Project ID	Project Title	Tт Description	TA	Max Team Size
1	Priority-Based Routing for Quality of Service in Quantum Networks	**Description** – Implement priority-based routing for quantum networks by extending path selection algorithms to account for differentiated QoS among users. **Tools** – Quantum network simulators (e.g., SquidASM, NetSquid, SeQUeNCe), Python, routing algorithms with priority handling. **Expected Outcomes** – Demonstration of QoS-aware quantum routing, providing differentiated performance for high-priority users. **Reference** – https://doi.org/10.1049/qtc2.7000018	Mallika	5
2	Quantum BGP: Online Path Selection through Network Benchmarking	**Description** – Implement and reproduce results of Quantum BGP protocol with online benchmarking-based path selection, ensuring efficient and adaptive routing in quantum networks. **Tools** – Quantum network simulators (SquidASM, NetSquid, SeQUeNCe), Python, routing protocol implementation. **Expected Outcomes** – Validation of Quantum BGP with benchmarking, showing adaptive online path selection that improves routing efficiency in quantum networks. **Reference** – doi: 10.1109/INFOCOM52122.2024.10621359	Mallika	5
3	Understanding Host Network Stack Overheads	**Description** – Gather understanding of Network stack. Analyze and reproduce the results published in the paper and extend the evaluation with various other scenarios. **Tools** – Wireshark/tcpdump (packet analysis), Python/Java (traffic generation and automation), Linux networking utilities. **Expected Outcomes** – Validation of published results, along with extended evaluation that highlights the impact of different network conditions and configurations on performance. **Reference** – https://doi.org/10.1145/3452296.3472888	Mallika	5
4	SDN based Network Automation for Traffic Management	**Description** – Create a SDN Northbound Application that can automatically reconfigure the network routing based on real-time traffic patterns. **Tools –** ONOS, Java (Northbound application development), Mininet-IP (network emulation), **Expected Outcomes** – Validation of adaptive routing through ONOS, demonstrating reduced congestion and improved throughput/latency under varying traffic loads compared to static routing. **Reference** – https://opennetworking.org/onos/	Mallika	5
5	Cross-Simulator Exploration of Quantum Network Applications	**Description –** Implement a common quantum network application (e.g., routing, QKD, entanglement swapping, teleportation) across multiple simulators to explore and compare their features. The goal is to demonstrate how the same application behaves differently across simulators and to analyze their strengths, limitations, and suitable use-cases. **Tools –** SquidASM, NetSquid, SeQUeNCe, SimulaQron, or other relevant quantum network simulators. **Expected Outcomes –** Side-by-side implementation across at least three simulators, comparative results on fidelity/latency/scalability/usability, documentation of strengths and limitations, and a reusable benchmark package for future exploration. **Reference –** https://github.com/sequence-toolbox/SeQUeNCe https://github.com/SoftwareQuTech/SimulaQron https://github.com/QuTech-Delft/squidasm https://netsquid.org/	Mallika	5
6	Simulation Platform for Distributed Quantum Computation	**Description –** Build a simulator for distributed quantum computing by interconnecting quantum processors through a network simulation framework. The objective is to model realistic quantum processors and their interactions over quantum networks, enabling the study of distributed quantum computation. **Tools –** Qiskit, SquidASM, NetSquid, SeQUeNCe, SimulaQron, or other relevant quantum simulators. **Expected Outcomes –** A functional simulation of distributed quantum computing with multiple networked quantum processors, demonstration of example tasks such as entanglement-based communication or distributed algorithms, and comprehensive documentation of the chosen simulators and implementation process. **Reference –** https://paragon.cs.northwestern.edu/papers/2020-MSThesis-NU-Choudhary.pdf	Mallika	5
7	C++/Python Coding Ground	**Description –** Develop an interactive online platform to host competitive programming, coding challenges, and technical assessments. The platform should support multiple roles (assessor and assessee), allow multiple users to attempt problems simultaneously, and provide features to build, compile, and run code online. **Tools –** Python (Django), React.js, Docker sandbox, PostgreSQL **Expected Outcomes –** A fully functional coding platform with role-based access, support for multiple users, problem hosting and evaluation features, real-time code execution and compilation, and an extensible framework comparable to platforms like HackerRank or CodeChef. **Reference –** https://www.hackerrank.com/	Mallika	4
