



**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**  
**(AN AUTONOMOUS INSTITUTE)**

**A MINI PROJECT BY:**  
**ARAVINTHAA S 230701032**  
**DARSHAN M 230701061**

**IN PARTIAL FULFILLMENT OF THE AWARD OF THE DEGREE**

**OF**

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**NOVEMBER 2024**

## **BONAFIDE CERTIFICATE**

Certified that this project 'COLLISION TRACKING APP' is the bonafide work of “**ARAVINTHAA S ,  
DARSHAN M** “ who carried out the project work under my supervision.

Submitted for the practical examination held on \_\_\_\_\_

SIGNATURE

Dr.G.Dharani Devi

Assistant Professor(SG), Computer Science and Engineering,  
Rajalakshmi Engineering College (Autonomous), Thandalam,  
Chennai-602105.

INTERNAL EXAMINER

EXTERNAL EXAMINER

## **ABSTRACT**

The Collision Tracking App is a software application developed using Java Swing and JDBC to provide a detailed analysis of space debris and its collision risks. Integrated with a MySQL database, the application enables users to load, filter, analyze, add, update, and delete space debris data.

The system allows real-time interaction with the database to assess collision risks based on debris characteristics like size, speed, and orbit. This app serves as a tool for space researchers to monitor and manage space debris effectively, ensuring data accuracy and user-friendly operations.

This project highlights the practical use of Java Swing for GUI development and JDBC for database connectivity, offering a functional solution to manage and track space debris.

## **TABLE OF CONTENTS**

### **1. INTRODUCTION**

- 1.1 Project Overview
- 1.2 Objective
- 1.3 Scope of the Application
- 1.4 Application Features
- 1.5 Benefits and Use Case

### **2. SYSTEM SPECIFICATIONS**

- 2.1 Hardware Specifications
- 2.2 Software Specifications

### **3. SAMPLE CODE**

- 3.1 GUI Design and Layout
- 3.2 Data Loading Operation
- 3.3 Collision Risk Analysis
- 3.4 Add, Update, and Delete Operations
- 3.5 Filtering Space Debris Data

### **4. SNAPSHOTS**

- 4.1 Home Page
- 4.2 Data Loading
- 4.3 Collision Risk Analysis

4.4 Add, Update, and Delete Features

4.5 Filter by Collision Risk

5. CONCLUSION

6. REFERENCE

## **INTRODUCTION**

### **1.1 Project Overview**

The Collision Tracking App is a Java-based application developed to assist space researchers in tracking and managing space debris. Using Java Swing for GUI and JDBC for database interaction, this tool enables users to analyze and mitigate potential collision risks.

### **1.2 Objective**

The objective of this project is to provide a streamlined solution to:

- Load and visualize space debris data from the database.
- Analyze and filter debris based on collision risk.
- Perform CRUD (Create, Read, Update, Delete) operations efficiently.

### **1.3 Scope of the Application**

This application offers:

- A user-friendly GUI to interact with space debris data.
- Real-time collision risk analysis.
- Comprehensive database integration to store and manage debris information.

### **1.4 Application Features**

- Load and display space debris details.
- Add, update, and delete debris records.
- Filter debris by collision risk (low/high).
- Analyze collision risk using real-time database data.

## **1.5 Benefits and Use Case**

### **Benefits:**

- **Real-Time Analysis:** The application allows users to analyze collision risks dynamically by pulling updated data from the database.
- **User-Friendly Interface:** Designed with a simple and intuitive GUI, making it accessible even for non-technical users.
- **Custom Filtering:** Provides the ability to filter data based on collision risk levels, ensuring precise risk assessment.
- **Efficient Management:** Enables space research teams to effectively track and manage debris, minimizing collision risks.

### **Use Case:**

The Collision Tracking App is particularly useful for:

- **Space Agencies:** To monitor debris and mitigate collision risks for satellites and space stations.
- **Academic Research:** Serves as a tool for students and researchers studying orbital debris and its impact.
- **Real-Time Simulations:** Can be extended for simulations in space mission planning and collision avoidance strategies.

