

CURRENCY CONVERTER APPLICATION



A PROJECT REPORT

Submitted by

JOSHINIKITHA V

(2303811710422070)

in partial fulfillment of requirements for the award of the course

CGB1201 - JAVA PROGRAMMING

In

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

NOVEMBER- 2024

**K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(AUTONOMOUS)**

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report on “**CURRENCY CONVERTER APPLICATION**” is the bonafide work of **JOSHINIKITHA V (2303811710422070)** who carried out the project work during the academic year 2024 - 2025 under my supervision.

CGB1201-JAVA PROGRAMMING
Dr.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR

CGB1201-JAVA PROGRAMMING
Mrs.K.VALLI PRIYADHARSHINI, M.E.,(Ph.D.),
SUPERVISOR
ASSISTANT PROFESSOR

SIGNATURE

Dr.A.Delphin Carolina Rani, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT

PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

SIGNATURE

Mrs.K.Valli Priyadharshini, M.E.,(Ph.D.),

SUPERVISOR

ASSISTANT PROFESSOR

Department of CSE

K.Ramakrishnan College of Technology
(Autonomous)

Samayapuram–621112.

Submitted for the viva-voce examination held on ...3.12.2024.....

CGB1201-JAVA PROGRAMMING
Mr.MANJARMANNAN A, M.E.,
INTERNAL EXAMINER
ASSISTANT PROFESSOR

INTERNAL EXAMINER

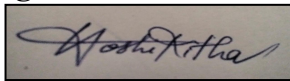
CGB1201-JAVA PROGRAMMING
Mrs.K.ARUNA PRIYA, M.E.,
EXTERNAL EXAMINER
ASSISTANT PROFESSOR
8104-DSEC, PERAMBALUR.

EXTERNAL EXAMINER

DECLARATION

I declare that the project report on “**CURRENCY CONVERTER APPLICATION**” is the result of original work done by us and best of our knowledge, similar work has not been submitted to “**ANNA UNIVERSITY CHENNAI**” for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the completion of the course **CGB1201 - JAVA PROGRAMMING**.

Signature



JOSHINIKITHA V

Place: Samayapuram

Date: 3.12.2024

ACKNOWLEDGEMENT

It is with great pride that I express our gratitude and in-debt to our institution “**K.Ramakrishnan College of Technology (Autonomous)**”, for providing us with the opportunity to do this project.

I glad to credit honourable chairman **Dr. K. RAMAKRISHNAN, B.E.**, for having provided for the facilities during the course of our study in college.

I would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.**, for forwarding to our project and offering adequate duration in completing our project.

I would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.**, Principal, who gave opportunity to frame the project the full satisfaction.

I whole heartily thanks to **Dr. A. DELPHIN CAROLINA RANI, M.E., Ph.D.**, Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

I express our deep expression and sincere gratitude to our project supervisor **Mrs. K. VALLI PRIYADHARSHINI, M.E., (Ph.D.)**, Department of **COMPUTER SCIENCE AND ENGINEERING**, for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

I render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

I wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

VISION OF THE INSTITUTION

To serve the society by offering top-notch technical education on par with global standards

MISSION OF THE INSTITUTION

- Be a center of excellence for technical education in emerging technologies by exceeding the needs of the industry and society.
- Be an institute with world class research facilities
- Be an institute nurturing talent and enhancing the competency of students to transform them as all-round personality respecting moral and ethical values

VISION OF DEPARTMENT

To be a center of eminence in creating competent software professionals with research and innovative skills.

MISSION OF DEPARTMENT

M1: Industry Specific: To nurture students in working with various hardware and software platforms inclined with the best practices of industry.

M2: Research: To prepare students for research-oriented activities.

M3: Society: To empower students with the required skills to solve complex technological problems of society.

PROGRAM EDUCATIONAL OBJECTIVES

1. PEO1: Domain Knowledge

To produce graduates who have strong foundation of knowledge and skills in the field of Computer Science and Engineering.

2. PEO2: Employability Skills and Research

To produce graduates who are employable in industries/public sector/research organizations or work as an entrepreneur.