8	Ultra Fast Packet Crafter/Ping utility: Blitzping.	**Description:** This project ports Blitzping to DPDK for high-speed packet processing and adds hop-by-hop network tracing functionality. You need modify the original code to use DPDK's direct memory access, implement TTL-based traceroute with precise timing measurements, and build a simple GUI to display network paths in real-time. The enhanced tool will demonstrate performance improvements over standard networking utilities through hardware-accelerated packet handling. **Tools:** DPDK, Intel PMD drivers, Hugepages, NUMA libraries,Mininet, Raw sockets, Libpcap, blitzping, iperf3, tcpdump, GCC/Clang. **Expected Outcomes:** Learn how to speed up packet processing using DPDK by building a high-performance network system. Explore network behavior by implementing hop-by-hop tracing with precise timing, and compare the performance of DPDK-based tools with traditional networking applications. **Reference:** https://doi.org/10.1016/j.comcom.2020.07.040 Extend the opensource blitzping to provide custom functionalities: 1. Port blitzping/hping3 to run on DPDK. 2. Build hop-by-hop node and delay information with minimal GUI https://github.com/Thraetaona/Blitzping	Naveen	5
9	P4-Based Programmable Data Plane	**Project Description:** This project designs a custom packet processing system using the P4 programming language, combining IPv6 Segment Routing (SRv6) for flexible source-based routing and In-band Network Telemetry (INT) to capture real-time packet data such as latency and queue occupancy. Its performance will be compared against a traditional software router to evaluate speed, efficiency, and overhead, highlighting how programmable hardware can enhance routing, monitoring, and overall network performance. **Tools/Technologies:** P4 Programming BMv2 Emulator & Tofino Model Wireshark, iperf3, Mininet. **Expected Outcomes:** P4 programming by building a custom data plane for SRv6 routing. Understand IPv6 Segment Routing (SRv6) through and In-band Network Telemetry (INT). Benchmark and compare the performance of P4-based pipelines against traditional routing approaches.	Naveen	5
10	Network File System(NFS)	**Project Description:** Implements a Distributed Network File System (NFS) that enables clients to remotely access and manage files over a network. The system follows a client–server model, supports multiple concurrent clients, and integrates features such as client-side caching with invalidation and replication for fault tolerance. Performance will be benchmarked against local file systems to evaluate latency, throughput, and scalability. **Tools/Technologies:** **Programming:** C / C++ / Python / Go **Networking:** TCP/UDP, RPC or gRPC **Network Analysis**: Wireshark, iperf3 **Simulation:** Mininet / Docker. **Expected Outcomes:** The project will provide insights into how file system operations behave over networks, the impact of caching and replication on latency and throughput, and how multiple clients affect scalability. It will also build practical skills in client–server communication, performance benchmarking.	Naveen	5
11	Network Telemetry Framework	**Project Description:** Develop a standardized framework to collect, transport, and visualize network performance metrics such as bandwidth usage and latency from routers and switches in real time. The system gathers metrics at regular intervals, streams them to a central backend, and processes the data for visualization and to monitor network load, identify latency issues, and quickly respond to performance problems. **Tools/Technologies:** Python, gRPC, SNMP, REST APIs, OpenTelemetry, Grafana, or other telemetry protocols; reference: [RFC 9232]. **Expected Outcomes:** Real-time monitoring dashboards that display live bandwidth utilization and latency trends across devices. Threshold-based alerting for performance issues (e.g., high bandwidth usage or latency spikes). Historical data access and export through REST APIs for reporting and integration. A modular and scalable system that can be extended to support more devices or metrics without affecting network performance.	Ekta	5
