Ultimately we decided to use React as our front-end solution and Flask as our backend.

One major reason for choosing React was that we both had a bit of experience working with it, thus making development easier. Another reason was its immense popularity and support with many different packages, which make development easier and more streamlined. While we did consider other options, none were quite as good of a fit. Angular, while also very popular, was much more comprehensive than we needed and, more importantly, neither of us had any experience with it, which is an issue considering the steep learning curve. We also considered using Vue but decided against it due to its lack of support in comparison to React, as well as the fact that Vue's selling point, flexibility, would ultimately not be needed in this simple project. We decided against using Svelte for a similar reason, Svelte's main selling point is its performance, which is not a huge factor in this simple application. Moreover, Svelte has much less popular support than React. Finally we briefly looked at Ember, but decided against it because, like Angular, it was a comprehensive framework with little flexibility and many concepts with which we were unfamiliar with, not to mention it's unpopularity. So, we decided to use React due to our familiarity with the technology, its extensive online support and many community packages.

Following similar reasoning on the backend, with our shared experience in python easing development and the availability of libraries, the backend would be simple to build and test. While we considered more comprehensive and robust frameworks, for the lightweight function of the backend that we had envisioned, Flask ended up being a perfect fit. With the bulkiness of certain frameworks inhibiting rapid development and unfamiliarity hindering the lighter alternatives, while they would have offered many respective benefits, for the given scope and required functions Flask allows us to produce our product quickly and with much less hassle.

We decided to work on a web application as we both have a bit of experience with web apps, and neither of us had any experience working on a mobile app. Moreover, a mobile app makes less sense for this project as it is a simple calculator, not something which is worth downloading a whole mobile app over, but probably worth visiting a web page for. Even if an end-user is on a mobile device, React (our frontend "framework" of choice) can be used to make clean and responsive mobile web pages which work on mostly any somewhat-modern device.

For the frontend we considered a few different options but ultimately chose to work with React. Other frameworks we seriously considered were Angular, Vue and Svelte, and we briefly looked at Ember as well.

We ultimately decided on Facebook's React "framework" (while technically not a framework it almost functions like one and is popularly referred to as one, so we will also refer to it as such) for a myriad of reasons. First and perhaps foremost is that we both had some experience working with React, but no experience with other frontend frameworks. If we had chosen a different framework we would have spent a lot of time fiddling around with features and implementations we aren't familiar with, which would have greatly reduced the speed and ease of development. Additionally, React is very easy to set up and for a project with a small scope this is a large boon. React is arguably the most popular front-end framework currently. This means that we would have no problem diagnosing issues or finding libraries to use alongside it. Since React is a well-established frontend solution, there is a plethora of available libraries which make working with React easier, which would come in especially useful should we ever decide to scale up the scope. And while React is a frontend framework it comes with tools to easily interact with virtually any backend giving us flexibility in that department as well.

Another option we seriously considered was Angular. Developed by Google, Angular is second only to React in terms of popularity. But, one of the biggest reasons we decided against using Angular was due to the steep learning curve. While we both had experience with React, React itself is not a particularly difficult framework to use (again this is because it's not really a framework). Angular however is a full-fledged framework and has many moving parts and concepts which we were entirely unfamiliar with, not to mention that we are unfamiliar with Typescript itself. Having to learn everything just for a project of this small scope would have made development a bigger head-ache than necessary. However, one of the tempting things is that Angular doesn't require any additional frameworks for advanced features, so if this were a project which started with a large scope there would be an argument for jumping straight into Angular to save on issues down the line.

Vue was yet another framework which we considered, unlike React and Angular Vue was developed by an ex-Google employee and has no large company backing. Vue is arguably the easiest to learn out of all of our options, and while we had no direct experience with it this likely would not have been a huge issue in such a simple project. However, Vue is not quite as popular as React, and most of its unique functionality and usefulness (for example its flexibility) would ultimately not be needed in our project. Thus we saw little reason to choose it over the more established React, which has more packages and more community support than Vue.

While we did seriously consider using Svelte we ultimately decided against it. Svelte was created by Rich Harris and also has no major company backing. Svlete's major selling point is its performance and simplicity, as unlike React or Vue, Svelte doesn't have a virtual DOM. Instead Svelte compiles code into high-optimized vanilla JavaScript (i.e. there is no framework code added at runtime). While this extra performance would certainly be useful in the right circumstance, it isn't something which is an issue for such a small application. However, Svelte's

light-weight solution may be the exact reason to use it for a small scope-project. But ultimately, Svelte isn't quite as easy to set up as React and more importantly Svelte has by far the least popular support in comparison to our first three options. This would mean extra trouble diagnosing problems and little support in the way of packages, which is a big reason why we chose React. So between the small community and our predisposition to React due to our familiarity we decided against using Svelte.

Finally, we briefly looked at using Ember.js. While (essentially) the oldest framework of all our options, we quickly decided against Ember.js for a few reasons. One was its general unpopularity in modern development stacks, it has arguably the least support out of any of our other framework options. Additionally, similar to Angular, Ember is an inflexible, comprehensive framework with which we are entirely unfamiliar. And like Angular it didn't make sense for us to spend time learning it for such a small project. While it is the oldest framework it doesn't have as much support as React in terms of available packages. For these reasons we quickly dismissed it as an option.

Moving on to the backend, we again decided that we would prefer to work in a language that we were comfortable in, but also something that was a popular web development language, so that our options would be comfortable but not limiting. Thus, we opened our investigation by considering solutions that used C#, Java, and Python.
Thus, we evaluated several popular frameworks:

- .NET (C#)
- Spring (Java)
- Struts (Java)
- Flask (Python)
- Django (Python)

Starting with assessing the languages, we wanted to work in Python since it's easy to write and has lots of great modules. While we love the structure and strong typing of Java and C# as a means for easy documentation and collaboration, none of these features seemed as appealing given the very limited scope of the project and backend, all of which suggested that the amount of code would be small and the benefits of good structure and performance optimization would be limited. Java and C# would also be faster at runtime, since they're compiled languages, but we did not imagine a scenario would arise where that would become an issue.

The second criterion was the amount of development time that we would require with each framework to build a backend. .NET was a tempting option, since one of us had done a work term working in it and was familiar with building ASP.NET core apps. We had also had past experience building webapps with Django. We decided that we would also be capable of working with Flask fairly easily as well since it is designed with expedited learning and development in

mind. Out of the three languages, Java would be our least familiar and we have not worked at all in Spring or Struts. However, out of the two, Struts is easy to learn and develop in, so we favored the latter of the two in terms of development time.

We wanted to pick something lightweight, since we anticipated that the backend would not require many features and wanted it to be quick to install. We therefore were leaning towards Flask and Spring, since they are very light frameworks, with the former being a "micro-framework" and the second with most of its modules as optional. We know from experience that .Net and Django are rather heavy. Struts was not advertised anywhere as lightweight, and we took this to mean that it is not.
Given all the criteria, we decided that it would be best to select Flask, since this would require the least amount of development time to learn and build, while remaining lightweight and practical.