MINI PROJECT-Expense Tracker

AIM:

To create a java swing project on ExpenseTracker with database connection

Java Code:

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

public class ExpenseTrackerApp {

    private JFrame frame;
    private JTable table;
    private ExpenseManager expenseManager;
    private JLabel lblTotalExpenses;

    public static void main(String[] args) {
        EventQueue.invokeLater(() -> {
            try {
                ExpenseTrackerApp window = new ExpenseTrackerApp();
                window.frame.setVisible(true);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    });
}

public ExpenseTrackerApp() {
    expenseManager = new ExpenseManager(); // Initialize ExpenseManager
    initialize();
}

private void initialize() {
    frame = new JFrame();
    frame.setBounds(100, 100, 800, 600);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(new BorderLayout());

    // Buttons
    JButton btnViewExpenses = new JButton("View Expenses");
    btnViewExpenses.addActionListener(e -> loadExpenses());

    JButton btnAddExpense = new JButton("Add Expense");
    btnAddExpense.addActionListener(e -> openAddExpenseForm());
```

```
    JButton btnDeleteExpense = new JButton("Delete Expense");
    btnDeleteExpense.addActionListener(e -> openDeleteExpenseForm());

    JPanel panel = new JPanel();
    panel.add(btnViewExpenses);
    panel.add(btnAddExpense);
    panel.add(btnDeleteExpense);

    frame.getContentPane().add(panel, BorderLayout.NORTH);

    // Table for displaying expenses
    table = new JTable();
    frame.getContentPane().add(new JScrollPane(table), BorderLayout.CENTER);

    // Total expenses label at the bottom
    lblTotalExpenses = new JLabel("Total Expenses: $0.00");
    frame.getContentPane().add(lblTotalExpenses, BorderLayout.SOUTH);
}

private void loadExpenses() {
    List<Expense> expenses = expenseManager.getAllExpenses();
    String[][] data = new String[expenses.size()][4];
    for (int i = 0; i < expenses.size(); i++) {
        Expense expense = expenses.get(i);
```

```
            data[i][0] = String.valueOf(expense.getExpenseId());
            data[i][1] = String.valueOf(expense.getAmount());
            data[i][2] = expense.getDescription();
            data[i][3] = expense.getDate().toString();
        }
        String[] columnNames = {"Expense ID", "Amount", "Description", "Date and Time"};
        table.setModel(new javax.swing.table.DefaultTableModel(data, columnNames));

        // Calculate and display total expenses
        double total = expenseManager.getTotalExpenses();
        lblTotalExpenses.setText("Total Expenses: $" + total);
    }

    private void openAddExpenseForm() {
        JFrame addExpenseFrame = new JFrame("Add Expense");
        addExpenseFrame.setBounds(100, 100, 400, 300);
        addExpenseFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        addExpenseFrame.setLayout(new GridLayout(3, 2));

        JLabel lblAmount = new JLabel("Amount:");
        JTextField txtAmount = new JTextField();

        JLabel lblDescription = new JLabel("Description:");
```

```
        JTextField txtDescription = new JTextField();


        JButton btnSave = new JButton("Save");
        btnSave.addActionListener(e -> {
            try {
                // Validate Amount
                double amount;
                try {
                    amount = Double.parseDouble(txtAmount.getText());
                } catch (NumberFormatException ex) {
                    JOptionPane.showMessageDialog(addExpenseFrame, "Please enter a
valid amount.");
                    return;
                }


                // Validate Description
                String description = txtDescription.getText().trim();
                if (description.isEmpty()) {
                    JOptionPane.showMessageDialog(addExpenseFrame, "Please enter a
description.");
                    return;
                }


                // Get current timestamp
```

```
            java.sql.Timestamp date = new
java.sql.Timestamp(System.currentTimeMillis());


            // Call ExpenseManager to add expense
            boolean isAdded = expenseManager.addExpense(amount, description,
date);
            if (isAdded) {
                JOptionPane.showMessageDialog(addExpenseFrame, "Expense added
successfully.");
                addExpenseFrame.dispose();
                loadExpenses();
            } else {
                JOptionPane.showMessageDialog(addExpenseFrame, "Failed to add
expense.");
            }
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(addExpenseFrame, "An error occurred:
" + ex.getMessage());
        }
    });


    addExpenseFrame.add(lblAmount);
    addExpenseFrame.add(txtAmount);
    addExpenseFrame.add(lblDescription);
    addExpenseFrame.add(txtDescription);
    addExpenseFrame.add(btnSave);
```