## **2. SYSTEM SPECIFICATIONS**

### **2.1 Hardware Specifications**

- Processor: Intel i5 or equivalent
- Memory: Minimum 4 GB RAM
- Storage: At least 500 GB of free disk space
- Display: 1024 x 768 resolution or higher

### **2.2 Software Specifications**

- Programming Language: Java (JDK 11 or higher)
- Database: MySQL 8.0
- Development Environment: IntelliJ IDEA / Eclipse
- Operating System: Windows 10 or Linux-based OS
- Libraries/Frameworks: Java Swing, JDBC



### 3. SAMPLE CODE

#### 3.1 GUI Design and Layout

The graphical user interface is designed using Java Swing with a focus on intuitive interactions. Below is a snippet to demonstrate the layout setup:

```
public CollisionTrackingApp() {  
  
    JFrame frame = new JFrame("Collision Tracking App");  
  
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
  
    frame.setSize(1000, 700);  
  
    frame.getContentPane().setBackground(new Color(240, 248, 255));  
  
  
    String[] columnNames = {"ID", "Size", "Speed", "Orbit", "Collision Risk"};  
  
    model = new DefaultTableModel(columnNames, 0);  
  
    table = new JTable(model);  
  
    table.setRowHeight(30);  
  
    table.setFont(new Font("Arial", Font.PLAIN, 14));  
  
    table.getTableHeader().setFont(new Font("Arial", Font.BOLD, 16));  
  
    table.getTableHeader().setBackground(new Color(173, 216, 230));  
  
  
    collisionRiskColumn = table.getColumnModel().getColumn(4);  
  
  
    JScrollPane scrollPane = new JScrollPane(table);  
  
    scrollPane.setBorder(BorderFactory.createTitledBorder("Space Debris Data"));  
  
  
    JPanel buttonPanel = new JPanel();
```

```

buttonPanel.setBackground(new Color(240, 248, 255));

setupButtonPanel(buttonPanel);


JPanel filterPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
filterPanel.setBackground(new Color(240, 248, 255));

JLabel filterLabel = new JLabel("Collision Risk Filter:");
riskFilterField = new JTextField(5);

filterPanel.add(filterLabel);

filterPanel.add(riskFilterField);


frame.add(scrollPane, BorderLayout.CENTER);

frame.add(buttonPanel, BorderLayout.SOUTH);

frame.add(filterPanel, BorderLayout.NORTH);


frame.setLocationRelativeTo(null);

frame.setVisible(true);


hideCollisionRiskColumn();

}


private void hideCollisionRiskColumn() {

    collisionRiskColumn.setMinWidth(0);

    collisionRiskColumn.setMaxWidth(0);

```

```

collisionRiskColumn.setPreferredWidth(0);

isCollisionRiskVisible = false;
}

private void showCollisionRiskColumn() {

    collisionRiskColumn.setMinWidth(75);

    collisionRiskColumn.setMaxWidth(200);

    collisionRiskColumn.setPreferredWidth(100);

    isCollisionRiskVisible = true;
}

```

### 3.2 Data Loading Operation

The loadData method loads debris details from the MySQL database and populates the table:

```

private void loadData() {

    model.setRowCount(0);

    String query = "SELECT debris_id, debris_size, debris_speed, debris_orbit, collision_risk FROM
SpaceDebris";

    try (Connection conn = DriverManager.getConnection(URL, USER, PASS);

        Statement stmt = conn.createStatement();

        ResultSet rs = stmt.executeQuery(query)) {

        while (rs.next()) {

```

```

        int debrisId = rs.getInt("debris_id");

        float size = rs.getFloat("debris_size");

        float speed = rs.getFloat("debris_speed");

        String orbit = rs.getString("debris_orbit");

        float collisionRisk = rs.getFloat("collision_risk");

        model.addRow(new Object[]{debrisId, size, speed, orbit, collisionRisk});
    }

    hideCollisionRiskColumn();

} catch (SQLException ex) {

    JOptionPane.showMessageDialog(null, "Database error: " + ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);

}

}

