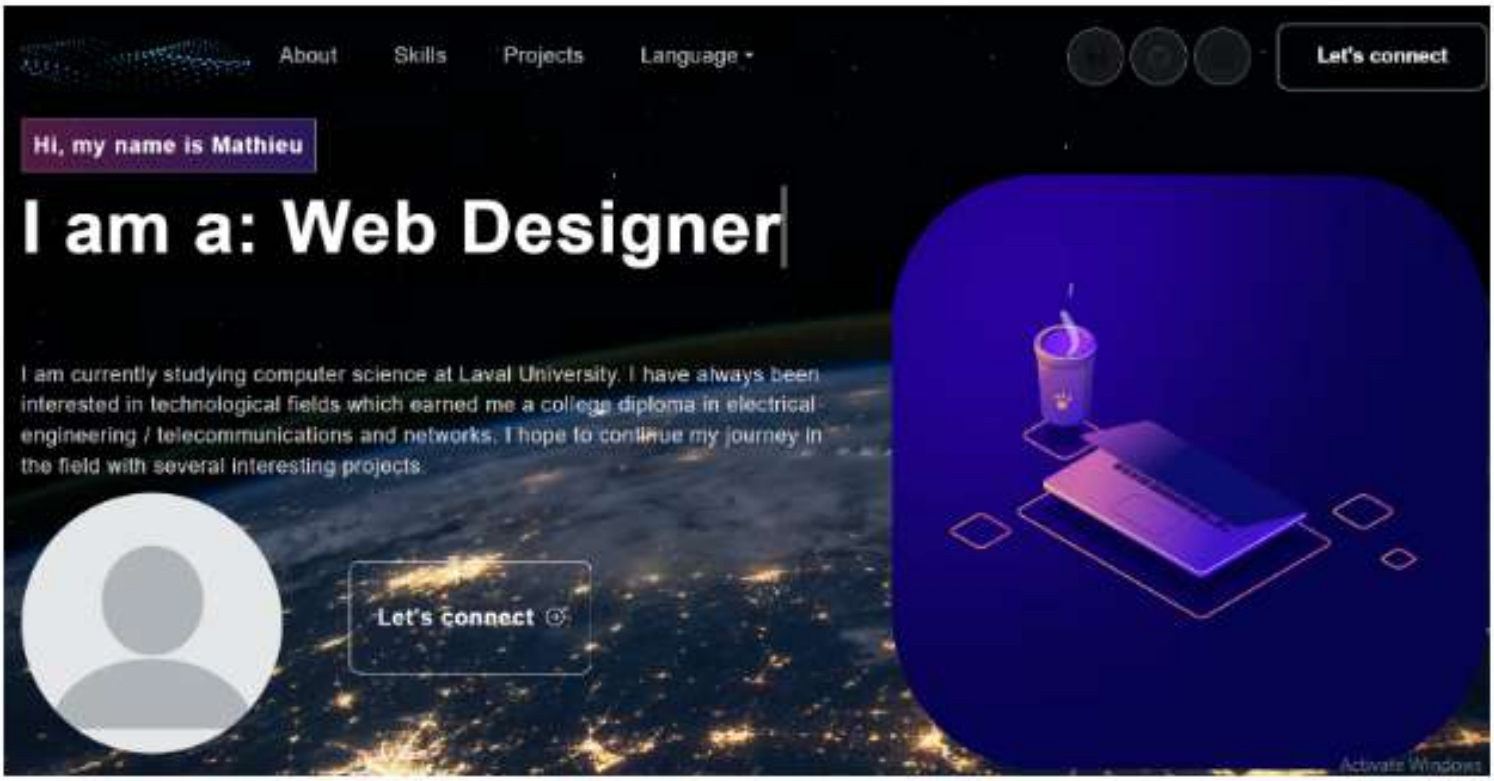
Portfolio



www.mathieuchretien.com

This is my career portfolio created in React. I use AWS amplify to host this website.

Dependency

- Frontend
  "@aws-amplify/ui-react": "^5.0.4" (Contact form input)
  "aws-amplify": "^5.3.3" (config AWS)
  "bootstrap": "^5.3.0" (UI framework css)
  "i18next": "^23.2.7" (Translation framework json file manager)
  "lottie-react": "^2.4.0" (Animation framwork)
  "react": "^18.2.0" (React framework)
  "react-bootstrap": "^2.8.0" (UI framework)
  "react-dom": "^18.2.0" (DOM renderer)
  "react-i18next": "^13.0.1" (Translation framework json file manager)
  "react-lazy-load-image-component": "^1.6.0" (lazy-load picture blur)
  "react-pdf": "^7.1.2" (PDF reader framework)
- Backend
  "aws-sdk": "^2.1413.0" (Service aws)
  "nodemailer": "^6.9.3" (Easier mail backend support attachement)

Frontend

All custom CSS is in App.css

- Main Componant
  NavBar
  Banner
  Skills
  Project
  ContactForm
  Footer
- Modal Componant
  CV in (Navbar, Contactform, Footer)
  ESP in (Project)
  Algo in (Project) Portfolio in (Project)
- Animation Componant
  AnimationBanner in (Banner lottie animation)
  AnimationContact in (Contactform lottie animation)
  TexteBanner in (Banner text animation)
  Note: The animation in the navbar and footer use a gif because lottie was creating a DOM too big (over 12,000 element).

Backend

The only backend needed for my website was the contact form for sending me email. To do so I used a graphql mutation link to a lambda function in AWS.

Here is the shema:

```
type Sender @model @auth(rules:[{allow:public, operations:[create]}]) {
  id: ID!
  name: String!
  email: String!
  phone: String!
  message: String!
}
```

Here is the Javascript function:

```
/* Amplify Params - DO NOT EDIT
        ENV
        REGION
        SES_EMAIL
Amplify Params - DO NOT EDIT */

/**
 * @type {import('@types/aws-lambda').APIGatewayProxyHandler}
 */
const aws = require('aws-sdk');
const nodemailer = require('nodemailer');

const ses = new aws.SES()
let transporter = nodemailer.createTransport({SES: ses})

exports.handler = async(event) => {
  for (const streamedItem of event.Records) {
               if (streamedItem.eventName == 'INSERT') {
                       //pull off items from stream
                       const userName = streamedItem.dynamodb.NewImage.name.S
                       const userEmail = streamedItem.dynamodb.NewImage.email.S
                       const userPhone = streamedItem.dynamodb.NewImage.phone.S
                       const userMessage = streamedItem.dynamodb.NewImage.message.S


      try{
        const mailFormat = await transporter.sendMail({
          from: process.env.SES_EMAIL,
          to: process.env.SES_EMAIL,
          subject: 'Portfolio message ',
          html:`<p> Name: ${userName}<br /> Email: ${userEmail}<br /> Phone: ${userPhone}<br /> Message: ${us
        })
        return mailFormat;
      } catch(e){
        console.error('ERROR', e)
      }
    }
  }
  return 'done';
}
```

With this when a qraphql is created the function will automatically trigger the node.js function. (Trigger Lambda)
Note: Email must be a verified email in the Simple Email Service of AWS. (SES_EMAIL = email of choice)