**Abstract**

This project implements a **DNS Resolver System** designed to simulate the operation of real-world DNS servers, featuring both a **local Proxy DNS server** and a **centralized Main DNS server**. The system aims to **remove abstraction** and provide a clear understanding of how DNS resolution works, demonstrating both how **recursive calls** are made and how DNS queries are processed step by step. By replicating the key components of DNS resolution, the project offers an educational platform to learn how DNS servers communicate, resolve domain names, and handle various types of queries.

The system includes features such as **database-driven query resolution**, **dynamic alias generation**, **load balancing**, and **multi-threaded client-server interaction** to simulate real-world DNS behaviors efficiently.

### **System Features and Components**

### **Main DNS Server**

* The Main DNS Server is the core of the system, managing DNS records and using an SQLite database (main\_dns.db) to store domain-to-IP address mappings. When a client sends a query, the Main DNS Server:
* Searches the database for the corresponding IP address.
* If the domain is not found, the system generates dynamic aliases (e.g., abc.example.com) using a random prefix algorithm and stores these in a canonical records database (canonical.db).
* This alias generation mimics DNS CNAME records, ensuring that unresolved domains are handled efficiently.
* Additionally, when a new website is introduced, the Main DNS Server allows new records to be added to the database. The system assigns a unique IP address to the new website and generates alias names for it, which are also stored in the database. This ensures that new domains can be resolved and managed effectively within the DNS system.

**Proxy DNS Server**

The Proxy DNS Server acts as an intermediary between the client and the Main DNS Server. It:

* Listens for incoming client queries.
* Forwards unresolved queries to the Main DNS Server.

(unresolved query- if request domain is not found in the proxy\_dns.db which is the local dns database)

* Returns resolved IP addresses or alias records back to the client.

(the resolved IP addresses is obtained from the main )

The Proxy DNS Server uses multi-threading to handle multiple client connections concurrently, allowing the system to scale efficiently under high query loads.

**Client Interaction**

Clients interact with the system by sending DNS queries. The interaction flow is as follows:

* The client sends a request to the Proxy DNS Server.
* If the domain is unresolved locally, the Proxy forwards the query to the Main DNS Server.
* The Main DNS Server queries its database for a matching IP address. If found, it returns the IP; if not, it generates an alias and updates the canonical database.
* The Proxy DNS Server relays the resolved IP or alias back to the client.

**Load Balancing and Rotation Mechanism**

To optimize performance, the system incorporates a **round-robin rotation mechanism** for load balancing. When multiple IP addresses exist for a single domain, the system cycles through the available IPs to evenly distribute client requests, simulating real-world DNS resolver behavior and ensuring fair resource usage.

**Error Handling and Robustness**

The system includes error-handling mechanisms for invalid queries, database connection issues, and network errors, ensuring the system remains stable and resilient during query processing.

### **Workflow of the Code**

1. **Initialization**

The Proxy DNS Server starts and listens for client requests. Databases (main\_dns.db and canonical.db) are initialized for DNS and alias management.

1. **Client Query Handling**

Clients send queries to the Proxy DNS Server. If the query is unresolved locally, the Proxy forwards it to the Main DNS Server.

1. **DNS Record Lookup**

The Main DNS Server checks the SQLite database for DNS records. If records are found, IP addresses are returned. If no records exist, aliases are generated dynamically and stored in the canonical database.

1. **Load Balancing**

When multiple IP addresses are available for a domain, the system uses a round-robin mechanism to distribute the load evenly across the available IPs.

1. **Response to Client**

The resolved IP address or alias is sent back to the Proxy DNS Server, which relays it to the client.

1. **Concurrency**

Multi-threading ensures the Proxy DNS Server can handle multiple client queries concurrently.

### **Key Contributions**

* **Database-Driven DNS Management**: Efficient use of SQLite for storing and retrieving DNS records and aliases.
* **Dynamic Alias Generation**: Provides a fallback mechanism for unresolved domains.
* **Load Balancing**: Even distribution of traffic across multiple IP addresses.
* **Scalable and Concurrent**: Multi-threaded proxy server handles simultaneous client connections.
* **Client-Server Communication**: Realistic DNS query and response workflows using socket programming.

### **Applications**

This DNS Resolver System can be used for:

* **Educational purposes** to understand DNS resolution, database interaction, and multi-threading.
* **Network simulations** to test DNS-based workflows and load balancing.
* **Small-scale DNS management applications** requiring alias generation and record lookups.