3. PEO3: Ethics and Values

To develop leadership skills and ethically collaborate with society to tackle real-world challenges.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO 1: Domain Knowledge

To analyze, design and develop computing solutions by applying foundational concepts of Computer Science and Engineering.

PSO 2: Quality Software

To apply software engineering principles and practices for developing quality software for scientific and business applications.

PSO 3: Innovation Ideas

To adapt to emerging Information and Communication Technologies (ICT) to innovate ideas and solutions to existing/novel problems

PROGRAM OUTCOMES (POs)

Engineering students will be able to:

- 1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- 2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
- 3. Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
- 4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions

5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

ABSTRACT

This project presents a Real-Time Currency Converter that integrates cutting-edge API technology to provide users with accurate and up-to-date currency exchange rates. The system not only supports instant conversion between a wide range of currencies but also incorporates a visual representation of exchange rate trends through dynamic graph rendering. In an increasingly globalized economy, currency conversion plays a pivotal role in financial transactions, trade, and travel. By leveraging the Exchangerate API, the application offers real-time insights and a user-friendly interface, ensuring precision and accessibility for both casual users and financial professionals. Its modular architecture and error-handling mechanisms ensure robustness, while the extensibility of the design allows for future enhancements, such as integration with investment tools and trade applications. Built with Java's AWT and Swing, the tool offers a responsive and intuitive interface while implementing robust error-handling mechanisms to ensure seamless user experience despite invalid inputs, API issues, or network failures. This project serves as a powerful demonstration of the practical applications of API integration, graphical data visualization, and intuitive GUI design in addressing real-world financial challenges.

ABSTRACT WITH POs AND PSOs MAPPING

CO 5 : BUILD JAVA APPLICATIONS FOR SOLVING REAL-TIME PROBLEMS.

ABSTRACT	POs MAPPED	PSOs MAPPED
The Real-Time Currency Converter is a dynamic tool designed to provide accurate currency conversions by leveraging real-time data from the ExchangerateAPI built using Java's AWT. The application features an interactive graph that visualizes exchange rate trends, offering deeper insights into currency fluctuations over time	PO1 -3 PO2 -3 PO3 -3 PO4 -3 PO5 -3 PO6 -3 PO7 -3 PO8 -3 PO9 -3 PO10 -3 PO11-3 PO12 -3	PSO1 -3 PSO2 -3 PSO3 -3

Note: 1- Low, 2-Medium, 3- High

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	
1	INTRODUCTION	
	1.1 Objective	11
	1.2 Overview	11
	1.3 Java Programming concepts	12
2	PROJECT METHODOLOGY	
	2.1 Proposed Work	13
	2.2 Block Diagram	13
3	MODULE DESCRIPTION	
	3.1 User Interface Module	14
	3.2 Input Module	14
	3.3 Event Handling Module	14
	3.4 Currency Conversion Module	15
	3.5 Graph Visualization Module	15
	3.6 Output Module	15
	3.7 Error handling Module	15
4	CONCLUSION & FUTURE SCOPE	
	4.1 Conclusion	16
	4.2 Future Scope	16
	REFERENCES	29
	APPENDIX A (SOURCE CODE)	17
	APPENDIX B (SCREENSHOTS)	27

CHAPTER 1

INTRODUCTION

1.1 Objective

To design and develop a currency converter application that allows users to convert amounts between different currencies in real-time. The application should fetch the latest exchange rates from a reliable source and provide accurate conversion results. The goal is to offer users a quick and easy way to convert currency in real-time, whether for travel, trading, or everyday transaction.

1.2 Overview

The currency converter project is a software application designed to facilitate real-time currency conversions using live exchange rate data. It features a user-friendly graphical interface that allows users to input the amount to be converted, select the source and target currencies, and view the conversion result. The system integrates with a currency exchange API to fetch the latest exchange rates and perform the conversion. Additionally, it offers a graph visualization of historical exchange rate trends to help users understand currency fluctuations. The project is built using Java and applies key Object-Oriented Programming concepts such as classes, encapsulation, inheritance, polymorphism, and exception handling to ensure a modular, scalable, and reliable solution. The tool is designed to be simple, efficient, and extendable, with potential for future enhancements such as mobile compatibility, multi-currency support, and advanced financial features.

