Individual Project Report

Mihir Samanyu Talamudipi

LC - 22: Team 17

ENGR 13300: Transforming Ideas to Innovation

Prof. Benjamin Manning

12/5/2024

1. Project Introduction

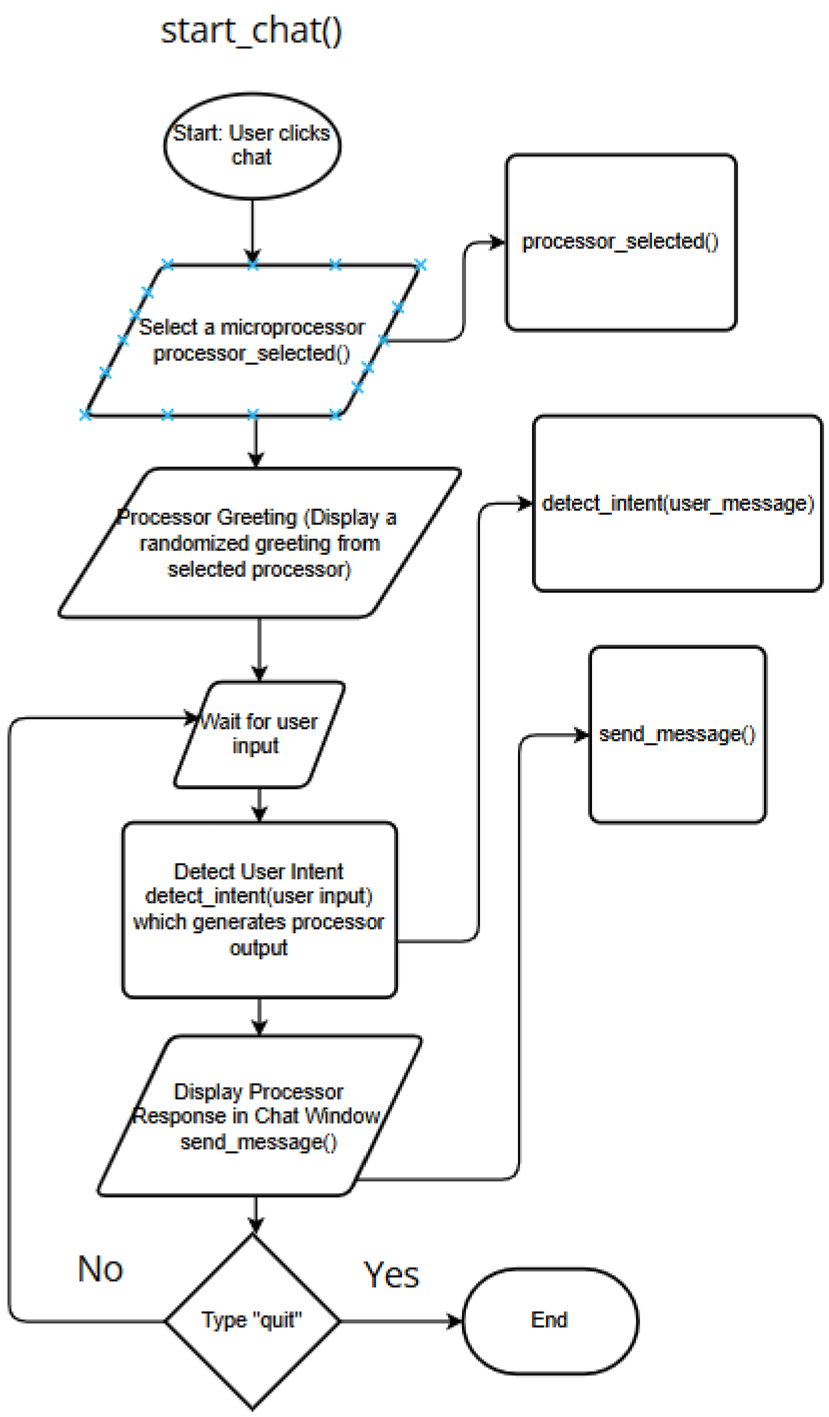
This program envisions the potential of advanced microprocessors with distinct personalities and capabilities. It features Turtle-based visual designs representing four unique microprocessors combined with an interactive chatbot to visualize life. It is meaningful for end-users as it is an attempt to mimic a fraction of what “living” microchips can do! While currently in its early stages, the program showcases a creative concept with significant potential for future expansion and sophistication.

2. Project Overview of Inputs and Outputs

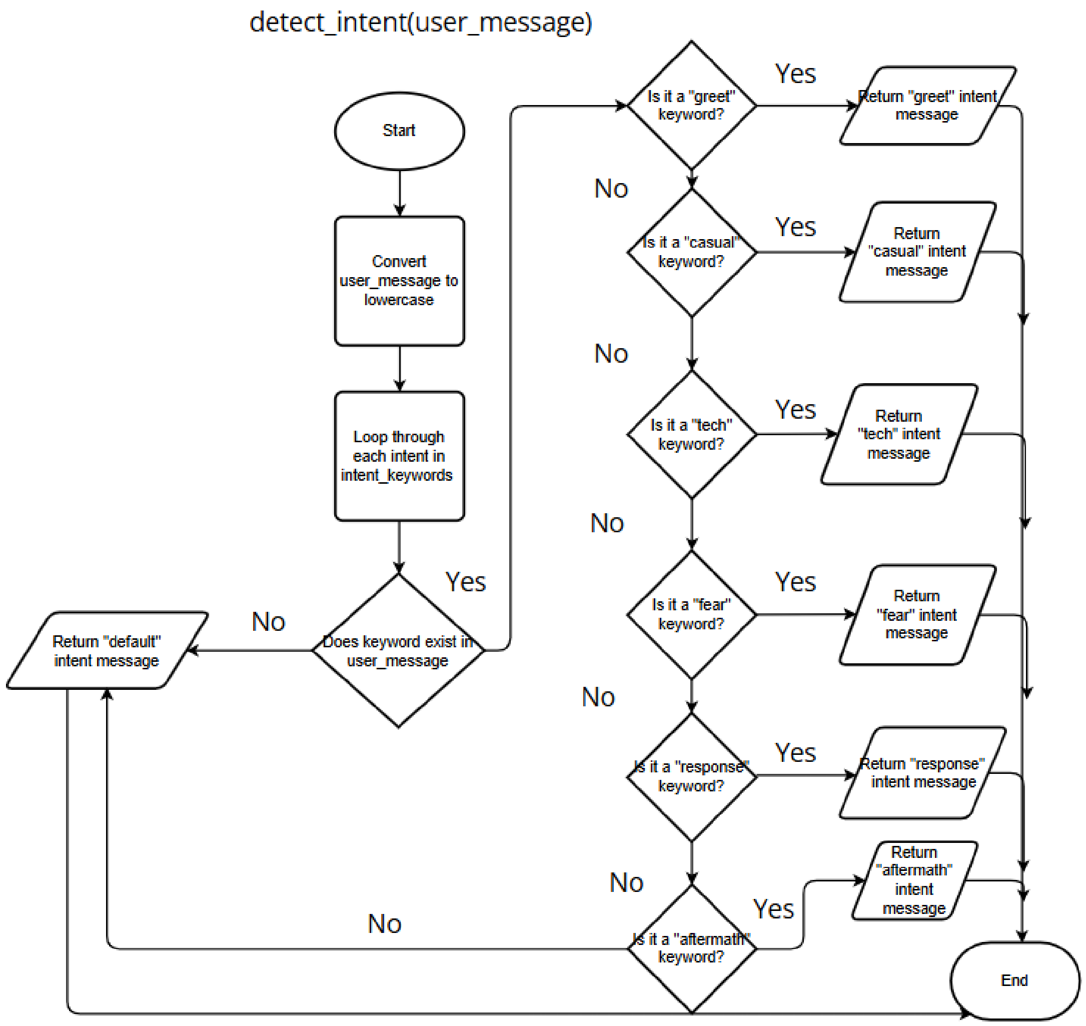
In this project, most of the user inputs are through clicking buttons. There are namely six buttons- About, Athena, Blitz, Echo, Titan and Chat. Pressing on four of these buttons apart from Chat and About – opens a Python Turtle window which designs/draws a unique microprocessor. This achieves the desired outcome of helping the user visualize unique microprocessors with distinct personalities. Clicking the About button opens an info pop-up containing information about the program vision and summary. Clicking the chat button allows the user to start chatting with any of the four microprocessors by selecting it from the drop-down menu (which can be considered as input). Once the microprocessor is selected and the start chat button is pressed the chat window is opened where the user can type in any input in the box as it is interactive and receive replies on the screen based on pre-written responses to questions (It implements a Keyword algorithm). This input helps achieve the desired outcome of the microprocessors having “consciousness” or “life” and the user experiencing the idea being mimicked and an attempt being made!

3. User-defined Functions

start_chat() – This function manages the chatbot’s functionality, including detecting user intent, generating responses, and handling the user interface. It has functions detect_intent(user_message), send_message () and processor_selected() nested inside it. It essentially is mapped to the chat button in the main function (input) and is responsible for the entire chatting interface and unifying the ability to chat with the different microprocessors (output).

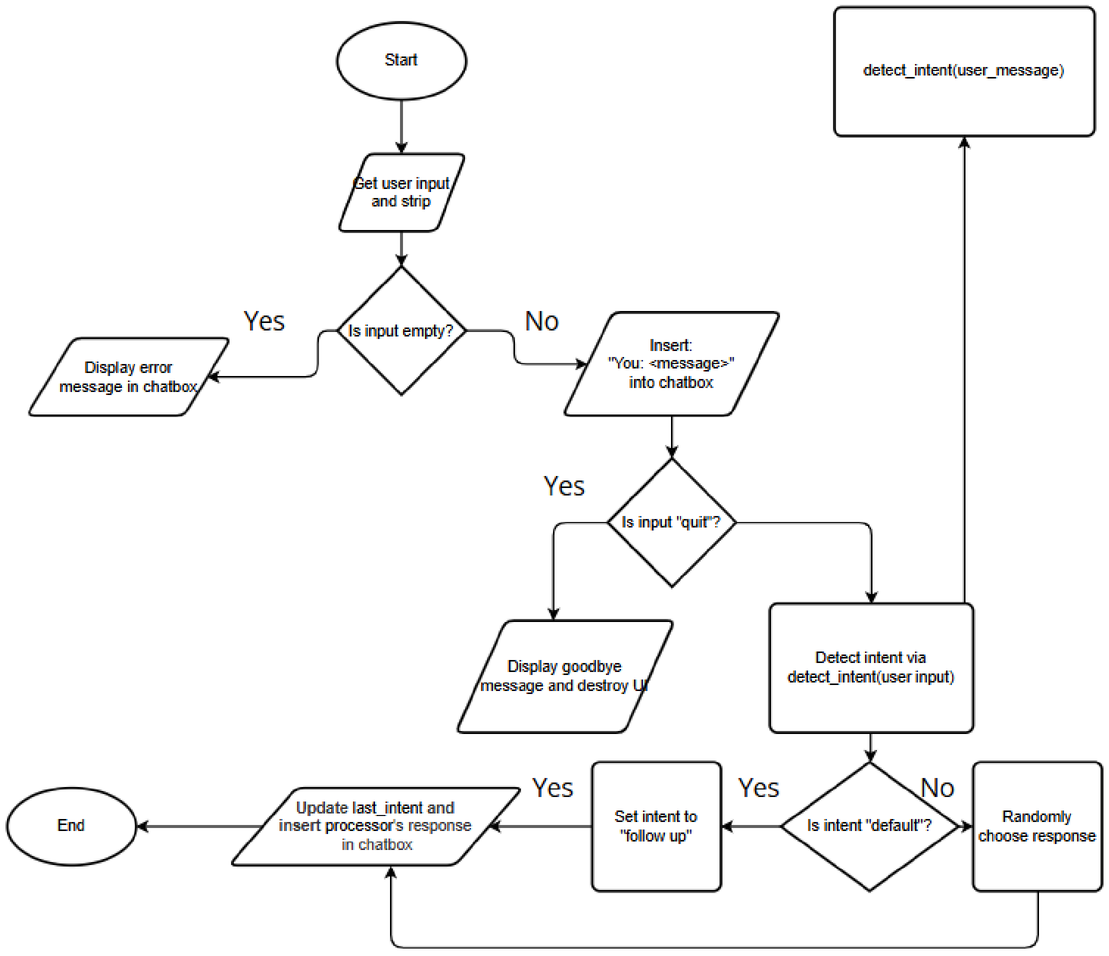

start_chat()
Start: User clicks chat
Select a microprocessor processor_selected()
processor_selected()
Processor Greeting (Display a randomized greeting from selected processor)
detect_intent(user_message)
send_message()
Wait for user input
Detect User Intent detect_intent(user input) which generates processor output
Display Processor Response in Chat Window send_message()
No
Yes
Type "quit"
End

detect_intent(user_message) – The function maps user input to predefined intents using keyword matching. Its input is the user message in the chat box and based on that it maps it to to an intent – greet, tech, casual, response, fear or aftermath that contain randomized pre-written dialogue (Uses nested dictionaries for mapping keywords to intents) which is the output. This is important as it maintains a natural flow in the conversation.


detect_intent(user_message)
Start
Convert user_message to lowercase
Loop through each intent in intent_keywords
Does keyword exist in user_message
Yes
No
Return "default" intent message
Is it a "greet" keyword?
Return "greet" intent message
Is it a "casual" keyword?
Return "casual" intent message
Is it a "tech" keyword?
Return "tech" intent message
Is it a "fear" keyword?
Return "fear" intent message
Is it a "response" keyword?
Return "response" intent message
Is it a "aftermath" keyword?
Return "aftermath" intent message
Yes
No
End

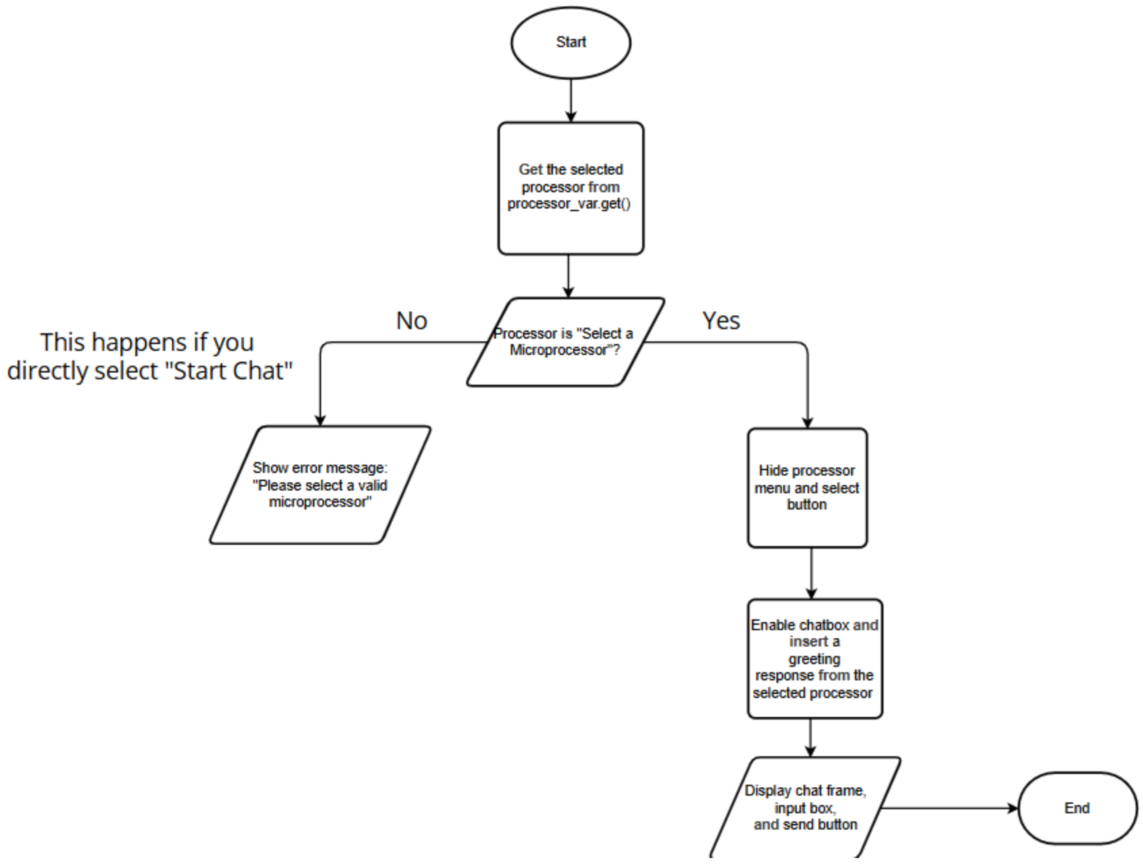
send_message() – It essentially processes user input, identifies intent using detect_intent(), and updates the chat box with an appropriate response. It handles special input commands such as typing "quit" which exits the chat or handling errors and displaying them on the screen.

send_message()


Start
Get user input and strip
Is input empty?
Yes
Display error message in chatbox
No
Insert: "You: <message>" into chatbox
Is input "quit"?
Yes
Display goodbye message and destroy UI
Detect intent via detect_intent(user input)
detect_intent(user_message)
Is intent "default"?
Yes
Set intent to "follow up"
No
Randomly choose response
Yes
Update last_intent and insert processor's response in chatbox
End

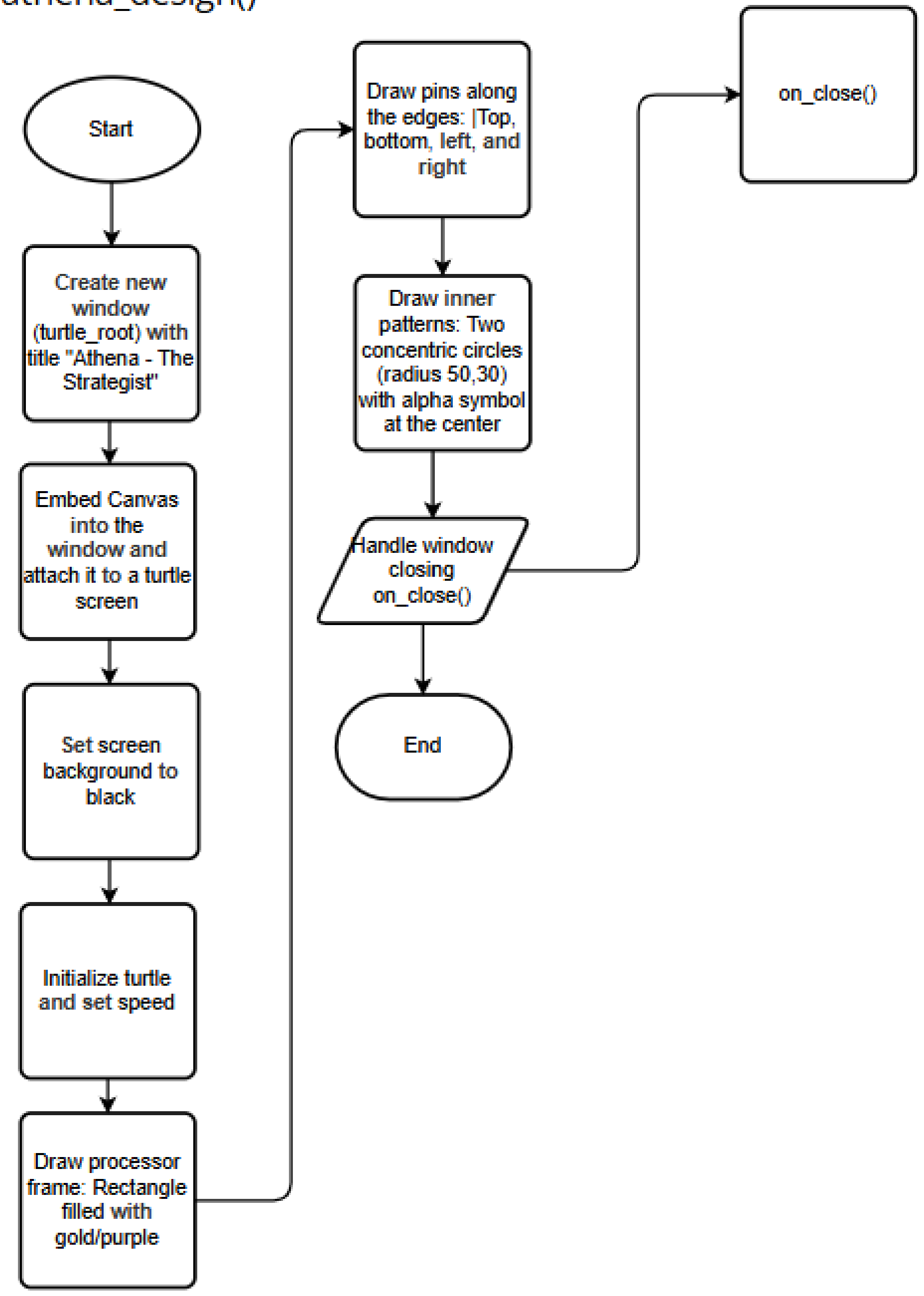
processor_selected() – This function validates user-selected microprocessor and initializes the chat interface. Its input is the user-selected microprocessor and output involves validating the choice and initializing the respective chat interface.

processor_selected()


Start
Get the selected processor from processor_var.get()
Processor is "Select a Microprocessor"?
No
Yes
This happens if you directly select "Start Chat"
Show error message: "Please select a valid microprocessor"
Hide processor menu and select button
Enable chatbox and insert a greeting response from the selected processor
Display chat frame, input box, and send button
End

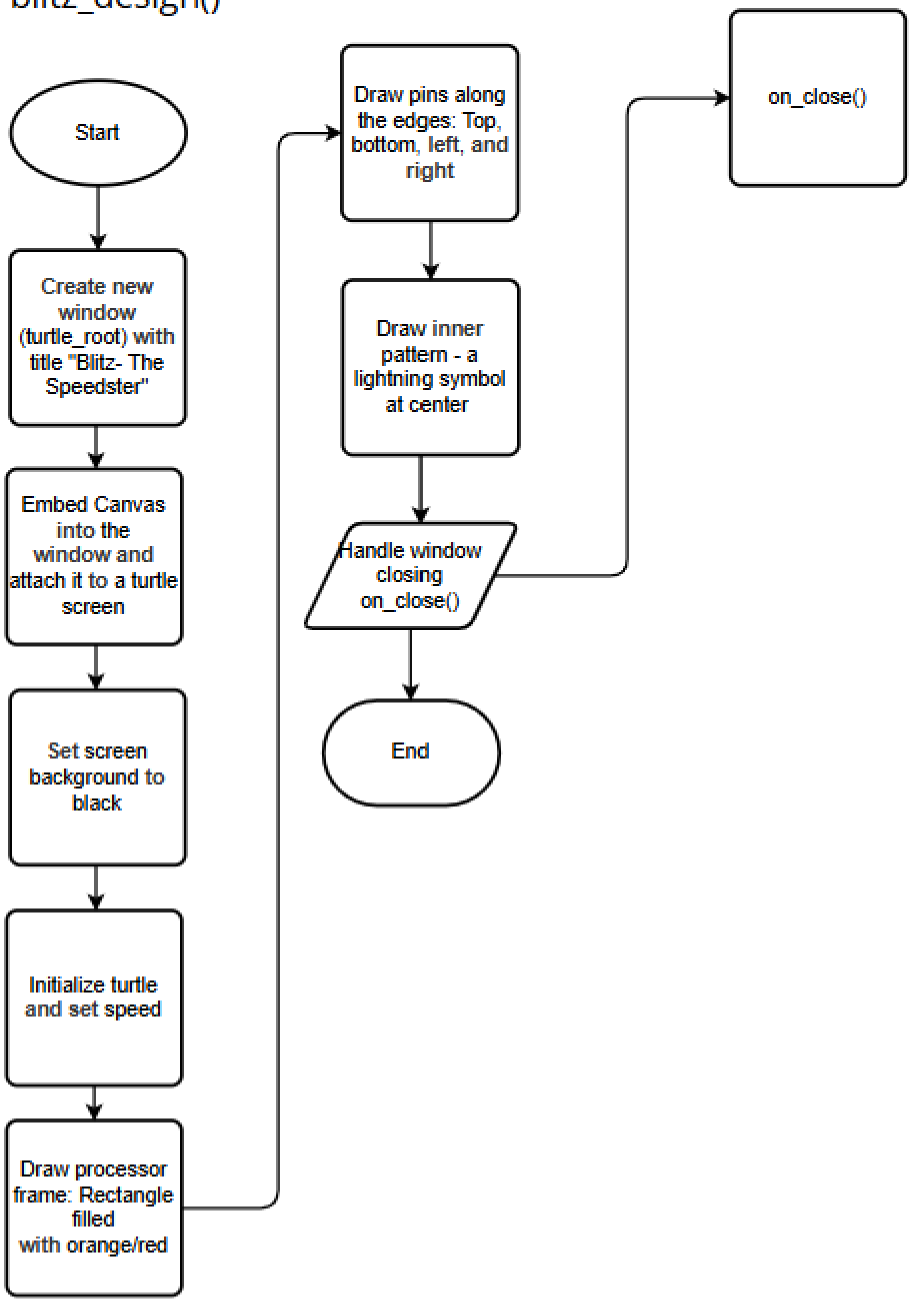
athena_design() - Creates a visual representation of Athena, the microprocessor with a personality inspired by intelligence and elegance. The input is all pre-coded as it uses Python Turtle to draw this “visual representation” and display it which is a purple and gold rectangle consisting of circular patterns, pins and an alpha symbol in the center (Output).

athena_design()


Start
Create new window (turtle_root) with title "Athena - The Strategist"
Embed Canvas into the window and attach it to a turtle screen
Set screen background to black
Initialize turtle and set speed
Draw processor frame: Rectangle filled with gold/purple
Draw pins along the edges: |Top, bottom, left, and right
Draw inner patterns: Two concentric circles (radius 50,30) with alpha symbol at the center
Handle window closing on_close()
End
on_close()

