205CDE

Developing the Modern Web

Name: James Lee

Student ID: 6950219

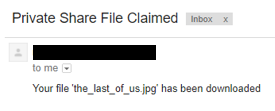
GitHub: https://github.coventry.ac.uk/205CDE-1718JANMAY/205CDE-Course-Work.git

2018

Introduction

During this module we were asked to create a website using modern web development techniques. These include NodeJS (NodeJS 2018), HTML and CSS. I decided to create an alternative private file sharing website. This website would be different from its competitors by including additional features with a focus on the user’s privacy; this was accomplished by enforcing the use of passwords. Files are also removed from the server once they have been up for an hour or after the first time they are downloaded. My website has a clean and modern interface which is simple yet powerful.

Feature list

* Time limit – Files will be removed from the server after an hour. This gives the user peace of mind.
* Password protected files – When the user uploads a file to the server it will be encrypted using the user’s password (The only detail that is preserved is the filename).
* Email receipt – After the user has shared their file using the service and the recipient has downloaded the file an optional email receipt will be formatted and sent to the user which uploaded the file. This is a feature which would be useful when sending files in a business environment. (Added as a suggestion by my Dad who works for Infineon (Infineon 2018)).
* Garbage collection – This is an important feature when handling files; if the user chooses not to download a file then it will remain on the server indefinitely. The garbage collection solves this issue.
* qrcode – Once the user has uploaded a file they will be sent a share page which contains a QRCode. This QRCode itself contains the URL to the download page.
* Secure handling of passwords – The web application uses the NodeJS module bcrypt (bcrypt 2018) to hash and compare passwords. ONLY the hashed versions of passwords are stored in the database.
* Persistent Data – Since all the relevant data is stored in the SQLite3 database the Express routes to files can be restored after a server restart. This means that all the data is not lost if the server fails and needs to be restarted. The files will still be removed properly after an hour by the Garbage Collector module.
* Minimalistic design – The website has a minimal but effective design which is easy to navigate and aesthetically pleasing.

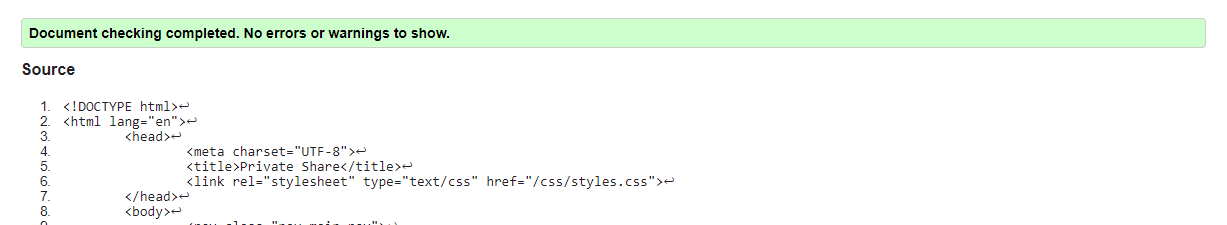
Discuss each of the technologies used in creating the website

NodeJS (NodeJS 2018) – NodeJS is the JavaScript runtime which is built on Google Chrome’s V8 engine. In my opinion this is a better alternative to using web development languages such as PHP because it can be more secure and uses the language of the web (JavaScript).

Express – Express is the minimal web framework which I have used to create my website. It has allowed me to easily create and remove dynamic routes to both efficiently and privately share files. I think that Express is a great alternative to other frameworks such as Flask (Flask 2018) and Django (Django 2018) because it has a solid base in JavaScript and is completely asynchronous.

SQLite3 (sqlite 2018) – SQLite is the database that I have used to make data persistent on my website. The website uses a three-stage normalised database which is connected using foreign keys. I chose to use SQLite (Relational) over and alternative such as Mongo (Document) or Neo4J (Graph) is because I am storing well-structured data.

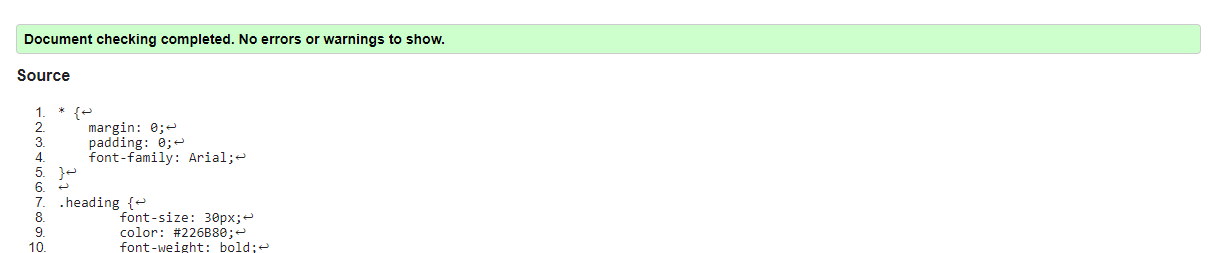
HTML – Hyper Text Mark-up Language is one of languages of the modern web. It is the structure of websites on the internet. All the HTML that I wrote passes the W3 validation.



