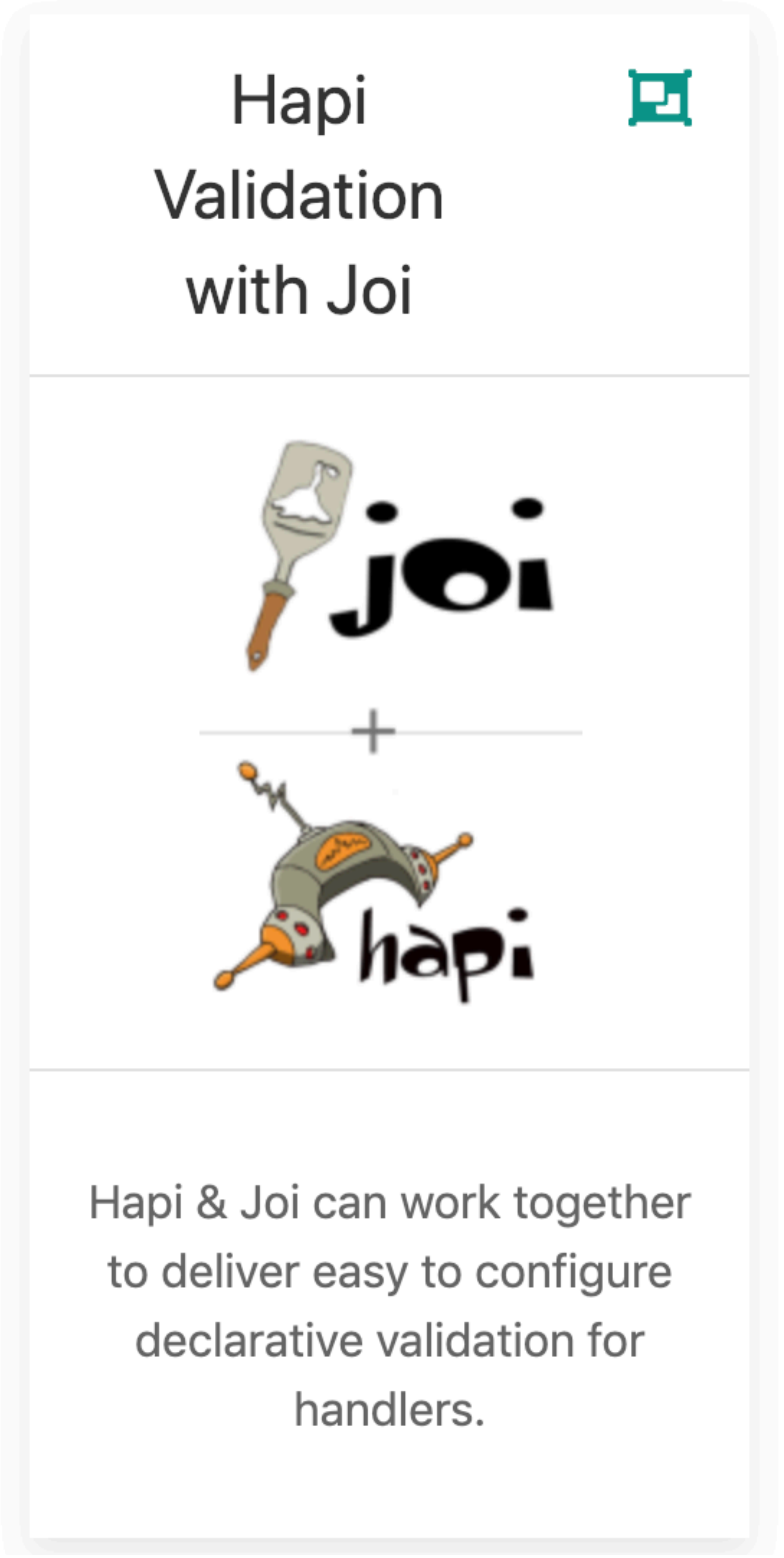
Joi + Hapi Validation

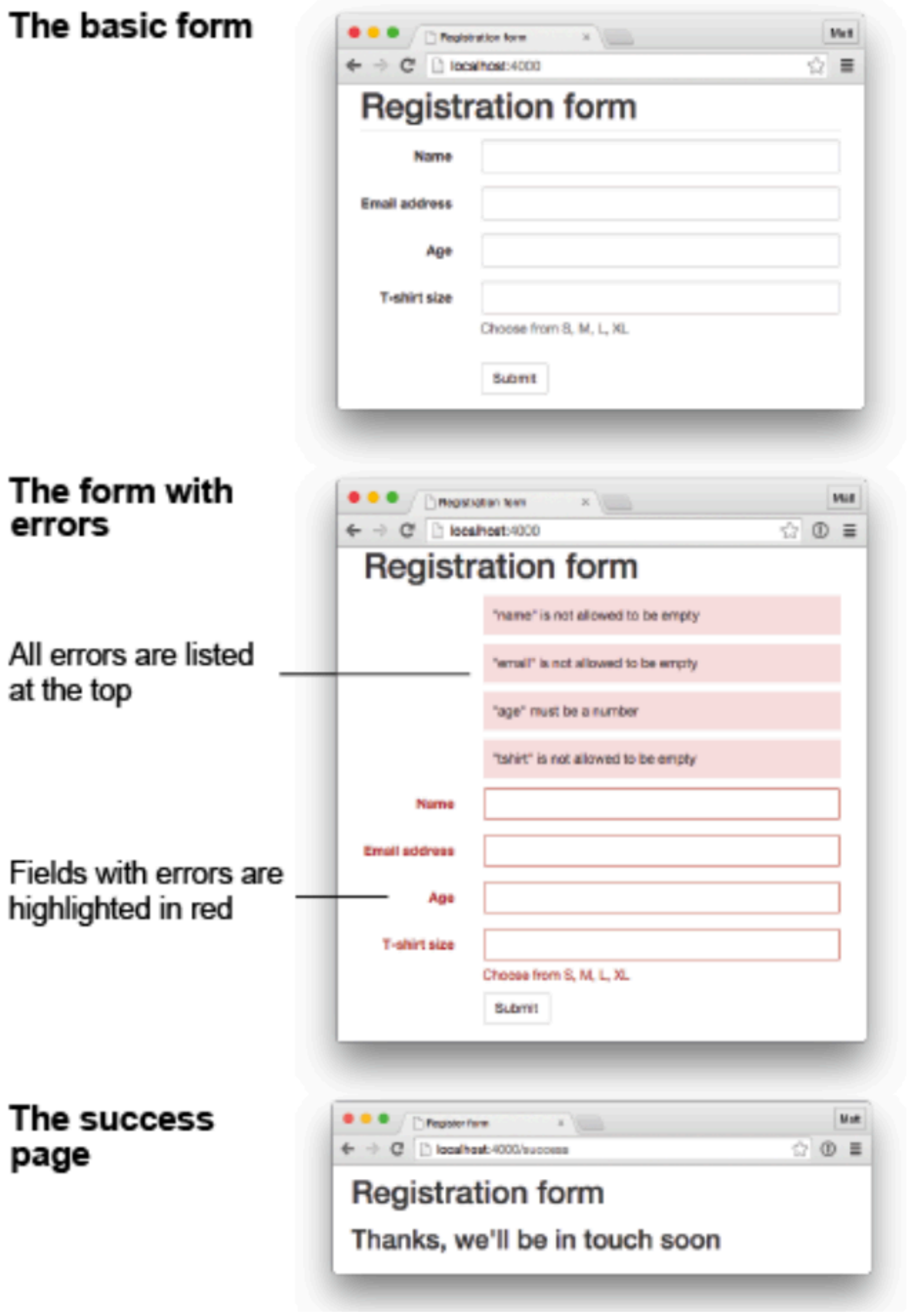


Hapi Validation with Joi

Hapi & Joi can work together to deliver easy to configure declarative validation for handlers.

