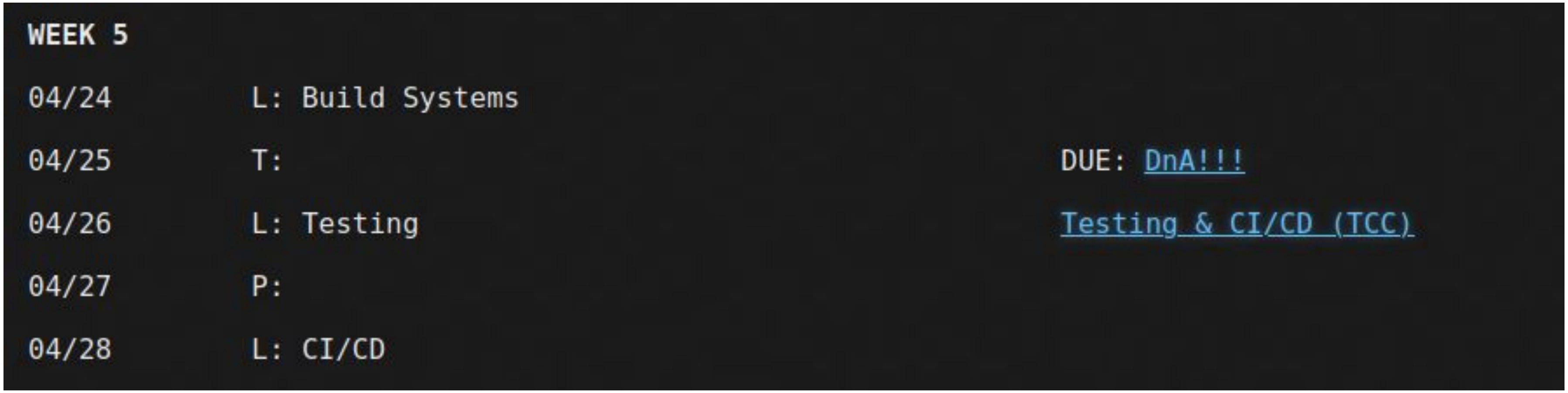
CSE 403

Software Engineering

Spring 2023

#13: Build systems

This week



WEEK 5

04/24	L: Build Systems	
04/25	T:	DUE: DnA!!!
04/26	L: Testing	Testing & CI/CD (TCC)
04/27	P:	
04/28	L: CI/CD	

What does a developer do?

- Get the source code
- Install dependencies
- Compile the code
- Run static analysis
- Generate documentation
- Run tests
- Create artifacts for customers
- Ship!

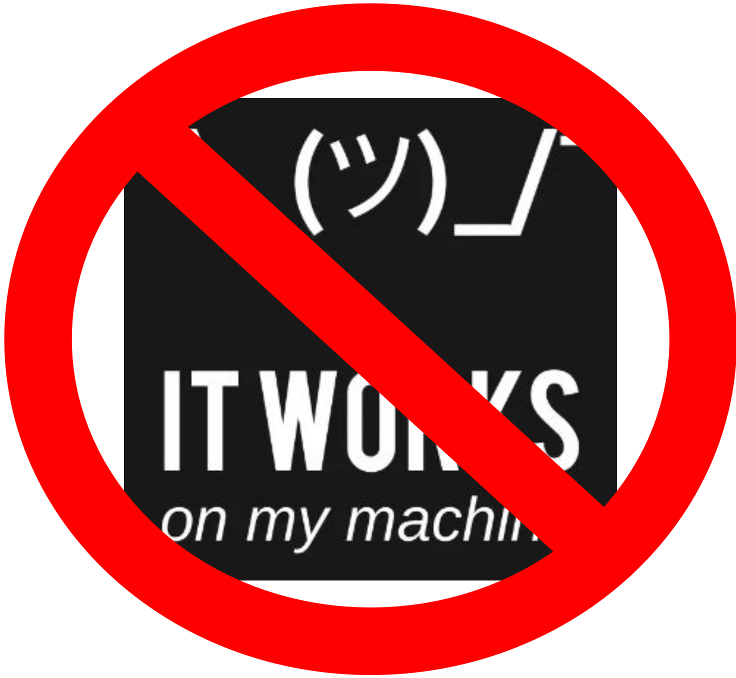
What does a developer do?

- Get the source code
- Install dependencies
- Compile the code
- Run static analysis
- Generate documentation
- Run tests
- Create artifacts for customers
- Ship!

Which of these tasks should be handled manually?

What does a developer do?

- Get the source code
- Install dependencies
- Compile the code
- Run static analysis
- Generate documentation
- Run tests
- Create artifacts for customers
- Ship!



Which of these tasks should be handled manually?

NONE!

How to automate these tasks?

- Get the source code
- Install dependencies
- Compile the code
- Run static analysis
- Generate documentation
- Run tests
- Create artifacts for customers
- Ship!

Orchestrate tasks with a build system!

Build systems: tasks

Tasks are code!

- Should be checked into version control
- Should be code-reviewed
- Should be tested

Best practices

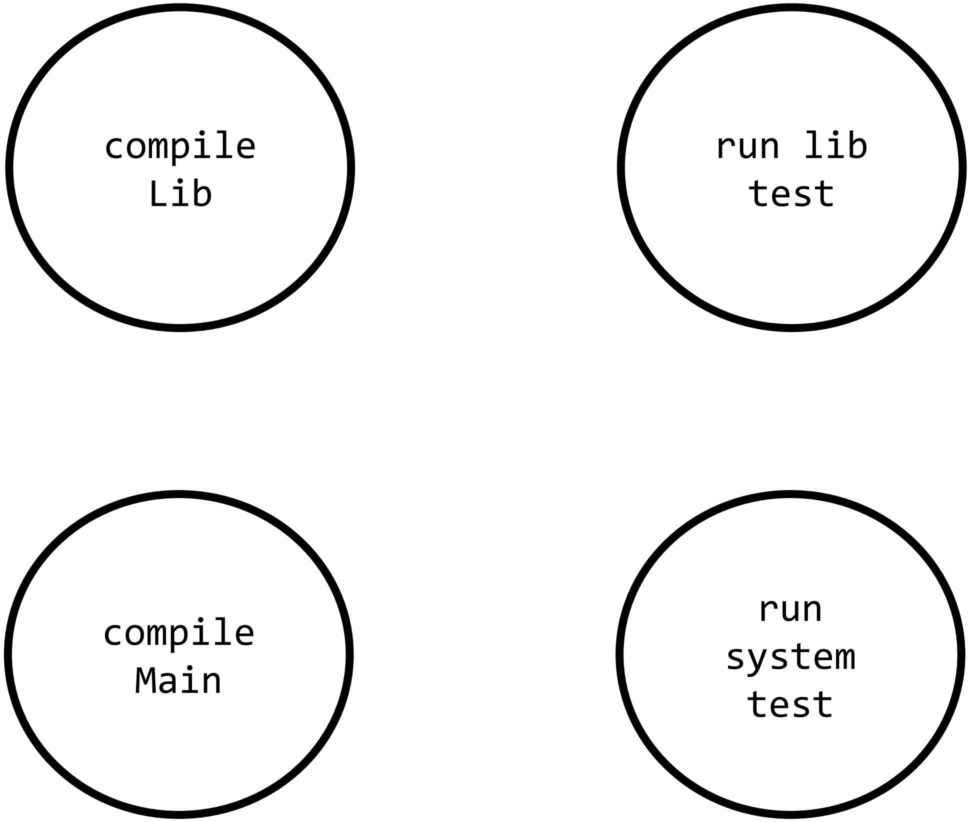
- Automate everything (one-step build)!
- Always use a build tool.
- Use CI to build and test your code on every commit.
- Don’t depend on anything that’s not in the build file (hermetic)!
- Don’t break the build!

Build systems: dependencies between tasks

Example code and corresponding tests:

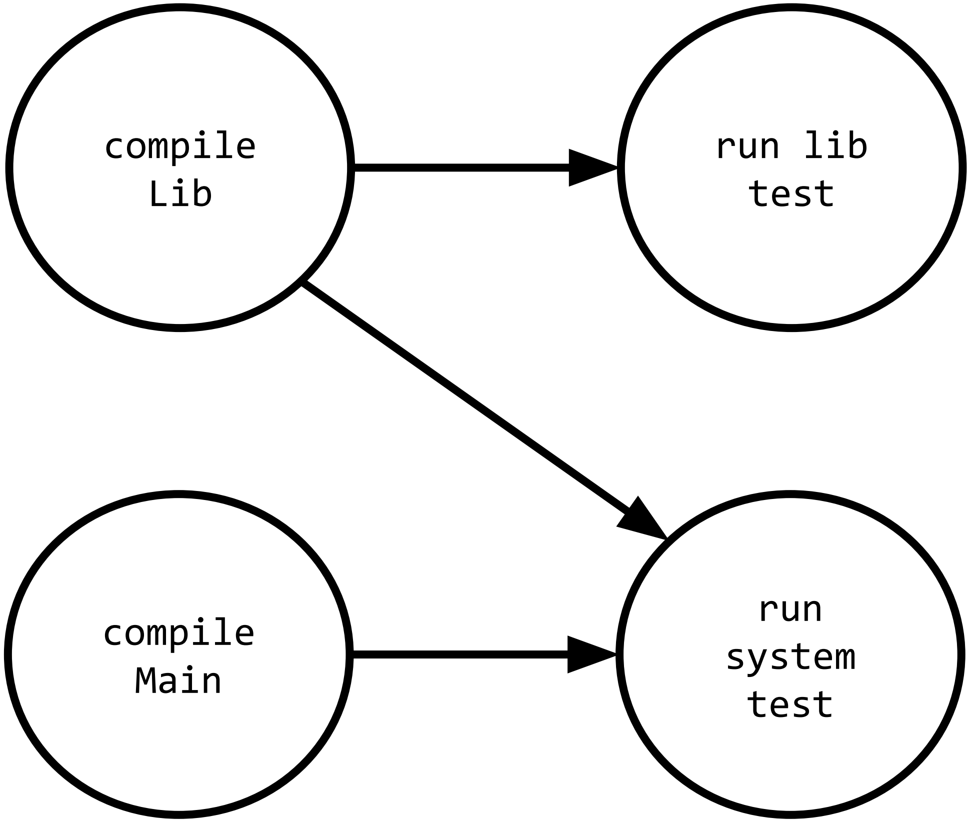
```
> ls src/

Lib.java   LibTest.java   Main.java    SystemTest.java
```

