**DE LA SALLE UNIVERSITY - MANILA**



**<**The 'B'ank of 'C'omputer 'S'cience's Currency exchange program**>**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A Term Project

Presented to Mr. Ramon Stephen Ruiz

In Partial Fulfillment of the

Requirements for the Course PROGRAMMING LOGIC AND DESIGN LABORATORY (LBYCPA1)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

by

Abarro, Christian Joshua R.

Azul, Sophia Marquel R.

Bao, Xinlong

EQ1

[I. Introduction 3](#_heading=)

[A. Background of the Study 3](#_heading=)

[B. Problem Statement 3](#_heading=)

[C. Objectives 3](#_heading=)

[C.1 General Objective 3](#_heading=)

[D. Significance of the Project 5](#_heading=)

[II. Review of Related Literature 6](#_heading=)

[III. Methodology 6](#_heading=)

[A. Conceptual Framework – IPO Chart (Input-Process-Output-Chart) 6](#_heading=)

[B. Hierarchy Chart 7](#_heading=)

[C. Flowchart 8](#_heading=)

[D. Pseudocode 8](#_heading=)

[IV. Results 10](#_heading=)

[V. Discussion of Results 11](#_heading=)

[VI. Analysis, Conclusion and Future Directives 11](#_heading=)

[References 11](#_heading=)

[Appendices 12](#_heading=)

[A. User’s Manual 12](#_heading=)

[B. Source Code 14](#_heading=)

C[. Source Code 14](#_heading=)

# Introduction

The currency conversion tool offered by the B'ank of C'omputer S'cience is a computer-based program that provides users with an easy way to convert one kind of currency into another. It is a very helpful tool for those who regularly travel abroad or for companies who do business internationally. Users of the program will have access to the most recent currency conversion rates from across the globe, making it simpler for them to decide how to swap their money. It provides a broad selection of currencies, including well-known ones like US dollars, euros, pounds, Japanese yen, and Chinese yuan.

## Background of the Study

A currency exchange application is a helpful tool used for various purposes. It helps travellers spend money in other countries at ease. This is also a helpful tool for businesses who have international transactions. To add to that, the increase in international travel and trade is one of the main causes that has influenced the creation of currency exchange technology. The demand for reliable and practical currency exchange services has grown as more people and companies conduct cross-border transactions. Bank offices and currency exchange kiosks, which are common forms of currency conversion, may be troublesome, time-consuming, and costly. This prompted the creation of online platforms for money conversion, including the B'ank of C’omputer S’cience's money exchange application, which provides a quicker, more practical, and less expensive substitute.

The development of currency exchange systems has been significantly influenced by technological advances in addition to the expansion of international travel and trade. It is now easy to access currency exchange services from anywhere in the globe at any time thanks to the development of the internet and mobile technologies. As a result, sophisticated internet platforms that offer real-time exchange rates, immediate transactions, and secure payment processing have been created. The creation of programs like the Currency exchange program from the B’ank of C’omputer S’cience has benefited from the study that has been done in the field of currency exchange.

Studies have shown that online platforms can provide significant benefits in terms of convenience, speed, and cost-effectiveness when compared to traditional currency exchange methods, which can be inefficient and overpriced. The design and functionality of currency conversion systems have been shaped by this study, ensuring that consumers' demands are effectively and efficiently met.

## Problem Statement

Traditional currency conversion methods, which may be difficult, time-consuming, and expensive, are a problem for many people. A more effective and affordable method of exchanging currencies is becoming increasingly necessary, especially for people who travel regularly or do cross-border business. This issue is addressed by the B'ank of C'omputer S'cience's currency exchange program, which offers a practical, user-friendly, and secure online platform for currency conversion.

## Objectives

## C.1 General Objective

The general objective of the Currency Exchange Program is to automate and streamline the process of currency exchange by providing accurate and up-to-date exchange rates for various currencies. The program should be able to efficiently handle 10 different currencies and provide buy-in and sell-out functions with appropriate rates.

