Assignment 1

CST8333 19F

Professor: Stanley Pieda

By: Lucas Estienne (esti0011)

040 819 959

## Research

* 1. **The TIOBE index**  
     The TIOBE index ranks programming languages based on search engine results, and the number of engineers, courses and vendors using the language.

Pros:

* + - Updated Monthly
    - Has data for programming language popularity dating to 2001

Cons:

* + - Is based on raw search results (number of web pages), perhaps not the most accurate metric with similarly named languages (C, Objective C, C#, etc. ). Also could cause some languages (i.e. Python) to be ranked higher due to “python” having other meanings.

[1]"TIOBE Index | TIOBE - The Software Quality Company", *Tiobe.com*, 2019. [Online]. Available: https://www.tiobe.com/tiobe-index/. [Accessed: 12- Sep- 2019].

* 1. **PYPL PopularitY of Programming Language**  
     The PYPL index is based on the search trends for tutorials for each language on Google. More searched languages are assumed to be more popular.

Pros:

* + - Based on popularity of learning resources
    - Based on Google Trends (i.e. popularity of search queries), not search result hits.

Cons:

* + - No historical data/comparison besides how ranking exactly one year ago
    - Limited to 22 languages.

[2]"PYPL PopularitY of Programming Language index", *Pypl.github.io*, 2019. [Online]. Available: http://pypl.github.io/PYPL.html. [Accessed: 12- Sep- 2019].

* 1. **Stack Overflow Developer Survey Results**The annual Stack Overflow Developer Survey Results provides insights on languages, frameworks, libraries, tools and more. The rankings are based on survey responses.

Pros:

* + - Based on responses from actual developers, not search results or search trends.
    - Large number and variety of responses (88883 responses, 179 countries represented)

Cons:

* + - Bias towards users of Stack Overflow (of which a large majority are either full-stack or front-end developers) which causes a heavy bias for JavaScript, for instance.

[3]"Stack Overflow Developer Survey 2019", *Stack Overflow*, 2019. [Online]. Available: https://insights.stackoverflow.com/survey/2019#technology. [Accessed: 12- Sep- 2019].

I would not say that any of the 3 websites was the “best”, it’s definitely a better idea to use all 3 to make an informed opinion as all 3 have different methodologies for ranking languages. If I had to pick one, it would be the one with the most “interesting” data, which would be the TIOBE index simply due to the fact that you can browse the data all the way back to 2001, but that doesn’t make it the overall best.

## Language Selection

I will be studying Go. I have from one to 3+ years of professional experience using all other languages listed on a day-to-day basis, since I’ve been working in an enterprise setting first as a Junior Applications Developer (8 months), followed by Intermediate Applications Developer (for 2 years), and now work as a DevOps engineer (last 7-8 months). (I had taken a break from Algonquin since the strike to focus on work, and I’m looking at a promotion to a Team Lead role around next May, which my boss has told me could be mine if I finished my diploma. I only have 3 courses + a PLAR left. My employer is sponsoring my return to school.)

Go is a language I have some familiarity with (but by no means proficiency) because of my current role (lots of interaction with Go-backed tools like Terraform, Hashicorp Vault, Kubernetes, Helm, and various others), and knowing Go is a definite asset when working with those tools, or extending them.

## Unit Testing

* 1. Go standard library “testing” package documentation

This is the documentation for the standard unit testing package, which it is best practice to use in Go. It’s fully featured, besides mocking. The documentation provides basic insight into how to write tests.

[4]"testing - The Go Programming Language", *Golang.org*, 2019. [Online]. Available: https://golang.org/pkg/testing/. [Accessed: 12- Sep- 2019].

* 1. Golang basics – writing unit tests

This is a post describing the basics around writing idiomatic unit tests, and running them, in Go.

[5]A. Ellis, "Golang basics - writing unit tests", *alex ellis' blog*, 2017. [Online]. Available: https://blog.alexellis.io/golang-writing-unit-tests/. [Accessed: 12- Sep- 2019].

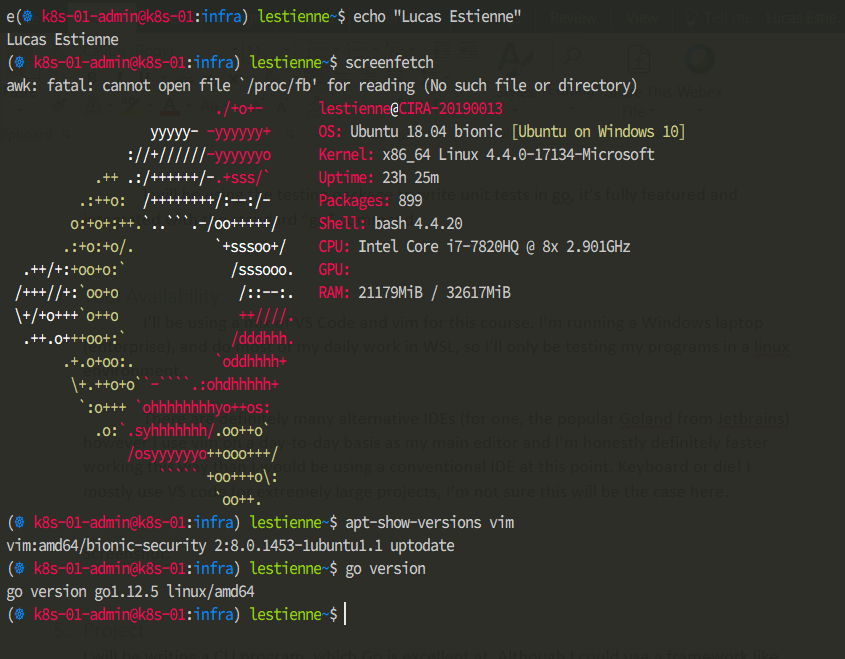
I will be using the testing package to write unit tests in go, it’s fully featured and integrated with the standard “go” command.

## Tool Availability

I’ll be using a mix of VS Code and vim for this course. I’m running a Windows laptop (enterprise), and do most of my daily work in WSL, so I’ll only be testing my programs in a linux environment.

There are definitely many alternative IDEs (for one, the popular Goland from Jetbrains) however I use vim on a day-to-day basis as my main editor and I’m honestly definitely faster working this way than I would be using a conventional IDE at this point. Keyboard or die! I mostly use VS code for extremely large projects, I’m not sure this will be the case here.

Screenshot:



## Project

I will be writing a CLI program, which Go is excellent at. Although I could use a framework like Buffalo and write a web app, I think that’s way too bulky for this course, and CLIs written in Go can be pretty sexy.

[6]M. Richards, "Software architecture patterns", *O'Reilly Media*, 2015. [Online]. Available: https://www.oreilly.com/ideas/software-architecture-patterns/page/2/layered-architecture. [Accessed: 13- Sep- 2019].

Layered architecture allows you to swap and reuse components at will, between even between multiple applications. Components can be individually maintained, updated, and deployed, at different times.