Tampereen ammattikorkeakoulu

Secure Programming

Final report

Tuong Hoang

1 Introduction

This is an old MERN project that I had built in the summer to practice my web development skills and I have been updating some new secure functionalities to it. The website allows customers to see the detail and pre-order dishes, create an account and log in to see all the orders in the past.

The main purpose of this project is about taking the advantage of React such as high usability components, routing different pages to make a magnificent online store with a big and different product ‘data.

In this project, you can interact with the UI and navigate to many pages such as home page, Menu pages, Cart page and so on. Each page was designed and coded with different product data but the same layout. User can add the desired products into cart, modify the amount, delete the product and purchase. Users can also create an account to sign in and change the password if they accidently forgot it.

2 Feartures

2.1 Routing

The below picture is the user interface of homepage. From this main page, user can navigate through other pages to explore many kinds of products and dishes such as sweets, sauces and so on by selecting titles in the white header.

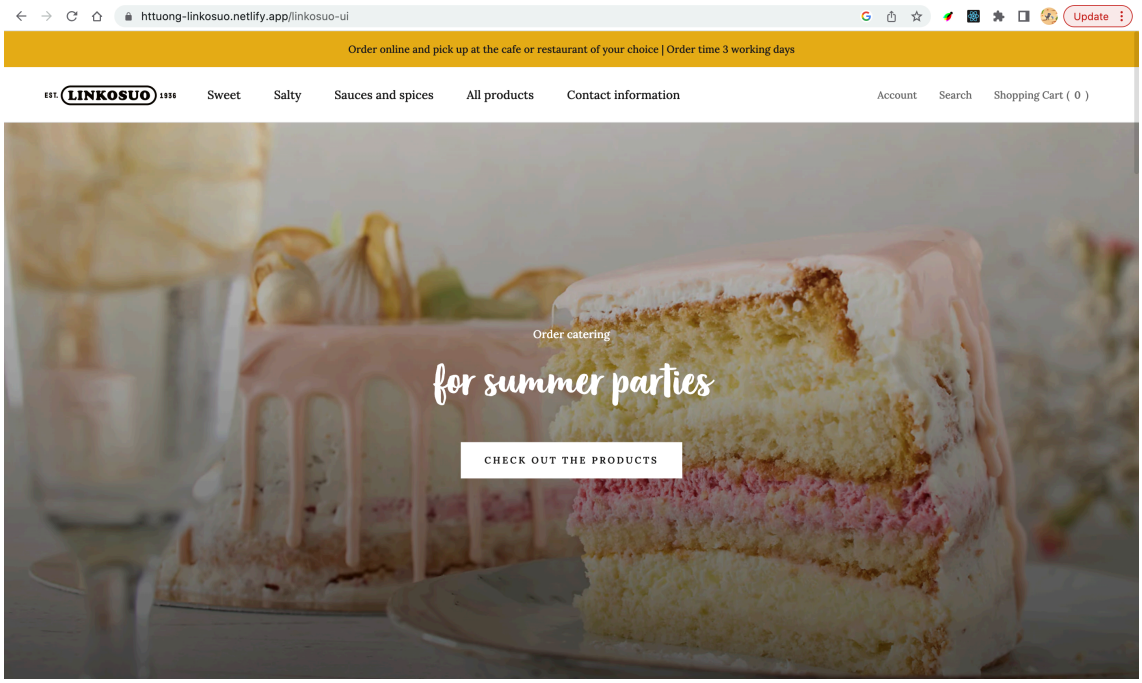


Figure 1. Homepage

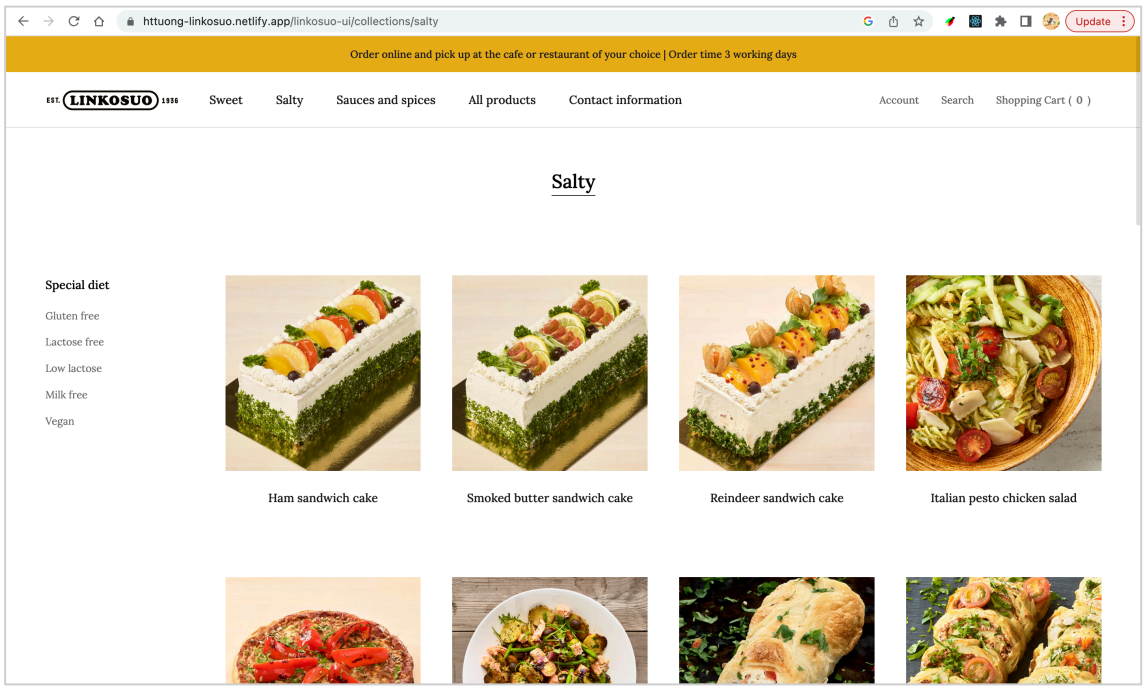


Figure 2. Salty page

2.2 User Interaction

2.2.1 Add product to cart.

User can inspect the detail of certain dishes or products by clicking on the products' image card. This will lead users to the detail page where expose all the information of that product.

