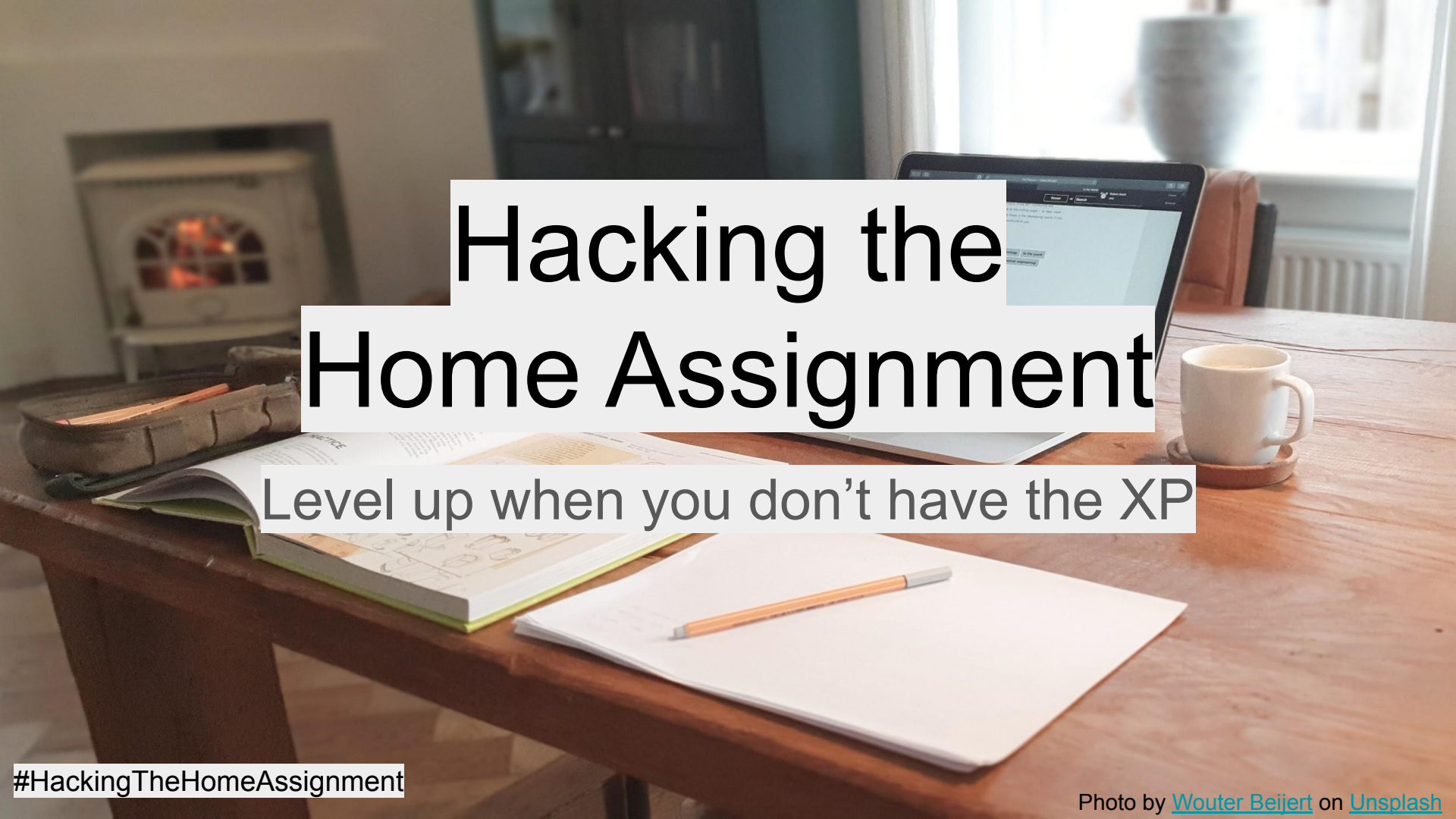


Hacking the Home Assignment

Level up when you don’t have the XP

#HackingTheHomeAssignment

Photo by Wouter Beijert on Unsplash

Nice to meet you!

I’m Eyal

Daniel McCullough Unsplash

Why?

Companies like home assignments

Candidates (should) like home assignments

Fullstack Assignment - To Do List

OVERVIEW:
write a simple, responsive, web app in client & server that handles tasks.
Client side should be in TypeScript and React.
Server side should be: TypeScript and node.js (express or restify)

Backend:
The server should have a list of API, which are written below

1. Create task
2. Get all tasks
3. Delete task
4. Edit task

Frontend:

- psd file is attached

Bonuses:
Add user interface for login and register new user (design style of this part is less important)
.