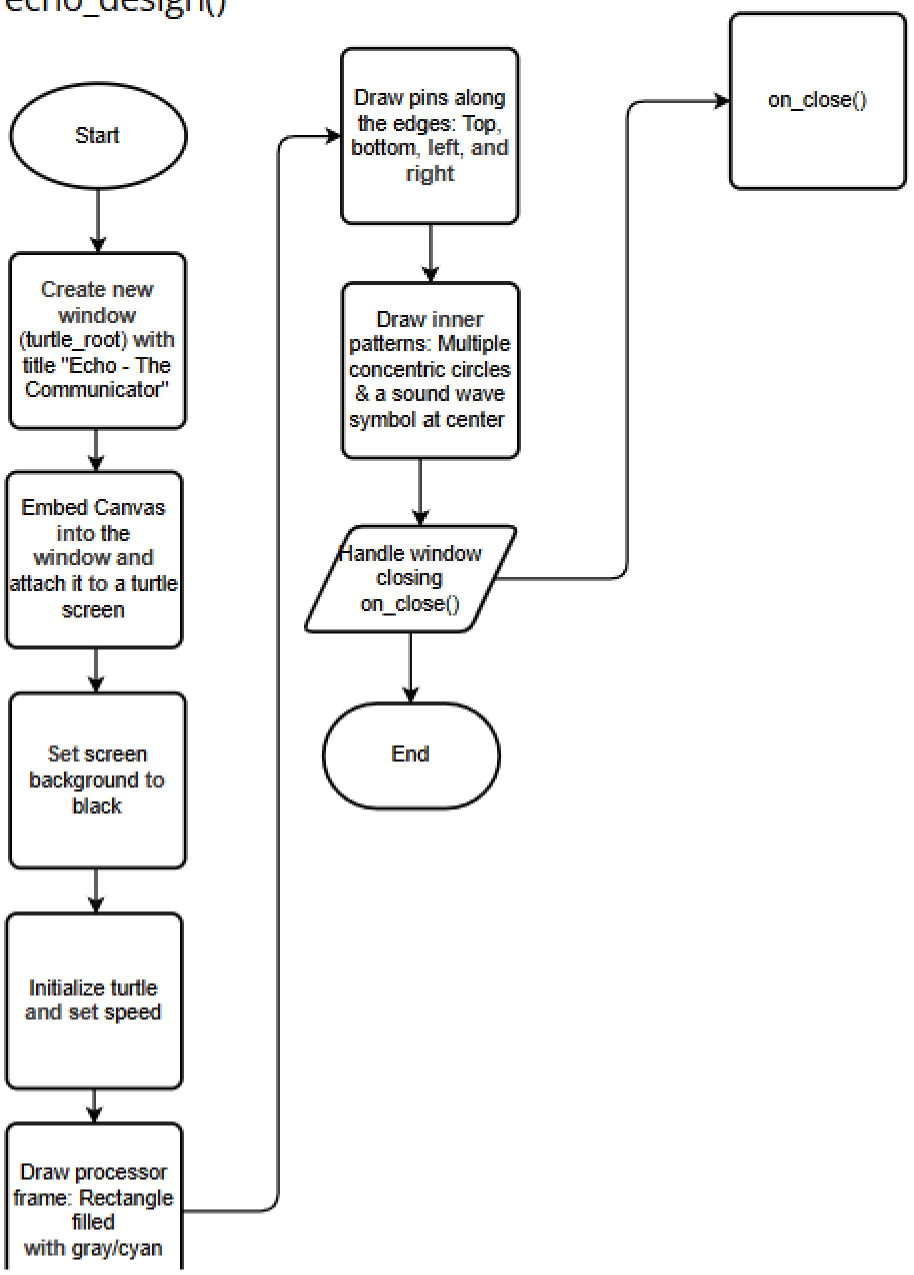
blitz_design() - Represents Blitz, a fast, high-energy microprocessor. The input is all pre-coded as it uses Python Turtle to draw this "visual representation" and display it which is a red and orange rectangle consisting of pins and an orange lightning symbol in the center (Output).

blitz_design()

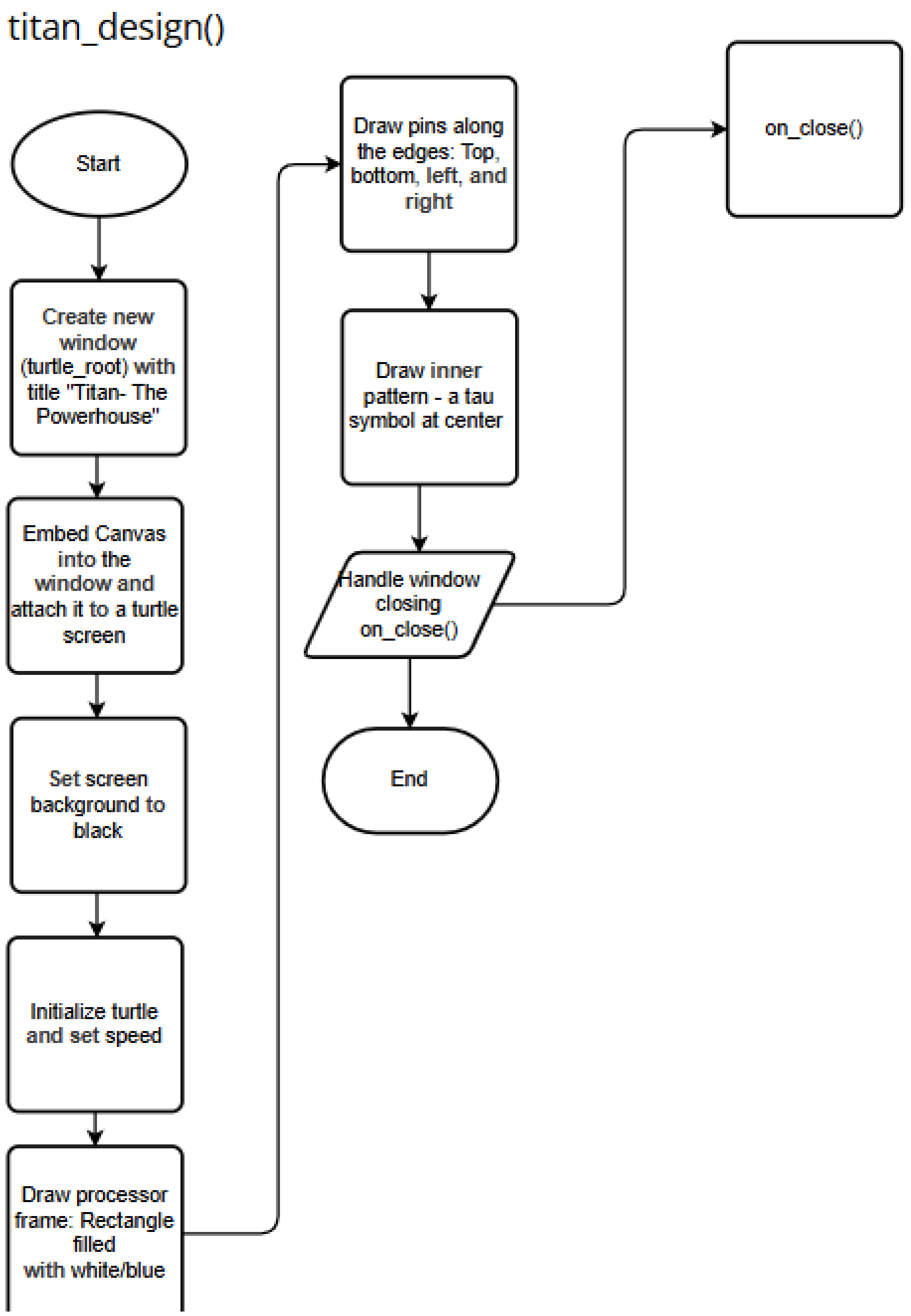

Start
Create new window (turtle_root) with title "Blitz- The Speedster"
Embed Canvas into the window and attach it to a turtle screen
Set screen background to black
Initialize turtle and set speed
Draw processor frame: Rectangle filled with orange/red
Draw pins along the edges: Top, bottom, left, and right
Draw inner pattern - a lightning symbol at center
Handle window closing on_close()
End
on_close()

echo_design() - Reflects Echo's personality as a futuristic, communicative microprocessor. The input is all pre-coded as it uses Python Turtle to draw this "visual representation" and display it which is a gray and cyan rectangle consisting of pins and a cyan wave symbol in the center (Output).

echo_design()

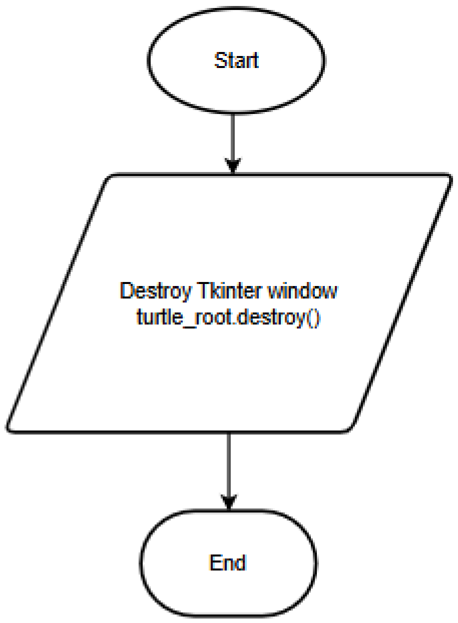

Start
Create new window (turtle_root) with title "Echo - The Communicator"
Embed Canvas into the window and attach it to a turtle screen
Set screen background to black
Initialize turtle and set speed
Draw processor frame: Rectangle filled with gray/cyan
Draw pins along the edges: Top, bottom, left, and right
Draw inner patterns: Multiple concentric circles & a sound wave symbol at center
Handle window closing on_close()
End
on_close()

titan_design() - Embodies Titan, the strong and dependable microprocessor. The input is all pre-coded as it uses Python Turtle to draw this "visual representation" and display it which is a white and blue rectangle consisting of pins and a blue tau symbol in the center (Output).

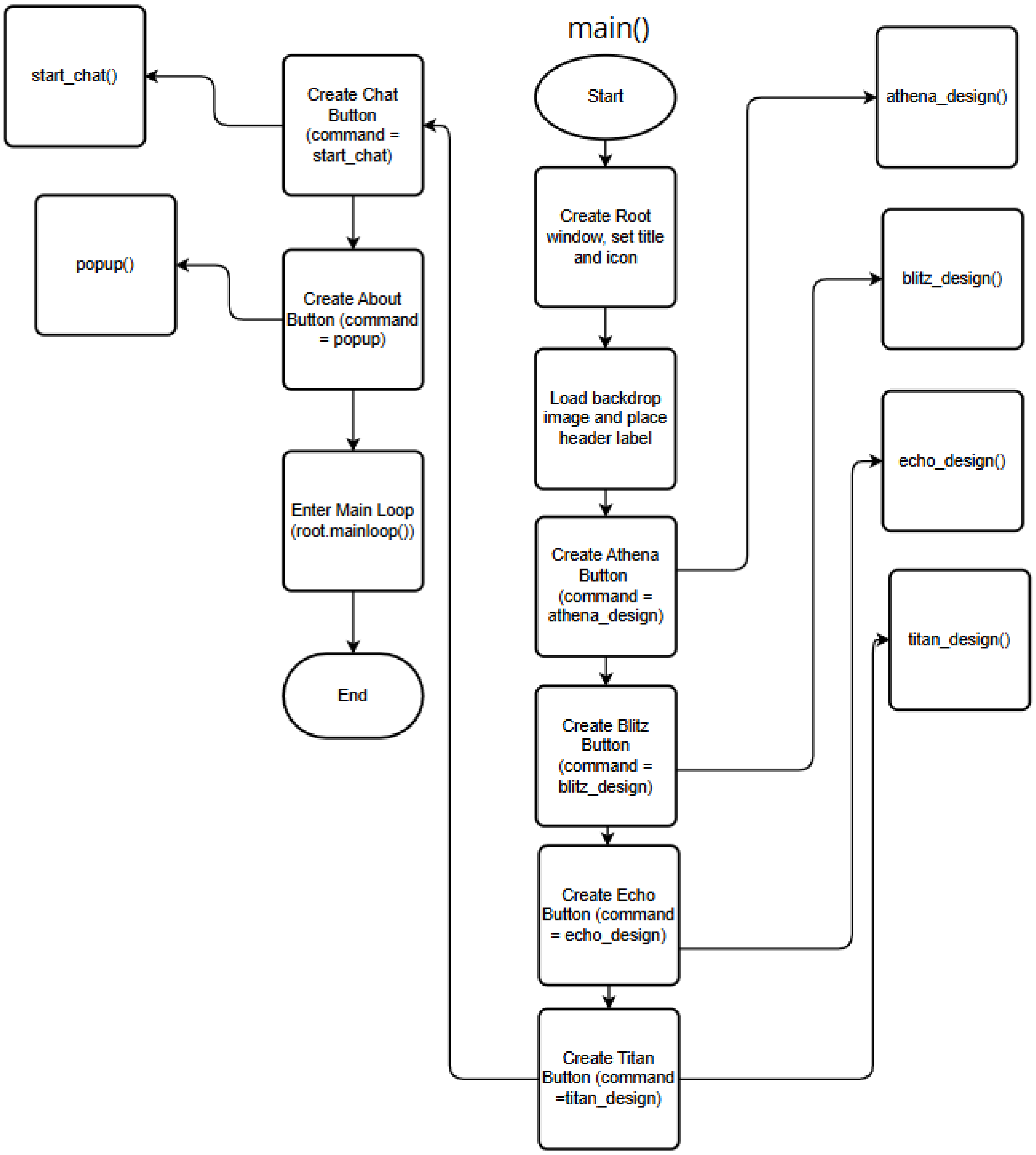

titan_design()
Start
Create new window (turtle_root) with title "Titan- The Powerhouse"
Embed Canvas into the window and attach it to a turtle screen
Set screen background to black
Initialize turtle and set speed
Draw processor frame: Rectangle filled with white/blue
Draw pins along the edges: Top, bottom, left, and right
Draw inner pattern - a tau symbol at center
Handle window closing on_close()
End
on_close()

on_close() – This function essentially destroys the turtle (tkinter) window when called. Its main role is to close the turtle design window (Output) which is why it called in all the four design functions. It was added to the code as there was a bug which was not allowing new designs to be made after closing the first design. For example, if I pressed athena_design() it would draw the microchip but if I closed the window and clicked blitz_design it would give me a blank turtle screen.