**C.2** **Specific Objectives**

1. Develop a user-friendly program that accepts three input parameters: the amount of currency to be exchanged, the currency code for the input currency, and the currency code for the output currency.
2. Implement a function that retrieves up-to-date exchange rates from reliable sources on the internet, ensuring that the program always uses current data for every transaction.
3. Incorporate buy-in and sell-out functions with appropriate rates, taking into account the buy-in rate for purchasing the input currency and the sell-out rate for selling the output currency.
4. Provide accurate and real-time calculations of the exchanged amount based on the input parameters and the current exchange rates, ensuring that customers receive the correct amount of currency after the exchange.
5. Implement error handling mechanisms to handle cases such as invalid input parameters, unavailable exchange rates, and unexpected errors, providing informative feedback to users and ensuring the reliability and stability of the program.
6. Incorporate security measures to protect customer data and transactions, including encryption of sensitive information and secure communication protocols.
7. Test the program thoroughly to ensure its accuracy, reliability, and performance under different scenarios and load conditions.
8. Provide documentation and support for users to understand and effectively use the Currency Exchange Program, including user guides, help manuals, and customer support channels.
9. Complete the development of the Currency Exchange Program within the specified deadline, ensuring timely delivery of the project to the BCS.
10. Continuously monitor and update the program to ensure that it remains up-to-date with the latest exchange rates and reliable sources of data, and to address any issues or improvements identified during its usage.

## Significance of the Project

1. Automation: The program automates the currency exchange process, eliminating the need for manual calculations and reducing human errors. This can streamline the exchange process, making it more efficient and saving time for both the customers and the BCS.
2. Multiple Currencies: The program aims to include 15-20 different currencies, providing a wide range of options for customers to exchange their currencies. This can cater to a diverse customer base and enhance the usability and versatility of the program.
3. Buy-in and Sell-out Functions: The program incorporates buy-in and sell-out functions with appropriate rates, allowing customers to accurately calculate the exchanged amount based on the current exchange rates. This ensures transparency and fairness in the exchange process, building trust with customers.
4. Up-to-Date Data: The goal of using up-to-date data for every transaction by automatically researching data on the internet can provide accurate and reliable exchange rates. This can help customers get the most current rates, ensuring that they receive a fair exchange value.
5. Deadline Adherence: The project's completion before the deadline is crucial to meet the contractual obligations with BCS. Timely delivery of the project showcases professionalism and reliability, enhancing the reputation of the development team and the BCS.
6. User-friendly Interface: Developing a user-friendly program can make it accessible to a wider audience, including customers with varying technical skills. This can enhance customer satisfaction and adoption of the program, leading to increased usage and potential business growth for BCS.
7. Security: Incorporating security measures to protect customer data and transactions is crucial to ensure the confidentiality and integrity of sensitive information. This can help build trust and confidence among customers in using the program for currency exchange transactions.

In summary, the Currency Exchange Program holds significant significance in automating and simplifying the currency exchange process, providing a wide range of currencies, incorporating up-to-date data, ensuring transparency and fairness, adhering to deadlines, and prioritizing security measures.

# Review of Related Literature

**Currency Converter using Arduino microcontroller**

An Arduino microcontroller, a Wi-Fi module, and a currency API are used to create a real-time currency converter in this article. The necessity of real-time currency conversion in cross-border commerce and travel is discussed in the paper, along with the drawbacks of conventional methods of currency exchange. The methodology portion of the article describes how the real-time currency converter was created and goes through the project's technical elements. The system's efficiency in converting currencies in real-time is shown in the results section, which also outlines the project's constraints and potential future directions. Real-time money exchange is emphasized in the article as being crucial to the Bank of Computer Science's agenda.

**Foreign Currency Exchange Services: Market Research Report**

The article offers a thorough study of the world foreign exchange market, taking into account market trends, expansion potential, and difficulties. The article's methodology section includes the study's findings and outlines the data collecting and analysis techniques utilized to gain market insights. The results section lists the major companies and market sectors along with statistical information on the size and expansion of the foreign exchange industry. The article's conclusion offers information on the market's potential futures, including the growing use of online and mobile currency exchange services, the demand for personalized services and real-time exchange rates, and the effects of regulatory and economic factors.

