

Industrial Internship Report on

"Project Name"

Console Based Expense Tracker

Prepared by

Rachit

Executive Summary

This report provides details of the Industrial Internship provided by upskill Campus and The IoT Academy in collaboration with Industrial Partner UniConverge Technologies Pvt Ltd (UCT).

This internship was focused on a project/problem statement provided by UCT. We had to finish the project including the report in 6 weeks' time.

My project was (Tell about ur Project)

This internship gave me a very good opportunity to get exposure to Industrial problems and design/implement solution for that. It was an overall great experience to have this internship.

TABLE OF CONTENTS

1	Preface	4
2	Introduction	5
2.1	About UniConverge Technologies Pvt Ltd	5
2.2	About upskill Campus	9
2.3	Objective	11
2.4	Reference	Error! Bookmark not defined.
3	Problem Statement	Error! Bookmark not defined.
4	Existing and Proposed solution	Error! Bookmark not defined.
5	Proposed Design/ Model	Error! Bookmark not defined.
5.1	High Level Diagram (if applicable)	Error! Bookmark not defined.
5.2	Low Level Diagram (if applicable)	Error! Bookmark not defined.
5.3	Interfaces (if applicable)	Error! Bookmark not defined.
6	Performance Test	Error! Bookmark not defined.
6.1	Test Plan/ Test Cases	Error! Bookmark not defined.
6.2	Test Procedure	Error! Bookmark not defined.
6.3	Performance Outcome	Error! Bookmark not defined.
7	My learnings	Error! Bookmark not defined.

1 Preface

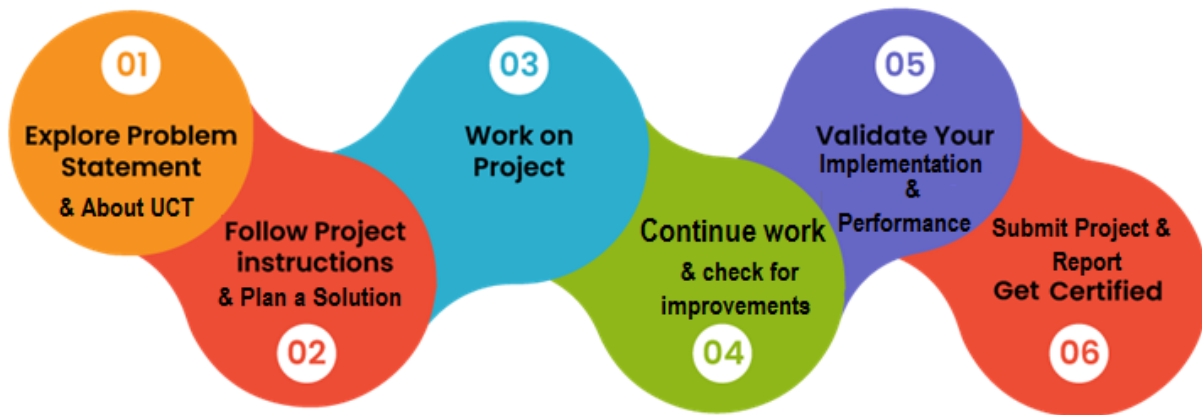
Summary of the whole 6 weeks' work.

About need of relevant Internship in career development.

Brief about Your project/problem statement.

Opportunity given by USC/UCT.

How Program was planned



Your Learnings and overall experience.

Thank to all (with names), who have helped you directly or indirectly.

Your message to your juniors and peers.

2 Introduction

2.1 About UniConverge Technologies Pvt Ltd

A company established in 2013 and working in Digital Transformation domain and providing Industrial solutions with prime focus on sustainability and RoI.

For developing its products and solutions it is leveraging various **Cutting Edge Technologies** e.g. **Internet of Things (IoT), Cyber Security, Cloud computing (AWS, Azure), Machine Learning, Communication Technologies (4G/5G/LoRaWAN), Java Full Stack, Python, Front end** etc.