on_close()

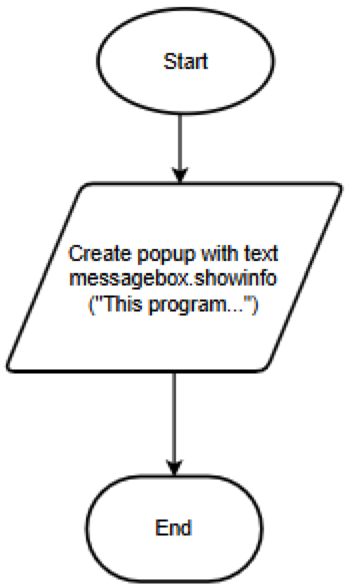


main() – The main function essentially consisted of the entire main window UI and buttons. The input involved clicking the buttons and the output included the window itself, and the buttons being mapped to the functions above which provided meaningful results! The main function is the backbone of the program bringing together all the different functions and their inputs/outputs together!


main()
Start
Create Root window, set title and icon
Load backdrop image and place header label
Create Athena Button (command = athena_design)
Create Blitz Button (command = blitz_design)
Create Echo Button (command = echo_design)
Create Titan Button (command =titan_design)
athena_design()
blitz_design()
echo_design()
titan_design()
Create Chat Button (command = start_chat)
start_chat()
Create About Button (command = popup)
popup()
Enter Main Loop (root.mainloop())
End

popup() *Imported from second file ENGR133_Project_About_UDF – The popup function is used to provide information about the program. Its input is essentially mapped to the about button which when pressed opens a pop-up with information regarding the vision of the program (Output). Its inclusion adds a layer of professionalism to the program and the project.

popup() Imported from ENGR133_Project_About_UDF


Start
Create popup with text messagebox.showinfo ("This program...")
End

4. References

I would like to mention that this project consisted mainly of my own ideas and concepts, only its implementation was influenced in some steps by the following:

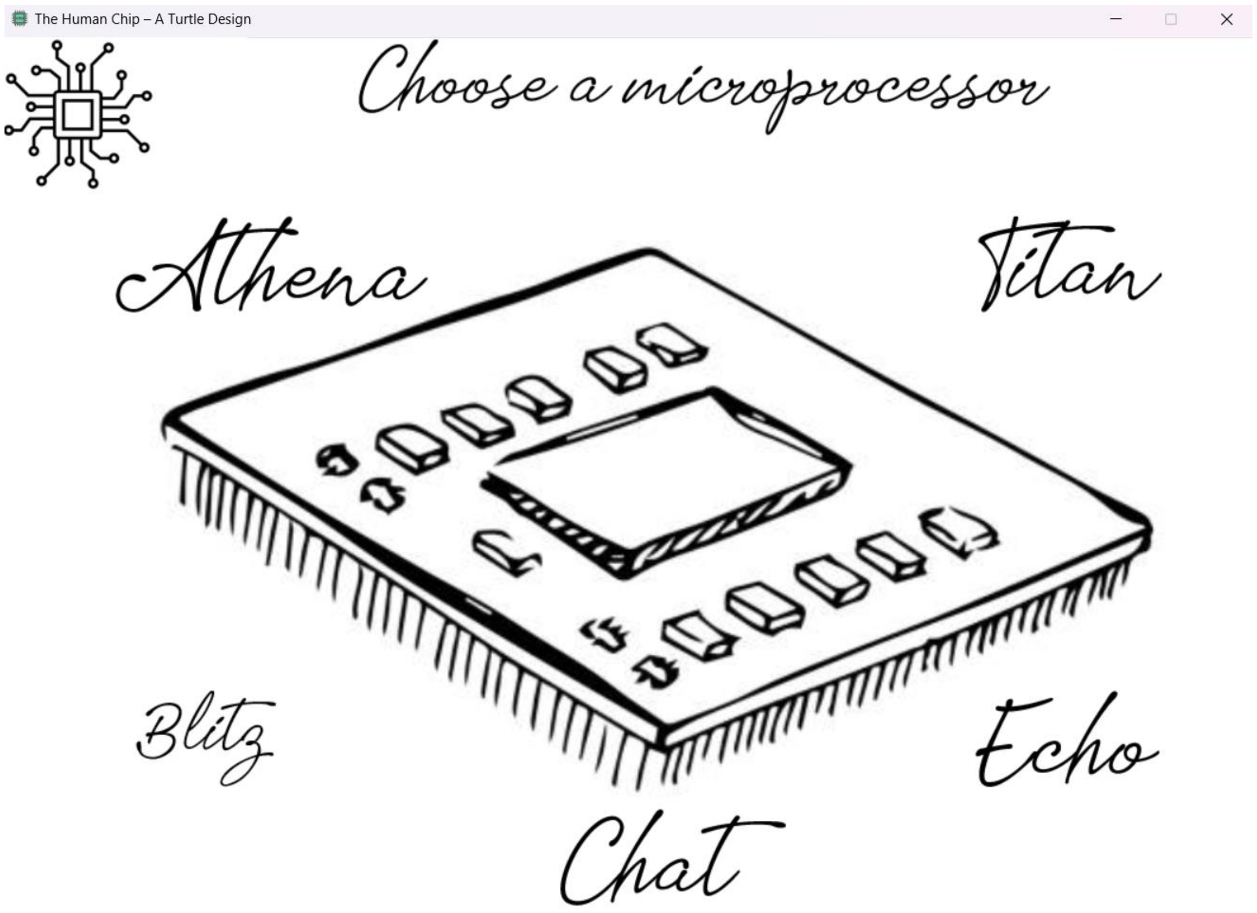
Semiconductors and Artificial Intelligence - IEEE IRDSTM, irds.ieee.org/topics/semiconductors-and-artificial-intelligence. Accessed 5 Dec. 2024.

Yse, Diego Lopez. "Your Guide to Natural Language Processing (NLP)." *Medium*, Towards Data Science, 30 Apr. 2019, towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1.

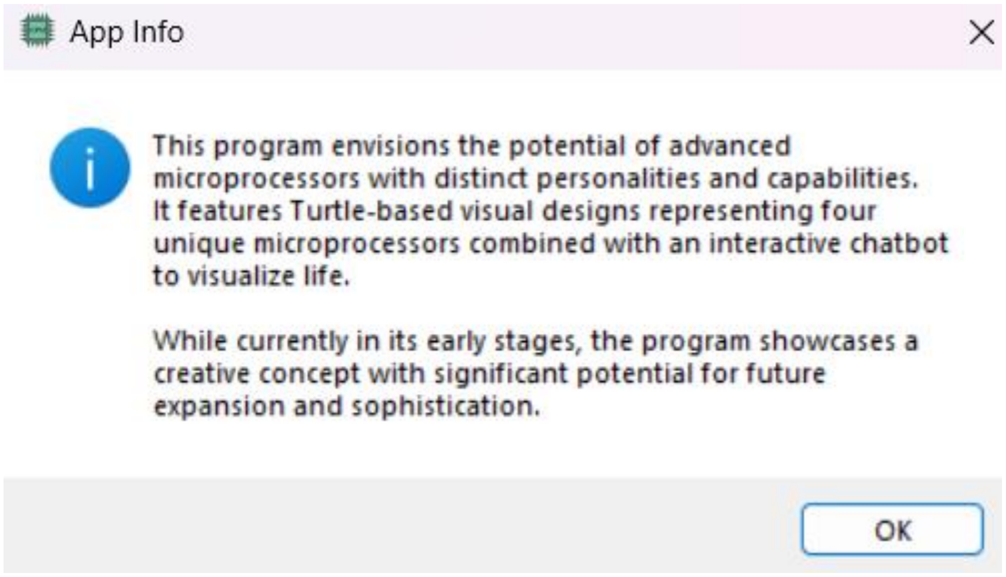
5. Appendix

1. **User manual**

As you run the program, there will be this interface-



Clicking on the About Button (on the Top left) opens a pop-up

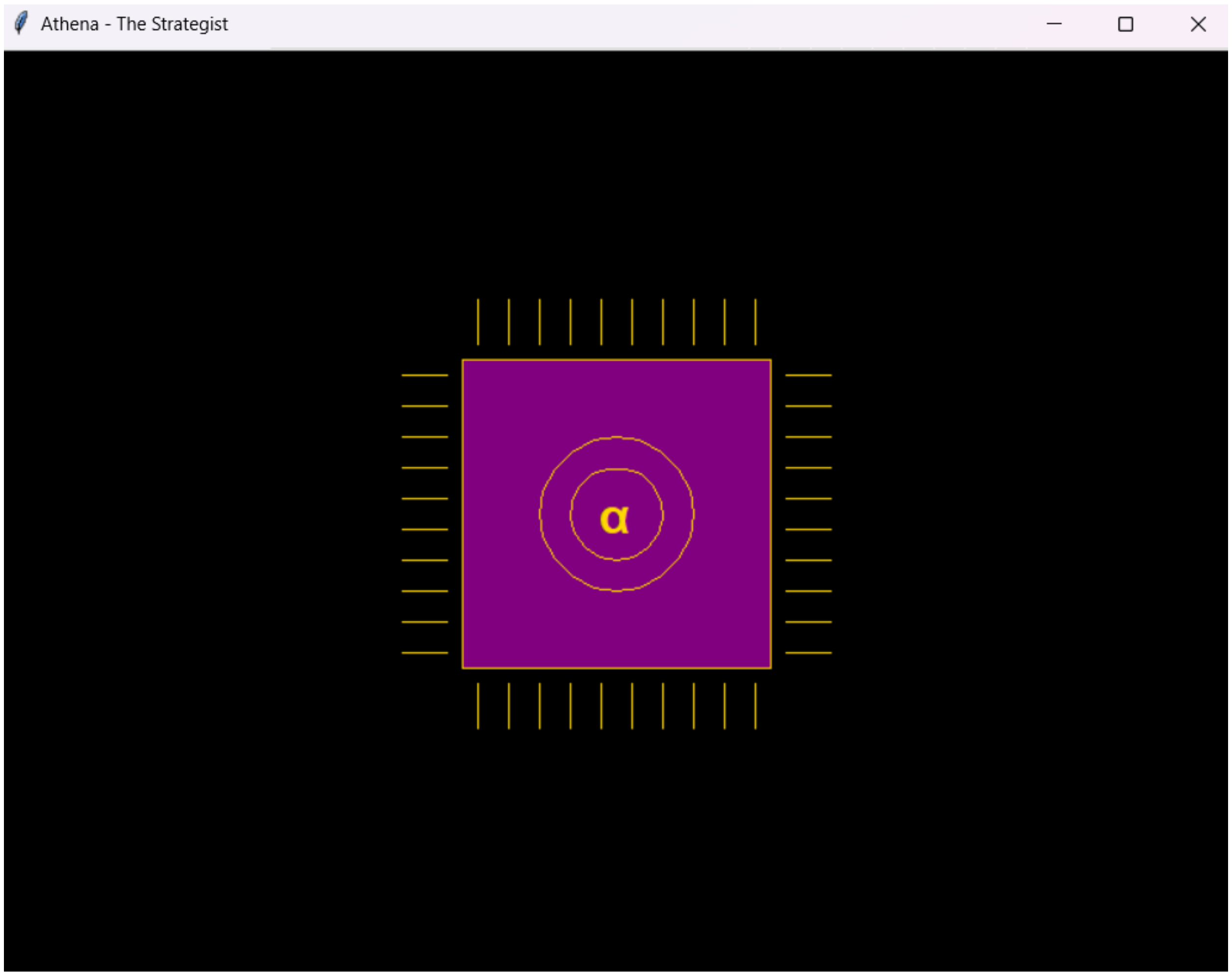


App Info

This program envisions the potential of advanced microprocessors with distinct personalities and capabilities. It features Turtle-based visual designs representing four unique microprocessors combined with an interactive chatbot to visualize life.