Build systems: dependencies between tasks

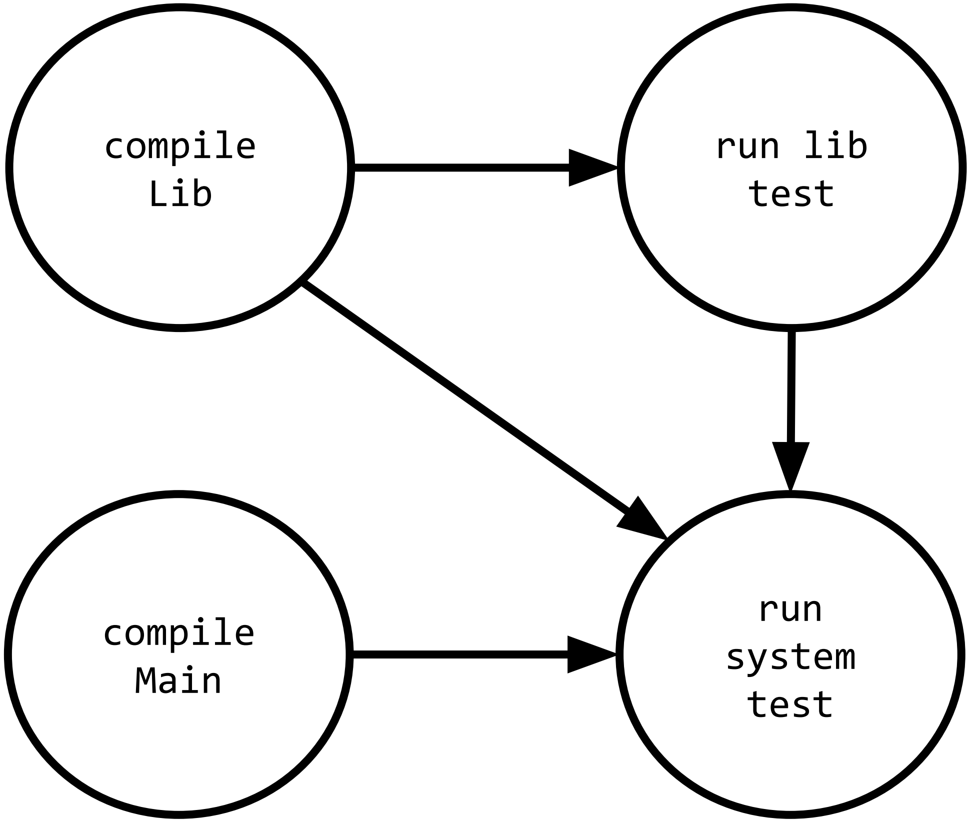


What are the dependencies between these tasks?

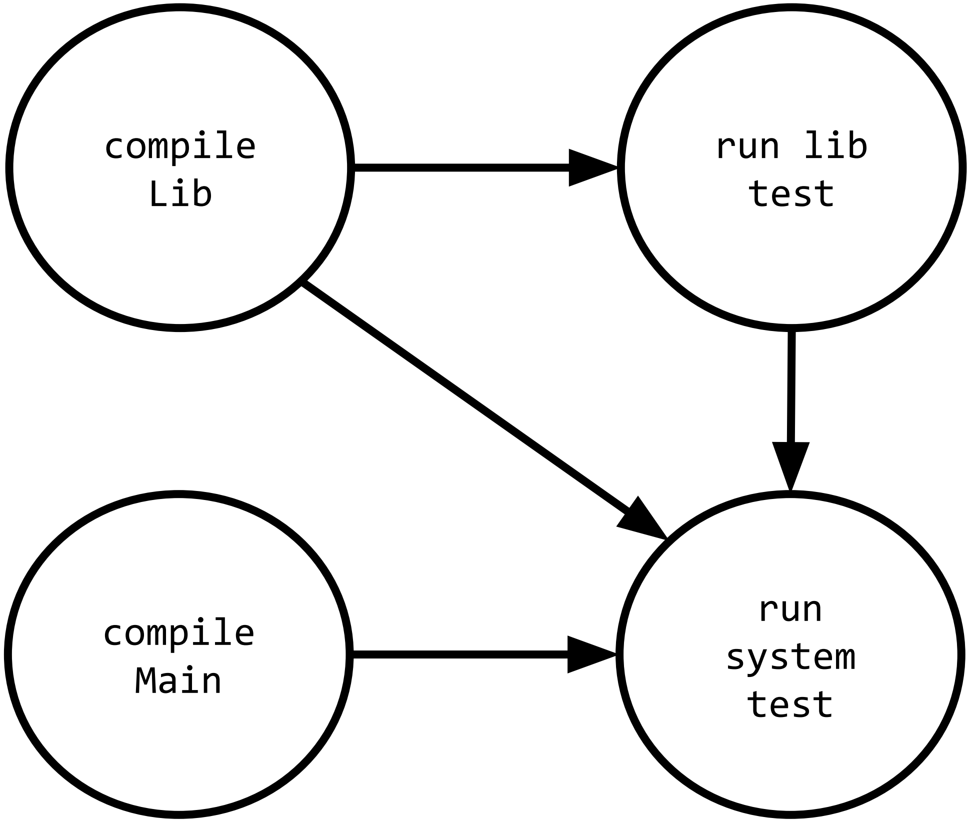
Build systems: dependencies between tasks



Build systems: dependencies between tasks



Build systems: dependencies between tasks



In what order should we run these tasks?

Build systems: determining task order

Large projects have thousands of tasks

- Dependencies between tasks form a directed acyclic graph.

Build systems: determining task/install order

Large projects have thousands of tasks

- Dependencies between tasks form a directed acyclic graph.

Also an issue for managing the code dependencies (libraries)

[Build systems: how to determine task order?]

Large projects have thousands of tasks

- Dependencies between tasks form a directed acyclic graph.

Also an issue for managing the code dependencies (libraries)

Curiosity: Topological sort

- Order nodes such that all dependencies are satisfied
- **Implemented by computing indegree** (number of incoming edges) for each node
  - No dependencies go first and open door to the others
- See extra slides for example!

Build systems: JAVA+

`gradle`

Open-source successor to ant and maven

- Groovy/Kotlin DSL (vs. xml)
- Many defaults for (maven) conventions
- Can query Maven Central for dependency resolution

`bazel`

Open-source version of Google’s internal build tool (blaze)

Build systems: Python

hatch

Implements standards from the Python standards

- Uses TOML files
- Integrates with PIP
  - Manages dependencies

poetry

Packaging and dependence manager

tox

Automate and standardize testing

Build systems: JavaScript+

npm

Standard package/task manager for Node
"Largest software registry in the world."

