RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM-602 105

CS23333 OOPS Using Java

Laboratory Record Note Book

Name :

Year / Branch / Section :

University Register No. : ..

College Roll No. : .

Semester : .

Academic Year : .

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

BONAFIDE CERTIFICATE

Name: ……

Academic Year: …… **Semester:** …… **Branch:** ……

Register No.

Certified that this is the bonafide record of work done by the above student in the.....Laboratory during the academic year 2025- 2026

Signature of Faculty in-charge

Submitted for the Practical Examination held on……

Internal Examiner **External Examiner**

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR
1		I/O, Data Types, Operators	
2		Control Structures	
3		Arrays	
4		Strings	
5		Classes & Objects	
6		Inheritance	
7		Interface	
8		Exceptions	
9		Collections	
10		Collections	
11		Project	
12		Lambda	

AMOUNT DENOTING SYSTEM

A MINI-PROJECT REPORT

Submitted by

DEEPIKA A 240701102

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"AMOUNT DENOTING SYSTEM"** is the bonafide work of **"DEEPIKA A"** who carried out the project work under my supervision.

SIGNATURE

Dr. B. DEEPA,

ASSISTANT PROFESSOR SG

Dept. of Computer Science and Engg,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The *Amount Denoting System* is a computational framework designed to automatically convert numerical values into their corresponding textual or verbal representations. This system ensures consistency, accuracy, and efficiency in processing monetary or quantitative data across various applications such as banking, accounting, invoicing, and document automation. By leveraging algorithmic parsing and linguistic rules, the system interprets numeric inputs (e.g., 1250.75) and generates their precise word equivalents (e.g., “One thousand two hundred fifty rupees and seventy-five paise”). The proposed model supports multilingual and regional currency formats, enhancing its adaptability in global financial and administrative contexts. Additionally, the system minimizes manual errors in official documentation and digital transactions. Through an optimized algorithm and modular design, the Amount Denoting System offers a scalable and reliable solution for integrating number-to-word conversion into modern software ecosystems.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Dr. B. DEEPA .** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. DEEPIKA A

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Amount Denoting System is designed to automate this conversion process efficiently and accurately. It accepts a numerical input and generates its equivalent in words, following proper linguistic and grammatical rules. The system can handle both integer and decimal values, accommodating different currency formats such as rupees, dollars, euros, etc. By implementing logical algorithms and modular programming techniques, the system provides a user-friendly interface and reliable performance.

1.2 SCOPE OF THE WORK

The Amount Denoting System is designed to provide an efficient and automated solution for converting numerical amounts into their equivalent words accurately. The scope of this project covers the design, development, and implementation of a software module capable of handling different numerical values, including integers and decimal fractions, and expressing them in linguistically correct textual form.

1.3 PROBLEM STATEMENT

In many financial, administrative, and commercial processes, numerical amounts are frequently represented in documents such as invoices, cheques, receipts, and reports. However, relying solely on numeric

representation can lead to confusion, misinterpretation, or even fraudulent alterations. For instance, a figure like “1500” can easily be modified or misunderstood whereas its textual representation “One thousand five hundred only” provides clarity and authenticity.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim

The main aim of the **Amount Denoting System** project is to develop an automated software application that accurately converts numerical amounts into their corresponding word representations. The system is designed to ensure accuracy, consistency, and efficiency in financial and official documentation, reducing manual effort and minimizing human error.

Objectives

The specific objectives of the project are as follows:

1. **To design and develop** a user-friendly system that accepts numerical input and generates the equivalent amount in words.

2. **To ensure accuracy and reliability** in the conversion process, handling both integer and decimal values.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	AMD
Memory Size	:	16GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 11
Front – End	:	JAVA
Back – End	:	SQL
Language	:	JAVA, SQL

CHAPTER 3

MODULE DESCRIPTION

This application consists of two modules. When the program runs, it will ask for a confirmation to the login window. The person who interacts can login as an Administrator or as a User. The description of the modules are as follows:

1. Admin login

When the person who interacts tries to login as Admin then he needs to login with his username and password. The administrator only has the power to change and manipulate the data in the database.