12	Network Congestion Control Simulator	**Project Description:** Create an interactive simulation platform that demonstrates how different TCP congestion control algorithms work under varying network conditions, such as bandwidth, delay, and packet loss. It provides visualization of throughput, congestion window changes, and retransmission behavior. **Tools/Technologies:** Python, JavaScript, React for GUI; Matplotlib for graphs; simulation frameworks like ns-3 or custom-built network models. **Expected Outcomes:** Configurable simulation environment for TCP algorithms (e.g., Reno, Cubic, BBR). Real-time graphs showing throughput and congestion window evolution. Animated timeline depicting packet traces and state transitions. Educational tool for networking students and instructors.	Ekta	2
13	Interactive DNS Query & Resolution Visualizer	**Project Description:** Develop a tool to simulate and visualize the DNS resolution process. Users can input domain names and query types, and the system will display the sequence of steps involved in resolving the query, including caches, resolvers, and authoritative servers. **Tools/Technologies:** Python, JavaScript, React, D3.js, Node.js for backend; DNS libraries like dnspython; visualization frameworks like D3.js. **Expected Outcomes:** Step-by-step graphical representation of DNS resolution. Interactive exploration of caching, DoH, DoT, and DNSSEC. Simulation of failures like server downtime or DNS poisoning. Educational resource for learning DNS internals and security considerations.	Ekta	2

14	Per-Process Bandwidth Tracker using eBPF	Build a real-time network usage tracker using eBPF to measure send/receive bandwidth per process. Support per-remote IP tracking, protocol filtering (TCP/UDP), and historical usage storage for reporting. **Tools** Linux eBPF (bcc / libbpf) Go or Python (user-space program) SQLite / Prometheus (storage) CLI or simple Web UI **Expected Outcomes** Hands-on eBPF networking experience Real-time per-process + per-IP bandwidth monitoring Protocol-based filtering (TCP/UDP) Historical usage reports	Yasir	4
15	Packet Monitor	Build an eBPF-based tool to trace where and why packets are dropped in the Linux networking stack. Use tracepoints like net_dev_xmit, kfree_skb, and skb_drop to capture drop events and provide insights. **Tools** Linux eBPF (tracepoints) bcc / libbpf Go or Python (user-space program) CLI or simple dashboard for visualization **Expected Outcomes** Learn eBPF tracing for networking internals Identify precise packet drop points and reasons Provide real-time drop monitoring for debugging and performance tuning	Yasir	4
16	Modifying Southbound Communication protocol in SDN: TCP to UDP for (RYU–OVS)	Modify the Ryu SDN controller and Open vSwitch (OVS) to use UDP instead of TCP for OpenFlow southbound communication. Implement basic reliability (acknowledgments/retransmissions) or operate in “best-effort” mode to study trade-offs in latency, performance, and failure handling. **Tools** Ryu SDN Controller (Python) Open vSwitch (C codebase) Wireshark/tcpdump for packet analysis Mininet for testing **Expected Outcomes** Understand Ryu–OVS communication internals Modify transport layer for OpenFlow messages Compare TCP vs. UDP trade-offs (latency, reliability) Prototype “UDP-based OpenFlow” for experimentation	Yasir	5
17	Energy Monitor usign e-Bpf Real-time Visualization	Attach eBPF programs to kernel events (process scheduling, context switches, network I/O) to collect fine-grained resource usage metrics (time, CPU cycles, etc.). Combine these with power models (RAPL counters, CPU frequency scaling) to estimate per-process energy consumption. Stream the data to a visualization layer (web or CLI) to display real-time usage and trends. **Tools** Linux eBPF (tracepoints, perf events) RAPL counters / CPU frequency scaling data Go or Python (collector + dashboard backend) Web UI or CLI dashboard for visualization **Expected Outcomes** Learn eBPF tracing for CPU and I/O events Estimate real-time per-process energy consumption Visualize energy usage and trends per process Prototype “per-process energy monitor” for Linux Dockerize the tool for easy deployment and distribution.	Yasir	5
18	Securing Ryu–OVS Southbound Communication with Post-Quantum Cryptography	Integrate PQC algorithms (e.g., Kyber, Dilithium from NIST PQC standards) into the Ryu–OVS southbound channel. Replace or augment classical TLS key exchange and authentication with PQC primitives to create a quantum-resistant SDN control plane. Evaluate performance and overhead compared to standard TLS. **Tools** Ryu SDN Controller (Python) Open vSwitch (C codebase) PQC libraries (e.g., liboqs, Open Quantum Safe, pqcrypto) Mininet for SDN testbed Wireshark/tcpdump for traffic verification **Expected Outcomes** Understand Ryu–OVS southbound TLS communication internals Implement PQC-secured OpenFlow channel Compare performance (latency, throughput) with classical TLS Prototype a quantum-resistant SDN southbound communication channel More Details.	Yasir	6
