GreenFlux Smart Charging Assignment The assignment is very simplified version of real requirements for capacity control in Smart Charging.

**Domain model:**

****

*Charge station* – has unique identifier (cannot be changed), name (can be changed), multiple connectors (at least one, but not more than 5).

*Connector* – has numerical identifier unique per charge station with (possible range of values from 1 to 10), Max current in Amps (can be changed) – value greater than zero.

*Group* – has unique identifier (cannot be changed), name(can be changed), c*apacity in Amps* (can be changed) – value greater than zero. Group can contains multiple charge stations.

**Functional requirements:**

**1**. Connector cannot exist in the domain without charge station.

**2.** Max current of existing connector can be changed.

**3.** Charge station cannot exist in the domain without *Group*. Charge station can be only in one group at the same time.

**4.** Only one charge station can be added/removed from a group in one call. **5.** Group can be created, update and removed.

**6.** If a group removed, all charge stations in the group should be removed as well.

7. Capacity of group should always be great or equal to sum of Max current in Amps of connector of all charge stations in the group.

8. Based on previous paragraph, If capacity of a group is not enough when adding a new connector – API should return response with suggestion of which connectors of which charge stations should be removed from the group to free space for the new connector.

a. The suggestion should be made with the most optimal way – should be removed the minimum connectors as possible to free exact amount of capacity for new value. b. If there are multiple equally optimal ways, the endpoint should return multiple suggestions.

**Technical requirements:**

1. Create RESTful ASP.NET Core API according to described requirements.

2. Use any convenient for you database to store data.

3. Think about performance of the solution.

4. Cover your code by necessary in your opinion unit and/or integration tests. 5. Use local Git repository for your code.

6. Nice to have Swagger.

7. Provide a ready-to-run solution (Visual Studio or Visual Studio Code) via GitHub or any other public Git repository.

We mostly judge your assignment on readability (not over-engineered), maintainability (what if we need to extend your solution in the future) and quality control (how code was covered by tests, which tests used for which purposes, not necessary to cover by tests everything).

The result doesn’t need to be a fully finished, as long as it shows your skills as a .NET developer. Please send us back the source code. Preferably via a GitHub URL repository, with a ready-to compile solution.

We appreciate and expect questions!

**Your additional questions can be send by emails:**

Matheus Calache: matheus.calache@greenflux.com

Raman Tsitou: raman.tsitou@greenflux.com

Subject of email should starts with **[assignment]**