# Methodology

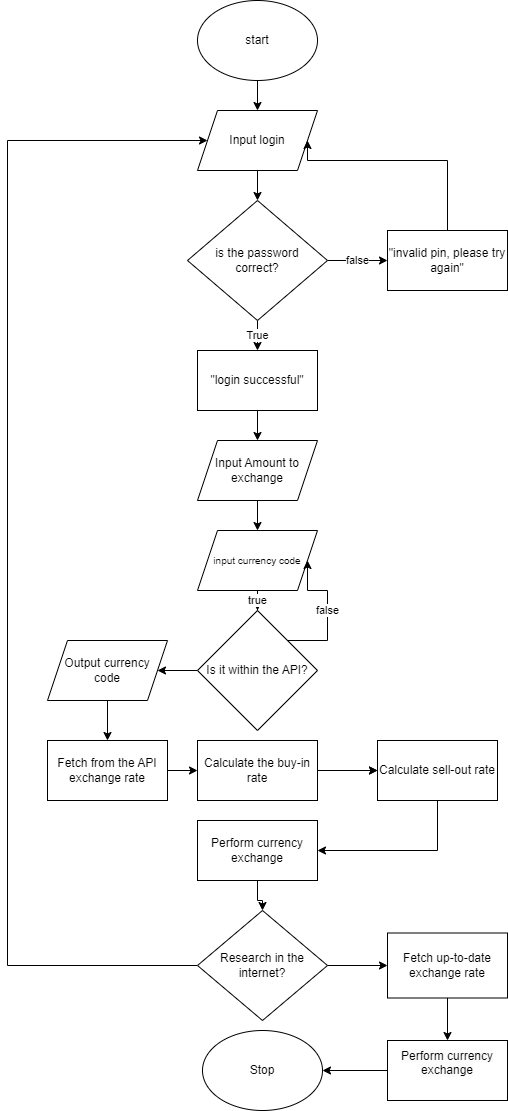
The development of the currency exchange program will follow a systematic approach that involves incorporating three input parameters: the amount of currency to be exchanged from the customer, the currency code for the input currency, and the currency code for the output currency. The program will implement a buy-in function with a buy-in rate and a sell-out function with a sell-out rate, enabling customers to exchange currencies efficiently. The goal is to include support for 15-20 different currencies and automate the slow currency exchange process. If possible, the program will automatically research data from the internet to ensure up-to-date exchange rate information for every transaction. The development methodology will emphasize efficiency, accuracy, and compliance with regulatory requirements, with a focus on meeting the project deadline and delivering a user-friendly program.

## Conceptual Framework – IPO Chart (Input-Process-Output-Chart)

| Input | Process | Output |
| --- | --- | --- |
| * Amount of currency to be exchanged from the customer * Currency code for the input currency * Currency code for the output currency | * Buy-in function with buy-in rate * Sell-out function with sell-out rate * Automatic research and retrieval of up-to-date exchange rate data from the internet (if possible and within deadline) | * Exchanged amount of currency in the output currency |

## Hierarchy Chart

## Flowchart



## Pseudocode

Start

Function get\_exchange\_rate(input\_currency, output\_currency)

Get current exchange rate from internet for input\_currency to output\_currency

Return exchange rate

Function calculate\_buy\_in(amount, input\_currency, output\_currency)

exchange\_rate = get\_exchange\_rate(input\_currency, output\_currency)

buy\_in\_rate = exchange\_rate + (exchange\_rate \* 0.05) // Add 5% to exchange rate

buy\_in\_amount = amount / buy\_in\_rate

Return buy\_in\_amount

Function calculate\_sell\_out(amount, input\_currency, output\_currency)

exchange\_rate = get\_exchange\_rate(input\_currency, output\_currency)

sell\_out\_rate = exchange\_rate - (exchange\_rate \* 0.05) // Subtract 5% from exchange rate

sell\_out\_amount = amount \* sell\_out\_rate

Return sell\_out\_amount

Function main()

Display "Welcome to Currency Exchange Program!"