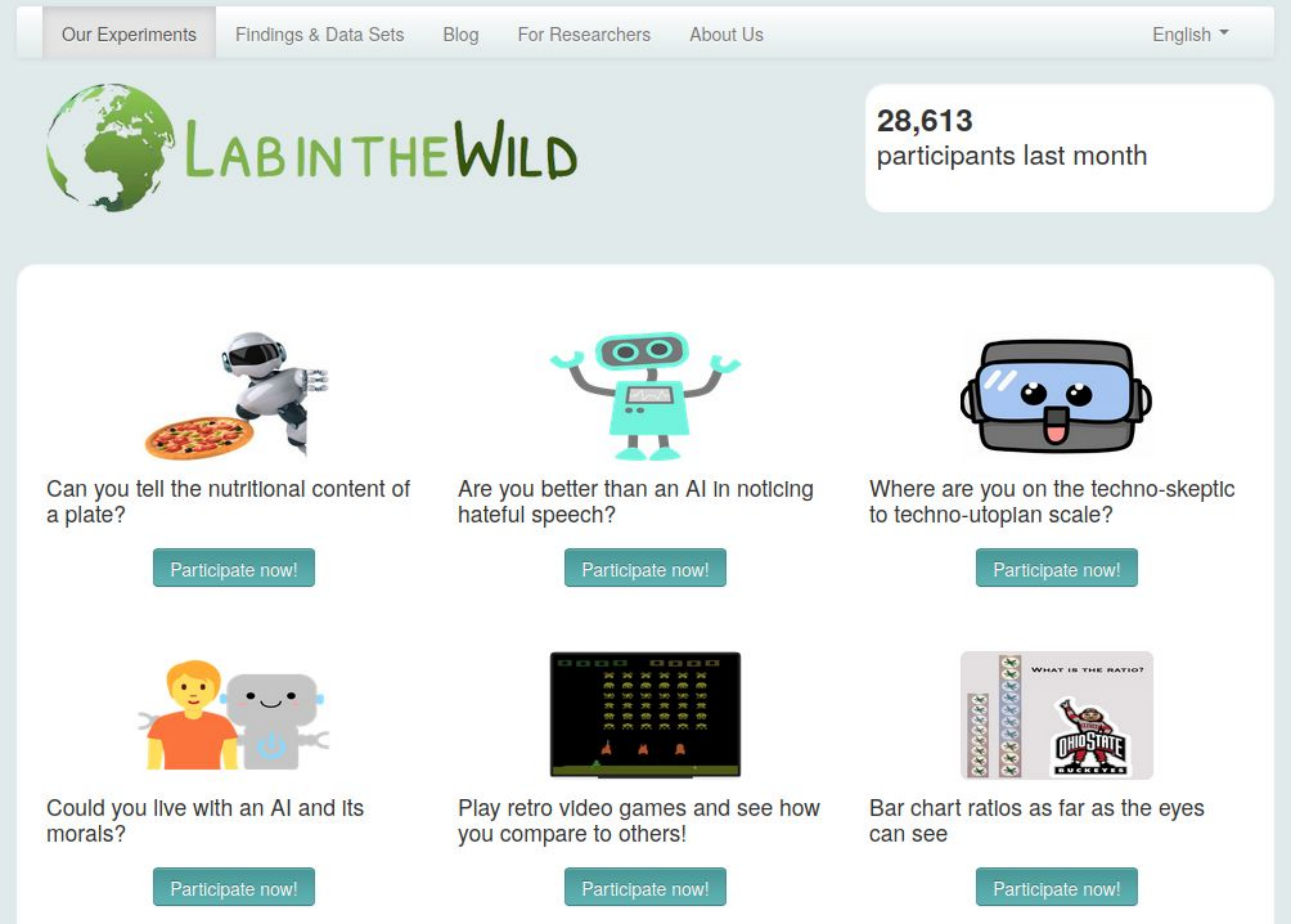
webpack

Module bundler for modern JavaScript applications

Gulp

Tries to improve dependency and packing

Demo?



Our Experiments | Findings & Data Sets | Blog | For Researchers | About Us | English

LabInTheWild

28,613
participants last month

Can you tell the nutritional content of a plate?

Participate now!

Are you better than an AI in noticing hateful speech?

Participate now!

Where are you on the techno-skeptic to techno-utopian scale?

Participate now!

Could you live with an AI and its morals?

Participate now!

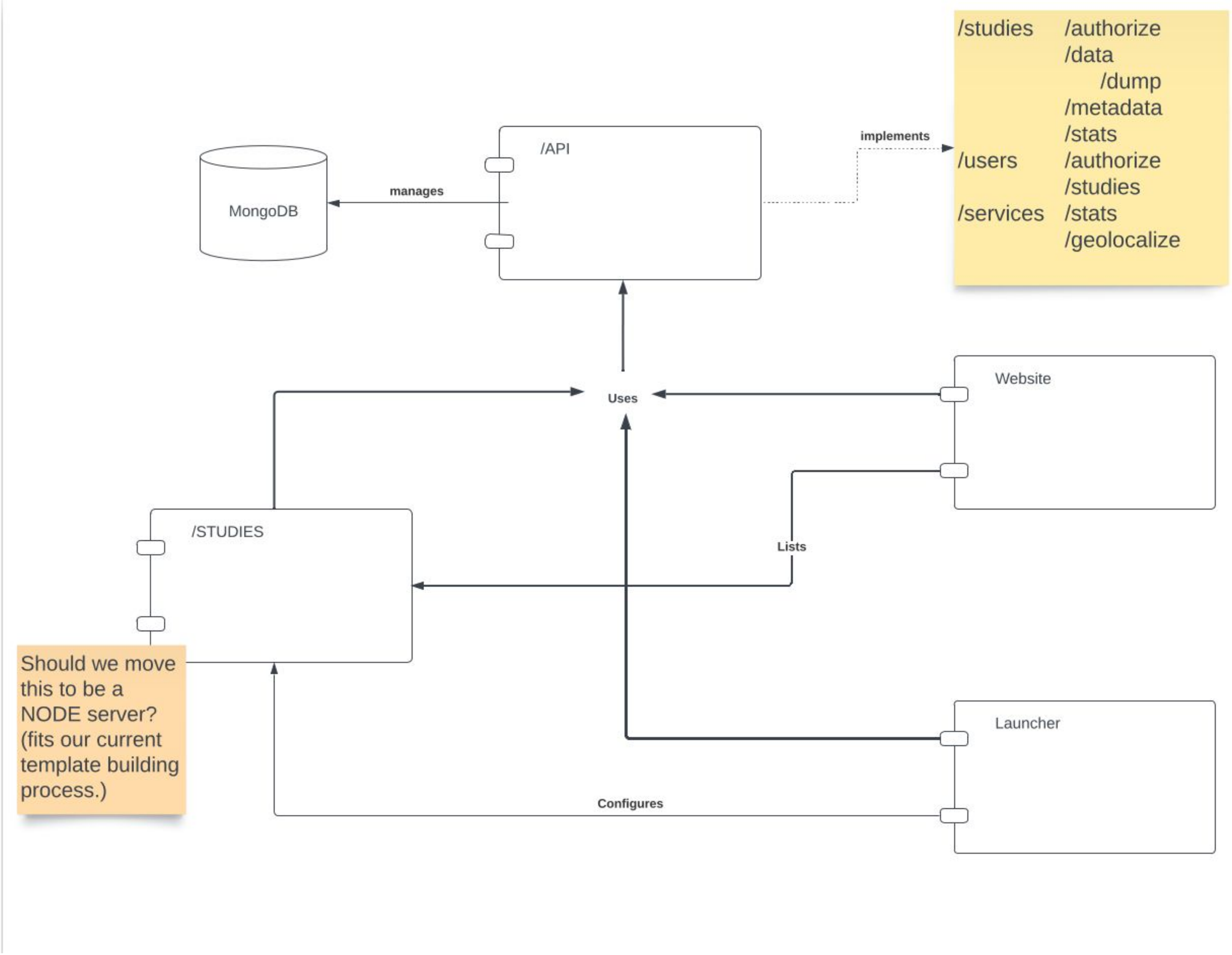
Play retro video games and see how you compare to others!

Participate now!

Bar chart ratios as far as the eyes can see

Participate now!

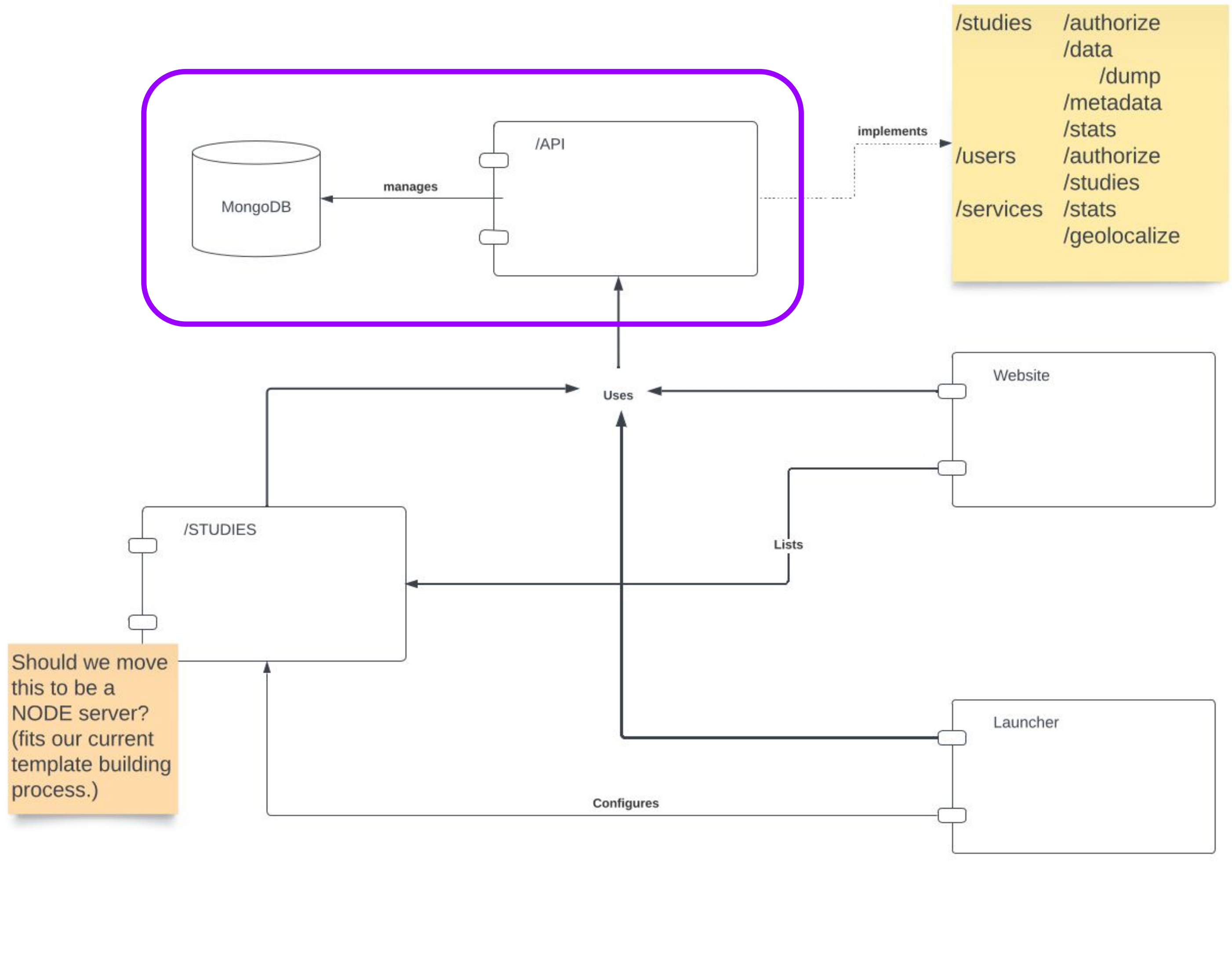
Demo: "Architecture"



