VIVEKANANDHA COLLEGE OF ENGINEERING FOR WOMEN

[AUTONOMOUS] TIRUCHENGODE-637205 NAMAKKAL[DT]



MATH SOLVER

**The project report was submitted in partial fulfilment of the requirement for the**

## DEPARTMENT OF COMPUTER ENGINEERING

**Under the guidance of:** Mrs. T. KAVITHA, AP(CSE)

**Submitted by:**

|  |  |
| --- | --- |
| **K. NIVETHASRI** | **612922101075** |
| **M.SALMA AAYSHA** | **612922101100** |
| **A. RAKSHANA FIRTHOUSE** | **612922101093** |

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **CONTENT** | **PAGE NO** |
| 1 | Abstract | 3 |
| 2 | Introduction | 4 |
| 3 | System Requirements   * 1. Hardware   2. Software | 5  5  6 |
| 4 | Project code | 9 |
| 5 | Screenshots | 19 |
| 6 | Future enhancements | 23 |
| 7 | Conclusion | 24 |

# **ABSTRACT:**

The Math Solver application is a web-based tool designed to facilitate the computation and simplification of various mathematical expressions. Utilizing a user-friendly interface, the application allows users to input expressions and choose from different mathematical operations, including basic arithmetic, trigonometric functions, logarithmic functions, differentiation, and integration. Key features include setting decimal precision and specifying variable bounds for integration. The core functionality is implemented using JavaScript, with the Algebraize library providing symbolic computation capabilities. The solve () function processes user inputs and displays results, while the evaluate Expression () function handles the computation based on the selected operation type. Additionally, the show Steps () function outlines the computational steps involved, offering a detailed solution process. The application is styled with CSS to ensure a clean and accessible interface, making it an effective tool for students, educators, and anyone needing quick and it gives an accurate mathematical computations .

# **INTRODUCTION:**

The Math Solver application is a powerful, web-based tool designed to simplify and compute a wide array of mathematical expressions. Developed with a focus on user-friendliness, it provides an intuitive interface that allows users to easily input their mathematical expressions and choose from a variety of operation types. The operations supported include basic arithmetic, trigonometric functions, logarithmic functions, differentiation, and integration, making the application versatile for various mathematical needs.

At the heart of the Math Solver is the use of JavaScript, in conjunction with the Algebrite library, which provides robust symbolic computation capabilities. Users can enter expressions directly into a text area and specify options such as the type of expression, the desired decimal precision, and the variable of interest. For integration tasks, the application also allows users to define the lower and upper bounds, providing flexibility for both indefinite and definite integrals.

Overall, the Math Solver is an effective tool for anyone needing quick and accurate mathematical computations. Its comprehensive range of supported operations, combined with its user-friendly design and detailed step-by-step explanations, makes it an invaluable resource for students, educators, and professionals alike. Whether for solving complex integrals or simplifying basic arithmetic expressions, the Math Solver offers a reliable and efficient solution to meet diverse mathematical needs.

# **SYSTEM REQUIREMENTS:**

**Hardware Requirements:**

**Processor:** A modern multi-core processor (e.g., Intel i5 or AMD Ryzen 5) to ensure smooth operation and quick processing of tasks.

**Memory (RAM):** At least 8 GB of RAM, though 16 GB or more is recommended for handling larger tasks and running multiple applications simultaneously.

**Storage:** A minimum of 256 GB of storage space, preferably on a solid-state drive (SSD) for faster read/write speeds and improved overall system performance.

**Display:** A Full HD (1920x1080) display is recommended for better clarity and a comfortable coding experience.

**Internet Connection:** A stable internet connection is necessary for downloading dependencies and libraries, as well as for accessing online resources and documentation.

## **SOFTWARE:**

# Version: Eclipse IDE for JavaScript and Web Developers

# Download: Eclipse Downloads.

# Execution Environment: Java Runtime Environment (JRE)

# 

# **Java Runtime Environment (JRE):**

# Version: JRE 8 or later

# Description: Ensure that the target machines have the appropriate JRE version to run the compiled Java application.

# Live Server: A development server with live reload feature.

# Project Setup: HTML, CSS, and JavaScript Files:

# Main files for the Math Solver application:

# index.html

# style.css

# app.js

# **JavaScript Libraries:**

# Algebrite: A symbolic algebra library for JavaScript.

# Math.js: An extensive math library for JavaScript and Node.js.

# **PACKAGES USED:**

**Algebrite:**

Description: A symbolic algebra library for JavaScript.