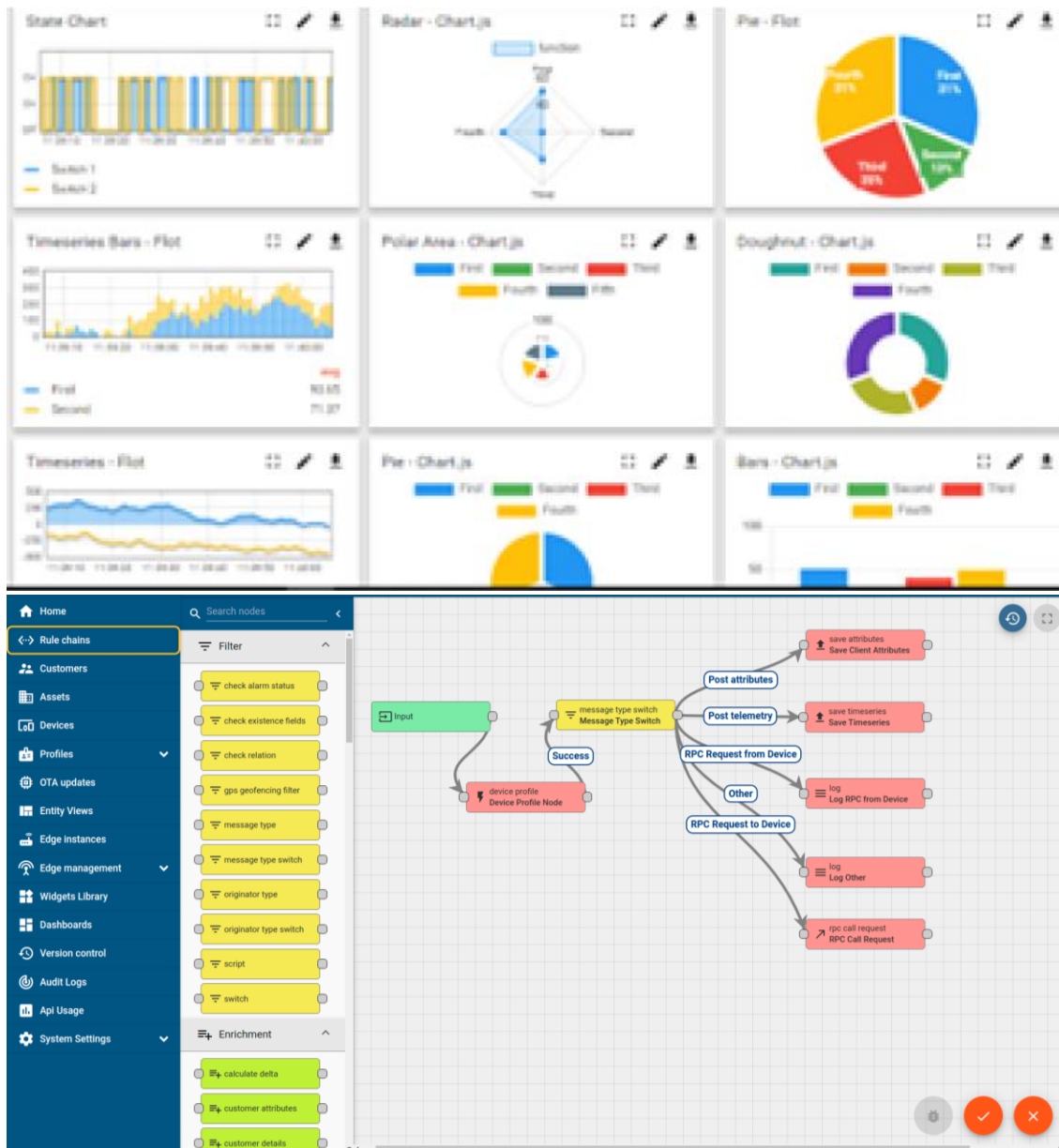
i. UCT IoT Platform ()

UCT Insight is an IOT platform designed for quick deployment of IOT applications on the same time providing valuable “insight” for your process/business. It has been built in Java for backend and ReactJS for Front end. It has support for MySQL and various NoSql Databases.

- It enables device connectivity via industry standard IoT protocols - MQTT, CoAP, HTTP, Modbus TCP, OPC UA
- It supports both cloud and on-premises deployments.

It has features to

- Build Your own dashboard
- Analytics and Reporting
- Alert and Notification
- Integration with third party application(Power BI, SAP, ERP)
- Rule Engine



ii. **Smart Factory Platform ()**

Factory watch is a platform for smart factory needs.