1.3 Java Programming Concepts

- ❖ **Classes and Objects:** Organize the code into distinct modules, each represented by a class that encapsulates related data and methods.
- ❖ **Encapsulation:** Protects data by restricting access to internal class members and exposing only necessary methods. **Inheritance:** Allows for code reuse by creating new classes based on existing ones, inheriting common functionalities.
- ❖ **Polymorphism:** Enables the use of the same method name with different implementations, allowing flexibility in method behavior.
- ❖ **Abstraction:** Hides complex details and exposes only the essential features, simplifying the user interface and interactions.
- ❖ **Interfaces:** Define common behaviors that classes must implement, ensuring consistency and flexibility in the design.
- ❖ **Exception Handling:** Manages errors gracefully, preventing the application from crashing and providing user-friendly feedback.

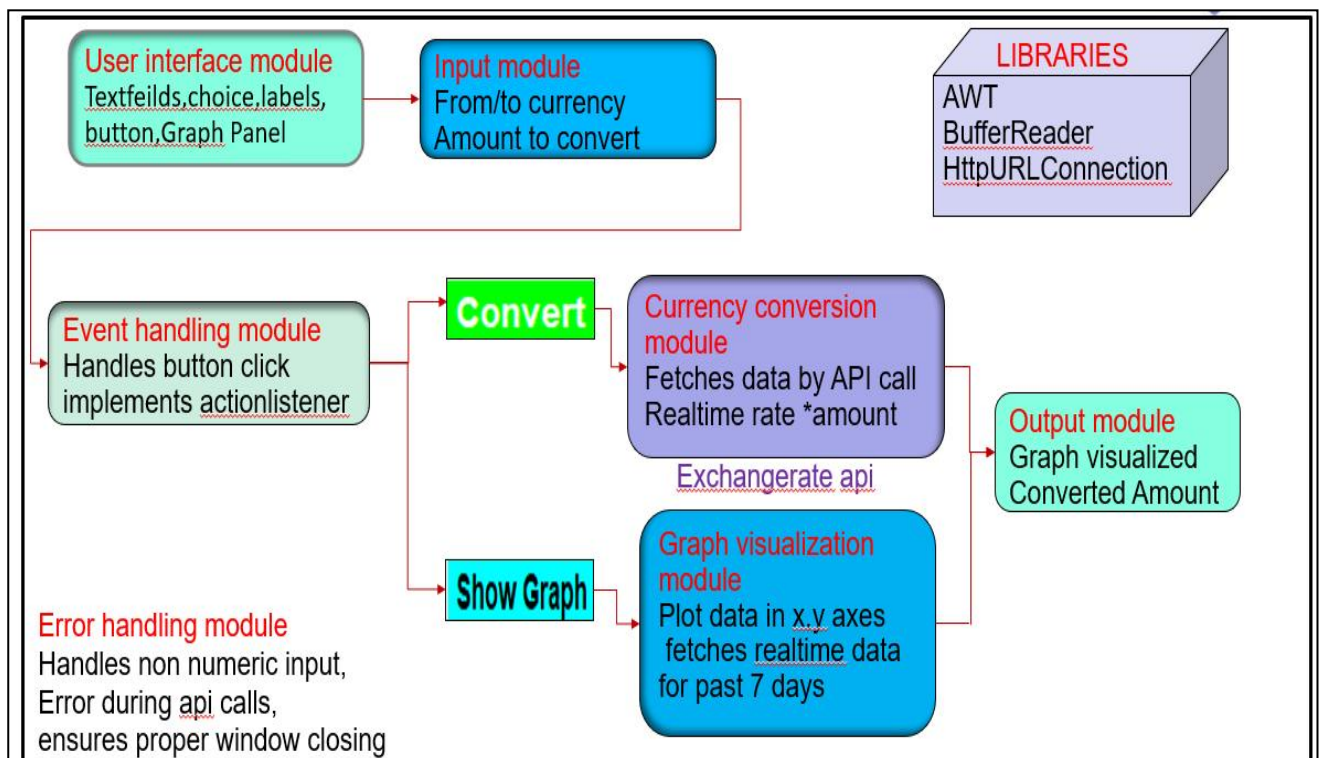
CHAPTER 2

PROJECT METHODOLOGY

2.1 Proposed Work

The proposed work for this currency converter project involves developing a real-time currency conversion tool that provides users with accurate, up-to-date exchange rates and visualizes historical trends. The system will have a user-friendly graphical interface, allowing users to input the amount to be converted, select source and target currencies, and display the results in both numerical and graphical formats. It will integrate with a currency exchange API to fetch real-time exchange rate data and perform the conversion. Additionally, the project will include a graph visualization module to display the historical trends of currency exchange rates over a specified period, such as the past 7 days.

2.2 Block Diagram



CHAPTER 3

MODULE DESCRIPTION

3.1 User Interface Module

The User Interface Module is responsible for creating the graphical interface that connects the user to the system. It includes components such as text fields for entering the amount to be converted, drop-down menus for selecting the source and target currencies, buttons for initiating actions like conversion and graph display, and a graph panel for visualizing historical exchange rate trends. This module ensures that the system is easy to use and visually appealing, providing users with a seamless experience.

3.2 Input Module

The Input Module handles the collection and validation of user input. It captures the "From" and "To" currencies, as well as the amount to be converted. This module ensures that the inputs are correctly formatted and valid before passing them to the backend processes. It plays a critical role in ensuring that erroneous or incomplete data does not disrupt the system.

3.3 Event Handling Module

The Event Handling Module manages user interactions with the interface. It listens for actions, such as button clicks, using an ActionListener mechanism, and triggers the appropriate backend operations. For example, it starts the conversion process when the "Convert" button is clicked or fetches and displays a graph when the "Show Graph" button is pressed. This module acts as the bridge between user actions and system functionalities.

