Coding skill evaluation test

Summary

We are relying on a webview library to ship cross platform graphical applications, which is split into two repositories:

- https://github.com/webview/webview
- https://github.com/webview/webview_go

The library we use works by providing bindings to the native OS webview libraries (msedge on Windows, cocoa-webkit on macOS and libwebkit on Linux) to the Golang language.

However, this library does not support setting custom user agents yet, which functionality should be implemented.

Goals

There is initial work done by the original author to support this option on Windows (msedge) in the core library, however this functionality is not exposed in the Go bindings yet.

There are two goals:

- Implement the same functionality for either linux OR macOS (will need both later)
- Expose the feature in the Go bindings

Implementation requirement

The Golang API to be implemented for accessing the feature should look like this (based on the original example):

```
package main

import webview "github.com/webview/webview_go"

func main() {
	w := webview.New(false)
	defer w.Destroy()
	w.SetTitle("Basic Example")
	w.SetSize(480, 320, webview.HintNone)
    w.SetUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/139.0.0.0 Safar
	w.Navigate("https://www.whatismybrowser.com/detect/what-is-my-user-agent/")
	w.Run()
}
```

Evaluation criteria

- A build script should be provided for the chosen platform of implementation (Linux or macOS)
- The build script should also install the necessary dependencies (there is none on macOS)
- The go source code should match the example defined above
- The build script should build the go source code into an executable binary
- Upon executing the binary, it should load the webpage which is supposed to display the user agent defined in the source code