Agenda

- Error Handling UI Strategies
- Joi Installation
- Joi in Donation



Registration form

Choose from S, M, L, XL

"name" is not allowed to be empty

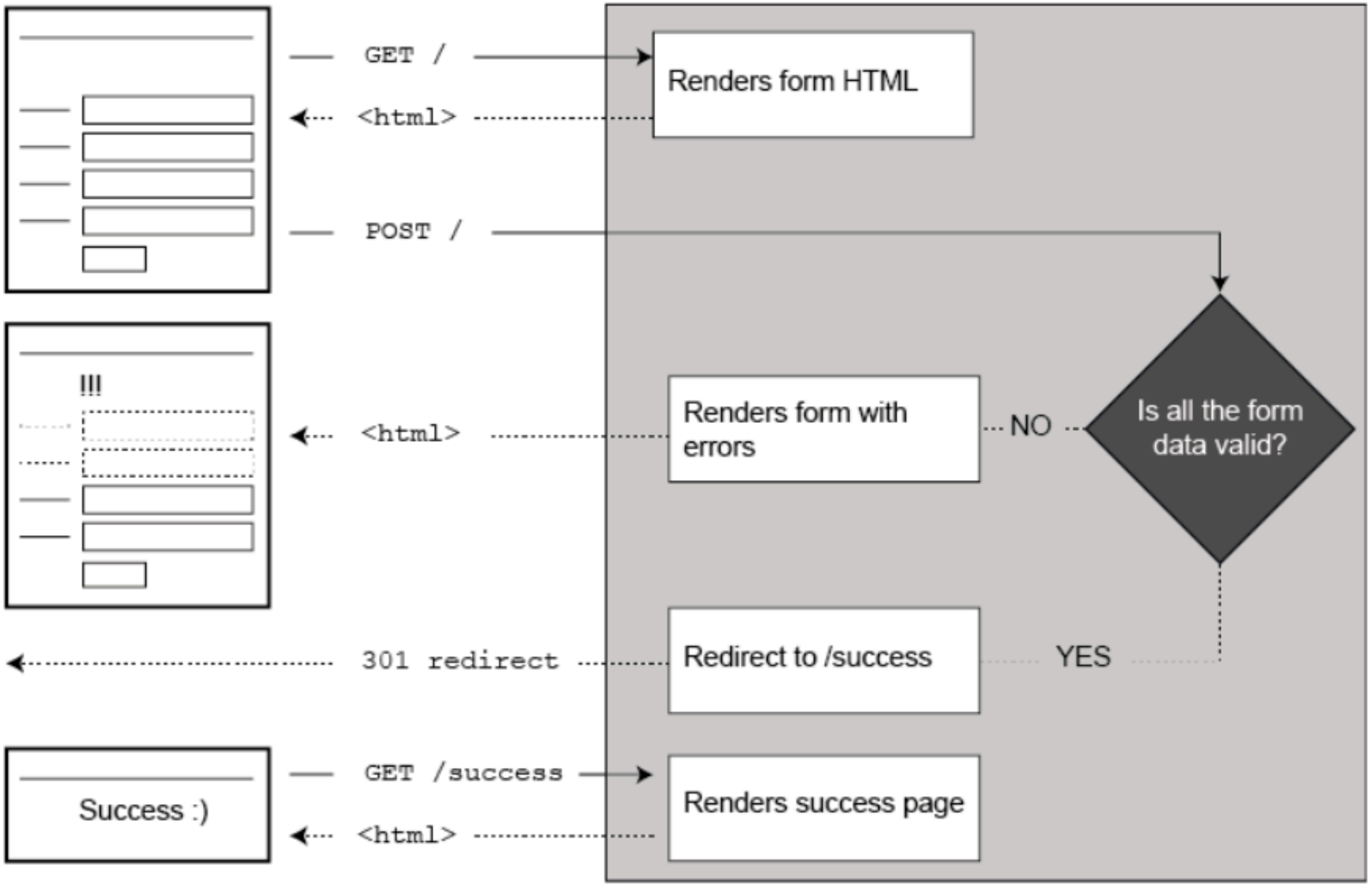
"email" is not allowed to be empty

"age" must be a number

"tshirt" is not allowed to be empty

Thanks, we'll be in touch soon

- Potential Error Reporting Strategy

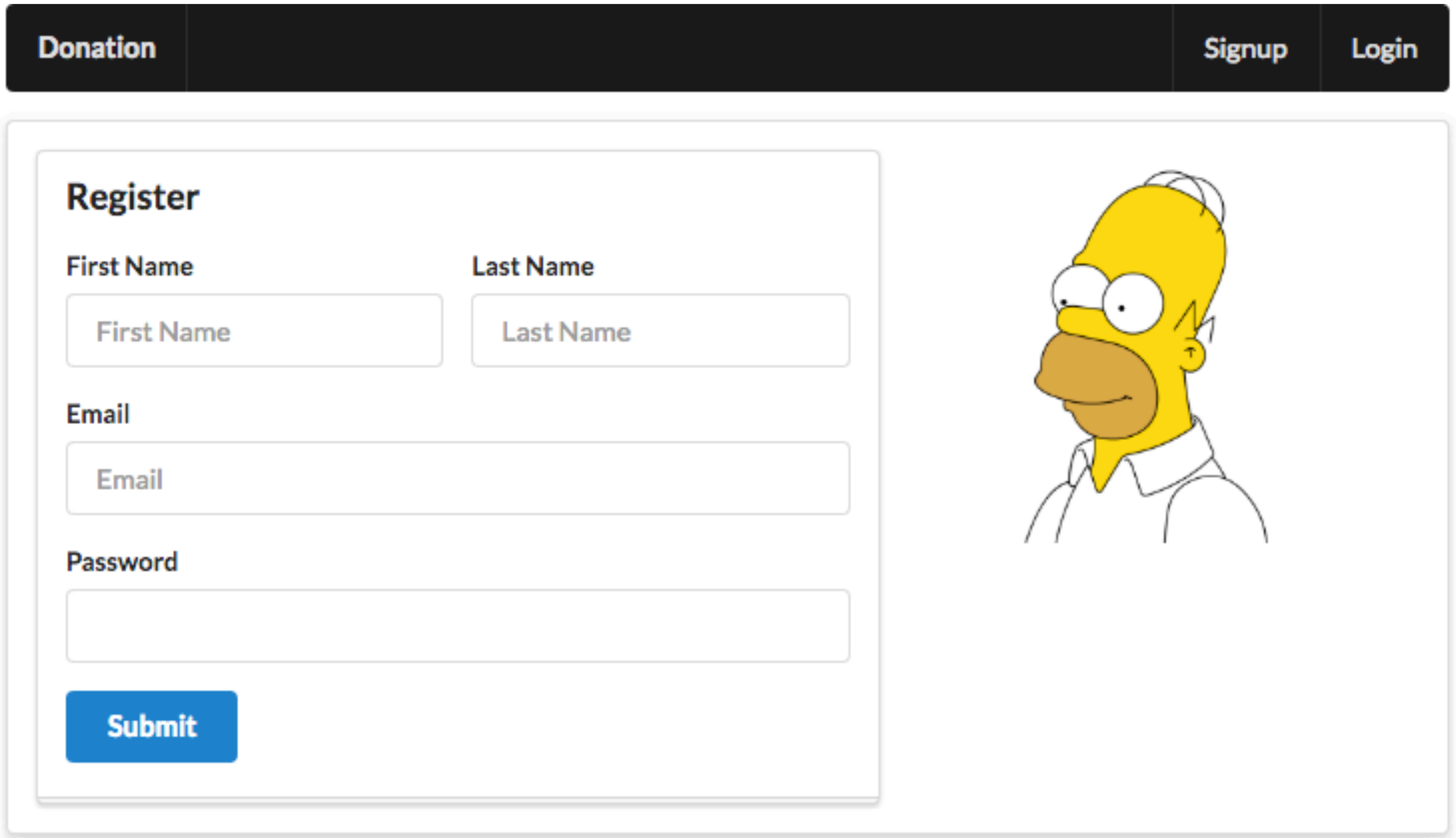

GET /
<html>
Renders form HTML
POST /
!!!
<html>
Renders form with errors
NO
Is all the form data valid?
301 redirect
Redirect to /success
YES
Success :)