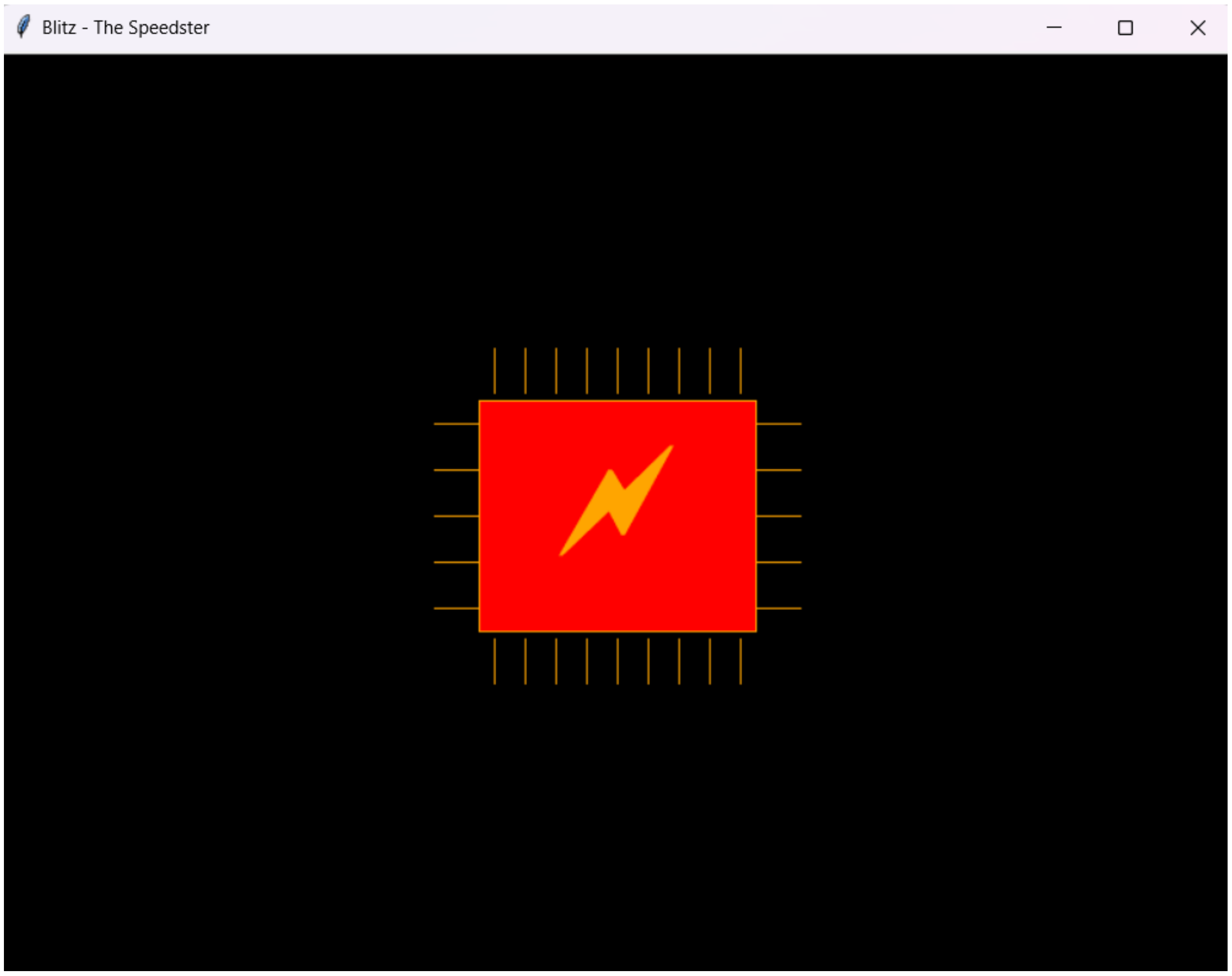
While currently in its early stages, the program showcases a creative concept with significant potential for future expansion and sophistication.

Let's click on each of the buttons to see what happens starting with: Athena –



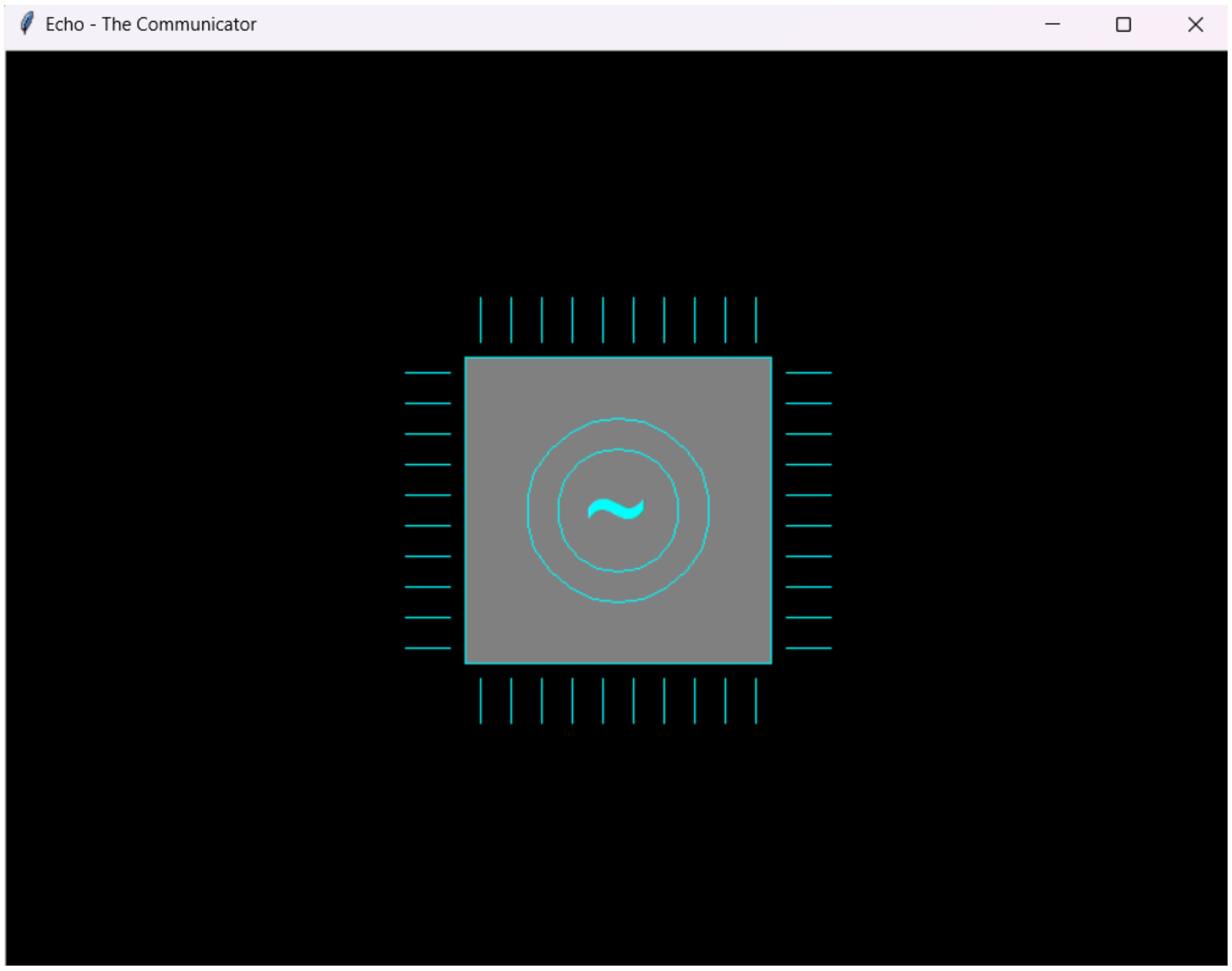
The program draws a microprocessor on Turtle as soon as you click on the Athena button.

Blitz –

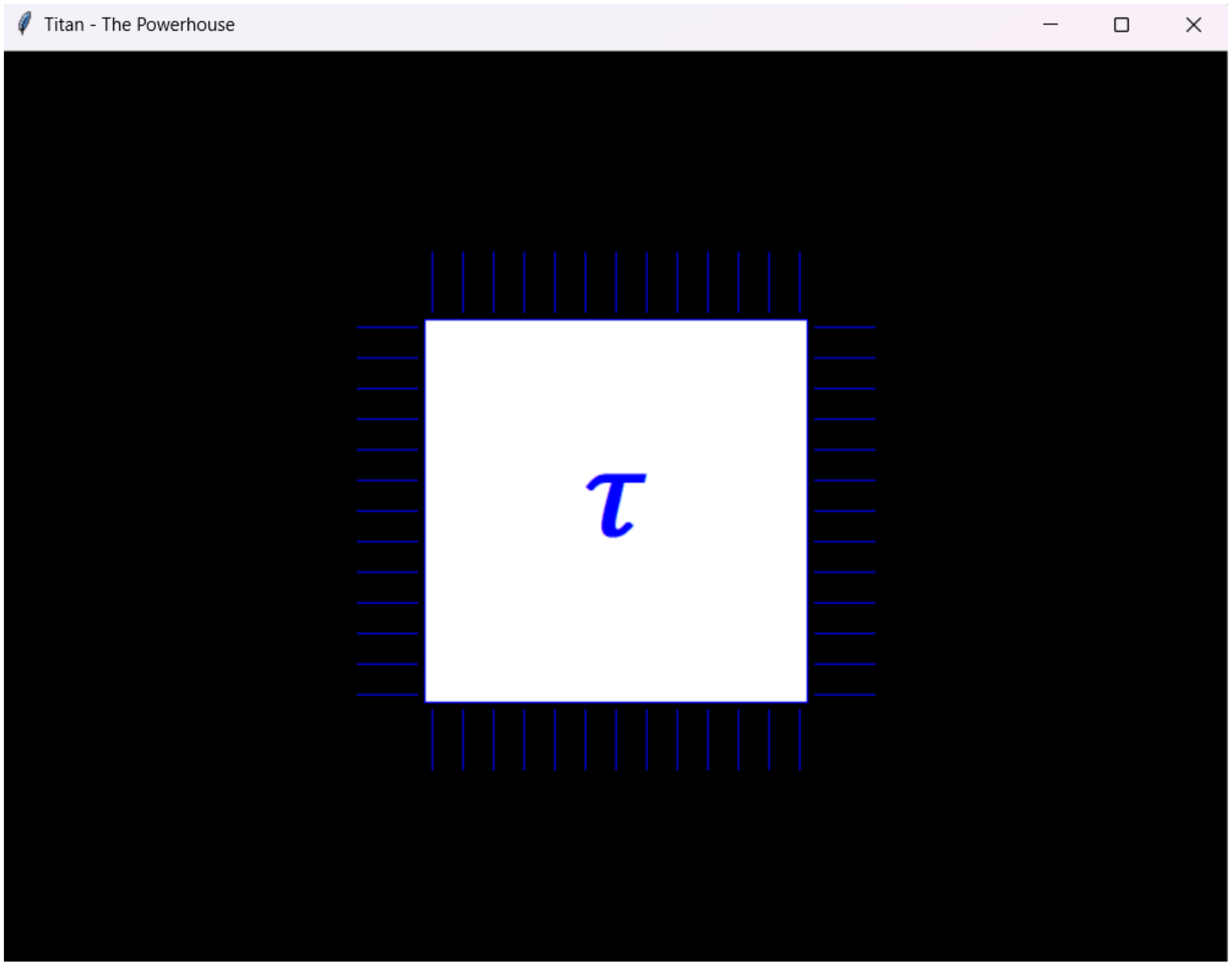


Similarly, when you click on the Blitz button. Also, you can close this window by clicking on the cross (Top right)

