PROJECT TITLE – 4

LAMBDA - Serverless Function

General Guidelines:

- Maintain a **Version Control** system with frequent commits to the GitHub repository for easier tracking of progress.
- Write clear documentation for each task completed, including design decisions, issues faced, and their solutions.
- Regularly update the CI/CD pipeline to ensure that the system is well-tested and deployable.
- Weekly_progress according to the assigned problem statement must be pushed to the repo, which will be considered during the evaluation.
- Your Team's weekly progress should be shown to the teacher on request.
- Name the GitHub repository with the four SRNs in order and the project title.

Problem Statement:

The objective of this project is to design and implement a serverless function execution platform, similar to AWS Lambda, that enables users to deploy and execute functions on-demand via HTTP requests. The system will support multiple programming languages (Python and JavaScript) and enforce execution constraints such as time limits and resource usage restrictions. To optimize execution, the platform will integrate at least two virtualization technologies, such as Firecracker MicroVMs, Nanos Unikernel, or Docker Containers, leveraging techniques like pre-warmed execution environments and request batching for improved performance. A key feature of the system is a web-based monitoring dashboard that provides real-time insights into function execution metrics, including request volume, response times, error rates, and resource utilization. The backend will handle function deployment, execution, and logging, while a frontend interface will allow users to manage and monitor their functions.

Week 1: Project Setup and Core Infrastructure (10 M)

Objective: : Simple function execution via API in Docker container.

- **Task 1:** Project Planning and Environment Setup.
  - Define project architecture and create system design diagrams
  - Set up development environment (Git repository, CI/CD pipeline)
  - Choose and install required dependencies
  - Create project folder structure
- **Task 2:** Backend API Foundation
  - Implement basic API server (Express/FastAPI)
  - Create database schema for function storage
  - Implement function metadata storage (name, route, language, timeout
  - settings)
  - Create basic CRUD endpoints for function management
- **Task 3:** Your First Virtualization Technology.
  - Set up Docker as the first virtualization technology
  - Create base container images for Python and JavaScript functions
  - Implement function packaging mechanism
  - Build basic execution engine that can run a function inside Docker
  - Implement timeout enforcement make sure this is clearly thought through

Deliverable: A simple working prototype using docker.

Week 2: Enhanced Execution and Second Virtualization Technology (15 M)

Objective: The goal for Week 2 is to enhance the function execution capabilities of the platform by improving the execution engine with more advanced routing, warm-up mechanisms, and error handling. Additionally, this week will introduce a second

virtualization technology alongside Docker, either **Firecracker MicroVMs**, **Nanos Unikernel**, or **gVisor**. This is crucial for comparing performance across different virtualization approaches and choosing the most effective one for handling serverless function executions at scale.

The second virtualization technology's integration will allow the system to support a broader range of deployment environments, which is necessary for scalability and performance optimization. We will also collect and aggregate execution metrics to monitor the function's performance in terms of response times, error rates, and resource utilization.

- **Task 1:** Execution Engine Improvements:
  - Implement request routing to appropriate function containers
  - Add request/response handling and error management
  - Implement function warm-up mechanism , i.e dummy caching and function
  - Create a container pool for improved performance.
- **Task 2:** Second Virtualization Technology.
  - Set up second virtualization technology (Firecracker MicroVMs or Nanos Unikernel),ps: if you find them hard to set up try using gvisor.
  - Create packaging mechanism for the second technology
  - Implement execution engine support for the second technology
  - Compare performance between the two virtualization approaches
- **Task 3:** Metrics Collection.
  - Implement metrics collection for function execution (response time, errors, resources)
  - Create storage mechanism for metrics
  - Implement basic aggregation of metrics

Deliverable: Almost a working system, with metric monitoring and 2 ways of virtualization.

Week 3: Frontend, Monitoring Dashboard, and Integration (15 M)

Objective: Make the system scalable and fault-tolerant.

- **Task 1:** Basic Frontend
  - Create frontend application structure (Streamlit or similar)
  - Implement function deployment interface
  - Create function management views (list, create, update, delete)
- **Task 2:** Monitoring Dashboard
  - Implement metrics visualization components
  - Create dashboard views for individual function performance
  - Implement system-wide statistics view
- **Task 3:** Integration and Polishing
  - Integrate all components (frontend, backend, execution engine)
  - Implement authentication/authorization (if time permits)
  - Conduct end-to-end testing
  - Fix bugs and optimize performance
  - Create documentation for the system

Deliverable: A complete system.

Additional Implementation (Optional):

- Implement auto-scaling based on request load.
- Add support for environment variables in functions.
- Create cost analysis comparing virtualization technologies.
- Add support for additional programming languages.