- User can be admin or regular user.
- Admin - can edit, delete and read all tasks
- Regular user - can read, edit and delete his tasks
- Pagination and search are not required

How To Submit:

1. Push your code to your git repository, make it public and don't forget database file, name the repo as firstName-lastName-dateOfStart (John-Doe-01-12-2018).

2. Deploy the compiled app to your server or Heroku/etc

Send us the git repository link and a link to the deployed app (2 links)

Lightricks: PM Candidate Home Assignment

1. **Research, Analytic thinking, Product overview:**
   Disclaimer:
   We are currently in the process of working on a similar project. It's possible that some of the ideas/concepts you suggest have already been discussed internally. Regardless, please be aware that Lightricks reserves the right to use any/all submitted materials.

Exam: Wix Home Assignment- Backend Developer

Your assignment is to build an online system for a fitness (gym) business.

It should allow customers to:

- View available classes (with the class name, description, price, duration, the maximum number of participants, the instructor name).
- View available time slots for the classes and how many places are left.
- Book a class.
- If the class is full - allow the customer to join a waiting list, the customer should be getting a notification (choose whichever way you want) should a vacancy open and be allowed to join the list.
- Get a Thank You/Confirmation page.
- Bonus: Allow customers to cancel their bookings.

Note: Class data (name, description, duration, time, max participants) should be pre-defined.

Please make sure:

When a class reaches its maximum number of participants, the next customer cannot book anymore.

You should pay **extra attention** to the backend API you define!

Technical instructions:

- The client-side needs to be created using React or Angular (If you are not familiar with both - we recommend React).
- Server-side should be Java (Jetty, Tomcat, Apache, any other web server you like) or Node.js
- Data should be saved using MySQL or Mongo.