Display "Please enter the amount of currency to be exchanged:"

Read amount

Display "Please enter the currency code for the input currency:"

Read input\_currency

Display "Please enter the currency code for the output currency:"

Read output\_currency

buy\_in\_amount = calculate\_buy\_in(amount, input\_currency, output\_currency)

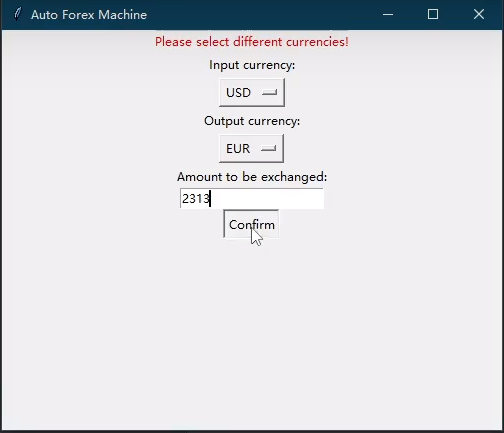
sell\_out\_amount = calculate\_sell\_out(amount, input\_currency, output\_currency)

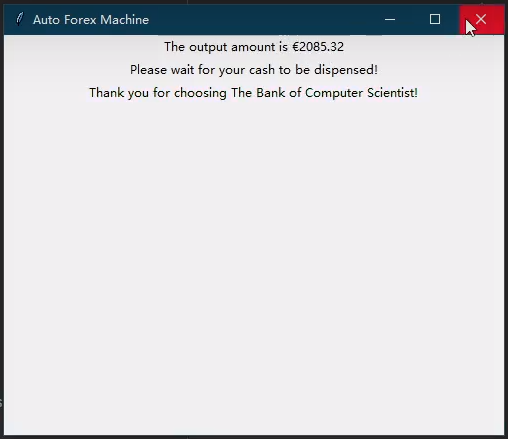
Display "Buy-in Amount: " + buy\_in\_amount + " " + output\_currency

Display "Sell-out Amount: " + sell\_out\_amount + " " + output\_currency

End

# Results





# Discussion of Results

Developing a currency exchange program with automation, up-to-date data, and support for multiple currencies can result in an efficient and competitive currency exchange process. Ensuring regulatory compliance, and providing a user-friendly interface are important factors to consider in the development process. Proper planning, development, and testing can help ensure successful implementation and satisfaction of the project.

# Analysis, Conclusion and Future Directives

In conclusion, the money exchange program offered by the B'ank of C'omputer S'cience is a useful tool. It offers excellent features including convenience, security, and use. However, if other currencies are made accessible, this might be further enhanced. It is recommended to additionally incorporate less commonly recognized currencies in order to draw in a more diverse audience.

# References

Bwisa, H. (2018). *The basics of writing a statement of the problem for your research proposal*. Retrieved from https://www.editage.com/insights/the-basics-of-writing-a-statement-of-the-problem-for-your-research-proposal

Sachdev, R. (2018). *How to write the background of your study*. Retrieved from editage.com: https://www.editage.com/insights/how-to-write-the-background-of-your-study?&placementsearch&placementblocktrendingsearch

Lane, P., & Shambaugh, J. (2007). *Financial Exchange Rates and International Currency*

*Exposures* https://www.imf.org/external/np/res/seminars/2007/arc/pdf/pl.pdf

Scott, G. (n.d.). *Currency Exchange Definition*. Investopedia. <https://www.investopedia.com/terms/c/currency-exchange.asp>

*Benefits of Currency converter Apps*. (n.d.). Www.linkedin.com. Retrieved April 14, 2023, from <https://www.linkedin.com/pulse/benefits-currency-converter-apps-ofonimeh-j-johnson>

*The Overlooked Importance Of Currency Conversion In Expense Software - ExpensePoint*. (n.d.). Retrieved April 14, 2023, from <https://www.expensepoint.com/overlooked-importance-currency-conversion-expense-software/>

*What is a Currency Exchange?* (n.d.). Robinhood. Retrieved April 14, 2023, from <https://learn.robinhood.com/articles/2q4o0SZxjeXyihnQ9E417q/what-is-a-currency-exchange/>