19	Exploring Model Context Protocol (MCP) for Networked AI Applications	**Project Description:** This project involves building a Model Context Protocol (MCP)-compliant server that exposes network diagnostic tools (e.g., DNS lookup, traceroute, ping) as callable functions. The core focus is connecting this server to an MCP-enabled client to demonstrate how natural language commands can execute these tools and return structured data, enabling AI-assisted network analysis and troubleshooting. Use Http SSL for connection and deploy server on cloud. **Tools/Technologies:** MCP frameworks (Anthropic MCP SDK, FastMCP), server framework (FastAPI, Python), network diagnostic tools (dig, traceroute, ping), clients (Claude Desktop, custom terminal client), development tools (Docker, Git). **Expected Outcomes:** A functional MCP server with API endpoints for core network operations A working client-server integration demonstrating natural language-driven tool invocation Demonstration of structured data exchange between AI systems and network tools Analysis of MCP's benefits for standardizing AI-tool interactions and improving contextual understanding Documentation and examples showcasing practical applications for network management	Harsh	4
20	Intelligent Network Configuration Assistant using MCP and Rule-Based Automation	**Project Description:** This project builds an intelligent assistant using MCP that helps network administrators configure devices securely and efficiently. It interprets user commands and applies configurations like firewall rules and bandwidth limits based on predefined policies and network context. **Tools/Technologies:** MCP frameworks (FastMCP, Anthropic MCP SDK), network tools (`iptables`, `tc`), scripting (Python, Bash), configuration files (YAML/JSON), monitoring tools (`ping`, `iperf3`), Docker for deployment. **Expected Outcomes:** Apply network configurations from user commands Validate changes through monitoring tools Provide structured, rule-based automation for secure and optimized settings **Deliverables**: Working assistant with example commands, rule files, deployment scripts, test cases, and a final report explaining setup, use cases, and validation results.	Harsh	5
21	DNS over HTTPS (DoH) vs DNS over TLS (DoT): Performance & Privacy	**Project Description:** This project sets up a test environment to compare DNS-over-HTTPS (DoH) and DNS-over-TLS (DoT) with traditional DNS. It focuses on measuring differences in latency, bandwidth overhead, and privacy by analyzing how resistant each method is to eavesdropping and packet inspection. **Tools/Technologies:** Testbed setup with DNS servers supporting DoH, DoT, and plain DNS Network analysis tools (Wireshark, tcpdump) Web browsers and benchmarking tools for page load measurement Traffic generation tools (iperf3) **Expected Outcomes:** Measure lookup and page load latency Analyze bandwidth overhead of encrypted DNS Evaluate privacy improvements through packet inspection	Harsh	5
22	Multi-Path QUIC (MP-QUIC) vs MPTCP: Performance and Reliability Evaluation	**Project Description:** Study and compare MP-QUIC and MPTCP protocols for applications like video streaming, file transfer, and VoIP, focusing on throughput, latency, and connection stability. **Tools/Technologies:** QUIC implementation: quic-go, Cloudflare’s quiche MPTCP: Linux kernel MPTCP, Multipath-enabled iproute2 Network emulation: NetEm, Mininet Monitoring: iperf3, Wireshark **Expected Outcomes:** Measure throughput, latency, and reliability under different network conditions Analyze how each protocol handles packet loss and congestion Gain practical experience configuring multipath protocols Provide recommendations for protocol use based on application needs	Harsh	5

23	SLA-Aware Autoscaler (Prototype with Queue Depth Metric)	**Project Description**: Build a Kubernetes-based custom autoscaler for FlexRIC xApps that scales pods based on queue depth, not just CPU. Instrument a sample xApp in FlexRIC to export queue depth (number of pending E2 messages). Collect metrics with Prometheus. Write a Kubernetes controller (Python Go, or use keda) that scales pods when queue depth > threshold. Compare performance with default HPA (CPU-based). **Tools:** FlexRIC/ ONF-SDRAN (near-RT RIC SDK), Kubernetes + Prometheus, Python/Go for controller **Expected Outcome:** Working autoscaler that reacts faster than CPU-based scaling. Graphs showing reduced SLA violations (lower queuing delay).	Ayushman	6
