Interactive Data Visualization Tool

Mini-Project Report of

Object Oriented Programming through Java

Submitted by

Chahel Gupta (C049)

Under the Guidance Of

PROF. SAHIL MEHTA

In partial fulfillment for the award of the degree of

B Tech. Integrated

COMPUTER ENGINEERING

At

MUKESH PATEL SCHOOL OF TECHNOLOGY MANAGEMENT & ENGINEERING

NMIMS (Deemed –to-be University)

JVPD Scheme Bhaktivedanta Swami Marg,

Ville Parle (W), Mumbai-400 056.

2023-2024

Table of Contents

I. Project Statement and Scope

The JavaFX project aims to build an interactive data visualization tool offering Pie Charts, Bar Graphs, and Line Graphs. Users input data for each chart type, customize visual aspects like colors and labels, and interact with zooming and tooltips. The user interface features a main menu for chart selection and intuitive input fields. Robust data validation and error handling ensure accurate chart rendering. Comprehensive documentation guides users, while thorough testing guarantees functionality. Deployment options include executable files for easy use. Continuous support and updates are planned based on user feedback for an optimal user experience in visualizing data trends and patterns.

SCOPE OF THE PROJECT:

1. *User Interface (UI):*

- Create a user-friendly UI using JavaFX to provide a seamless experience for users.
- Include a main menu screen with options for Pie Chart, Bar Graph, and Line Graph.
- Implement navigation between different chart options and back to the main menu.
- Include input fields or dialogs for users to input their data points for visualization.

2. *Chart Types:*

- *Pie Chart:*
- Allow users to input data values and labels for creating a Pie Chart.
- Display the Pie Chart with interactive features such as tooltips showing data labels and values.

- *Bar Graph:*
- Provide options for users to input data names/numbers and their frequencies to create a Bar Graph.
- Display the Bar Graph with labeled axes and appropriate scaling based on the data.

- *Line Graph:*
- Allow users to input data points with X and Y coordinates to create a Line Graph.
- Display the Line Graph with labeled axes and markers connecting data points.

3. *Data Handling:*

- Validate user input to ensure it is in the correct format for each chart type.
- Handle data input errors and display appropriate error messages to users.

4. *Chart Customization:*

- Provide options for users to customize the appearance of the charts, such as colors, legends, titles, axis labels, etc.

5. *Interactivity:*

- Implement interactive features such as zooming, panning, and tooltips to enhance the user experience.
- Enable users to save the generated charts as images or export data for further analysis.

II. Project Synopsis

The interactive data visualization tool developed using JavaFX offers a comprehensive solution for users to analyze and present their data effectively. With a focus on user experience, the tool provides a range of chart options including Pie Charts for categorical data, Bar Graphs for frequency distribution, and Line Graphs for trend analysis over time. Users can input their data directly into the tool, customize the appearance of charts with colors, labels, and legends, and explore interactive features such as zooming and tooltips for in-depth analysis.

The tool's robust data validation ensures data accuracy and integrity, while clear error handling guides users in correcting input mistakes. Extensive documentation and a user-friendly interface simplify the process of creating and customizing charts. Thorough testing across various scenarios guarantees the tool's functionality and reliability.

Deployment options include generating executable files for easy installation and usage across different platforms. Continuous support and updates based on user feedback ensure that the tool remains intuitive, efficient, and adaptable to evolving data visualization needs.