I have also implemented the use of scheme.org as can be seen in the Google data scraper. This means that If I was to have my website indexed by Google there would be some data displayed about the website in the search engine.



I was better of using HTML for the websites structure because it is more versatile. This means that I can use many different technologies together instead of being tied to PHP for example.

CSS – Cascading Style Sheet is the way that we add style to HTML. This is the tool that I have used to give my website a ‘modern’ look. All the CSS that I used to style my website passes the W3 validator

As you can see in the screenshots below I am using some advanced CSS selectors in for my website which allow fine-grained control over the styling of my website.



I am also using a mixture of external and inline CSS which again allows for finer control over the style of my website.

Discuss any issues you came across when developing the website

While I was developing the website, I needed to keep concurrency in mind. This was because I would need to anticipate having more than one person using the website at a time. I would need to be able to keep track of the user’s files and make sure that when they download files they are getting the correct files which they are able to decrypt. I overcame these issues with the use of a node module called uuid (uuid 2018). This module allowed me to create a unique identifier for each separate transaction. This neatly and quickly solved the issue of concurrency by allowing me to make sure that users actions while using the website didn’t clash.

Storing passwords is another issue which I needed to overcome. In the end I used a NodeJS module called bcrypt (bcrypt 2018) which allowed me to hash passwords. I could then securely compare passwords to determine if the user should have access to the file download. Plain text passwords are NEVER stored in the database.

Another issue which I came across while making my website was the dynamic download page creation and dynamic Express (Express 2018) route creation and removal. Since my web service is allowing users to share their files I needed to dynamically create URL’s which the user can then send to the recipient which allows them access to the file. I used several different Node modules and self-written modules in conjunction to solve this issue. These modules were uuid (uuid 2018), node-schedule (node-schedule 2018), express-es6-template-engine (express-es6-template-engine 2018) and route-remover.

* uuid – The uuid module allowed me to create a unique id which used for the actual URL. This makes sure that I have never have any conflicting URL’s.
* express-es6-template-engine – The template engine module allowed me to inject the download URL into a share page which means that the user can share the download URL easily.
* node-schedule – This was the module which allowed me to schedule the garbage collection method.
* route-remover – This is a module which I wrote which allows the complete removal of express routes by their available path.

What have I learnt over the term?

During this module I have been able to learn about the best practices when it comes to creating modern websites. These are websites that offer dynamic content to the user. Using this technique means that the developer can reuse huge amounts of HTML and CSS by using template engines and JavaScript. An example of this dynamic page creation would be the ‘download’ and ‘share’ page for my website. Each of which changes depending on the download URL and ID of the shared file.

It has been a great insight learning about how asynchronous JavaScript works and how to write it. The simplicity allows the creation of powerful websites which don’t bottleneck and force users to wait.

Over this semester I have also learned how powerful NodeJS’s package system is. When creating the qrcode for my share page I found a Node module called qrcode (qrcode 2018) which allowed me to easily create a qrcode. The qrcode was already base64 encoded. This allowed me to simply use es6-renderer to place it on my share page.

How would I improve on my website?

* Time limit – The user could decide how long they want files to remain on the host server at the time of upload. This would be a helpful alternative to the hour time limit included in my initial implementation.
* Download limit – Instead of purging the files from the server the user could determine how many times they would like to allow the file to be accessed. The feature would be extremely useful when sharing the same file with multiple users.
* If I had more time to work on the website I would like to give it full responsive web functionality. This would allow users on mobile devices to use the file sharing service. I have not added this currently as it would require a completely new design for mobile and tablet.
* Improve the information that is sent in the optional email. For instance, you could include the browser type and IP of the client that downloaded the file.

Extending the website with the features listed above would be relatively simple. These features were not included in the initial implementation of the website because they would have taken slightly more time to implement and fine tune.

Conclusion

During this module I have been able to refine my skills which JavaScript, HTML and CSS. These are all skills which I will need in an industry environment. I have been able to make a website which is effective and does everything that I wanted it to do; my website had a complex and interesting backend while maintaining a simple and effective user interface.

References

NodeJS is a JavaScript runtime which is based on Google Chrome’s V8 JavaScript Engine. [Online] Available at ‘https://nodejs.org/en/’.

uuid [Online] Available at ‘https://www.npmjs.com/package/uuid’

node-schedule [Online] Available at ‘https://github.com/node-schedule/node-schedule’

express-es6-template-engine [Online] Available at ‘https://www.npmjs.com/package/express-es6-template-engine’

bcrypt [Online] Available at ‘https://www.npmjs.com/package/bcrypt’

qrcode [Online] Available at ‘https://www.npmjs.com/package/qrcode’

sqlite3 [Online] Available at ‘https://www.sqlite.org/index.html’

Infineon [Online] Available at ‘https://www.infineon.com/’

Flask [Online] Available at ‘http://flask.pocoo.org/’

Django [Online] Available at ‘https://www.djangoproject.com/’