Lab 2 - Interactive Portfolio Enhancement with Git Workflow, Grade 3

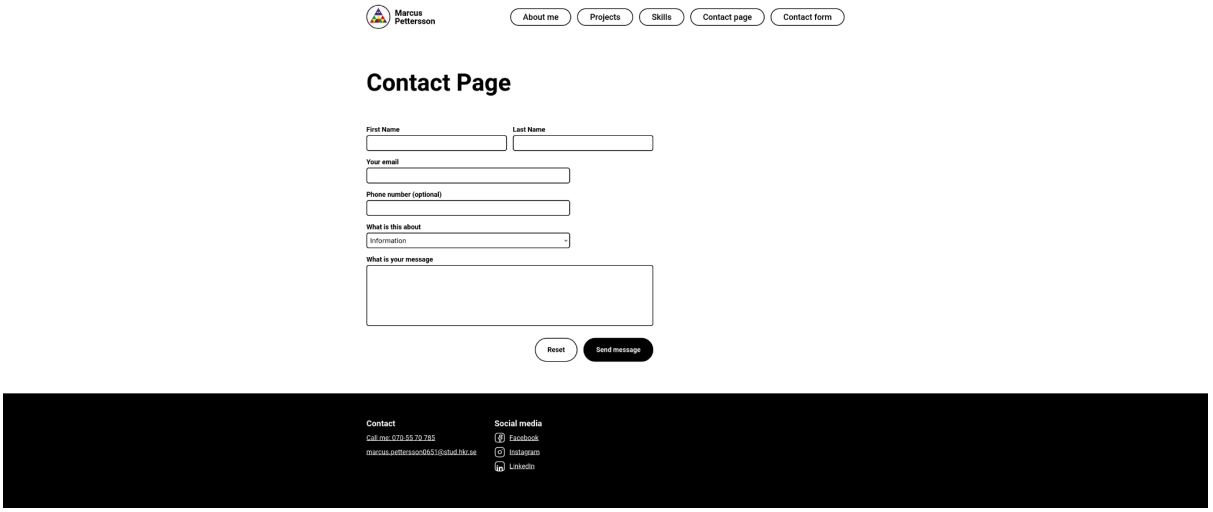
Marcus Pettersson, marcus.pettersson0651@stud.hkr.se

Github repository

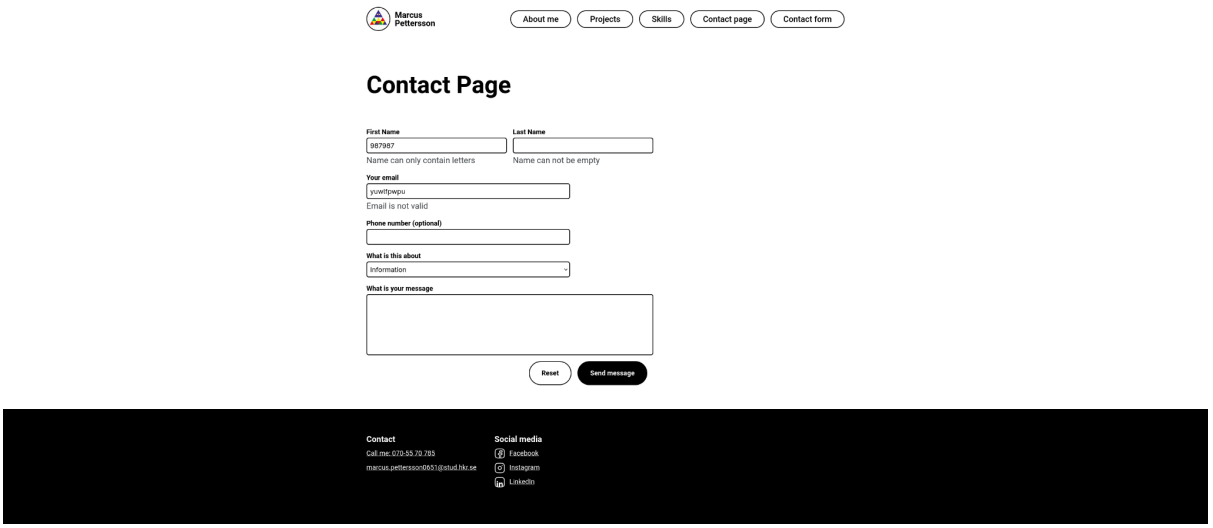
https://github.com/McBEASTse/js-hkr-course

Screenshots

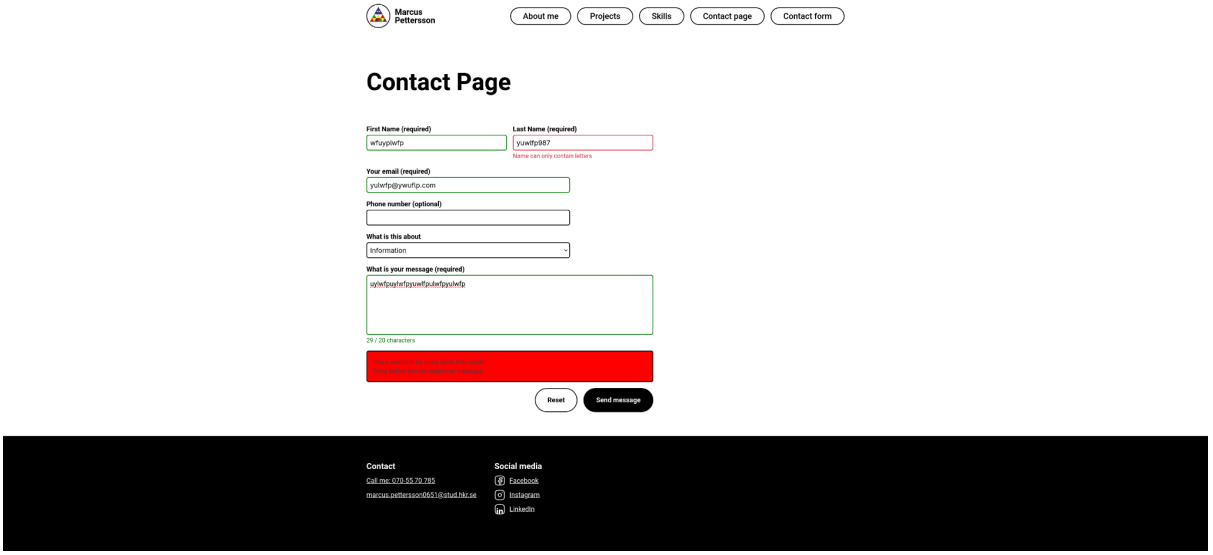
The outline of the form



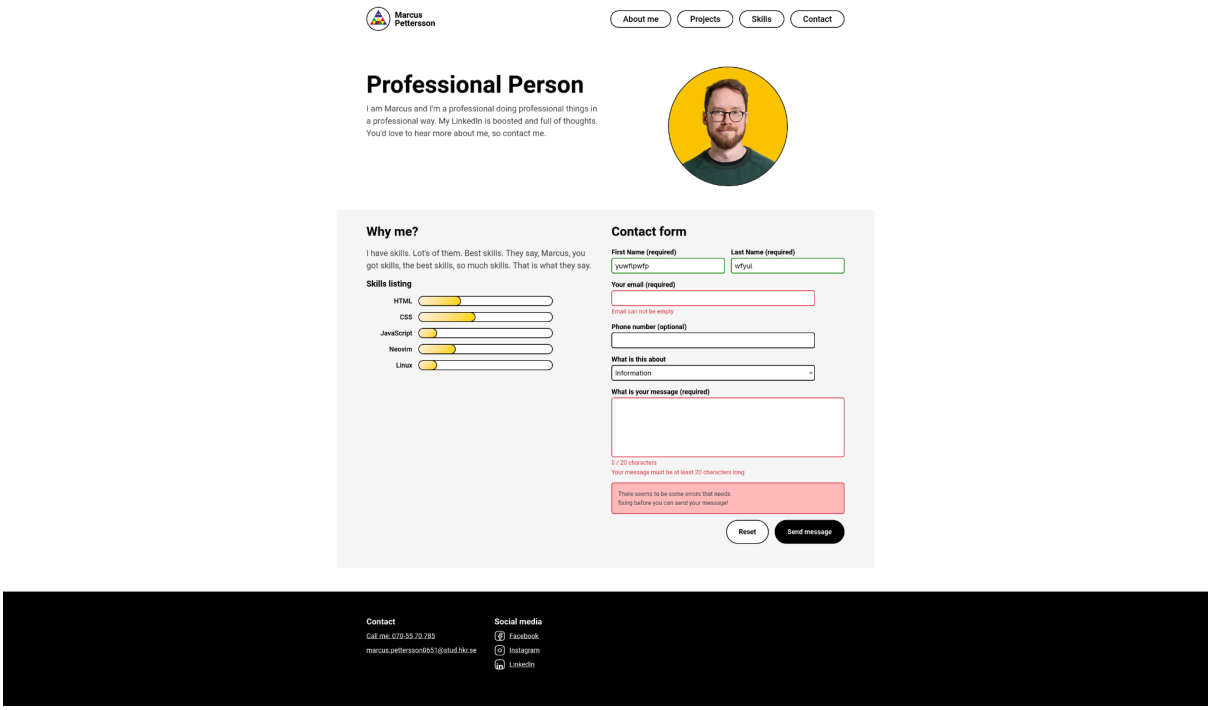
Adding error messages



Styling accordingly



The final result



Challenges and explaining some decisions

Using regular expressions

The first hurdle that I had was how to validate the input from the user: how can I do a check for characters that are not allowed, and how “complete” can I do this check?

```
function validateName(input) {
  // First, trim the input from spaces (so that the input field can not be empty with just spaces).
  const nameValue = input.value.trim();
  // Create a regular expression that is everything NOT in the clamp (^ inverts the regex value, \s\ is a space, \- is for dashes).
  const validateLetter = /[^a-zåäöA-ZÅÄÖ\s\-]/;

  // We test if the input, nameValue, with the regular expression that we created.
  if (validateLetter.test(nameValue)) {
    // If there is anything else than what we put in to the regex, we get an error.
    showError(input, "Name can only contain letters");
    // If the input field is empty, we get another error.
  } else if (nameValue === "") {
    showError(input, "Name can not be empty");
    // If everything i OK, the error is cleared.
  } else {
    clearError(input);
  }
}
```

This is where regular expressions are handy: I can manually set which characters are allowed and which are not, and also do some basic “filters” like checking if an email is valid or not (although checking which emails are truly valid is a big hurdle to tackle (https://e-mail.wtf/)).

Another thing that the user can do is send empty forms with just spaces. Therefore I added trim() to remove spaces before and after the valid string input. This way it’s not possible to just spam spaces and send a form, you actually have to write something.

Then we have the issue with user experience: should the user get an error while typing (for example that the email is not legit until an @ and .com/.se/.whatever is added), or after when the user switches or clicks outside of the input field? For this assignment I chose to give feedback directly to the user as the user writes, but maybe a setTimeout for checking would be smarter?

The use of genAI

Yes, I used genAI for doing this lab, but not by giving me the full answer or code. Instead, I specifically gave the genAI instructions to not give me any code and use Socratic teaching to guide me. Every line has been written by me, and I’ve done my best to learn and understand how everything works by heart. It will take some time and a lot of writing JavaScript to learn all the different pre-existing functions, methods and how to use all different concepts, but some day I will hopefully know enough to solve problems on the fly without looking up everything on the web.

Don’t repeat yourself, refactoring and edge cases

Another difficulty I encountered is that it’s easy to make something work, but it gets tough when you need to write clean and readable code. Everything is basically if statements, and it’s quite difficult to know if there is an existing function of this that I can use, or do I need to write the function myself for what I want to do?

For me there were two distinct parts where - for me - I was able to make the code easier to read: using switch statements instead of if statements when validating the whole form when it gets submitted, and nesting setTimeout() so that each animation finishes before the next animation begins.

```
// Here we validate each input field from earlier with a switch statements instead of us
function validateForm(input) {
  switch (input.id) {
    case "first_name":
    case "last_name":
      return validateName(input);
    // The returned value is checked when the form is submitted.

    case "email":
      return validateEmail(input);

    case "phone_number":
      return validatePhoneNumber(input);

    case "sender_message":
      return validateMessage(input);

    // As we don't validate the dropdown, we just return all defaults as true.
    default:
      return true;
  }
}
```

```
contactForm.addEventListener("submit", function (event) {
  event.preventDefault();
  // We assume that all input fields are valid until they are not.
  let formValidated = true;

  // Here we check each input field if they are valid. If any of them is not, the form is not valid.
  inputElements.forEach((input) => {
    if (!validateForm(input)) {
      formValidated = false;
    }
  });

  // I chose to add the HTML for validation/error message as a string as it just felt easier in this case.
  if (formValidated) {
    formFeedback.innerHTML = `<div class="message_send_base message_send_valid"><p>Thank you ${firstNameInput.value} ${lastNameInput.value}! I will get back to you as soon as possible.</p></div>`;
    clearForm();
  } else {
    formFeedback.innerHTML = `<div class="message_send_base message_send_error"><p>There seems to be some errors that needs fixing before you can send your message!</p></div>`;
  }
  // Here I nested the timeouts so they run in order instead of risking that they interupt each other.
  setTimeout(() => {
    setTimeout(() => {
      setTimeout(() => {
        // Remove the validation/error message.
        formFeedback.innerHTML = ``;
      }, 500);
      // Remove the opacity so it fades out.
      formFeedback.firstElementChild.classList.remove("message_send_visible");
    }, 2500);
    // Add the opacity to the element so it fades in.
    formFeedback.firstElementChild.classList.add("message_send_visible");
  });
});
```

Also, should I use variables for error messages, or just write the error message as parameters? How many variables are too many, and what makes the code more readable? I guess time will tell and by writing more code, and reading others, I will find out what the best practice is (or just ask genAI and get the answer directly).