CS Capstone Technology Review

November 22, 2017

Aerolyzer

Prepared for

NASA JPL

Kim Whitehall

Prepared by

Daniel Ross

Aerolyzer

Daniel Ross

Kin-Ho Lam

Logan Wingard

Abstract

This Technology review covers Web Frameworks, Database Systems, and Visualization & Display. The Web Framework for the Aerolyzer project must be fast and secure. The Database system has to be reliable and fast. The tools used for visuals have to look good and have all the features that development may require.

Contents

1 Web Framework

1.1 Overview

Web frameworks are what will enable the Aerolyzer library to interact with the users, so the framework's performance will directly affect the end product of the Aerolyzer app. Depending on which framework the Aerolyzer app utilizes, the integration of the Aerolyzer python library could be simpler. The web framework will also inform the structure of any user input, so clarity in this aspect is critical.

1.2 Criteria

1) Speed
2) Security
3) Web Development features

1.3 Options

1.3.1 Express.Js

Express is a streamlined version of Node.js. Express itself doesn't have database support, but is capable of running third-party modules to interface with the database. Express is useful for API development as it includes the neccesary http functionality to provide users functions. Since Express is a form of Node, an app built on Express will crash if an exception goes uncaught. This can make potential debugging a hassle since the app will have to be restarted completely if anything goes wrong. Express typically performs a little quicker than it's parent framework, Node.js, but it's speed can suffer significantly under high traffic. Security isn't built in Express functionality, but there is third-party middleware that sets up HTTP headers and transmission security. [1]

1.3.2 Ruby on Rails

Ruby on Rails, 'Rails' for short, is a Ruby based web framework. Rails claims to value web application conventions and will default to many of those conventions in the case that the developer didn't make custom configurations. This hopes to make Rails code shorter and less buggy than other frameworks that require minutiae from the developer. Rails runs off of a SQLite3 database and is best suited for interfacing with that type of database. Rails has a default HTTP security built directly into the framework. Being written in Ruby and having many features built-in means that Rails isn't the fastest framework. [2]

1.3.3 Django

Django is a python based framework that claims to make database-driven Web development fast and easy. Django also supports clean URL designs and in framework templating. The first step to using Django is writing the model which maps out your database layout. [3] Since Django apps are centered around these models, Django has a built-in API for manipulating any objects in the model. [4]

1.4 Web Framework Discussion

Express immediately takes 3rd place amongst these choices, because it's overly reliant on third party middleware[1]. Rails and Django are similar in respect to included features, but Django has it's model API for increased ease of use[2][4].

1.5 Conclusion

Django offers the most complete set of framework features while maintaining speed. Since Django is written in python and the Aerolyzer library is also written in python, integrating the Aerolyzer library would be easiest with Django. [4]

2 Database Management System

2.1 Overview

A database is how computers store data and its relationships with other values. For Aerolyzer to work properly we'll need a database to store photos and weather analysis under the zip code they apply to. Since the 3 most reasonable choices are all SQL based, the management system that works with the web framework will most likely be the best choice.

2.2 Criteria

1) Speed
2) Size

2.3 Options

2.3.1 SQLite3

SQLite boasts it's status as the most deployed database in the world[5]. SQLite manages the SQL database within a single disk file and the library itself is about 1.8 MiB[5]. Since SQLite is self-contained and lightweight it's suitable for most low traffic applications[6]. Where SQLite starts to suffer is it's speed when given requests for data that are large relative to the available memory. For Aerolyzer this may be problematic, since the database hase to be capable of fetching high definition pictures quickly[5].

2.3.2 MySQL

MySQL is Oracle's SQL database server. MySQL has a larger library to install and has administrative functionality included. This means that MySQL would require considerable set up and potential maintenance. The size and effort required for MySQL is all in order to provide a database that can handle large requests and high traffic.[7]

2.3.3 PostgreSQL

PostgreSQL is the SQL supporting descendant of the POSTGRES database system. PostgreSQL includes a wide function library with Python support. The transfer size PostgreSQL supports is more than enough for the Aerolyzer app. The administrative functions included in PostgreSQL would make monitoring the database efficient.[8]

2.4 Database Management System Discussion

All three database systems would work in the Aerolyzer app, but the question remains which would work the best. PostgreSQL and MySQL would most likely have similar performance once set up, while SQLite might have issues under high traffic.