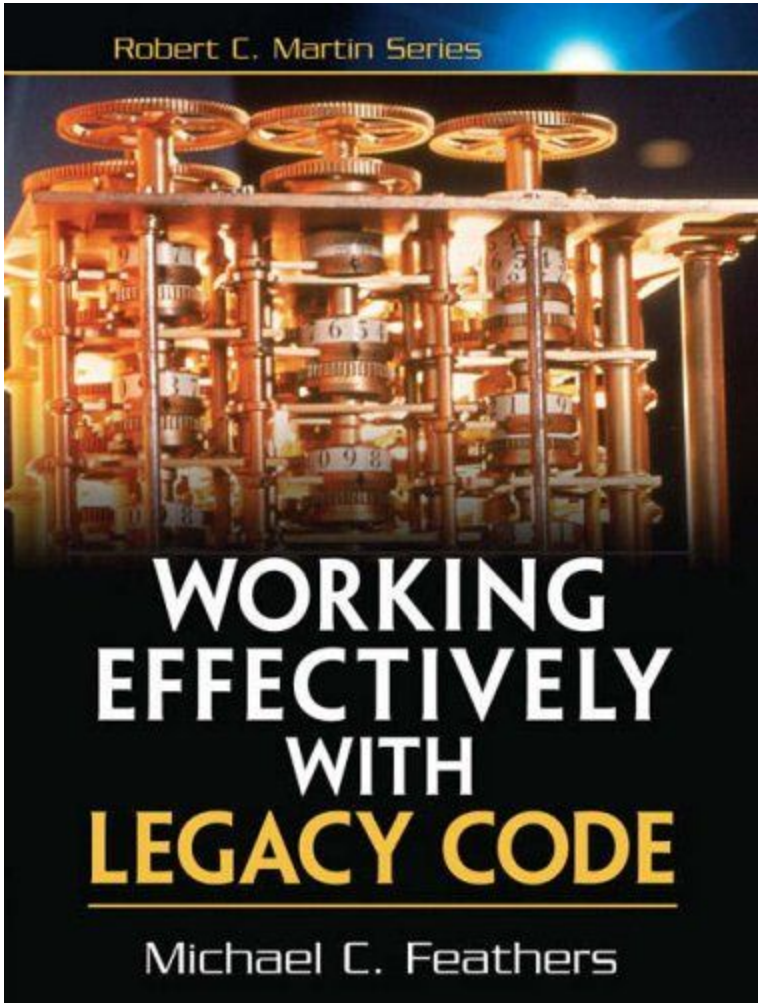
Packaging & submission

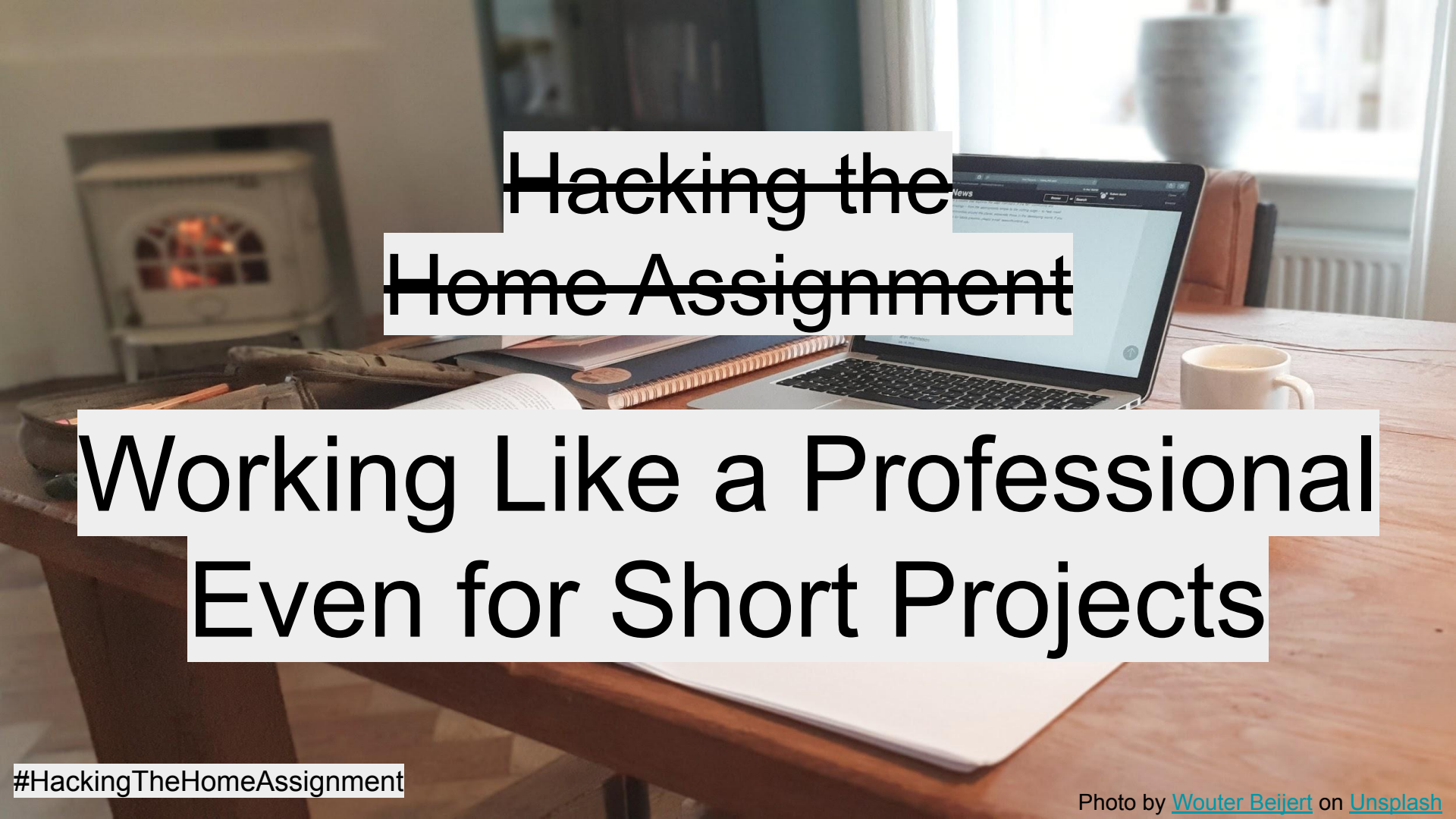
- Server: zip your .jar/.war file with text containing the instructions of how to run your server as server.zip. Include all of your source code in a separate folder in that zip file.
- Client: all sources must be un-minified. Packaged in client.zip.
- DB credentials: Should be DBUser: root Pass: 1234

nd it to alexn@wix.com, lironb@wix.com with ubject line: WIX JUNIOR POSITION TEST - R NAME>

העברה תשובה לכולם תשובה

The title is a lie


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers



~~Hacking the Home Assignment~~

Working Like a Professional Even for Short Projects

#HackingTheHomeAssignment

Our plan for today

1. Analyzing an assignment, asking questions
2. Project Structure
3. Version Control

   [Exercise #1]

4. Code Structure & Design

   [Exercise #2]

5. Testing

   [Exercise #3]

Our plan for today (cont.)

6. Environment Reproducibility

   [Exercise #4]

7. Documentation

   [Exercise #5]

8. Packaging & Distribution

   [Exercise #6]

9. Wrap up & Checklist

Our Home Assignment

Welcome to Yulco!