2. User login

When the person tries to login as a user then he/she will be prombted to enter
the number of symptoms and the final result will be printed in table

CHAPTER 4

SAMPLE CODING

```
import javax.swing.*;

import javax.swing.table.DefaultTableModel;

import javax.swing.border.EmptyBorder;

import java.awt.*;

import java.awt.event.*;

import java.sql.*;

import java.io.FileWriter;

import java.io.IOException;
```

```
public class DonationManagerApp extends JFrame {

    private Connection conn;

    private JPanel mainPanel;

    private JTabbedPane tabbedPane;

    private JTable table;

    private DefaultTableModel model;

    private JTextField donorNameField, amountField, searchField;

    private JLabel totalDonorsLabel, totalAmountLabel, highestDonationLabel,
avgDonationLabel;

    private boolean darkTheme = false;



    private static final Color PRIMARY = new Color(33, 150, 243);  // Blue

    private static final Color SECONDARY = new Color(240, 248, 255); // Soft
white-blue



    public DonationManagerApp() {

        setTitle(" Donation Management System");

        setSize(950, 650);
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setLayout(new BorderLayout());
        connectToDB();
        showLoginPage();
    }

    private void connectToDB() {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            conn =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"giri");
        } catch (Exception e) {
            JOptionPane.showMessageDialog(this, "Database connection failed: " +
e.getMessage());
        }
    }

    private void showLoginPage() {
```

```
        JPanel loginPanel = new JPanel() {
            @Override
            protected void paintComponent(Graphics g) {
                super.paintComponent(g);
                Graphics2D g2 = (Graphics2D) g;
                GradientPaint gp = new GradientPaint(0, 0, PRIMARY, getWidth(),
getHeight(), Color.WHITE);
                g2.setPaint(gp);
                g2.fillRect(0, 0, getWidth(), getHeight());
            }
        };
        loginPanel.setLayout(new GridBagLayout());

        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10);

        JLabel title = new JLabel("Donation Management Login",
SwingConstants.CENTER);
        title.setFont(new Font("Segoe UI", Font.BOLD, 26));
```

```
title.setForeground(Color.WHITE);


JLabel userLabel = new JLabel("Username:");

JTextField userField = new JTextField(18);

JLabel passLabel = new JLabel("Password:");

JPasswordField passField = new JPasswordField(18);


JButton loginBtn = new JButton("Login");

styleButton(loginBtn);


gbc.gridx = 0; gbc.gridy = 0; gbc.gridwidth = 2;

loginPanel.add(title, gbc);


gbc.gridwidth = 1;

gbc.gridy++;

loginPanel.add(userLabel, gbc);

gbc.gridx = 1;

loginPanel.add(userField, gbc);
```

```
        gbc.gridx = 0; gbc.gridy++;
        loginPanel.add(passLabel, gbc);
        gbc.gridx = 1;
        loginPanel.add(passField, gbc);


        gbc.gridy++;
        gbc.gridx = 0; gbc.gridwidth = 2;
        loginPanel.add(loginBtn, gbc);


        loginBtn.addActionListener(e -> {
            String user = userField.getText();
            String pass = new String(passField.getPassword());
            try {
                PreparedStatement ps = conn.prepareStatement("SELECT * FROM users
WHERE username=? AND password=?");
                ps.setString(1, user);
                ps.setString(2, pass);
                ResultSet rs = ps.executeQuery();
```

```
            if (rs.next()) {
                getContentPane().removeAll();
                showMainPage();
                revalidate();
                repaint();
            } else {
                JOptionPane.showMessageDialog(this, "Invalid login!");
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    });

    add(loginPanel, BorderLayout.CENTER);
}

private void showMainPage() {
    tabbedPane = new JTabbedPane();
```

```
tabbedPane.setFont(new Font("Segoe UI", Font.PLAIN, 14));


tabbedPane.add(" Add Donation", createAddDonationPage());

tabbedPane.add(" View Donations", createViewDonationsPage());

tabbedPane.add(" Search Donation", createSearchPage());

tabbedPane.add(" Summary Dashboard", createSummaryPage());

tabbedPane.add(" Leaderboard", createLeaderboardPage());


JPanel topPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

topPanel.setBackground(SECONDARY);


JButton refreshBtn = new JButton(" Refresh");