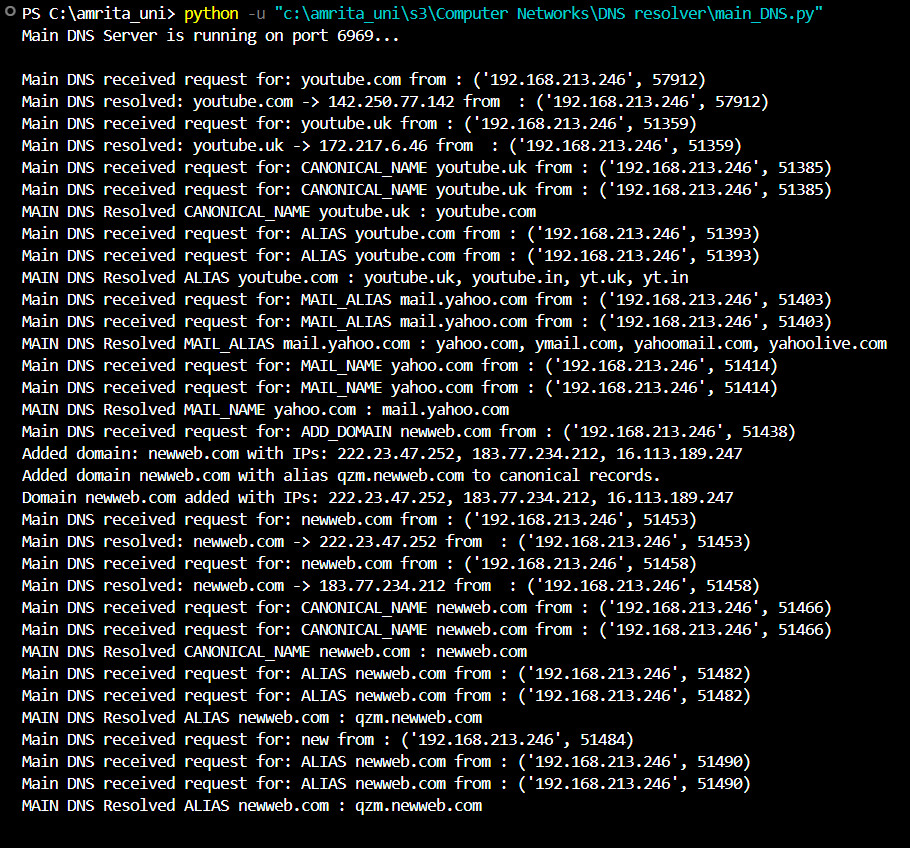
### **Conclusion**

This project demonstrates a modular, database-driven approach to DNS resolution, removing abstraction and showing how recursive DNS queries are processed between local and centralized servers. It incorporates dynamic aliasing, load balancing, and concurrent request handling, providing a robust and scalable solution for simulations, learning, and future enhancements.

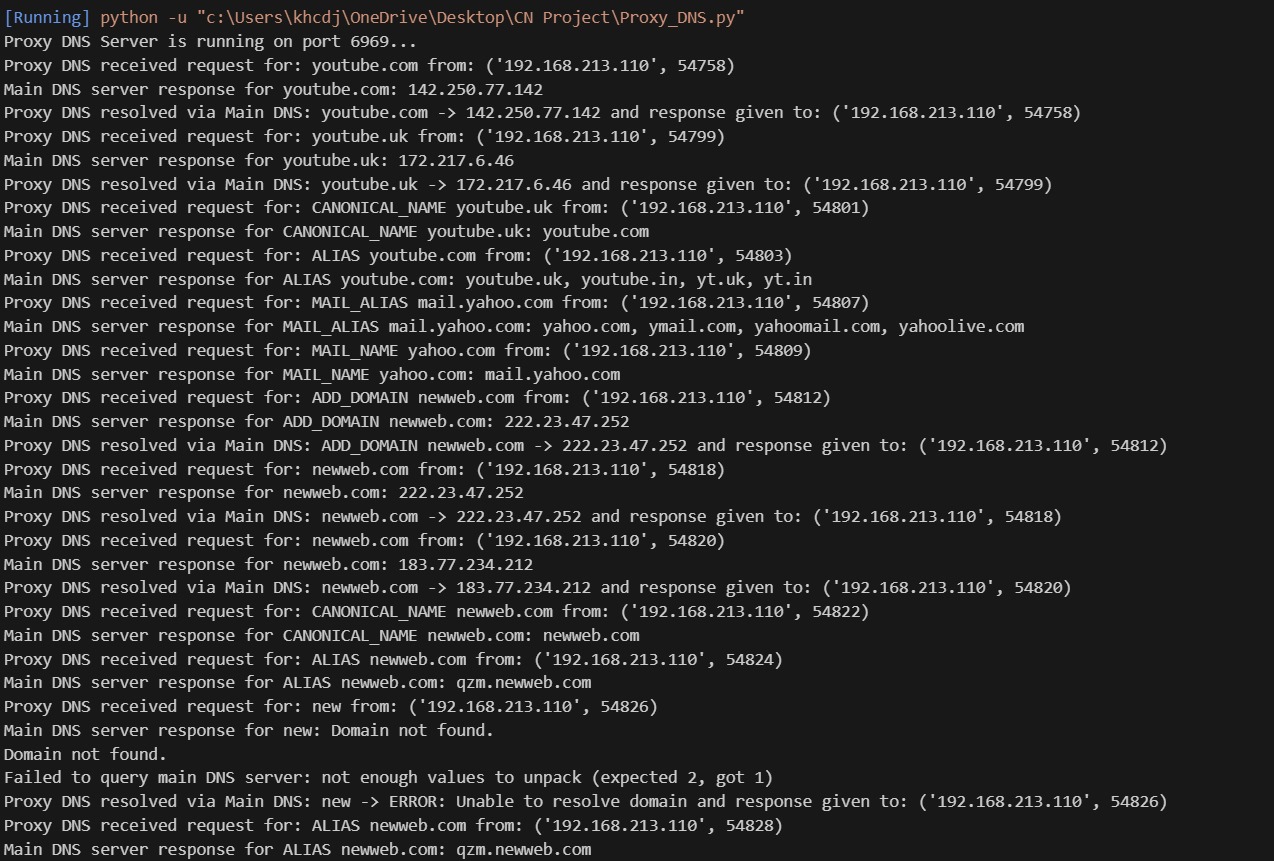
(Look below for Codes and Working model screenshots )