- Your goal is to write IFS - Insecure File System. It's a REST API for performing operations on your file system.
- GET /fs/<path> : Return a JSON response structured as:

```
{"success": true,
 "fs": {"filename": "<path from request>",
        "dirs": ["dir1", ...],
        "files": ["a", ...]}}
```

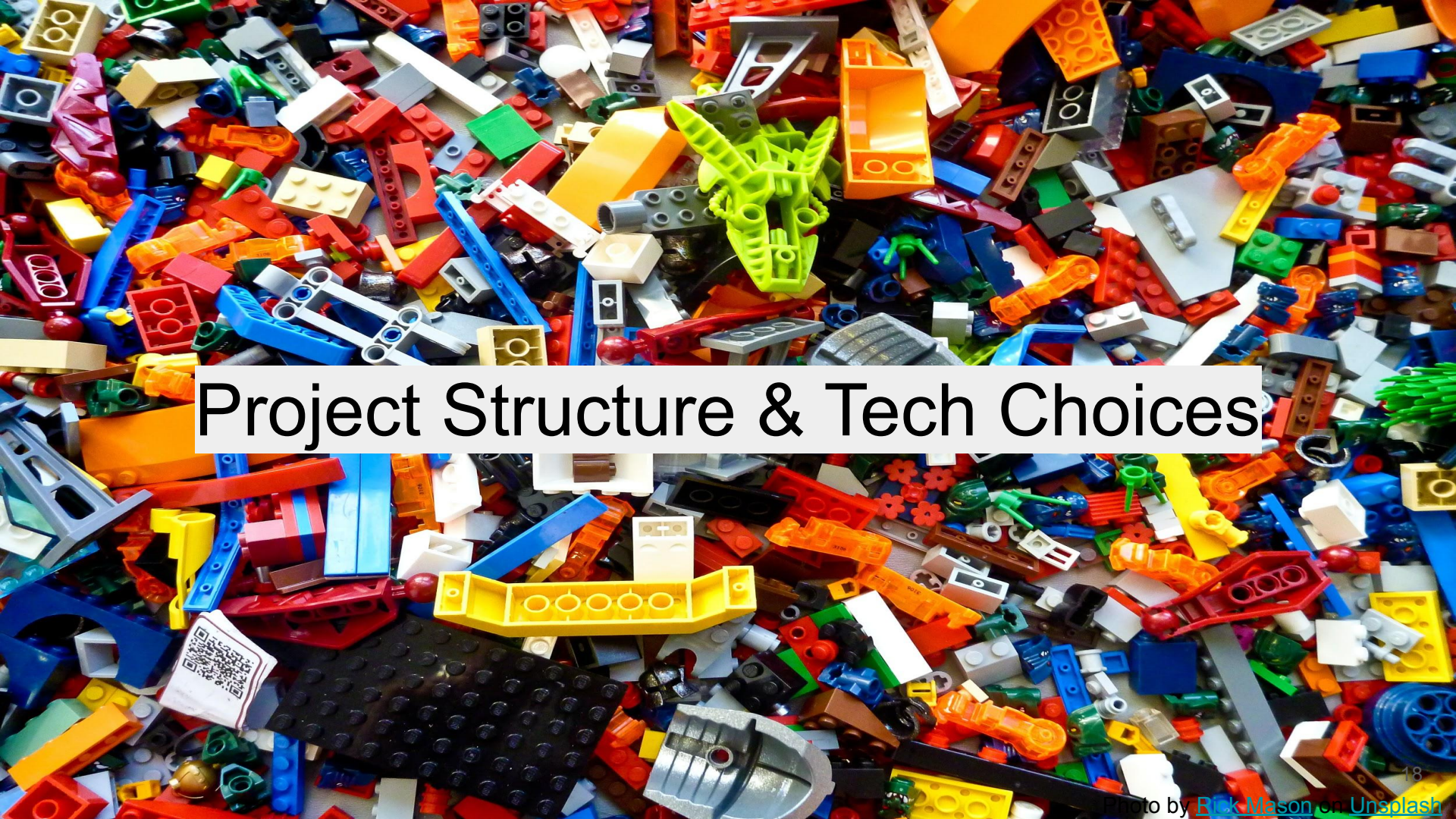
- DELETE /fs/<path> : Delete the file or directory at this path.
- PUT /fs/<path> with body `{"name": "<new path>"}` : Move file to new path

https://github.com/yulkes/PublicSpeaking/blob/master/HackingTheHomeAssignment/SampleHomeAssignment.md

Photo by Andrew Neel on Unsplash

What do we need to do?

