

# **Project 13 – CIDR Subnet & Supernet Simulator**

**Done By :** Rifat (21BRS1183) , Vellore Institute of Technology

**Supervisor/Mentor:** Dr Swaminathan

**Institution/Organization:** Vellore Institute of Technology (VIT), Chennai

Date : 8<sup>th</sup> November 2025

# 1. Abstract

This project presents an interactive simulator for Classless Inter-Domain Routing (CIDR) that helps students plan and understand IPv4 subnetting and supernetting. The problem addressed is the difficulty many learners face when mapping theory to practical network design choices such as selecting prefix lengths, allocating hosts, and aggregating routes. The objective is to provide a clean, step-by-step interface that accepts common inputs (base IP/prefix, number of networks, or host requirements) and outputs clear tables with network ID, first/last usable addresses, broadcast address, and address-wastage metrics. The methodology follows a lightweight software engineering process: requirements capture, modular design, and iterative prototyping using Python and Streamlit. The implementation includes three subnetting modes—equal split, VLSM (split by host counts), and a two-level hierarchical split—plus a supernetting module that derives the longest common prefix for a set of CIDR blocks and reports routing-table reduction. Results show that the tool explains each decision transparently and generates reproducible tables and visual cues, making it suitable for labs and demonstrations. In conclusion, the simulator offers an approachable path from theory to practice and can be extended with IPv6, persistence, and access-control features in future work.

## 2. Introduction

### Background

CIDR replaced rigid classful addressing by expressing networks as  $\langle\text{network}\rangle/\langle\text{prefix}\rangle$ . While this increased efficiency and routing scalability, students often struggle to calculate usable hosts, plan subnets of varying sizes, and visualise how separate networks can be aggregated.

### Motivation

Labs typically show static examples. A hands-on, parameterised simulator allows experimentation with real values, helping learners build intuition about prefix manipulation and address planning.

### Project Scope

The project covers IPv4 subnetting (equal split, VLSM, and two-level hierarchical splitting) and supernetting (CIDR aggregation). It focuses on calculation logic, tabular summaries, and explanatory tips. It does not include IPv6, DHCP/DNS integration, or persistence to external databases.

### Objectives

- Provide an interactive interface to calculate subnets and supernets.
- Present network ID, broadcast, first/last usable, and address-wastage metrics clearly.
- Offer explanatory notes that map theory to decisions taken for each split mode.
- Include lightweight visualisation for equal/hierarchical splits.

## **Expected Outcome**

A working Streamlit app that students can drive with typical lab inputs to obtain validated, well-formatted results.

## **4. Literature Review**

### **Existing Solutions**

Common subnet calculators compute ranges, but many do not explain intermediate decisions or support hierarchical splits. Teaching material often remains static, without interactive routing aggregation examples.

### **Gap Analysis**

This project emphasises learning: side-by-side modes (equal, VLSM, hierarchical), transparent steps, and a supernetting view that demonstrates common-prefix discovery and routing-entry reduction.

### **Technologies Used (Comparison)**

Python with Streamlit provides quick UI iteration compared with raw Flask/Django for teaching tools. The built-in `ipaddress` module ensures standards-compliant IPv4 arithmetic, while pandas structures tabular output. Matplotlib is used for compact visual cues.