*A Genetic Programming Approach for EUR/USD - ProQuest*. (n.d.). Www.proquest.com. Retrieved April 14, 2023, from <https://www.proquest.com/docview/1449254996?accountid=190474&parentSessionId=9thiHAn6QE6EXpdXTy1Oo%2B96wSFYFGPNJGhRgNNQpMk%3D&pq-origsite=primo&forcedol=true>

Durgade, B., Kadappa, B., Patil, S., & Malapure, N. (n.d.). *REAL TIME CURRENCY CONVERTER*. Retrieved April 14, 2023, from http://ijariie.com/AdminUploadPdf/Real\_Time\_Currency\_Converter\_ijariie13241.pdf

# Appendices

## User’s Manual

Step 1: Accept User Input

* Prompt the user to enter the amount of currency to be exchanged.
* Prompt the user to enter the currency code for the input currency (e.g., USD, EUR, GBP, etc.).
* Prompt the user to enter the currency code for the output currency (e.g., CAD, AUD, JPY, etc.).
* Validate the user input to ensure it's in the correct format and within acceptable ranges.

Step 2: Fetch Exchange Rates

* If the program is set to automatically fetch up-to-date exchange rates, use an API to retrieve the latest exchange rates for the specified input and output currencies.
* If the program uses a pre-populated database of exchange rates, retrieve the appropriate exchange rates from the database based on the input and output currency codes.

Step 3: Calculate Buy-in and Sell-out Rates

* Using the fetched or retrieved exchange rates, calculate the buy-in rate and sell-out rate.
* The buy-in rate can be calculated as the exchange rate for the output currency divided by the exchange rate for the input currency.
* The sell-out rate can be calculated as the inverse of the buy-in rate.

Step 4: Display Rates and Ask for Confirmation

* Display the calculated buy-in and sell-out rates to the user along with other relevant details such as transaction fees, if any.
* Ask the user for confirmation to proceed with the exchange, and prompt for any additional information if required, such as user authentication or payment details.

Step 5: Perform Currency Exchange

* If the user confirms the exchange, apply the appropriate buy-in or sell-out rate to the input amount to calculate the exchanged amount.
* Display the exchanged amount to the user as the final transaction result.

Step 6: Update Exchange Rates (if required)

* If the program is designed to automatically fetch up-to-date exchange rates, update the rates in the database for future transactions.

Step 7: Repeat or Exit

* Provide the option for the user to perform another transaction or exit the program.
* Handle any errors or exceptions that may occur during the transaction process, and provide appropriate error messages to the user for troubleshooting.

Step 8: Automate Slow Currency Exchange Process (if required)

* If the program needs to automate the slow currency exchange process, implement features such as scheduling automatic exchange transactions at specific intervals or automatically initiating exchange transactions based on certain triggers, such as reaching a certain exchange rate threshold.

