Project
Serverless Function Execution Platform - Week By week implementation Plan

Week 1: Project Setup and Core Infrastructure

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
- [ ] **Checkpoint**: Repository initialized with basic structure and README

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
- [ ] **Checkpoint**: API can store and retrieve function definitions

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**
- [ ] **Checkpoint**: Simple function execution via API in Docker container :YUPP:

Week 2: Enhanced Execution and Second Virtualization Technology

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 calls
- [ ] Create container pool for improved performance / K8's
- [ ] **Checkpoint**: Functions can be executed reliably with proper request/response handling

Second Virtualization Technology

- [ ] Set up second virtualization technology (Firecracker MicroVMs or Nanos Unikernel),ps: if you find them hard to setup 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
- [ ] **Checkpoint**: Functions can be executed using either virtualization technology

Metrics Collection

- [ ] Implement metrics collection for function execution (response time, errors, resources)
- [ ] Create storage mechanism for metrics
- [ ] Implement basic aggregation of metrics
- [ ] **Checkpoint**: System collects and stores execution metrics

Week 3: Frontend, Monitoring Dashboard, and Integration

Basic Frontend

- [ ] Create frontend application structure (Streamlit or similar)
- [ ] Implement function deployment interface
- [ ] Create function management views (list, create, update, delete)
- [ ] **Checkpoint**: Users can deploy and manage functions through the UI

Monitoring Dashboard

- [ ] Implement metrics visualization components
- [ ] Create dashboard views for individual function performance
- [ ] Implement system-wide statistics view
- [ ] **Checkpoint**: Dashboard displays metrics and statistics

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
- [ ] **Checkpoint**: Complete working system with documentation

Bonus Tasks (if time permits)

- [ ] 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