/studies	/authorize /data /dump /metadata /stats
/users	/authorize /studies
/services	/stats /geolocalize

Should we move this to be a NODE server? (fits our current template building process.)

Demo: LITW API



- /studies
  - /authorize
  - /data
    - /dump
  - /metadata
  - /stats
- /users
  - /authorize
  - /studies
- /services
  - /stats
  - /geolocalize

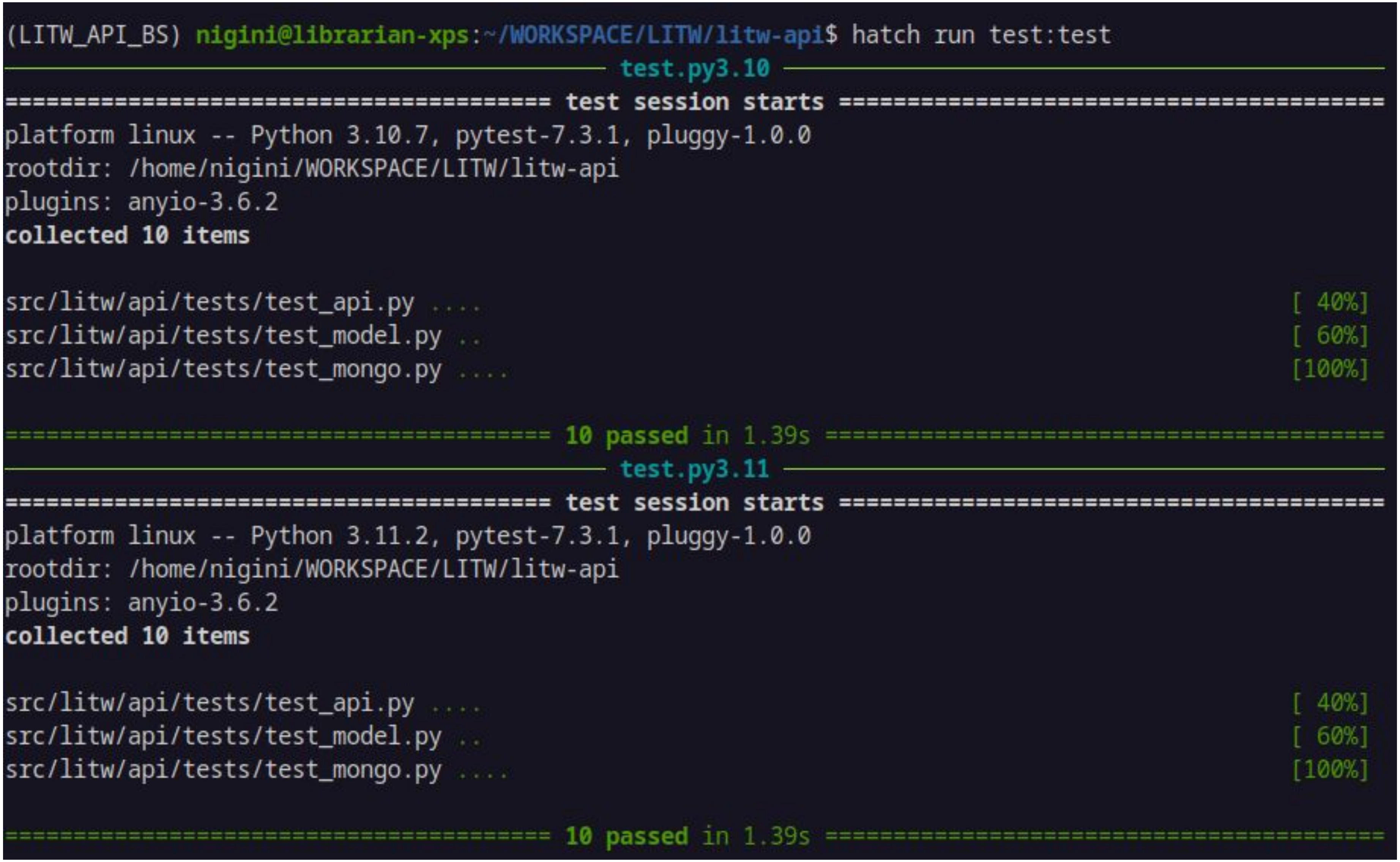
Should we move this to be a NODE server? (fits our current template building process.)

Demo: LITW-API

```
pyproject.toml  (New Python standard)
start_server.py
~~requirements.txt~~
Dockerfile
README.md
src/
   litw/
      api/
         data/
         tests/
         api.py  (A FastAPI App)
```

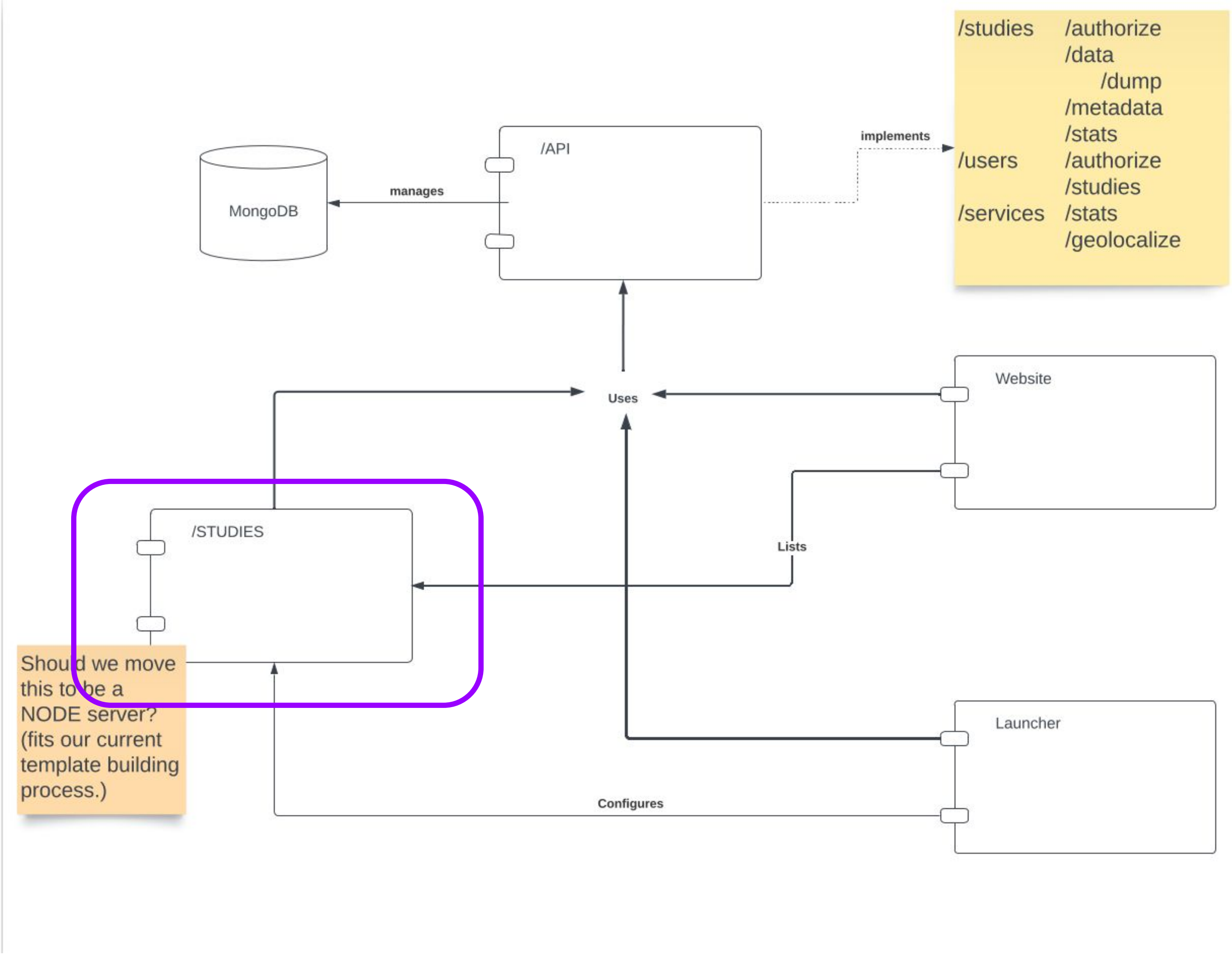
Demo: pyproject.toml

```
[project]
…

[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"


dependencies = [
  "fastapi[all]",
  "pymongo",
  "python-jose[cryptography]"
]

[tool.hatch.version]
path = "src/litw/api/__about__.py"

[tool.hatch.envs.test]
dependencies = [
  "pytest"
]

[tool.hatch.envs.test.scripts]
test = "pytest {args:src/litw/api/tests}"

[[tool.hatch.envs.test.matrix]]
python = ["3.9", "3.10", "3.11"]
```



```
(LITW_API_BS) nigini@librarian-xps:~/WORKSPACE/LITW/litw-api$ hatch run test:test
──────────────────────────────────────────── test.py3.10 ────────────────────────────────────────────
======================================== test session starts ========================================
platform linux -- Python 3.10.7, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigini/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2
collected 10 items

src/litw/api/tests/test_api.py ....                                                           [ 40%]
src/litw/api/tests/test_model.py ..                                                           [ 60%]
src/litw/api/tests/test_mongo.py ....                                                         [100%]

======================================== 10 passed in 1.39s =========================================
──────────────────────────────────────────── test.py3.11 ────────────────────────────────────────────
======================================== test session starts ========================================
platform linux -- Python 3.11.2, pytest-7.3.1, pluggy-1.0.0
rootdir: /home/nigini/WORKSPACE/LITW/litw-api
plugins: anyio-3.6.2
collected 10 items

src/litw/api/tests/test_api.py ....                                                           [ 40%]
src/litw/api/tests/test_model.py ..                                                           [ 60%]
src/litw/api/tests/test_mongo.py ....                                                         [100%]

======================================== 10 passed in 1.39s =========================================
```

