ecosmic

SOFTWARE ENGINEERING LEAD
TAKE-HOME ASSIGNMENT

Please note that this take-home exercise is a fictional scenario designed solely to assess your technical and problem-solving skills. While the topic may be loosely inspired by the kind of work we do, it does **not** reflect our actual processes, systems, or client situations. The goal is to evaluate your approach, reasoning, and hands-on capabilities—not to simulate a real-world project.

Goal

Ecosmic is building a **Collision Probability Estimation Service** to support satellite operators in space traffic management.

Objectives of the Assignment

- Design the architecture and create a stub for IaC
- Bootstrap the repository (structure + onboarding docs)
- Implement a basic prototype of the service
- Draft a development roadmap (tasks, priorities, assignments)

Framework

- We expect to onboard:
  - 3 customers in 3 months, each with ~10 satellites and an average of 1 run / day / satellite, with an SLA defining a maximum time-to-result of 1 hour
  - A large customer in month 4 with ~1000 satellites and an average of 5 runs / day / satellite, with an SLA defining a maximum time-to-result of 1 hour
- The service must be **scalable, reliable, observable**, and built with **long-term maintainability** in mind.
- Team composition:
  - You (Tech Lead)
  - Mid-level DevOps
  - Junior Software Engineer (Python/C++)
  - Space Engineer (C++, domain expert)

Project Constraints

- The team should be able to **deploy with confidence**, keeping in mind code quality, basic cybersecurity best practices, and the automation of common processes (CI/CD).
- The architecture should **support long-term maintainability**, with appropriate observability patterns to aid in continuous improvement.
- The system features a core algorithm written in C++ with an average runtime of 10 minutes.
- You must call this algorithm asynchronously from a **Python RESTful API using FastAPI**.
- The inputs and outputs of the C++ algorithm shall be:
    - Inputs: the NORAD ID, the state vector, and the covariance matrix of the two space objects involved, along with the TCA.
    - Outputs: the probability of collision calculated with our algorithm.
- The API endpoint shall take a CCSDS CDM file in KVN as input, parse it and reprocess it through the core algorithm to get the aforementioned inputs.
- The endpoint shall be usable only by customers who own the space object identified as primary in the CCSDS CDM.
    - Primary is defined as the OBJECT1 from the CCSDS CDM spec in the CDM METADATA section.
    - Ownership is represented as a mapping between NORAD ID and the organization the user belongs to. Refer to the OBJECT_DESIGNATOR field in the CDM METADATA section.
- Internal teams must be able to **trigger the execution of the C++ algorithm via CLI**.
- The algorithm shall produce as output a CCSDS CDM file, containing the same information of the one provided by the user. The fields "ORIGINATOR" and "COLLISION_PROBABILITY" shall be updated, namely, with "Ecosmic" and the result of the algorithm. The delivery mechanism is up to you to decide.
- There is no constraint on AuthN, but justify your pick.

What You Must Deliver

Percentage represents how important each point is for our evaluation.

- Architecture diagram of a cloud-native architecture based on AWS, along with a high-level diagram of service components (e.g., C4 model) - 40%
- Infrastructure-as-code stub (tool of your choice) - 10%
- A development roadmap: tasks with priorities, complexity, team assignments (GitHub projects) - 30%
- FastAPI & CLI prototype with minimal logic and test coverage - 20%

How To Deliver

You're expected to work in a private GitHub repository created by you, in which you should add FrancescoFoglia-ecosmic and LorenzoFarinelli-ecosmic as collaborators. Once you're satisfied with your work, please notify us via email to allow us to review your project. You should not push new commits after the deadline agreed on via email.

Useful resources: CCSDS CDM specification: https://ccsds.org/Pubs/508x0b1e2c2.pdf