Analysis of Alternatives

We Showed Up

The Google API had support for 8 different coding languages. This limited our choice of language for the project. Given that the team only had experience in three of these languages, we only accounted for these options. The three choices were JavaScript, Java and Python.

The factors we needed to analyse for choosing the language were how much experience the team has, how much extra will have to be learnt to implement the app, OS limitations and ease of GUI output. The client agreed that it could be a desktop app, which ruled out the need for compatibility with phones and tablets. All of these factors are raised below for analysis.

**JavaScript**

After surveying the team, the majority of our JavaScript experience was coding a mobile app in ENG1003 Mobile Apps. To have a console output, we would have to run the .js file in node. None of us had experience using node and our JavaScript experiences from 1003 were negative. The sole advantage of JavaScript was that it is easy to create a GUI for output. In 1003 we all learnt how to use Material Design Lite to create a GUI in the form of a website.

Advantages

* Easy to create a web-app embedded in HTML
* Can run it all as one webpage

Disadvantages

* The team is unfamiliar with the language
* Complicated language compared to Python

**Java**

The team has very limited experience using Java, only 1 of us had actually used it, and only

in FIT2099. Even with this, we still preferred it to using JavaScript due to our negative Mobile Apps experience. To output using Java, we could output to the terminal, or code a complete GUI in an unfamiliar language, which is not ideal. The only way to make Java compatible with a web page would be to make an output file for javascript to analyse or implementing a framework such as JQuery. Given our limited Java coding skills, it seemed like a hard choice to choose.

Advantages

* Support for Object Oriented Design

Disadvantages

* The team is not proficient at coding Java
* Have to learn a framework in an unfamiliar language

**Python**

The majority of the team’s first coding language was python, making us very comfortable coding in it. We had all done complicated projects that implemented python and are confident in our abilities. To run the program, we can output to a terminal, create a GUI for output or use Django to create a website to display the data. To test if the task was truly possible in python, we did a spike. The results of it were that within 10 minutes of coding, some revision data was accessible.

Advantages

* All of the team has lots of experience coding in Python
* Simple language to use
* Already tested to work

Disadvantages

* Web based coding and implementation can be tricky to implement in python
* Creating a GUI can pose a challenge

|  |  |  |  |
| --- | --- | --- | --- |
| Language | Team Experience | How much is needed to learn | OS limited |
| Java | Low | High | No |
| JavaScript | Medium | Medium | No |
| Python | High | Low | No |

**Summary**

From the analysis of all the options, the team chose Python. This was mainly based upon the team’s high level of knowledge of the language in comparison to both Java and JavaScript. All three of the languages would require learning a framework or library for displaying the data outside the console. JavaScript had the advantage of being compatible with HTML without any frameworks, but to make it look good, it would still need a library such as MDL. Java was considered the worst option of the three as the team has a very low knowledge of the programming language. This ruled out java as the unfamiliarity of the language showed too much potential to cause issues.

Ultimately Python provided a reasonable middle ground between team members knowledge and the potential difficulty in setting up the relevant infrastructure and coding frame-work to carry out the project. This middle ground led to the logical choice in deciding to implement our project using python.