

Validation

Welcome to this lab activity

In this lab activity, you will implement validation in your Node.js application. This essential technique will protect your application from invalid data.

Task 1: Make a copy of your previous lab

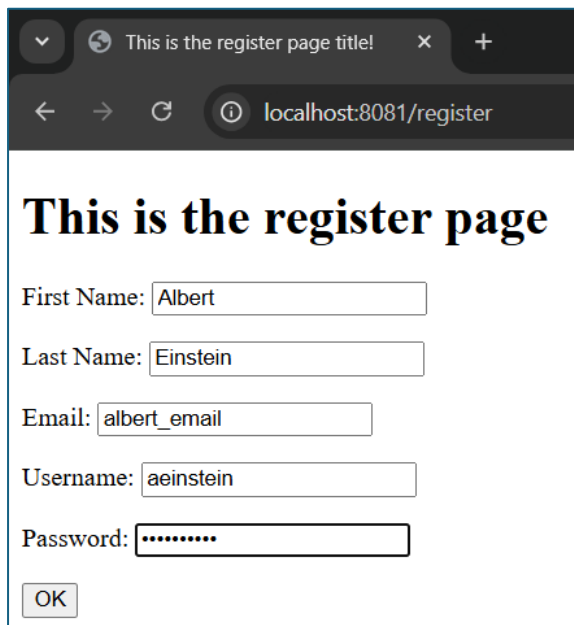
You will start with your previous lab code.

1. Make a copy of the folder `25_authentication_and_authorisation` from your previous lab and call it `26_validation`.
2. Open the new folder in Visual Studio Code.
3. Run the code and confirm that it works as expected.

Task 2: Add validation to the /registered route handler

Currently, the resister form allows any text to be entered in the email field. Clearly, you want to only capture valid email addresses.

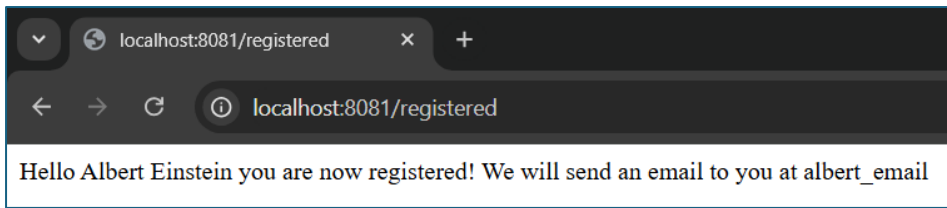
4. First, register a user with an invalid email address to prove this is possible:



The screenshot shows a web browser window with the title 'This is the register page title!'. The address bar shows 'localhost:8081/register'. The page content includes a heading 'This is the register page' and a registration form with the following fields:

- First Name:
- Last Name:
- Email:
- Username:
- Password:

At the bottom left of the form is an 'OK' button.



Let's now add some code to ensure that only correctly formatted email addresses are accepted by this field.

5. You will need a module called `express-validator`. Install it using the following command from a terminal session:

```
npm install express-validator
```

6. Import this module in your `main.js`:

```
// Set up the validator
const { check, validationResult } = require('express-validator');
```

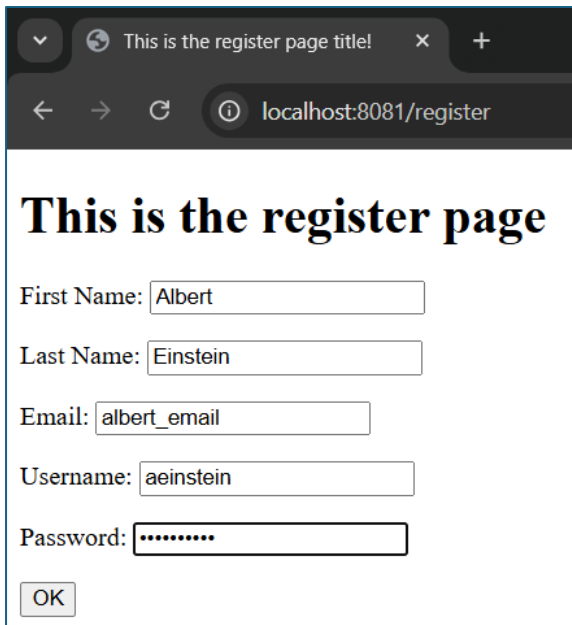
7. Modify the `/registered` route to check that the email field is actually a valid email address format. Change your code to include the code shown in yellow below:

```
router.post("/registered",
  [check('email', "Invalid email address").isEmail()],
  (req, res, next) => {
    // Validation
    const errors = validationResult(req);
    if (!errors.isEmpty()) {
      return res.send(errors.mapped())
    }
  })
```

...rest of code...

Task 3: Test your validation

Now rerun the test from before, checking that the invalid email address is now rejected:



This is the register page

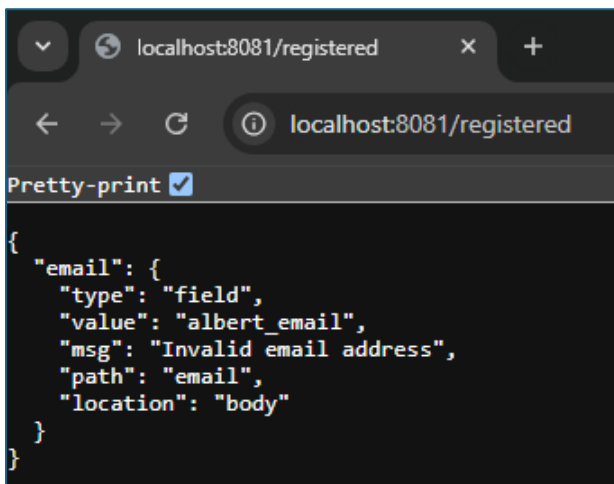
First Name:

Last Name:

Email:

Username:

Password:



```
{
  "email": {
    "type": "field",
    "value": "albert_email",
    "msg": "Invalid email address",
    "path": "email",
    "location": "body"
  }
}
```

Task 4: Explore further

When tackling these lab activities, it's always good to stretch yourself by doing some research and attempting some changes on your own.

What other fields in your application need validation? Read about different checks that can be performed with [express-validator](#). A good place to start is to read [express-validator](#). Apply them to the fields that you have identified.

End of lab

Congratulations on completing this lab.

You have successfully implemented validation in your web application!

If this link is broken, please let us know via the [Student Portal](#).