Ruby Backend Assessment - Blog Posts

In this assessment, you will write a simple backend JSON API. If you notice something is not working (like the API, or any of the links in this document), please contact hello@hatchways.io.

This assessment will be evaluated based on the following criteria:

- Correctness: Is your solution complete and does it pass different test cases?
- Code Organization, Readability, & Maintainability: Is your code easy to read and well organized?
- Code Performance: Is your code efficient? Did you use appropriate data structures?
- Best Practices: Did you utilize good programming practices (write unit tests, avoid anti-patterns)? Did you show a good grasp of your language/framework of choice?
- Completion speed: A fast completion time comparable to the completeness of your solution. This is the least important criteria.

We use the following rubric to evaluate your submission. **Please note that if your submission does not attempt to complete all of the requirements, we will be unable to provide feedback on it.**

You must use **Ruby on Rails** to complete this assessment. Please document in a Readme how to start the application.

Data Source

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

Request:

Route: https://api.hatchways.io/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://api.hatchways.io/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: ● id ● reads ● likes ● popularity	id	popularity
direction	String (optional)	The direction for sorting. The acceptable fields are: ● desc ● asc	asc	asc

Successful Response:

The API response 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 field should be used to sort the returned results. This 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 should look:

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

Here is what you will need to do to complete this task:

- 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://api.hatchways.io/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. Be sure to test your code after implementing this to be sure it still works as expected.

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
- ❏ Caching (bonus)

Submission Details

Please submit your code in a compressed folder on the Hatchways platform. The max submission size is 5MB.

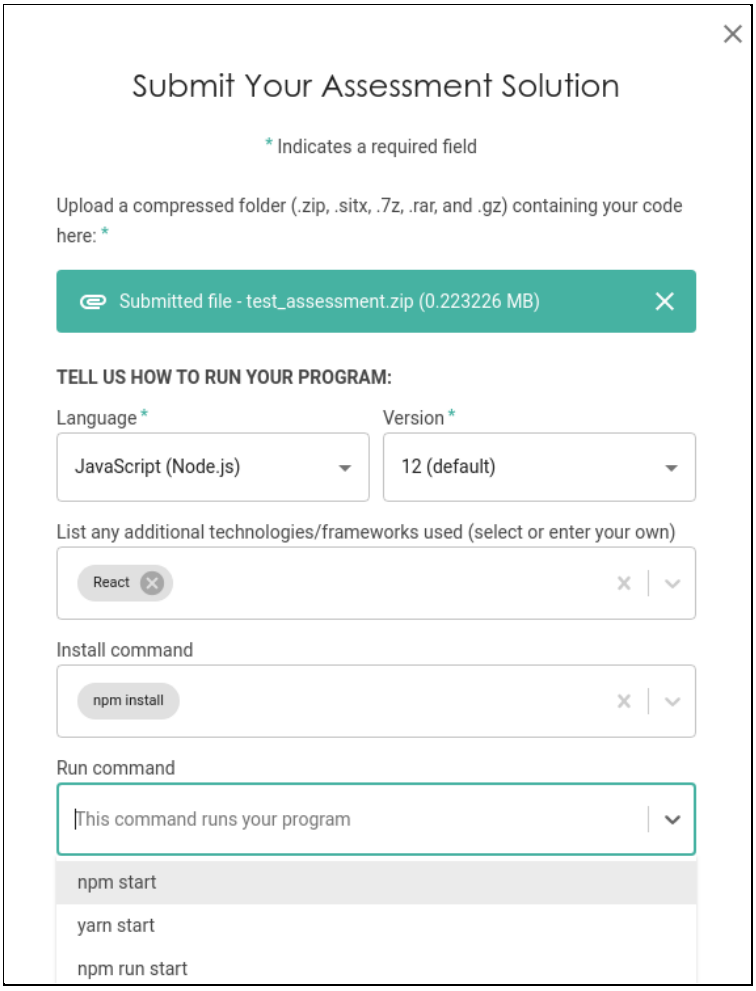
Upon clicking the submission button, you will see a form as pictured below. We need this information to be able to test your application.

1. Choose which **language** and additional technologies you used to develop your solution. Be sure to select the appropriate version for the language you have used.
2. Provide us with the **install command**, the **run command**, and the **port** that you used to run your application.
3. If you cannot find your commands in our suggestions, simply type your own and select “Use command”.

Please note that these commands will be used to run automated tests, so filling in every relevant field and providing accurate commands will allow you to receive feedback more quickly on your submission. If you have any notes to provide about your submission, please put them in a README, not in the submission form. Additionally, note that the install and run commands will be run from the root level of your submission, so please organize your files accordingly.

Do not submit any built folders, since the compressed folder will be too large. **Do not submit your external dependencies (like the node_modules folder), since the compressed folder will be too large. We will be installing your dependencies before we run your code.**

If your submission is too big and you can't figure out how to compress, you are welcome to email your solution to hello@hatchways.io. Please include your name, and use the email you signed up with on the Hatchways platform. Use the subject line “Back-end Assessment Submission”.



Public Repositories

Do not post your solution to a public repository. We understand that you may want to share projects you have worked on, but many hours go into developing our tools so we can provide a fair skills evaluation.