```

### 3.3 Collision Risk Analysis

This method analyzes the collision risk and reveals the "Collision Risk" column:

```

private void analyzeCollisionRisk() {

    if (isCollisionRiskVisible) {

        JOptionPane.showMessageDialog(null, "Collision Risk is already visible!", "Info",
JOptionPane.INFORMATION_MESSAGE);

        return;
    }
}

```

```
}
```

```
showCollisionRiskColumn();
```

```
}
```

### 3.4 Add, Update, and Delete Operations

#### Add Data:

```
private void addData() {  
    String debrisIdStr = JOptionPane.showInputDialog("Enter Debris ID:");  
    String sizeStr = JOptionPane.showInputDialog("Enter Debris Size:");  
    String speedStr = JOptionPane.showInputDialog("Enter Debris Speed:");  
    String orbitStr = JOptionPane.showInputDialog("Enter Debris Orbit:");  
  
    try {  
        int debrisId = Integer.parseInt(debrisIdStr);  
        float size = Float.parseFloat(sizeStr);  
        float speed = Float.parseFloat(speedStr);  
        String orbit = orbitStr;  
  
        float collisionRisk = 0; // Default value  
  
        model.addRow(new Object[]{debrisId, size, speed, orbit, collisionRisk});  
    } catch (NumberFormatException ex) {
```

```

        JOptionPane.showMessageDialog(null, "Invalid data format.", "Error",
JOptionPane.ERROR_MESSAGE);

    }

}

```

### **Delete Data:**

```

private void deleteData() {

    int selectedRow = table.getSelectedRow();

    if (selectedRow != -1) {

        model.removeRow(selectedRow);

    } else {

        JOptionPane.showMessageDialog(null, "Please select a row to delete.", "Error",
JOptionPane.ERROR_MESSAGE);

    }

}

```

### **Update Data:**

```

private void updateData() {

    int selectedRow = table.getSelectedRow();

    if (selectedRow != -1) {

        String debrisIdStr = JOptionPane.showInputDialog("Enter new Debris ID:");

        String sizeStr = JOptionPane.showInputDialog("Enter new Debris Size:");

        String speedStr = JOptionPane.showInputDialog("Enter new Debris Speed:");

        String orbitStr = JOptionPane.showInputDialog("Enter new Debris Orbit:");

```

```

try {
    int debrisId = Integer.parseInt(debrisIdStr);

    float size = Float.parseFloat(sizeStr);

    float speed = Float.parseFloat(speedStr);

    String orbit = orbitStr;

    model.setValueAt(debrisId, selectedRow, 0);

    model.setValueAt(size, selectedRow, 1);

    model.setValueAt(speed, selectedRow, 2);

    model.setValueAt(orbit, selectedRow, 3);

    model.setValueAt(0, selectedRow, 4); // Default collision risk

} catch (NumberFormatException ex) {

    JOptionPane.showMessageDialog(null, "Invalid data format.", "Error",
JOptionPane.ERROR_MESSAGE);

}

} else {

    JOptionPane.showMessageDialog(null, "Please select a row to update.", "Error",
JOptionPane.ERROR_MESSAGE);

}

}