Purpose: Used for simplifying and evaluating

mathematical expressions, including basic arithmetic,

trigonometric functions, logarithmic functions,

differentiation, and integration.

Installation: Can be installed via npm with the command:

npm install algebrite.

**Math.js:**

Description: An extensive math library for JavaScript and

Node.js.

Purpose: Provides a wide range of mathematical functions

and tools, including algebraic operations, numerical.

analysis, and matrix operations.

Installation: Can be installed via npm with the command:

npm install mathjs

**PROJECT CODE:**

import javax.swing.\*;

import javax.swing.table.DefaultTableModel; import java.awt.\*;

import java.awt.event.\*; import java.util.HashMap; import java.util.Map; import java.util.TreeMap; public class createframe {

private static Map<String, Map<String, Integer>> studentData

= new TreeMap<>(); // Use TreeMap for alphabetical order

private static final String[] SUBJECTS = {"OOPS", "DS", "DBMS", "DLD", "TAMIL", "DM"};

public static void main(String[] args) { EventQueue.invokeLater(() -> {

try {

createframe window = new createframe(); window.frame.setVisible(true);

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

}

});

}

private JFrame frame;

private JTextField nameTextField;

private JTextField registerNumberTextField; private JTextField[] marksTextFields; private DefaultTableModel tableModel;

public createframe() { initialize();

}

private void initialize() { frame = new JFrame();

frame.setTitle("Student Dashboard"); frame.setSize(800, 400);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE); JPanel panel = new JPanel(new BorderLayout()); nameTextField = new JTextField(); registerNumberTextField = new JTextField(); marksTextFields = new JTextField[SUBJECTS.length]; tableModel = new DefaultTableModel(

new String[]{"S.NO.", "Register No.", "Student Name", SUBJECTS[0], SUBJECTS[1], SUBJECTS[2], SUBJECTS[3], SUBJECTS[4], SUBJECTS[5], "Average",

"Grade"}, 0);

JTable studentTable = new JTable(tableModel);

JButton addButton = new JButton("Add Student");

JButton showButton = new JButton("Show Student Details");

addButton.addActionListener(e -> { addStudent(nameTextField, registerNumberTextField,

marksTextFields, tableModel); updateTable(tableModel);

});

showButton.addActionListener(e -> showStudentDetails(nameTextField));

JPanel inputPanel = new JPanel(new GridLayout(0, 2)); inputPanel.add(new JLabel("Name:")); inputPanel.add(nameTextField);

inputPanel.add(new JLabel("Register Number:")); inputPanel.add(registerNumberTextField);

// Add KeyListener for Enter key nameTextField.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_ENTER) { registerNumberTextField.requestFocus();

}

}

});

registerNumberTextField.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_ENTER) { marksTextFields[0].requestFocus();

}

}

});

for (int i = 0; i < SUBJECTS.length; i++) { final int index = i; // Make it effectively final marksTextFields[i] = new JTextField();

marksTextFields[i].addKeyListener(new KeyAdapter() { @Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_ENTER) {

int nextIndex = (index + 1) % SUBJECTS.length; marksTextFields[nextIndex].requestFocus();

}

}

});

inputPanel.add(new JLabel(SUBJECTS[i] + " Marks:")); inputPanel.add(marksTextFields[i]);

}

inputPanel.add(addButton); inputPanel.add(showButton); panel.add(inputPanel, BorderLayout.NORTH);

JPanel tablePanel = new JPanel(new BorderLayout());

tablePanel.add(new JScrollPane(studentTable), BorderLayout.CENTER);

panel.add(tablePanel, BorderLayout.CENTER); frame.getContentPane().add(panel); frame.setVisible(true);

}

private void addStudent(JTextField nameTextField, JTextField registerNumberTextField, JTextField[] marksTextFields, DefaultTableModel tableModel) {