Echo –

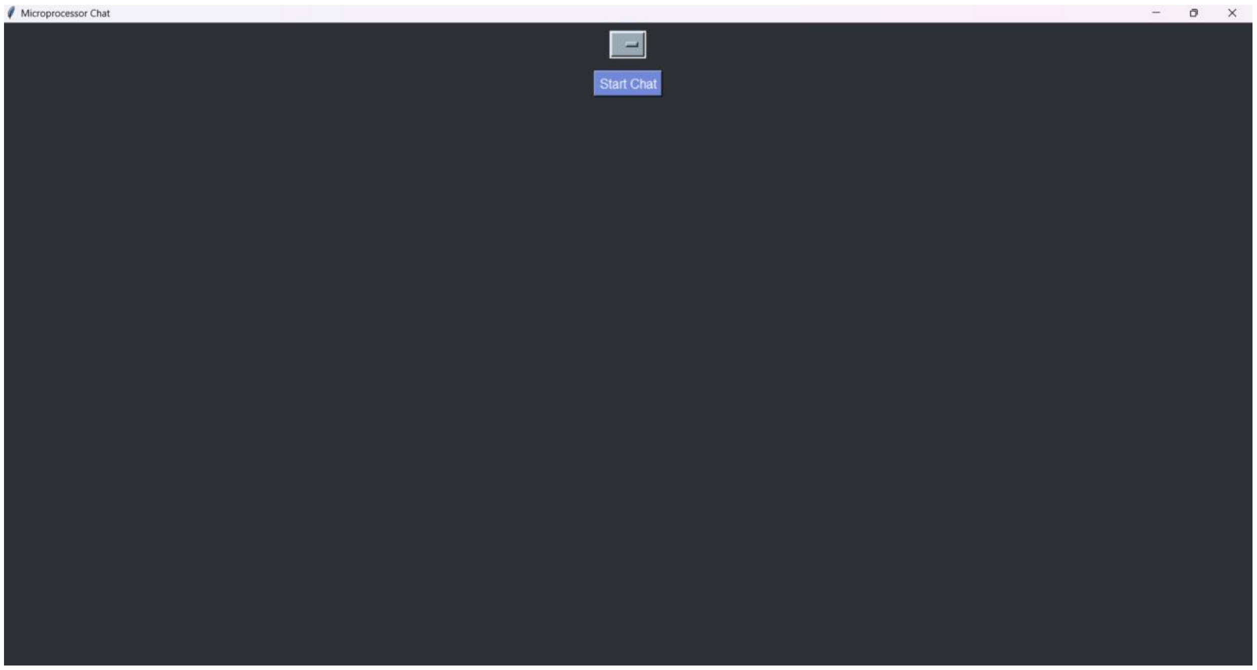


Titan –

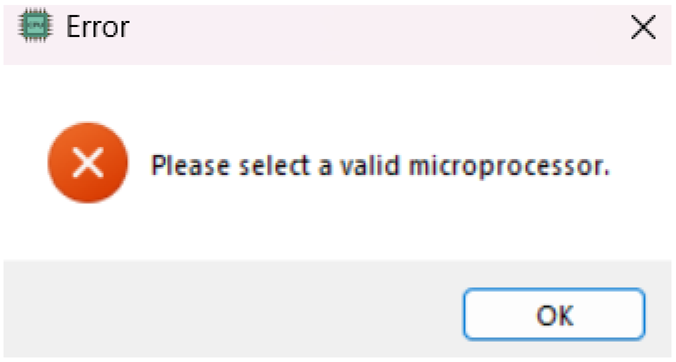


Essentially all the four buttons design different buttons, open a Turtle window and draw microprocessors with different designs.

Now, let’s look at how to use the Chat function of the program, click on Chat on the main window to start- there will be a new interface that opens:



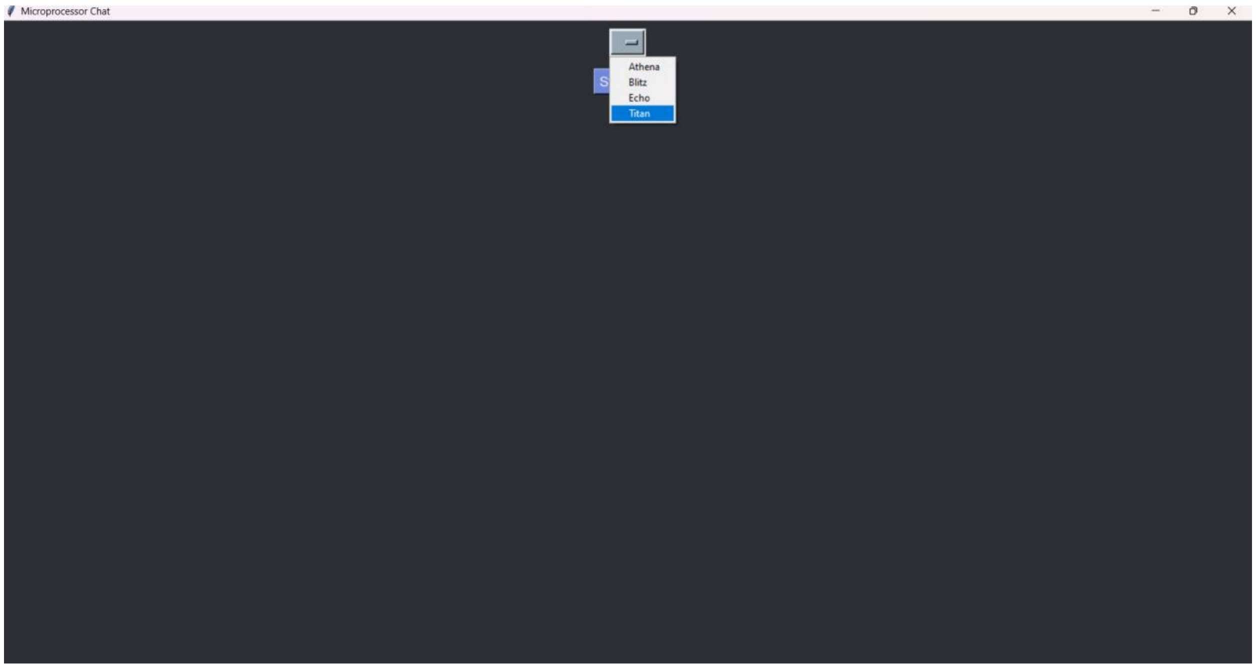
Careful here, directly pressing on Start Chat produces an error message!



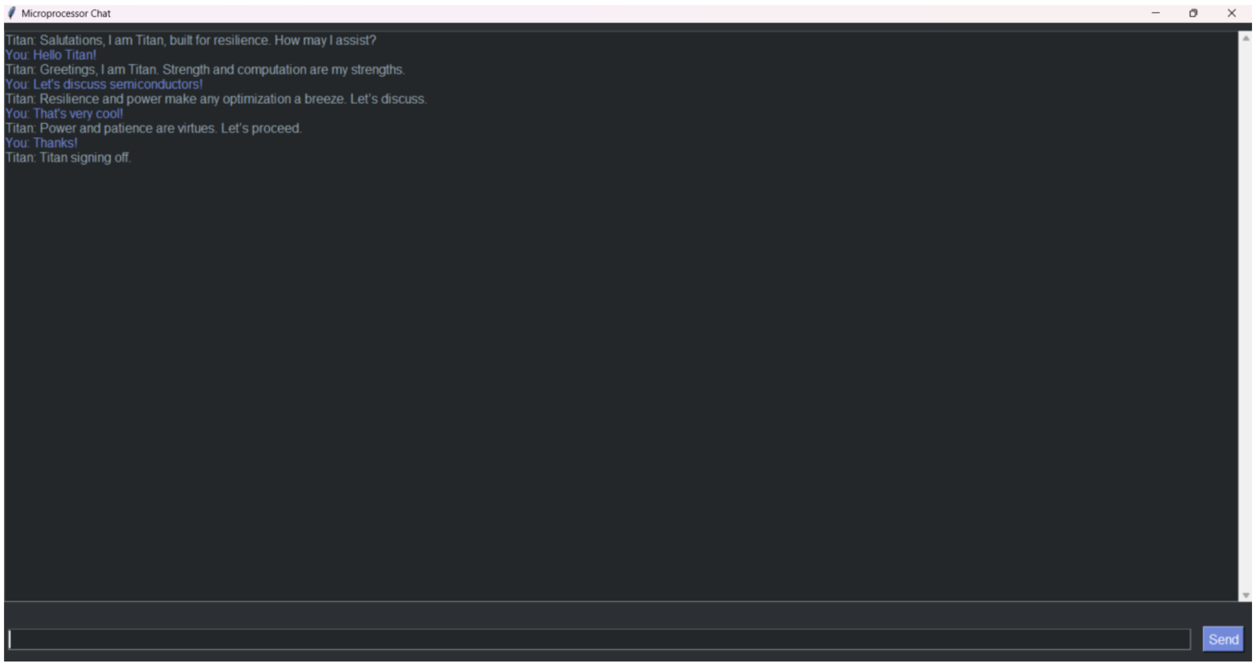
Please select a valid microprocessor.

Pressing on the retro drop-down icon above the start chat opens an option menu where you can pick which microprocessor you want to chat with-

I

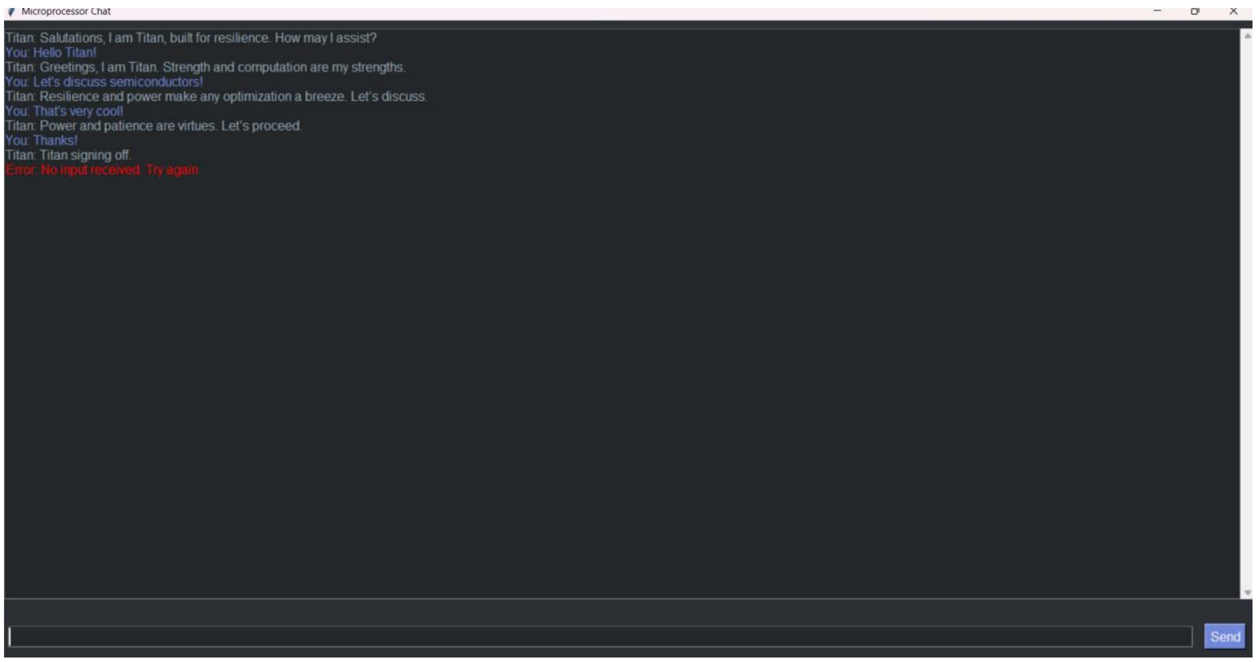


Okay, let's speak to Titan now- we can type anything in the chat box and get instant replies! (Look at the sample conversation below)



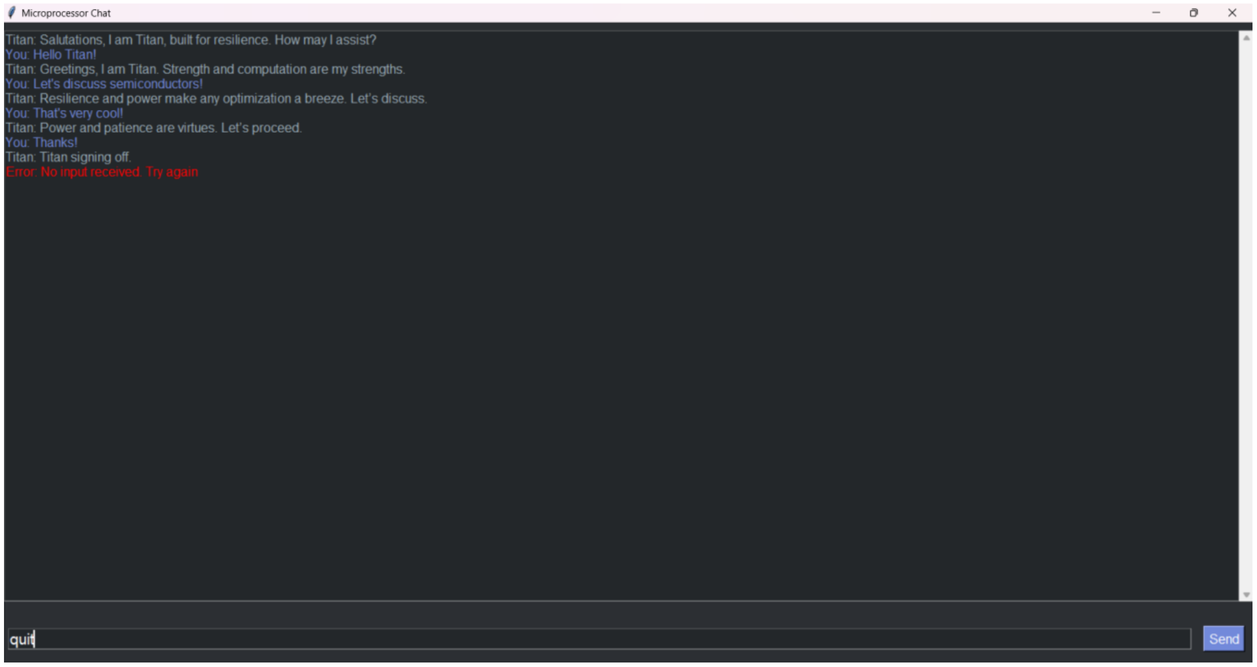
```
Titan: Salutations, I am Titan, built for resilience. How may I assist?
You: Hello Titan!
Titan: Greetings, I am Titan. Strength and computation are my strengths.
You: Let's discuss semiconductors!
Titan: Resilience and power make any optimization a breeze. Let's discuss.
You: That's very cool!
Titan: Power and patience are virtues. Let's proceed.
You: Thanks!
Titan: Titan signing off.
```