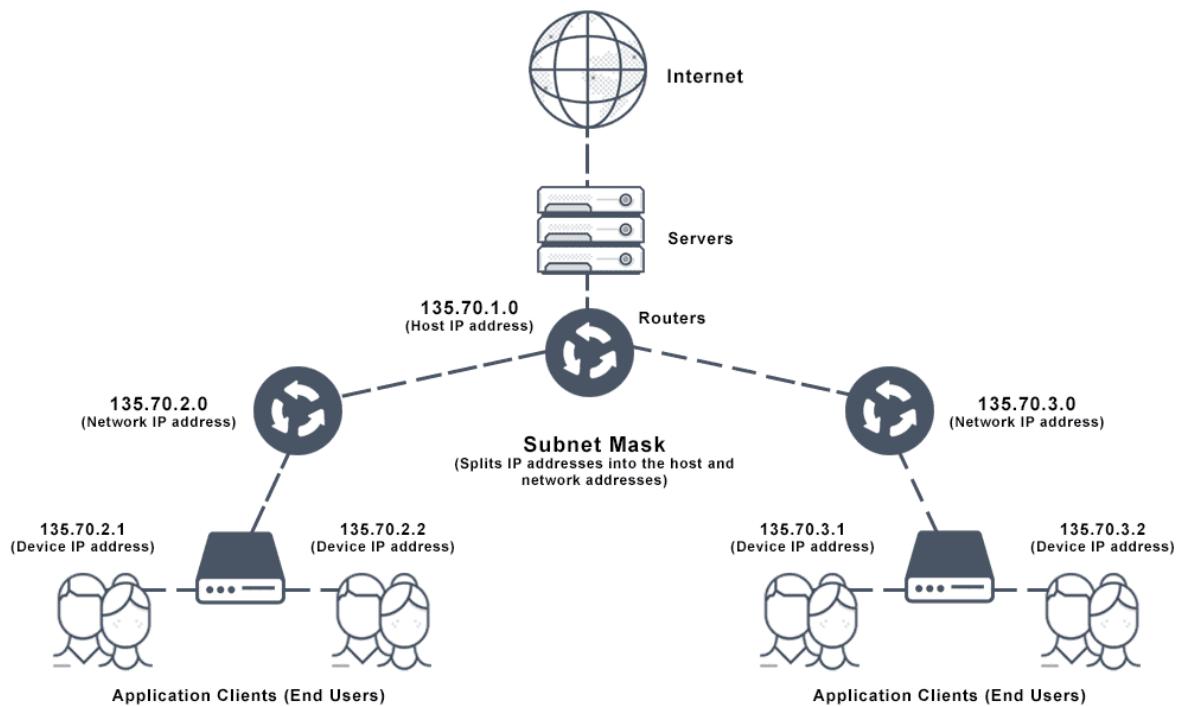
# 5. System Design & Architecture

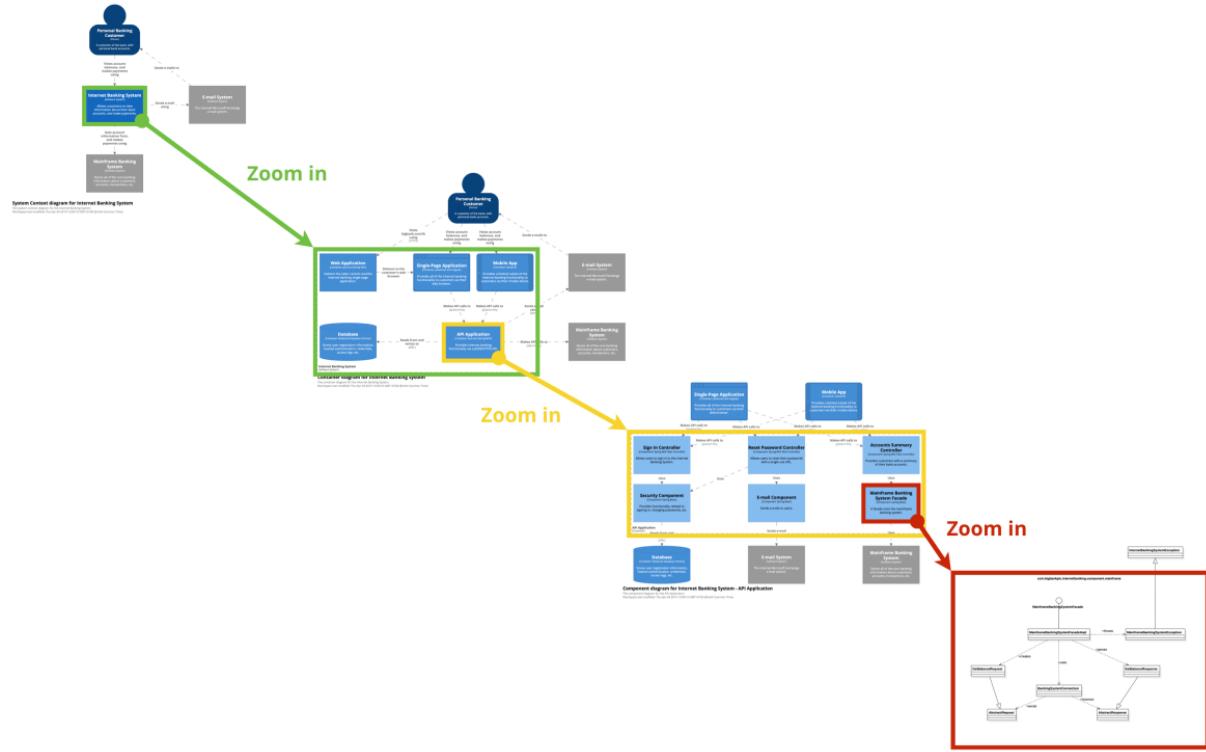
## 5.1 Software System Design

### System Overview

The app accepts a base network and user preferences, computes subnets or a supernet, and prints results in tables, with optional tips and a small visual block layout for equal/hierarchical splits.

