Scenario

- A Satellite Operator is a company that manages many satellites.
- Consider a system for alerting a satellite operator of a possible collisions on its satellites.
- When there’s a likelihood for a collision in orbit to occur, the satellite operators need to
- be alerted as early as possible.
- After being alerted, they can perform manoeuvres on their satellites to avoid collision.
- They are alerted, for a single collision event, through multiple messages, which are
- received as time approaches the expected collision time.
- A satellite operator can have many thousand satellites
- A satellite can have multiple collision events in a single day

The “Collision Events API” is meant to allow the system to capture collision messages and expose collision warnings regarding a collision event. It’s meant to be used in mostly 3 different scenarios:

- Scenario 1: A front-end website will use the API to retrieve collision warnings.
- Scenario 2: An external system should be able to submit messages about new collision events. An external system can also cancel previously submitted collision messages.
- Scenario 3: A service will query the APIs every minute and issue warnings (email) for the satellite operators.

Exercise

Write a very simplified version of Collisions Event REST API adhering to standards similar to what you do if production ready code.

- For the purpose of simplicity, you don’t need to create the database (you can if you wish). Regardless, make sure you generate enough fake data (in memory or injectable into the database) so that that allows you to test your code.
- For the purpose of simplicity, you don’t need to authorize/authenticate the endpoints as you would otherwise do in a production system.

Functionally, the API should have endpoints for following purposes:

Purpose	**Details**
Return the collision status of warning for all the satellites of an operator (who is using the API).	A satellite should have collision warning issued when: ● A probability of collision of 0.75 or more was received on a collision event message. ● The earliest collision date is in the future. The earliest collision date is the date closer to when a collision may occur (as per the collision date received in collision event message) ● It belongs to the operator invoking the API* The API should return a set of collision event messages matching the conditions above (see payload below).

Accept new collision events messages.	The collision events messages should be accepted only if: ● The message relating to the event hasn't been recorded ● yet (message_id is not present in the database) ● The probability of collision is between [0..1] ● The Collision Date is in the future ● It belongs to the operator invoking the API*
Canceling a collision event message	A collision message should be canceled if: ● It already exists in database ● The collision data is in the future ● It belongs to the operator invoking the API*

For example, an "operator_id" is passed to the invoking API on the headers.

Example payload for a collision event message:

```
{
"message_id": "54321aaabbbccc",
"collision_event_id": "1213aaabbbccc",
"satellite_id": "1999-1232",
"operator_id": "123456789",
"probability_of_collision": 0.5,
"collision_date": "20220312T20510100Z",
"chaser_object_id": "2016-4321"
}
```

Example payload for a collision status for a single satellite:

```
{
"satellite_id": "1999-1232",
"highest_probability_of_collision": 0.5,
"earliest_collision_date": "20220312T20510100Z",
"chaser_object_id": "2016-4321"
}
```

Important things to take into consideration:

- ☐ REST principles are respected if REST API is chosen
- ☐ Deliver the code as close to production as possible, including containerization of the API.
- ☐ If you create a database, please make sure that it’s possible to run it from the container.
- ☐ Share a git repository with your code.
- ☐ Document your code as if it was going to be done in production.
- ☐ Make sure your code is well organized as it would be if run in production
- ☐ Write the unit tests that you think make sense in the context of the exercise.

The main purpose of the exercise is for us to be able to discuss your code and approach focusing on:

- Design considerations and code structure
- Code maintainability
- Documentation
- Tests