It provides Users/ Factory

- with a scalable solution for their Production and asset monitoring
- OEE and predictive maintenance solution scaling up to digital twin for your assets.
- to unleash the true potential of the data that their machines are generating and helps to identify the KPIs and also improve them.
- A modular architecture that allows users to choose the service that they want to start and then can scale to more complex solutions as per their demands.

Its unique SaaS model helps users to save time, cost and money.



Machine	Operator	Work Order ID	Job ID	Job Performance	Job Progress		Output		Rejection	Time (mins)				Job Status	End Customer
					Start Time	End Time	Planned	Actual		Setup	Pred	Downtime	Idle		
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i
CNC_S7_81	Operator 1	WO0405200001	4168	58%	10:30 AM		55	41	0	80	215	0	45	In Progress	i



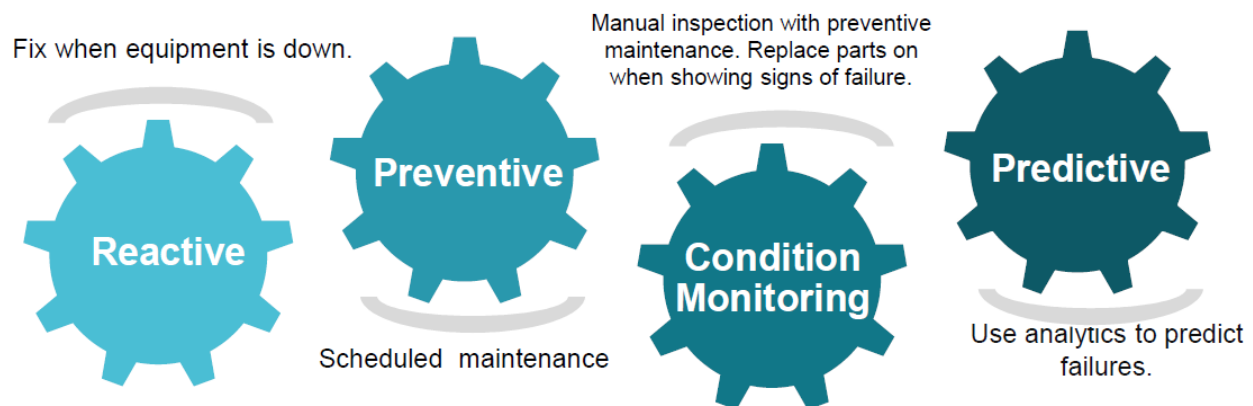


iii. based Solution

UCT is one of the early adopters of LoRAWAN technology and providing solution in Agritech, Smart cities, Industrial Monitoring, Smart Street Light, Smart Water/ Gas/ Electricity metering solutions etc.

iv. Predictive Maintenance

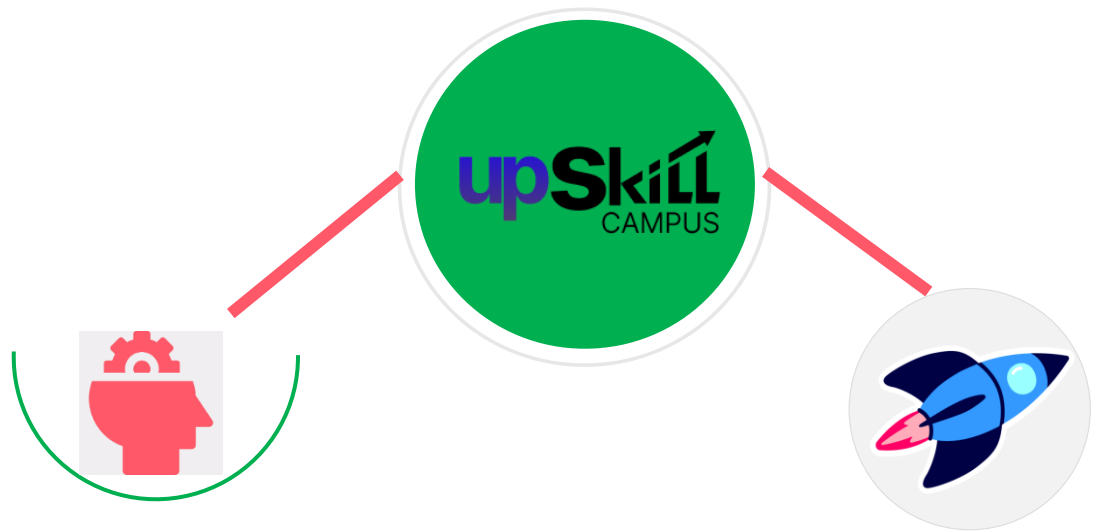
UCT is providing Industrial Machine health monitoring and Predictive maintenance solution leveraging Embedded system, Industrial IoT and Machine Learning Technologies by finding Remaining useful life time of various Machines used in production process.