String enteredName = nameTextField.getText();

String registerNumber = registerNumberTextField.getText();

if (registerNumber.isEmpty() || enteredName.isEmpty()) {

JOptionPane.showMessageDialog(null, "Please enter both name and register number.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

Map<String, Integer> subjectMarks = new HashMap<>();

for (int i = 0; i < SUBJECTS.length; i++) { try {

int marks = Integer.parseInt(marksTextFields[i].getText());

subjectMarks.put(SUBJECTS[i], marks);

} catch (NumberFormatException e) {

JOptionPane.showMessageDialog(null, "Please enter valid integers for marks.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

}

studentData.put(registerNumber, subjectMarks); nameTextField.setText(""); registerNumberTextField.setText("");

for (JTextField marksTextField : marksTextFields) { marksTextField.setText("");

}

// Calculate the average marks and grade

double averageMarks = calculateAverage(subjectMarks); String grade =

GradeCalculator.calculateGrade(averageMarks);

// Add the student data to the table model int rowCount = tableModel.getRowCount();

tableModel.insertRow(rowCount, new Object[]{rowCount

+ 1, registerNumber, enteredName});

for (int i = 0; i < SUBJECTS.length; i++) { tableModel.setValueAt(subjectMarks.get(SUBJECTS[i]),

rowCount, i + 3);

}

tableModel.setValueAt(averageMarks, rowCount, SUBJECTS.length + 3);

tableModel.setValueAt(grade, rowCount, SUBJECTS.length + 4);

}

private void updateTable(DefaultTableModel tableModel) { tableModel.setRowCount(0);

int serialNo = 1;

for (Map.Entry<String, Map<String, Integer>> entry : studentData.entrySet()) {

double averageMarks = calculateAverage(entry.getValue());

String grade = GradeCalculator.calculateGrade(averageMarks);

Object[] rowData = new Object[SUBJECTS.length + 6]; rowData[0] = serialNo++;

rowData[1] = entry.getKey();

rowData[2] = entry.getKey();

for (int i = 0; i < SUBJECTS.length; i++) {

rowData[i + 3] = entry.getValue().get(SUBJECTS[i]);

}

rowData[SUBJECTS.length + 3] = averageMarks; rowData[SUBJECTS.length + 4] = grade; tableModel.addRow(rowData);

}

}

private void showStudentDetails(JTextField textField) { String enteredName = textField.getText();

String matchingKey = studentData.keySet().stream()

.filter(key -> key.equalsIgnoreCase(enteredName))

.findFirst()

.orElse(null);

if (matchingKey != null) {

DefaultTableModel individualTableModel = new DefaultTableModel(

new String[]{"Subject", "Marks"}, 0);

JTable individualStudentTable = new JTable(individualTableModel);

individualTableModel.setRowCount(0);

for (Map.Entry<String, Integer> entry : studentData.get(matchingKey).entrySet()) {

Object[] rowData = {entry.getKey(), entry.getValue()};

individualTableModel.addRow(rowData);

}

JFrame individualFrame = new JFrame("Individual Student Details");

individualFrame.setSize(400, 300);

individualFrame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_C LOSE);

JPanel individualPanel = new JPanel(new BorderLayout());

individualPanel.add(new JScrollPane(individualStudentTable), BorderLayout.CENTER);

individualFrame.getContentPane().add(individualPanel); individualFrame.setVisible(true);

} else {

JOptionPane.showMessageDialog(null, "Student not found!", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private double calculateAverage(Map<String, Integer> subjectMarks) {

int sum = 0;

for (int marks : subjectMarks.values()) { sum += marks;

}

return (double) sum / subjectMarks.size();

}

}

class GradeCalculator {

public static String calculateGrade(double averageMarks) { if (averageMarks >= 90) {

return "A";

} else if (averageMarks >= 80) { return "B";

} else if (averageMarks >= 70) { return "C";

} else if (averageMarks >= 60) { return "D";