## Source Code

## import tkinter as tk

## from forex\_python.converter import CurrencyRates

## from forex\_python.converter import CurrencyCodes

## c = CurrencyRates()

## s = CurrencyCodes()

## # Create a list of currency options

## currency\_options = ['USD', 'EUR', 'JPY', 'GBP', 'AUD', 'CAD', 'CHF', 'CNY']

## class AFM:

## def \_\_init\_\_(self, master):

## self.master = master

## self.master.title("Auto Forex Machine")

## self.master.geometry("500x400")

## self.label1 = tk.Label(self.master, text="Welcome to BCS", font=("Arial", 20))

## self.label1.pack(pady=20)

## self.label2 = tk.Label(self.master, text="Enter your PIN:", font=("Arial", 12))

## self.label2.pack()

## self.entry = tk.Entry(self.master, show="\*")

## self.entry.pack(pady=10)

## self.button = tk.Button(self.master, text="Enter", command=self.login, font=("Arial", 12))

## self.button.pack(pady=10)

## self.label4 = tk.Label(self.master, text="")

## self.label4.pack()

## def animate(self, label, colors):

## current\_color = label.cget("fg")

## if current\_color in colors:

## next\_index = (colors.index(current\_color) + 1) % len(colors)

## next\_color = colors[next\_index]

## else:

## next\_color = colors[0]

## label.config(fg=next\_color)

## label.after(500, self.animate, label, colors)

## def login(self):

## pin = self.entry.get()

## if pin == "1234":

## self.entry.pack\_forget()

## self.label1.configure(text="Login successful!", fg="green")

## self.label2.configure(text="Good day computer scientist!")

## self.animate(self.label2, ["black", "blue", "red", "green"])

## self.label3 = tk.Label(self.master, text="What would you like to do today?")

## self.label3.pack()

## self.button.pack\_forget()

## self.buy\_button = tk.Button(self.master, text="Buy", command=self.show\_buy\_options)

## self.buy\_button.pack()

## self.sell\_button = tk.Button(self.master, text="Sell", command=self.show\_sell\_options)

## self.sell\_button.pack()

## self.logout\_button = tk.Button(self.master, text="Logout", command=self.logout)

## self.logout\_button.pack()

## else:

## self.label1.configure(text="Invalid PIN, please try again.", fg="red")

## def show\_buy\_options(self):

## self.entry.pack\_forget()

## self.label2.pack\_forget()

## self.label3.pack\_forget()

## self.label1.configure(text="Cash or Remittance?", fg="black")

## self.buy\_button.pack\_forget()

## self.sell\_button.pack\_forget()

## self.logout\_button.pack\_forget()

## self.cash\_button = tk.Button(self.master, text="Cash", command=self.buy\_cash)

## self.cash\_button.pack()

## self.remit\_button = tk.Button(self.master, text="Remittance", command=self.buy\_remit)

## self.remit\_button.pack()

## def show\_sell\_options(self):

## self.entry.pack\_forget()

## self.label2.pack\_forget()

## self.label3.pack\_forget()

## self.label1.configure(text="Cash or Remittance?")

## self.buy\_button.pack\_forget()

## self.sell\_button.pack\_forget()

## self.logout\_button.pack\_forget()

## self.cash\_button = tk.Button(self.master, text="Cash", command=self.sell\_cash)

## self.cash\_button.pack()

## self.remit\_button = tk.Button(self.master, text="Remittance", command=self.sell\_remit)

## self.remit\_button.pack()

## def buy\_cash(self):

## self.label1.pack\_forget()

## self.cash\_button.pack\_forget()

## self.remit\_button.pack\_forget()

## self.input\_label = tk.Label(self.master, text="Input currency:")

## self.input\_label.pack()

## # Create StringVars to store the selected currencies

## self.input\_currency = tk.StringVar()

## self.output\_currency = tk.StringVar()

## # Create the dropdown menus

## self.input\_currency\_dropdown = tk.OptionMenu(self.master, self.input\_currency, \*currency\_options)

## self.input\_currency\_dropdown.pack()

## self.output\_label = tk.Label(self.master, text="Output currency:")

## self.output\_label.pack()

## self.output\_currency\_dropdown = tk.OptionMenu(self.master, self.output\_currency, \*currency\_options)

## self.output\_currency\_dropdown.pack()

## self.amount\_label = tk.Label(self.master, text="Amount to be exchanged:")

## self.amount\_label.pack()

## self.amount\_entry = tk.Entry(self.master, show="")

## self.amount\_entry.pack()

## self.confirm\_button = tk.Button(self.master, text="Confirm", command=self.cash\_buy\_in)

## self.confirm\_button.pack()

## def buy\_remit(self):

