CS CAPSTONE TECHNOLOGY REVIEW

NOVEMBER 10, 2018

TESLA WEB APPLICATION

PREPARED FOR

INGINEERIX, INC

PHIL SADOW

PREPARED BY

THE INGINEERS

BURTON JAURSCH

Abstract

One thing that makes Tesla cars unique is that Tesla, their manufacturer, doesn't want their cars to be repaired after they have been totaled. Because of this, Teslas that hold a salvage title do not get all the support that a new Tesla would get, including access to the Tesla app, which gives the user a control panel for their cars that they can use from their phones. There has been some work done on an existing web application that can be used on these repaired Teslas. This has been initiated by Ingineerix Inc., however, it requires an overhaul to the back-end to control the car through the vehicle API, and enhancements to the application to improve the user experience and usability.

CONTENTS

Introduction

At a high-level, the Tesla Web Application seeks to replace the official Tesla native iOS and Android application with a web-based application for owners of cars that have been blacklisted from Tesla due to the vehicle being totaled or considered by Tesla to be ”beyond repair”. These Teslas that have been repaired lose the ability to connect to the official Tesla app and cannot use the features the application once had. Phil Sadow has been working to bring these vehicles back to life, and has found a need for a better application than his current implementation of this app. Our job is to replace this application and provide similar usability and functionality as the official Tesla app.
There are numerous steps to make this happen and pieces of technology we will need to consider in order for this project to really make an impact. First off is a complete redesign of the back-end of the application that provides the data that the application and vehicle need to communicate effectively. In this tech review we will examine some of the frameworks and languages of this back-end. Furthermore, we will also need to consider where this server will be stored for Phil and what possible decisions associated with the location we will need to make to ensure his long-term success with his business.

1 Back-End Frameworks

Back-end frameworks are the collection of tools and languages used in server-side programming. This is the core logic that the application uses and especially the information that is given and taken by an application [1]. Within the Tesla Web Application, this will generally be the agent that will communicate with the Tesla API, server, and front-end. It will be important for this application to relay information regarding the car quickly and easily for the front-end to render the information for the user. Important criteria for picking a good back-end application could be as follows:

- Learning Curve - How easy will it be for our team to learn how to develop with the framework?
- Core Library - Is the framework more on the simple constricting side or complicated yet open?
- Vulnerability - Established frameworks with committed developers can have quick patches and present bug tracking.
- Documentation - Is there help and guides with the framework when developers need it?
- Hosting mode - If an application can be hosted on a dedicated server or shared, and how willing the framework is to support that choice. *This will depend on Section 3, Location of Server!*

1.1 Django

Django is a high-level Python web framework that is a great addition to database-focused web applications [1]. This framework uses the Model-View-Template architecture specifically for reusability and seamless plugin ability. This architecture is pretty similar to a standard MVC pattern and uses clear paths for user navigation like url resolvers, middleware, and view processors.

1.1.1 Pros of Django

The highly marketed part about this framework is the customizability of the framework, as virtually all components of the back-end tools can be swapped out like the template framework, Object-relational-mapping, and database. The framework is very popular and has a huge community for support [2]. Finally, because of Python, Django is pretty secure against common attacks and providing admin access to developers.

1.1.2 Cons of Django

Some of the issues faced by Django are speed and lack of convention. Django works poorly with static, 1-page applications. Though this does not mean it's designed for a industrial scale, python is not the fastest language out there. To make up for this though, Django can benchmark itself and provide abilities to find areas of slowdown. Furthermore, there is a lot of need in Django for specific definitions, and its not always willing to accept what it is given. This can slow down development, but is common within Python-based frameworks. [3]

1.2 Laravel

Laravel is a PHP framework specifically designed for extensive backend web application. Similar to Django, it follows the MVC pattern. This framework is fairly new, as it was introduced in 2011 and has become popular due to features like database migration, an intuitive interface, and blade template engine.

1.2.1 Pros of Laravel

Especially compared to other frameworks, Laravel is known to be extremely easy to use. Its use in the industry is pretty expansive and can be found in virtually all types of applications on the web. It also handles scale pretty well and can help develop small and enterprise-level applications.

