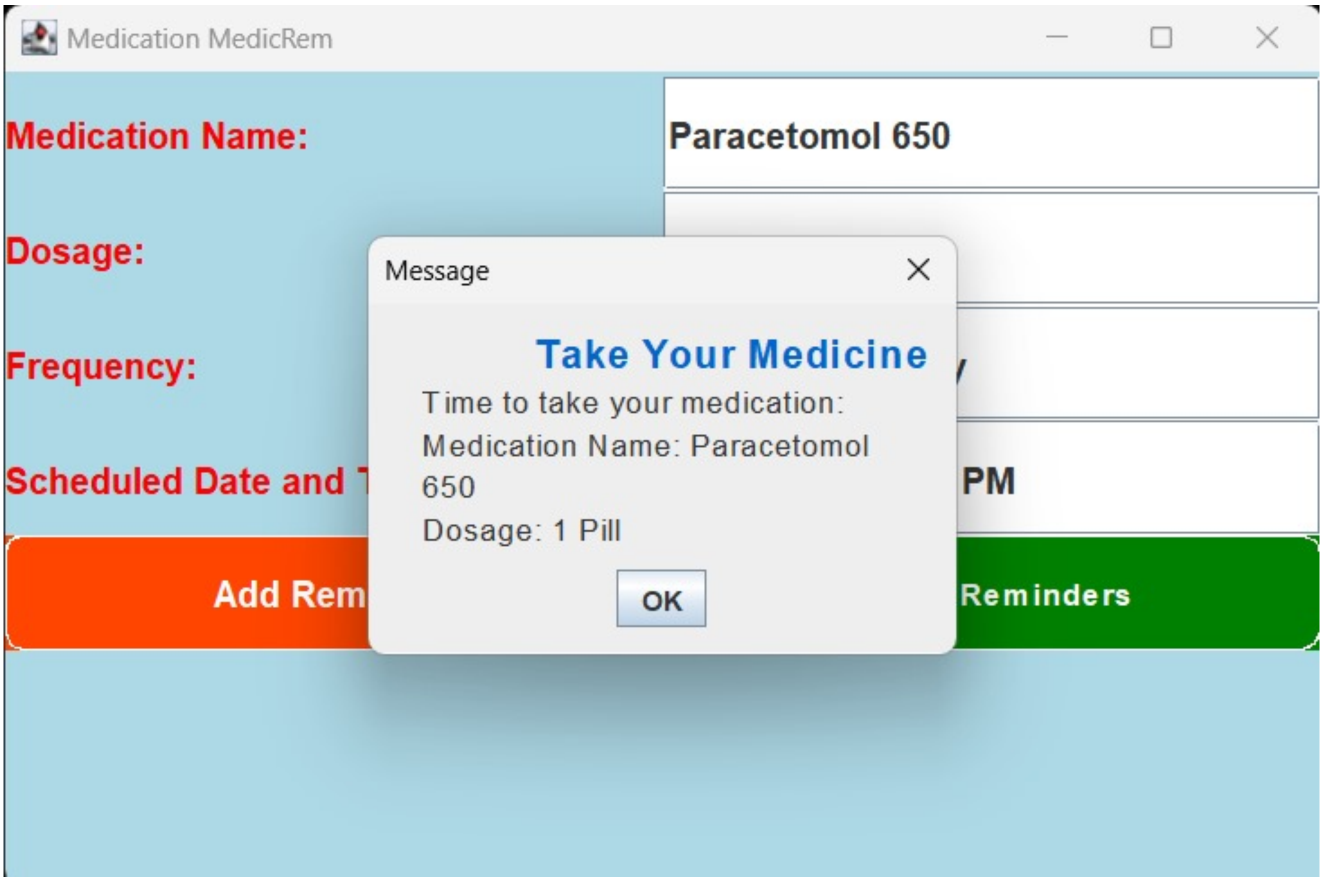
Message

Active Remin...