Demo: LITW Template



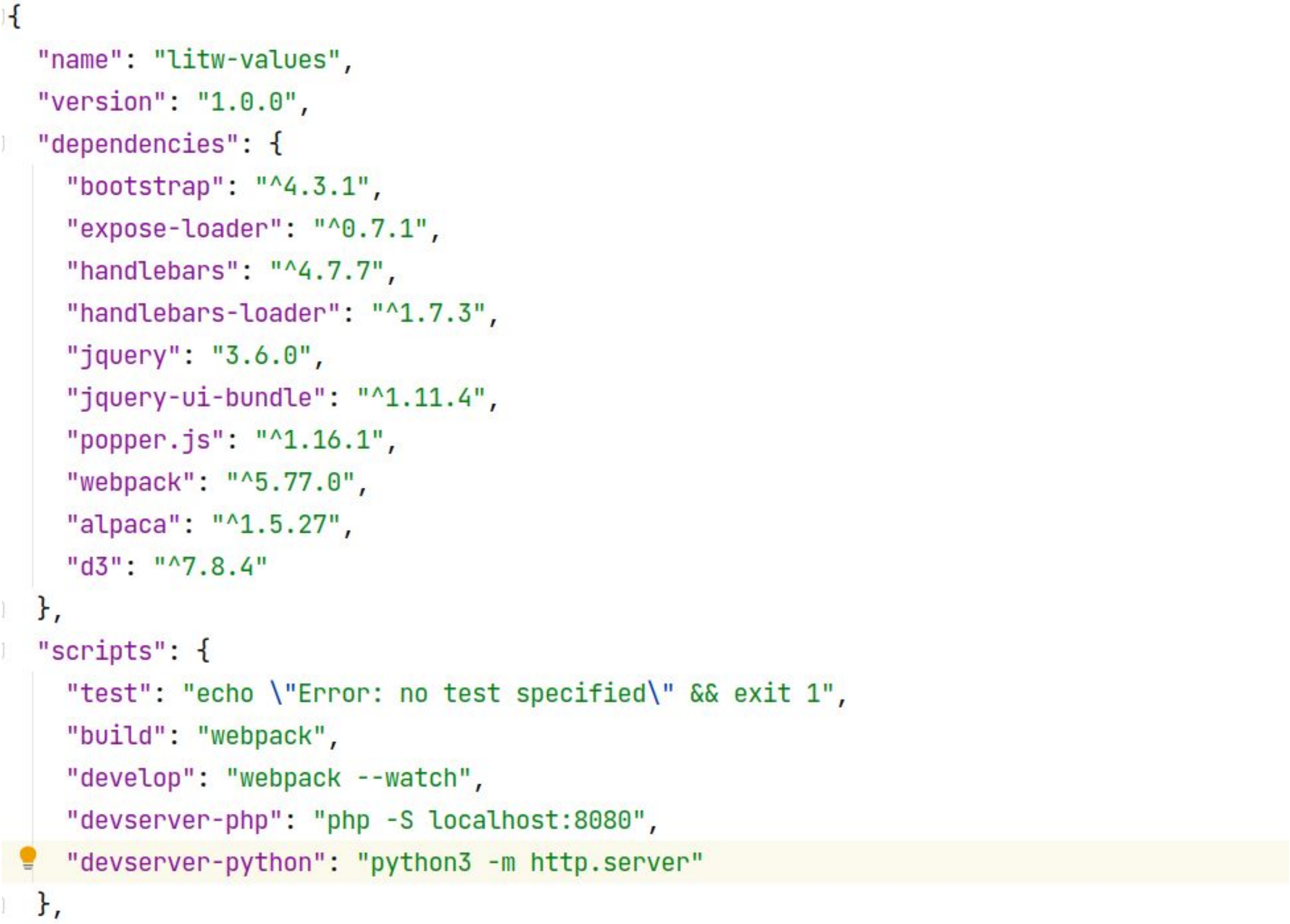
- /studies
  - /authorize
  - /data
    - /dump
  - /metadata
  - /stats
- /users
  - /authorize
  - /studies
- /services
  - /stats
  - /geolocalize

Should we move this to be a NODE server? (fits our current template building process.)

Demo: LITW-Template

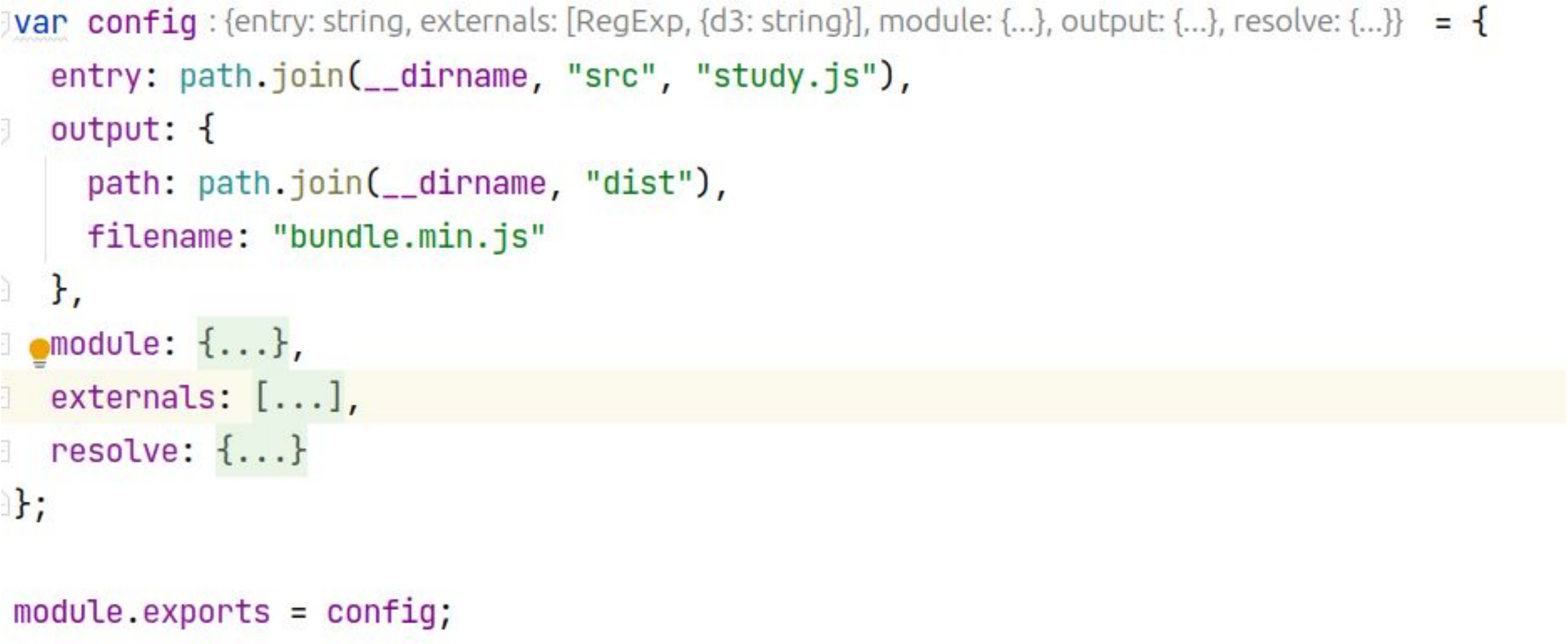
```
docs
template
   /css + /img + /js + …
   /src
      study.js
   /templates
   index.html
   package.json           (NPM config file)
   webpack.config.js      (WebPack config file)
```