24	Extending Kubernetes Self-Healing for Networking Failure	**Project Description:**Kubernetes provides built-in self-healing features like restarting failed pods or rescheduling workloads when nodes go down. However, it does not automatically recover from networking-related failures such as CNI plugin crashes, CoreDNS issues, or broken NetworkPolicies. In this project, a custom Kubernetes operator/controller will be developed to detect and fix such failures. The operator will monitor: CNI plugin health (Calico/Flannel pods) DNS health (CoreDNS latency and failures) Pod-to-pod connectivity (via periodic probes) NetworkPolicy enforcement (detect unreachable services) When a problem is detected, the operator will apply automated remediation e.g., restarting pods, reapplying NetworkPolicies, or switching to a backup DNS configuration.. **Tools**: Kubernetes (Minikube/KIND) Prometheus for health metrics + Alertmanager Python (client-go) or Go (Kubebuilder) for operator development Calico/Flannel as CNI plugin Loki/Grafana for log monitoring (optional) **Expected Learning Outcomes:** Understand limits of Kubernetes' built-in self-healing Learn how to extend Kubernetes with custom controllers Debug and recover networking failures in clusters Gain hands-on experience with Kubernetes monitoring and networking stack	Ayushman	6
25	Dynamic UE Handover Simulation using srsRAN	**Project Description**: Mobility is a core challenge in 5G networks. In this project, a UE (emulated) will move between two gNBs in an srsRAN setup. Students will simulate handover events and analyze key network performance metrics such as handover delay, packet loss, and signaling overhead. The project helps students understand how mobility is managed in real 5G networks and how service continuity is maintained. **Tools**: srsRAN (gNB + UE emulation) URRANSIM (mobility and handover simulation) **Learning Outcome:** Measured handover delay and packet loss during UE movement Signaling overhead analysis during handover Visualization of handover events and network continuity Understand UE mobility and handover mechanisms in 5G Gain hands-on experience with srsRAN and mobility emulation Quantify network performance metrics under mobility conditions	Ayushman	4
26	Simulation of 5G NTN (Non-Terrestrial Networks) Using Open-Source Tools	**Project Description**: This project studies the performance of 5G networks over satellite-based links (GEO and LEO). Students will emulate satellite-induced delay and Doppler effects on 5G traffic and validate results using MATLAB modeling. The aim is to compare terrestrial, GEO, and LEO links in terms of latency, throughput, and reliability, highlighting the challenges of non-terrestrial communication in 5G networks. **Tools**: srsRAN → gNB and UE emulation Open5GS → 5G Core network GNU Radio or Linux tc/netem → To emulate satellite delay and Doppler effects Wireshark + iperf3 → Traffic capture and performance measurement MATLAB → Analytical modeling of GEO and LEO link delay and Doppler **Learning Outcome:** Measured latency, throughput, and reliability for terrestrial, GEO, and LEO links Doppler effect analysis on different non-terrestrial platforms Comparison graphs showing performance degradation due to satellite links Understand non-terrestrial network challenges in 5G Gain hands-on experience with satellite link emulation and analysis Learn how Doppler shift and propagation delay affect network performance	Ayushman	6
27	Doppler Shift Analysis in LEO, MEO, and HAPS Links	**Project Description**: Mobility and relative motion introduce Doppler shifts in non-terrestrial wireless links, which can impact 5G/6G communications. In this project, students will analyze and simulate Doppler shifts for LEO satellites, MEO satellites, and HAPS(High Altitute Platform Stations). They will calculate Doppler frequency offsets based on altitude, velocity, and carrier frequency, and study the effect on simple communication signals. Students will compare terrestrial, LEO, MEO, and HAPS links in terms of frequency offsets, latency, and potential symbol errors, and discuss mitigation strategies such as frequency tracking or adaptive equalization. **Tools**: MATLAB / Python → For Doppler calculation, plotting Doppler vs time, and analytical modeling GNU Radio / srsRAN (optional) → To simulate communication links under Doppler offsets GMAT (optional) → To