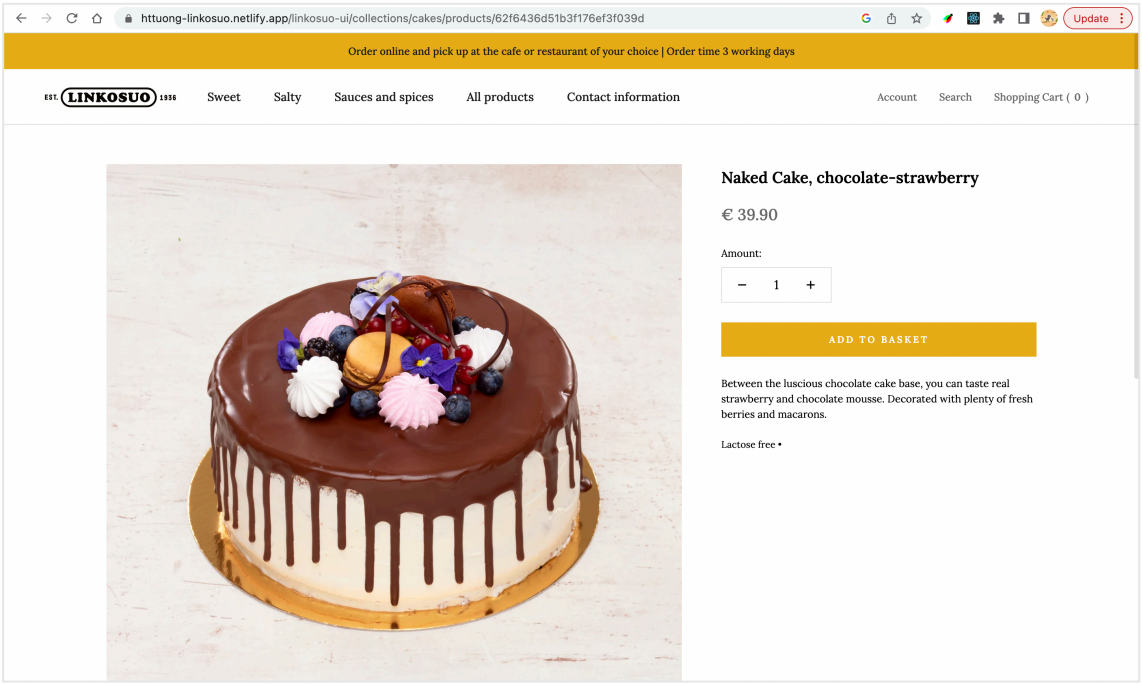
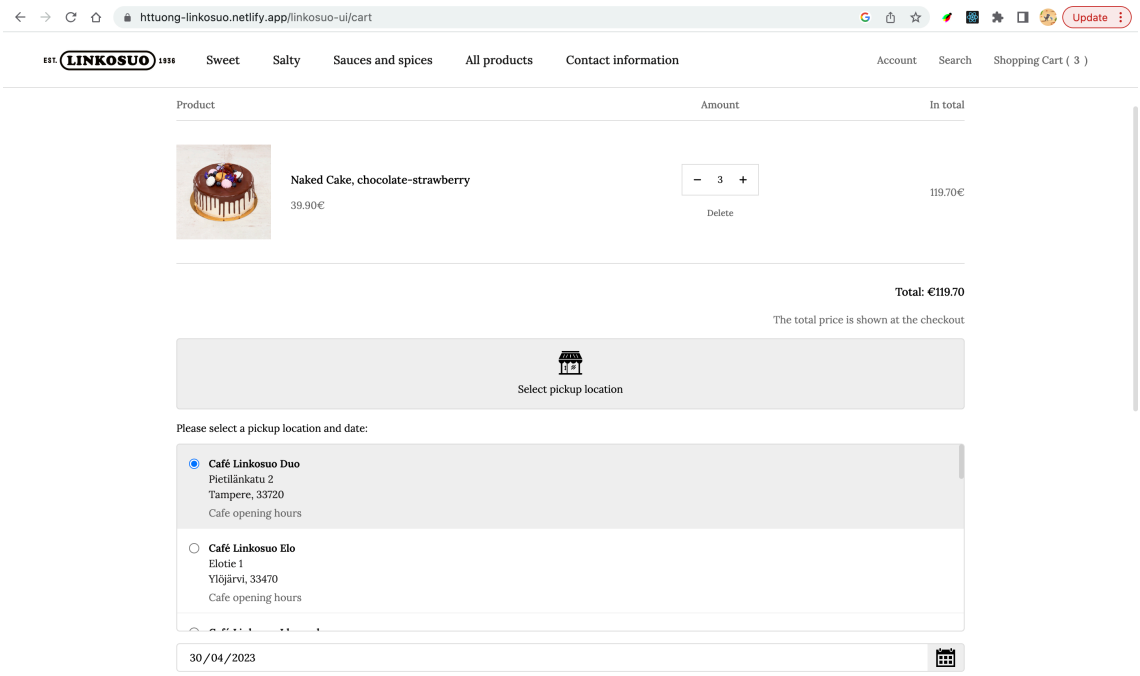


Figure 3. Detail page

From this page, user can increase or decrease the number of desired products and then add to cart for payment.