JButton themeBtn = new JButton(" Toggle Theme");

JButton exportBtn = new JButton(" Export CSV");


styleButton(refreshBtn);

styleButton(themeBtn);

styleButton(exportBtn);
```

```
    topPanel.add(refreshBtn);
    topPanel.add(themeBtn);
    topPanel.add(exportBtn);

    refreshBtn.addActionListener(e -> refreshTables());
    themeBtn.addActionListener(e -> toggleTheme());
    exportBtn.addActionListener(e -> exportToCSV());

    add(topPanel, BorderLayout.NORTH);
    add(tabbedPane, BorderLayout.CENTER);
}

private JPanel createAddDonationPage() {
    JPanel panel = new JPanel(new GridBagLayout());
    panel.setBackground(SECONDARY);
    panel.setBorder(new EmptyBorder(50, 150, 50, 150));
```

```
GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(15, 15, 15, 15);

gbc.fill = GridBagConstraints.HORIZONTAL;


JLabel nameLabel = new JLabel("Donor Name:");

nameLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));

donorNameField = new JTextField(20);

styleField(donorNameField);


JLabel amountLabel = new JLabel("Donation Amount:");

amountLabel.setFont(new Font("Segoe UI", Font.BOLD, 14));

amountField = new JTextField(20);

styleField(amountField);


JButton addBtn = new JButton(" Add Donation");

styleButton(addBtn);


gbc.gridx = 0; gbc.gridy = 0; panel.add(nameLabel, gbc);
```

```
        gbc.gridx = 1; panel.add(donorNameField, gbc);


        gbc.gridx = 0; gbc.gridy = 1; panel.add(amountLabel, gbc);
        gbc.gridx = 1; panel.add(amountField, gbc);


        gbc.gridx = 0; gbc.gridy = 2; gbc.gridwidth = 2;
        panel.add(addBtn, gbc);


        addBtn.addActionListener(e -> {
            String name = donorNameField.getText();
            String amtText = amountField.getText();
            if (name.isEmpty() || amtText.isEmpty()) {
                JOptionPane.showMessageDialog(this, "Please enter all fields!");
                return;
            }
            try {
                double amt = Double.parseDouble(amtText);
                PreparedStatement ps = conn.prepareStatement("INSERT INTO donors
(donor_name, amount) VALUES (?, ?)");
```

```
            ps.setString(1, name);
            ps.setDouble(2, amt);
            ps.executeUpdate();
            JOptionPane.showMessageDialog(this, "Donation added successfully!");
            donorNameField.setText("");
            amountField.setText("");
            refreshTables();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    });
    return panel;
}

private JPanel createViewDonationsPage() {
    JPanel panel = new JPanel(new BorderLayout());
    model = new DefaultTableModel(new String[]{"ID", "Donor Name",
"Amount", "Date", "Category"}, 0);
    table = new JTable(model);
```

```
    table.setRowHeight(28);
    table.getTableHeader().setFont(new Font("Segoe UI", Font.BOLD, 13));
    refreshTables();
    panel.add(new JScrollPane(table), BorderLayout.CENTER);
    return panel;
}

private JPanel createSearchPage() {
    JPanel panel = new JPanel(new BorderLayout());
    panel.setBackground(SECONDARY);

    JPanel top = new JPanel(new FlowLayout());
    top.setBackground(SECONDARY);

    searchField = new JTextField(15);
    styleField(searchField);
    JButton searchBtn = new JButton("Search");
    styleButton(searchBtn);
```

```
        top.add(new JLabel("Enter donor name:"));
        top.add(searchField);
        top.add(searchBtn);
        panel.add(top, BorderLayout.NORTH);

        DefaultTableModel searchModel = new DefaultTableModel(new
String[]{"ID", "Name", "Amount", "Date"}, 0);
        JTable searchTable = new JTable(searchModel);
        searchTable.setRowHeight(26);
        panel.add(new JScrollPane(searchTable), BorderLayout.CENTER);

        searchBtn.addActionListener(e -> {
            searchModel.setRowCount(0);
            try {
                String query = "SELECT * FROM donors WHERE
LOWER(donor_name) LIKE ?";
                PreparedStatement ps = conn.prepareStatement(query);
                ps.setString(1, "%" + searchField.getText().toLowerCase() + "%");
```

```
            ResultSet rs = ps.executeQuery();
            while (rs.next()) {
                searchModel.addRow(new Object[]{
                        rs.getInt(1), rs.getString(2), rs.getDouble(3), rs.getDate(4)
                });
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    });
    return panel;
}

private JPanel createSummaryPage() {
