Pinggy

contact@pinggy.io

Screening assignment for Next.js + Spring Boot full-stack position

Objective:

The objective of this assignment is to assess your proficiency in Next.js and Spring Boot development. You will be tasked with creating a tiny Spring Boot application with a filter that performs custom authentication based on the presence of a specific header in the HTTP request. Additionally, you'll need to implement a controller that retrieves and saves the value of the post data, as well as the value of the header in the memory.

Backend:

- Initialize the backend project from https://start.spring.io/
- Use Spring Boot 3.4.4
- Use Java 17 / Java 21 / Java 24. Mention the version in the readme file.
- Mention your name and email in the readme file.

Custom Authentication Filter:

- Create a Spring Boot filter that intercepts incoming HTTP requests.
- This filter should check if the header "**`PinggyAuthHeader`**" is present in the request.
- If the header is present and value is not empty:
    - Store the value of the header in a **ThreadLocal** variable.
- If the header is not present or the value is empty string:
    - Respond with HTTP status code 401 (Unauthorized). The request should not reach the controller.

Controller Implementation:

- Implement a Spring MVC controller that handles a POST request on path "**/post**"
    - The controller will accept a JSON body in the format { "title": "Post title", "body": "Post body"}
    - This controller will also retrieve the value of the header "**`PinggyAuthHeader`**" from the ThreadLocal storage.
    - The controller will store the post body, as well as this corresponding header value in a variable in the memory. [Note: no databases required]
    - Define proper DTO class for handling the request body
- Implement a Spring MVC controller that handles a GET request on path "**/list**"

- This controller returns a list of posts already in memory, in the format: [{ "title": "Post title", "body": "Post body", "PinggyAuthHeader":"xyz" }]
- Define proper DTO class for handling the response type

Requirements:

- Use Java and Spring Boot for implementation.
- Ensure proper error handling and response status codes.
- Use appropriate dependencies and configurations in the Spring Boot project.
- Organize the project structure efficiently.

Instructions:

- Document any important decisions made during implementation in a README.md file.
- The Readme file should also contain the instructions to run the application.

Additional Notes:

- You are free to use any additional libraries or frameworks if necessary.

Frontend:

- Create a Next.js application with a single page that allows users to submit and view posts.
- The page will contain two major components: A form to submit new posts. And a list showing existing posts.
- Form:
  - Implement a form with three fields: **Title**, **Body**, and **Auth Code**.
  - On submission, send a `POST` request to `/post` with the following payload:

    json
    CopyEdit
    `{ "title": "Post title", "body": "Post body" }`
  - Include the **Auth Code** in the request header as `PinggyAuthHeader`.
- Post list:
  - Fetch and display posts from the `/list` API.
- Implement basic error handling for failed submissions.
- Add a loading state while fetching posts.
- Ensure a responsive and user-friendly design.

Instructions:

- Document any important decisions made during implementation in a README.md file.

- The Readme file should also contain the instructions to run the application.

Submission:

- Once completed, arrange the backend and frontend into two separate folders named “backend” and “frontend”. Put both in another folder named “<yourName>_<yourEmail>”.
- Include some screenshots.
- Remove the “node_modules” folder and similar unnecessary folders.
- Zip the folder and upload to google drive or any other similar service. Share a downloadable link in the form.

- Form link: https://forms.gle/DbdqmG9qcFScyHRN8