```
        addExpenseFrame.setVisible(true);
    }

    private void openDeleteExpenseForm() {
        JFrame deleteExpenseFrame = new JFrame("Delete Expense");
        deleteExpenseFrame.setBounds(100, 100, 400, 150);
        deleteExpenseFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        deleteExpenseFrame.setLayout(new GridLayout(2, 2));

        JLabel lblExpenseId = new JLabel("Expense ID:");
        JTextField txtExpenseId = new JTextField();

        JButton btnDelete = new JButton("Delete");
        btnDelete.addActionListener(e -> {
            try {
                int expenseId = Integer.parseInt(txtExpenseId.getText());
                boolean isDeleted = expenseManager.deleteExpense(expenseId);
                if (isDeleted) {
                    JOptionPane.showMessageDialog(deleteExpenseFrame, "Expense
deleted successfully.");
                    deleteExpenseFrame.dispose();
                    loadExpenses();
                } else {
                    JOptionPane.showMessageDialog(deleteExpenseFrame, "Failed to
delete expense.");
```

```
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(deleteExpenseFrame, "Please enter a valid Expense ID.");
        }
    });

    deleteExpenseFrame.add(lblExpenseId);
    deleteExpenseFrame.add(txtExpenseId);
    deleteExpenseFrame.add(btnDelete);
    deleteExpenseFrame.setVisible(true);
  }

  // ExpenseManager class to handle database interaction
  public static class ExpenseManager {
    private static final String URL = "jdbc:mysql://localhost:3306/ex"; // Change to your database
    private static final String USER = "root"; // Your MySQL username
    private static final String PASSWORD = "password"; // Your MySQL password

    public boolean addExpense(double amount, String description, java.sql.Timestamp date) {
        int expenseIdToUse = getAvailableExpenseId();
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD)) {
```

```
            String query = "INSERT INTO expenses (expenseId, amount, description,
date) VALUES (?, ?, ?, ?)";
            try (PreparedStatement stmt = conn.prepareStatement(query)) {
                stmt.setInt(1, expenseIdToUse); // Set the available expense ID
                stmt.setDouble(2, amount);
                stmt.setString(3, description);
                stmt.setTimestamp(4, date);
                int rowsAffected = stmt.executeUpdate();
                return rowsAffected > 0;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }

    private int getAvailableExpenseId() {
        List<Integer> usedIds = new ArrayList<>();
        try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
            String query = "SELECT expenseId FROM expenses ORDER BY expenseId
ASC";
            try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(query)) {
                while (rs.next()) {
                    usedIds.add(rs.getInt("expenseId"));
```

```
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }

    // Find the smallest available ID
    int availableId = 1;
    while (usedIds.contains(availableId)) {
        availableId++;
    }
    return availableId;
}

public boolean deleteExpense(int expenseId) {
    try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
        String query = "DELETE FROM expenses WHERE expenseId = ?";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, expenseId);
            int rowsAffected = stmt.executeUpdate();
            return rowsAffected > 0;
        }
    } catch (SQLException e) {
        e.printStackTrace();
```

```
            return false;
        }
    }

    public List<Expense> getAllExpenses() {
        List<Expense> expenses = new ArrayList<>();
        try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
            String query = "SELECT * FROM expenses";
            try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(query)) {
                while (rs.next()) {
                    int expenseId = rs.getInt("expenseId");
                    double amount = rs.getDouble("amount");
                    String description = rs.getString("description");
                    Timestamp date = rs.getTimestamp("date");
                    expenses.add(new Expense(expenseId, amount, description, date));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return expenses;
    }
```

```
    public double getTotalExpenses() {
        double total = 0;
        try (Connection conn = DriverManager.getConnection(URL, USER,
PASSWORD)) {
            String query = "SELECT SUM(amount) AS total FROM expenses";
            try (Statement stmt = conn.createStatement(); ResultSet rs =
stmt.executeQuery(query)) {
                if (rs.next()) {
                    total = rs.getDouble("total");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return total;
    }
}

public static class Expense {
    private final int expenseId;
    private final double amount;
    private final String description;
    private final Timestamp date;
```

```
    public Expense(int expenseId, double amount, String description, Timestamp
date) {
        this.expenseId = expenseId;
        this.amount = amount;
        this.description = description;
        this.date = date;
    }

    public int getExpenseId() {
        return expenseId;
    }

    public double getAmount() {
        return amount;
    }

    public String getDescription() {
        return description;
    }

    public Timestamp getDate() {
        return date;
    }
  }
}
```

SQL Code:

```
CREATE DATABASE ex;


USE ex;


-- Create the table for storing expenses
CREATE TABLE  expenses (
    expenseId INT PRIMARY KEY,
    amount DECIMAL(10, 2) NOT NULL,
    description VARCHAR(255) NOT NULL,
    date TIMESTAMP NOT NULL);
```