# Assignment 2 Report

## Options We Considered

### Summarized Comparison

We chose to build a web app for this project over a mobile app. For the frontend we considered the options React, Angular, and just pure HTML/CSS, in which we chose to go with React due to its popularity and support, since it would be a better asset to learn as well as have many libraries to help with it. For backend, between Java, Python, and Javascript, we chose to go with Java, utilizing the Spring framework to build an API that can act as an interface to the backend. There are many resources out there to help with this, and we were already very familiar with Java, making it a lot easier to work with. We can also use the JUnit testing framework along with some help from the Spring framework to create a test infrastructure. With this, we chose to go with PostgreSQL as the DB option over MongoDB and Firebase, since it has built in support with Java and can be used with an ORM to make things easier. It is also extremely popular, being the #1 used technology for databases according to the Stackoverflow survey results.

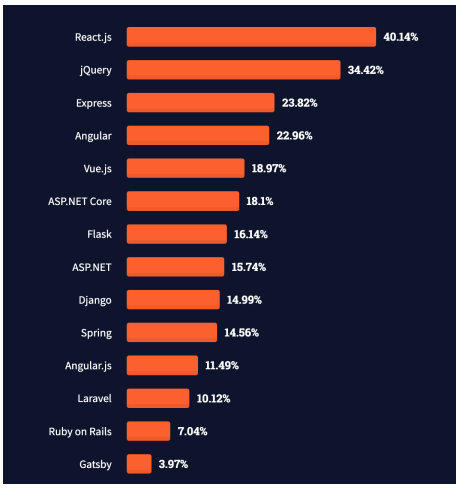
### Web app vs Mobile app

To build out this shopping list, we considered making either a web application or a mobile app. A mobile app is nice because it serves the purpose of making a quick calculation or storing some information on the go, but we ended up choosing a web app. The web app gives us more freedom in terms of space on the screen. We are able to display more things on a single screen and not have to rely on navigating to different pages. Web apps are also more widely used by the industry meaning we will have more resources to use in case we need help. It will also help develop our web development skills which is a valuable asset as a software engineer.

### Frontend Options

For our frontend, we considered 3 options. React, Angular, and just plain HTML/CSS. In the end, we ended up choosing React. Here’s why.
HTML/CSS is probably the most basic way to go about frontend. You don’t need to use any libraries or frameworks, you just do everything from scratch. Although this is basic, it does make your job a bit harder. Using libraries and frameworks does some work for you and makes it easier. React and Angular would be easier, but take longer to learn. We already knew HTML/CSS and React, so those were much easier to pick up, whereas Angular would take a while to learn.
In terms of maturity, HTML/CSS has some libraries such as bootstrap or tailwind to help with CSS. Angular also has some useful libraries like Angular Material. React definitely is the most mature in this regard though, and has libraries like Material UI for components, but also

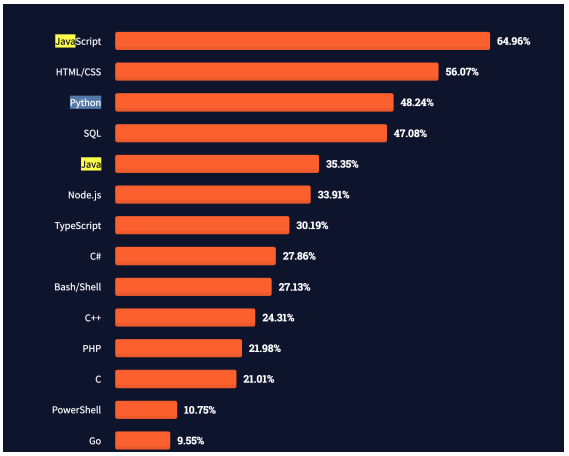
lots of other libraries that can be installed. It also has the testing framework Jest to help test the frontend.
Angular and React are frameworks/libraries built on Javascript. This would be a good combo if we do the backend in Javascript as well, with something like Node.js or Next.js. It would ease development in terms of the language, while HTML/CSS are separate frontend technologies that don't require javascript (but could use it as well).
React is easily the most popular, as seen by the Stackoverflow survey as well below:




Angular is close as well but still quite far from the top. This makes React a very valuable asset to learn as well.
In terms of performance, Angular has better performance for large enterprise-level apps, whereas React works better for smaller apps. Since this is a pretty small app, React wins in this category.
Overall, React is more popular and better suited for smaller apps, which is why we chose it.

## Backend Options

For our backend. The main 3 options we considered were Java, Python, and Javascript/Typescript. Each of these languages are very popular and highly used, meaning there is lots of support for them and plenty of resources online if we need help with any concepts. Each is also very common in industry, guaranteeing us a valuable asset on our resume with a project to show for it. If you look at the Stackoverflow 2021 Survey, Javascript, Python, and Java are listed as first, third, and fifth highest popularity technologies respectively, putting them all very high on the list.



In terms of familiarity with the languages. We both used Java in CSC207, which makes us pretty familiar with it. The benefit of this is that we don't need to focus on learning the language, and can put our sole focus into making a good design of the backend instead. On the other hand, we are slightly less familiar with Python but we still used it back in CSC108 and CSC148, so we know a good amount about creating a backend. Javascript is also a technology we are familiar with since it is very common in making frontend applications, but Typescript would be something we would likely want to use, and we aren't too familiar with that.

