Peak Technical Test

Overview: your task is to build a microservice that powers the shopping cart of an ecommerce website. You don't need to hook your service up to a real database, the data can be stored in memory for the purpose of demonstration. However, we also ask for a small write-up, which explains your design choices and architectural decisions for example. This write-up could include some notes on what database you might use for this service in production and why.

The service: the service must have the following capabilities:

- Endpoint for adding an item.
- Endpoint for removing an item.
- Endpoint for clearing the entire basket.
- Endpoint for listing all the items in the basket, must have an ID, a title and a price.
- Bonus Question: End point which takes a list of item ids, their weight in KG, days it will take to deliver item as input parameters. The end point will return list of item ID's which can be fit in a single delivery box with a capacity of 10 kilograms and total sum of delivery days as low as possible from the list sent as input. The endpoint will also return sum of delivery days item that can fit in that box.
    - All items weights are between 1 and 9 KG and always in full kilograms
    - Max weight allowed in the delivery box is 10 KG.
    - Must put item in the box if an item weighs less than remaining capacity.
    - Example: Input
        - ID: Item1, Item2, Item3, Item4, Item5, Item6
        - Weight: 1,8,7,4,3,2
        - Delivery Days: 4,1,2,10,3,5
    - Example: Output (if more than one correct answers, choose any 1)
        - ID: Item 2, Item1
        - Delivery days sum: 5

We will be looking for:

- Architectural decisions in the way your lay out your application.
- Approach to automated testing.
- Automation, portability and ease of set-up.
- Documentation.
- Code quality.
- A brief write-up which explains how you would deploy your service in a real-world scenario, explaining things such as database choices, deployment pipeline and how would you ensure your service could scale.

You can use whatever language, framework, stack you wish. But please make it's easy to run locally, and documented.

Upload your results to Github, try to commit often so we can see your progress and thought process, when you're complete, send over the link. Try to keep it under a couple of hours, we don't want to take up too much of your time!

If you can't or don't want to upload to Github, feel free to zip it up and send it over via Email or Dropbox etc.