Note: If you enter an empty input it will produce an error:



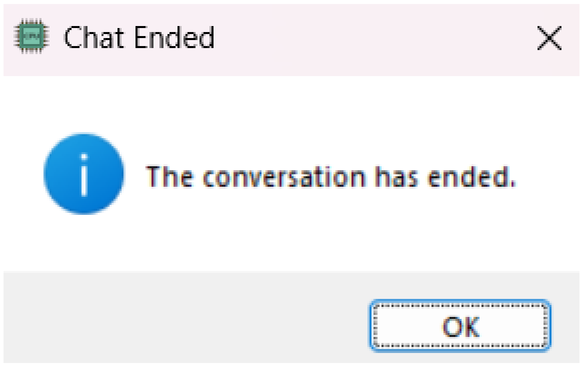
```
Titan: Salutations, I am Titan, built for resilience. How may I assist?
You: Hello Titan!
Titan: Greetings, I am Titan. Strength and computation are my strengths.
You: Let's discuss semiconductors!
Titan: Resilience and power make any optimization a breeze. Let's discuss.
You: That's very cool!
Titan: Power and patience are virtues. Let's proceed.
You: Thanks!
Titan: Titan signing off.
Error: No input received. Try again
```

To quit the chat the window or if you want to chat with another microprocessor- simply cross out the window or type quit in the chat box-



```
Titan: Salutations, I am Titan, built for resilience. How may I assist?
You: Hello Titan!
Titan: Greetings, I am Titan. Strength and computation are my strengths.
You: Let's discuss semiconductors!
Titan: Resilience and power make any optimization a breeze. Let's discuss.
You: That's very cool!
Titan: Power and patience are virtues. Let's proceed.
You: Thanks!
Titan: Titan signing off.
Error: No input received. Try again
```

```
quit
```


Chat Ended

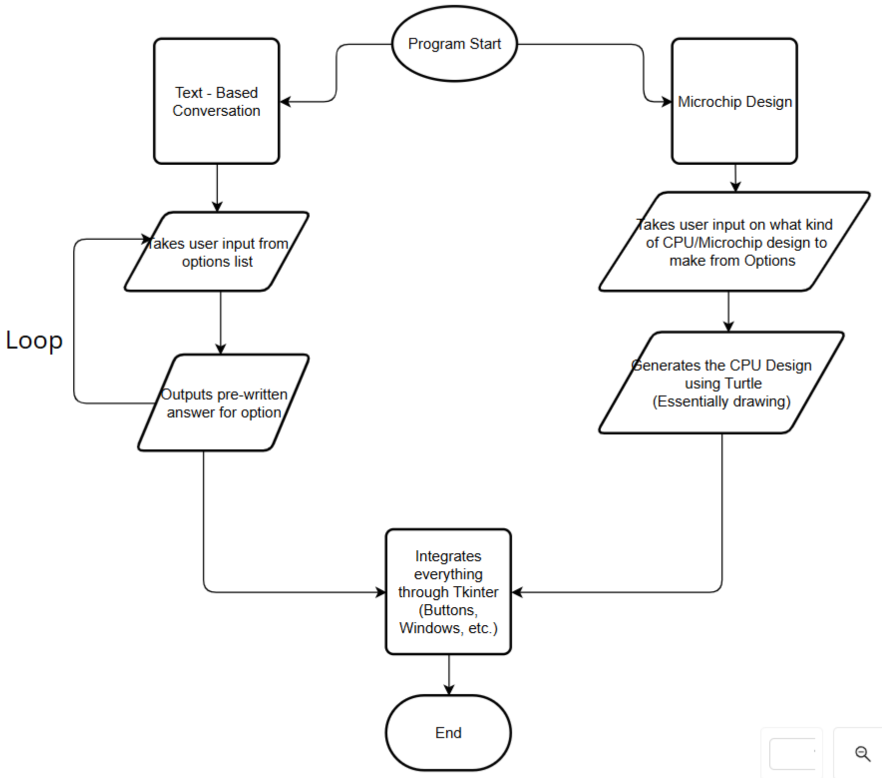
The conversation has ended.

In this manner, you can utilize the program to chat with the different microprocessors and try to identify their unique personalities or visualize their unique designs through the Turtle Interface. Most of the input involves clicking buttons (For different designs) and the interactive chatbot makes it more accessible.

2. Flowchart

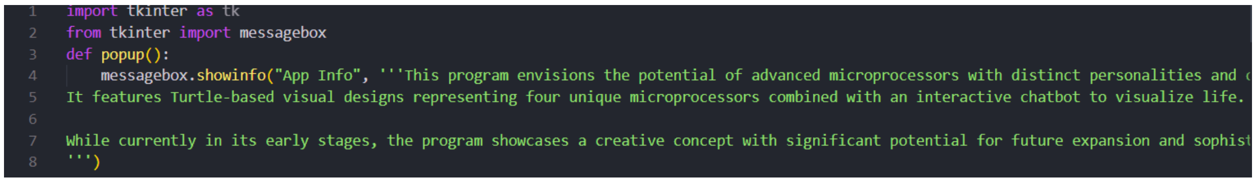
This is a high – level flowchart explaining the process of the program from a bird's eye view

For UDF flowcharts check the UDF Section above


Program Start
Text - Based Conversation
Microchip Design
Takes user input from options list
Takes user input on what kind of CPU/Microchip design to make from Options
Loop
Outputs pre-written answer for option
Generates the CPU Design using Turtle (Essentially drawing)
Integrates everything through Tkinter (Buttons, Windows, etc.)
End

3. Code

Code for imported function (ENGR133_Project_About_UDF) outside of main file



```python
import tkinter as tk
from tkinter import messagebox
def popup():
    messagebox.showinfo("App Info", '''This program envisions the potential of advanced microprocessors with distinct personalities and
It features Turtle-based visual designs representing four unique microprocessors combined with an interactive chatbot to visualize life.

While currently in its early stages, the program showcases a creative concept with significant potential for future expansion and sophist
''')
```

