✅ Canonical Scope of Work: Stream Node Dynamic Network Configuration (Ubuntu 24.04)

🧩 General Objective

Create a robust, production-grade network setup for a mobile stream node running Ubuntu 24.04 that can dynamically adapt to various site infrastructures worldwide. It must provide DHCP to LAN clients via redundant interfaces, obtain internet from WAN interfaces dynamically, and optionally act as a wireless AP or STA. Control must be possible via Flask using executable scripts located in /home/cornerpins/portal/.

🔌 Interface Roles & Requirements

1. LAN Interfaces (Always Serve DHCP to Clients)

These interfaces must *always* run dnsmasq and never receive DHCP from external sources.

- eno1: Copper NIC
- enp5s0f0: Fiber NIC (LAN side)
- wlp9s0 (in AP mode only): Wi-Fi LAN

These are part of the internal subnet (192.168.83.0/24) and must always advertise DHCP, regardless of WAN state. They must be fully isolated from upstream DHCP servers to prevent leakage and conflict.

---

2. WAN Interfaces (Must Request IP via DHCP)

These interfaces are WAN uplinks that must *never* run DHCP server services.

- enp5s0f1: Fiber NIC (WAN)
- enp6s0: Copper NIC (WAN)

These must request a dynamic IP via DHCP client and provide internet access to LAN via NAT/masquerading. DHCP server must never bind or advertise on these interfaces.

---

3. Wireless Interface: wlp9s0

This interface must support toggling between:

- Access Point (AP) mode:
    - SSID: CornerPins
    - Password: l.@m.b@tm@n.83

    - Must serve DHCP (on same 192.168.83.0/24 subnet) and share internet from available WAN
- Client (STA) mode:
    - Connects to existing Wi-Fi
    - Acts as WAN source and passes internet to LAN + host system
    - No DHCP served here in STA mode

A toggle script must exist at /home/cornerpins/portal/wifi_toggle.sh, and instructions must be documented in README_WIFI.txt.

📄 DHCP Reservations

- Format: JSON file located at /home/cornerpins/portal/dhcp.json
- Flask app will write/update this file
- dnsmasq must parse this JSON and serve static leases appropriately
- Restart of dnsmasq will be triggered by Flask app upon changes
- No separate script is needed to parse JSON — Flask does it

⚙️ System Requirements

DHCP Server (dnsmasq)

- Must always bind only to:
    - eno1
    - enp5s0f0
    - wlp9s0 (in AP mode only)
- Must never bind to:
    - enp5s0f1
    - enp6s0
    - Loopback

NAT Routing

- Outbound internet traffic from LAN must be NAT'd via whichever WAN is active
- Enable forwarding in /etc/sysctl.conf
- Use MASQUERADE iptables rule (or nftables) for active WAN interface

Scripts

All control scripts must live in /home/cornerpins/portal/ and be executable by user cornerpins.

- /home/cornerpins/portal/wifi_toggle.sh:
  Toggle between AP and STA modes
- /home/cornerpins/portal/manage_dnsmasq.sh:
  Dynamically enable/disable dnsmasq based on interface state
- Flask must be able to call all scripts programmatically via subprocess or os.system()

🔒 Security & Failover

- dnsmasq must never advertise on WAN
- When WAN changes or drops, system must:
  - Fall back to another active WAN if available
  - Keep DHCP running on LAN interfaces
- Wireless AP must not conflict with STA mode. When switching:
  - AP mode: start hostapd, stop wpa_supplicant
  - STA mode: stop hostapd, start wpa_supplicant

🧠 Behavior Summary

Interface	Type	DHCP Role	Notes
eno1	LAN	DHCP server	Always active, 192.168.83.1
enp5s0f0	LAN	DHCP server	Redundant to eno1
enp5s0f1	WAN	DHCP client	Never serve DHCP
enp6s0	WAN	DHCP client	Never serve DHCP
wlp9s0	Wi-Fi	Dual-mode	AP = DHCP server, STA = DHCP client

🧪 Testing Checklist

- ✅ Static leases via Flask → reflected in dnsmasq
- ✅ DHCP only on LAN (never WAN)
- ✅ DHCP server works if either eno1 or enp5s0f0 are connected

- ✅ Wi-Fi toggle functions correctly from Desktop + Flask
- ✅ DNSMASQ only runs when required interfaces are up
- ✅ Internet passthrough works from WAN to LAN
- ✅ No DHCP leak to WAN ever