III. Implementation

MainScreen

```
package com.example.oopjproj;

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class mainscreen extends Application {

    @Override
    public void start(Stage primaryStage) {
        primaryStage.setTitle("OOPJ MINI PROJECT");

        Label headerLabel1 = new Label("Interactive Data Visualization Tool");
        headerLabel1.setStyle("-fx-font-size: 28px; -fx-font-weight: bold; -fx-font-family: Georgia"); // Align header label to center
        Label headerLabel2 = new Label("C034, C044, C049");
        headerLabel2.setStyle("-fx-font-size: 20px;");

        Button pieButton = new Button("Pie Chart");
        Button barButton = new Button("Bar Chart");
        Button lineButton = new Button("Line Chart");

        pieButton.setStyle("-fx-font-size: 18px; -fx-min-width: 300px; -fx-min-height: 60px; -fx-background-color: #2DD881; -fx-text-fill: white;-fx-font-weight: bold;-fx-font-family: Cambria; -fx-effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0, 0, 2);");
        barButton.setStyle("-fx-font-size: 18px; -fx-min-width: 300px; -fx-min-height: 60px; -fx-background-color: #4EA699; -fx-text-fill: white;-fx-font-weight: bold;-fx-font-family: Cambria; -fx-effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0, 0, 2);");
        lineButton.setStyle("-fx-font-size: 18px; -fx-min-width: 300px; -fx-min-height: 60px; -fx-background-color: #140D4F; -fx-text-fill: white;-fx-font-weight: bold;-fx-font-family: Cambria; -fx-effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0, 0, 2);");

        pieButton.setOnAction(event -> {
            pie pieChartExample = new pie();
            pieChartExample.start(new Stage());
        });

        barButton.setOnAction(event -> {
            bar barChartExample = new bar();
```

```
            barChartExample.start(new Stage());
        });

        lineButton.setOnAction(event -> {
            line lineChartExample = new line();
            lineChartExample.start(new Stage());
        });

        VBox buttonsVBox = new VBox(20, pieButton, barButton, lineButton);
        buttonsVBox.setStyle("-fx-padding: 20px; -fx-alignment: center;");

        VBox root = new VBox();
        root.getChildren().addAll(headerLabel1, headerLabel2, buttonsVBox);
        root.setStyle("-fx-alignment: center;-fx-background-color: white; -fx-font-family:
    Cambria");

        Scene scene = new Scene(root, 800, 600);


        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Pie Chart

```
package com.example.oopjproj;

import javafx.application.Application;
import javafx.beans.property.SimpleDoubleProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.chart.PieChart;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class pie extends Application {

    @Override
    public void start(Stage primaryStage) {
        Label label = new Label("Enter data for the pie chart:");
        label.setStyle("-fx-font-size: 18px; -fx-font-weight: bold; -fx-text-fill: #333333;");
        TextField nameInput = new TextField();
        nameInput.setStyle("-fx-font-size: 14px;");

        TextField dataInput = new TextField();
        dataInput.setStyle("-fx-font-size: 14px;");

        Button addButton = new Button("Add Data");
        addButton.setStyle("-fx-font-size: 14px; -fx-background-color: #2DD881; -fx-text-fill: white; -fx-effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0, 0, 2);");
        PieChart pieChart = new PieChart();

        TableView<ItemData> tableView = new TableView<>();

        TableColumn<ItemData, String> itemNameCol = new TableColumn<>("Item Name");
        itemNameCol.setCellValueFactory(cellData -> cellData.getValue().itemNameProperty());

        TableColumn<ItemData, Number> itemValueCol = new TableColumn<>("Value");
        itemValueCol.setCellValueFactory(cellData -> cellData.getValue().itemValueProperty());
```

```
        tableView.getColumns().addAll(itemNameCol, itemValueCol);


        ObservableList<ItemData> itemDataList = FXCollections.observableArrayList();

        addButton.setOnAction(event -> {
            try {
                String itemName = nameInput.getText();
                double dataValue = Double.parseDouble(dataInput.getText());
                ItemData itemData = new ItemData(itemName, dataValue);
                itemDataList.add(itemData);


                tableView.setItems(itemDataList);

                ObservableList<PieChart.Data> pieChartData =
FXCollections.observableArrayList();
                double totalValue = 0.0;
                for (ItemData item : itemDataList) {
                    totalValue += item.getItemValue();
                }
                for (ItemData item : itemDataList) {
                    double percentage = (item.getItemValue() / totalValue) * 100;
                    pieChartData.add(new PieChart.Data(item.getItemName() + " (" +
String.format("%.2f", percentage) + "%)", item.getItemValue()));
                }
                pieChart.setData(pieChartData);

                nameInput.clear();
                dataInput.clear();
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a number.");
            }
        });

        VBox root = new VBox(10, label, new HBox(10, new Label("Item Name:"),
nameInput),
                new HBox(10, new Label("Value:"), dataInput), addButton, tableView,
pieChart);
        root.setStyle("-fx-padding: 20px; -fx-background-color: #f4f4f4;-fx-font-family:
Cambria");

        Scene scene = new Scene(root, 800, 600);
        primaryStage.setTitle("PIE CHART GENERATOR");
        primaryStage.setScene(scene);
        primaryStage.show();
```