2.5 Conclusion

MySQL is the most complete SQL system of these choices, and its performance is most likely going to be similar to PostgreSQL. The fact that MySQL has a more expansive library means that it's more likely to have the database solutions we need during development.

3 Visualization & Display

3.1 Overview

Presenting the output from the Aerolyzer library is the key step in making the user understand the project. Having unclear or unappealing visuals to show the results could potentially be worse than simple outputs. Most of the presentation can be acheived with HTML alone, but the main display will need a script to insert charts.

3.2 Criteria

1) Clarity
2) Reliability

3.3 Options

3.3.1 Google Charts

Google Charts is a JavaScript that is inserted into the web page and is then fed the data to be charted. There are 28 different types of charts and they all are rendered as SVG or VML. Google Charts lets the chart make a query to the database directly and displays customizable tooltips when moused over.[9]

3.3.2 CanvasJS

CanvasJS, in addition to having 30 types of charts, claims to be an ultra fast HTML5 charting library. Rather than render SVG charts, CanvasJS inserts HTML code when rendered. With a wide library of customization, CanvasJS claims to make the developer's charts work on Chrome, Firefox, Safari, and Internet Explorer.[10]

3.3.3 ChartJS

ChartJS renders charts into HTML5, like CanvasJS, but is open source. ChartJS only has 8 chart types, but offers custimization that lets developers mix chart types.[11]

3.4 Visualization & Display Discussion

CanvasJS is the most robust JavaScript of the 3, which may take some additional code to get the desired results, but the end product will most likely be closer to the ideal visualization. [10]

3.5 Conclusion

Having visuals rendered into HTML5 elements would be ideal. It ensures that the graphics are rendered with the rest of the page. Of these three libraries, CanvasJS is most appealing but isn't open source. Aerolyzer is an open source project and avoiding conflicting licences is important. Since this is the case, Google Charts is the best option since it's terms of use won't interfere with Aerolyzer's liscencing.[9][10]

Database Management System		
Choice	Pros	Cons
SQLite3	Lightweight, Fast	Not suited for high traffic
MySQL	Fast, Robust features	Large, Administration required
PostgreSQL	Fast	Administration required

Web Framework		
Choice	Pros	Cons
Express.js	Lightweight, Quick	Requires Middleware for many features, Prone to crash
Ruby on Rails	Robust features	Not particularly fast
Django	Robust features, Quick	Largest framework

Visualization & Display		
Choice	Pros	Cons
Google Charts	Free to use, sufficient features	not as fast as CanvasJS
ChartJS	Open Source	Not as many features
CanvasJS	Very Robust	Not Open Source

REFERENCES

[1] (2017) Express guide. Node.js Foundation. [Online]. Available: https://expressjs.com/en/guide/routing.html
[2] (2017) Rails guide. [Online]. Available: http://guides.rubyonrails.org/getting_started.html
[3] (2017) Getting started with django. Django Software Foundation. [Online]. Available: https://www.djangoproject.com/start/
[4] (2017) Django at a glance. Django Software Foundation. [Online]. Available: https://docs.djangoproject.com/en/1.11/intro/overview/
[5] (2017) About sqlite. Hwaci. [Online]. Available: https://sqlite.org/about.html
[6] (2017) Appropriate uses for sqlite. Hwaci. [Online]. Available: https://sqlite.org/whentouse.html
[7] (2017) Mysql 5.7 reference manual. Oracle. [Online]. Available: https://dev.mysql.com/doc/refman/5.7/en/
[8] P. G. D. Group. (2017) Postgresql 10.1 documentation. [Online]. Available: https://www.postgresql.org/docs/10/static/index.html
[9] (2017) Google charts. Google. [Online]. Available: https://developers.google.com/chart/glossary
[10] (2017) Canvasjs. CanvasJS Team. [Online]. Available: https://canvasjs.com/docs/charts/basics-of-creating-html5-chart/
[11] (2017) Chartjs. https://github.com/chartjs/Chart.js/graphs/contributors. [Online]. Available: http://www.chartjs.org/docs/latest/