WDI-Project-4

This was my fourth project for WDI London

Introduction

This app was an individual project, built by me over six days and was the fourth of four projects during our time on the Web Development Immersive course at General Assembly, London. The requirements of this project were to build an authenticated RESTful application using either a MEAN stack a Rails stack or an Angular front-end with a Rails API. I took this opportunity to have a closer look at Rails and built an Angular front with Rails API.

The App

LexMachine is an online application designed to help the user expand their vocabulary. For the past year or so, if I have found a word which I find interesting and I am unsure of the meaning, I will look up the definition and email it to myself. I now have a list of interesting and slightly unusual words mostly gathered

from my favourite books and articles. But, I thought, there must be an easier way... This is why I designed the Lex Machine It is an authenticated application which simply allows users to store a list of words and their meanings.

How to use the site

When registered and logged in the user will be invited to add words to their words index. For each letter typed into the text input on the words index page a dictionary API is called meaning that the user can select words from the auto complete options. This also means that words which have homonyms and/or are used in different parts of speech can be filtered out at this stage. Once a word is selected and submitted the definition, example sentences and other etymological information obtained from the dictionary API and that word's page is populated with the relevant information. The user is also able to add their own notes on each individual word such as their own example sentences or the context in which they first heard it. The idea is that a user is able to quickly look up and store information on any word they wish and compile their own personal dictionary.

How it was built

LexMachine uses an Angular/Express application on the front-end with a Rails API on the back-end. It was made using Bootstrap CSS framework, SASS, and ES6. User authentication was achieved using JWT and Bcrypt. I made use of Bower to manage front end packages, NPM to manage back end packages, Babel to minify and transpile SCSS and ES6 and Gulp as our task runner.

Challenges during the build

As this was the first time I had built a Rails API/SQL back-end (my previous experience was with MongoDB) I found this in itself challenging. Stepping outside of my 'MEAN stack comfort zone' also meant that linking up the front and back-ends was challenging. Having said that I can certainly see the benefits of using the Rails generator and scaffold features over the MongoDB approach. Another problem I encountered was with the dictionary API. Although it returned lots of useful data in JSON, it was structured in a rather strange way and extracting what I proved difficult.

Getting started

These steps are required in order to run the application:

1. Ruby, Rails, Gulp and Node must be installed
2. The front and the back end of the application are in two different Git repositories named WDI-Project-4-frontend and WDI-Project-4-backend.
3. Install all the relevant NPM and Bower packages by running the commands `npm i` and `bower i` in the root directory WDI-Project-4-frontend.
4. Install the relevant Gems by running the command `bundle` when in the root directory WDI-Project-

4-backend.

5. Populate the database by running the following command in the root directory WDI-Project-4-backend: `rails db:drop db:create db:migrate db:seed`
6. While in the root directory WDI-Project-4-backend run `rails s` to start up the rails server.
7. In a separate tab in the terminal, in the root directory WDI-Project-4-frontend run the command `gulp`.
8. Go to http://localhost:3000 in your browser to view the site.
9. NOTE if the website does not appear to be working correctly please try restarting Gulp in the front-end directory.

Credits

- The font used throughout the site was taken from Google Fonts
- The background picture was taken from Unsplash
- The dictionary API used was Pearson Dictionaries API the documentation for which can be found here
- GA instructors Alex Chin and Rane Gowan