    JPanel panel = new JPanel(new GridLayout(4, 1, 10, 10));
    panel.setBackground(SECONDARY);
    totalDonorsLabel = new JLabel();
    totalAmountLabel = new JLabel();
```

```
        highestDonationLabel = new JLabel();
        avgDonationLabel = new JLabel();

        for (JLabel lbl : new JLabel[]{totalDonorsLabel, totalAmountLabel,
highestDonationLabel, avgDonationLabel}) {
            lbl.setFont(new Font("Segoe UI", Font.BOLD, 16));
            lbl.setForeground(Color.DARK_GRAY);
            panel.add(lbl);
        }
        updateSummary();
        return panel;
    }

    private JPanel createLeaderboardPage() {
        JPanel panel = new JPanel(new BorderLayout());
        DefaultTableModel leaderboardModel = new DefaultTableModel(new
String[]{"Rank", "Donor Name", "Total Amount", "Category"}, 0);
        JTable leaderboardTable = new JTable(leaderboardModel);
        leaderboardTable.setRowHeight(26);
```

```
        panel.add(new JScrollPane(leaderboardTable), BorderLayout.CENTER);



        try {

            Statement st = conn.createStatement();

            ResultSet rs = st.executeQuery("SELECT donor_name, SUM(amount) total
FROM donors GROUP BY donor_name ORDER BY total DESC");

            int rank = 1;

            while (rs.next() && rank <= 5) {

                double amt = rs.getDouble("total");

                String category = amt < 500 ? "Bronze" : (amt <= 2000 ? "Silver" :
"Gold");

                leaderboardModel.addRow(new Object[]{rank++, rs.getString(1), amt,
category});

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

        return panel;

    }
```

```
private void styleField(JTextField field) {
    field.setFont(new Font("Segoe UI", Font.PLAIN, 14));
    field.setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createLineBorder(PRIMARY, 1),
            BorderFactory.createEmptyBorder(6, 8, 6, 8)
    ));
}

private void styleButton(JButton btn) {
    btn.setBackground(PRIMARY);
    btn.setForeground(Color.WHITE);
    btn.setFocusPainted(false);
    btn.setFont(new Font("Segoe UI", Font.BOLD, 14));
    btn.setBorder(BorderFactory.createEmptyBorder(8, 16, 8, 16));
}

private void refreshTables() {
    model.setRowCount(0);
```

```
        try {
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM donors ORDER BY
donor_id DESC");
            while (rs.next()) {
                double amt = rs.getDouble("amount");
                String cat = amt < 500 ? "Bronze" : (amt <= 2000 ? "Silver" : "Gold");
                model.addRow(new Object[]{rs.getInt(1), rs.getString(2), amt,
rs.getDate(4), cat});
            }
            updateSummary();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void updateSummary() {
        try {
            Statement st = conn.createStatement();
```

```
            ResultSet rs1 = st.executeQuery("SELECT COUNT(*), SUM(amount),
AVG(amount) FROM donors");

            if (rs1.next()) {

                totalDonorsLabel.setText(" Total Donors: " + rs1.getInt(1));

                totalAmountLabel.setText(" Total Amount: ₹" + rs1.getDouble(2));

                avgDonationLabel.setText(" Average Donation: ₹" +
String.format("%.2f", rs1.getDouble(3)));

            }



            ResultSet rs2 = st.executeQuery("SELECT donor_name, MAX(amount)
FROM donors");

            if (rs2.next()) {

                highestDonationLabel.setText(" Highest Donation: ₹" + rs2.getDouble(2)
+ " by " + rs2.getString(1));

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }
```

```
    private void toggleTheme() {
        darkTheme = !darkTheme;
        Color bg = darkTheme ? new Color(45, 45, 45) : SECONDARY;
        getContentPane().setBackground(bg);
        SwingUtilities.updateComponentTreeUI(this);
    }


    private void exportToCSV() {
        try (FileWriter fw = new FileWriter("donations_export.csv")) {
            fw.write("ID,Donor Name,Amount,Date\n");
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery("SELECT * FROM donors");
            while (rs.next()) {
                fw.write(rs.getInt(1) + "," + rs.getString(2) + "," + rs.getDouble(3) + "," +
rs.getDate(4) + "\n");
            }
            JOptionPane.showMessageDialog(this, "Exported to donations_export.csv
successfully!");
        } catch (IOException | SQLException e) {
```

```
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> new DonationManagerApp().setVisible(true));
    }
}
```