2.2.2 Make an order.

Users can also see all the added products in the shopping cart. In this page, the number of products can still be modified or deleted conveniently. The pickup location and date have to be chosen so that the restaurant can know where and when users come.



2.2.3 Authentication

Users can create the account for their own usage with email address and password. They can also change the password as desired when forgetting it.

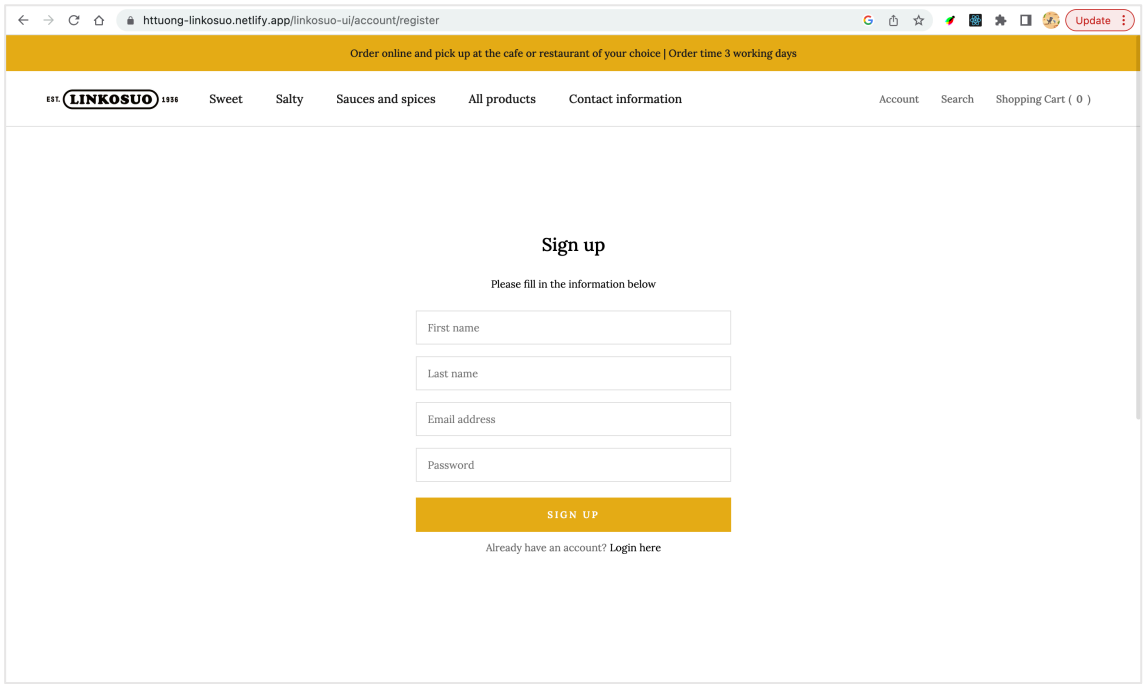


Figure 4. Sign up screen

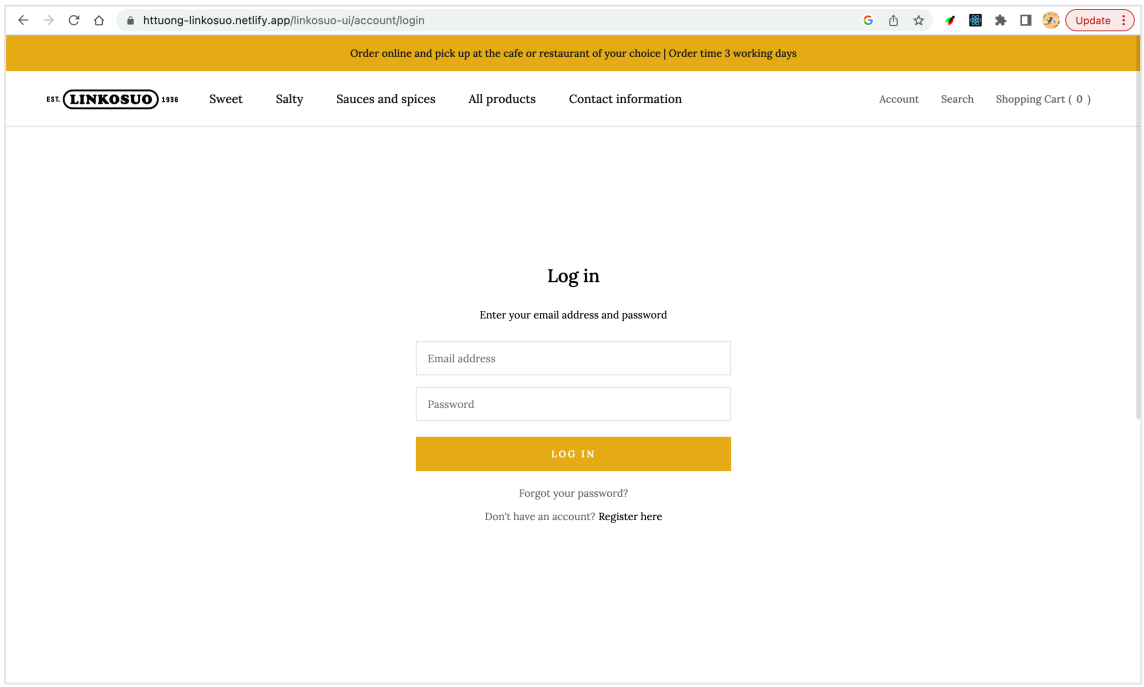
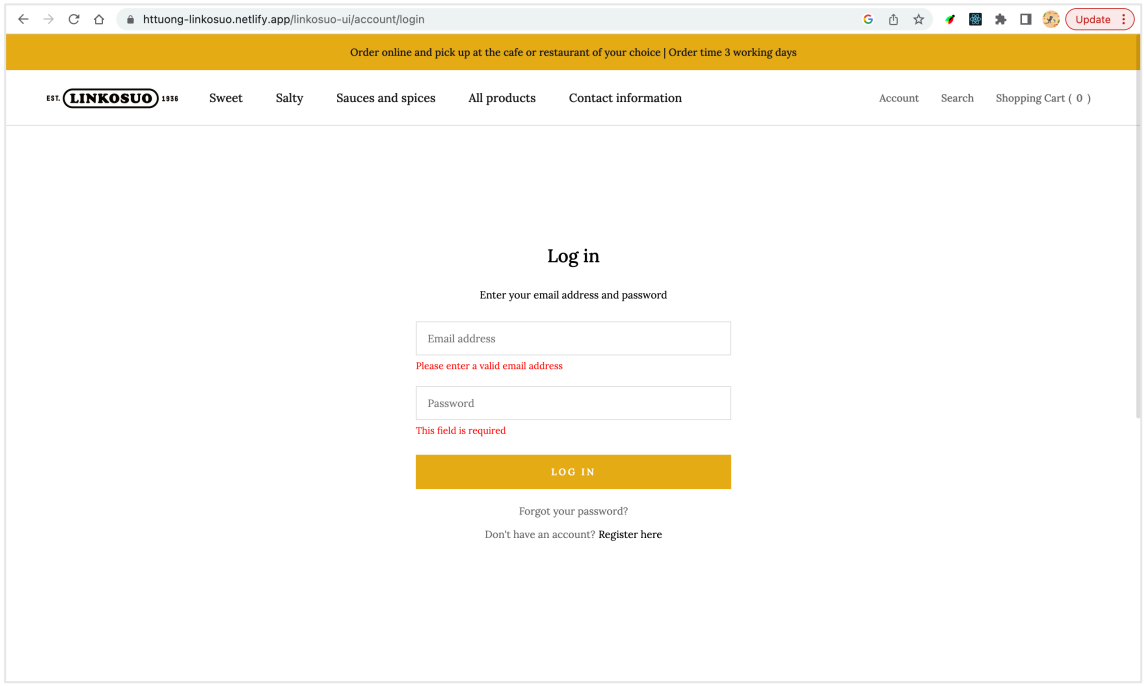


Figure 5. Login screen

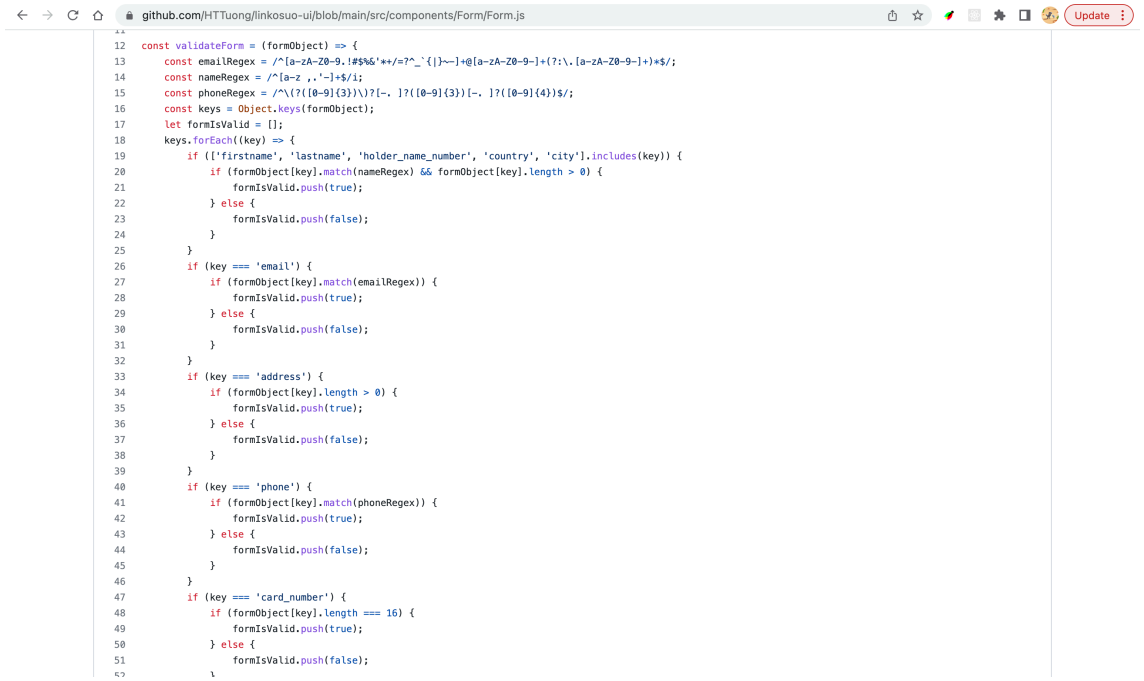
3 Security solutions

3.1 Input validation

In Login and Signup form, each specific input field will have a regex function to validate the typed value. This enhances the security of the form when users submit their value to server and reduce the rate of XSS attack.