generate realistic satellite orbits and velocities Wireshark / iperf3 → To capture and measure network traffic if simulating signal-level effects **Learning Outcomes:** Graphs showing Doppler shift vs time for LEO, MEO, and HAPS platforms Maximum Doppler shift comparison for different altitudes and carrier frequencies Optional BER vs SNR plots for a simple OFDM/LTE link under Doppler Analysis of Doppler mitigation techniques like frequency tracking or adaptive equalization Understand the effect of relative motion on wireless signals in NTN environments Hands-on experience with simulating satellite/HAPS links using MATLAB/Python and/or GNU Radio Learn how Doppler affects network performance (latency, throughput, error rate) Gain experience in modeling and analyzing non-terrestrial wireless networks, bridging theory and simulation	Ayushman	6
28	SmartGuard	**Project Description:** Develope a centralized monitoring system that collects and analyzes real-time environmental data such as temperature, light intensity, humidity, and air quality from classrooms and labs. Using a network of simulated sensor clients (via Mininet), data is transmitted to a central server where a dashboard presents detailed insights. The dashboard provides heatmaps to visualize real-time conditions across rooms, time-series graphs for tracking historical trends, and anomaly detection alerts for cases such as high CO_2 levels, low light, or overheating. This system helps monitor classroom environments efficiently while also testing scalability and network performance in a simulated setup. Use standard protocols like MQTT. **Tools**: Mininet, MQTT Protocol, Python/Node.js for backend data collection, InfluxDB/TimescaleDB for storing time-series data. Grafana/Plotly Dash for dashboards and heatmaps. **Expected Outcomes:** Students build a centralized dashboard and learn about MQTT protocol for sensors and build a scalable architecture that can take data from let say around 100 clients.	Naveen	4
29	Traffic Prediction + Energy-Aware Link Activation in SDN	**Project Description**: Develop a simulation prototype that reduces energy consumption in a 5G backhaul by predicting per-link traffic and dynamically putting under-utilized links to sleep without hurting throughput. Using historical 4G/5G traffic traces, a time-series ML model (e.g., LSTM) predicts per-link load for each time slot. An SDN controller (Ryu or ONOS) in a Mininet-emulated backhaul topology uses those predictions to keep only the minimum set of active links required to satisfy predicted demand plus a redundancy margin, and moves other links to a low-power "sleep" state. The system measures energy saved, throughput maintained, and changes in link utilization. **Tools**: Mininet (network emulation), SDN controller Ryu or ONOS, Traffic datasets: public 4G/5G flow/trace datasets, Measurement/visualization Matplotlib / Grafana / Plotly **Expected Outcome:** Student build the prototype and analyze the metrices like % Energy Saved, % Throughput Maintained, Latency / Packet loss impact	Yasir	5
30	eBPF/OS - System Intelligence	**Project Description**: Build an eBPF-based monitor that watches a small set of syscalls often used in privilege escalation (execve, setuid/setgid, ptrace) and streams lightweight event records to a user-space agent. The agent applies simple rules and anomaly checks, raises alerts to a remote server, and can optionally trigger containment actions (e.g., kill or suspend a process). The aim is fast, low-overhead detection and basic mitigation with useful forensic context. **Tools / Technologies** eBPF toolchain: clang/llvm, bpftool, libbpf (or BCC for faster prototyping) User-space agent: Python (async) or Go to read ring-buffer events and send alerts Transport: HTTPS/gRPC to remote alert collector (or simple POST) Dashboard/storage: Elastic/InfluxDB + Grafana or simple Flask + SQLite for logs Testing: VMs/containers, strace/auditd to validate behavior **Expected Outcomes** Low-overhead eBPF probes that capture target syscalls. Real-time alert pipeline to a remote collector. Basic anomaly detection rules with measurable true/false positive observations on test scenarios. Forensic logs (process, parent, timestamp, binary ID) to investigate incidents.	Sameer	6