```

### 3.5 Filtering Space Debris Data

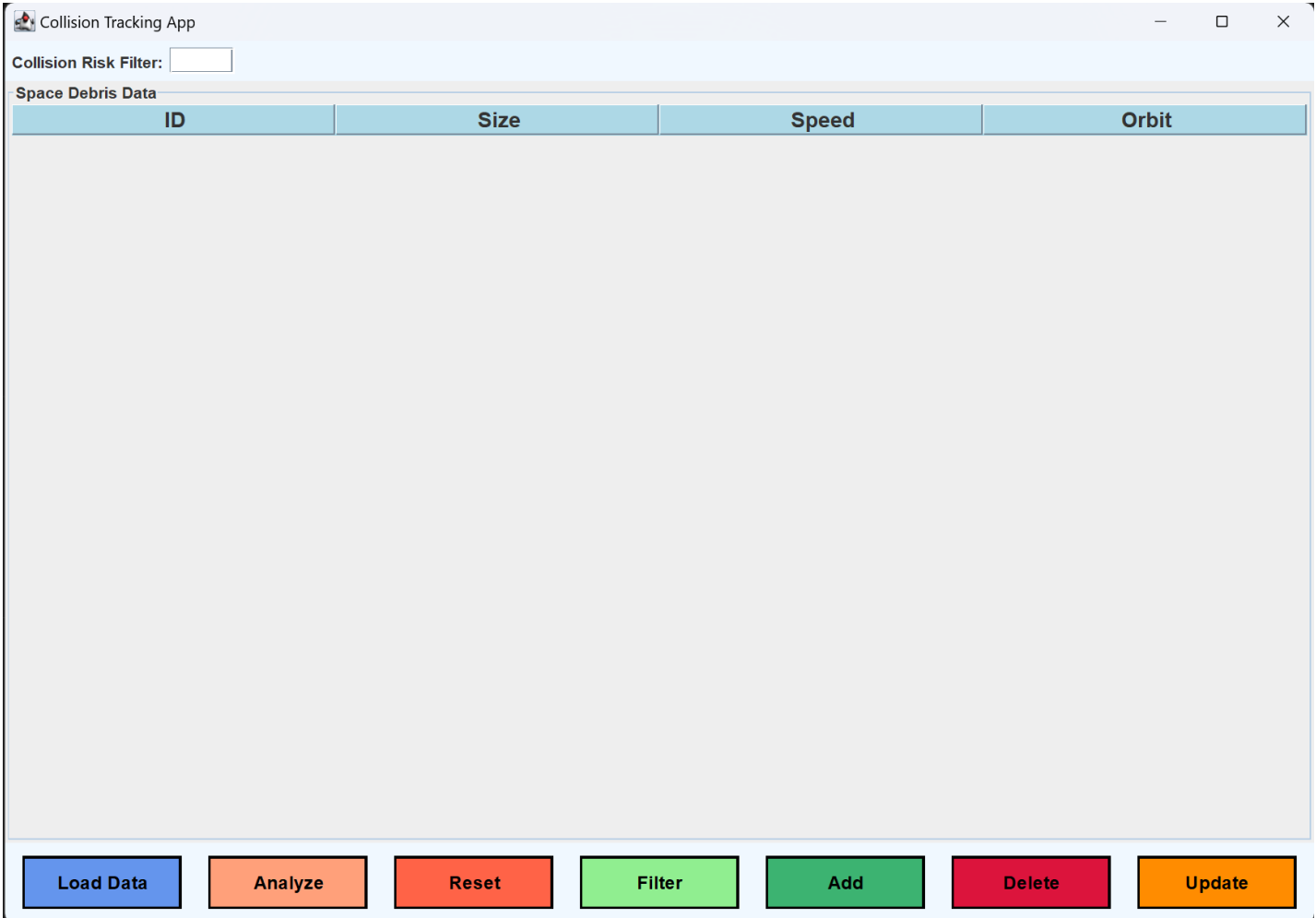
The method filters debris data based on the entered collision risk threshold:

```
private void filterByCollisionRisk() {  
    String riskThresholdStr = riskFilterField.getText();  
    if (!riskThresholdStr.isEmpty()) {  
        try {  
            float riskThreshold = Float.parseFloat(riskThresholdStr);  
            for (int i = 0; i < model.getRowCount(); i++) {  
                float collisionRisk = (float) model.getValueAt(i, 4);  
                if (collisionRisk < riskThreshold) {  
                    model.removeRow(i);  
                    i--; // Adjust index after removal  
                }  
            }  
        } catch (NumberFormatException ex) {  
            JOptionPane.showMessageDialog(null, "Invalid collision risk filter value.", "Error",  
JOptionPane.ERROR_MESSAGE);  
        }  
    }  
}
```



4. SNAPSHOTS

4.1. Home Page



4.2 Data Loading

Collision Tracking App

Collision Risk Filter:

Space Debris Data

ID	Size	Speed	Orbit
4	215.55	7.804	Payload
5	3825.37	8.196	Payload
0	1.0	1.0	1
89338	432.89	7.68	Payload
89453	296.69	7.75	Payload
89460	1160.12	7.33	Debris
89439	563.15	7.7	Debris
89458	6106.6	8.79	Debris
89491	915.09	7.89	Debris
89475	928.9	7.36	Payload
0	1.0	1.0	1
0	1.0	1.0	1

