# ← APG1 Bootcamp. Day06

Revie

## Submit the project

Finish project

## Survival camp

Show all

| | |
|---|---|
| APG1 Bootcamp. Day00 | ✓ 100% |
| APG1 Bootcamp. Day01 | ✓ 100% |
| APG1 Bootcamp. Day02 | ✓ 100% |
| APG1 Bootcamp. Day03 | ✗ 0% |
| APG1 Bootcamp. Day04 | ✓ 100% |
| APG1 Bootcamp. Team00 | ✗ 0% |
| APG1 Bootcamp. Day05 | |
| APG1 Bootcamp. Day06 | |
| APG1 Bootcamp. Day07 | |
| APG1 Bootcamp. Day08 | |
| APG1 Bootcamp. Day09 | |

## Private Git project

ssh://git@repos-ssh.21-school.ru:2289/students/Go_Day06.ID_376227/mlarra_student...

Copy link Open

Task

# Day 06 - Go Boot camp

## Fortress of Solitude

## Contents

# Chapter I

## General rules

- Your programs should not quit unexpectedly (giving an error on a valid input). If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work won't have to be submitted and won't be graded. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.
- Submit your work to your assigned git repository. Only the work in the git repository will be graded.
- If your code is using external dependencies, it should use Go Modules for managing them

# Chapter II

## Rules of the day

- You should only turn in `*.go` files and (in case of external dependencies) `go.mod + go.sum`
- Your code for this task should be buildable with just `go build`
- Additional steps, e.g. for creating tables in a database, should be included in *admin_credentials.txt*

# Chapter III

## Intro

So - this is it! You've just got superpowers! And, as a new superhero, you definitely need to think about your street cred and overall recognition, don't you think?

Fireballs - check! Almost indecently tight leotard - check! Secret base - oh yeah, baby!

Anything else you forgot? Any other shenanigans to perform? OH WAIT~

# Chapter IV

### Exercise 00: Amazing Logo

The only requirement here is to generate an awesome logo! It should be a 300x300px PNG file named 'amazing_logo.png'. Make it as cool as you can!

Image should appear in the same directory as the launched binary executable after compiling.

NOTE: you shouldn't cheat and download anything from the internet in your code. Just don't rely on any external sources and unleash your fantasy in programming! Also, just plain single color or transparent logo doesn't count!

# Chapter V

### Exercise 01: Place for Your Thoughts

Okay, so what else do you need now? A website, of course! It should be a blog where everybody will be able to read your ideas on the world improvement. Here is a list of features it should have:

- Database (you should use PostgreSQL)

- Admin panel (on '/admin' endpoint) where only you can login with just a form for posting new articles (let's forget about editing old ones for now, superheroes don't ever look back)
- Basic markdown support (so it can at least show "###" headers and links in generated HTML)
- Pagination (show no more than 3 thoughts on one page for people to not get too much of your awesomeness)
- Application UI should use port 8888

All additional files (images, css, js if you decide to use any of those) should be submitted as a *zip* file to be unpacked in the same directory as binary itself, resulting into something like this: .

```
├── css
│   └── main.css
├── images
│   └── my_cat.png
├── js
│   └── scripts.js
└── myblog-binary
```

Admin credentials for posting access (login and password) and database credentials (database name and user) should be submitted separately as well in a file called *admin_credentials.txt*. If there are additional commands to be run to create tables in a database, put them into the same file.

Main page should include your logo from EX00, links to articles and (optionally) some short preview of their content, as well as pagination (if there are more than 3 articles in a database).

When clicking a link to article, user should get to a page with a rendered markdown text and a single "Back" link which brings him/her back to main page.

# Chapter VI

## Exercise 02: Haters Gonna Hate

Now, when you already have a cool website, let's update it a little and protect ourselves from the villains trying to bring it down! All you need to do is implement rate limiting, so if ther are more than a hundred clients per second trying to access it, they should get a "429 Too Many Requests" response. Of course when you're getting more famous we'll raise that limit! It's just for testing for now.