Tech interview tasks

CARSWIP

Hey there,

We would like you to complete the following <4> tasks because we're curious to see how you **think**, **solve problems** and of course, how you **code**. You have **5 days** to complete the tasks. When you're ready, just drop an email to george@carswip.dk with your answers or links.

Keep in mind, that we value *quality* over *quantity* and feel free to ask any questions because *communication is key*!

Good luck,
The Carswip Team

Task #1

Here's a small github project: https://github.com/carswip/devtest
There are two counters on the page, both are programmed to crash when incremented to 5, however, the **whole page** breaks when any of the counters throw an error. Fix the issue, so each component should only affect 'itself' and should not affect the whole page. Please upload your final code into a new repository, and share the link in your answer.

Task #2

There is an anchor in a web page. The code of the anchor is the following:

```
<a href="https://a-video-company.s3.amazonaws.com/origin/a_video.mp4" download>Download Video</a>
```

When the user clicks the rendered Link, then instead of actually downloading the video, the video just opens in a new tab, without starting the download.
Why does it happen? How could it be resolved?

Task #3

A company has a web application that runs 24/7. The server receives requests throughout the day, but it has idle time on weekends and late evenings (when almost no one uses the application). None of the developers want to work on weekends or evenings, but every week, they have to release a new version of the web application, both for the backend and the frontend.
What solution would you introduce for the company and how it solves the releasing problem without any or small downtime?

Task #4

Create a simple web application (client side rendered) containing few pages:

- **Home**: A lorem-ipsum page
- **Login**: No need for security or authentication, can accept random credentials but must send a proper request
- **Item-list**: User should be able to select items and then submit them (Simple pageable list / grid)
- **Item-details**: A simple page showing the clicked item details

Other details:

- Unauthorized user has access: **Home** and **Login** pages. Authorized user: **Home**, **Item-list** and **Item-details** pages. The app should be made the most user-friendly way in terms of logic, the UI design can be as simple as possible. **No** need to set up **database**, data can be hardcoded.
- "Submitting the selected items": the frontend shall send a request to backend (then a simple log is enough)
- What do we mean by "User-friendly by code logic": For example, the selection of items should persist if user clicks the item details but should not persist after refreshing the page or navigating to the Home-page
- The backend should be simple, but secure. E.g.: the backend should check for some kind of authentication headers to allow/disallow access, but it's enough if you check that there is such a header or not
- The more the application works as a production application, the better.