Website Report

Front End:

The three front end tech stacks that were considered for the website were: Bootstrap 4 with jQuery, Html 5 with Angular, and Html 5 with ReactJS.

Some pros regarding the Bootstrap 4 with jQuery stack is the simplicity and ease of development using it. Bootstrap 4 by default comes with several JavaScript plugins using jQuery. It makes designing and coding html much simpler and easier opposed to using html by itself; notably that in our website, we did not need to create a CSS file to clean up the user interface, which bootstrap handles itself. It is a popular front-end framework, it is well documented and has lots of community support along with being fast, lightweight, and responsive. Additionally, many templates and guides are available for reference and use. The jQuery JavaScript library is heavily tied into Bootstrap 4, making it the main JavaScript language to pair with Bootstrap 4, along with being a popular library with an easier learning curve compared to others. Two cons that come with this stack are that it might not be fully customizable, as being easy to use sacrifices a lot of freedom with customizability. Additionally, Bootstrap 4 is tied to jQuery, restricting other JavaScript libraries and leaving a lot of plugins and features unused. Alongside this, the stack also isn’t the most efficient at handling larger tasks.

Regarding Html 5 with angular, the pros and cons contrast with the bootstrap and jQuery stack. Some pros in this stack is that with Html 5, websites can be made to be fully customizable by creating unique Html and CSS files specific to the website’s needs, in contrast to a preset library in Bootstrap’s case. Html and CSS are also very popular and thus well documented, with many questions having answers. Pros regarding Angular are that its elements and modules feature allows for great customizability, being efficient in loading times for large websites, and being easily maintainable for future updates. The cons again contrast with the first tech stack, as the learning curve is a lot steeper and requires more time to learn and use efficiently. With angular, it is also a difficult framework to learn and very complex, making it more suitable to larger and more complex projects than simpler ones.

The final stack we considered was using html 5 along with ReactJS. The pros and cons for html 5 from the previous stack apply here as well. Some pros regarding ReactJS are that the community is large and it is a popular framework. Alongside popularity it is also very efficient by updating the document model only when a change is triggered, while also keeping a copy to compare small changes making it scalable. ReactJS is also modular, where previously created components are reusable and can be brought to other projects with no changes. Some cons include that ReactJS has a steep learning curve and is also is poorly documented, which magnifies the greater learning curve.

Back End:

The three backend frameworks we considered were: Ruby on Rails, CakePHP, and Flask.

The pros of Ruby on Rails are its large community and extensive library of plugins available. A lot of the large websites like Shopify or GitHub are built on RoR, so using it promotes the best standards and web development practises. RoR is not new so almost every problem that you can encounter has been documented and answered by the large community. Another benefit of a large community is the extensive library of plugins created by the community. The cons of using Ruby on Rails are learning a new language, and lack of flexibility. Neither of us are familiar with the Ruby programming language, so the time taken to learn a new programming language would have taken away from development time. RoR comes with a lot of hard dependencies and modules that we would have to configure at the start. The defaults might have suited us, but in the chance that they did not, we would have had to spend time overhauling our Rails application.

CakePHP comes with the pros of allowing developers to build quickly and includes considerable built in functionality. The code generation and scaffolding feature enables for rapid development. CakePHP comes with a lot of built in functionality like validation and authentication which speeds up development time by avoiding "reinventing the wheel." The prominent cons of CakePHP are learning a new language, and lack of documentation. We are not familiar with PHP, so choosing CakePHP as our backend framework would have required learning a new programming language. CakePHP is considered to be a more complex PHP framework and many users feel that the documentation of CakePHP is lacking which would have made development arduous.

The final framework we considered was Flask, a python-based backend framework. The biggest pros of Flask were its ease of development and extensive documentation. Flask is a flexible microframework that allows for the ability to build prototypes quickly. Flask is known to facilitate experimentation which allows for faster development time. Flask also provides extensive documentation with great examples. Our prior knowledge of Python also made Flask more attractive. The cons of Flask are its lack of standardization and fewer tools. As Flask is not very opinionated, a lot of things can be done in lots of different ways. The lack of standardization can build bad habits in developers as non-guideline code is not punished. With Flask being a microframework, it does not come with a lot of tools compared to monolithic frameworks like Django. The lack of tools leads to either searching for plugins/libraries or building things yourself, adding to development time.

CI/CD:

The three CI/CD options we considered were Github Actions, Circle CI, and Jenkins.

Some pros regarding Github Actions is that it is documented well, easy to setup, is free, and it is already integrated inside of the Github repository.

Pros regarding Circle CI are that multiple platforms are supported for CI/CD, very effective and efficient in carrying out tests and tasks. The cons include not having full access unless a user pays a fee and also being a little challenging to setup with poor documentation.

Some pros regarding Jenkins are that it is free, has many plugins and lots of community support. Some cons are that it is documented a little poorly and challenging to setup with an unintuitive user interface.

Conclusion:

Ultimately, we decided to go with Bootstrap 4 and jQuery for the front end, Python Flask for the backend, Github Actions for the CI/CD and sqlite for the database.

The reasoning for our actions is that we wanted to use a tech that was simple and easy to learn, yet still functional for our shopping list calculator. Given the time constraints and the nature of the project, we did not require a very customizable nor efficient and scalable framework. For the frontend, using bootstrap helped save time as we could forget about styling by creating a custom CSS file and let the library do most of the hard work. jQuery was also very easy to learn allowing us to focus more on functionality rather than worrying about aesthetics and spending too much time learning. And for CI/CD, Github actions seemed like the simplest and most efficient solution as it is built into the repository and free.