```
    }

    public static void main(String[] args) {
        launch(args);
    }

    public static class ItemData {
        private final SimpleStringProperty itemName;
        private final SimpleDoubleProperty itemValue;

        public ItemData(String itemName, double itemValue) {
            this.itemName = new SimpleStringProperty(itemName);
            this.itemValue = new SimpleDoubleProperty(itemValue);
        }

        public String getItemName() {
            return itemName.get();
        }

        public SimpleStringProperty itemNameProperty() {
            return itemName;
        }

        public double getItemValue() {
            return itemValue.get();
        }

        public SimpleDoubleProperty itemValueProperty() {
            return itemValue;
        }
    }
}
```

Bar Graph

```
package com.example.oopjproj;

import javafx.application.Application;
import javafx.beans.property.SimpleDoubleProperty;
import javafx.beans.property.SimpleStringProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.XYChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class bar extends Application {

    @Override
    public void start(Stage primaryStage) {

        Label label = new Label("Enter data for the bar chart:");
        label.setStyle("-fx-font-size: 18px; -fx-font-weight: bold; -fx-text-fill: #333333;");
        TextField nameInput = new TextField();
        nameInput.setStyle("-fx-font-size: 14px;");

        TextField dataInput = new TextField();
        dataInput.setStyle("-fx-font-size: 14px;");

        Button addButton = new Button("Add Data");
        addButton.setStyle("-fx-font-size: 14px; -fx-background-color: #4EA699; -fx-text-fill: white; -fx-effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0, 0, 2);");
        BarChart<String, Number> barChart = new BarChart<>(new CategoryAxis(), new NumberAxis());

        TableView<ItemData> tableView = new TableView<>();

        TableColumn<ItemData, String> itemNameCol = new TableColumn<>("Item Name");
        itemNameCol.setCellValueFactory(cellData -> cellData.getValue().itemNameProperty());
```

```
        TableColumn<ItemData, Number> itemValueCol = new
TableColumn<>("Value");
        itemValueCol.setCellValueFactory(cellData ->
cellData.getValue().itemValueProperty());

        tableView.getColumns().addAll(itemNameCol, itemValueCol);


        ObservableList<ItemData> itemDataList = FXCollections.observableArrayList();

        addButton.setOnAction(event -> {
            try {
                String itemName = nameInput.getText();
                double dataValue = Double.parseDouble(dataInput.getText());


                boolean itemExists = false;
                for (ItemData item : itemDataList) {
                    if (item.getItemName().equals(itemName)) {
                        itemExists = true;
                        break;
                    }
                }

                if (!itemExists) {
                    ItemData itemData = new ItemData(itemName, dataValue);
                    itemDataList.add(itemData);


                    tableView.setItems(itemDataList);


                    XYChart.Series<String, Number> series = new XYChart.Series<>();
                    series.getData().add(new XYChart.Data<>(itemName, dataValue));
                    barChart.getData().add(series);
                } else {
                    System.out.println("Item already exists. Please enter a unique item
name.");
                }

                nameInput.clear();
                dataInput.clear();
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a number.");
            }
        });
```

```
        VBox root = new VBox(10, label, new HBox(10, new Label("Item Name:"),
    nameInput),
                new HBox(10, new Label("Value:"), dataInput), addButton, tableView,
    barChart);
        root.setStyle("-fx-padding: 20px; -fx-background-color: #f4f4f4;-fx-font-family:
    Cambria");

        Scene scene = new Scene(root, 800, 600);
        primaryStage.setTitle("BAR CHART GENERATOR");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

    public static class ItemData {
        private final SimpleStringProperty itemName;
        private final SimpleDoubleProperty itemValue;

        public ItemData(String itemName, double itemValue) {
            this.itemName = new SimpleStringProperty(itemName);
            this.itemValue = new SimpleDoubleProperty(itemValue);
        }

        public String getItemName() {
            return itemName.get();
        }

        public SimpleStringProperty itemNameProperty() {
            return itemName;
        }