Demo: LITW-Template (NPM)

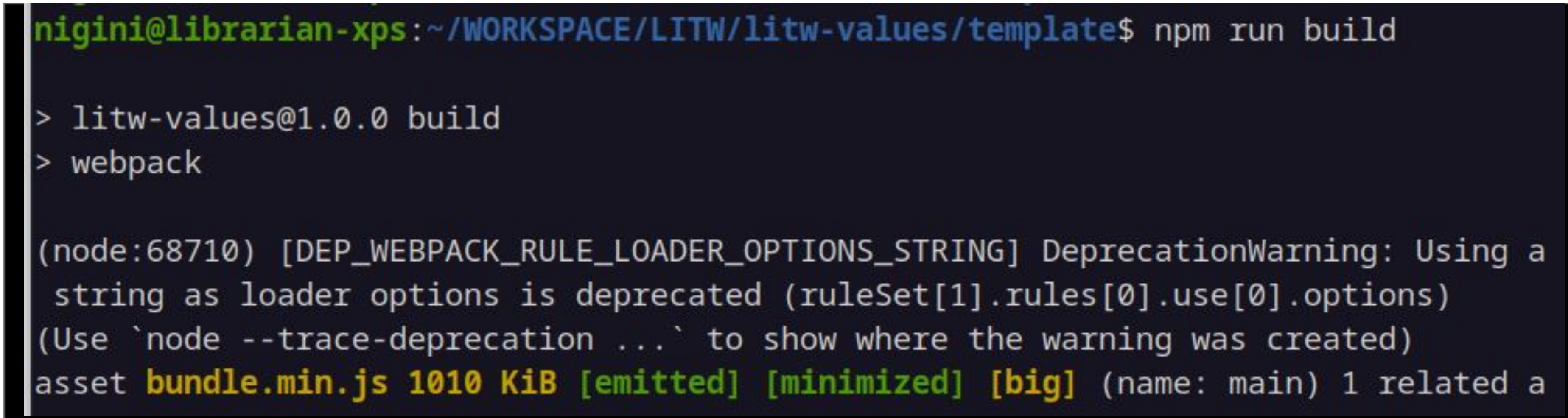


```
{
  "name": "litw-values",
  "version": "1.0.0",
  "dependencies": {
    "bootstrap": "^4.3.1",
    "expose-loader": "^0.7.1",
    "handlebars": "^4.7.7",
    "handlebars-loader": "^1.7.3",
    "jquery": "3.6.0",
    "jquery-ui-bundle": "^1.11.4",
    "popper.js": "^1.16.1",
    "webpack": "^5.77.0",
    "alpaca": "^1.5.27",
    "d3": "^7.8.4"
  },
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "build": "webpack",
    "develop": "webpack --watch",
    "devserver-php": "php -S localhost:8080",
    "devserver-python": "python3 -m http.server"
  },
```

Demo: LITW-Template (WebPack)

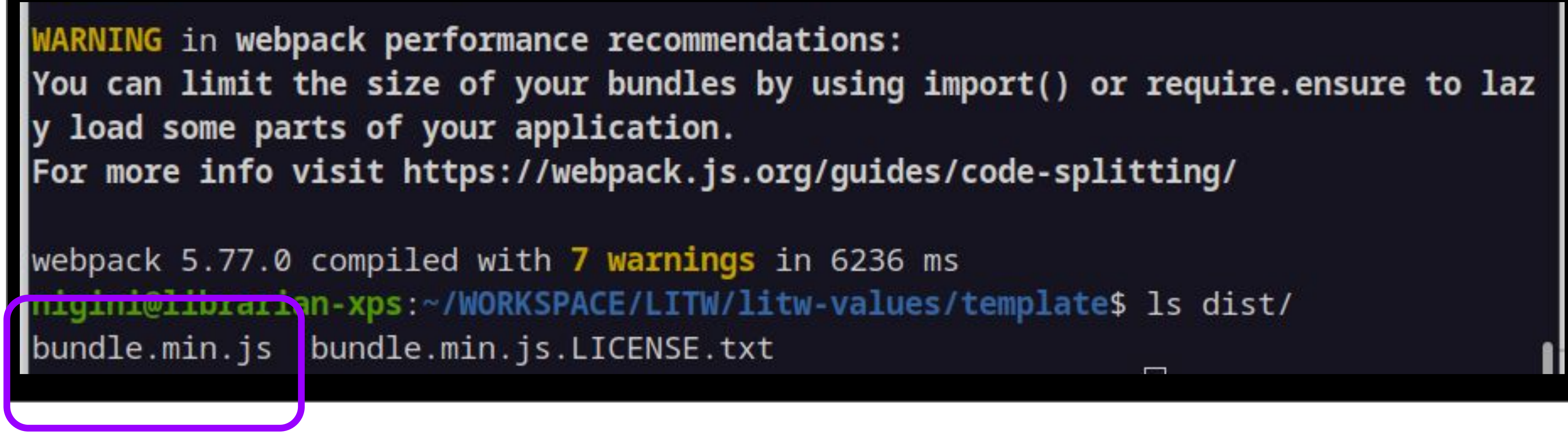


```
var config : {entry: string, externals: [RegExp, {d3: string}], module: {…}, output: {…}, resolve: {…}}  = {
  entry: path.join(__dirname, "src", "study.js"),
  output: {
    path: path.join(__dirname, "dist"),
    filename: "bundle.min.js"
  },
  module: {...},
  externals: [...],
  resolve: {...}
};

module.exports = config;
```



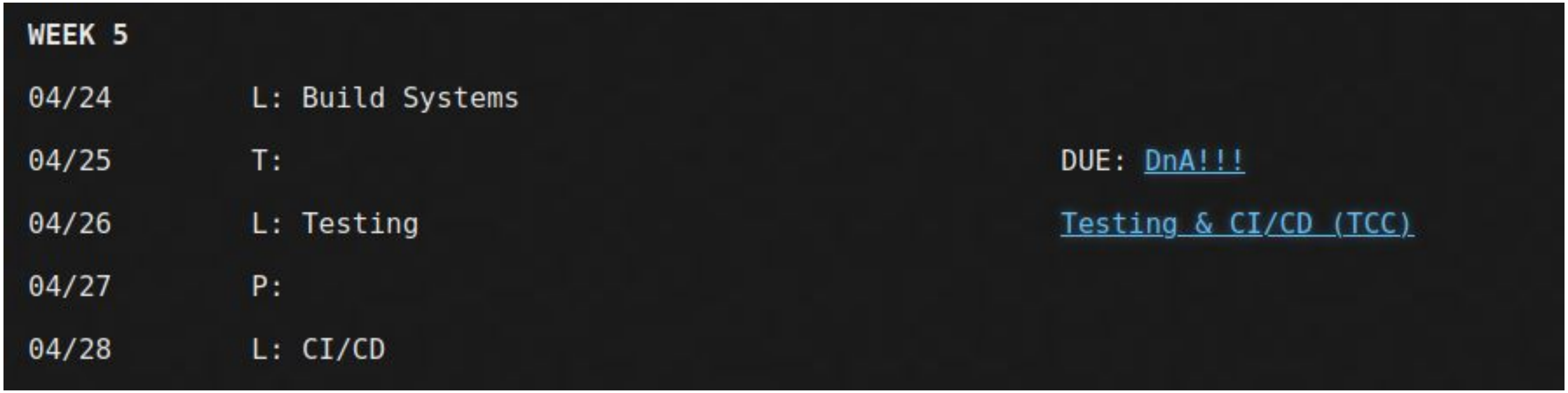
```
nigini@librarian-xps:~/WORKSPACE/LITW/litw-values/template$ npm run build

> litw-values@1.0.0 build
> webpack

(node:68710) [DEP_WEBPACK_RULE_LOADER_OPTIONS_STRING] DeprecationWarning: Using a
 string as loader options is deprecated (ruleSet[1].rules[0].use[0].options)
(Use `node --trace-deprecation ...` to show where the warning was created)
asset bundle.min.js 1010 KiB [emitted] [minimized] [big] (name: main) 1 related a
```

Many WARNINGS after!!!



```
WARNING in webpack performance recommendations:
You can limit the size of your bundles by using import() or require.ensure to laz
y load some parts of your application.
For more info visit https://webpack.js.org/guides/code-splitting/

webpack 5.77.0 compiled with 7 warnings in 6236 ms
nigini@librarian-xps:~/WORKSPACE/LITW/litw-values/template$ ls dist/
bundle.min.js  bundle.min.js.LICENSE.txt
```

