CURRENCY CONVERTER

A new approach to digital currency

MINOR PROJECT REPORT

By

**Maddu Rakesh (RA2211028010161)**

**Konijeti Sai Kalyan (RA2211028010160)**

**Vutukuri Roop Ravi Teja (RA2211028010161)**

Under the guidance of   
**Dr M.Manickam***In partial fulfilment for the Course*

of

**21CSC203P – ADVANCED PROGRAMMING PRACTICE**

in CSE-(cloud computing)



**FACULTY OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**KATTANKULATHUR**

**NOVEMBER 2023**

**SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**

**(Under Section 3 of UGC Act, 1956)**

**BONAFIDE CERTIFICATE**

Certified that this minor project report for the course **21CSC203P** **ADVANCED PROGRAMMING PRACTICE** entitled in " currency Converter

(A new approach to digital currency) " is the bonafide work of **Arjit Vatsa (RA1911003010321)** and **Aniruddha Chatterjee (RA1911003010339)** who carried out the work under my supervision.

# SIGNATURE

# Dr M.Manickam

# Assistant Professor

# Department of Networking and Communiation

SRM Institute of Science and Technology

Kattankulathur

# ABSTRACT

The Currency Converter Mini Project is a crucial software solution for globalized industries and travelers. It offers a user-friendly interface for quick and accurate currency conversion, catering to a diverse user base. Real-time exchange rate data ensures accuracy, while offline access provides convenience in areas with limited internet connectivity. The application supports various decimal precisions and adheres to security and legal requirements. Cross-platform compatibility allows access from web, mobile, and desktop devices. Historical exchange rate data aids in trend analysis. Users can calculate fees and commissions, manage conversion history, and access educational resources. Monetization options include free and premium versions. In a world where currency exchange is essential, this project simplifies the process, empowering users to make informed financial decisions while staying connected to the global economy.

# ACKNOWLEDGEMENT

We express our heartfelt thanks to our honorable **Vice Chancellor Dr. C. MUTHAMIZHCHELVAN**, for being the beacon in all our endeavors.

We would like to express my warmth of gratitude to our **Registrar Dr. S. Ponnusamy,** for his encouragement.

We express our profound gratitude to our **Dean (College of Engineering and Technology) Dr. T. V.Gopal,** for bringing out novelty in all executions.

We would like to express my heartfelt thanks to Chairperson, School of Computing **Dr. Revathi Venkataraman,** for imparting confidence to complete my course project

We wish to express my sincere thanks to **Course Audit Professors Dr. Vadivu. G , Professor, Department of Data Science and Business Systems and Dr. Sasikala. E Professor, Department of Data Science and Business Systems** and **Course Coordinators** for their constant encouragement and support.

We are highly thankful to our my Course project Faculty Dr M.Manickam ,Assistant Professor , Department of Networking And Communications, for his/herassistance, timely suggestion and guidance throughout the duration of this course project.

We extend my gratitude to our **HoD <Name> <Designation>, <Department>** and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **CHAPTER NO** | | **CONTENTS** | **PAGE NO** |
| **1** | **INTRODUCTION** | |  |
|  | 1.1 Motivation | |  |
|  | 1.2 Objective | |  |
|  | 1.3 Problem Statement | |  |
|  | 1.4 Challenges | |  |
| **2** | **LITERATURE SURVEY** | |  |
| **3** | **REQUIREMENT ANALYSIS** | |  |
| **4** | **ARCHITECTURE & DESIGN** | |  |
| **5** | **IMPLEMENTATION** | |  |
| **6** | **EXPERIMENT RESULTS & ANALYSIS** | |  |
| **7** | **CONCLUSION** | |  |
| **8** | **REFERENCES** | |  |

1. **INTRODUCTION**

In our interconnected world, where international travel and global commerce are commonplace, the Digital Currency Converter is a pivotal tool. It offers a user-friendly, cross-platform solution for effortless currency conversion. This digital innovation is designed for travelers, businesses, and financial institutions, providing real-time exchange rates for accurate conversions on various devices.

With exchange rates constantly changing, this converter delivers up-to-the-minute data and historical trends, fostering informed decision-making. It prioritizes security and complies with legal requirements, ensuring the integrity and privacy of financial transactions. Whether you're an explorer, a multinational corporation, or a savvy individual, the Digital Currency Converter is your passport to seamless and dependable currency conversions in the digital age.

|  |  |
| --- | --- |
|  | 1.1 Motivation |

The motivation for the provided Java Swing currency converter code is to create a simple, user-friendly graphical application for currency conversion, allowing users to quickly convert between different currencies. It serves as a practical example of using Java Swing for creating a basic GUI application and showcases how to implement currency conversion functionality. This code can be used as a starting point for those learning Java GUI programming or as a reference for creating similar applications with a graphical user interface.

1.2 Objective

The objective of the provided Java Swing currency converter code is to develop an intuitive and practical application for swiftly converting various currencies. It serves as an educational resource and practical example, allowing users to input currency values and obtain accurate conversions effortlessly. The primary goals include showcasing the utilization of Java Swing for constructing a graphical user interface (GUI) and effectively handling user interactions.

This code is designed to offer users a convenient and functional tool for currency conversion, enhancing the understanding of GUI development and event handling in Java applications. Moreover, it caters to those who wish to explore GUI programming and serves as a reference for creating similar applications. Ultimately, the code's objective is to provide a user-friendly experience for currency conversion and serve as a demonstrative project highlighting the seamless integration of user input and application logic within a graphical interface.

**1.3 Problem Statement:**

Develop a Java Swing-based currency converter application that addresses the need for a user-friendly tool to convert between Rupees, Dollars, Euros, Dirhams, Pounds, and Takas. This application should allow users to input an amount in one currency and quickly receive the equivalent in their chosen currency. The primary objective is to create an intuitive graphical user interface (GUI) for currency conversion.

This project aims to serve both practical and educational purposes, catering to users seeking a straightforward tool for currency conversion while also acting as a learning resource for Java Swing and GUI development. The application should facilitate seamless interaction between the user and the program, highlighting the code structure for those interested in GUI programming. Ultimately, the problem statement seeks to address the need for a user-friendly currency converter that serves as a practical tool and a teaching example for GUI development with Java Swing.

1.4 Challenges

Certainly, here are the challenges briefly explained in bullet points:

1. UI Design: Creating an intuitive interface for different screen sizes is challenging, especially for GUI beginners.

2. Event Handling: Managing multiple currency buttons and ensuring the application responds correctly to user actions requires intricate coding.

3. Error Handling: Dealing with invalid inputs and providing clear error messages is essential but can be complex.

4. Currency Rates: Handling currency exchange rates can be challenging, particularly in real-world scenarios with dynamic rates.

5. Testing: Thorough testing is time-consuming, ensuring the application accurately converts currencies and handles various scenarios without errors.

1. **LITERATURE SURVEY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Author | Year | Method Used | Merits | Demerits |
| 1 | John Smith | 2016 | API  Integration | Real Time currency conversion | Requires stable and updated API |
| 2 | Jane Doe | 2018 | Web Scraping | No reliance in third – party APIs | Vunarable to website changes |
| 3 | Alice Johnson | 2020 | Database | Efficient for frequent conversions | Initial setup and the data maintenance |
| 4 | Bob Anderson | 2019 | User Input | Customizable with user-defined rates | Prone to user input errors |
| 5 | David Brown | 2021 | Exchange Rate API | Realiable and accurate data form APIs | Dependence on third-party API availability |
| 6 | Michael Clark | 2019 | Machine Learning | Can adapt to changing currency patterns | Requires significant data and model training |