- Functional Requirements
- Finding the catch
  - Performance, logic, security, amount of work, concurrency
- Edge Cases
- Verification: How do you know it works?
- What's the MVP
  - Complete one use case end-to-end (Get filelist)
  - One subsystem (Filesystem commands)
- Decide on a timeframe
- Read the docs
- Acknowledge & Ask



Project Structure & Tech Choices

Project Structure & Tech Choices

- Tech Stack Choices: Latest version of your best-known language
- Template:
  - Cookiecutter
  - Create-react-app
  - https://start.spring.io/
  - Google: <language> <web framework> project template
- Minimum files at directory root
  - organize under `app/src, tests, static, config`
- Strict requirements: filenames, classes, functions
- Optionals and nice-to-haves

Photo by Yancy Min on Unsplash

Using Version Control

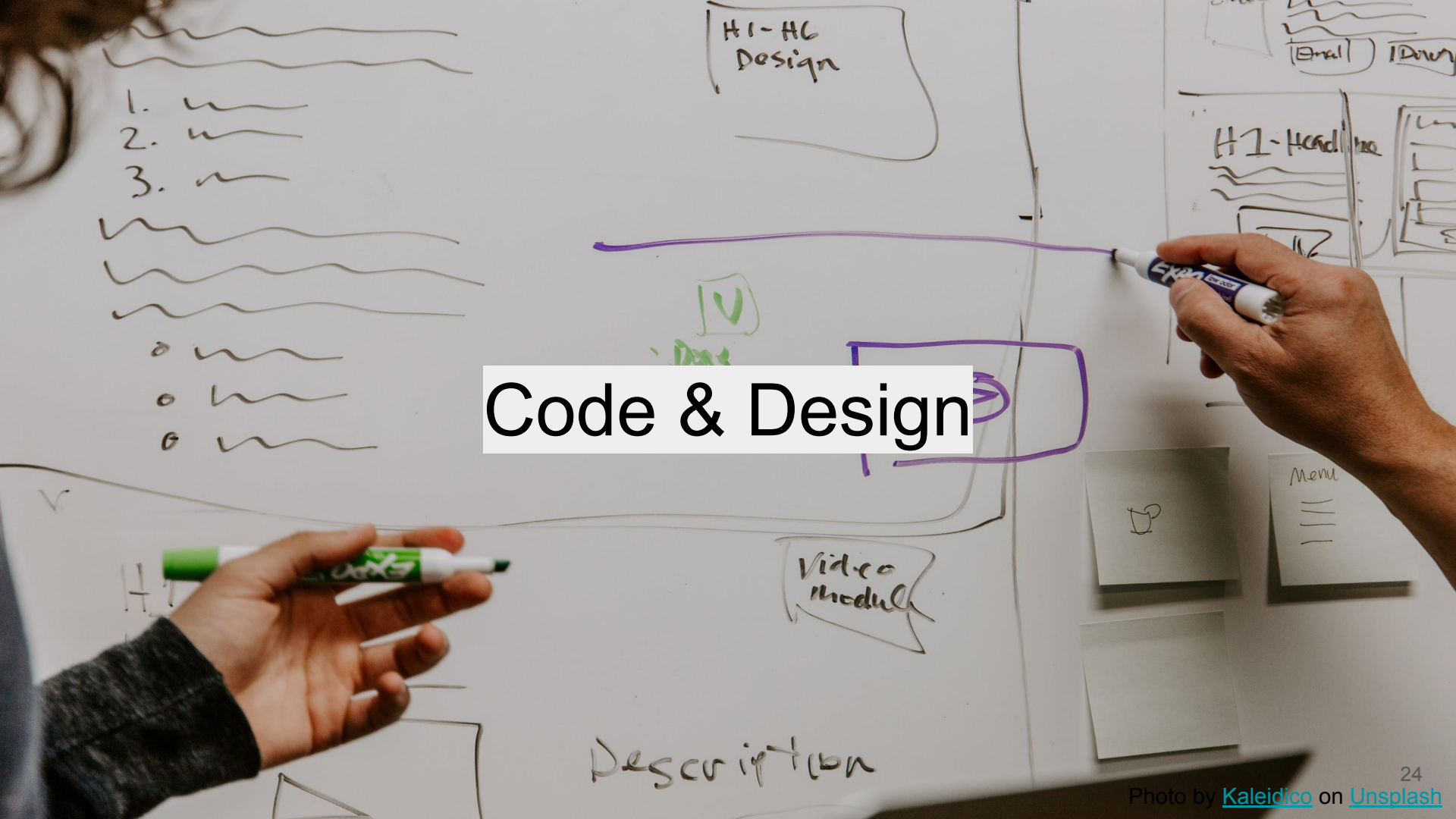
- Git is Global (GitHub is free)
- Complete Code
- Show Progress
- .gitignore
  - https://github.com/github/gitignore
  - Add your IDE's files to your gitignore (i.e.: `.idea` directory)

Exercise #1

Photo by Anastasiia Kamil on Unsplash

Exercise #1 - 20 minutes

- Create GitHub repo, clone it locally
- Start a project from a template
- Assess what you have:
  - Explore
  - understand what you don’t like about some - is it too opinionated? Too complex? Don’t understand how the code works?
- Clean up a bit
- First commit - from here on it’s your work
- Start layout of the solution.



