# Web Application

An important aspect to LogiSteps is the ability to portray user data generated from their shoes in a clean, easy to use manner. While there may be several approaches for displaying the data (mobile application, native application, etc.), one of the most dynamic and flexible options is using an HTML based web user interface - doing so presents numerous advantages over other display options. While the advantages are numerous and vary from use case to user case, a few important advantages to consider are the following:

* **Cross Platform** – by serving user interfaces from a web server, it allows the application to be designed without worrying about compatibility across Windows, MacOS, and Linux. UX code will render and run in the browser.
* **Installation-less** – Since the user interface is being served from a web server, there is no need to design software for installing and booting the user interface. End users only need to know the right URL for accessing the application.
* **Updates** – Updates can be easily deployed to a centralized location without the need to push updates software to numerous distributed users.
* **Administrative Overhead** – by deploying the end user application to a web server, our project will be able to mitigate much of the administrative overhead placed by companies such as Apple.
* **Fast Prototyping/Development** – The abundant availability of front end frameworks help get the project into deployable state without needing to design UI components from the ground up.

Of course, there are more advantages than the ones considered, however they help illustrate the case for using a web application for presenting data to users.

While there are numerous advantages to this approach, one must also consider valid disadvantages, and factor them into the decision-making process. The use of a web server for processing and displaying user data lacks behind mobile and native apps when considering the following issues:

* **Processing Overhead** – Using front end and back end frameworks for building an interactive application for a user comes with a lot of extra files. Web applications often require an extensive number of files for rendering and processing data, much of which may not be used if the user does not navigate to that respective part of the application. Additionally, since the application is abstracted from the system, it often runs slower than native desktop and mobile apps. The speed of the application also becomes dependent on the web browser being used, an aspect of design that the developer cannot control.
* **Programming Language** – Since the front-end portion of the application is being run in a browser, development is limited to javascript, HTML, and CSS. Additionally, the back-end server will also be limited to a language supported by web frameworks and web servers – typically NodeJS, Python, Ruby, etc.
* **Resources** – The application will not have access to many of the client’s system resources. This will somewhat limit the design flexibility for displaying information to the user.
* **Security** – Resources will be more viable to attack and probe from outside sources. Additionally, data being transmitted over the public internet may be vulnerable to spies. Security becomes a much larger problem when using the public internet to serve and present resources to users.

One last thing to note that by presenting information to users using a web application, the application becomes decoupled, with a distinct separation of client side and server-side software. This can be an advantage if designed correctly but can make it more difficult to design and develop as well. The server and client may not be able to easily access resources that could be used if both aspects of the application were closely coupled (like a native mobile/desktop app).

The web application will be a composition of 4 logically separated components – the front-end framework, the back-end framework, the long-term storage, and the web server. A critical review of available resources are discussed in the following sections of the Web Application technology and concept report.

# Front-End Framework

Choosing a front-end web framework for a web application can be a difficult task due to the vast multitude of available frameworks – each claiming to be better than the next. Each framework excels aspects that other frameworks may lack, and lack in others. After doing initial technology research, team Omicron discovered that three of the most popular front-end web frameworks available to developers are Angular, ReactJS, and VueJS. Other notable front end frameworks worth mentioning are Ember, Elm, knockout, and more.

When considering front-end frameworks, several attributes were explored, some of which included available documentation, architecture, data binding, load times, syntax, DOM manipulation, element expression, supporting community, learning curve, and more. Each of these attributes were closely examines, and used to compare each framework, helping drive a choice. Each of the comparable attributes are examined in finer detail below.

## Documentation

Angular, React, and Vue are three of the most popular front-end frameworks in the world. As a result, each has an extensive amount of rich, detailed documentation and tutorials. All three frameworks have the necessary documentation to begin development and reference if there is any trouble in development. Of all three, it is difficult to choose which framework has the best documentation, but it appears that React may lag behind both Angular and Vue in documentation, with simple explanations of the software[\*\*\*CITATION HERE\*\*\*]. This claim appears to be backed up by several members of the software development community as well, with many claiming that React lacks in some aspects of its documentation. Having little experience in web development, documentation is an essential attribute for Team Omicron to consider when choosing a proper front-end framework.

## Architecture

Architecture of a framework has a large impact on how flexible a developer’s application can be. Through Team Omircron’s research, the group found Angular to be highly opinionated on how a developer’s application should be structured, with React and Vue providing a much more flexible approach, allowing developers to structure their application in a manner that best fits it’s needs. This has much to do with the fact that React and Angular are only View layer libraries, while Angular uses a model, view, view model structure with structured components and services.

Having a strict/strongly opinionated framework can make design work simpler by limiting the developer’s decisions, but it limits the flexibility of the application, while also adding overhead that is often unnecessary. This makes Angular a much heavier framework than Vue and React, introducing a lot of complexity that is likely unneeded. On the other hand, having a structure to develop off can sometimes be beneficial in implementing an application. Due to this, both Vue and React have templates and third-party plugins for modeling client-side data and structuring an application. This makes it possible to integrate model layers into a front-end application without requiring an application to adhere by a strict set of rules (as it is in Angular). The benefits of structure can be exploited without dependency on it.