2.2 About upskill Campus (USC)

upskill Campus along with The IoT Academy and in association with Uniconverge technologies has facilitated the smooth execution of the complete internship process.

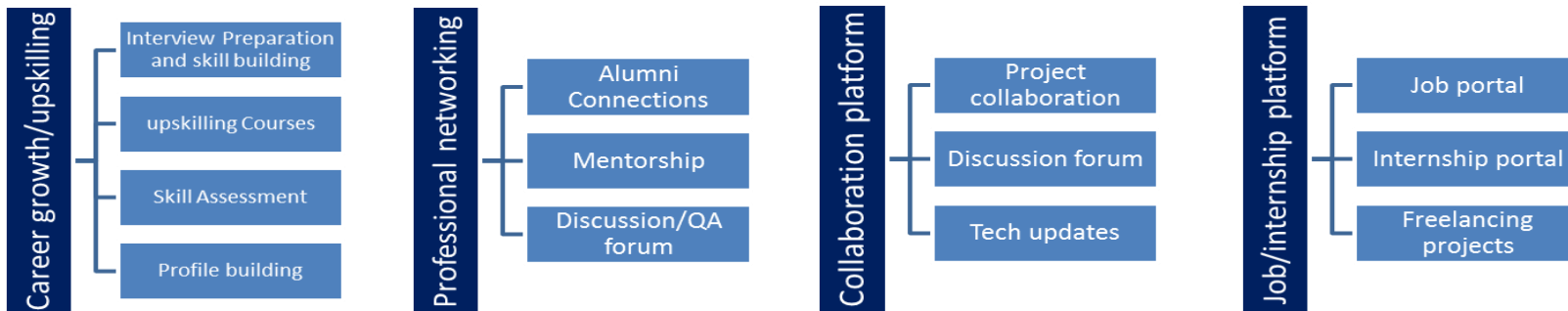
USC is a career development platform that delivers **personalized executive coaching** in a more affordable, scalable and measurable way.



Seeing need of upskilling in self paced manner along-with additional support services e.g. Internship, projects, interaction with Industry experts, Career growth Services

upSkill Campus aiming to upskill 1 million learners in next 5 year

<https://www.upskillcampus.com/>



2.3 The IoT Academy

The IoT academy is EdTech Division of UCT that is running long executive certification programs in collaboration with EICT Academy, IITK, IITR and IITG in multiple domains.

2.4 Objectives of this Internship program

The objective for this internship program was to

- get practical experience of working in the industry.
- to solve real world problems.
- to have improved job prospects.
- to have Improved understanding of our field and its applications.
- to have Personal growth like better communication and problem solving.

This report presents an overview of a console-based expense tracker developed in the Java programming language. The application aims to provide users with a convenient way to track and manage their expenses efficiently. The report outlines the features, design, and implementation details of the expense tracker.

1. Introduction: Managing personal finances effectively is crucial for individuals and businesses alike. An expense tracker helps individuals monitor their spending habits, track expenses, and make informed financial decisions. The console-based expense tracker is designed to fulfil these requirements.
2. Features: The console-based expense tracker offers the following key features:

a) Expense Recording: Users can record their expenses by providing details such as date, category, description, and amount.

b) Expense Listing: The tracker allows users to view a list of recorded expenses, including their details.

c) Expense Filtering: Users can filter expenses based on criteria such as date range, category, or specific keywords.

d) Expense Statistics: The application provides statistical information, such as total expenses, average expense per day, and category-wise expense distribution.

e) Data Persistence: The tracker ensures that expense data is stored persistently, allowing users to access their records across multiple sessions.

3. Design and Architecture: The expense tracker follows a modular and object-oriented design to enhance maintainability and extensibility. The core components of the application are:

a) Expense Class: Represents an individual expense and encapsulates its attributes, such as date, category, description, and amount.

b) Expense Manager Class: Manages the expense records, providing functionality to add, retrieve, and filter expenses.

c) ExpenseTrackerApp Class: Acts as the main driver class, facilitating user interactions and menu-based navigation.

