

Multi-threaded Dictionary Server Report

Server Architecture:

Our server follows a client-server architecture using sockets and threads. It implements a multi-threaded design to handle concurrent client requests. The server listens for incoming connections on a specified port and responds to client queries with the meanings of words from the dictionary.

Design Choices:

- We chose to use Python for the implementation due to its simplicity and extensive standard library support for networking and threading. - We opted for a multi-threaded design to handle concurrent client requests. - Error handling is implemented to manage network communication and data processing errors.

Features:

- Allows clients to query the meaning of a word. - Supports concurrent connections using a worker pool architecture. - Implements error handling for network communication and data processing. - Provides a command-line interface for launching the server.

Words Dictionary:

hello: used as a greeting

world: the earth and all people and things on it

python: a large constricting snake found in tropical and subtropical regions

algorithm: a step-by-step procedure or set of rules for solving a problem or accomplishing a task

database: an organized collection of structured information or data, typically stored electronically in a computer system

framework: a basic structure underlying a system, concept, or text

encryption: the process of converting information or data into a code, especially to prevent unauthorized access

artificial intelligence: the theory and development of computer systems capable of performing tasks that typically require human intelligence

machine learning: a subset of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to learn and improve from experience

big data: extremely large datasets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior and interactions

cloud computing: the delivery of computing services, including servers, storage, databases, networking, software, and analytics, over the internet (the cloud) to offer faster innovation, flexible resources, and economies of scale

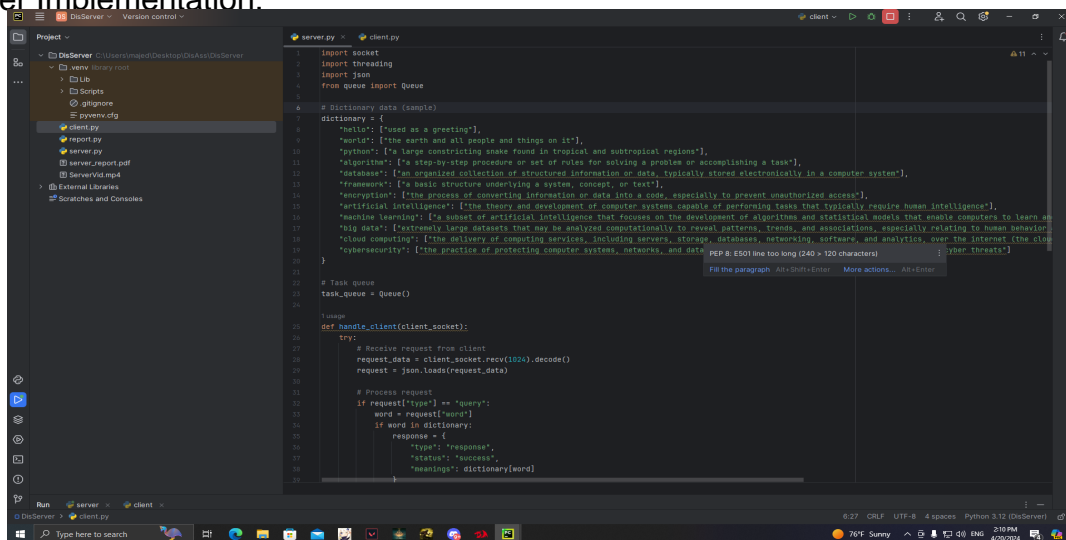
cybersecurity: the practice of protecting computer systems, networks, and data from digital attacks, unauthorized access, and other cyber threats

Video Demo:

[Click here to watch the video demo](#)

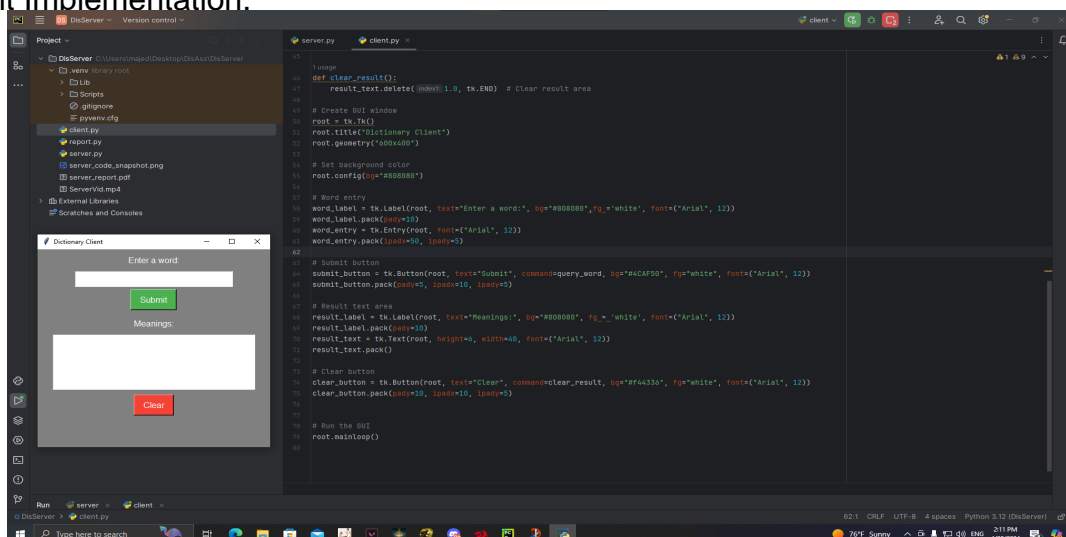
Code Snapshots:

Server Implementation:



```
1 import socket
2 import threading
3 import json
4 from queue import Queue
5
6 # Dictionary data (sample)
7 dictionary = {
8     "hello": ["used as a greeting"],
9     "world": ["the earth and all people and things on it"],
10    "python": ["a large constricting snake found in tropical and subregional regions"],
11    "algorithm": ["a step-by-step procedure or set of rules for solving a problem or accomplishing a task"],
12    "database": ["an organized collection of structured information or data, typically stored electronically in a computer system"],
13    "framework": ["a basic structure underlying a system, concept, or text"],
14    "encryption": ["the process of converting information or data into a code, especially to prevent unauthorized access"],
15    "artificial intelligence": ["the theory and development of computer systems capable of performing tasks that typically require human intelligence"],
16    "machine learning": ["a subset of artificial intelligence that focuses on the development of algorithms and statistical models that enable computers to learn and improve from experience"],
17    "big data": ["extremely large datasets that may be analyzed computationally to reveal patterns, trends, and associations, especially relating to human behavior"],
18    "cloud computing": ["the delivery of computing services, including servers, storage, databases, networking, software, and analytics, over the internet (the cloud)"],
19    "cybersecurity": ["the practice of protecting computer systems, networks, and data from digital attacks, unauthorized access, and other cyber threats"]
20 }
21
22 # Task queue
23 task_queue = Queue()
24
25 # Handle client request
26 def handle_client(client_socket):
27     try:
28         # Receive request from client
29         request_data = client_socket.recv(1024).decode()
30         request = json.loads(request_data)
31
32         # Process request
33         if request["type"] == "query":
34             word = request["word"]
35             if word in dictionary:
36                 response = {
37                     "type": "response",
38                     "status": "success",
39                     "meanings": dictionary[word]
40                 }
41             else:
42                 response = {
43                     "type": "response",
44                     "status": "failure",
45                     "meanings": []
46                 }
47             # Send response back to client
48             response_data = json.dumps(response)
49             client_socket.send(response_data.encode())
50     except Exception as e:
51         print(f"Error: {e}")
52
53 # Main function
54 def main():
55     # Create server socket
56     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
57     server_socket.bind(('0.0.0.0', 8080))
58     server_socket.listen(5)
59
60     # Accept client connections
61     while True:
62         client_socket, address = server_socket.accept()
63         # Create a new thread to handle the client request
64         thread = threading.Thread(target=handle_client, args=(client_socket,))
65         thread.start()
```

Client Implementation:



```
1 # Import Tkinter
2 from tkinter import *
3
4 # Create a window
5 root = Tk()
6 root.title("Dictionary Client")
7 root.geometry("400x400")
8
9 # Set background color
10 root.config(bg="#f0f0f0")
11
12 # Word entry
13 word_label = tk.Label(root, text="Enter a word:", bg="#f0f0f0", fg="white", font=("Arial", 12))
14 word_label.pack(pady=10)
15 word_entry = tk.Entry(root, font=("Arial", 12))
16 word_entry.pack(pady=10, ipady=3)
17
18 # Submit button
19 submit_button = tk.Button(root, text="Submit", command=query_word, bg="#f0f0f0", fg="white", font=("Arial", 12))
20 submit_button.pack(pady=10, ipady=3)
21
22 # Result text area
23 result_label = tk.Label(root, text="Meanings:", bg="#f0f0f0", fg="white", font=("Arial", 12))
24 result_label.pack(pady=10)
25 result_text = tk.Text(root, height=4, width=40, font=("Arial", 12))
26 result_text.pack(pady=10)
27
28 # Clear button
29 clear_button = tk.Button(root, text="Clear", command=clear_result, bg="#f0f0f0", fg="white", font=("Arial", 12))
30 clear_button.pack(pady=10, ipady=3)
31
32 # Run the GUI
33 root.mainloop()
```

Conclusion:

Overall, our multi-threaded dictionary server provides a reliable and efficient solution for querying word meanings concurrently. With its flexible architecture and robust error handling, it meets the requirements of the project and demonstrates the effective use of sockets and threads.