Angular Challenge

Background:
In aircraft maintenance, tasks can come due in multiple ways. Some tasks can be date-driven (change oil every 3 months) or based on other metrics such as hours, cycles, or landings (change the oil every 20 hours).

Maintenance can also come due by multiple metrics at the same time.

Challenge:
Your goal is to write an Angular application in your preferred architecture to do the calculations to arrive at the NextDueDate for a maintenance task.

Requirements:

1. Build an Angular application
2. Have a way to select an aircraft
    a. Default to the values given but allow updating the DailyHours and CurrentHours (in-memory no need to persist this)
3. Display a list of tasks for the aircraft with a calculated NextDueDate
    a. Use the list of tasks provided below.
4. Provide a way to recalculate the task list when the aircraft information changes

Aircraft:
Fake calling an API and look up the following Aircraft with Utilizations:

AircraftId: 1
DailyHours: 0.7
CurrentHours: 550

AircraftId: 2
DailyHours: 1.1
CurrentHours: 200

Tasks:
Fake calling an API to get back an array of tasks with the following data types:

```
Task {
   ItemNumber       int
   Description      string
   LogDate          date
   LogHours         int(Null)
   IntervalMonths   int(Null)
   IntervalHours    int(Null)
}

Tasks: [
            {
               ItemNumber: 1,
               Description: "Item 1",
               LogDate: "2018-04-07T00:00:00",
               LogHours: null,
               IntervalMonths: null,
               IntervalHours: null
            },
            {
               ItemNumber: 2,
               Description: "Item 2",
               LogDate: "2018-04-07T00:00:00",
               LogHours: 100,
               IntervalMonths: 12,
               IntervalHours: 500
            },
            {
               ItemNumber: 3,
               Description: "Item 3",
               LogDate: "2018-06-01T00:00:00",
               LogHours: 150,
               IntervalMonths: null,
               IntervalHours: 400
            },
            {
               ItemNumber: 4,
               Description: "Item 4",
               LogDate: "2018-06-01T00:00:00",
               LogHours: 150,
               IntervalMonths: 6,
               IntervalHours: null
            }
   ]
```

Display:

The UI will display the Tasks with the following data types:

DueTasks: [

{

ItemNumber	int
Description	string
LogDate	date
LogHours	int(Null)
IntervalMonths	int(Null)
IntervalHours	int(Null)
NextDue	date(Null)

}

]

NextDueDate Formula Rules:

IntervalMonthsNextDueDate =
LogDate + IntervalMonths

- Note: IntervalMonthsNextDueDate will be null if either LogDate or IntervalMonths are null

DaysRemainingByHoursInterval =
((LogHours + IntervalHours) – MockedAircraft.CurrentHours) /
MockedAircraft.DailyHours

- Note: DaysRemainingByHoursInterval could be null

IntervalHoursNextDueDate =
DaysRemainingByHoursInterval + Today

- Note: IntervalHoursNextDueDate could be null
- Note: For unit tests assume today is 6/19/2018

NextDueDate =
MIN(IntervalMonthsNextDueDate, IntervalHoursNextDueDate) OR Null

Sorting:

NextDueDate ASC with nulls at the end then by Description ASC

Test Results using provided data:

Expected NextDueDate and Sort Order for Aircraft 1:
ItemNumber: 3 NextDueDate: 6/19/2018,
ItemNumber: 2 NextDueDate: 8/29/2018
ItemNumber: 4 NextDueDate: 12/1/2018
ItemNumber: 1 NextDueDate:

Expected NextDueDate and Sort Order for Aircraft 2:
ItemNumber: 4 NextDueDate: 12/1/2018,
ItemNumber: 2 NextDueDate: 4/7/2019
ItemNumber: 3 NextDueDate: 5/3/2019
ItemNumber: 1 NextDueDate: