Project Lifecycle Stage	**Arushi Mangala (33.33%)**	**Mrudhula Lokesh (33.33%)**	**Ragha Yalamarthy (33.33%)**
Idea Generation	Recommended Fault-Tolerant Architecture using Kafka Replication.	Suggested Geospatial Targeting for delivering location-based alerts.	Proposed the Kafka-Powered Pub/Sub Model for large-scale alerts.
Requirements Analysis	Selected technology stack including Kafka, WebSockets, and cloud deployment.	Identified non-functional requirements such as fault tolerance, latency, and scalability.	Defined functional requirements like publisher, consumer, and message broker interactions.
System Design	Designed data flow for alert distribution from producers to subscribers.	Created the user authentication and security mechanisms.	Designed the microservices architecture for producers, brokers, and consumers.
Backend Development	Integrated WebSocket communication between frontend and backend.	Worked on Kafka consumers that subscribe and process disaster alerts.	Developed the Kafka producers responsible for publishing alerts.
Frontend Development	Integrated frontend with WebSocket server for bidirectional communication.	Implemented real-time alert display using WebSocket.	Developed the subscription UI components for selecting locations.
Integration Testing	Performed end-to-end integration testing across all system components.	Conducted frontend testing, verifying UI updates for incoming alerts.	Led backend testing, ensuring proper message flow in Kafka topics.
Deployment	Configured backend and WebSocket server deployment on EC2.	Managed frontend deployment on cloud infrastructure.	Handled Kafka deployment and topic replication settings.
Performance Analysis	Evaluated system scalability by simulating high alert traffic.	Measured frontend responsiveness and WebSocket efficiency.	Analyzed Kafka performance metrics (latency, throughput).
AWS Setup	Setting up EC2 Instances, security groups and volume storage.	Configuring Application Load Balancer and target groups to connect to the EC2 instances and route traffic accordingly. Monitoring health checks.	Setting up AWS CLI to connect to the AWS services and deploy our Docker containers.
Report Writing	Covered fault tolerance, scalability, and concurrency topics.	Documented security mechanisms and system architecture.	Wrote Kafka and message processing sections in the final report.
Final Presentation	Ensured clarity in explanations and polished the presentation.	Designed visuals for frontend architecture and user interactions.	Created slides explaining Kafka Pub/Sub model and message flow.