MCOMD3PFL - Programming Frameworks and Languages Assignment 1

Nathan Shuttleworth

Due 8/11/2021

Table of Contents

[Analysis of Existing Prototype 2](#_Toc87176328)

[Proposed Technical Direction 3](#_Toc87176329)

[Server Implementation 3](#_Toc87176330)

[Language Justification 3](#_Toc87176331)

[Framework Justification 3](#_Toc87176332)

[Client Implementation 5](#_Toc87176333)

[Language Justification 5](#_Toc87176334)

[Framework Justification 5](#_Toc87176335)

[Roadmap 6](#_Toc87176336)

[References 7](#_Toc87176337)

# Analysis of Existing Prototype

The existing prototype framework is not fit for use by anyone, let alone business use. It does not meet even basic standards for security, maintainability, general implementation and completeness. The main root of the issues with this framework is that it has been created by FreeCycle themselves. There are numerous frameworks available for use with varying scale, functions and complexity. Due to this framework being made by the company, it does not have adequate documentation and community support, meaning that the developers are on their own with fixing its many issues.

Some of the main issues with the framework itself are:

* Does not adhere to HTTP standards - only handles 12 of the response codes, misses important ones such as 302 (found) and 502 (bad gateway). The lack of 5xx codes in particular is a problem since these are the ones relating to server errors.
* Prototype is clearly broken - in the http\_server.py file, on line 120 there is a comment describing a ‘hack’ to get the system to work properly.
* No utf-8 data handling, which is the encoding used for characters from languages with different alphabets (e.g. Japanese, Russian). This alienates a large international audience from being able to use this framework.
* Lack of stability - on line 112 of http\_server.py, ‘*If the request does not come through in a single recv/packet then this server will fail and will not composite multiple TCP packets. Sometimes the head and the body are sent in sequential packets. This happens when the system switches task under load.*’
* No security - nowhere that even a simple log-in system could be implemented.
* No modularity - other examples such as Laravel have a whole ecosystem of side programs and features.
* Not WSGI compliant – whilst not essential this is useful, means that this framework is not as well optimised as it could be.
* ‘Frankenstein’ code – Particularly in the client files, the index.html file is just a collection of examples pulled from Stack overflow and Code pen.
* General lack of organisation – The entire repo for the framework is very poorly organised, it is unclear what each folder and file represent, and is hard to navigate.

In the industry, it is very rare for a company to create their own framework such as this one. There is little reason to when there are solutions already available with faster spin up time, feature completeness and not having to use the time, money, and effort to create your own.

# Proposed Technical Direction

## Server Implementation

### Language Justification

There are numerous languages that are used for server framework programming. Currently, “Just above 80% of websites are running on PHP” (Thor, 2018). However, this does not mean it is the best choice. PHP was not defined, it was organically developed and so whilst that does mean it has huge community support, it does mean it is internally inconsistent.

Python should be used; it is a desirable choice because:

* Syntax - It has a ‘clear and easy to read syntax’ (Thor, 2018) , meaning it can be easily learned and picked up quickly by new developers
* Used for many frameworks – e.g. Django, Falcon, and flask, so is already a language of choice for many developers.
* Libraries - particularly for using and visualising numbers e.g. NumPy and Matplotlib. This extendibility is not seen in PHP and Perl.
* Portability - supported on all modern OS’ and is well supported by web browsers. Don’t have to worry about porting code for other platforms.

Perl is not recommended, due to its age. It was used out of necessity when HTML was only in text files, there is no longer reason to use it except for people's familiarity. If FreeCycle want to future proof their systems, Python is the best option.

### Framework Justification

Since Python is being used, the most obvious framework to use is Django. According to Mozilla (2021), Django is:

* Complete – Has all the features a developer will need out of the box
* Versatile- works with any client framework
* Scalability – Each part is independent of others meaning that as traffic increases, more hardware can be added without affecting the framework itself.
* Portability – Benefit of using Python.
* Administration - Built-in login/administration system, saves the developers from having to build their own.
* Industry standard - Users include Instagram, Spotify and Mozilla.

However, Django is not suitable for smaller scale projects, and it also by default does not have support for web sockets. It also has a fairly high spin up time (not as high as Laravel or Ruby on rails, but still high). An alternative could be Falcon. Falcon is:

* WSGI compliant - this means requests are received from the client via the webserver. The framework is a layer of middleware. It is the separation of application and server code.
* Very light weight
* Very quick to set up
* RESTful – uses less bandwidth, more efficient
* Easy to debug – few files and simple logic paths.
* High scalability
* Portability – Again, a benefit of using python.
* Industry standard - users include LinkedIn, OpenStack and Opera.