Code & Design

Code Structure & Design

- Code Architecture - Model, View, Controller (MVC)
- Layer Separation
  - REST Layer
  - Class for every outside interaction: Filesystem, DB, 3rd party API
  - Your own model classes between the REST layer and service wrapper
- Data Types: Beyond strings and ints
- Multiple Files
- Code Style: Linter, autoformatters
  - Beautify
  - black
- Limits and Constraints

Exercise #2

Exercise #2 - 30 minutes

Create empty files dedicated to help you write the code:

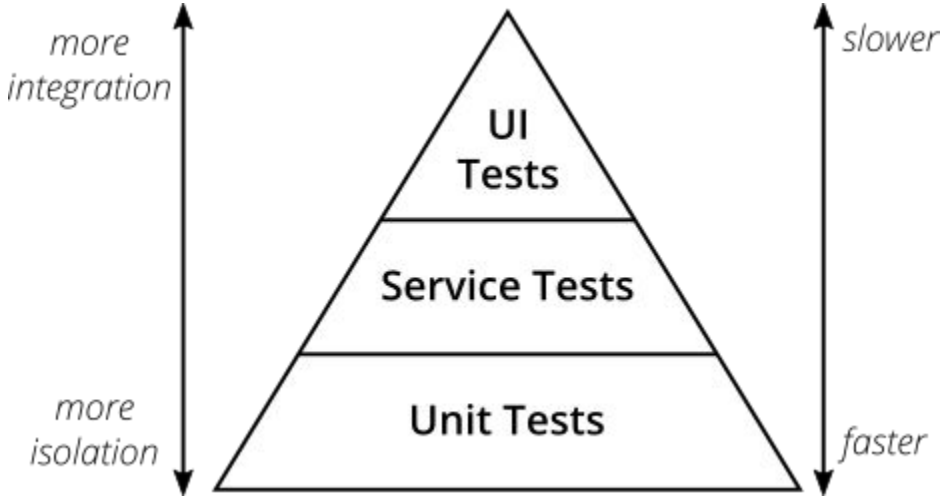
- One for the REST interfaces you'll have.
  - You can create the interfaces to do nothing, just be exposed at the right path.
- One for models - Logic bound objects, that don't access anywhere else.
  - They shouldn't import anything.
  - They'll be used for communication between the views (REST) and the service that actually runs the File System operations
- One for the file system operations wrapper.
  - Construct the interface with your model objects, fewer string/int parameters.
  - Makes it easier to see relations, constraints, validations, and enforce the correctness of the interface even without integration tests.

Start Implementing!

Testing

The Test Pyramid

- Multiple Unit Tests
  - Catch errors
  - Different parameters, failures
  - High value for you
- Few Integration Tests
  - Combinations of components
  - Subsystem interaction
- One-two E2E Tests
  - Sanity
  - Internal or external
  - High value for reviewers



https://martinfowler.com/articles/practical-test-pyramid.html

Exercise #3

Exercise #3 - 20 minutes

- If you haven't had tests already, choose a testing framework, and add unit tests.
- If you've never used a testing framework before, write a script that uses your REST service:
  - Fetch data from `localhost:5000/fs/...`
  - Make a change
  - Fetch data again to see that the change is in affect.

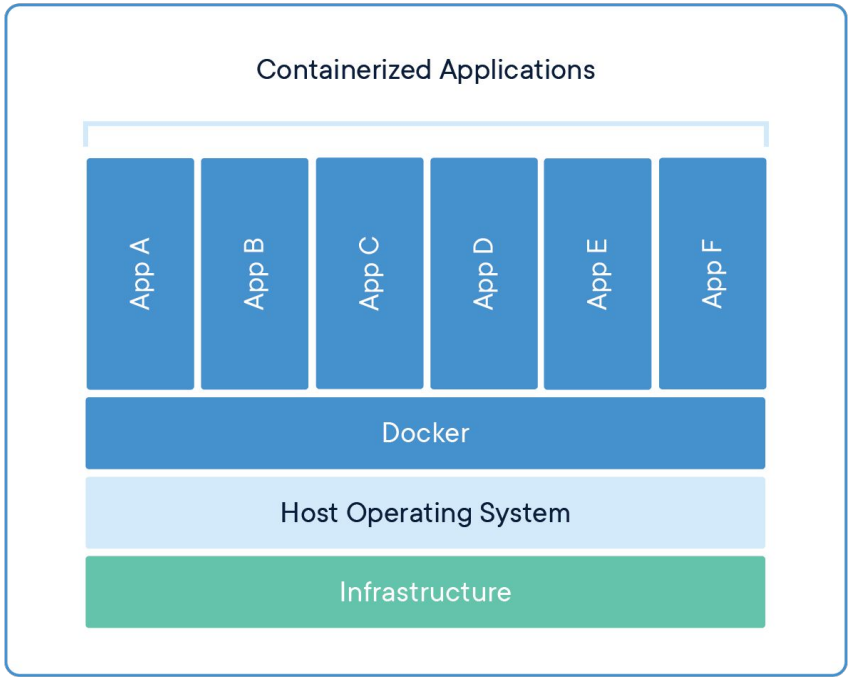
To run this e2e test, start the server, wait a couple of seconds, and run the test script. Know the commands to do all these - you'll need them later on.