4. Implementation Details: The expense tracker is implemented using the Java programming language and adheres to best practices in software development. Key implementation details include:

a) User Input Handling: The application utilizes input streams and scanner objects to accept and process user input.

b) Data Storage: Expense records are stored in a file-based format, such as CSV (Comma-Separated Values), enabling easy data persistence and retrieval.

c) File I/O Operations: Java's file I/O libraries are used to read and write expense records from/to the storage file.

d) Data Validation: The application incorporates input validation to ensure the accuracy and integrity of the recorded expenses.

5. User Guide: To use the console-based expense tracker, users follow these steps:

a) Launch the application: Run the Java application from the command line or IDE.

b) Main Menu: The main menu provides options to record expenses, view expenses, apply filters, or view statistics.

c) Recording Expenses: Users can input the expense details prompted by the application.

d) Viewing Expenses: The application displays a list of recorded expenses along with their details.

e) Applying Filters: Users can specify filter criteria to view a subset of expenses based on date range, category, or keywords.

f) Viewing Statistics: Users can access statistical information about their expenses.

6. Conclusion: The console-based expense tracker in Java offers a practical solution for managing personal finances efficiently. It provides essential features for recording, listing, filtering, and analyzing expenses. The modular design and adherence to software development best practices ensure maintainability and extensibility.

7. Future Enhancements: To further improve the expense tracker, potential enhancements could include:

a) User Authentication: Adding user authentication to secure expense data.

b) Graphical User Interface (GUI): Developing a GUI version to enhance the user experience.

c) Budgeting and Goal Setting: Incorporating budgeting features and goal setting to help users achieve financial objectives.

d) Data Analysis: Integrating data visualization tools to generate visual representations of expense patterns.

```
import java.sql.*;

import javax.swing.JOptionPane;

/**
 *
 * @author Rachit
 */
public class Category extends javax.swing.JFrame {

    /**
     * Creates new form Category
     */
    public Category() {
        initComponents();
        getEntries();
    }
    private void getEntries(){
        try{
            javax.swing.table.DefaultTableModel dtm=
            (javax.swing.table.DefaultTableModel)table.getModel();
            int rc=dtm.getRowCount();
            while(rc--!=0){
                dtm.removeRow(0);
            }

            ResultSet rs=db.DbConnect.st.executeQuery("select * from category_info");
            int sno=0;
            while(rs.next()){
                String category=rs.getString("category");

                // Object o[]={++sno,category};
                // dtm.addRow(o);
            }
        }
    }
}
```

```
java.util.Vector row=new java.util.Vector();
row.add(++sno);
row.add(category);
dtm.addRow(row);
}

}catch(Exception ex){
JOptionPane.showMessageDialog(null, ex);
}
}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN: initComponents
private void initComponents() {

jPanel1 = new javax.swing.JPanel();
jPanel2 = new javax.swing.JPanel();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();
t = new javax.swing.JTextField();
jButton1 = new javax.swing.JButton();
jScrollPane1 = new javax.swing.JScrollPane();
table = new javax.swing.JTable();
jButton2 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
setTitle("Category");

jPanel1.setBackground(new java.awt.Color(204, 153, 0));

jPanel2.setBackground(new java.awt.Color(255, 204, 102));
```

```
jLabel1.setFont(new java.awt.Font("Segoe UI", 1, 18)); // NOI18N
jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel1.setText("Add New Category");

javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
jPanel2.setLayout(jPanel2Layout);
jPanel2Layout.setHorizontalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(18, 18, 18)
            .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 516, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(22, 22, 22)
        )
);
jPanel2Layout.setVerticalGroup(
    jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addGap(18, 18, 18)
            .addComponent(jLabel1)
            .addGap(22, 22, 22)
        )
);

jLabel2.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
jLabel2.setText("Category :");

t.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        tActionPerformed(evt);
    }
});

jButton1.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
jButton1.setText("ADD");
jButton1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton1ActionPerformed(evt);
    }
});
```



```
});
```

```

javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(
    jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(27, 27, 27)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 124, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(12, 12, 12)
                    .addComponent(t, javax.swing.GroupLayout.PREFERRED_SIZE, 304, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                    .addComponent(jButton1, javax.swing.GroupLayout.PREFERRED_SIZE, 85, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGap(35, 35, Short.MAX_VALUE))
            .addContainerGap())
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE)
            .addGap(18, 18, 18)
            .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                .addComponent(jLabel2)
                .addComponent(t, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE)
                .addComponent(jButton1))
            .addGap(25, 25, Short.MAX_VALUE))
);

