System Description:

An API backend service to safely compile and execute provided code and optional test cases in text form. Returns compiler and runtime errors as well as the status of test cases if provided.

Functional Requirements:
- EPIC 1:
    - Sandbox: Create a secure, isolated environment to execute C programs safely
- EPIC 2:
    - Execution and Testing:Execute C programs and optional test cases
- EPIC 3:
    - Feedback on the code: Provide actionable feedback to users about their code's execution and test results
- EPIC 4:
    - API Endpoints: Design and implement a robust API that allows other software components to interact with the Execute & Test Service

- EPIC 5:
    - Make the application usable by other people: Ensure the service is user-friendly, scalable, and accessible to other developers and systems.

System Description
- The system takes in text or a text file as input, compiles it, runs it, and returns the output and any other needed information in a log file

Architectural Overview
- API Gateway
- Sandbox manager
- Compiler and Executor
- Result aggregator
- Logging

Non-Functional Requirements
- The process of creating the environment and compiling the code should not take more than 2s
- The user should not be able to have unprivileged access to the device
- Resource Management
- The application should be able to scale
- The service's runtime environment must be secure against the execution of malicious code

- Application execution time should be less than 30 sec

Technologies and Frameworks
- Django/Flask/FastAPI
- Docker
- Python

Define your Minimum Viable Product (MVP)
- Be able to send a code text using the api endpoint, create a container, execute the code, run the test cases, and return the response

Create a Preliminary Roadmap for the Semester
- Familiarize with the technology
- Programatically create a container
- Compile C code
- Compile C code in the container
- Parse the response to create a response message

Possible Containers/Sandboxes
- Docker

System Diagram