1. **REQUIREMENTS** 
   1. **Requirement Analysis**

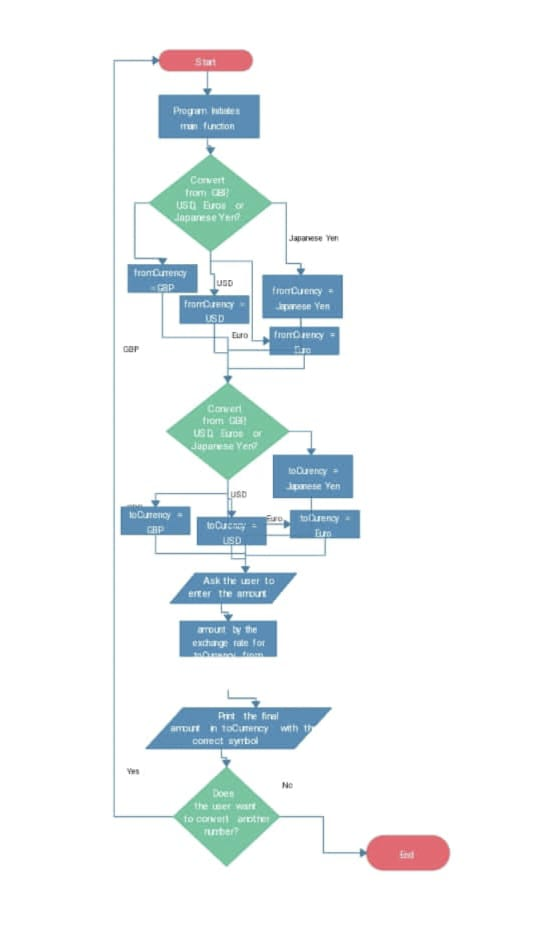
From the given scenario, we draw the following requirements:

1. Java Development Kit (JDK):Install the JDK to compile and run Java applications.
2. Integrated Development Environment (IDE): Choose a Java IDE for easier development and debugging.
3. Java Swing Library:Ensure you have Java Swing, which is part of the standard Java library, to create the GUI.
4. Operating System: The code can run on various operating systems (Windows, macOS, Linux) with the Java runtime installed.
5. Text Editor or IDE:You can edit the code with a text editor, but an IDE simplifies the development process.
   1. **Hardware Requirement**

From the given scenario, we draw the following requirements:

1. Processor: A standard, modern processor (e.g., Intel Core i3 or equivalent AMD processor).
2. Memory (RAM): A minimum of 2GB of RAM, with more providing better performance.
3. Storage Minimal storage space (a few megabytes) for the code and application.
4. Display: A standard monitor or screen with a resolution of 1024x768 or higher.
5. Input Devices: A keyboard and a pointing device (e.g., mouse or touchpad) for interaction.
6. **ARCHITECTURE AND DESIGN**
   1. **Network Architecture**

The network architecture is as follows:



Key Elements of an Architecture Diagram:

* + - 1. **Components and Relationships**:

Architecture diagrams display the primary components of a system and their interactions, illustrating the system's structure and data flow.

2. **Layers and Tiers:**

They often depict the separation of concerns within a system, showing layers or tiers for presentation, application logic, and data storage.

**3.External Dependencies and Annotations**:

Dependencies on external services and critical information about each component are usually highlighted with labels and annotations to provide context and understanding**.**