All 3 languages are very mature since they have been out for a long time. For Java, the most common backend framework is Spring. Spring was released in 2003 and has many sub-frameworks under it as well such as Spring Web, Spring Security, Spring MVC, Spring Boot, and many more. These come with lots of documentation and lots of tutorials to teach us how to use them. Python's two most common frameworks for backend web development are Django and Flask. I've used Flask before, and it is pretty straightforward, but Django is known to be more feature-heavy with more functionality built in. Both frameworks are very mature and have lots of support with them. For Javascript, there are also many frameworks out there for web development. A common one that I looked into is next.js, which is a framework that's specifically built to be used on top of React as the frontend. This is a very good option for us since we are already using React as the frontend technology, and by using next.js it would be very familiar with the frontend side of things making development a lot easier. It also has built in support for Typescript. Javascript/Typescript wins in this category.

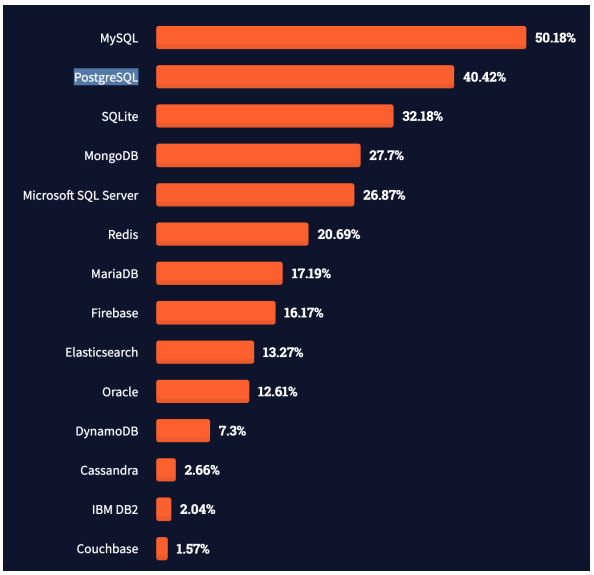
As touched on, Javascript/Typescript's domains are both frontend and backend, meaning we can share the same language throughout the entire application. This makes it very easy for the frontend developer and backend developer to interact and assist each other when needed. With next.js, there is even further integration, since next.js works nicely with the frontend framework React. For Python, the advantage comes in its capabilities and libraries for Data Science and ML. If we need to do any computations with data, such as analyzing the data on items in our shopping cart, or figure out trends based on pricing, Python is the best option here. Upon consideration of this though, we realized that we don't want to go too in depth into the data science part here, so python doesn't have too much else to offer in this space. Java is primarily a backend framework, so that's where it's domain has always been. There have been some frameworks to allow it to work in the frontend as well such as JSP, but those didn't go too well. Java has always excelled in the backend, especially with its framework Spring, which adds a lot of functionality to it.

In terms of scale, Java scales very well and especially while following OOP principles such as SOLID and Clean Architecture, it can be extremely scalable. This is why it is the most commonly used technology for enterprise level backends. It also has good performance and speed. Python isn't too good in the speed category, and in my opinion doesn't scale as good as Java, and neither does Javascript/Typescript.

In the end, considering all of these points, we chose to go with **Java** as the backend technology on top of its Spring framework. The main point that sold us is the fact that we're most familiar with it, so it will be easier to focus on the product itself and not have to worry too much about learning the language features.

# Database Options

For the database options, we were deciding between SQL and NoSQL databases. Specifically, the 3 DBMS' PostgreSQL, MongoDB, and Firebase. We ended up choosing PostgreSQL, but here are some of the considerations we made.
In terms of ease of development, Firebase is clearly the easiest to use. It's a NoSQL database management system and provides a library in many languages that is very familiar and easy to use. MongoDB is also easy, since again NoSQL databases just require you to store your Java objects, which makes it pretty simple. PostgreSQL would be the hardest if used plainly, but with an ORM, it should also be pretty straightforward.
In terms of maturity, PostgreSQL has been around the longest. It has very extensive documentation found on postgresql.org. Firebase and MongoDB are newer, but they are also more modern and the documentation looks a lot nicer. All 3 have good support and will probably last a while in the industry. PostgreSQL has an ORM that makes it easier to use, which is provided with the use of the `javax persistence` library. This seems like the most common way to access PostgreSQL in a Java application, so it's likely how we will do it.
The domains for them are that Firebase and MongoDB are NoSQL, but PostgreSQL is a SQL database.
Each of the 3 options are very popular database options these days as you can see from this chart in the Stackoverflow 2021 survey:




PostgreSQL as highlighted is the second highest in demand database option. This was a major reason why we chose it. It is a valuable technology to learn in today's industry, whereas things like Firebase and MongoDB are also up there, but not as high.
In terms of performance, PostgreSQL would likely be the slowest because of the use of an ORM rather than raw SQL queries. MongoDB and Firebase would be much faster. This wasn't really a big issue for our decision, because we don't need to worry about speed too much with an application this small.