Codes and Working Model pics

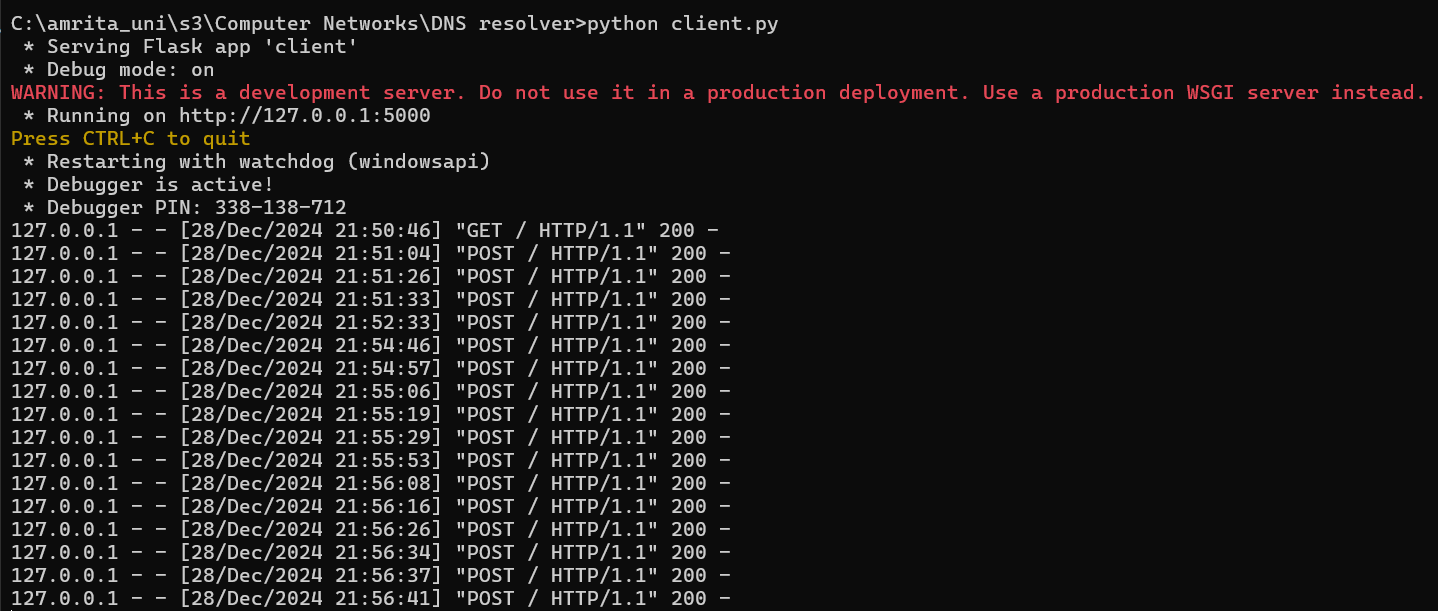
main\_DNS.py output :- (while client requests, abstraction behind main dns)



Proxy\_DNS.py output :- (while client requests, abstraction behind proxy dns)



Client.py output :-



Webpage working of client: (working of each button and function in code)

