Practical Assessment for Software Engineer Intern

Congratulations! *You've progressed to the next stage of our Software Engineer Internship selection process. This challenge will assess your skills across various platforms and your ability to make independent decisions. Take your time, be creative, and have fun*!

Note : To avoid rejections, it's crucial to follow the instructions precisely. Only complete the tasks outlined, such as creating the login page, and NOT register page, even if they may seem relevant.

1.Create a GitHub Account: If you don't have one already, create a GitHub account. Make sure to create a new public repository for your assignment and share the repository link with us.

2.Commit at Every Task: As you progress through the tasks, commit your changes to the repository on GitHub. Each commit should correspond to a specific task number and include a descriptive message following the convention

"#<task number>, <commit message>"

Examples:

"#1, project initiated with root folder"
"# 5.1, login screen created"

While incremental commits are encouraged, it's crucial to commit whenever a requirement is completed.

3.Choose Your Preferred Language and Framework:

- Frontend: HTML, React, Angular
- Backend: Node.js, Java, Python
- Framework (Optional): NestJS, Spring Boot, Django

Assignment

You're tasked with building an imaginary school system for managing student attendance.

1.Set Up Project Structure:

- Create a root project folder named "attendance-system".
- Inside the root folder, create subfolders for "server", "client", and "docs".

```
attendance-system
├── server
├── client
└── docs
```

2.Initialize Server Code:

- Choose your preferred language or framework.
- Initialize the server code inside the "server" folder.

3.Implement Login REST API:

- Write a REST API in the server to handle login requests.
- The API should receive a username and password from the client and respond with a JWT token.
- You can store username and password in a file for simplicity (no need to connect to a database).

4.Create Client Pages:

- Develop two pages in the client application: a login page and a home page.
- The login page should have input fields for username and password, along with a login button.
- The home page should display a welcome message and a logout button.

4.Client-Server Interaction: (commit every step)

- Write a service in the client to connect to the server via REST API.
- Implement the following client behaviors:
  - Validate invalid username or password on the login screen.
  - Successfully log in the user and redirect to the home page.
  - Redirect to the home page on logout.
  - Redirect to the login page if the user is not logged in.
  - Redirect to the home page if the user is already logged in.

5.Create a ReadMe.md File:

- Write a clear and comprehensive ReadMe.md file at the root of the project.
- Include instructions on how to set up and run the project.
- Provide login credentials for testing purposes.

6.(Optional) Add screenshots

- Whenever feasible, like after completing significant tasks such as the login screen, capture a screenshot.
- Add the captured screenshot to the "docs" folder.
- Ensure the screenshot is appropriately labeled with the respective task number for easy reference.

Your final folder structure will resemble the following

```
attendance-system
├── server
│   └── ...
├── client
│   └── ...
├── docs
│   └── ...
└── ReadMe.md
```

We look forward to reviewing your submission!!