Website Project Report

Houkai Qian 260849921
Xuechen Wang 260833548
Group 15

General introduction

This project was made by Houkai Qian and Xuechen Wang. The project is to redesign McGill computer science website. Most of page in this project was initially wrote in NextJs framework, then we noticed that NextJs which is a server-side rendering framework doesn't really work for our project. We recreated our site in React for front end in the end, and the transition cost us a lot of time. We use NodeJS for our backend and Mongodb Atlas for our database, all the data including user information and post contents are store online in www.mongodb.com.

Front end

TECH STACK

React is the framework we used for front end, it uses single page structure. Our website switches component controlled by route to achieve displaying different content when user interact with our website. There are more than 40 components added in our project. Bootstrap is imported for styling our website, in addition we also coded hundreds of css style rules manually.

DESIGN THEME

Our website need to be designed for demand. The user portrait of https://www.cs.mcgill.ca will most likely be the non-McGill students/ researchers who wants to learn about McGill Computer Science department, since McGill students and professors don't have the demand on going to this website regularly for information. Therefore, the main focus of this website designs will be to display useful information simple and clean.

The current design of cs.mcgill.ca has many problems. First, the white-grey color theme is boring and unattractive. Second and also the biggest problem, there is little rich media like photos but too much text, it can make the whole website boring and hard to read.

We first designed the layout and style and made a prototype:

https://www.figma.com/file/uzfWMR4ptT9S4lwkovyjxZ/COMP-307-Project?node-id=0%3A1

Then we code the front end following the prototype. We didn't use any template except bootstrap, codes are all coded by ourself from scratch.

RGB(230,0,0) is chosen to be the website's theme color, it is red close to the color of McGill logo, but it's not pure red since pure red is too bright and too attractive. The main color used in background will be white. White background can maximize the contrast, let user focus on the main content. There is also a small design element acts like a "symbol" of our website, that is a small underline under the important texts like titles to make them outstanding.

Layout of our website was referenced by the White House website. A big carousel was placed on the homepage shows most exciting content to users. Then a login form follows, it allows user

who are not coming for information but the other service to sign in right away. Then is a group of cards displaying collection of information. This kind of layout could display different kind of information in an interesting and attractive way.

USER EXPERIENCE

To make a clean and simple website outstanding, user experience is the key for us. Element in our page only resize when the browser width is less than 1510px. When the browser is wider than 1510px, only the white space aside increases so the layout will not be destroyed if user are using a wide screen, and the content will remain in the middle of the page so users do not need to turn head around to see all the content. This page works on all screen size including phone and table. Other details like, the red underline will always align with the end of text not depending on the length of the text; the user icon will direct you to different page depending on the login state, all are what we are proud of.

Back end

TECH STACK

We use Express.js as the framework of our backend app and MongoDB as our database. We also added an node API Mongoose to help write database controlling codes.

The database we use for this website contains two collections, “users” and “updates”. Documentations in “users” stores our user information. User documentations have properties username, password, and email. This collection was used for user login and sign up function. “Updates” collection stores the articles on our website. Documentations in “updates” has properties type, title, author, and txt, where title, author and txt corresponds to the article title, author’s name and the content. “Type” is also significant since it decides on which page of our site an article will be shown.

Functions

USER LOGIN/SIGN UP

This function interacts with the “users” collection in our database. When users go to the login page at our website, they can choose to login or to sign up. If they want to login, they enter username and password. These strings will be sent to our backend, and our backend will use the user information in database to implements an authorization. If the information is correct, user will enter a profile page; if the authorization fails, an error text will be displayed.

If users want to sign up, they click on sign up button and will be linked to sign up page, where they enter username, email, and password. We provide alarm messages if the email address entered is not in an appropriate form, or the password entered is less than 6 characters. Then username, email and password will be sent to backend. Our backend checks if this email already

exists in out database. If it already exists, error message will show up on page. Otherwise, our back end adds a new user documentation into our database. And now this new user can login.

At frontend, there is a state for each user to record if she/he is logged in. When the user login successfully, this state will change. At profile page, there is also a logout button, by clicking which the user can logout and the state shifts again.

FILE READING/UPLOADING

This function interacts with the "updates" collection in our database. On our website, there are four pages containing lists of articles, namely update, research, article, and announcement. Each article documentation in our database has a property "type" whose value is one of those four. And for those four pages, we load data with the corresponding type from the database, and display the article lists on the pages. When user click on a title on the article list, a new page with the content of this article loaded from database will open.

When user login, at profile page there is a form to post new articles. The form has 4 text fields, corresponding to article type, title, author, and content. User fills these four files and click submit button, then these data will be passed to backend, packaged by our Mongoose, and finally tored at database in the appropriate form.

How to install the website

BEFORE INTEGRATE FRONTEND AND BACKEND

Back end

1. Open the backend folder in our project folder
2. –-npm install (to install dependencies)
3. –-node server.js

Front end

1. Open the frontend folder in our project folder
2. –-npm install (to install dependencies)
3. –-npm start (please make sure you run back end first before this)
4. Open it on localhost: xxxx, where xxxx is port in use.

Note: now front end and back end are running at two different ports. Sometimes in order to make change to the front end files you have to do so, because after integrating front end and back end you are not able to access some of the front end files.

RUN INTEGRATED VERSION

In this version the frontend and backend can run at one port.

To run it, in integrated version folder, –-node server.js. And it will run at localhost: xxxx, where xxxx is port you assigned for backend.

How to update

ADD NEW ARTICLES

There are 2 ways to do this.

A. Using our posting form.

1. Open the site and login

2. Go to profile page, and there is a posting form where you can enter title, author name, type of article, and the texts. Then press submit. The new article will be uploaded to database and then shown on the page of its type.

B. Directly interact with our database.

Edit the database using usual methods. Here I recommend two tools.

1. Mongo Compass, MongoDB's official visualization app. You can connect to our database with it and make any change to the documents easily.

2. Postman. This is another convenient tool to go into our database. You can install its package and then open its online app page. Just enter URL of our database then you can do any change like posting or deleting.

DELETE ARTICLES

There is no function for deleting articles on our website. Therefore, the only way to delete materials is using database tools. Please look at *Add new articles- Directly interact with our database.*

UPDATE INFORMATION ON PEOPLE PAGE AND EMPLOY PAGE

For now we do not have database collections for these two pages. To update these pages, please code by hand. Please go to the frontend-src-component folder in our project, find

“people.component.js” and “employ.component.js”, and then you can make changes on those js files. Building database connections for these two pages may be the next step for this website developing project.

CHANGE PICTURES ON THE PAGES

All of our pictures are stored at frontend-public folder. You can rename, delete or add pictures in that folder.