Active Reminders:

Medication Name: Paracetomol 650
Dosage: 1 Pill
Frequency: 2 Times for 1 Day
Scheduled Time: 11/06/2023 14:07

Remainder Pop Up


Medication MedicRem

Medication Name: Paracetomol 650

Dosage:

Frequency:

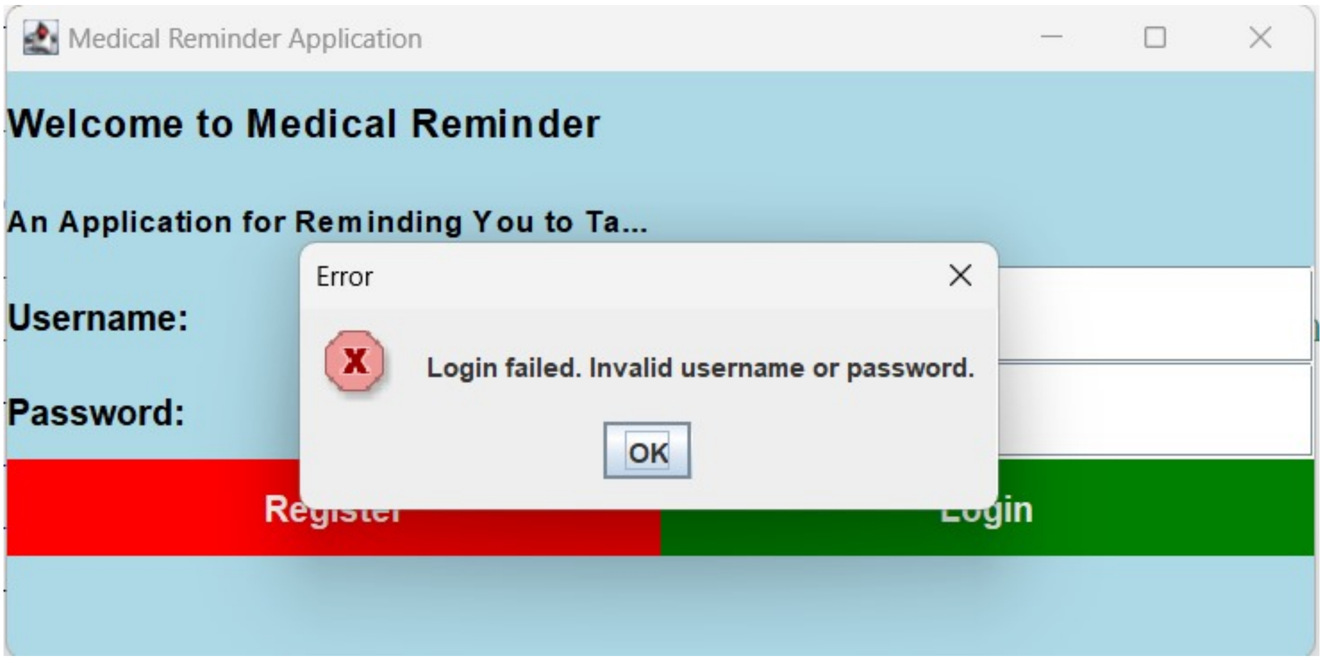
Scheduled Date and T PM

Message

Take Your Medicine

Time to take your medication:
Medication Name: Paracetomol 650
Dosage: 1 Pill

Failed Login


Medical Reminder Application

Welcome to Medical Reminder

An Application for Reminding You to Ta...

Username:

Password:

Error

Login failed. Invalid username or password.

6.CONCLUSION

In conclusion, the Medication Reminder System represents a valuable and user-centric solution to a critical healthcare challenge. By addressing the need for improved medication adherence and simplifying medication management, this system offers numerous benefits to individuals, healthcare providers, and the overall healthcare ecosystem.This project leverages Java and associated technologies to provide a versatile, cross-platform application that can run on various devices, ensuring accessibility and convenience. It not only serves as a practical solution for medication adherence but also allows for future developments and integrations to enhance its capabilities further.