### Software Architecture:

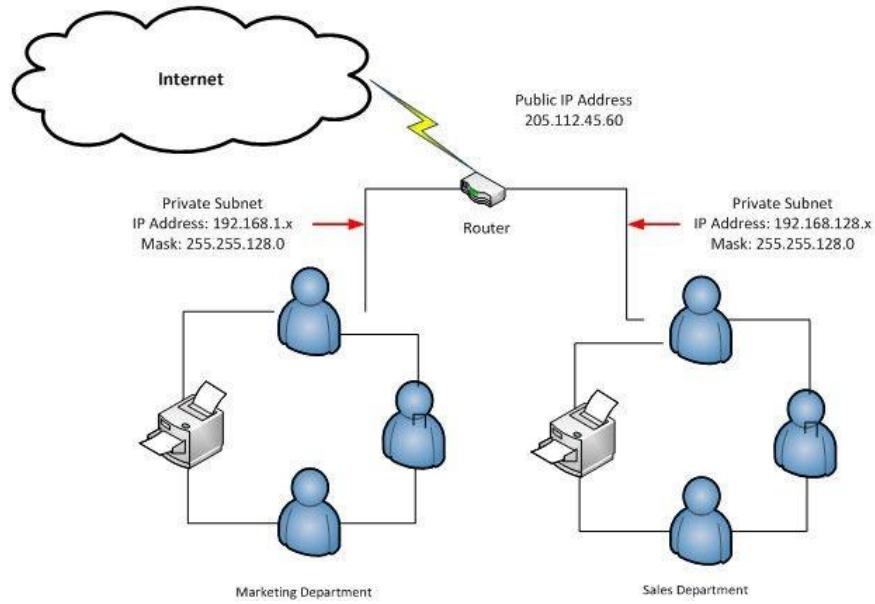




## Technologies Used

- Language: Python 3
- Framework/UI: Streamlit
- Libraries: ipaddress, pandas, matplotlib
- Runtime: Google Colab with Cloudflared for public link exposure

## Use Case Diagram



## 5.2 Required Hardware System

A standard laptop/desktop with a modern browser is sufficient. The app runs in Colab or locally with Python 3 installed.

## 6. Implementation

### 6.1 Software Implementation

Development followed an iterative, test-as-you-build approach. Calculators were unit-tested against known /26, /27, and /30 cases,

and the UI was refined to keep inputs minimal and outputs readable.

## Module Description

- Equal Split: borrows  $\lceil \log_2(N) \rceil$  bits from the host portion and lists each child block.
- VLSM: sorts/requested sizes, allocates the next fitting block, and advances the cursor while staying within the base block.
- Hierarchical Split: performs equal Level-1 split, then applies a VLSM-like allocation within the first branch.
- Supernetting: normalises inputs, finds the longest common prefix, expands if needed to cover the full min–max range, and reports the aggregate.

## Algorithm/Pseudocode (Core Ideas)

Equal split:  $\text{new\_prefix} = \text{base\_prefix} + \lceil \log_2(N) \rceil$ ;  $\text{child\_size} = 2^{32 - \text{new\_prefix}}$ ; iterate  $\text{child\_size}$  offsets.

VLSM: for each host need  $h$ , compute  $\text{pre} = 32 - \lceil \log_2(h + \text{overhead}) \rceil$ ; align to  $\text{block\_size}$ ; emit range; move cursor.

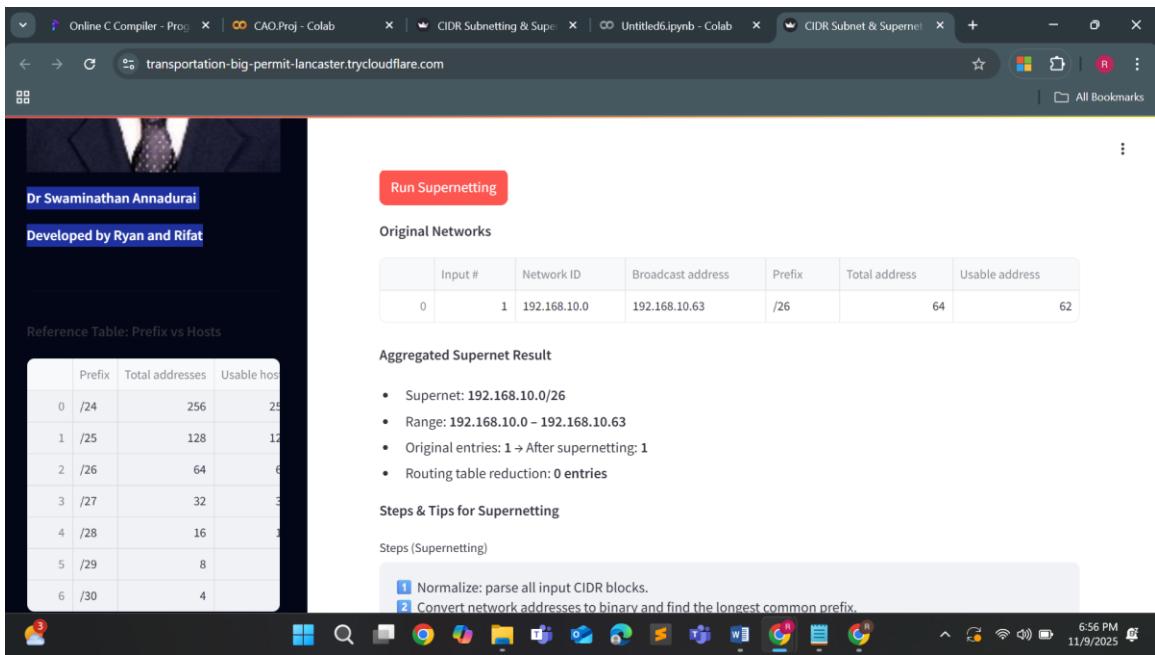
Supernet: sort networks, compute bitwise common prefix over network addresses, then ensure resulting block covers min–max.

