Interview Programming Exercise

Please complete the following programming exercise to produce a work sample for evaluation by the hiring team. If you have not worked in Go before, you should first complete the Go tutorial and browse the standard library.

Please plan to devote three hours to the exercise itself, implementing as much of the specified behavior as possible in that amount of time. You're welcome to spend more time on it, but no such expectation is set. Coding to a high standard is more essential than 100% completion; what we're looking for is a sample of your best work. For any functionality not implemented, please be prepared to discuss how your design would accommodate it.

Create a free private repository in GitHub including your code and some basic instructions on how to build, test, and run the application. When viewing your private repository, visit the Settings tab, select Collaborators, and add `radar-hiring-code-review` as a collaborator.

Description: Mini Incident ETL

Story: *As a user, I want a tool to consume privacy incidents formatted in JSON, and output them in a specified sort order in CSV format, so that I can adapt web service output for a bulk DB insert.*

Create a command-line Go application that consumes incident data in JSON, transforms the incidents, and outputs them in CSV format.

Desired Functionality

The following items describe desired functionality in order of importance.

The application should be able to:

1. Read JSON input in the format specified below, and output its data in CSV format.
2. Sort the incidents by `discovered` date, from least recent to most recent.
3. Sort the incidents by `status`, in the order specified below.
4. Accept a command-line argument `sortfield`, specifying the field to sort on.
5. Accept a command-line argument `sortdirection`, specifying the direction of the sort, which recognizes parameter values `ascending` and `descending`.
6. Accept a command-line argument `columns` specifying which columns to output in CSV, accepting a list of field names like `id,name`, which will restrict the output of the CSV to only the columns specified.

JSON Input Format

The contents of the JSON input are structured according to this example:

```
[
  {
    "id": 2,
    "name": "Misdirected fax",
    "discovered": "2018-04-02",
    "description": "Patient's medical records faxed to wrong
number.",
    "status": "New"
  },
  {
    "id": 1,
    "name": "Lost laptop",
    "discovered": "2018-02-19",
    "description": "Doctor lost her laptop while on vacation.",
    "status": "Done"
  },
  {
    "id": 3,
    "name": "Lost iPad with medical record",
    "discovered": "2018-04-01",
    "description": "Nurse misplaced a patient's medical record
while in the office.",
    "status": "In progress"
  }
]
```

`status` has the following possible values and sort order:

```
["New", "In progress", "Done"]
```