React – V (view) - <https://medium.com/@cabot_solutions/flux-the-react-js-application-architecture-a-comprehensive-study-fd2585d06483>

Vue – V (view)

Angular – MVVC

## Data Binding

Data binding is a property that maps underlying data to the view layer of the application. This attribute relates to the architecture of the framework being used. In the frameworks that are primarily the view layer of an application, data modeling is typically one way, from the model to the user interface. Vue and React follow this pattern; data has a downward flow, and in React child elements don’t have any effect on parent data. This can be advantageous because it keeps the application logic simple – data originates and the state of the data lives in only one place. The disadvantage of this attribute is that it makes it more difficult to directly change the state of data from user interaction. Other tools and features of the framework must be exploited to update the underlying data.

Angular differs from Vue and React in that it offers two-way data binding. Two-way data binding is a powerful feature that allows actions by the user to directly update and change the state of underlying data models. This becomes possible in Angular because of its model and view model layers underneath the view layer of the framework. Although this feature can be powerful in applications, it can lead to bugs that are difficult to test, find, and fix. It also adds extra logic to the application. For applications where data is primarily viewed, with no great need for user manipulation, this feature of the framework becomes less useful.

Since LogiSteps will be using the user interface to primarily display user data, two-way data binding is likely an unnecessary feature. One-way data binding will most likely provide a cleaner conduit for conveying data to the end user.

## Performance and Load Times

A noticeable metric for an end user of a web application is the performance of the web page and how quick the initial load time is. John Hannon did a study on the performance and load times for various front-end web frameworks, and he found that for keyed implementations of the frameworks, Vue beat out Angular and React significantly in both typical operations as well as initial load times. Angular and React appeared to perform similarly for typical operations, with React slightly edging out Angular in most metrics. When comparing initial load times, React was significantly faster than Angular, but still significantly slower than Vue.

The web application for LogiSteps is not expected to be an enormous web application performing complex algorithms. Due to this, there is an expectation that the web page should be fast and respond to users quickly. The page should also load quickly when a user navigates to the page to view their statistics. An analysis of this metric points to Vue as being the best option when picking from Angular and React.

## Syntax

Syntax is an important aspect of a framework, as it directly correlates to the learning curve associated with the framework; the more unfamiliar a syntax is, the longer it will take to begin development of the application. Based on research done by team Omicron, Angular and React present syntaxes unlike others found in other languages and frameworks. React represents all UI elements using JSX and rendering functions. JSX is an XML like syntax that is used within JavaScript. While JSX can be powerful, allowing JavaScript to be interwoven with UI, it means that using React would require learning a new language for expressing UI elements, presenting a steep learning curve to all members of Team Omicron. Vue, on the other hand, supports JSX, but allows UI elements to be expressed with standard HTML, CSS, and JavaScript.

Angular, like React, provides a rich syntax that is complex in nature. Additionally, Angular makes use of Typescript rather than JavaScript. This means the developers using Angular would need to be learn the Angular specific syntax for element expression, as well as Typescript for data modeling. This would also present a large learning curve to developers on Team Omicron. Vue keeps syntax simple, using HTML, CSS, and JavaScript for UI element expression, while also picking a select few expressions used in AngularJS for more powerful UI expression.

Vue provides a simple syntax for its core functionality, while expanding upon the strengths of other frameworks to allow for more powerful element expression and HTML based templates.

## Community Support

Another important aspect of a front-end framework is the amount of third party and community support that the framework has. This metric was checked by team Omicron by comparing Github statistics. Angular, React, and Vue has the following community activity.

|  |  |  |  |
| --- | --- | --- | --- |
| Metric | Angular | React | Vue |
| Github Stars | 40,903 | 111,824 | 114,636 |
| Github Contributors | 731 | 1,241 | 193 |
| Stack Overflow Tags | 132,505 | 103,665 | 23,029 |

Table - Community Support Statistics for Angular, React, and Vue.

From the quick metics collected in table 1, Angular and React appear to have massive following in the developer community. Angular has a significantly lower Github star count than the others, but this is likely a consequence of Angular migrating versions. Both Angular and React also have a large amount of questions on stack overflow, meaning that many potential questions likely have answers available on stack overflow. Vue has a significantly lower number of Github contributors and stack overflow questions. This means that it may be more difficult to find answers to questions during the development process.

## Other

Angular, React, and Vue all provide several other features as well which have been considered when performing technology research. All three frameworks provide view templating routing, and expressive element expression. When considering framework transfer size, Vue wins out, beating both React and Angular based on a study performed by Jacek Schae.

## Conclusion

All 3 front end frameworks provide similar speed performance, however, Angular typically has a longer initial load time. In terms of learning curve, both Angular and React have steep learning curves, with Angular having complex syntax, and React using JSX for UI development. Vue offers the smallest learning curve by using native javascript, CSS, and HTML to implement rich, expressive user interfaces. Vue also has advantage in terms of size. Due to Angular's heavy use of libraries, complex syntax, and other features, it becomes bloated. Both Vue and React are much smaller in size, with libraries being included as they are required. Angular provides more structure, but this structure can be restrictive at times; React and Vue allow for much greater flexibility. The flexibility of Vue and React can sometimes make it more difficult to start projects, but Vue offers a powerful CLI that helps setup projects with various configuration settings available. Vue is currently less popular than Angular and React - leading to a smaller supporting community and a smaller collection of libraries, but Vue is quickly growing in community size and popularity. In the end, Vue likely offers the best option for our project due to its small size, and smaller learning curve. Team Omicron’s front-end UI is meant to be simple, providing a portal for viewing statistics. The added size of Angular, and complexity of React and Angular most likely make them a less beneficial option for our project.

# Back-End Framework