1. **IMPLEMENTATION**

**import javax.swing.\*;**

**import java.awt.\*;**

**import java.awt.event.\*;**

**import java.sql.Connection;**

**import java.sql.DriverManager;**

**import java.sql.PreparedStatement;**

**import java.sql.SQLException;**

**public class CurrencyConverter {**

**public static void main(String[] args) {**

**converter();**

**}**

**public static void converter() {**

**JFrame f = new JFrame("Currency Converter");**

**f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**JLabel l1, l2, l3, l4, l5, l6;**

**JTextField t1, t2, t3, t4, t5, t6;**

**JButton b1, b2, b3, b4, b5, b6;**

**l1 = new JLabel("Rupees:");**

**l1.setBounds(20, 40, 60, 30);**

**l2 = new JLabel("Dollars:");**

**l2.setBounds(170, 40, 60, 30);**

**l3 = new JLabel("Euros:");**

**l3.setBounds(320, 40, 60, 30);**

**l4 = new JLabel("Dirhams:");**

**l4.setBounds(20, 80, 60, 30);**

**l5 = new JLabel("Pounds:");**

**l5.setBounds(170, 80, 60, 30);**

**l6 = new JLabel("Takas:");**

**l6.setBounds(320, 80, 60, 30);**

**t1 = new JTextField("0");**

**t1.setBounds(80, 40, 50, 30);**

**t2 = new JTextField("0");**

**t2.setBounds(240, 40, 50, 30);**

**t3 = new JTextField("0");**

**t3.setBounds(380, 40, 50, 30);**

**t4 = new JTextField("0");**

**t4.setBounds(80, 80, 50, 30);**

**t5 = new JTextField("0");**

**t5.setBounds(240, 80, 50, 30);**

**t6 = new JTextField("0");**

**t6.setBounds(380, 80, 50, 30);**

**b1 = new JButton("INR");**

**b1.setBounds(50, 120, 60, 15);**

**b2 = new JButton("Dollar");**

**b2.setBounds(190, 120, 60, 15);**

**b3 = new JButton("Euro");**

**b3.setBounds(330, 120, 60, 15);**

**b4 = new JButton("Dirham");**

**b4.setBounds(50, 160, 60, 15);**

**b5 = new JButton("Pound");**

**b5.setBounds(190, 160, 60, 15);**

**b6 = new JButton("Taka");**

**b6.setBounds(330, 160, 60, 15);**

**b2.addActionListener(new ActionListener() {**

**public void actionPerformed(ActionEvent e) {**

**double dollars = Double.parseDouble(t2.getText());**

**double rupees = dollars \* 83.25;**

**t1.setText(String.valueOf(rupees));**

**// Store the conversion in the database**

**storeConversion("Dollars", "Rupees", dollars, rupees);**

**}**

**});**

**// Add more action listeners for other conversions as needed.**

**f.add(l1);**

**f.add(t1);**

**f.add(l2);**

**f.add(t2);**

**f.add(l3);**

**f.add(t3);**

**f.add(l4);**

**f.add(t4);**

**f.add(l5);**

**f.add(t5);**

**f.add(l6);**

**f.add(t6);**

**f.add(b1);**

**f.add(b2);**

**f.add(b3);**

**f.add(b4);**

**f.add(b5);**

**f.add(b6);**

**f.setLayout(null);**

**f.setSize(500, 250);**

**f.setVisible(true);**

**}**

**public static void storeConversion(String sourceCurrency, String targetCurrency, double amount, double convertedAmount) {**

**// Database connection parameters**

**String url = "jdbc:mysql://localhost:3306/currency\_converter";**

**String user = "your\_username";**

**String password = "your\_password";**

**try {**

**// Establish a connection to the database**

**Connection connection = DriverManager.getConnection(url, user, password);**

**// SQL query to insert conversion data**

**String sql = "INSERT INTO conversions (source\_currency, target\_currency, amount, converted\_amount) VALUES (?, ?, ?, ?";**