        public double getItemValue() {
            return itemValue.get();
        }

        public SimpleDoubleProperty itemValueProperty() {
            return itemValue;
        }
    }
}
```

Line Graph

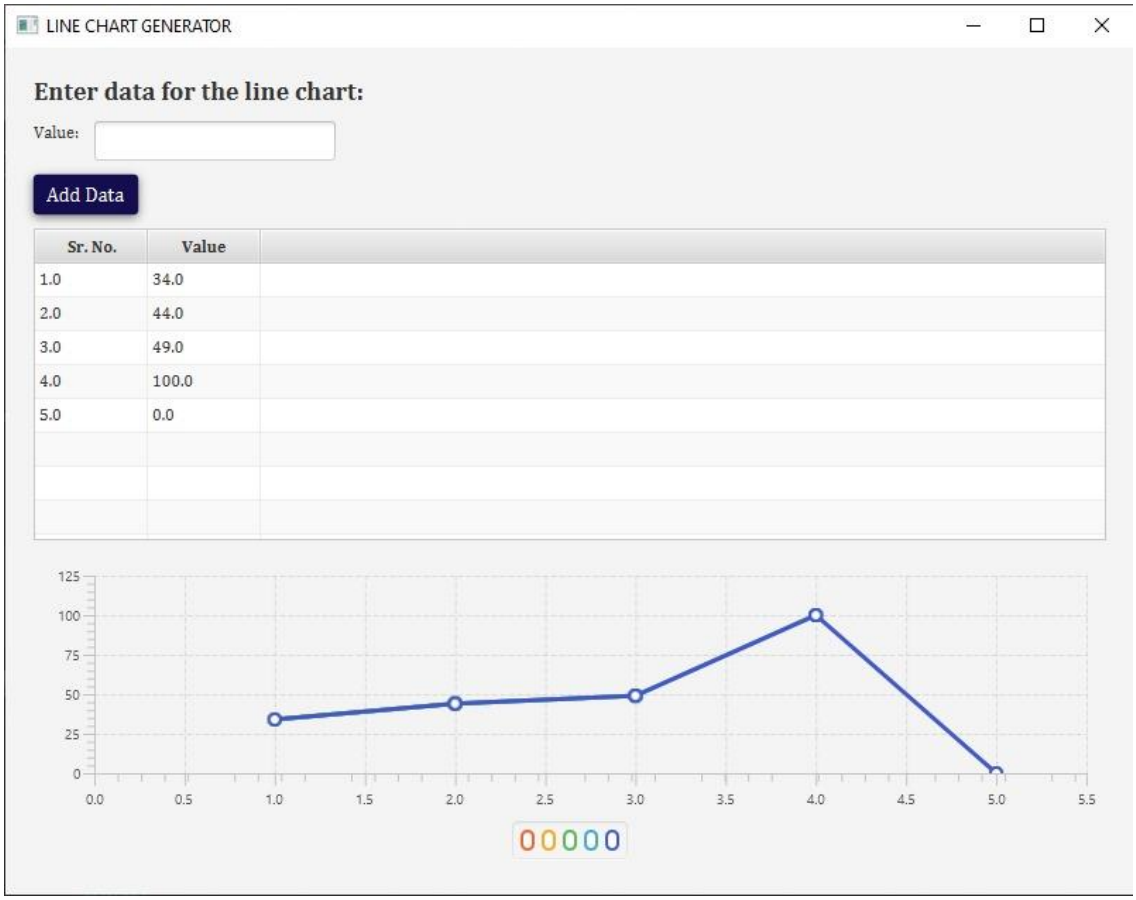
```
package com.example.oopjproj;

import javafx.application.Application;
import javafx.beans.property.SimpleDoubleProperty;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class line extends Application {

    @Override
    public void start(Stage primaryStage) {
        Label label = new Label("Enter data for the line chart:");
        label.setStyle("-fx-font-size: 18px; -fx-font-weight: bold; -fx-text-fill:
#333333;");
        TextField valueInput = new TextField();
        valueInput.setStyle("-fx-font-size: 14px;");
        Button addButton = new Button("Add Data");
        addButton.setStyle("-fx-font-size: 14px; -fx-background-color: #140D4F; -fx-
text-fill: white; -fx-effect: dropshadow(gaussian, rgba(0,0,0,0.5), 10, 0, 0, 2);");
        LineChart<Number, Number> lineChart = new LineChart<>(new NumberAxis(),
new NumberAxis());

        TableView<ItemData> tableView = new TableView<>();