## self.label1.pack\_forget()

## self.cash\_button.pack\_forget()

## self.remit\_button.pack\_forget()

## self.input\_label = tk.Label(self.master, text="Input currency:")

## self.input\_label.pack()

## # Create StringVars to store the selected currencies

## self.input\_currency = tk.StringVar()

## self.output\_currency = tk.StringVar()

## # Create the dropdown menus

## self.input\_currency\_dropdown = tk.OptionMenu(self.master, self.input\_currency, \*currency\_options)

## self.input\_currency\_dropdown.pack()

## self.output\_label = tk.Label(self.master, text="Output currency:")

## self.output\_label.pack()

## self.output\_currency\_dropdown = tk.OptionMenu(self.master, self.output\_currency, \*currency\_options)

## self.output\_currency\_dropdown.pack()

## self.amount\_label = tk.Label(self.master, text="Amount to be exchanged:")

## self.amount\_label.pack()

## self.amount\_entry = tk.Entry(self.master, show="")

## self.amount\_entry.pack()

## self.confirm\_button = tk.Button(self.master, text="Confirm", command=self.remit\_buy\_in)

## self.confirm\_button.pack()

## def cash\_buy\_in(self):

## input\_currency = self.input\_currency.get()

## output\_currency = self.output\_currency.get()

## input\_amount = float(self.amount\_entry.get())

## if input\_currency != output\_currency:

## self.input\_currency\_dropdown.pack\_forget()

## self.output\_currency\_dropdown.pack\_forget()

## self.amount\_entry.pack\_forget()

## self.confirm\_button.pack\_forget()

## rate = c.get\_rate(input\_currency, output\_currency)

## output\_amount = float(input\_amount) \* rate \* 0.98

## rounded\_output = round(output\_amount, 2)

## symbol = s.get\_symbol(output\_currency)

## self.input\_label.configure(text="The output amount is " + symbol + str(rounded\_output))

## self.output\_label.configure(text="Please wait for your cash to be dispensed!")

## self.amount\_label.configure(text="Thank you for choosing The Bank of Computer Scientist!", fg="black")

## self.animate(self.amount\_label, ["black", "blue", "red", "green"])

## if self.label4.winfo\_exists():

## self.label4.pack\_forget()

## else:

## self.label4.configure(text="Please select different currencies!",fg="red")

## def remit\_buy\_in(self):

## input\_currency = self.input\_currency.get()

## output\_currency = self.output\_currency.get()

## input\_amount = float(self.amount\_entry.get())

## if input\_currency != output\_currency:

## self.input\_currency\_dropdown.pack\_forget()

## self.output\_currency\_dropdown.pack\_forget()

## self.amount\_entry.pack\_forget()

## self.confirm\_button.pack\_forget()

## rate = c.get\_rate(input\_currency, output\_currency)

## output\_amount = float(input\_amount) \* rate \* 0.99

## rounded\_output = round(output\_amount, 2)

## symbol = s.get\_symbol(output\_currency)

## self.input\_label.configure(text="The output amount is " + symbol + str(rounded\_output))

## self.output\_label.configure(text="It has already been transferred to your "+ output\_currency+" account!")

## self.amount\_label.configure(text="Thank you for choosing The Bank of Computer Scientist!")

## self.animate(self.amount\_label, ["black", "blue", "red", "green"])

## if self.label4.winfo\_exists():

## self.label4.pack\_forget()

## else:

## pass

## else:

## self.label4 = tk.Label(self.master, text="Please select different currencies!",fg="red")

## self.label4.pack()

## def sell\_cash(self):

## self.label1.pack\_forget()

## self.cash\_button.pack\_forget()

## self.remit\_button.pack\_forget()

## self.input\_label = tk.Label(self.master, text="Input currency:")

## self.input\_label.pack()

## # Create StringVars to store the selected currencies

## self.input\_currency = tk.StringVar()

## self.output\_currency = tk.StringVar()

## # Create the dropdown menus

## self.input\_currency\_dropdown = tk.OptionMenu(self.master, self.input\_currency, \*currency\_options)

## self.input\_currency\_dropdown.pack()

## self.output\_label = tk.Label(self.master, text="Output currency:")

## self.output\_label.pack()

## self.output\_currency\_dropdown = tk.OptionMenu(self.master, self.output\_currency, \*currency\_options)

## self.output\_currency\_dropdown.pack()

## self.amount\_label = tk.Label(self.master, text="Amount to be exchanged:")

## self.amount\_label.pack()

## self.amount\_entry = tk.Entry(self.master, show="")