} else {

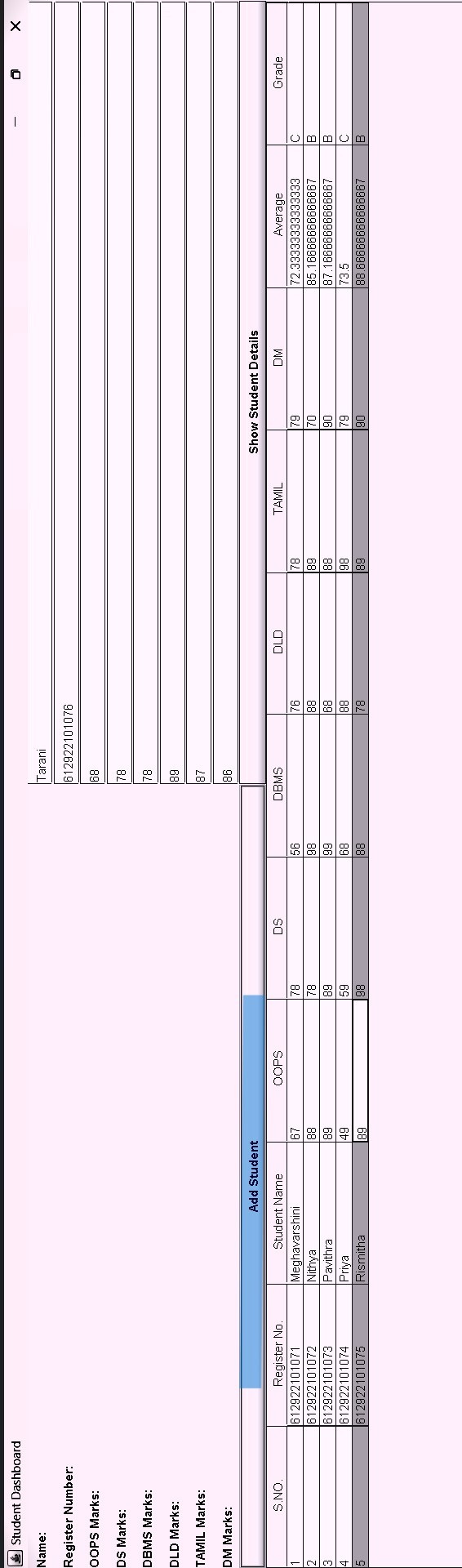
return "F";