        TableColumn<ItemData, Number> xValueCol = new TableColumn<>("Sr. No.");
        xValueCol.setCellValueFactory(cellData ->
cellData.getValue().xValueProperty());

        TableColumn<ItemData, Number> valueCol = new TableColumn<>("Value");
        valueCol.setCellValueFactory(cellData ->
cellData.getValue().yValueProperty());

        tableView.getColumns().clear();
        tableView.getColumns().addAll(xValueCol, valueCol);

        ObservableList<ItemData> itemDataList = FXCollections.observableArrayList();
```

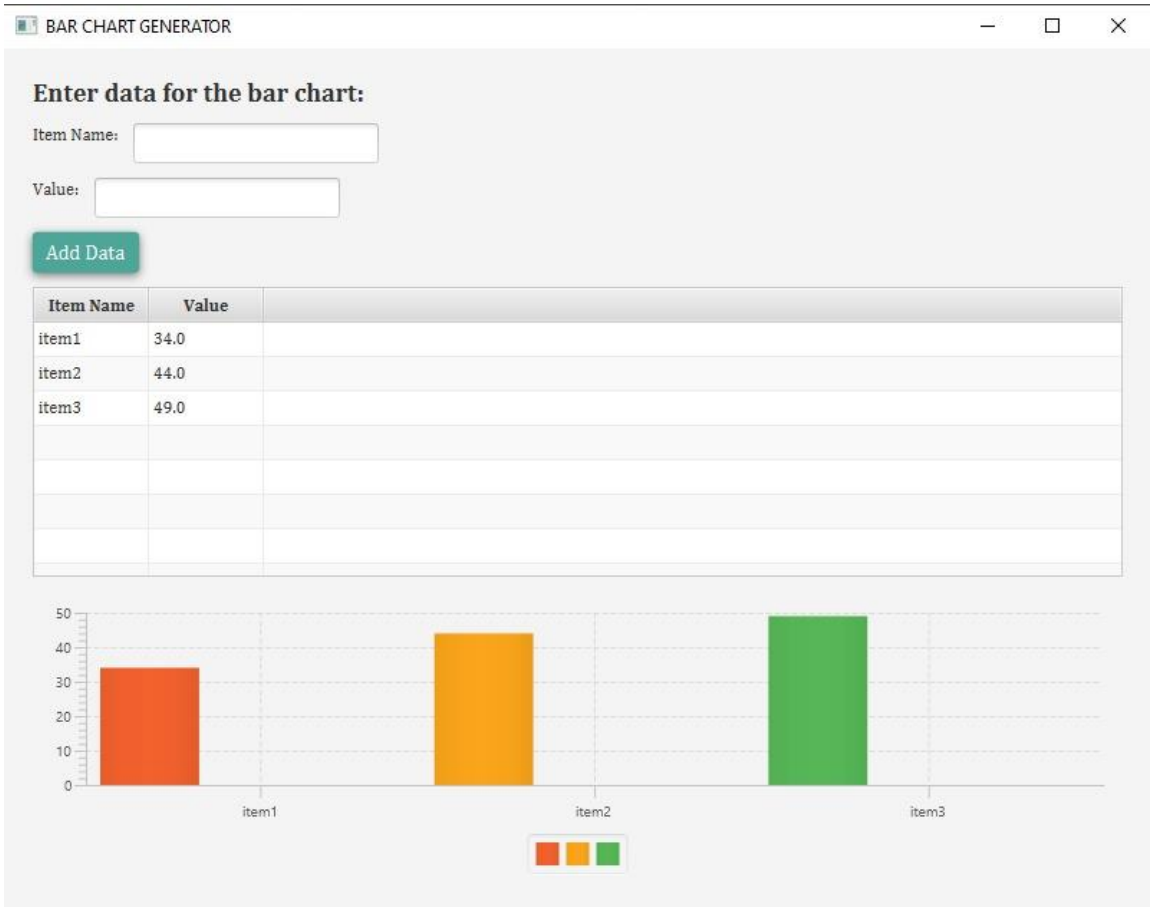
```
        addButton.setOnAction(event -> {
            try {
                double value = Double.parseDouble(valueInput.getText());

                int xValue = itemDataList.size() + 1;

                ItemData itemData = new ItemData(xValue, value);
                itemDataList.add(itemData);

                tableView.setItems(itemDataList);