1.2.2 Cons of Laravel

The issues with Laravel are pretty dispersed and context-dependant. Laravel has had known issues when being deployed on shared-hosting servers. This may become an issue if the application is deployed on a shared server. Furthermore, Laravel has a liberal use of database queries that can slow down performance, yet is not an issue as long as the traffic is minimal for the database. Finally, Laravel can struggle with mobile applications due to the amount of data Laravel sends on a full-page reload [4]. This definitely could become an issue as the webpage will be designed to be displayed on a phone.

1.3 Flask

Flask is very commonly considered the most popular Python web application framework that avoids using tools or libraries. Flask uses a RESTful request dispatch that provides inter-network interoperability. It is very common for the framework to be implemented on integrated devices due to the simplistic and light-weight design. [2]

1.3.1 Pros of Flask

Lots of the perks of Flask come from the simple design and architecture of Flask. It is extremely flexible and has a small learning curve that some of the other frameworks explored don't come near. The routing of URLs is very simple within Flask so it is very quick to help prototyping and starting small apps.

1.3.2 Cons of Flask

Like the perks, some of the downsides of Flask come from this lightweight approach. Specifically limited to the Tesla application's goals, Flask is not friendly to asynchronous refresh, and does not have integrated support for databases and Object-relational-mapping. Ultimately, Flask lacks features due to not having any tools or libraries that it can use. [5]

Framework	Pros	Cons
Django	Lots of support Very customizable	Issues w/ speed and conventions Not good with small web apps
Laravel	Extremely easy to learn Handles scale well Uses PHP	Issues with shared server-space Not good with mobile applications
Flask	Simple design Very quick to develop in	Very little support from the framework Not good with advanced web features (ex: async refresh, database support, ORM)

Table 1: Quick Analysis of Back-end Frameworks

2 Back-End Languages

While languages are tightly coupled with frameworks, it is important to look at the individual languages and see how they stack up against each other in the context of the Tesla Web Application. Languages will almost always be an indicator of what to expect with certain frameworks as well as be the key piece in designing a proper back-end. Furthermore, understanding the pros and cons of certain languages will help our project go beyond even the extents of what a framework can help support. If there are features we want to add that a framework may not provide direct help with, we will have to hope that the language we've chosen can pick up the slack and easily guide us to creating what we imagine [6]. Criteria for these languages is similar to frameworks, but as follows:

- Learning Curve - Is the syntax easy to understand and read? Is it similar to other languages we know from experience or in class?
- Support - Are there popular frameworks that use this language? Is there an active community that helps learners of the language?
- Security - Is the language set up to provide or allow packages for extra security? Are there known issues with the language that can compromise our application's data?
- Speed - Is the language known to be quick in execution? Can the language support the application's usability goals?

2.1 PHP

Hypertext Preprocessor is an extremely common and popular server-side scripting language designed specifically for web development in 1994. It provides a close comparison to C without having some of the issues that C generally has when learning the language. Furthermore, it is designed to be concise, and this opens the door for a variety of flexibility.

2.1.1 Pros of PHP

One of the greatest strengths of PHP is the ease of learning the language. PHP provides low barriers to entry and is quick to learn due to the extensive documentation and community support for those trying to learn the language. The lack of resource management seen in C allows for those interested in server-side coding to do as they please, and quickly. For dedicated developers in PHP, there is much more to explore due to plentiful libraries and extensions created by the community for virtually any feature imaginable.

2.1.2 Cons of PHP

Lots of the strengths of PHP ultimately lead to the issues facing the language. One of the biggest issues with PHP is just how easy and welcoming the language can be to developers. This can cause a lot of bad PHP applications to be released and open for the public to use, and due to the number of people learning the language, these practices can spread quickly. Though PHP may be considered a pretty secure language, the low barriers to entry have caused a variety of easily hackable applications and low security expectations for the language. [7]

2.2 Python

Python is another very common and popular language that has been with the server-side scripting language community since 1991. Python is highly-portable and has a multi-programming paradigm that heavily involves object-oriented programming throughout its design. It is very common for Python to be used in tandem with Java, and has the reputation for being able to get a job done with fewer lines of code.

2.2.1 Pros of Python

One of the biggest perks of Python is the easy syntax that makes understanding your code simple and quick. This allows debugging and error checking to move much faster relative to other languages. Python is generally considered to be faster than PHP as well as more secure. Yet like PHP, it has lots of libraries and community support with popular frameworks such as those listed above. Finally, due to the portability of Python, it can be used for mobile development easily, and can interact with numerous web features.

