Question: Develop a Golang microservice for supporting the space voyagers who are embarking on a journey to study different exoplanets. Exo-planets are the planets outside of our solar systems.

Currently there are only two kinds of exoplanets:

1. **Gas giant : composed of only gaseous compounds**
2. **Terrestrial : earth like planets, a bit more rocky and larger than earth**

Scenario: You are tasked with building a microservice for managing different exoplanets. The microservice should provide the following functionality:

1. **Add an Exoplanet:** Users should be able to add new planet *(belonging to either of the above exoplanet types)* by providing only following properties :
    1. name
    2. description
    3. distance from earth
    4. radius
    5. mass *(will be provided only in case of Terrestrial type of planet)*
    6. type of exoplanet : GasGiant or Terrestrial
2. **List Exoplanets:** Users should be able to retrieve a list of all available exoplanets.
3. **Get Exoplanet by ID:** Users should be able to retrieve information about a specific exoplanet by its unique ID.
4. **Update Exoplanet:** Users should be able to update the details of an existing exoplanet.
5. **Delete Exoplanet:** Users should be able to remove a exoplanet from the catalog.
6. **Fuel estimation:** User should be able to retrieve an overall fuel cost estimation for a trip to any particular exoplanet for given crew capacity. (*Please refer below for fuel estimation formula)*

Fuel estimation to reach an exoplanet can be calculated as :

f = d / (g^2) * c units

d -> distance of exoplanet from earth

g -> gravity of exoplanet

c -> crew capacity (int)

Logic to calculate gravity for each type is as follows :

- **Gas Giant** :

  g = (0.5/r^2)

- **Terrestrial** :

  g = (m/r^2)

m -> mass

r -> radius

Requirements:

- Use the Go programming language to create the microservice.
- Implement a RESTful API for the above functionalities.
- Use any web framework or library of your choice for handling HTTP requests (e.g., Gorilla Mux).
- Persist application data in memory.
- Provide clear and consistent error handling.
- Create a Dockerfile for your microservice.

Constraints :

- $10 < d < 1000$ (light years) : int
- $0.1 < m < 10$ (Earth-Mass unit) : double
- $0.1 < r < 10$ (Earth-radius unit) : double

Evaluation Criteria:

- Correctness of the implementation (Does it meet the specified requirements?).
- Code quality, organization, and readability.
- Proper use of Go language features and best practices.
- Error handling and validation of user inputs.
- Ease of extensibility : How easily can we extend the service when more types of exoplanet are introduced?
- Dockerization of the service.
- Clarity and completeness of the README file.

Bonus (Optional):

- Implement validation for explanets data, ensuring that fields like planet's name, description, distance, radius and mass are provided correctly.
- Add unit tests to ensure the reliability of the service.
- Implement sorting or filtering options for listing planets if time permits. (based on radius or mass)