Unsure? Write CURL calls that show how to use the service

American Public Power Association on Unsplash

Environment Reproducibility

- Docker is the leading tool for sharing environments
- Your code runs in isolated containers on the reviewer's machine



Exercise #4

Exercise #4 - 20 minutes

- Get your project to run in Docker:
  - Find an existing Dockerfile for your language and Web Framework
  - Modify it to your needs

- Add commands that'd add a directory structure into your docker image
- Run a test script against your running docker container.
  You can always kill it and restart it exactly to the same state it was before.

Documentation

Photo by Mark Rasmuson on Unsplash

Documentation

- It’s never well-defined what’s expected here
- Current Functionality
- Code structure
- Assumptions and limitations
- Important Commands
- What it doesn’t do at the moment:
  - Which of the functional requirements you’ve missed
  - Which non-functional requirements you think would’ve made sense, but aren’t here.

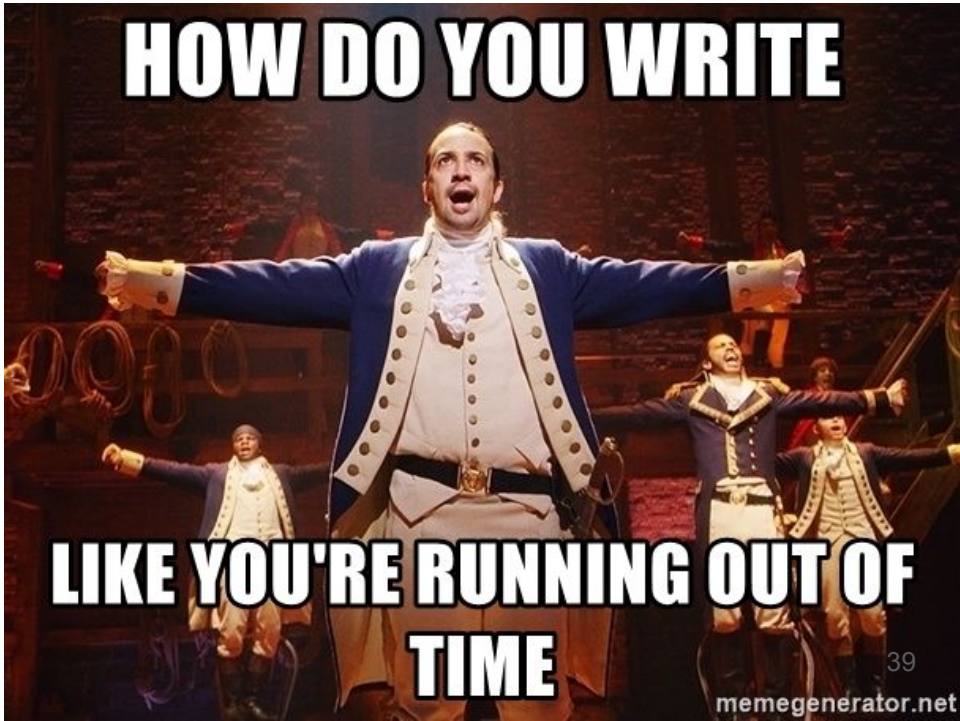
For each requirement that you’ve missed, in case you’re called back for a follow up, have an explanation ready in mind on how you’d implement it.

Exercise #5

Exercise #5 - 10 minutes

Document Furiously:

- What it does right now
- Code structure
- Important components
- Technical assumptions and limits
- Commands:
  - How to run tests
  - How to run the service (Docker)
- What’s missing



Packaging & Distribution

Photo by Kira auf der Heide on Unsplash

Packaging & Distribution

- Unused files and code
- Remove print() and debug statements
- Makefile for commands
  - Learn on https://makefiletutorial.com/
  - For Frontend - Something like yarn
- Clean Zip Files
- Follow your instructions
- Send & ask for confirmation

Exercise #6

Exercise #6 - 20 minutes

- Create a Makefile with the commands you used the most
- Clean the project directory
- Clean the codebase itself
- Run the linters
- Run the tests
- Create a zip file
- Unzip it and follow your instructions. Does it run?