2.2.2 Cons of Python

Though a gross overstatement, Python can be considered an opposite version of PHP. This is largely seen in the steep learning curve Python that can slow development when experience with the language is lacking. Python can be very memory-intensive compared to Java, and can create performance drops during run-time. While this is not a huge issue for the application, 3D graphics can take a hit if they were to be implemented in the Tesla Web App. [8]

2.3 NodeJS

NodeJS is a pure JavaScript approach to back-end development first released in 2009 by Ryan Dahl. This language was designed to be incredibly quick and dynamic in its development. One of the biggest features of NodeJS is the single threaded event call back mechanism which allows scripting languages to communicate with network programming.

2.3.1 Pros of NodeJS

Out of the languages explored here, NodeJS is the fasted for sure, especially in its initial stages. It can handle common features in today's web applications like data-streaming, queued inputs, and proxy quite well. NodeJS works well with organizing multi-core systems and responding to concurrent requests. Finally, the application is easy to deploy and monitor.

2.3.2 Cons of NodeJS

One of the main issues with the language is the lack of coding standards for its parent, JavaScript. Without a strict style guide, NodeJS can break easily. The support IDEs have with NodeJS is not necessarily great for features like Call-backs, debugging, error handling, and other maintenance features. [8]

Language	Pros	Cons
PHP	Easy to learn Variety of libraries & extensions	Bad reputation for poor open-source code Possible issues for security hacks
Python	Simple syntax == quick debugging Very portable Good for mobile development	Steep learning curve Memory-intensive
NodeJS	Fastest of listed languages Adapted to current web app features Easy to deploy	Lacks coding standards Supported IDE's not very powerful

Table 2: Quick analysis of back-end languages

3 Location of Server

By location of server, yes, we mean the physical location of our server that will host the Tesla Web Application. While websites and web applications are seemingly headed for the cloud, we believe this project is going to be pretty particularly focused on those within our client's area, so the idea that the server should be sent to the cloud may not be necessary, and want to explore the other options. From a technical standpoint, as long as a site is hosted in the same country, there really isn't much worry about slow connections. Especially with the use (or lack thereof) the server would have, there wouldnt be a heavy load that would compound with the distance the server would be from its clients [9]. Instead, this would be pragmatic reasoning, where the server could have maintenance and easy supervision or upgrades. The criteria of this search will be much more subjective and weigh the options in a less linear fashion.

3.1 Amazon Web Services

AWS is a suite of cloud computing services that can provide a range of features for business and individuals alike. In the context of the Web Application, the database and other back-end resources would be stored using AWS services, making the web application completely in the cloud. AWS has been around since 2006, but is quickly becoming a leader in the cloud hosting industry. [10]

3.1.1 Pros of AWS

The main perks of using AWS is the amount of computing power and resources Amazon has through AWS. Especially considering the growth potential for our client's business, scaling would be incredibly easy. Deploying new updates or changes to the back-end wouldn't necessarily be difficult as lots of the work could be done by Amazon. By using cloud computing, we would be at the cutting-edge of technology and would set a precedent for using new technology within our client's business.

3.1.2 Cons of AWS

Some of the main drawbacks of AWS generally come from the scale of AWS as a whole. One of the biggest issues would be the possible threats from hackers who have had a history of trying to break into AWS. Another issue would be that the services would cost money. [11] While our client would undoubtedly be willing to pay for a feature such as our application, he may not be interested in a continuous payment when other options might be free.

3.2 Oregon State Servers

While far-reaching, we could potentially host the Tesla Web Application on Oregon State Engineering Servers. Our servers at Oregon State have had a great history of hosting a variety of projects. While many of these projects have been research-based, it could very well be an option to support a business associated with our capstone program.

3.2.1 Pros of Oregon State Servers

One of the main pros of having the application's server located on-campus is that our team could quickly connect to the server and provide quick uploads or patches if need be. This shows the support of the University towards our clients project and may foster a relationship with our client to continue providing projects for the capstone program

3.2.2 Cons of Oregon State Servers

The main issue with hosting the server on-campus, would be the extended stay the server would have and the problems that would come up in the event the server would want to be moved or changed in a drastic way. This could be used as leverage against our client in an event where there are disagreements either during our project or after. Our client may not be open to having his project potentially on display for students and faculty not involved in the project.