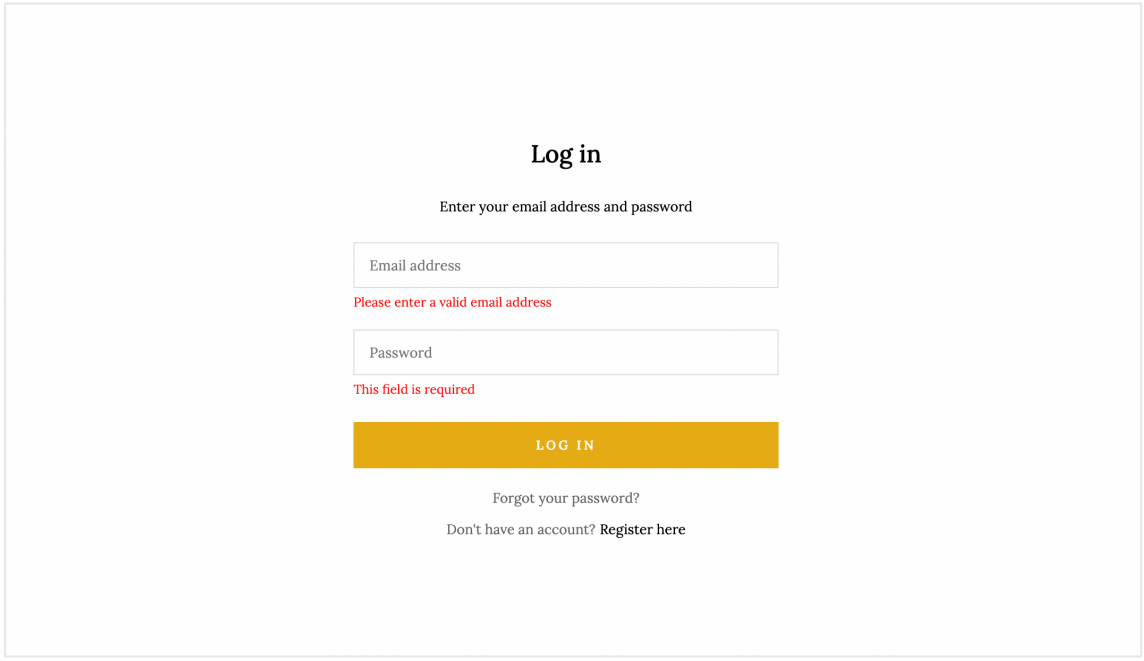
The below picture shows the logic behind this security feature.



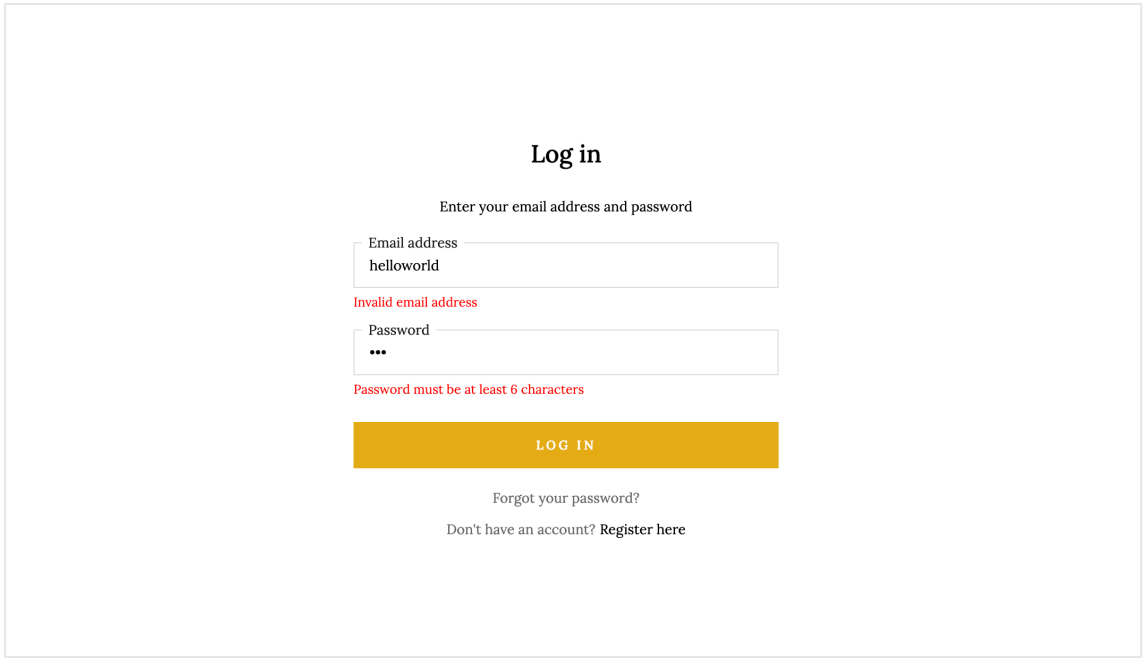
Only when all the required input passes the validation, the form can submit them to server. If one of them is invalid, users will be asked to try another one until the value is right.