When developing the project, the developers should initially use Falcon due to its fast set up time and scalability. It will be suitable for the first few months when the project still has lower traffic. Meanwhile, development can focus on implementing the Django framework. Since Django takes longer to set up, it will need more time. Time will also have to be spent ensuring that there is support for web sockets.

I suggest using these over popular examples such as Laravel and Ruby on Rails (ROR). This is because both are very large, and whilst Laravel has a whole ecosystem of features (including a micro framework), it has 4gb of decencies that must be installed. The languages that they use also limit them (PHP for Laravel and Ruby for RoR). RoR is a poor choice because it requires JavaScript, SQL and Yarn to be installed to use. It was released in 2004 and so is getting quite old, its popularity is declining every year.

## Client Implementation

### Language Justification

With regards to client frameworks, nearly all of them are written in JavaScript and HTML/CSS, or in some cases one or the other. For instance, React uses JSX which is the developers (Facebook) own extension of JavaScript that is used ‘with React to describe what the UI should look like’ – React (2021). Vue on the other hand just uses HTML/CSS and JS, meaning that there is a higher availability of people able to use it, as these languages are very common.

### Framework Justification

The best client framework to use for this project is Vue. Whilst React is currently the most popular web framework (Statista 2021), Vue is better suited. Some of the benefits/ features of Vue are:

* Language- as stated above, only needs HTML/CSS and JavaScript knowledge - easier for less experienced frontend developers to use.
* Open-source project – free to use, fewer security concerns (lots of people working on it to notice them) and designed with user needs in mind.
* Clear design vision – initially created by one developer, who according to Nowak (2021), wanted to combine ‘the best approaches to frontend development from Angular, Ember, and React with other features that made writing Web apps faster, easier, and more pleasant.’
* Scales up and down – React can scale up well but not down like Vue can if the requirements were to shrink.
* Performance - ‘For the sake of speed, Vue.js utilizes virtual DOM (Document Object Model): Think of this as a copy of an original DOM that figures out what elements to update, without rerendering (sic) the whole DOM. This approach makes page rendering pretty quick and improves application performance.’ (Altexsoft, 2019). This has been proven to be faster than React and Angular in testing.

Like how the server should be developed, the client side should begin development using a smaller framework, in this case, Hyperapp. I have chosen Hyperapp because:

* Small - only takes up 1kb.
* Extremely low spin up time - very fast to set up due to small size.
* Excellent choice for early releases - as stated by Chalaris (2018), ‘It provides fewer tools to work with …, but it can help you prototype quickly'
* Adaptable – 'code you write for it is compatible with the other two frameworks (Vue and React) either out of the box or with slight changes.’ (Chalaris, 2018). It is easy to adapt existing Hyperapp code to a different framework later.

At first the developers should focus on developing a small, stable release using Hyperapp to get something out as soon as possible. They should then spend time adapting its code to Vue, which is a more long-term and stable solution.

# Roadmap

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Time | 0 months | 2 months | 3 months | 4 months | 6 months | 7 months | 8 months | 9 months |
| Job - server | Set up Falcon server framework | Deploy falcon server | Begin to develop Django framework |  | Begin web socket development for Django. |  | Test Django server | Deploy Django server |
| Job - client | Set up Hyperapp client framework | Deploy Hyperapp client | Begin to develop Vue client | Start porting Hyperapp code to Vue | Test Vue client | Deploy Vue client |  |  |

# References

Thor, Wei-Ming (2018) *5 top programming languages to learn server-side web development* Accessed at: <https://twm.me/best-programming-languages-and-frameworks-for-server-side-web-development/> Accessed on 25.10.2021

Mozilla (2021) *Django introduction* Accessed at: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction> Accessed on 29.10.2021

Statista (2021) *Most used web frameworks among developers worldwide, as of 2021* Accessed at: <https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/> Accessed on 2.11.2021

Norwak, Maja (2021) *Vue vs React in 2021: Which Framework to Choose and When* Accessed at: <https://www.monterail.com/blog/vue-vs-react-2021> Accessed on 2.11.2021

Chalaris, Angelos (2018) *JavaScript Framework Comparison with Examples (React, Vue & Hyperapp)* Accessed at: <https://hackernoon.com/javascript-framework-comparison-with-examples-react-vue-hyperapp-97f064fb468d> Accessed on 2.11.2021

React (2021) *Introducing JSX* Accessed at: <https://reactjs.org/docs/introducing-jsx.html> Accessed on: 4.11.2021

Altexsoft (2019) The Good and the Bad of Vue.js Framework Programming Accessed at: <https://www.altexsoft.com/blog/engineering/pros-and-cons-of-vue-js/> Accessed on: 4.11.2021