Sample 2

Sample 2 depicts the booking part of the code, where it displays booking details and enter user data and store it in database

CHAPTER 5

SCREEN SHOTS

Fig 5.1 login page

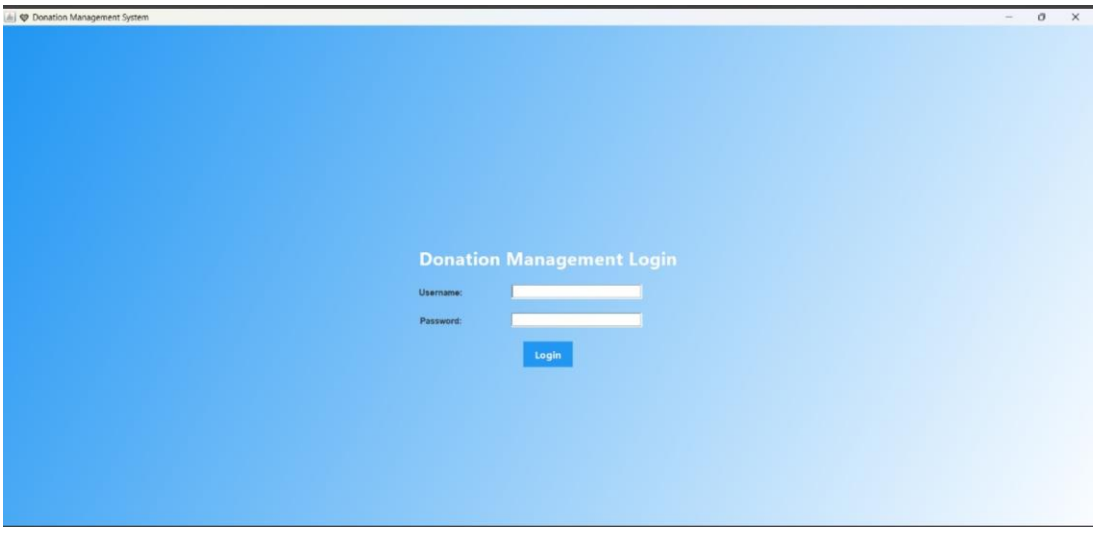


Fig 5.2 Donating Log

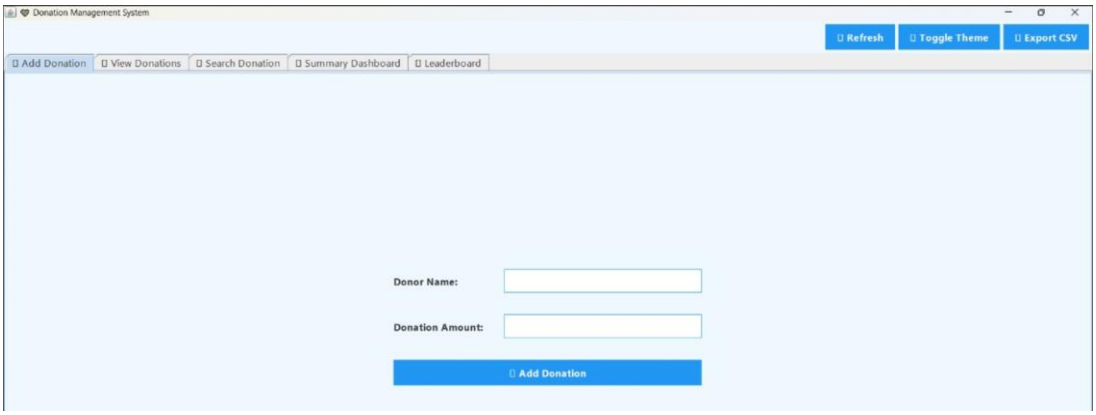


Fig 5.3 Donor Details

Fig 5.4 Search Donation

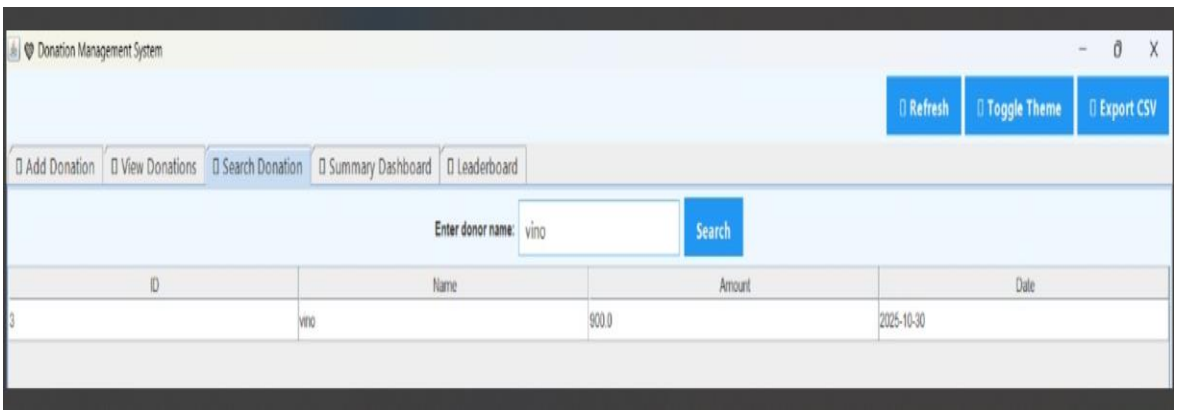


Fig 5.5 Summary Dashbord