Security test

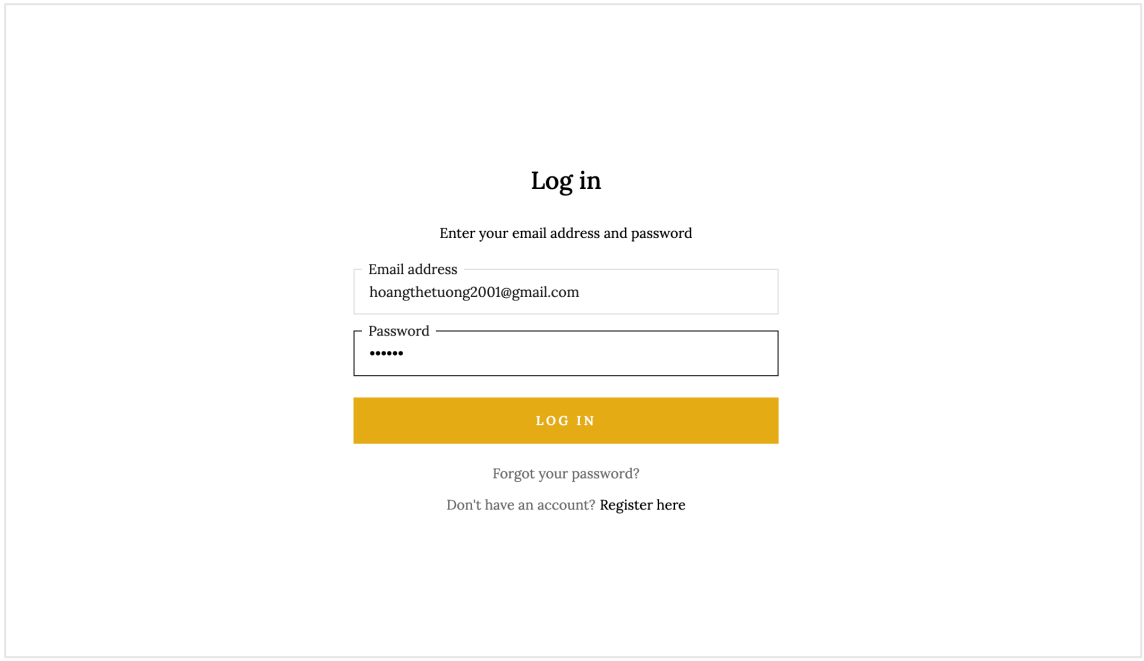
If the required field is empty, the site will notify and warn users to fill out the inputs.



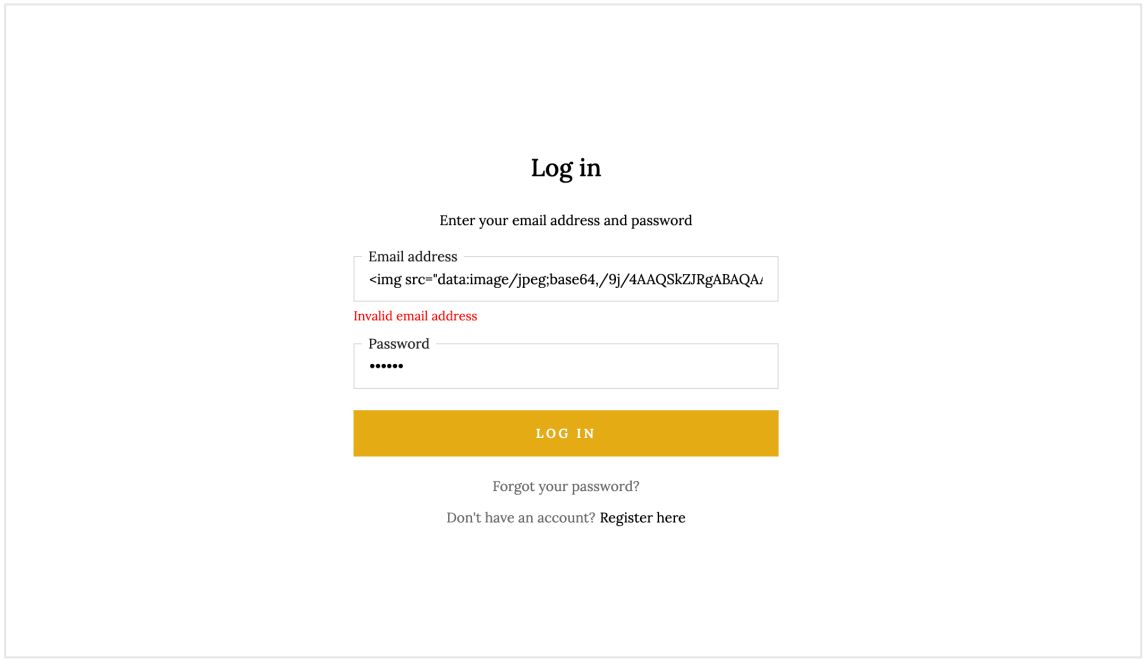
Each input field has a unique validation. The site can check if the entered value is a valid email or password.



When all the input is valid, now users can login or signup to be processed.