3.4 Currency Conversion Module

The Currency Conversion Module performs the main task of the system: converting currencies. It fetches real-time exchange rates from an API, such as the ExchangeRate API, and calculates the converted amount using the formula: $\text{Converted Amount} = \text{Input Amount} \times \text{Exchange Rate}$. This module ensures the conversion is accurate and up-to-date, making it a core component of the project.

3.5 Graph Visualization Module

The Graph Visualization Module enhances the system by providing a visual representation of historical exchange rate trends. It fetches data for a specified period (e.g., the past seven days) and plots it on a graph, with dates on the X-axis and exchange rates on the Y-axis.

3.6 Output Module

The Output Module is responsible for displaying results to the user. It shows the converted amount in a clear and readable format and presents the graph generated by the Graph Visualization Module.

3.7 Error Handling Module

The Error Handling Module ensures the system's robustness by managing potential issues. It detects and handles invalid inputs, such as non-numeric values, manages errors during API calls (e.g., network issues or API unavailability), and ensures proper system behavior, like closing windows gracefully during unexpected scenarios. This module provides meaningful feedback to users and ensures smooth operation even under circumstances.

CHAPTER 4

CONCLUSION & FUTURE SCOPE

4.1 CONCLUSION

The currency converter project serves as an effective tool for converting currencies in real-time using a user-friendly interface and integrating live exchange rate data. By leveraging Java's GUI capabilities (AWT), APIs for fetching exchange rates, and data visualization tools, the system provides a comprehensive solution that is both functional and accessible. The inclusion of modules for error handling and input validation ensures robustness, while the graphical representation of historical data offers valuable insights into currency fluctuations. This system not only fulfills the basic functionality of currency conversion but also enhances user experience through clear output and visual aids.

4.2 FUTURE SCOPE

The future scope of the currency converter project offers various opportunities for expansion and enhancement. One potential development is adapting the system for mobile platforms, such as iOS and Android, using frameworks like Flutter or React Native, to increase accessibility for users. Additionally, incorporating multi-language support would make the tool more versatile, allowing it to cater to a global audience. The system could also be improved by integrating real-time market data, such as currency trends and forecasts, which would add value for users needing more in-depth financial insights. Expanding the project to include additional features like currency charts, financial news, and alerts for specific exchange rate thresholds would make it a comprehensive financial tool. Furthermore, integrating AI-based models to predict future exchange rate trends using historical data could provide users with more predictive and analytical capabilities.