GET /success
Renders success page
<html>

Install Joi

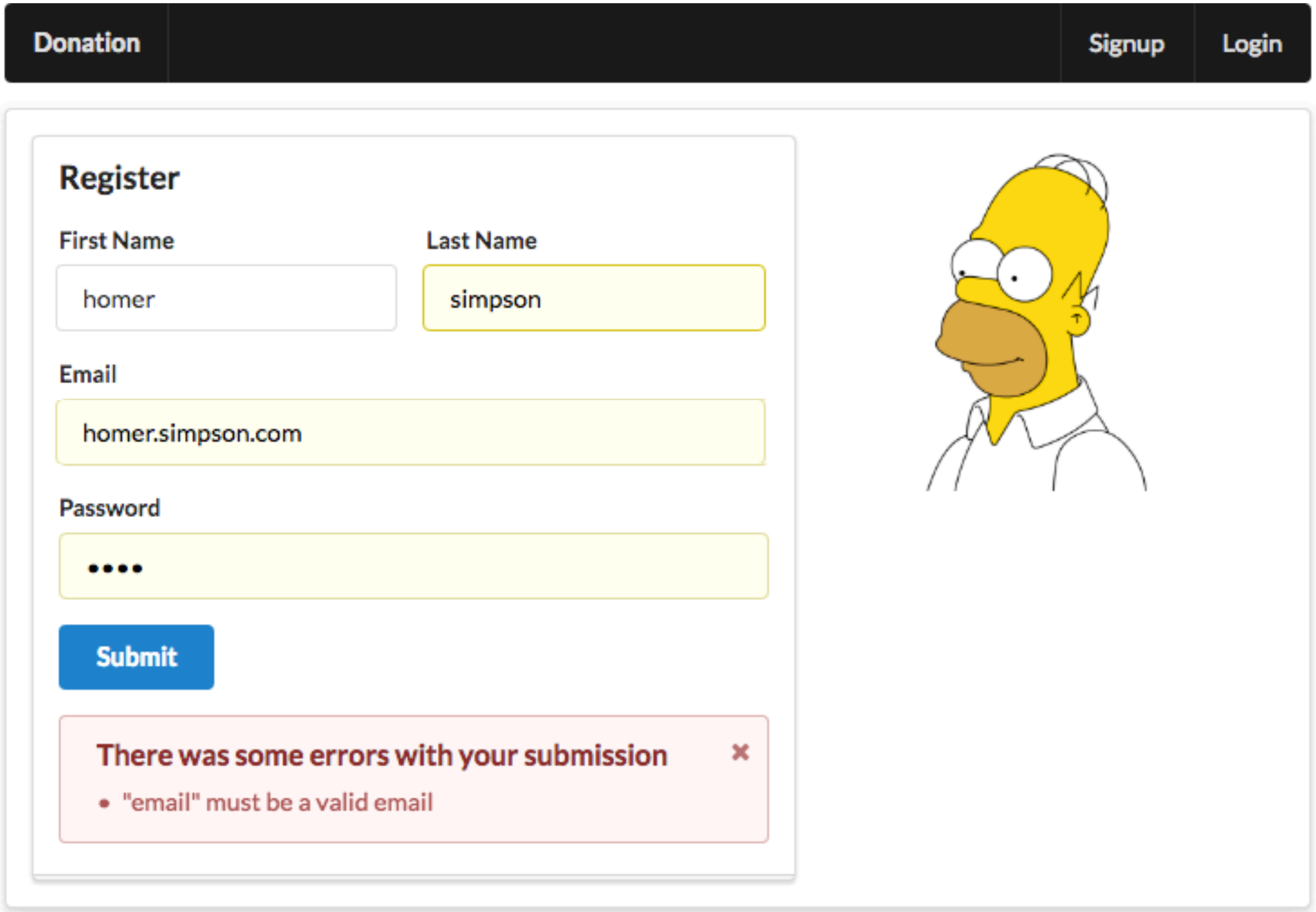
```
npm install joi
```

```
{
  "name": "donation-web",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "boom": "^7.3.0",
    "dotenv": "^6.2.0",
    "handlebars": "^4.0.12",
    "hapi": "^18.0.0",
    "hapi-auth-cookie": "^9.1.0",
    "inert": "^5.1.2",
    "joi": "^14.3.1",
    "mongoose": "^5.4.7",
    "vision": "^5.4.4"
  },
  "devDependencies": {
    "prettier": "^1.16.0"
  },
  "prettier": {
    "singleQuote": true,
    "printWidth": 120
  }
}
```