```

```
table.setModel(new javax.swing.table.DefaultTableModel(
```

```
new Object [][] {
```



```
new Object [][] {  
    },  
new String [] {  
    "S.No.", "Category"
```

```

    }
    {
        boolean[] canEdit = new boolean [] {
            false, false
        };

        public boolean isCellEditable(int rowIndex, int columnIndex) {
            return canEdit [columnIndex];
        }
    });

    jScrollPane1.setViewportViewView(table);

    jButton2.setBackground(new java.awt.Color(255, 102, 102));
    jButton2.setFont(new java.awt.Font("Segoe UI", 1, 14)); // NOI18N
    jButton2.setText("Delete");
    jButton2.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton2ActionPerformed(evt);
        }
    });

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 593,
                        javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(42, 42, 42)
                        .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 525,
                            javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGap(42, 42, 42))
            .addContainerGap(18, Short.MAX_VALUE)
    );
}

```

```
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)))
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);
layout.setVerticalGroup(
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
.addGroup(layout.createSequentialGroup()
.addContainerGap()
.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 253,
javax.swing.GroupLayout.PREFERRED_SIZE)
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE, 34, javax.swing.GroupLayout.PREFERRED_SIZE)
.addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
);

pack();
setLocationRelativeTo(null);
} // </editor-fold> // GEN-END: initComponents

private void tActionPerformed(java.awt.event.ActionEvent evt) { // GEN-FIRST:event_tActionPerformed
jButton1ActionPerformed(null);
} // GEN-LAST:event_tActionPerformed

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { // GEN-
FIRST:event_jButton1ActionPerformed
try{
String category=t.getText();
if(!category.equals("")){
db.DbConnect.st.executeUpdate("insert into category_info values('"+category+"')");
JOptionPane.showMessageDialog(null, "Category Added Successfully");
getEntries();
}
else{
JOptionPane.showMessageDialog(null, "Please Enter any value");
```

```

    }
    }catch(SQLIntegrityConstraintViolationException ex){
        JOptionPane.showMessageDialog(null, "Category Already Exist");
    }catch(Exception ex){
        JOptionPane.showMessageDialog(null, ex);
    }
}
} //GEN-LAST:event_jButton1ActionPerformed

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) { //GEN-FIRST:event_jButton2ActionPerformed
    int ri=table.getSelectedRow(); //get row index of selected row
    if(ri!=-1){
        int r= JOptionPane.showConfirmDialog(null,
            "Do you really wanna Delete","Deletion Confirmation" ,
            JOptionPane.YES_NO_OPTION);
        if (r==JOptionPane.YES_OPTION){

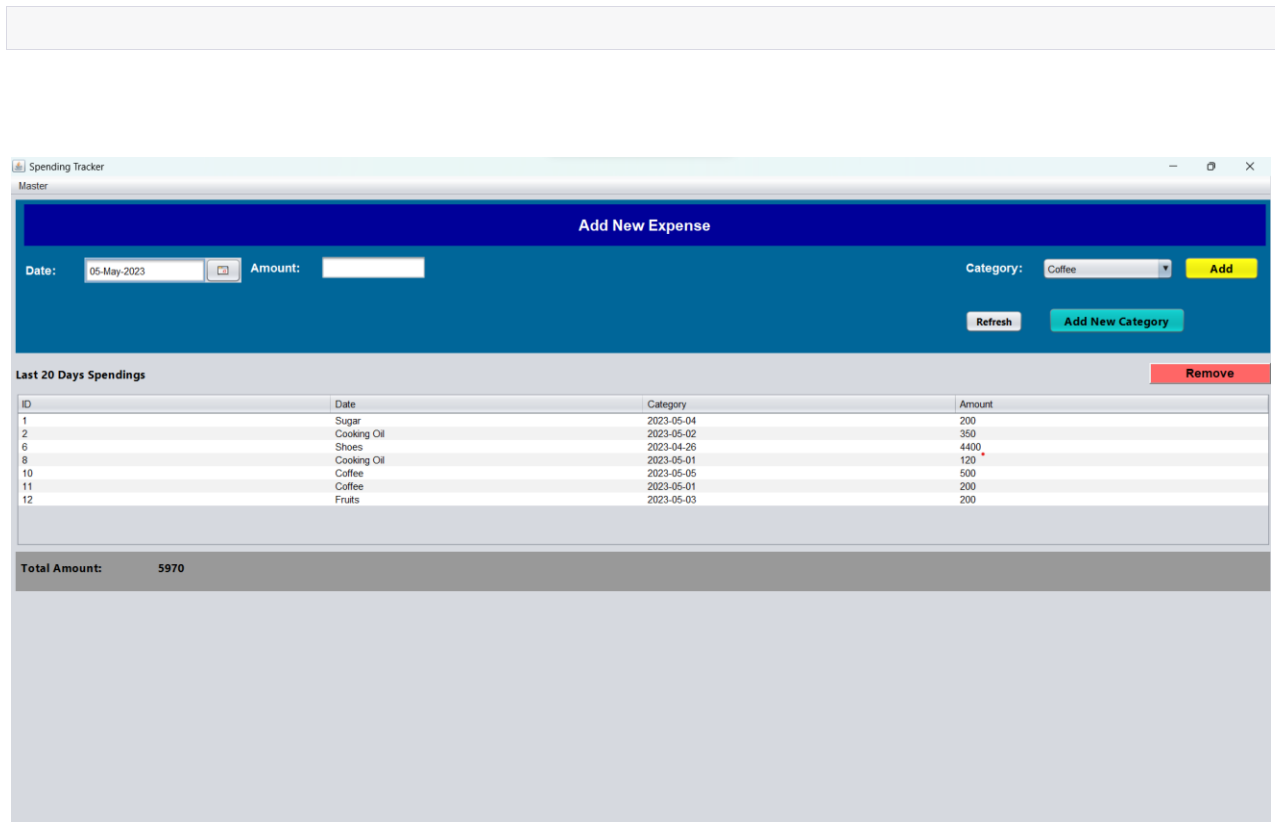
            String category=(String)table.getValueAt(ri, 1);
            try{
                db.DbConnect.st.executeUpdate("delete from category_info where category='"+category+"'");
                JOptionPane.showMessageDialog(null, ""
                    + "Category deleted Successfully");
                getEntries();

            }catch(Exception ex){
                JOptionPane.showMessageDialog(null, ex);}
            }}
        } //GEN-LAST:event_jButton2ActionPerformed

// Variables declaration - do not modify //GEN-BEGIN:variables
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;