3.3 Client Server

Finally, we could host the back-end of our project on our client's server, as it was in its first iteration. While we aren't sure of the technical details regarding our client's hardware, we can assume it should be able to host a similar load as it has been currently with new technology.

3.3.1 Pros of Client Server

Obviously the biggest positive to having the server with our client is so once the project is complete, he can continue maintenance easily and without worry about where his work will go once our team moves on. A big issue that will be raised is what will happen with maintenance on the web application once we are gone, but in this event, our client will easily be able to access our files and edit them as necessary.

3.3.2 Cons of Client Server

The only downside we currently see as of now is that we will not be able to physically deploy the back-end on our clients server and will need our client to do so whenever there is a patch or change that will go live on the server. While this is an issue, as long as communication is strong with our client we assume we can work past this.

Server Location	Pros	Cons
AWS	No issues with memory or processing power Easy to scale Cutting-edge	Paid service Possible target for attacks
Oregon State	Quick connection to team Get support from University	Possible ”political” issues post-project dev Client’s privacy may feel violated
Client Server	No ”political” issues Easy to maintain post-project	Team cannot physically deploy changes

Table 3: Analysis of Server Location

Conclusion

There are a variety of decisions we will need to make when it comes to the back-end of our project, and it goes beyond what is listed in this tech review. While some of these decisions have already been decided, it is important to look into the different possibilities our system could be designed with and understand if the choices we made are appropriate. From this tech review, we have made a couple recommendations and decisions regarding the back-end of our application and physical location of our server.

First off, our applications server will remain with our client and his resources. This has been decided previously, but we felt it necessary to explore other options if they exceedingly outweighed keeping the server local to our client. Because of the dubious future of the project once our team has given over control back to our client, we decided it was best to keep the server in our clients control throughout our tenure.

In regards to the back-end of our application, I recommend that we use the Django framework in our development. One of the main concerns we have is the security issues associated with PHP, and while it can be easy to develop in, we are worried that our lack of industry experience could result in security breaches. We want to develop in a safe manner and feel that Django as a framework has great built-in security features along with a rock-steady Python foundation. Further, speed is not necessarily a worry as the server at this time will not face huge issues with activity as only a few hundred people are currently using our clients application, and there have been no reported performance issues. While other frameworks can still be considered, Djangos support also seems fairly helpful and can work with the learning curve we will need to be ready for.

References

[1] S. K. Tripathi, "Top 7 backend web development frameworks in 2018," Mar 2018. [Online]. Available: https://www.kelltontech.com/kellton-tech-blog/top-7-backend-web-development-frameworks-2018

[2] M. Malhotra, "Top 7 backend web frameworks to use in 2019 hacker noon," Jul 2018. [Online]. Available: https://hackernoon.com/7-best-web-development-backend-frameworks-in-2018-22a5e276cdd

[3] M. Poczwardowski, "Pros and cons of django - when to use python framework," Jul 2018. [Online]. Available: https://www.netguru.co/blog/pros-and-cons-of-django-as-framework-for-python-development

[4] R. Campbell, "Php development with laravel comparing the pros and cons," Aug 2017. [Online]. Available: http://www.bitrebels.com/technology/php-development-laravel-pros-cons/

[5] A. Dojo, "Choosing python web frameworks: Django and flask," Feb 2016. [Online]. Available: https://www.codingdojo.com/blog/choosing-python-web-frameworks/

[6] J. Long, "I don't speak your language: Frontend vs. backend," Dec 2016. [Online]. Available: https://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend

[7] J. O'Dell, "8 experts break down the pros and cons of coding with php," Nov 2010. [Online]. Available: https://mashable.com/2010/11/19/pros-cons-php/#Z3fzcwiVSEqF

[8] Intersog, "Developing solutions in nodejs vs python: Pros and cons," Jul 2017. [Online]. Available: https://medium.com/@Intersog/developing-solutions-in-nodejs-vs-python-pros-and-cons-4ab4cea68ff0

[9] Goran, "Does it matter where my server is physically located?" 2016. [Online]. Available: https://www.altushost.com/does-it-matter-where-my-server-is-physically-located/

[10] "What is aws? - amazon web services." [Online]. Available: https://aws.amazon.com/what-is-aws/

[11] K. Davis, "Pros and cons: Digging into amazon web services," Jun 2017. [Online]. Available: https://npifinancial.com/blog/pros-and-cons-digging-into-amazon-web-services/