Joi in Donation



```
const schema = {
  firstName: Joi.string().required(),
  lastName: Joi.string().required(),
  email: Joi.string().email().required(),
  password: Joi.string().required(),
},
```



Register

First Name	Last Name
homer	simpson

Email

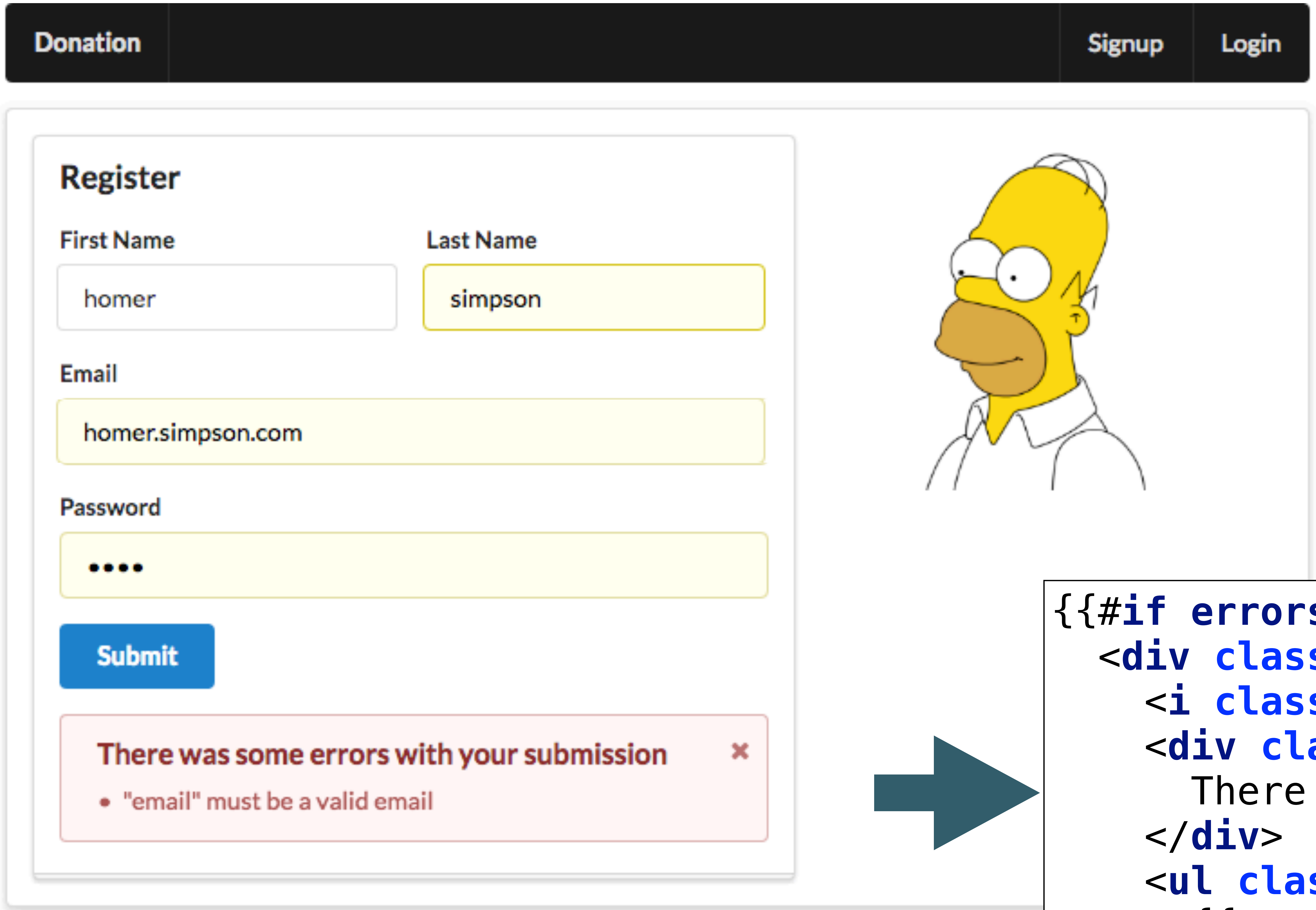
homer.simpson.com

Password

••••

There was some errors with your submission

- "email" must be a valid email



There was some errors with your submission

- "email" must be a valid email

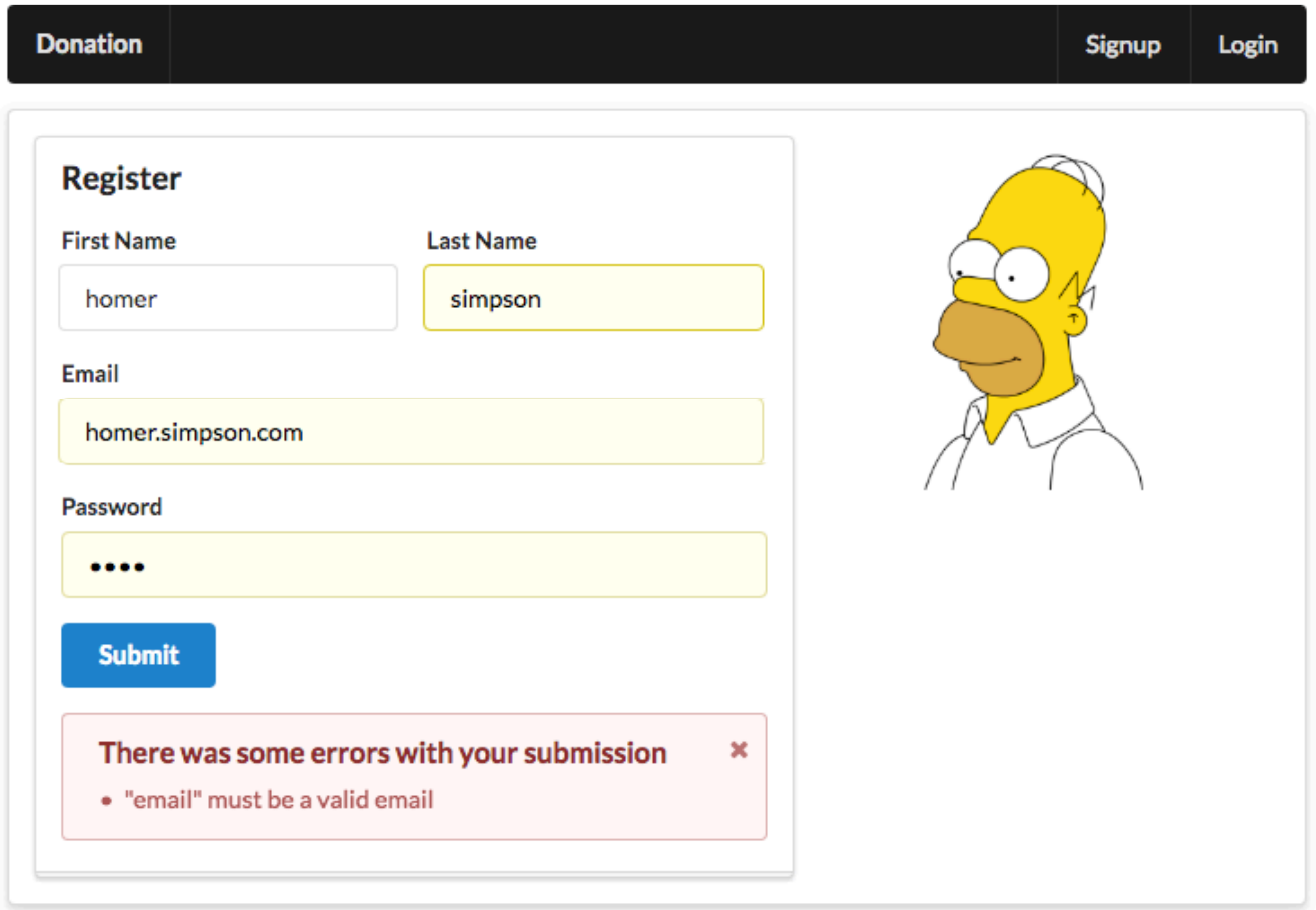
error.hbs

```
{{#if errors}}
  <div class="ui negative message transition">
    <i class="close icon"></i>
    <div class="header">
      There was some errors with your submission
    </div>
    <ul class="list">
      {{#each errors}}
        <li>{{message}}</li>
      {{/each}}
    </ul>
  </div>
{{/if}}
```

signup.hbs

```
{{> welcomemenu }}

<section class="ui raised segment">
  <div class="ui grid">
    <div class="ui ten wide column">
      <div class="ui stacked fluid form segment">
        <form action="/register" method="POST">
          <h3 class="ui header">Register</h3>
          <div class="two fields">
            <div class="field">
              <label>First Name</label>
              <input placeholder="First Name" type="text" name="firstName">
            </div>
            <div class="field">
              <label>Last Name</label>
              <input placeholder="Last Name" type="text" name="lastName">
            </div>
          </div>
          <div class="field">
            <label>Email</label>
            <input placeholder="Email" type="text" name="email">
          </div>
          <div class="field">
            <label>Password</label>
            <input type="password" name="password">
          </div>
          <button class="ui blue submit button">Submit</button>
        </form>
        {{> error }}
      </div>
    </div>
    <aside class="ui five wide column">
      <img src="images/homer3.png" class="ui medium image">
    </aside>
  </div>
</section>
```

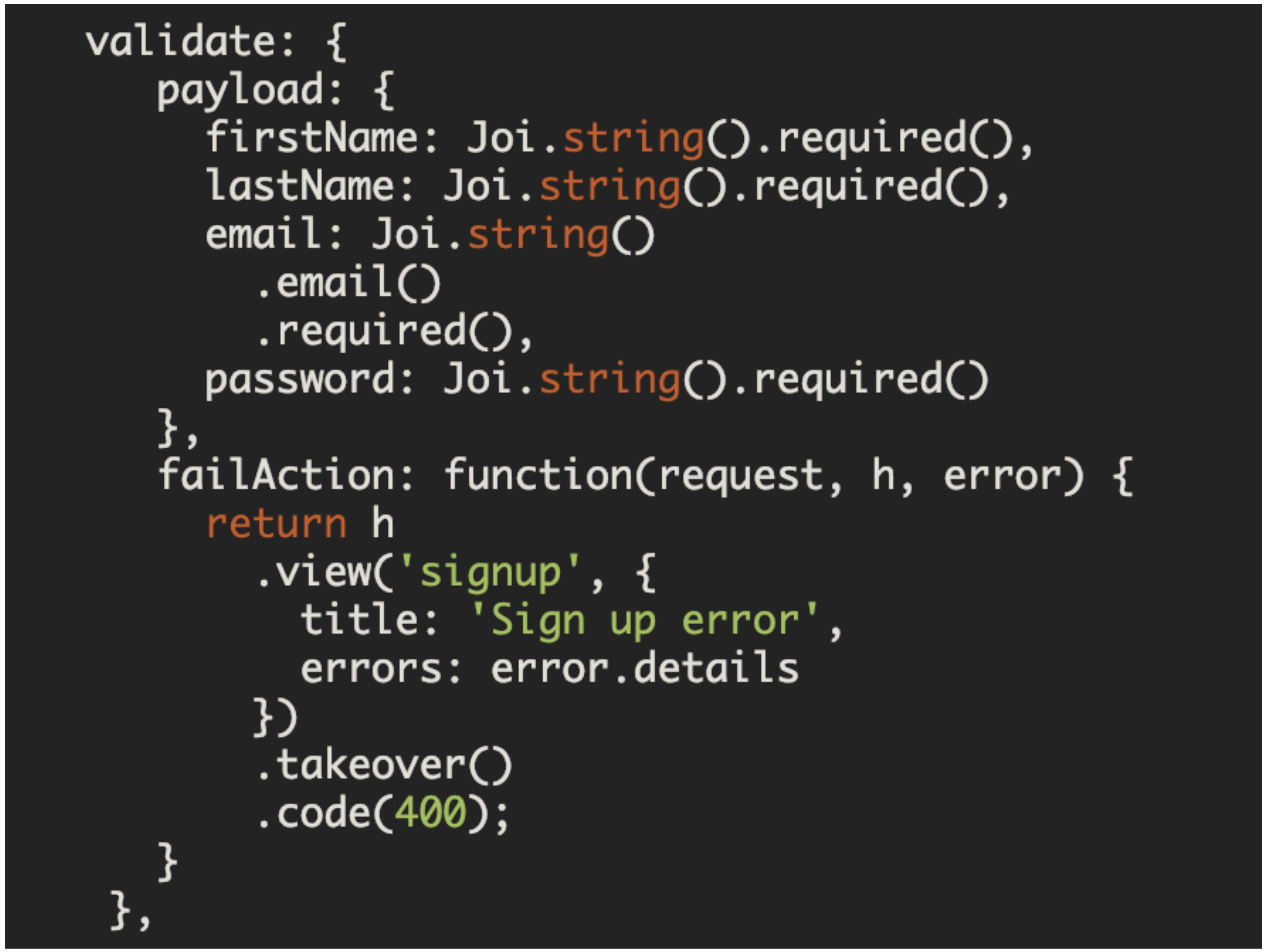


There was some errors with your submission

- "email" must be a valid email

```
{{#if errors}}
  <div class="ui negative message transition">
    <i class="close icon"></i>
    <div class="header">
      There was some errors with your submission
    </div>
    <ul class="list">
      {{#each errors}}
        <li>{{message}}</li>
      {{/each}}
    </ul>
  </div>
{{/if}}
```

validate Property in Hapi handler



```
validate: {
  payload: {
    firstName: Joi.string().required(),
    lastName: Joi.string().required(),
    email: Joi.string()
      .email()
      .required(),
    password: Joi.string().required()
  },
  failAction: function(request, h, error) {
    return h
      .view('signup', {
        title: 'Sign up error',
        errors: error.details
      })
      .takeover()
      .code(400);
  }
},
```

- payload: This defines a schema which defines rules that our fields must adhere to.
- failAction: This is the handler to invoke of one or more of the fields fails the validation