```

```
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTextField t;
private javax.swing.JTable table;
// End of variables declaration//GEN-END:variables
}
```



ID	Date	Category	Amount
1	2023-05-04	Sugar	200
2	2023-05-02	Cooking Oil	350
6	2023-04-26	Shoes	4400
8	2023-05-01	Cooking Oil	120
10	2023-05-05	Coffee	500
11	2023-05-01	Coffee	200
12	2023-05-03	Fruits	200

Total Amount: 5970

Spending Tracker - Master

Add New Expense

Date: 05-May-2023 Amount: Category: Coffee

Last 20 Days Spendings

ID	Date	Category	Amount
1		Sugar	200
2		Cooking Oil	350
6		Shoes	4400
8		Cooking Oil	120
10		Coffee	500
11		Coffee	200
12		Fruits	200

Total Amount: 5970

View Spending

View Spending Date Wise

From: 01-May-2023 To: 05-May-2023

Total Amount: 1570

Date	Category	Amount
2023-05-01	Cooking Oil	120
2023-05-01	Coffee	200
2023-05-02	Cooking Oil	350
2023-05-03	Fruits	200
2023-05-04	Sugar	200
2023-05-05	Coffee	500

View Spending Category Wise

Category: Coffee

From: 01-May-2023 To: 05-May-2023

Total Amount: 700

Date	Category	Amount
2023-05-01	Coffee	200
2023-05-05	Coffee	500

Spending Tracker - Master

Add New Expense

Date: 05-May-2023 Amount: Category: Coffee

Last 20 Days Spendings

ID	Date	Category	Amount
1		Sugar	200
2		Cooking Oil	350
6		Shoes	4400
8		Cooking Oil	120
10		Coffee	500
11		Coffee	200
12		Fruits	200

Total Amount: 5970

Add New Category

Category :

S.No.	Category
1	Coffee
2	Cooking Oil
3	Fruits
4	Shoes
5	Sugar