Senior Software Engineer - Home Assignment

Overview

Design and implement a URL fetching service that executes multiple HTTP requests while respecting individual and global timeout constraints.

Requirements

Service Specification

Your service should accept requests containing:

1. `execution_timeout` (optional, default: 800ms)
    - Global timeout for the entire operation
    - Data type: your choice
    - Any URL requests not completed within this timeframe should be aborted
    - Successfully completed requests must still be returned
2. `urls` (required)
    - Array of URL objects to fetch
    - Each object contains:
        - `url` (required): string - the URL to fetch
        - `timeout` (optional): individual timeout for this specific request
        - `headers` (optional): custom HTTP headers as key-value pairs (data structure of your choice)

Example input structure (JSON format):

```
{
  "execution_timeout": 800,
  "urls": [
    {
      "url": "https://google.com",
      "timeout": 300,
      "headers": {
        "X-Device-IP": "10.20.3.15"
      }
    },
    {
      "url": "https://amazon.com",
      "timeout": 200,
      "headers": {
        "X-Device-IP": "10.10.30.150"
      }
    }
  ]
}
```

Response Specification

Return an object containing a `results` array with one entry per URL request (in the same order), where each entry includes:

- `code`: HTTP status code (success, failure)
- `error`: string describing the error (optional, present if request failed)
- `payload`: response body as string (optional, present if request succeeded)

Example output structure (JSON format):

```
{
  "results": [
    {
      "code": 200,
      "payload": "<!DOCTYPE html><html>...</html>"
    },
    {
      "code": 0,
      "error": "Request aborted by timeout"
    }
  ]
}
```

Technical Requirements

- **Timeout Handling**:
  - Respect both global `execution_timeout` and individual `timeout` per URL
  - Aborted requests should still appear in results with appropriate error messages
- **Headers**: Support custom headers per request

Implementation Choices (Your Decision)

You are free to choose:

- Communication protocol
- Data format
- Programming language
- Libraries and frameworks
- Response structure (must include `results` array as specified)
- Execution strategy

Deliverables

1. Source Code

- Service implementation
- Client implementation (for exotic communication protocols, data formats)

2. README Documentation

Your README must include:

1. **How to run the service** (setup instructions, dependencies, commands)
2. **How to run the client** with sample payloads
3. **Technical decisions rationale**

Example Implementation (Illustrative Only)

This example uses HTTP POST with JSON payload. **You may use any protocol/format you prefer.**

Request

```
POST http://localhost:8080/execute
Content-Type: application/json
```

```
{
  "execution_timeout": 800,
  "urls": [
    {
      "url": "https://google.com",
      "timeout": 300,
      "headers": {
        "X-Device-IP": "10.20.3.15"
      }
    },
    {
      "url": "https://amazon.com",
      "timeout": 200,
      "headers": {
        "X-Device-IP": "10.10.30.150"
      }
    }
  ]
}
```

Response

```
{
  "results": [
    {
      "code": 200,
      "payload": "<!DOCTYPE html><html>...</html>"
    },
    {
      "code": 0,
      "error": "Request aborted by timeout"
    }
  ]
}
```

Evaluation Criteria

Your solution will be assessed on:

- **Correctness**: Proper handling of timeouts and execution logic
- **Code quality**: Clean, maintainable, well-structured code
- **Performance**: Efficient execution approach
- **Documentation**: Clear README with rationale for technical choices
- **Error handling**: Robust handling of network failures, timeouts, and edge cases

---

Questions?

If any requirements are unclear, document your assumptions in the README.

Good luck!