Load Data

Analyze

Reset

Filter

Add

Delete

Update

4.3 Collision Risk Analysis

Collision Tracking App

Collision Risk Filter:

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89338	432.89	7.68	Payload	0.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89458	6106.6	8.79	Debris	0.0
89491	915.09	7.89	Debris	1.0
89475	928.9	7.36	Payload	1.0
0	1.0	1.0	1	1.0
0	1.0	1.0	1	1.0

Load Data

Analyze

Reset

Filter

Add

Delete

Update

4.4 Add, Update, and Delete Features

Add

Collision Tracking App

Collision Risk Filter:

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89491	915.09		Debris	1.0
89475	928.9		Payload	1.0
0	1.0		1	1.0
0	1.0		1	1.0

Input

?

Enter Debris ID:

OK

Cancel

Load Data

Analyze

Reset

Filter

Add

Delete

Update

Before Update :

Collision Tracking App

Collision Risk Filter: 1.0

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89491	915.09		Debris	1.0
89475	928.9		Payload	1.0
0	1.0		1	1.0

Input

?

Enter new Debris ID:

899934

OKCancel

Load Data

Analyze

Reset

Filter

Add

Delete

Update

After Update:

Collision Tracking App

Collision Risk Filter:

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89491	915.09	7.89	Debris	1.0
89475	928.9	7.36	Payload	1.0
899934	215.89	7.89	1	0

Load Data

Analyze

Reset

Filter

Add

Delete

Update

Before Delete:

Collision Tracking App

Collision Risk Filter:

1.0

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89491	915.09	7.89	Debris	1.0
89475	928.9	7.36	Payload	1.0
0	1.0	1.0	1	1.0
0	1.0	1.0	1	1.0

Load Data

Analyze

Reset

Filter

Add

Delete

Update

After Delete:

Collision Tracking App

Collision Risk Filter:

1.0

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89491	915.09	7.89	Debris	1.0
89475	928.9	7.36	Payload	1.0
0	1.0	1.0	1	1.0

Load Data

Analyze

Reset

Filter

Add

Delete

Update



Reset :

Collision Tracking App

Collision Risk Filter:

1.0

Space Debris Data

ID	Size	Speed	Orbit
----	------	-------	-------

Load Data

Analyze

Reset

Filter

Add

Delete

Update

4.5 Filter by Collision Risk

Collision Tracking App

Collision Risk Filter:

Space Debris Data

ID	Size	Speed	Orbit	Collision Risk
4	215.55	7.804	Payload	1.0
5	3825.37	8.196	Payload	1.0
0	1.0	1.0	1	1.0
89453	296.69	7.75	Payload	1.0
89460	1160.12	7.33	Debris	1.0
89439	563.15	7.7	Debris	1.0
89491	915.09	7.89	Debris	1.0
89475	928.9	7.36	Payload	1.0
0	1.0	1.0	1	1.0
0	1.0	1.0	1	1.0

Load Data

Analyze

Reset

Filter

Add

Delete

Update

## CONCLUSION

The **Collision Tracking App** provides an efficient and user-friendly solution for managing and analyzing space debris data. By leveraging Java Swing for an intuitive graphical user interface and JDBC for seamless database integration, the application offers essential features like real-time collision risk analysis, data filtering, and CRUD (Create, Read, Update, Delete) operations.

The project demonstrates practical applications of software development concepts, including GUI design, database management, and real-time data analysis. This app is particularly beneficial for space research agencies and academic institutions, offering a robust tool to monitor, track, and mitigate the risks posed by space debris.

The development of this application highlights the integration of technology and problem-solving skills, paving the way for future enhancements such as predictive modeling and integration with live satellite tracking systems.

## 6. REFERENCES

1. <https://www.javatpoint.com/java-tutorial>
2. <https://dev.mysql.com/doc/>
3. <https://docs.oracle.com/javase/tutorial/>