This week



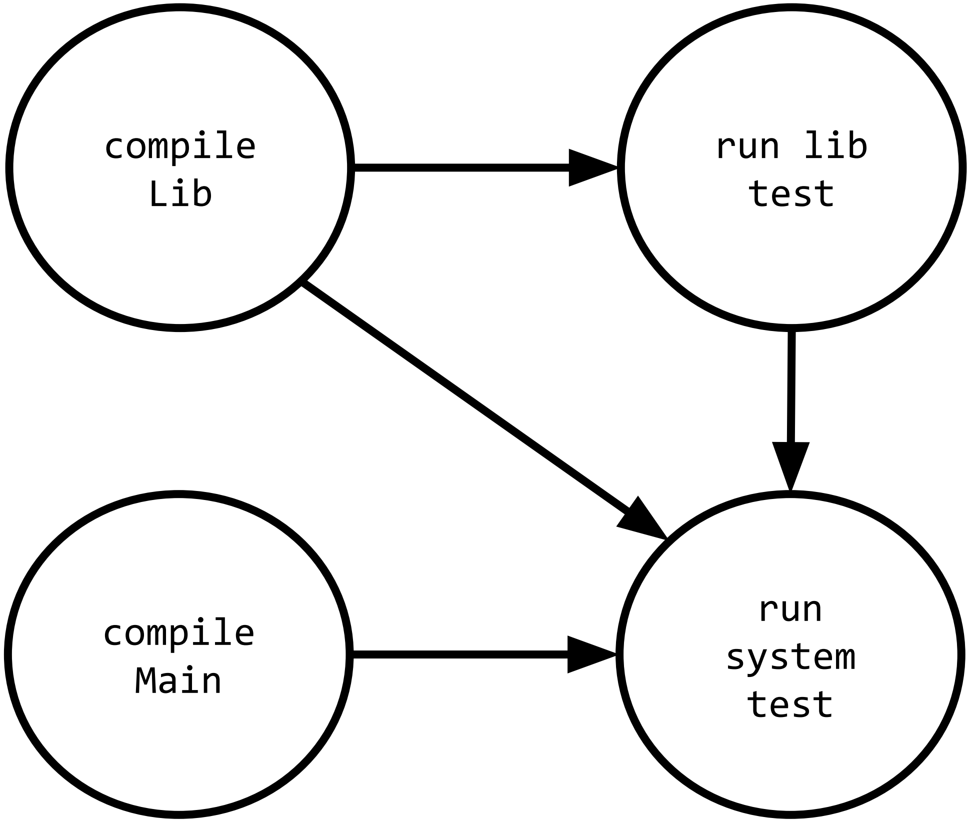
WEEK 5

04/24	L: Build Systems	
04/25	T:	DUE: DnA!!!
04/26	L: Testing	Testing & CI/CD (TCC)
04/27	P:	
04/28	L: CI/CD	

Questions, please!

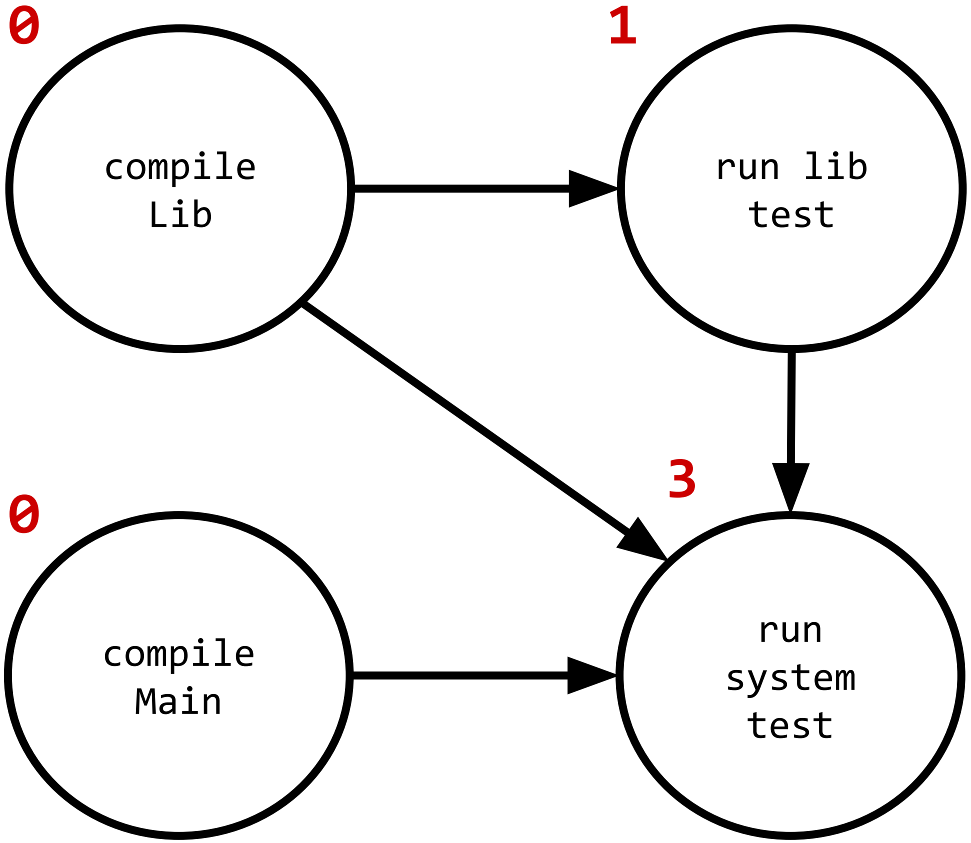
EXTRA MATERIAL

Build systems: topological sort

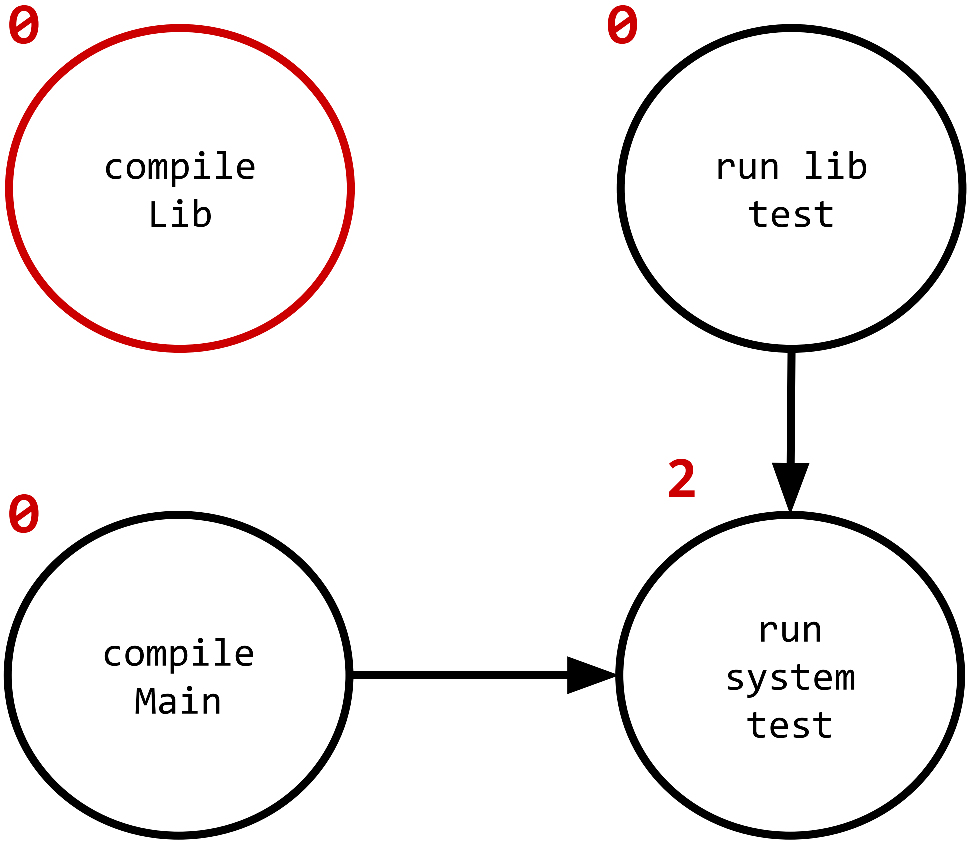


What’s the indegree of each node?