Wrap Up

The Home Assignment Checklist

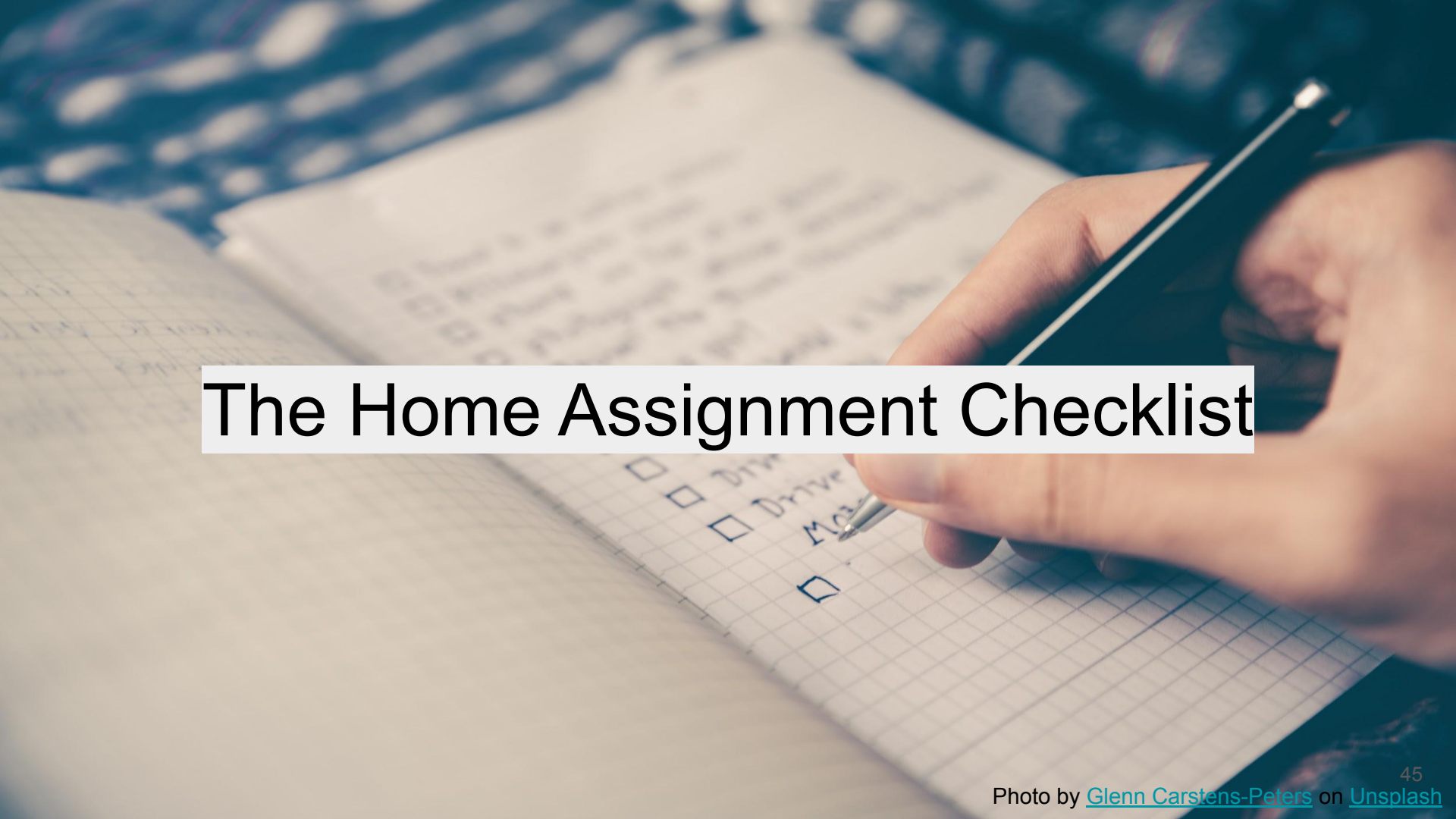


Photo by Glenn Carstens-Peters on Unsplash

- Acknowledge
- Quick go-over, questions, choices and misunderstandings
- Start from template
- Implement any required interfaces
- Commit your work early and frequently
- Write tests that show it works as required
- Write tests that protect you
- Complete all required functionality
- Run in Docker
- Quick commands
- Document: What you've done, How to use it
- Generate shareable asset
- Verify it works
- Submit & Follow up

https://bit.ly/HomeAssignmentChecklist

Good Luck!

@yulkes

yulkes@gmail.com

#HackingTheHomeAssignment

Photo by Wouter Beijert on Unsplash