Contents

1 Go Resume Design Document

1.1 Design Document Contributors

- Jorge Henriquez (contact@jorgehenriquez.dev)

1.2 Introduction & Motivation

If you're in the job market you need to have a couple resumes, plural. This is because if you scour the internet for job seeking advice, you'll find that one of the many recommendations is to tailor your resume to each job you apply to. That's hard.

Having to locate your `.docx` (or `.tex` if you're fancy), edit it (which can be annoying if you're using a word processor), and convert it to a `.pdf`, *is a pain*. All I should I have to do is just build my base resume once, then edit a small file to produce a resume. This is where ***Go Resume*** steps in.

1.3 Overall Goal

The good folks over github.com/saadq/resumake.io have made an amazing resume system. So amazing that a `pandoc-loving-can't-write-latex-to-save-my-life` nerd can use it and make amazing resumes! However, I just want to be able to use it once to build the initial system, then use a `CLI+yaml` to edit my resumes just how I want it.

The good news is that once we create the resume on their website we can download a `resume.json` file that holds our entire resume! Even better, the website also has an API endpoint that'll produce our `resume.pdf` from our `resume.json`! If we use a configuration file and some good old-fashioned programming, we can create an easy resume system!

While they do have a live website it would be rude to bombard their servers. So we'll just add their repo as a sub-module in our repo! Then we can just build their code and we'll be off to the races.

Also, I want this to be cross-platform so we'll try to avoid any *NIX or Windows specific tools, with the obvious exception, of this `Makefile` generated design document. Whoops…

1.4 Design

To motivate the design let's examine the use case of the program.

1. Build the initial resume using the website generator.
2. Download the `resume.json`
3. Configure a yaml file that has the desired-job attributes (skill and projects).
4. Combine the yaml file with the `resume.json` to yield a `customized-resume.json` file.
5. Send the customized file to the api endpoint.
6. Receive and write the `resume.pdf` to the file system.
7. Submit the application.
8. Find a new job posting
9. Goto 3

The first thing that stands out is having to run a script that start up the `resumake.io` services. Since a goal of this project is to be cross platform we can't use a `.*sh` file to execute the program. But we can use a programming language (we'll use Go for this) to create a CLI (`resume-start`) to automate this for us.

1.4.1 Launching a local instance of resumake.io

This task is pretty trivial. In our programming language, we just spawn a process that executes the instructions as specified by `resumake's` `contributing.md`.

To build `resumake.io` we need to run `npm run build` and `npm start`. Building takes a while and only needs to be done once. We'll provide an option to the CLI to skip building the frontend (`-skip-build`) with its default value set to false. To also allow for more developer flexibility we'll also provide a path to the resumake.io directory with the `-resumake-dir` flag. This is needed so that the tool knows where to execute the build instructions.

Additionally the process should be able to handle interrupts gracefully when running the server and client. It should return a success (typically 0) exit code when running them. If an interrupt is received during the build process it should return a failure exit code (typically 1).

Since the GUI is only potentially used once it makes little sense to run it every time the user needs to tailor a resume. Therefore, the `-no-client` flag should exist. This option will not run or build the client. It follows that it will also adhere to the `-skip-build` flag and have a graceful shutdown properties.

The following command (`resume-start`) will encompass the above design.

```
Usage of resume-start:
  -log string
        the file where resume-start logs to (default "./.resume-start.log")
  -no-client
        disable running the client
  -resumake-dir string
        the directory where resumake.io resides (default "./resumake.io")
  -skip-build
        skip building resumake.io dependencies
```

To build `resume-start` it should be as easy as running the following cross-platform shell command:

```
go build ./cmd/resume-start
```