## self.amount\_entry.pack()

## self.confirm\_button = tk.Button(self.master, text="Confirm", command=self.cash\_sell\_out)

## self.confirm\_button.pack()

## def sell\_remit(self):

## self.label1.pack\_forget()

## self.cash\_button.pack\_forget()

## self.remit\_button.pack\_forget()

## self.input\_label = tk.Label(self.master, text="Input currency:")

## self.input\_label.pack()

## # Create StringVars to store the selected currencies

## self.input\_currency = tk.StringVar()

## self.output\_currency = tk.StringVar()

## # Create the dropdown menus

## self.input\_currency\_dropdown = tk.OptionMenu(self.master, self.input\_currency, \*currency\_options)

## self.input\_currency\_dropdown.pack()

## self.output\_label = tk.Label(self.master, text="Output currency:")

## self.output\_label.pack()

## self.output\_currency\_dropdown = tk.OptionMenu(self.master, self.output\_currency, \*currency\_options)

## self.output\_currency\_dropdown.pack()

## self.amount\_label = tk.Label(self.master, text="Amount to be exchanged:")

## self.amount\_label.pack()

## self.amount\_entry = tk.Entry(self.master, show="")

## self.amount\_entry.pack()

## self.confirm\_button = tk.Button(self.master, text="Confirm", command=self.remit\_sell\_out)

## self.confirm\_button.pack()

## def cash\_sell\_out(self):

## input\_currency = self.input\_currency.get()

## output\_currency = self.output\_currency.get()

## input\_amount = float(self.amount\_entry.get())

## if input\_currency != output\_currency:

## self.input\_currency\_dropdown.pack\_forget()

## self.output\_currency\_dropdown.pack\_forget()

## self.amount\_entry.pack\_forget()

## self.confirm\_button.pack\_forget()

## rate = c.get\_rate(input\_currency, output\_currency)

## output\_amount = float(input\_amount) \* rate \* 1.02

## rounded\_output = round(output\_amount, 2)

## symbol = s.get\_symbol(output\_currency)

## self.input\_label.configure(text="The output amount is " + symbol + str(rounded\_output))

## self.output\_label.configure(text="Please wait for your cash to be dispensed!")

## self.amount\_label.configure(text="Thank you for choosing The Bank of Computer Scientist!")

## self.animate(self.amount\_label, ["black", "blue", "red", "green"])

## if self.label4.winfo\_exists():

## self.label4.pack\_forget()

## else:

## pass

## else:

## self.label4 = tk.Label(self.master, text="Please select different currencies!",fg="red")

## self.label4.pack()

## def remit\_sell\_out(self):

## input\_currency = self.input\_currency.get()

## output\_currency = self.output\_currency.get()

## input\_amount = float(self.amount\_entry.get())

## if input\_currency != output\_currency:

## self.input\_currency\_dropdown.pack\_forget()

## self.output\_currency\_dropdown.pack\_forget()

## self.amount\_entry.pack\_forget()

## self.confirm\_button.pack\_forget()

## rate = c.get\_rate(input\_currency, output\_currency)

## output\_amount = float(input\_amount) \* rate \* 1.01

## rounded\_output = round(output\_amount, 2)

## symbol = s.get\_symbol(output\_currency)

## self.input\_label.configure(text="The output amount is " + symbol + str(rounded\_output))

## self.output\_label.configure(text="It has already been transferred to your " + output\_currency + " account!")

## self.amount\_label.configure(text="Thank you for choosing The Bank of Computer Scientist!")

## self.animate(self.amount\_label, ["black", "blue", "red", "green"])

## if self.label4.winfo\_exists():

## self.label4.pack\_forget()

## else:

## pass

## else:

## self.label4 = tk.Label(self.master, text="Please select different currencies!",fg="red")

## self.label4.pack()

## def logout(self):

## self.master.destroy()

## 

## root = tk.Tk()

## my\_gui = AFM(root)

## root.mainloop()

## 

1. **Work breakdown**

| Student Name | Tasks Assigned | Percentage of the Work Contribution |
| --- | --- | --- |
| Christian Abarro | login authentication  Objectives  Significance of the project  research the API of the code | 30% |
| Sophia Azul | Phase 1 Introduction Backgrond of the study  Problem statement  Conclusion | 30% |
| Xinlong Bao | Methodology – Pseudocode  Provided the demo video and the majority of applying and fixing the groupmates code. | 40% |

1. **Personal Data Sheet**

| Name | Picture |
| --- | --- |
| Abarroo, Christian |  |
| Azul, Sophia |  |
| Bao, Xinlong |  |