APPENDIX A

(SOURCE CODE)

```
import java.awt.*;
import java.awt.event.*;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import org.json.JSONObject;

public class CurrencyConverter extends Frame implements ActionListener {
    // Declare components
    Label lblHeader, lblFooter, lblFrom, lblTo, lblAmount, lblResult, lblInfo;
    TextField txtAmount, txtResult;
    Choice fromCurrency, toCurrency;
    Button convertButton, showGraphButton;
    public CurrencyConverter() {
        // Set up the frame
        setTitle("Real-Time Currency Converter");
        setSize(400, 500); // Adjusted size
        setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.insets = new Insets(10, 10, 10, 10); // Add padding
        setBackground(new Color(240, 248, 255));
        // Header
        lblHeader = new Label("REAL-TIME CURRENCY CONVERTER",
Label.CENTER);
        lblHeader.setFont(new Font("Arial", Font.BOLD, 16));
        gbc.gridx = 0;
```

```

gbc.gridy = 0;
gbc.gridwidth = 2;
add(lblHeader, gbc);
// Label for "Currencyrate from exchangerateapi.com" (just below header with no
space)
Label lblSnippet = new Label("CurrencyRate from exchangerateapi.com",
Label.CENTER);
lblSnippet.setFont(new Font("Arial", Font.PLAIN, 10)); // Set small font
lblSnippet.setForeground(Color.GRAY); // Set grey color

// Adjust GridBagConstraints for lblSnippet
gbc.gridy = 1; // Position it immediately below the header, no space
gbc.insets = new Insets(0, 0, 20, 0); // No padding
gbc.weighty = 0.1; // Allow a little vertical space, but not stretching
gbc.anchor = GridBagConstraints.CENTER; // Center the label
add(lblSnippet, gbc);
// Label for "Customizing conversion between currencies" (with a very small
vertical space)
lblInfo = new Label("Customizing conversion between currencies",
Label.CENTER);
lblInfo.setFont(new Font("Arial", Font.PLAIN, 14)); // Same font size as before
lblInfo.setForeground(Color.BLUE); // Set blue color

// Adjust GridBagConstraints for lblInfo
gbc.gridy = 2; // Place it below the previous label
gbc.insets = new Insets(1, 0, 30, 0); // Very small vertical space (top and
bottom)
add(lblInfo, gbc);

```

```

// From Currency
lblFrom = new Label("From Currency  :");
gbc.gridx = 0;
gbc.gridy = 3;
gbc.anchor = GridBagConstraints.WEST;
add(lblFrom, gbc);
fromCurrency = new Choice();
String[] currencies = { "USD", "EUR", "INR", "GBP", "JPY", "AUD",
"CAD" };
for (String currency : currencies) {
    fromCurrency.add(currency);
}
gbc.gridx = 1;
gbc.anchor = GridBagConstraints.WEST;
add(fromCurrency, gbc);
// To Currency
lblTo = new Label("To Currency  :");
gbc.gridx = 0;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.WEST;
add(lblTo, gbc);
toCurrency = new Choice();
for (String currency : currencies) {
    toCurrency.add(currency);
}
gbc.gridx = 1;
gbc.anchor = GridBagConstraints.WEST;
add(toCurrency, gbc);
// Amount
lblAmount = new Label("Amount to convert:");
gbc.gridx = 0;
gbc.gridy = 5;