## Database Schema

Not applicable; data is computed in-memory.

# 7. Results and Testing

## 7.1 Software Testing



Run Supernetting

Original Networks

	Input #	Network ID	Broadcast address	Prefix	Total address	Usable address
	0	192.168.10.0	192.168.10.63	/26	64	62

Aggregated Supernet Result

- Supernet: 192.168.10.0/26
- Range: 192.168.10.0 – 192.168.10.63
- Original entries: 1 → After supernetting: 1
- Routing table reduction: 0 entries

Steps & Tips for Supernetting

Steps (Supernetting)

- Normalize: parse all input CIDR blocks.
- Convert network addresses to binary and find the longest common prefix.

Functional tests: verified network/broadcast/usable counts for standard prefixes (/24, /26, /27, /30) and edge prefixes (/31, /32). Non-functional: the app renders quickly for classroom-scale inputs. Security considerations are basic, limited to client-side usage.

Sample observation: For base 192.168.10.0/26, equal split into four /28s shows 14 usable hosts per child and clear broadcast boundaries; VLSM runs respect requested host counts and flag over-allocation when the base block is insufficient.

## **8. Discussion and Analysis**

Compared with typical calculators, the simulator emphasises explanation and planning rather than only final ranges. Key challenges included handling alignment for VLSM allocations, edge cases for /31 and /32, and ensuring that tables remain readable. Performance is more than adequate for lab use; limits appear only with very large lists of networks.

Limitations: IPv4 only; no persistent storage; minimal security model; diagrams are intentionally lightweight to prioritise clarity.

## **9. Conclusion & Future Scope**

The project delivers a student-friendly CIDR planner that connects textbook theory to realistic network planning. Future work can add IPv6 support, CSV/PDF export, role-based access for class submissions, and richer diagrams (tree views of VLSM allocations).

## **10. References**

- RFC 4632 — Classless Inter-Domain Routing (CIDR) address assignment and aggregation plan: <https://www.rfc-editor.org/rfc/rfc4632>
- RFC 1518 — Architecture for IP address allocation with CIDR (background, historic): <https://www.rfc-editor.org/rfc/rfc1518>
- RFC 950 — Internet Standard Subnetting Procedure (classful-era foundations): <https://www.rfc-editor.org/rfc/rfc950>
- RFC 3021 — Using 31-bit prefixes on point-to-point links (/31 usability): <https://www.rfc-editor.org/rfc/rfc3021>

- RFC 791 — Internet Protocol (IPv4) specification: <https://www.rfc-editor.org/rfc/rfc791>
- Python `ipaddress` module docs — standards-compliant IPv4/CIDR arithmetic API: <https://docs.python.org/3/library/ipaddress.html>
- Streamlit Documentation — quick UI framework used to build the simulator: <https://docs.streamlit.io/>
- Cisco Guide to VLSM — Variable Length Subnet Masking concepts and examples: <https://www.cisco.com/c/en/us/support/docs/ip/routing-information-protocol-rip/13788-3.html>
- APNIC: What is CIDR? — concise operator-level explanation with examples: <https://www.apnic.net/get-ip/faqs/cidr/>
- Kurose & Ross, *Computer Networking: A Top-Down Approach* — core textbook coverage of IP addressing/CIDR (Pearson page): <https://www.pearson.com/en-us/subject-catalog/p/computer-networking-a-top-down-approach/P200000006512/>

## 11. Appendix

**Github Link :** <https://github.com/rifata2021-design/13cidr>