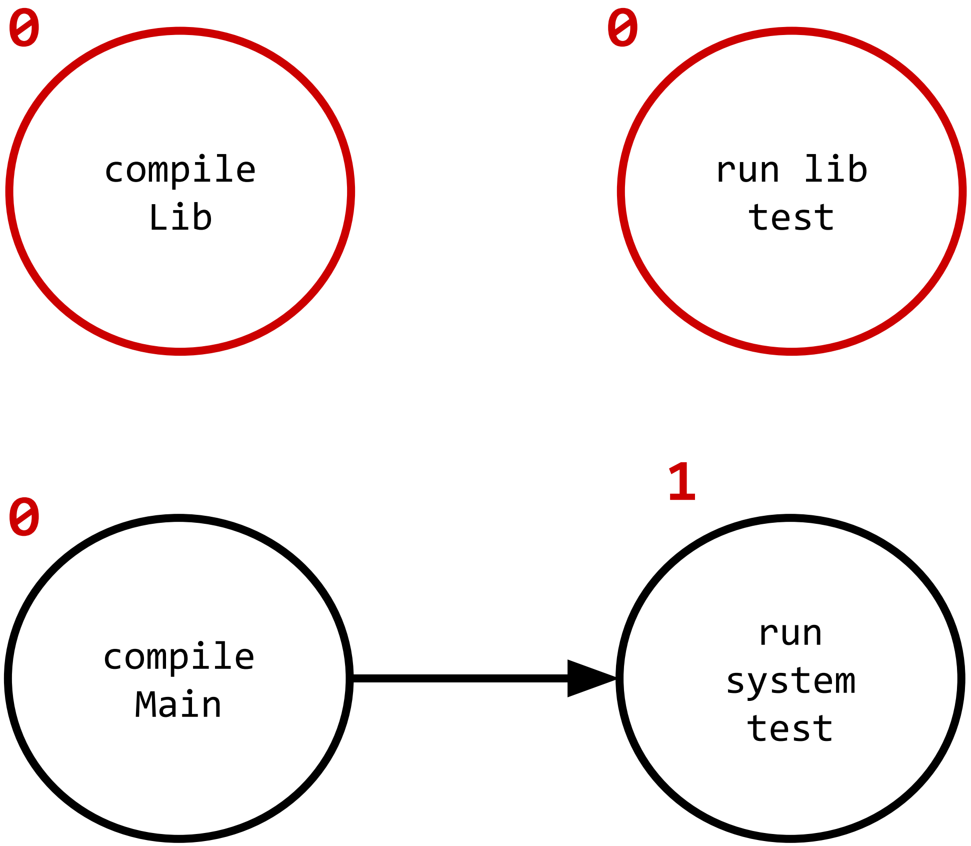
Build systems: topological sort



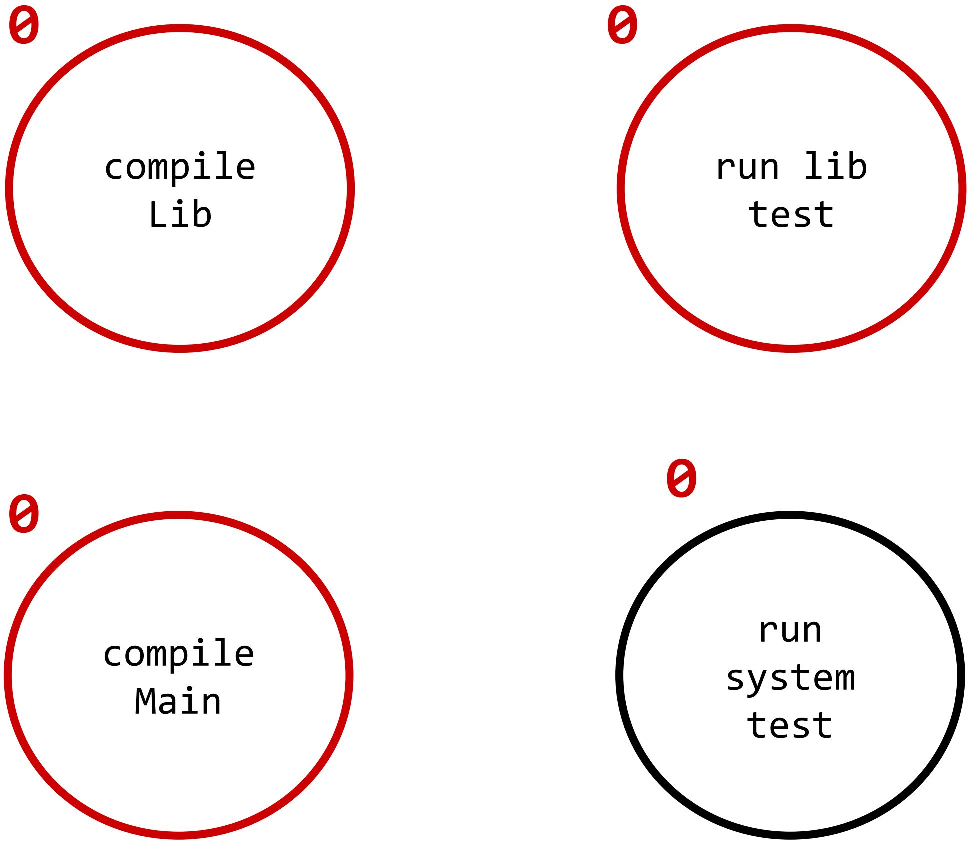
Build systems: topological sort



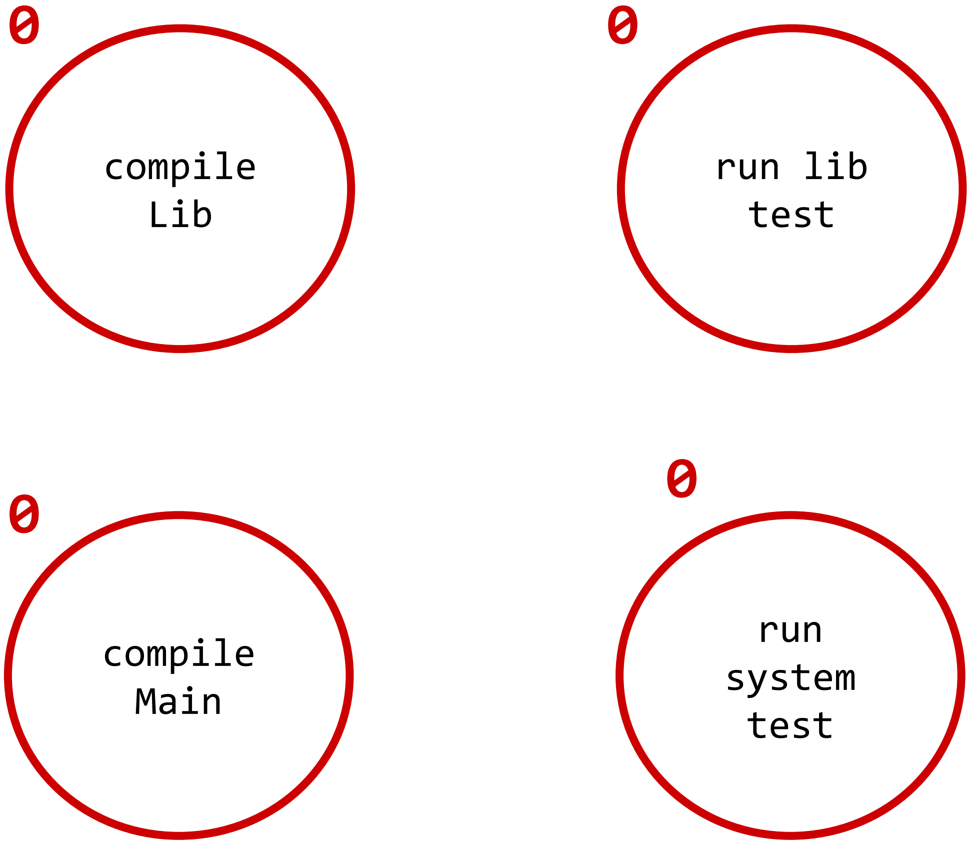
Build systems: topological sort



Build systems: topological sort



Build systems: topological sort



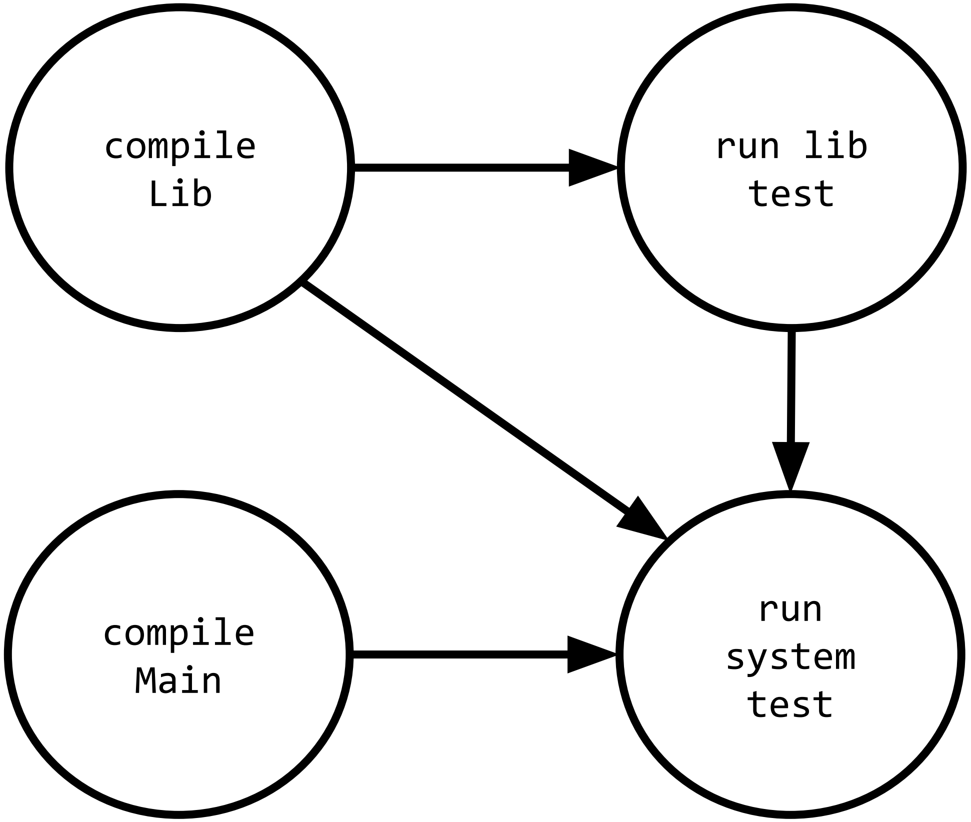
Build systems: topological sort

Valid sorts:

1. compile Lib, run lib test, compile Main, run system test

2. compile Main, compile Lib, run lib test, run system test

3. compile Lib, compile Main, run lib test, run system test


compile Lib
run lib test
compile Main
run system test