

# Backend Assessment - Blog Posts

This assessment will be evaluated based on the following criteria:

- **Completion:** Did you complete all the steps? Does each part work properly?  
This is the most important criterion. Please note that we mark for completion automatically, therefore it's your job to ensure all the routes, parameters and responses are named and formatted correctly.
- **Fundamentals:** Is your code efficient? Do you use the appropriate data structures?
- **Code Organization and Quality:** How organized is your solution?
- **Testing:** Did you write some tests for the solution? How is the coverage?
- **Speed:** We are looking for solutions that are completed within an appropriate time frame (around 6 hours is an appropriate time frame).

## Resources

If you are not familiar with how to set up a basic API, we recommend [Node + Express](#).

# Data Source

You will be building an API that requires you to **fetch data** from this API:

## Request:

**Route:** <https://hatchways.io/api/assessment/blog/posts>

**Method:** GET

## Query Parameters:

Field	Type	Description
tag	String (required)	The tag associated with the blog post.

Notice that the parameter is a query parameter - you can read more about query parameters [here](#). An example of sending the tag parameter is <https://hatchways.io/api/assessment/blog/posts?tag=tech>.

**Our API can only filter one tag at a time - notice that the field “tag” is singular and not plural.**

It will return a JSON object with an array of blog posts. An example response is:

```
{
  "posts": [{
    "id": 1,
    "author": "Rylee Paul",
    "authorId": 9,
    "likes": 960,
    "popularity": 0.13,
    "reads": 50361,
    "tags": [ "tech", "health" ]
  },
  ...
]
```

# API Requirements

You need the following routes in your API:

## Route 1:

### Request:

**Route:** /api/ping

**Method:** GET

### Response:

Response body (JSON):

```
{  
  "success":true  
}
```

Response status code: 200

## Route 2:

### Request:

**Route:** /api/posts

**Method:** GET

### Query Parameters:

Field	Type	Description	Default	Example
tags	String (required)	A comma separated list of tags.	N/A	science,tech
sortBy	String (optional)	The field to sort the posts by. The acceptable fields are: <ul style="list-style-type: none"><li>• id</li><li>• reads</li><li>• likes</li><li>• popularity</li></ul>	id	popularity
direction	String (optional)	The direction for sorting. The acceptable fields are: <ul style="list-style-type: none"><li>• desc</li><li>• asc</li></ul>	asc	asc

## Successful Response:

The response of the API will be a list of all the blog posts that have **at least one tag specified** in the tags parameter.

The sortBy parameter specifies which parameter should be used to sort the results return. It is an optional parameter, with a default value of `id`.

The direction parameter specifies if the results should be returned in ascending order (if the value is "asc") or descending order (if the value is "desc"). The default value of the direction parameter is `asc`.

Here is how the response would look like:

Response body (JSON):

```
{
  "posts": [{
    "id": 1,
    "author": "Rylee Paul",
    "authorId": 9,
    "likes": 960,
    "popularity": 0.13,
    "reads": 50361,
    "tags": [ "tech", "health" ]
  },
  ...
]
```

Response status code: 200

## Error Responses:

If `tags` parameter is not present:

Response body (JSON):

```
{
  "error": "Tags parameter is required"
}
```

Response status code: 400

If a `sortBy` or `direction` are invalid values, specify an error like below:

Response body (JSON):

```
{
  "error": "sortBy parameter is invalid"
}
```

Response status code: 400

In order to complete this task you will need to:

- For every tag specified in the tags parameter, fetch the posts with that tag using the Hatchways API (make a separate API request for every tag specified)
- Combine all the results from the API requests above and remove all the repeated posts (try to be efficient when doing this)
- **You will get a better score on our assessment if you can make concurrent requests to the API**(making the requests in parallel) (we understand that this job is easier in some languages vs. others)

We have provided an API with the correct solution. This should only be used to verify your results. **Do not call this API in your application.** Here it is in action:

<https://hatchways.io/api/assessment/solution/posts?tags=history,tech&sortBy=likes&direction=desc>

## Step 3

An important part of development is testing. In this step, we want to see tests written for your routes. **Do not use the solutions API route to perform testing in this step.** Think about the different ways to test the app, and the best way to get good coverage.

## Step 4 (Bonus!)

Making API calls to other servers can be expensive. How can you reduce the number of calls you make to a server? You can cache the results of an API call on your server. Try to implement a server side cache to our API. Two tips are 1) keep it simple, and 2) feel free to use existing libraries/frameworks.

## Config file

Along with your submission, please provide a json configuration file that lets us know how to install and run your application. Here is the format of the configuration file:

```
{
  "install": "npm install",
  "run": "npm run dev",
  "port": 3000
}
```

The file must be called config.json and must be stored in the root level of your project. Here are a couple of example config files for different languages/frameworks:

- [Javascript config file](#)

- [Python config file](#)

## Checklist

Before submitting your assessment, make sure you have:

- ☐ An **/api/posts** route that handles the following query parameters:
  - ☐ tags (mandatory) : any number of comma-separated strings
  - ☐ sortBy (optional) : one of “id”, “reads”, “likes”, “popularity”
  - ☐ direction (optional) : one of “asc”, “desc”, defaults to “asc”
- ☐ Error handling: Return an error message if:
  - ☐ tags parameter is missing
  - ☐ sortBy or direction has an invalid value
- ☐ Testing without using our solution API route
- ☐ A config.json file as described above
- ☐ Caching (bonus)