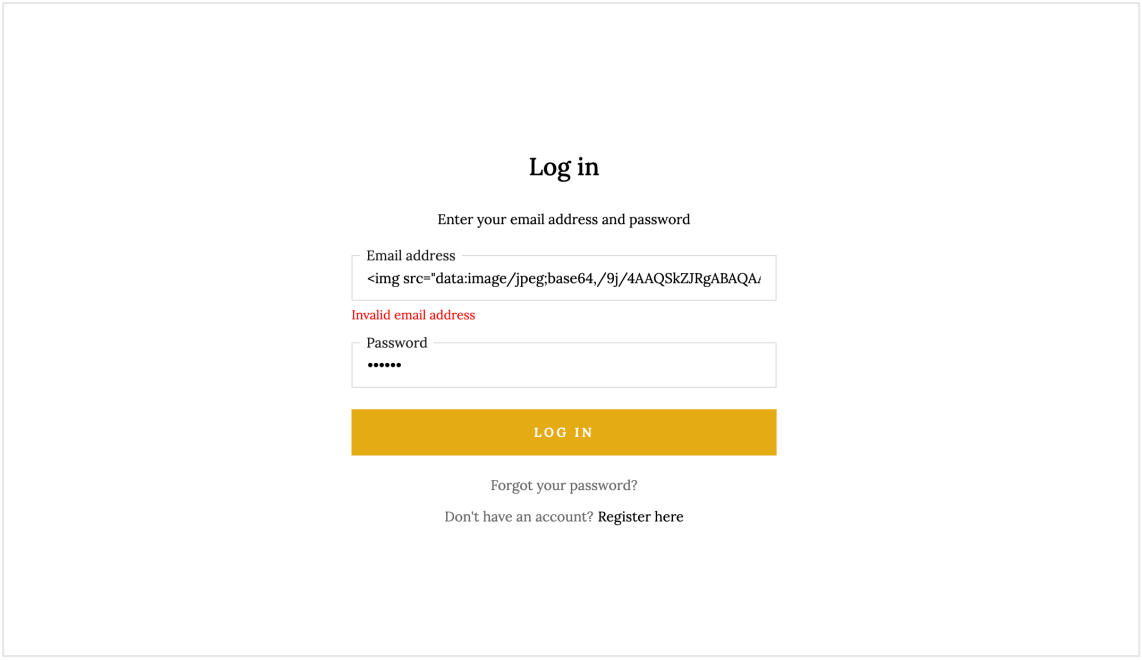
Let’s try a XSS attack by passing a html tag or script into the input field.



In this case, because the validation logic will check if the email is valid based on email regex, the warning still pop on the screen to notify users. However, for better implementation, input sanitization should be applied too.

3.2 Input sanitization

Thank to email regex, invalid value in email field will not be submitted to server although a img tag is passed into the field.



And by default, React DOM automatically escapes any value embedded in JSX before rendering them. Therefore, there is no need to care about input sanitization.

3.3 Server input's validation

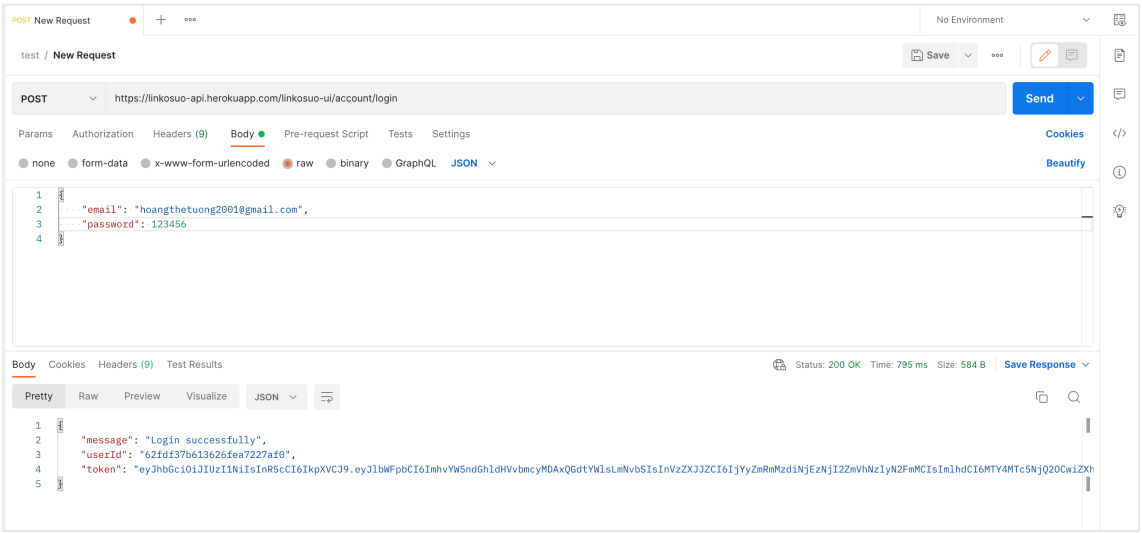
The server of this application was built from NodeJS and Express with many error handling middleware and validation.

For every incoming request, server will check the value in body with the same logic in the front-end side. This is the second security layer and a safe mechanism to make sure that the data is all valid before saving in the database.