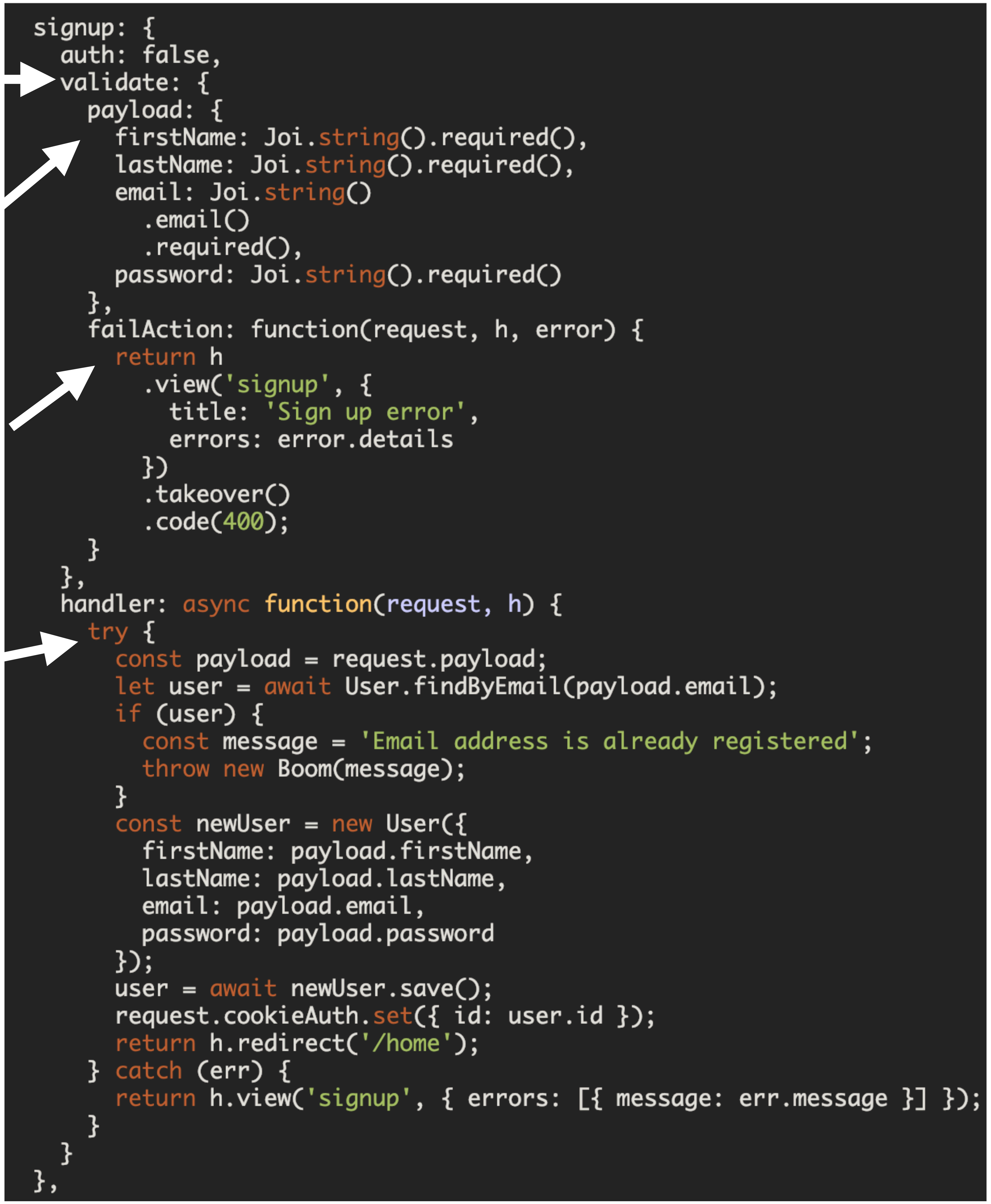
enable validation for this handler

the joi schema - tied to payload from request

handler if validation fails

the main handler for register, now validated

signup handler



```
signup: {
  auth: false,
  validate: {
    payload: {
      firstName: Joi.string().required(),
      lastName: Joi.string().required(),
      email: Joi.string()
        .email()
        .required(),
      password: Joi.string().required()
    },
    failAction: function(request, h, error) {
      return h
        .view('signup', {
          title: 'Sign up error',
          errors: error.details
        })
        .takeover()
        .code(400);
    }
  },
  handler: async function(request, h) {
    try {
      const payload = request.payload;
      let user = await User.findByEmail(payload.email);
      if (user) {
        const message = 'Email address is already registered';
        throw new Boom(message);
      }
      const newUser = new User({
        firstName: payload.firstName,
        lastName: payload.lastName,
        email: payload.email,
        password: payload.password
      });
      user = await newUser.save();
      request.cookieAuth.set({ id: user.id });
      return h.redirect('/home');
    } catch (err) {
      return h.view('signup', { errors: [{ message: err.message }] });
    }
  }
},
```