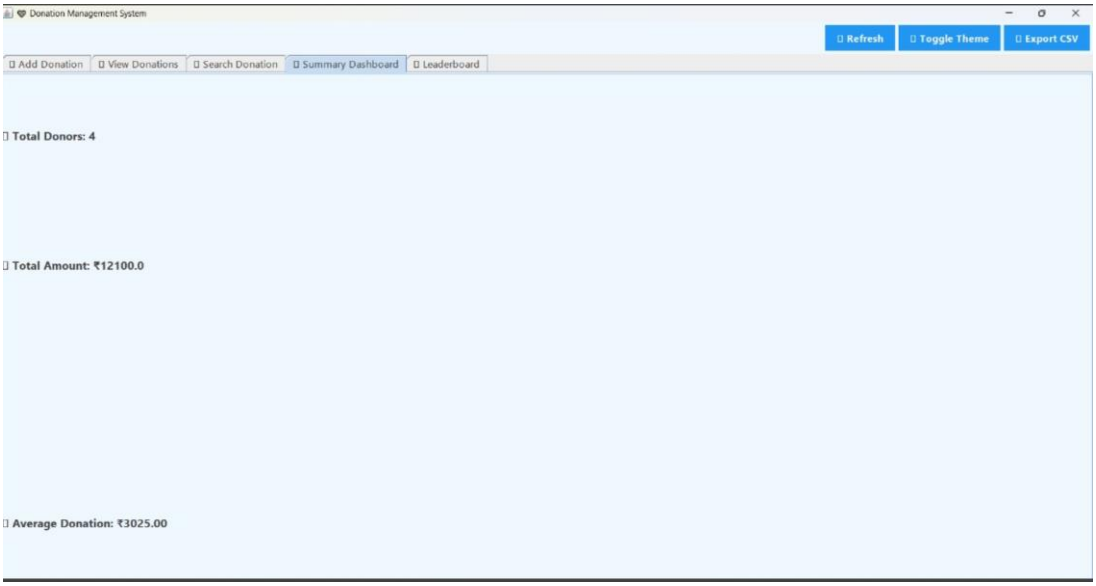
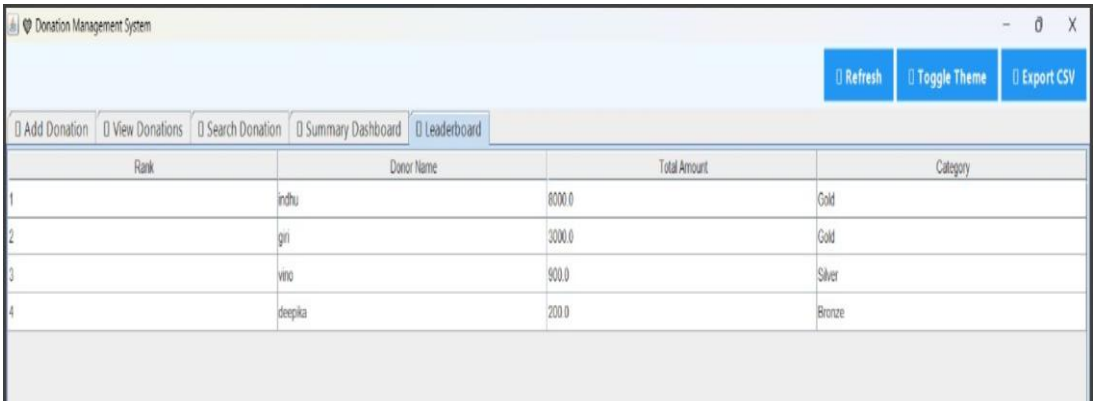


Fig 5.6 Database creation



CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The **Amount Denoting System** provides an effective and reliable solution for converting numerical amounts into their corresponding word representations. Through the automation of this process, the system minimizes human errors, saves

time, and ensures accuracy in financial and official documentation. It plays a vital role in areas such as banking, accounting, billing, and report generation, where precision and clarity of monetary values are essential.

REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.tutorialspoint.com/sqlite/index.htm
3. https://www.wikipedia.org/
4. https://www.learnpython.org/
5. https://www.codecademy.com/learn/learn-python