

Bajaj Finserv Health Dev Challenge (Qualifier 1)

SRM Institute of Science and Technology | 22nd September'24

Note: This is a full stack challenge – involves both backend and frontend parts. Completion of both challenges is required for all candidates.

Candidates are free to use online resources including, but not limited to [OpenAI ChatGPT](#), [Google Gemini](#), [Anthropic Claude](#) and [StackOverflow](#).

Note: Refer to the end for submission instructions

Backend

Objective

Build and deploy a REST API with one endpoint that accepts requests with both GET and POST methods.

POST method endpoint takes in the request (JSON) and returns the following:

1. Status
2. User ID
3. **College Email ID**
4. **College Roll Number**
5. Array for numbers
6. Array for alphabets
7. Array with the highest lowercase alphabet (refer to logic explanation below)
8. File
 - a. Boolean that indicates the validity of the file
 - b. MIME type of the file
 - c. File size in KB

GET method endpoint doesn't take any user input, it just returns an *operation_code*

- ***Please refer to Annexure (A) for request/response samples***

Hosting

Any provider of your choice, if you don't have one already, please use Heroku / Netlify / Vercel / Firebase or any other provider that provides REST API deployment.

Logic – (2 Parts)

1. Route: /bfhl | Method: POST

- a. Example: <https://testbfhl.herokuapp.com/bfhl> [POST Method]
 - b. Response should always contain your user_id (fullname_dob) in the following format
 - i. "user_id": {full_name_ddmmyyyy}E.g.: "user_id": "john_doe_17091999"
 - c. "is_success" should be returned in the response to mark the status of operation. It can be true / false
 - d. File handling:
 - i. file_valid: Boolean that indicates the validity of the file from Base 64 string. This should be false if no file is sent / the sent file isn't valid.
 - ii. file_mime_type: The MIME type of the file
 - iii. file_size_kb: Size of the file in KB
-

2. Route: /bfhl | Method: GET

- a. Example: <https://testbfhl.herokuapp.com/bfhl> [GET Method]
- b. ***Doesn't take any input from the user. The endpoint will be hit with a GET request, that's it.***
- c. Expected HTTP Status Code: 200
- e. Expected Response Body (hardcoded) (JSON):

```
{  
  
  "operation_code":1  
  
}
```

Note: Be sure to follow other best practices including exception handling, input validation and others.

Annexure (A) - (ONLY FOR POST REQUEST)

Example A

Note: If an alphabet is present in the input, it will always be a single character, never a word. And "highest_lowercase_alphabet" refers to the lowercase alphabet from the input array which occurs last in the a-z series.

Request

```
{  
  
  "data": ["M", "1", "334", "4", "B", "Z", "a"],  
  
  "file_b64": "BASE_64_STRING"  
  
}
```

Response

```
{  
  
  "is_success": true,  
  
  "user_id": "john_doe_17091999",  
  
  "email": "john@xyz.com",  
  
  "roll_number": "ABCD123",  
  
  "numbers": ["1", "334", "4"],  
  
  "alphabets": ["M", "B", "Z", "a"],  
  
  "highest_lowercase_alphabet": ["a"],  
  
  "file_valid": true,  
  
  "file_mime_type": "image/png",  
  
  "file_size_kb": "400"
```

```
}
```

Example B

Request

```
{  
  
  "data": ["2", "4", "5", "92"],  
  
  "file_b64": "BASE_64_STRING"  
}
```

Response

```
{  
  
  "is_success": true,  
  
  "user_id": "john_doe_17091999", "email": "john@xyz.com",  
  
  "roll_number": "ABCD123",  
  
  "numbers": ["2", "4", "5", "92"],  
  
  "alphabets": [],  
  
  "highest_lowercase_alphabet": [],  
  
  "file_valid": true,  
  
  "file_mime_type": "doc/pdf",  
  
  "file_size_kb": "1800"  
}
```

Example C

Request

```
{  
  
  "data": ["A", "C", "Z", "c", "i"]  
  
}
```

Response

```
{  
  
  "is_success": true,  
  
  "user_id": "john_doe_17091999", "email": "john@xyz.com",  
  
  "roll_number": "ABCD123",  
  
  "numbers": [],  
  
  "alphabets": ["A", "C", "Z", "c", "i"],  
  
  "highest_lowercase_alphabet": ["i"],  
  
  "file_valid": false  
  
}
```

Frontend

Objective

Develop and deploy a frontend application that uses the backend you've developed to process the input and get response, then render it on frontend. (Preferably React based – including frameworks like Next.js, but that's optional).

Logic

1. The application should have a text input that accepts JSONs that are from the previous section's (backend's) request. Example - `{ "data": ["A", "C", "z"] }`
2. The frontend application should call the REST API that you've created to process the data in request and return the response as per the logic mentioned in backend challenge.

Details

Create a user interface with the following features:

1. Text input field and submit button:

- Accepts only valid JSON input
- Validates JSON format on submission, show error in case of invalid input
- Calls the REST API that you've created, with payload from the text input
- Gets the response and then uses the following logic to render it

2. Multi-Select Dropdown:

- Appears after valid JSON submission
- Contains the following options:
 - a. Alphabets
 - b. Numbers
 - c. Highest lowercase alphabet

3. Response shown on frontend will be basis options selected from dropdown. E.g. if user selects Alphabets & Numbers then response should render data associated only with alphabets & numbers
4. Also, the website title should be your roll number

Eg: In the following image, "Google" is the title of the first tab.



Sample Output:

In case, user selects only “Numbers” from the multiselect filter, response will look as following:

API Input

```
{"data":["M","1","334","4","B"]}
```

Submit

Multi Filter

Numbers x

x

▼

Filtered Response

Numbers: 1,334,4

Submission

1. Host your backend & frontend applications
2. Share your backend API endpoint (ending in /bfhl) and frontend application URL in the following form:
Form Link: <https://forms.office.com/r/xVucYvKHvg>
3. Fastest valid submissions will be considered
4. Any attempt to duplicate submissions or assist others will lead to disqualification of both candidates

DO NOT FORWARD THIS DOCUMENT