```

```

gbc.anchor = GridBagConstraints.WEST;
add(lblAmount, gbc);
txtAmount = new TextField(10);
gbc.gridx = 1;
gbc.anchor = GridBagConstraints.WEST;
add(txtAmount, gbc);
// Buttons - Convert and Show Graph placed side by side
Panel buttonPanel = new Panel();
buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
convertButton = new Button("Convert");
convertButton.setBackground(Color.GREEN);
convertButton.setForeground(Color.WHITE);
convertButton.setFont(new Font("Arial", Font.BOLD, 14));
buttonPanel.add(convertButton);

showGraphButton = new Button("Show Graph");
showGraphButton.setBackground(Color.CYAN);
showGraphButton.setForeground(Color.BLACK);
showGraphButton.setFont(new Font("Arial", Font.BOLD, 14));
buttonPanel.add(showGraphButton);
gbc.gridx = 1;
gbc.gridy = 6;
gbc.gridwidth = 2;
add(buttonPanel, gbc);
lblResult = new Label("Converted Amount:");
gbc.gridx = 0;
gbc.gridy = 7;
gbc.gridwidth = 1;
gbc.anchor = GridBagConstraints.EAST;

```

```

add(lblResult, gbc);
txtResult = new TextField(10);
txtResult.setEditable(false);
gbc.gridx = 1;
gbc.anchor = GridBagConstraints.WEST;
add(txtResult, gbc); // Footer
lblFooter = new Label("Made by Joshi", Label.CENTER);
lblFooter.setFont(new Font("Arial", Font.ITALIC, 12));
gbc.gridx = 1;
gbc.gridy = 8;
gbc.gridwidth = 2;
add(lblFooter, gbc);
// Add action listeners for the buttons
convertButton.addActionListener(this);
showGraphButton.addActionListener(this);
// Add window listener for closing
addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        dispose();
    } }); setVisible(true); }

@Override
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == convertButton) {
        try { String from = fromCurrency.getSelectedItem();
            String to = toCurrency.getSelectedItem();
            double amount = Double.parseDouble(txtAmount.getText());
            double convertedAmount = getConvertedAmount(from, to, amount);
            txtResult.setText(String.format("%.2f", convertedAmount));
        } catch (Exception ex) {

```

```

        txtResult.setText("Error!");
    }
} else if (e.getSource() == showGraphButton) {
    String from = fromCurrency.getSelectedItemAt();
    String to = toCurrency.getSelectedItemAt();
    showGraph(from, to);
} }

private double getConvertedAmount(String from, String to, double amount) {
    String apiKey = "5c9bb566ed62b8722511f24b"; // Replace with your actual
API key

    String apiUrl = "https://v6.exchangerate-api.com/v6/" + apiKey + "/latest/" +
from; try {
        // Create URL and open connection
        URL url = new URL(apiUrl);
        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
        connection.setRequestMethod("GET");
        // Read response
        BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuilder response = new StringBuilder();

        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        } in.close();
        // Parse JSON response
        JSONObject jsonResponse = new JSONObject(response.toString());
        JSONObject conversionRates =

```

```

jsonResponse.getJSONObject("conversion_rates");
    // Get conversion rate and calculate result
    double rate = conversionRates.getDouble(to);
    return amount * rate;
} catch (Exception ex) {
    ex.printStackTrace(); }
return 0.0; }

private void showGraph(String from, String to) {
    String apiKey = "5c9bb566ed62b8722511f24b";
    String apiUrl = "https://v6.exchangerate-api.com/v6/" + apiKey + "/latest/" +
from;    try {
        URL url = new URL(apiUrl);
        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
        connection.setRequestMethod("GET");
        BufferedReader in = new BufferedReader(new
InputStreamReader(connection.getInputStream()));
        String inputLine;
        StringBuilder response = new StringBuilder();
        while ((inputLine = in.readLine()) != null) {
            response.append(inputLine);
        } in.close();
        JSONObject jsonResponse = new JSONObject(response.toString());
        JSONObject                conversionRates                =
jsonResponse.getJSONObject("conversion_rates");
        // Mock data: Populate graph with random values for demo purposes
        double[] rates = new double[7];
        for (int i = 0; i < 7; i++) {
            rates[i] = conversionRates.getDouble(to) + (Math.random() - 0.5);

```

```

    } // Display graph
    JFrame graphFrame = new JFrame("Exchange Rate Graph");
    graphFrame.setSize(800, 600);
    graphFrame.add(new GraphPanel(rates, from, to));
    graphFrame.setVisible(true);
} catch (Exception ex) {
    ex.printStackTrace();
    showErrorDialog("Failed to fetch exchange rate data.");
} } private void showErrorDialog(String message) {
    Dialog errorDialog = new Dialog(this, "Error", true);
    errorDialog.setSize(300, 150);
    errorDialog.setLocation(250, 100);
    errorDialog.add(new Label(message));
    Button okButton = new Button("OK");
    okButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            errorDialog.setVisible(false);
        } });
    errorDialog.add(okButton);
    errorDialog.setVisible(true);
} // Inner class for GraphPanel
class GraphPanel extends Panel {
    private final double[] rates;
    private final String fromCurrency;
    private final String toCurrency;

    public GraphPanel(double[] rates, String fromCurrency, String toCurrency) {
        this.rates = rates;
        this.fromCurrency = fromCurrency;
    }
}

```



```

        this.toCurrency = toCurrency;
    }

    @Override
    public void paint(Graphics g) {
        int width = getWidth();
        int height = getHeight();
        int margin = 50;
        // Draw axes
        g.setColor(Color.BLACK);
        g.drawLine(margin, height - margin, width - margin, height - margin);
        g.drawLine(margin, margin, margin, height - margin); // Y-axis
        // Draw title
        g.setFont(new Font("TimesNewRoman", Font.BOLD, 16));
        g.drawString("Exchange Rate Graph: " + fromCurrency + " to " +
toCurrency, width / 4, margin / 2);
        // Draw X and Y labels
        g.setFont(new Font("Arial", Font.BOLD, 12));
        g.drawString("Time (Days)", width / 2, height - margin / 5); // X label
        // Rotate the Y-axis label
        Graphics2D g2d = (Graphics2D) g;
        g2d.rotate(-Math.PI / 2);
        g.drawString("Exchange Rate", -height / 2, margin / 3);
        g2d.rotate(Math.PI / 2); // Reset rotation
        // Scale rates
        double maxRate = Double.MIN_VALUE;
        double minRate = Double.MAX_VALUE;
        for (double rate : rates) {
            maxRate = Math.max(maxRate, rate);

```

```

        minRate = Math.min(minRate, rate);
    }
    // Plot data points and draw X, Y points
    int numPoints = rates.length;
    int graphWidth = width - 2 * margin;
    int graphHeight = height - 2 * margin;
    for (int i = 0; i < numPoints - 1; i++) {
        int x1 = margin + (i * graphWidth / (numPoints - 1));
        int y1 = height - margin - (int) ((rates[i] - minRate) / (maxRate - minRate)
* graphHeight);
        int x2 = margin + ((i + 1) * graphWidth / (numPoints - 1));
        int y2 = height - margin - (int) ((rates[i + 1] - minRate) / (maxRate -
minRate) * graphHeight);
        g.setColor(Color.BLUE);
        g.drawLine(x1, y1, x2, y2); // Connect points
        // Draw coordinates as points on the graph
        g.setColor(Color.RED);
        g.fillOval(x1 - 3, y1 - 3, 6, 6); // Draw point at (x1, y1)
        g.fillOval(x2 - 3, y2 - 3, 6, 6); // Draw point at (x2, y2)
        // Annotate points with their coordinates
        g.setColor(Color.BLACK);
        g.drawString("(" + i + ", " + String.format("%.2f", rates[i]) + ")", x1 + 5,
y1 - 5);
        g.drawString("(" + (i + 1) + ", " + String.format("%.2f", rates[i + 1]) + ")",
x2 + 5, y2 - 5);
    } } } public static void main(String[] args) {
    new CurrencyConverter();
}

```

APPENDIX B

Real-Time Currency Converter

CurrencyRate from exchangerateapi.com

Customizing conversion between currencies

From Currency : **USD** ▼

To Currency : USD ▼

Amount to convert:

Convert **Show Graph**

Converted Amount:

Made by Joshi

Currency converter

Real-Time Currency Converter

CurrencyRate from exchangerateapi.com

Customizing conversion between currencies

From Currency : USD ▼

To Currency : **USD**
EUR
INR
GBP
JPY
AUD
CAD

Amount to convert:

Convert **Show Graph**

Converted Amount:

Made by Joshi

From currency

Real-Time Currency Converter

CurrencyRate from exchangerateapi.com

Customizing conversion between currencies

From Currency : USD ▼

To Currency : **USD** ▼
USD
EUR
INR
GBP
JPY
AUD
CAD

Amount to convert:

Convert **Show Graph**

Converted Amount:

Made by Joshi

To currency

Real-Time Currency Converter

CurrencyRate from exchangerateapi.com

Customizing conversion between currencies

From Currency : USD ▼

To Currency : JPY ▼

Amount to convert:

Convert **Show Graph**

Converted Amount:

Made by Joshi

Amount to convert

Real-Time Currency Converter

REAL-TIME CURRENCY CONVERTER

CurrencyRate from exchangerateapi.com

Customizing conversion between currencies

From Currency : USD

To Currency : JPY

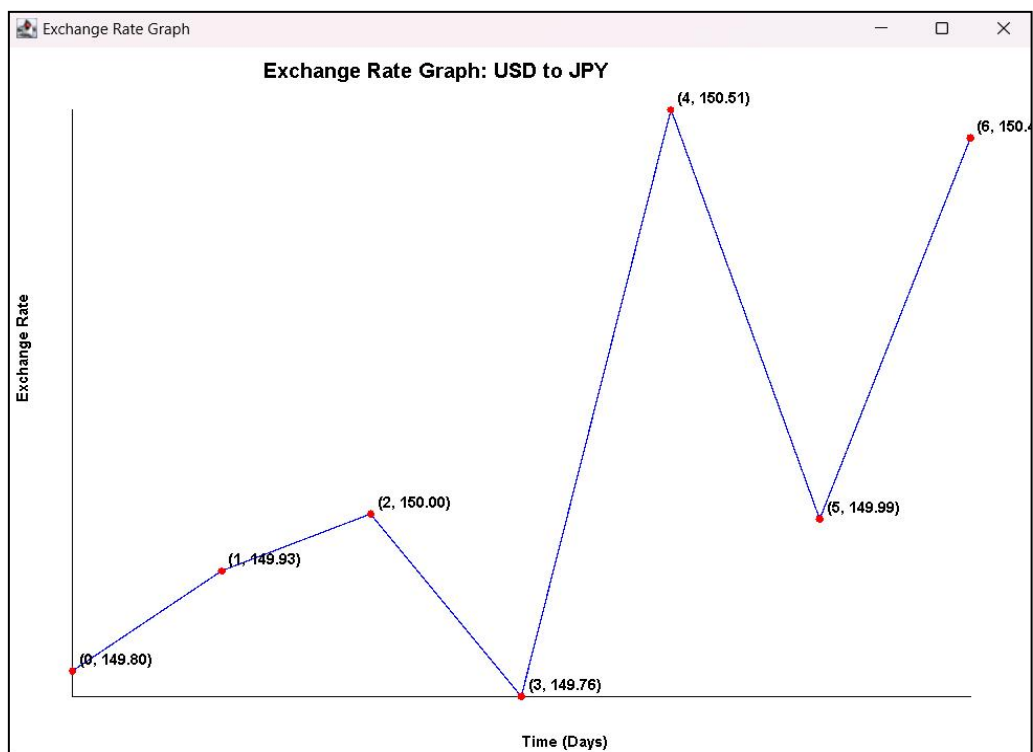
Amount to convert: 2000

Convert Show Graph

Converted Amount: 300170.20

Made by Joshi

Convert button result



Show graph button

REFERENCES

1. ExchangeRateAPI:
My API KEY: 5c9bb566ed62b8722511f24b
MyAPIURL:<https://v6.exchangerate-api.com/v6/5c9bb566ed62b8722511f24b/latest/USD>
2. JfreeChart
website:<https://docs.oracle.com/javase/8/docs/api/java/net/URLConnection.html>
3. JavaAWTDocumentation:<https://docs.oracle.com/javase/8/docs/api/java/awt/package-summary.html>
4. Java tutorials by oracle:<https://docs.oracle.com/javase/tutorial/>
5. Chatgpt: <https://chatgpt.com/c/674ae9b6-9550-8012-b6d3-ca142ec9e285>