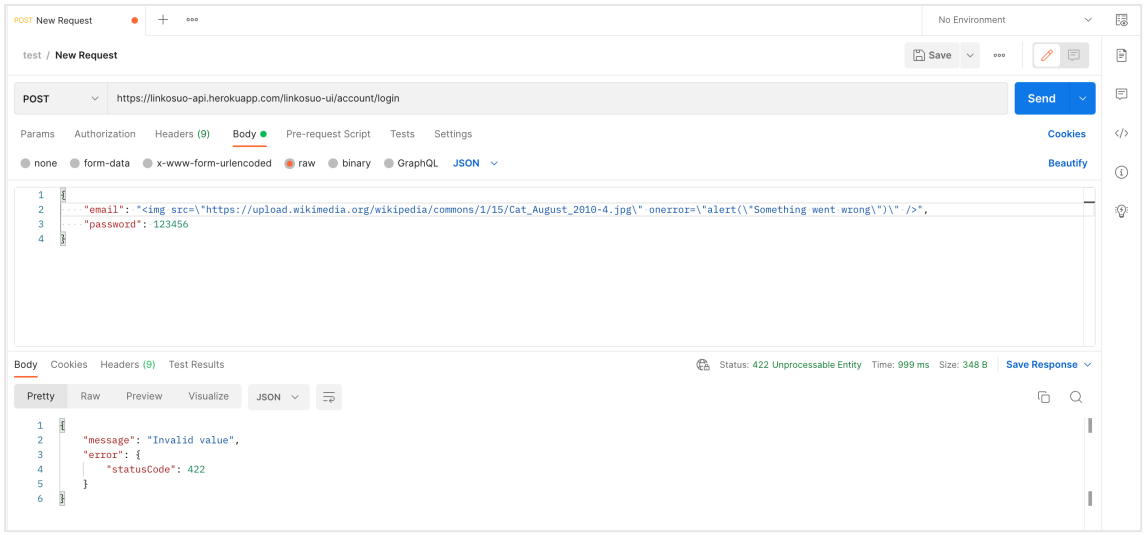
```
// /linkosuo-ui/account

router.post(
    '/register',
    [
        body('lastname').trim().not().isEmpty().withMessage('Invalid last name'),
        body('firstname').trim().not().isEmpty().withMessage('Invalid first name'),
        body('email')
            .isEmail()
            .custom((value, { req }) => {
                return User.findOne({ email: value }).then((user) => {
                    if (user) {
                        return Promise.reject(
                            'This email address is already associated with a customer account. If this account is yours, you can change the password.',
                        );
                    }
                });
            }),
        body('password').trim().isLength({ min: 6 }).withMessage('Password must be at least 6 characters'),
    ],
    authController.signup,
);

router.post(
    '/login',
    [
        body('email')
            .isEmail()
            .custom((value, { req }) => {
                return User.findOne({ email: value }).then((user) => {
                    if (!user) {
                        return Promise.reject('No account associated with this email address');
                    }
                });
            }),
        body('password').trim().isLength({ min: 6 }).withMessage('Password must be at least 6 characters'),
    ],
    authController.login,
);
```

Security test

This is a typical submission of valid values.

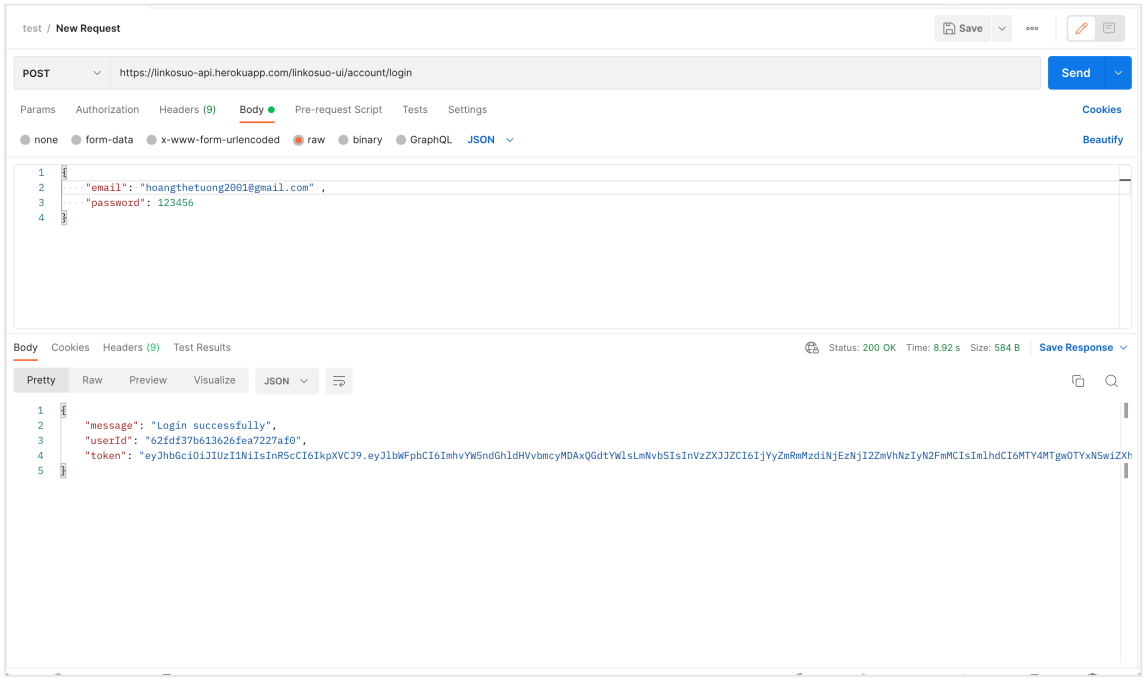


If I try to submit the above value without FE validation, the server can detect the invalid value and send the error response.



3.4 Authorization and JWT

Some of routes and APIs are only available for authenticated user because of JWT. Each incoming request with data to the server will trigger a new JSON web token's generation. This token will be sent back to client for the next request's usage.



4 Security vulnerabilities

First, the leak of password can lead to the unauthorized access to account. Hence, the credential of that account can be changed including the password. However, if hacker want to do stuff like above, they must verify the change password email. Therefore, this vulnerability is not so much serious.

Secondly, if the JSON web token is somehow stolen, it can be attached and sent to server to modify user's information. This is the most difficult scenario for server to detect because the token is the only thing that helps server know who sent the request.

5 Deployment

The front-end of this application is hosted by Netlify while the back-end server is hosted by Heroku.

6 References

Github: https://github.com/HTTuong/Secure-Programming
Netlify: https://app.netlify.com/
Heroku: https://www.heroku.com/
Linkosuo: https://kauppa.linkosuo.fi/