31	DNS Resolver using AF_XDP	**Project Description**: This project implements a DNS resolver that sends and receives DNS packets using an AF_XDP (XSK) socket instead of the usual UDP/TCP socket. An XDP program attached to the NIC redirects DNS traffic (UDP dst port 53) to an AF_XDP socket; a user-space resolver crafts raw Ethernet/IPv4/UDP/DNS frames, transmits them via the XSK TX ring, and receives responses on the XSK RX ring. The project demonstrates low-latency, high-throughput packet I/O, zero-copy packet handling tradeoffs, and the extra plumbing required (UMEM, rings, descriptor management) compared to conventional sockets. **Tools / Technologies** Linux kernel AF_XDP / XDP support (recent kernel) clang/llvm (for compiling the XDP BPF program) libbpf / libxsk (or equivalent XSK helper code) for user-space XSK setup C (user-space resolver) for packet crafting and XSK ring handling bpftool / xdp-loader (to load/attach the XDP program) tcpdump / Wireshark / pkt-gen (for validation and packet inspection) **Expected Outcome:** Comparison vs standard UDP sockets: measurements of latency, throughput, and CPU overhead showing AF_XDP tradeoffs (zero-copy performance benefits vs implementation complexity).	Ayushman	4
32	Digital Twins for Large Scale Data Center Networks	**Project Description**: In this project, you will explore and simulate the networking infrastructure of a data center using NVIDIA Air (Digital Twin). You will create a digital twin model of a data center network, focusing on key components such as routers, switches, servers, and links. The goal is to simulate and visualize network traffic, latency, bandwidth usage, and failure scenarios to optimize and predict network behavior in the data center. You can extend and write python code to simulate and build any networking scenario like, use of machine learning to predict network traffic patterns and automatically adjust configurations based on predicted loads or simulate security aspects like DDoS attacks, firewall configurations, or encrypted traffic analysis into the network model. **Tools / Technologies:** NVIDIA Air Access (Register for a Developer Account) Python **Expected Outcome:** Learn about DigitalTwins and develop a fully functional DigitalTwin Model for network simulations. Demonstrate the Digital Twin and insights on traffic monitoring, prediction or security analysis.	Sameer	2
33	SoNIC Application for Traffic Engineering	**Project Description**: Software for Open Networking in the Cloud (SONiC) is an open source network operating system (NOS) based on Linux that runs on switches from multiple vendors and ASICs. In this project, you will work on the SoNIC NOS and build a traffic monitoring system using the SDN controller that monitors network load and adjusts the paths taken by data packets to optimize for latency, bandwidth, or fault tolerance. **Tools/Technologies**: SoNiC, OpenFlow, ONOS Controller, Packet/Flow Generators like TRex. **Expected Outcome**: Demonstrate a dynamic network where traffic is continuously monitored and optimized to avoid congestion and maximize throughput. [optional] You can use the programmable switch to capture network traffic statistics (packet counts, bandwidth utilization, etc.), and build a system that analyzes traffic patterns and alerts administrators when unusual traffic behavior or potential security issues (e.g., DDoS attacks, abnormal traffic spikes) are detected.	Sameer	3 + 1
34	Efficient Protocol Analysis using DPU	**Project Description:**This project implements an inline protocol security monitor for PFCP (control plane, N4 interface) and GTP-U (user plane, N3 interface) traffic using a Data Processing Unit (DPU). The DPU is inserted between SMF–UPF (for PFCP) and gNB–UPF (for GTP-U), where it parses live packets at line rate. A modular parser extracts key fields (SEIDs, PDR/FAR in PFCP; TEID, sequence numbers in GTP-U), while a detection engine checks for known anomalies such as session hijacking, malformed rules, TEID guessing, and tunnel hijacking. Malicious traffic is dropped inline, while metadata is exported to a telemetry collector. The project demonstrates low-latency protocol parsing on DPUs and quantifies CPU offload benefits compared to CPU-only monitoring. **Tools / Technologies** NVIDIA BlueField DPU (BF-2) SDK DPDK / eBPF/XDP for fast packet parsing and filtering C (DPU user-space applications) for protocol parsing and rule engine Free5GC or Open5GS (Core) to generate PFCP traffic srsRAN or OAI (RAN) to generate GTP-U tunnels Wireshark / tcpreplay / Scapy for packet injection, attack simulation, and validation **Expected Outcome** Working PFCP and GTP-U protocol parsers on the DPU Inline detection of at least 3 attack types: PFCP session hijacking, malformed PDR/FAR injection, GTP-U TEID hijacking Measurement of latency overhead and CPU utilization with/without DPU offload Evidence that DPU-based monitoring achieves line-rate packet inspection with lower host CPU load compared to CPU-only monitoring	Ayushman	5
