GuardianLink Document

GuardianLink Document

Summary

- This is a mock website that allows volunteers and organizations to sign up in order to communicate and for organizations to receive help.
- Some things to keep in mind, this is not meant for real world environment, it does use security precautions especially dealing with SQL injections, but some features are left out to keep it simple. Such as using HTTP instead of HTTPS only, or using memory to store session data, instead writing data to a disk.
- It will allow users to register and securely sign in. Once they are signed in they would be able to edit their profile info and find other user types. They can send messages to other user types.
- Admin user will be able to edit users, create basic users, and delete account.
- User will also be able to delete their account, sign out and request password reset. To be honest, password reset is the least fleshed out feature, but I decided it would be better to finish the project instead of making major changes to it.

Planning Docs

1. Create weblayout, pages, skeleton of website
2. Decorate the pages with CSS
3. Create DB for web to store user data and communication messages.
4. Implement a secure login
5. Create registration forms
6. Add contact us page
7. Profile page to display user info and link to view other user types.
8. Admin page with tools for admin to use.
9. Add forgot password
10. Fix features I am not happy with.
11. Review files, clean up code.

Technology Stack

- This took just over a month. At first, I started with a WAMP stack, I was pretty far along in the project, but I kept dreading how PHP integrates with HTML so I made the decision to scrap the project and start over with JS/NodeJS. I am glad I did it, JS really grew on me, I learnt a lot, but it was a long and gruesome undertaking.

- Reason I chose JS, its mainly because of DOM and it just works so well with HTML. Another reason is NodeJS, its a very powerful environment and well it uses JS so another good reason to wanted to use JS. One of the biggest reasons I did switch to JS, I wanted notifications/feedback messages to show up instantly on the page, instead of opening another page to display messages. With PHP, I was really struggling with this feature.
- CSS is another wonderful tool to use with HTML, I did use SASS when I was working with WAMP, I did like it, but CSS on its own works just fine so I stuck with that.
- HTML, well had to use HTML for basic site layout. Another big reason I moved away from PHP, I found I had no choice but to write some web pages as PHP pages, and it was very messy to maintain. PHP and HTML mixed in a single document was not fun to work with or look at.

Testing Methods

- I mainly used White Box testing, traversing the website see if I could access APIs that were restricted to me. I tested sign in feature of course, and make sure Sign Out works as intended. After sign out, I am unable to traverse back and view restricted content without signing back in again. I also tried sharing a link with another browser and the session did not carry over to another device.

Challenges Faced & Lessons Learned

- I did not use a Wireframe diagram, and well first project it is a good time to learn. I do regret this choice, because as code grew, files multiplied, it started to get harder to keep track of things. It wasn't impossible, but could have made my life easier. I did make a plan before I started, which outlined steps to take and which order to take them. This did help initially, but once the website design was finished, I found I did not use that any longer.
- Honestly, biggest challenge was learning JS/NodeJS. The amount of JS taught in this course was very minimal. I used YouTube full course on JS and NodeJS to teach myself the skills required to finish the project.
- Learning HTML layouts, at first I was not using FlexBox, and I was really struggling with the web layout. I read more about it on W3S and that really made things easier.
- Just learning NodeJS modules was very time consuming, but worth the effort. Such as learning how to integrate Mutler, bcryptjs, mysql, and many more.
- At first I was using form tags, then I found new ways of doing it which made going forward way easier, but initial learning curve was massive. I moved on to XMLHttpRequest and after that I learned about fetch() API, which is the best one to use. I learned about it too late so most of my requests are sent with xhr.
- Comments, I could definitely add more comments. I did notice at times I forgot what something did after coming back to it later, which could have saved some time if my comments were better.
- I realized you learn the most while doing projects. Reading books is necessary and worthwhile, but without actually doing a project, putting what you read to use, you do not really learn the content.

Future Recommendations

- Better hands on content added in the course to learn JS. Some content in the course can be reduced greatly, I felt like I wasted a big chunk of time watching some videos or reading books which were really too long or just not useful at all. I can think of few off the top of my head. The first IT Essentials, too bloody long. if I did this course again, honestly I would just skip that. Another one I can think of, videos on DevSecOps. Some of that content in my opinion was absolute trash and extremely long too.
- Maybe add variety to end of course project for different interests, full stack, front end or back-end, such as being able to pick between lets say creating this website, making an app, or an automated scheduler for work, etc. Reason for it, personally I have no interest in web design. Biggest takeaway for me was backend stuff, JS/NodeJS, but again it was great to learn HTML more in depth.