

# SPRIBE

## Test Assignment for the Angular Developer Position

### Short description:

Take the existing project from the provided archive and complete the following tasks:

1. Create an array of forms and submit them as one
2. After submitting the forms, display a timer and provide the option to cancel the submission
3. Create Directive for input validation message

Provide your solution as a ZIP archive; do not publish it on GitHub.

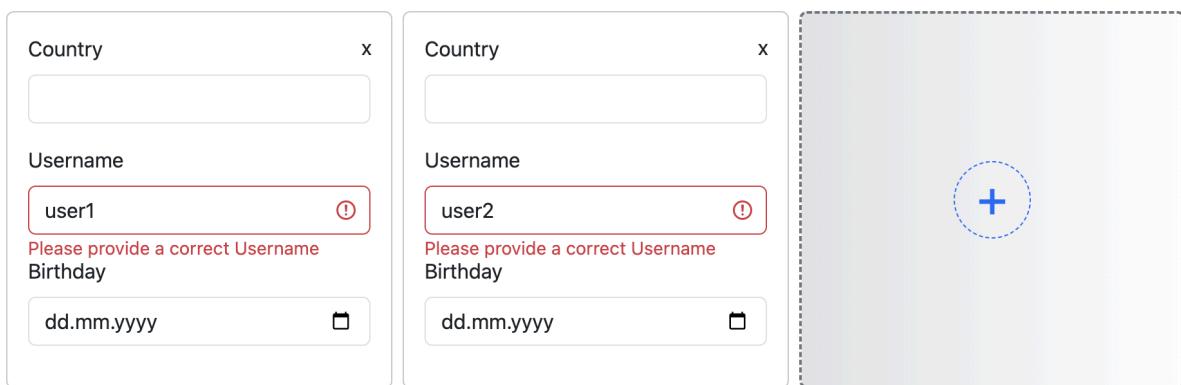
### Detailed description:

- Created a page with form cards and a Submit button. Each card should have three inputs.

The diagram illustrates a user interface for a form submission. It features a main button at the top labeled "Submit all forms". Below it is a vertical column of three input fields: "Country" (with an "x" icon), "Username", and "Birthday" (with a date input field and a calendar icon). To the right of these fields is a dashed rectangular area containing a blue circle with a plus sign (+) in the center, suggesting a placeholder for more input fields.

- First input, Country should be a text input. While the user types, it should suggest values from the `Country` enum ([src/app/shared/enum/country.ts](#)). It should validate and not allow to submit forms with values not listed in the `Country` enum.
- Second input, Username should be a text input with backend validation. After the user stops typing, the value from this input should be sent to [/api/checkUsername](#). If the backend responds with `{isAvailable: true}`, it means that the username is available. If the backend responds with `{isAvailable: false}`, the user should choose another username. To emulate the backend, all requests are handled by an interceptor. If you need to get `{isAvailable: true}`, the value sent to [/api/checkUsername](#) should include the word 'new'. In other cases, [/api/checkUsername](#) returns `{isAvailable: false}`.
- Third input, Birthday, just datepicker. Birthdays cannot be after the current date.
- Create a directive for input validation message. If validation fails, the directive should mark the input and add text below the input: 'Please provide a correct Country/Username/Birthday' (depending on which input validation fails).
- If there are invalid forms, show their amount next to the Submit button.

[Submit all forms](#) Invalid forms: 2



The image shows two separate form inputs side-by-side. Each form has a header with a 'Country' label and an 'x' close button. Below each header is a text input field. The first form's text input contains 'user1' and has a red border, indicating it is invalid. A red error message 'Please provide a correct Username' is displayed below the input. The second form's text input contains 'user2' and also has a red border, indicating it is invalid. A red error message 'Please provide a correct Username' is displayed below the input. Both forms have a 'Birthday' section with a date input field containing 'dd.mm.yyyy' and a calendar icon. To the right of the forms is a large dashed rectangular box containing a blue circular button with a white plus sign (+).

- A large button next to the form should create another form identical to the first one. Multiple forms can be created (up to 10)

Submit all forms

Country <input type="text"/>	Country <input type="text"/>	Country <input type="text"/>
Username <input type="text"/>	Username <input type="text"/>	Username <input type="text"/>
Birthday <input type="text"/> dd.mm.yyyy	Birthday <input type="text"/> dd.mm.yyyy	Birthday <input type="text"/> dd.mm.yyyy

- A small "x" button in the right corner of each form card should delete that form card.
- The "Submit all forms" button should block all forms from editing and trigger the timer (set it to 5 seconds). The "Submit all forms" button should change to a "Cancel" button. Display the timer next to the button.

Cancel 0:05

Country Nepal	
Username newUser	
Birthday 01.04.2024	

- If the user presses "Cancel," the timer stops, and the user can edit all forms again. If the timer runs out, the values from all forms should be sent to the `/api/submitForm` endpoint, the forms should be cleared, the timer should disappear, and the "Cancel" button should change back to the "Submit all forms" button.

## Acceptance criteria:

- All form validations work
- The user can't submit invalid data
- Input validation message works
- Amount of invalid forms shown

- Adding more form cards works
- Removing form cards works
- Submit with timer works
- Cancel submit works

## Points of focus:

- Code cleanliness, following programming principles and design patterns.
- Architecture, code separation by files, project structure.

## Additional Instructions:

- Markup is not important. Try to make the HTML and SCSS code as good as possible, but don't worry about the appearance of the page.
- Bootstrap is already added to the project; you are free to use it.
- You can find the backend emulator in `src/app/shared/mock-backend/mock-backend.interceptor.ts`.  
`/api/checkUsername` - gets the username, returns `{isAvailable: boolean}` (true if the name contains the word 'new', false in other cases).  
`/api/submitForm` - gets the form array value.
- If you need to explain your decisions, use comments in the code.
- If you need to provide any additional information about the project setup and usage do it in a README file.
- Provide your solution as a ZIP archive; do not publish it on GitHub.

Good luck!)