35	Network Monitoring: OpenTelemetry vs SNMP	**Project Description**: compare and integrate two major observability tools: OpenTelemetry and Simple Network Management Protocol (SNMP), to monitor and observe the performance and health of a network system. You will explore how both tools collect, process, and visualize performance metrics, and ultimately, how they can be used together or separately to improve network monitoring. **Tools and Technologies**: OpenTelemetry SDK and Collector, SNMP v2 or v3. Prometheus and Grafana for monitoring frontend/dashboard. Netwok emulator/simulator like Mininet and GNS3. **Expected Outcome**: Understanding of OpenTelemetry and SNMP for network monitoring.Real-time visualization of both application performance and network device metrics. Comparative analysis of the two tools in terms of data collection, integration complexity, and visualization.	Sameer	4
36	Grade Management Tool	**Description** – Develop a Grade Management Tool that automates the grading process, i.e. streamline the process of managing and viewing academic grades. The system allows instructors to populate a grading matrix for their respective courses, specifying the weightage of various assessments such as Quizzes, Assignments, Exams, and Attendance. Instructors can then upload student scores for each assessment. Students can securely select their courses and view their grades, which will be updated as instructors enter new scores. This system aims to enhance efficiency, improve the grading process, and maintain privacy in grade sharing. **Features:** Instructor Dashboard to create and update courses, set the grade matrix (components like #Quizzes, Assignments, Exams,Attendance, etc wtih respective weightage(%)) and upload the grades for respective components. Student Dashboard - to facilitate students to register for the courses and view the grades achieved in their respective courses. Support role based access with distinct roles for Student/Instructor Login. Optional: Automate grade calcuation based on the entered grades. Display of histogram and heatmap mode for grade distributions for the class vs individual student grades. **Tools** – HTML/CSS, Javascript. ReactJS/VueJS, (Choose any relevant frontend and backend tool) Database (SQLite, MySQL or MongoDB) for storing user credentials and course list, and course grade matrix.	Sameer	6
37	Exam APP (IoS support)	**Description**: Develop an iOS application that runs in kiosk mode, ensuring that the device is fully locked down to the app with restricted system access and communication functions. **More Details:** [Document]	Sameer	4
38	eTRAN: Extensible Kernel Transport with eBPF	**Description** – Gather understanding of Network stack. Analyze and reproduce the results published in the paper and extend the evaluation with various other scenarios. **Tools** – eBPF, eTRAN, Linux networking utilities. **Expected Outcomes** – Validation of published results, along with extended evaluation that highlights the impact of different network conditions and configurations on performance. **Reference** – https://www.usenix.org/system/files/nsdi25-chen-zhongjie.pdf	Sameer	4
39	User Access Management Tool for Networked Devices Based on Allowed Time Slots	**Description**: The User Access Management Tool is a networking tool designed to control and manage user logins on devices (both local and remote) based on time-slot restrictions. The tool allows administrators to set specific time frames during which users can access devices. Access will be granted or denied based on the configured time slots, providing enhanced security and better control over device usage. This can be particularly useful for managing shared devices in an organization, remote servers, or IoT devices in a restricted environment. The tool will feature a web interface (for ease of use) where administrators can configure allowed login times for each user, and a backend that enforces these time-based access restrictions on the devices. Tools – HTML/CSS, Javascript. ReactJS/VueJS, (Choose any relevant frontend and backend tool) Database (SQLite, MySQL or MongoDB) for storing user credentials and login restrictions (time slots). Networking and OS tools: Linux/Windows CLI: For executing remote login commands (SSH/RDP). Systemd (Linux) or Task Scheduler (Windows): For enforcing login time restrictions on devices. Cron Jobs (Linux) or Scheduled Tasks (Windows): For managing scheduled access rules.	Sameer	6