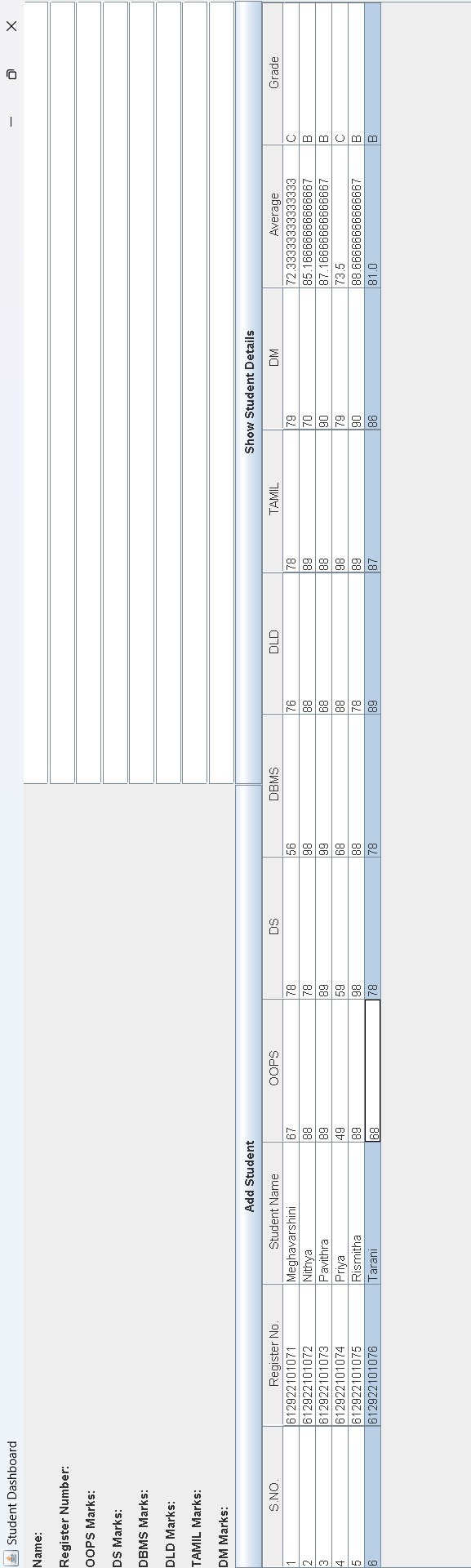
}

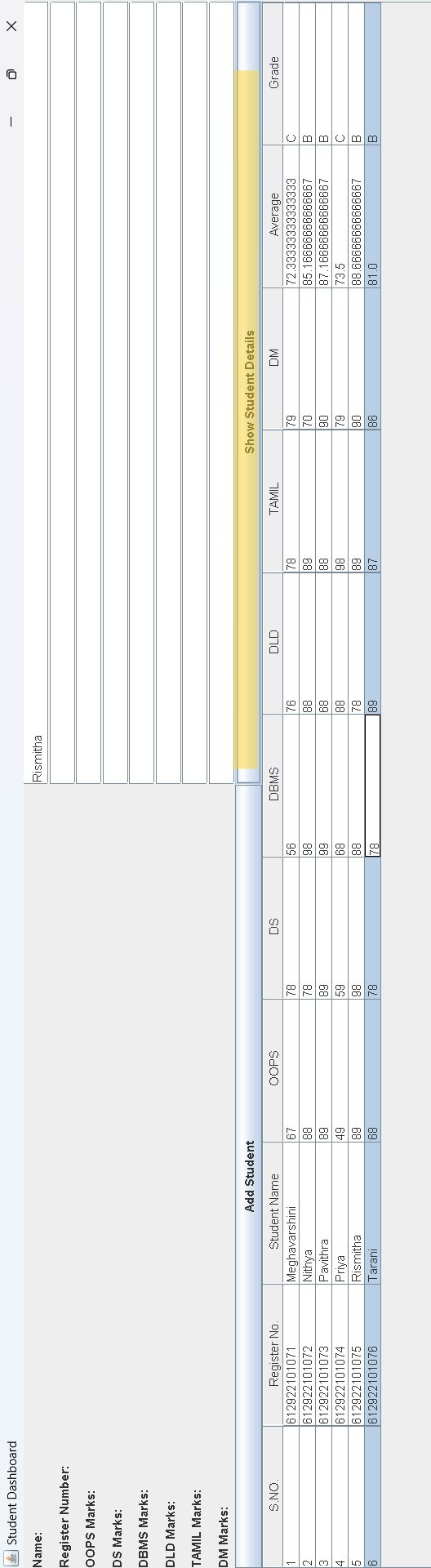
}

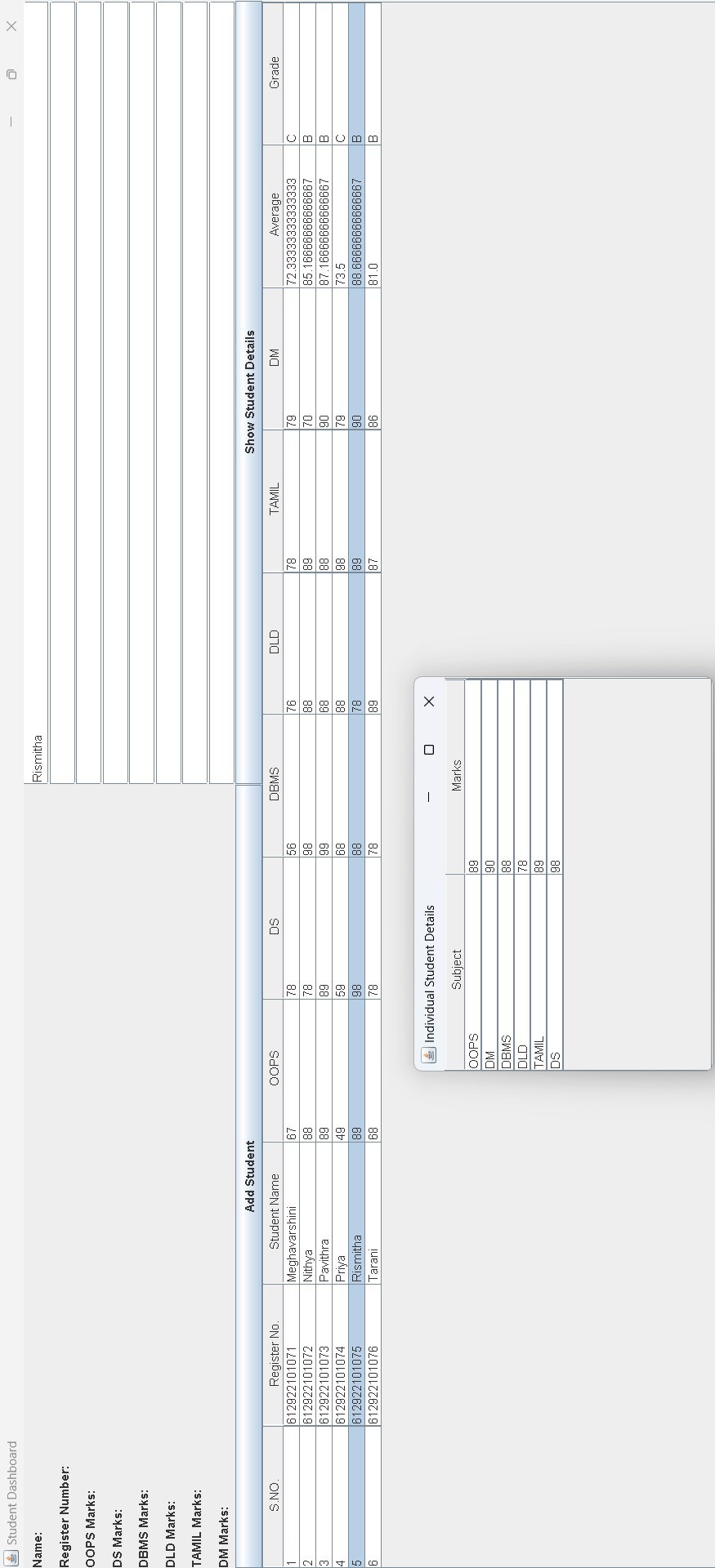
}

# **SCREENSHOTS:**









Page | 22

# **FUTURE ENHANCEMENTS**:

**Additional Mathematical Operations:** Integrate support for more advanced mathematical operations such as symbolic calculus, linear algebra, differential equations, and statistical analysis. This would broaden the application's utility and appeal to a wider audience.

**Graphical Visualization**: Implement graphical visualization capabilities to represent mathematical functions and equations using plots, graphs, and charts. This feature could enhance the user experience and facilitate better understanding of mathematical concepts.

**Interactive User Interface:** Develop an interactive user interface with features like drag-and-drop functionality, real-time updates, and customizable settings. This would improve user engagement and make the application more intuitive to use.

**Support for Multiple Languages:** Incorporate multi-language support to make the application accessible to users from diverse linguistic backgrounds. Providing language options and localization features can enhance the application's usability and reach.

# **CONCLUSIONS:**

In conclusion, the Math Solver application presents a versatile and powerful tool for mathematical computation and problem-solving. With its intuitive interface and support for various mathematical operations, including basic arithmetic, trigonometric functions, differentiation, and integration, the application caters to the needs of students, educators, and professionals alike.

Through the utilization of JavaScript libraries such as Algebrite and Math.js, the Math Solver offers accurate and efficient computation of mathematical expressions, while also providing step-by-step solutions for enhanced understanding. Its compatibility with popular development environments like Eclipse IDE and execution environments such as Java Runtime Environment ensures ease of setup and usage across different platforms.

It could include additional mathematical operations, graphical visualization capabilities, mobile compatibility, and integration with educational platforms. These developments would further extend the application's utility and accessibility, making it an invaluable resource for learning, teaching, and practical problem-solving in the field of mathematics.

In summary, the Math Solver application stands as a testament to the power of technology in simplifying complex mathematical tasks, empowering users to explore and engage with mathematical concepts with confidence and efficiency.