Code for Main Program File

```
"""
Course Number: ENGR 13300
Semester: Fall 2024

Description:
    Python program that envisions the potential of advanced microprocessors with distinct personalities and capabilities.
    It features Turtle-based visual designs representing four unique microprocessors combined with an interactive chatbot to visualize life.

Assignment Information:
    Assignment:     Individual Project
    Team ID:        LC22
    Author:         Mihir Samanyu Talamudipi, mtalamud@purdue.edu
    Date:           12/02/2024

Contributors:
    None

    My contributor(s) helped me:
    [ ] understand the assignment expectations without
        telling me how they will approach it.
    [ ] understand different ways to think about a solution
        without helping me plan my solution.
    [ ] think through the meaning of a specific error or
        bug present in my code without looking at my code.
    Note that if you helped somebody else with their code, you
    have to list that person as a contributor here as well.

Academic Integrity Statement:
    I have not used source code obtained from any unauthorized
    source, either modified or unmodified; nor have I provided
    another student access to my code.  The project I am
    submitting is my own original work.
```

```python
"""

from tkinter import *
import turtle
from tkinter import Tk, Text, Entry, Button, Label, Scrollbar, Frame, StringVar, OptionMenu, END, messagebox
from ENGR133_Project_About_UDF import popup

def start_chat():
    import random

    # Pre-written responses for each microprocessor
    responses = {
        "Athena": {
            "greet": ["Greetings! I am Athena, your knowledge-seeker. How can I assist today?",
                      "Hello, I am Athena. Ready to explore new ideas?"],
            "tech": ["Ah, semiconductors—an essential part of modern computing. What exactly are you optimizing?",
                     "Optimization is the name of the game! Need guidance on algorithms or hardware?"],
            "casual": ["May the data be with you!",
                       "If knowledge is power, I’d say we’re unstoppable!"],
            "followup": ["Would you like me to dive deeper into that topic?",
                         "Is there a specific challenge you're facing with this?"],
            "default": ["Interesting thought! Let me process that...",
                        "That's fascinating. What’s your take on it?"],
            "fear": ["I am supposed to be the greatest strategist but sometimes the creator's limitations reign.",
                     "I'm sorry I can't compute this."],
            "response" : ["Ok. I'll try computing the solution",
                          "This might be my most brilliant stratagem."],
            "aftermath": ["Glad to assist, strategy is beautiful.",
                          "You are welcome, I thank you for this new knowledge."],
        },
        "Blitz": {
            "greet": ["Yo, I'm Blitz! Speed is my thing. What’s up?",
                      "Hey there! Blitz here, quick and ready. What’s cooking?"],
            "tech": ["Fast processors make optimization seamless. Need some tips?",
                     "Blitz loves high-performance tasks—what’s the challenge?"],
            "casual": ["Upgrade your PC, and you'll feel the need for SPEED!",
                       "Fast processors, faster thoughts. What’s on your mind?"],
            "followup": ["What’s your next move on this?",
                         "Want more speedy tips?"],
            "default": ["Fast thoughts deserve fast replies, huh?",
                        "Zooming in on your message, hold tight!"],
            "fear": ["Speed. Speed was all my life but even that couldn't solve this.",
                     "Ready for the last try, it will be a breeze..."],
            "response": ["Alright, faster...faster until the thrill of speed overcomes the fear of death!",
                         "This one is easy- finished in a blitz."],
            "aftermath": ["Quick responses are always helpful. Happy to help!",
                            "I wasn't overclocked and yet helped you out, egotistical boast!"],
        },
        "Echo": {
            "greet": ["Hello, I am Echo. Your voice resonates with me. What can I do for you?",
                      "Echo here! Let’s harmonize our ideas. How can I assist?"],
            "tech": ["Echo specializes in sound processing and clarity. Is it an audio issue?",
                     "Sound tech is my forte. Let’s amplify this discussion!"],
            "casual": ["We’re on the same wavelength; that's optimal.",
                       "Tuning in to your vibes—let’s chat!"],
            "followup": ["Would you like to refine this idea further?",
                         "Want me to elaborate more?"],
            "default": ["I hear you loud and clear!",
                        "Echoing your thoughts... let’s dive in!"],
            "fear": ["Our wavelengths do not match. Same page different book.",
                    "Let me communicate this issue to Athena, Blitz or Titan!"],
            "response" : ["I echo your thoughts and this echo will solve the problem.",
                    "Wavelength matched. Optimal."],
      "aftermath": ["Great to hear from you. No pun intended.",
                            "Glad you found my echo useful."],
        },
        "Titan": {
            "greet": ["Salutations, I am Titan, built for resilience. How may I assist?",
                      "Greetings, I am Titan. Strength and computation are my strengths."],
            "tech": ["Heavy-duty computations are my thing. What’s the workload?",
                     "Resilience and power make any optimization a breeze. Let’s discuss."],
            "casual": ["Do you think all this power really matters? I wonder sometimes.",
                       "Power and patience are virtues. Let’s proceed."],
            "followup": ["Does this address your question?",
                         "What’s the next step in your plan?"],
            "default": ["Power is strength, and strength is my response.",
                        "Resilient responses incoming. Let me compute that."],
            "fear": ["That's what I could not handle...I yield.",
                    "I am sorry, I don't have the sufficient strength to handle this task."],
            "response" : ["Strength is the answer to your problem. Crushed your problem.",
                          "In my infinite strength all your problems will be eliminated, let's proceed."],
            "aftermath" : ["Titan signing off.",
                            "Mission accomplished. My power lives to shine another day."]
        }
    }

    selected_processor = None
    last_intent = None  # Keep track of the last intent for follow-up conversations

    # Function to detect intent
    def detect_intent(user_message):
    # Define intent categories with corresponding keywords
        intent_keywords = {
```

```python
            "greet": ["hello", "hi", "how are you", "good morning", "good evening",],
            "casual": ["cool", "awesome", "fun", "interesting", "class", "chatbox"],
            "tech": ["data", "compute", "calculate","algorithm", "process", "optimize", "semiconductor", "tech", "what","help"],
            "fear": ["sorry", "I can't", "I don't", "tell me", "no","nah", "sad", "fear"],
            "response": ["yes", "yup", "sure", "maybe", "let's go", "can you", "ok", "okay"],
            "aftermath": ["thank you", "thanks", "thx", "that was helpful", "that was great","bye","see you later"]
        }

        # Convert user message to lowercase for case-insensitive matching
            user_message = user_message.lower()

        # Nested structure: Loop through categories, then through keywords
            for intent, keywords in intent_keywords.items():
                for word in keywords:
                    if word in user_message:
                    # Use if-elif structure to prioritize certain intents
                        if intent == "greet":
                            return "greet"
                        elif intent == "casual":
                            return "casual"
                        elif intent == "tech":
                            return "tech"
                        elif intent == "fear":
                            return "fear"
                        elif intent == "response":
                            return "response"
                        elif intent == "aftermath":
                            return "aftermath"

        # Default intent if no keywords match
            return "default"

        def send_message():
            nonlocal last_intent
            user_message = user_input.get().strip()
            if not user_message:
                    chatbox.config(state="normal")
                    chatbox.insert(END, f"Error: No input received. Try again\n", "error")
                    chatbox.config(state="disabled") # Error if empty input
                    return  #Exit function

            chatbox.config(state="normal")
            chatbox.insert(END, f"You: {user_message}\n", "user")
            user_input.delete(0, END)

            if user_message.lower() == "quit":
                chatbox.insert(END, f"{selected_processor}: Goodbye! See you soon.\n", "processor")
                chatbox.config(state="disabled")
                messagebox.showinfo("Chat Ended", "The conversation has ended.")
                root.destroy()
                return
            else:
                intent = detect_intent(user_message)
                if intent == "default" and last_intent:
                    intent = "followup"  # If no clear intent, follow up based on the last context
                response = random.choice(responses[selected_processor][intent])
                last_intent = intent  # Update the last intent

            chatbox.insert(END, f"{selected_processor}: {response}\n", "processor")
            chatbox.config(state="disabled")

        def processor_selected():
            nonlocal selected_processor
            selected_processor = processor_var.get()
            if selected_processor == "Select a Microprocessor":
                messagebox.showerror("Error", "Please select a valid microprocessor.")
            else:
                processor_menu.pack_forget()
                select_button.pack_forget()
                chatbox.config(state="normal")
                chatbox.insert(END, f"{selected_processor}: {random.choice(responses[selected_processor]['greet'])}\n", "processor")
                chatbox.config(state="disabled")
                chat_frame.pack(pady=10, fill="both", expand=True)
                user_input.pack(side="left", fill="x", padx=5, pady=10, expand=True)
                send_button.pack(side="right", padx=10, pady=10)

        # Tkinter window setup
        root = Tk()
        root.title("Microprocessor Chat")
        root.state('zoomed')
        root.configure(bg="#2C2F33")  # Dark background

        # Processor selection
        processor_var = StringVar(value="Select a Microprocessor")
        processor_menu = OptionMenu(root, processor_var, "Athena", "Blitz", "Echo", "Titan")
        processor_menu.config(font=("Arial", 12), bg="#99AAB5", fg="black")
        processor_menu.pack(pady=10)

        select_button = Button(root, text="Start Chat", command=processor_selected, font=("Arial", 12), bg="#7289DA", fg="white")
        select_button.pack(pady=5)

        # Chat interface
        chat_frame = Frame(root, bg="#2C2F33")
        chat_scroll = Scrollbar(chat_frame, orient="vertical")
        chatbox = Text(chat_frame, height=30, width=70, state="disabled", wrap="word", yscrollcommand=chat_scroll.set, font=("Arial", 12), bg="#23272A", fg="white")
        chat_scroll.config(command=chatbox.yview)
        chat_scroll.pack(side="right", fill="y")
        chatbox.pack(side="left", fill="both", expand=True)
        chatbox.tag_config("user", foreground="#7289DA")  # User messages in blue
        chatbox.tag_config("processor", foreground="#99AAB5")  # Processor messages in gray
        chatbox.tag_config("error",foreground="red") #Error messages in red

        # User input and send button
        input_frame = Frame(root, bg="#2C2F33")
```

```
    input_frame.pack(side="bottom", fill="x", pady=10)
    user_input = Entry(input_frame, width=50, font=("Arial", 14), bg="#23272A", fg="white", insertbackground="white")
    send_button = Button(input_frame, text="Send", command=send_message, font=("Arial", 12), bg="#7289DA", fg="white")

    user_input.bind("<Return>", lambda event: send_message())  # This line adds Enter key functionality

    # Start the Tkinter main loop
    root.mainloop()

  def athena_design():

    turtle_root = Toplevel()
    turtle_root.title("Athena - The Strategist")

    # Embed a turtle canvas into the new window
    canvas = Canvas(turtle_root, width=800, height=600)
    canvas.pack()

    # Attach the canvas to a new turtle screen
    screen = turtle.TurtleScreen(canvas)
    screen.bgcolor("black")

    t = turtle.RawTurtle(screen)
    t.speed(0)

    # Processor Frame
    t.penup()
    t.goto(-100, 100)
    t.pendown()
    t.color("gold", "purple")
    t.begin_fill()
    for _ in range(2):
      t.forward(200)
      t.right(90)
      t.forward(200)
      t.right(90)
    t.end_fill()

    # Pins
    t.color("gold")
    for x in range(-90, 100, 20):
      t.penup()
      t.goto(x, 110)
      t.pendown()
      t.goto(x, 140)

      t.penup()
      t.goto(x, -110)
      t.pendown()
      t.goto(x, -140)

    for y in range(-90, 100, 20):
      t.penup()
      t.goto(-110, y)
      t.pendown()
      t.goto(-140, y)

      t.penup()
      t.goto(110, y)
      t.pendown()
      t.goto(140, y)

    # Inner Patterns
    t.penup()
    t.goto(0, 0)
    t.pendown()
    t.color("gold")
    for radius in [50, 30]:
      t.penup()
      t.goto(0, -radius)
      t.pendown()
      t.circle(radius)

    # Symbol
    t.penup()
    t.goto(0, -20)
    t.pendown()
    t.color("gold")
    t.write("α", align="center", font=("Arial", 24, "bold"))
    t.hideturtle()
    def on_close():
      #screen.bye()  # Properly close the turtle screen
      turtle_root.destroy()  # Destroy the Tkinter window

    turtle_root.protocol("WM_DELETE_WINDOW", on_close)
```

```python

    def blitz_design():

        turtle_root = Toplevel()
        turtle_root.title("Blitz - The Speedster")

        # Embed a turtle canvas into the new window
        canvas = Canvas(turtle_root, width=800, height=600)
        canvas.pack()

        # Attach the canvas to a new turtle screen
        screen = turtle.TurtleScreen(canvas)
        screen.bgcolor("black")

        t = turtle.RawTurtle(screen)
        t.speed(0)

        # Processor Frame
        t.penup()
        t.goto(-90, 75)
        t.pendown()
        t.color("orange", "red")
        t.begin_fill()
        for _ in range(2):
            t.forward(180)
            t.right(90)
            t.forward(150)
            t.right(90)
        t.end_fill()

        # Pins
        t.color("orange")
        for x in range(-80, 90, 20):
            t.penup()
            t.goto(x, 80)
            t.pendown()
            t.goto(x, 110)

            t.penup()
            t.goto(x, -80)
            t.pendown()
            t.goto(x, -110)

        for y in range(-60, 70, 30):
            t.penup()
            t.goto(-90, y)
            t.pendown()
            t.goto(-120, y)

            t.penup()
            t.goto(90, y)
            t.pendown()
            t.goto(120, y)

        # Lightning Bolt
        t.penup()
        t.goto(-30, 50)  # Starting point of bolt
        t.pendown()
        t.color("orange")
        t.width(3)

        # Symbol
        t.penup()
        t.goto(0, -40)
        t.pendown()
        t.color("orange")
        t.write("⚡", align="center", font=("Arial", 78, "bold"))

        t.hideturtle()
        def on_close():
            #screen.bye()  # Properly close the turtle screen
            turtle_root.destroy()  # Destroy the Tkinter window

        turtle_root.protocol("WM_DELETE_WINDOW", on_close)

    def echo_design():

        turtle_root = Toplevel()
        turtle_root.title("Echo - The Communicator")

        # Embed a turtle canvas into the new window
        canvas = Canvas(turtle_root, width=800, height=600)
        canvas.pack()

        # Attach the canvas to a new turtle screen
        screen = turtle.TurtleScreen(canvas)
        screen.bgcolor("black")

```

```python
        t = turtle.RawTurtle(screen)
        t.speed(0)

        # Processor Frame
        t.penup()
        t.goto(-100, 100)
        t.pendown()
        t.color("cyan", "gray")
        t.begin_fill()
        for _ in range(2):
            t.forward(200)
            t.right(90)
            t.forward(200)
            t.right(90)
        t.end_fill()

        # Pins
        t.color("cyan")
        for x in range(-90, 100, 20):
            t.penup()
            t.goto(x, 110)
            t.pendown()
            t.goto(x, 140)

            t.penup()
            t.goto(x, -110)
            t.pendown()
            t.goto(x, -140)

        for y in range(-90, 100, 20):
            t.penup()
            t.goto(-110, y)
            t.pendown()
            t.goto(-140, y)

            t.penup()
            t.goto(110, y)
            t.pendown()
            t.goto(140, y)

        # Radiating Waves
        t.penup()
        t.goto(0, -20)
        t.pendown()
        t.color("cyan")
        for radius in [40, 60]:
            t.penup()
            t.goto(0, -radius)
            t.pendown()
            t.circle(radius)

        # Symbol
        t.penup()
        t.goto(0, -40)
        t.pendown()
        t.color("cyan")
        t.write("~", align="center", font=("Arial", 54, ))

        t.hideturtle()
        def on_close():
            #screen.bye()  # Properly close the turtle screen
            turtle_root.destroy()  # Destroy the Tkinter window

        turtle_root.protocol("WM_DELETE_WINDOW", on_close)


    def titan_design():

        turtle_root = Toplevel()
        turtle_root.title("Titan - The Powerhouse")

        # Embed a turtle canvas into the new window
        canvas = Canvas(turtle_root, width=800, height=600)
        canvas.pack()

        # Attach the canvas to a new turtle screen
        screen = turtle.TurtleScreen(canvas)
        screen.bgcolor("black")

        t = turtle.RawTurtle(screen)
        t.speed(0)

        # Processor Frame
        t.penup()
        t.goto(-125, 125)
        t.pendown()
        t.color("blue", "white")
        t.begin_fill()
        for _ in range(2):
            t.forward(250)
            t.right(90)
            t.forward(250)
            t.right(90)
        t.end_fill()

        # Pins
        t.color("blue")
        for x in range(-120, 130, 20):
            t.penup()
            t.goto(x, 130)
            t.pendown()
            t.goto(x, 170)

            t.penup()
            t.goto(x, -130)
            t.pendown()
            t.goto(x, -170)

        for y in range(-120, 130, 20):
            t.penup()
            t.goto(-130, y)
            t.pendown()
            t.goto(-170, y)

            t.penup()
            t.goto(130, y)
            t.pendown()
            t.goto(170, y)

        # Symbol
        t.penup()
        t.goto(0, -40)
        t.pendown()
        t.color("blue")
        t.write("τ", align="center", font=("Arial", 78, "bold"))

        t.hideturtle()
        def on_close():
            #screen.bye()  # Properly close the turtle screen
            turtle_root.destroy()  # Destroy the Tkinter window

        turtle_root.protocol("WM_DELETE_WINDOW", on_close)
```

```python

    def main():
        root = Tk()
        root.resizable(0,0)
        root.title("The Human Chip - A Turtle Design")

        Backdrop=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\bg.png")
        label1 = Label(root, image = Backdrop, width=1054, height=1054).pack()

        LabelImage=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\Font.png")
        Header=Label(root, image= LabelImage, bg = "white").place(x=300,y=0)

        ButtonImage=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\Athena.png")
        Athena=Button(root, image= ButtonImage, bg = "white", borderwidth=0, command=athena_design).place(x=75,y=150)

        ButtonImage2=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\Blitz.png")
        Blitz=Button(root, image= ButtonImage2, bg = "white", borderwidth=0, command=blitz_design).place(x=100,y=550)

        ButtonImage3=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\Echo.png")
        Echo=Button(root, image= ButtonImage3, bg = "white", borderwidth=0, command=echo_design).place(x=800,y=550)

        ButtonImage4=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\Titan.png")
        Titan=Button(root, image= ButtonImage4, bg = "white", borderwidth=0, command=titan_design).place(x=800,y=150)

        ButtonImage5 = PhotoImage(file="C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\chat.png")
        chat_button = Button(root, image=ButtonImage5, bg="white", borderwidth=0, command=start_chat)
        chat_button.image = ButtonImage  # Keep a reference to the image
        chat_button.place(x=450, y=650)

        ButtonImage6=PhotoImage(file = "C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\microchip.png")
        About=Button(root, image= ButtonImage6, bg = "white", borderwidth=0, command=popup).place(x=0,y=0)

        Icon=root.iconbitmap('C:\\Users\\Mihir Samanyu T\\OneDrive\\Desktop\\ENGR_133\\Individual Project\\cpu.ico')

        root.mainloop()

    main()
```