                XYChart.Series<Number, Number> series = new XYChart.Series<>();
                series.getData().add(new XYChart.Data<>(xValue, value));


                for (int i = 0; i < itemDataList.size() - 1; i++) {
                    ItemData currentItem = itemDataList.get(i);
                    ItemData nextItem = itemDataList.get(i + 1);
                    XYChart.Data<Number, Number> currentPoint = new
XYChart.Data<>(currentItem.getXValue(), currentItem.getYValue());
                    XYChart.Data<Number, Number> nextPoint = new
XYChart.Data<>(nextItem.getXValue(), nextItem.getYValue());
                    series.getData().addAll(currentPoint, nextPoint);
                }

                lineChart.getData().add(series);

                valueInput.clear();
            } catch (NumberFormatException e) {
                System.out.println("Invalid input. Please enter a valid number for the
value.");
            }
        });


        VBox root = new VBox(10, label, new HBox(10, new Label("Value:"),
valueInput), addButton, tableView, lineChart);
        root.setStyle("-fx-padding: 20px; -fx-background-color: #f4f4f4;-fx-font-family:
Cambria");


        Scene scene = new Scene(root, 800, 600);
        primaryStage.setTitle("LINE CHART GENERATOR");
        primaryStage.setScene(scene);
```

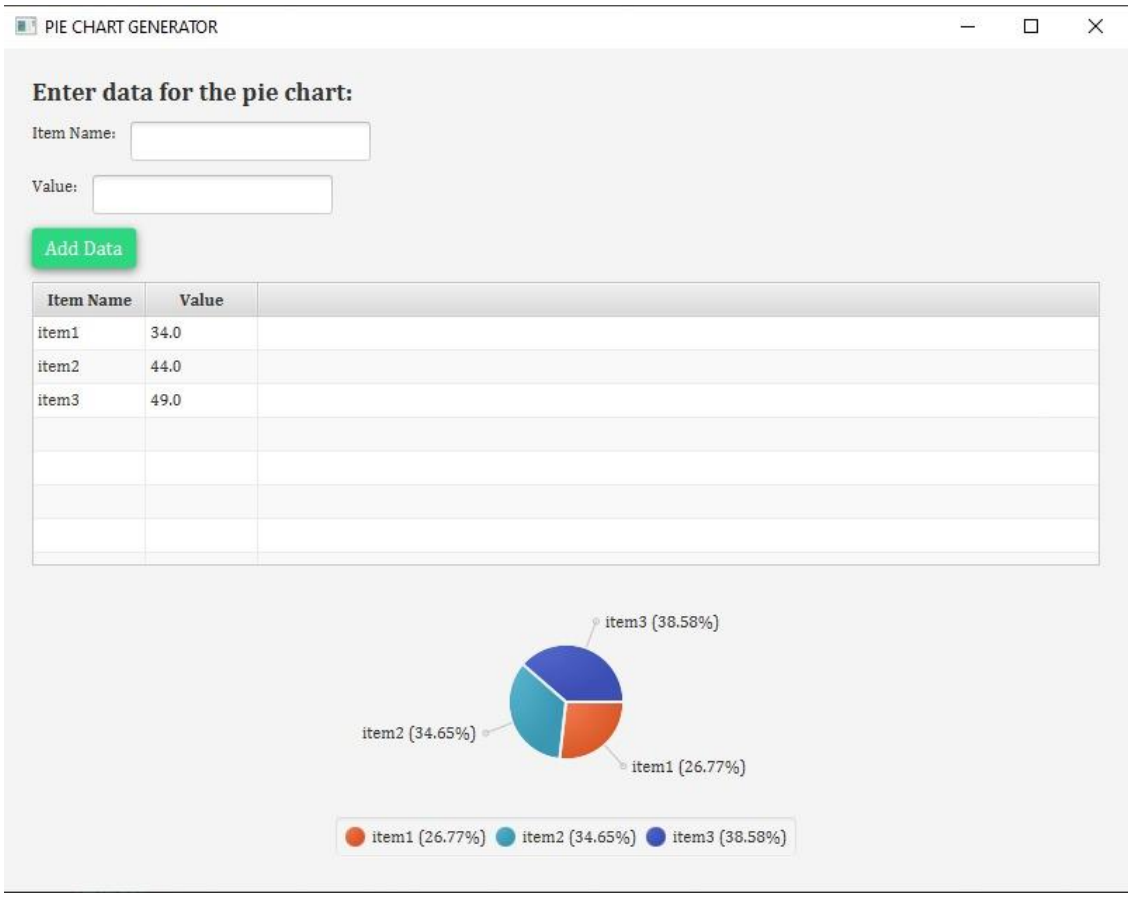
```
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }


    public static class ItemData {
        private final SimpleDoubleProperty xValue;
        private final SimpleDoubleProperty yValue;

        public ItemData(double xValue, double yValue) {
            this.xValue = new SimpleDoubleProperty(xValue);
            this.yValue = new SimpleDoubleProperty(yValue);
        }

        public double getXValue() {
            return xValue.get();
        }

        public SimpleDoubleProperty xValueProperty() {
            return xValue;
        }

        public double getYValue() {
            return yValue.get();
        }

        public SimpleDoubleProperty yValueProperty() {
            return yValue;
        }
    }
}
```

IV. Results


OOPJ MINI PROJECT
Interactive Data Visualization Tool
C034, C044, C049
Pie Chart
Bar Chart
Line Chart


LINE CHART GENERATOR
Enter data for the line chart:
Value:
Add Data
Sr. No.
Value
1.0
34.0
2.0
44.0
3.0
49.0
4.0
100.0
5.0
0.0
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5


BAR CHART GENERATOR
Enter data for the bar chart:
Item Name:
Value:
Add Data
Item Name
Value
item1
34.0
item2
44.0
item3
49.0
50
40
30
20
10
0
item1
item2
item3


PIE CHART GENERATOR
Enter data for the pie chart:
Item Name:
Value:
Add Data
Item Name
Value
item1
34.0
item2
44.0
item3
49.0
item3 (38.58%)
item2 (34.65%)
item1 (26.77%)
item1 (26.77%)
item2 (34.65%)
item3 (38.58%)

V. Conclusion

In conclusion, the interactive data visualization tool developed using JavaFX offers a powerful and user-friendly solution for individuals and businesses to analyze and present their data effectively. With a diverse range of chart options, robust data validation, and interactive features, the tool empowers users to gain valuable insights from their data with ease.

The project's focus on user experience is evident in the intuitive interface, comprehensive documentation, and seamless navigation options. Thorough testing ensures the tool's reliability and functionality across different data scenarios, while continuous support and updates based on user feedback guarantee its relevance and adaptability.

By providing users with the ability to create and customize Pie Charts, Bar Graphs, and Line Graphs, the tool enables them to visualize trends, patterns, and distributions in their data, facilitating informed decision-making and communication of insights.

Overall, the project represents a significant step towards simplifying data analysis and visualization, enhancing productivity, and fostering data-driven decision-making processes.

VI. References

- Sharma, Deepti. (2017). Developing applications using JavaFX. International Journal of Innovative Research in Computer and Communication Engineering. 5. 6672-6676.
- Oracle Corporation. (Year). "3D Features Planned for Version 8". Chien Yang. Retrieved from https://www.researchgate.net/publication/316959919_Developing_applications_using_JavaFX
- Oracle Corporation. (n.d.). JavaFX Charts: Creating Charts Using JavaFX. Retrieved from https://docs.oracle.com/javafx/2/charts/jfxpub-charts.htm
- Oracle Corporation. (n.d.). JavaFX Charts: Pie Chart. Retrieved from https://docs.oracle.com/javafx/2/charts/pie-chart.htm
- Oracle Corporation. (n.d.). JavaFX Charts: Line Chart. Retrieved from https://docs.oracle.com/javafx/2/charts/line-chart.htm
- Oracle Corporation. (n.d.). JavaFX Charts: Bar Chart. Retrieved from https://docs.oracle.com/javafx/2/charts/bar-chart.htm
- Oracle Corporation. (n.d.). JavaFX UI Controls: TableView. Retrieved from https://docs.oracle.com/javafx/2/ui_controls/table-view.htm
- Oracle Corporation. (n.d.). JavaFX ObservableList. Retrieved from https://docs.oracle.com/javase%2F9%2Fdocs%2Fapi%2F%2F/javafx/collections/ObservableList.html