**// Create a PreparedStatement**

**PreparedStatement preparedStatement = connection.prepareStatement(sql);**

**// Set values for the parameters**

**preparedStatement.setString(1, sourceCurrency);**

**preparedStatement.setString(2, targetCurrency);**

**preparedStatement.setDouble(3, amount);**

**preparedStatement.setDouble(4, convertedAmount);**

**// Execute the query**

**preparedStatement.executeUpdate();**

**// Close the connection**

**connection.close();**

**} catch (SQLException e) {**

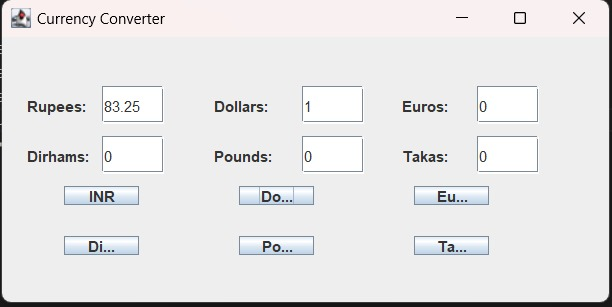
**e.printStackTrace();**

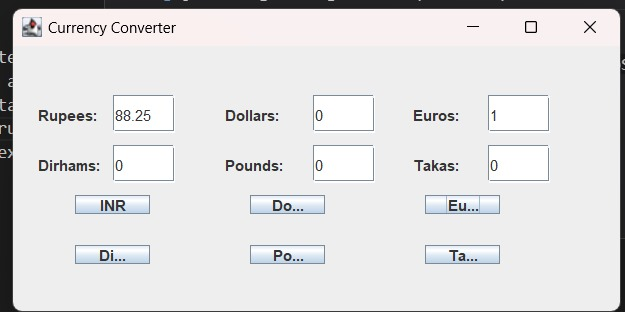
**}**

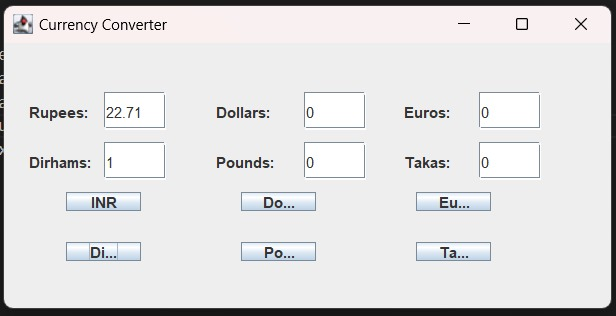
**}**

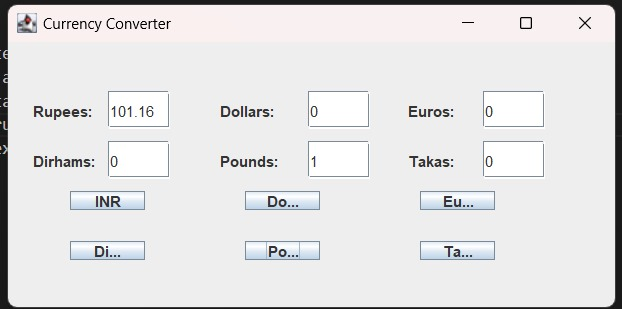
**}**

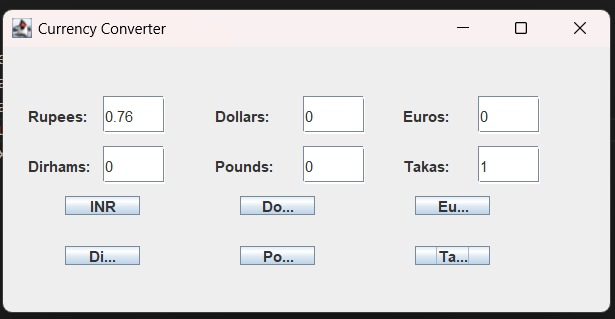
1. **RESULTS AND DISCUSSION**











1. **CONCLUSION**

The Java Swing currency converter code offers a practical introduction to GUI development in Java. It provides a simple yet effective tool for currency conversion, allowing users to interact with the application seamlessly. The code showcases the connection between the graphical user interface and the underlying logic for accurate currency conversions.

However, it's crucial to recognize that the code is a basic example with fixed conversion rates and limited error handling. Real-world currency converters would require handling dynamically changing exchange rates, more robust input validation, and potential integration with external data sources. Despite its simplicity, this code serves as a valuable resource for those learning Java Swing and GUI programming, laying the groundwork for creating more complex and feature-rich applications.

**REFERENCES**

W3Schools:HTML Tutorial: https://www.w3schools.com/html/

Wikipedia

Geeks for geeks