Here is a detailed step-by-step Python project to build a **Test Coverage Dashboard** using your specified tools—coverage.py, pytest, matplotlib, and Flask or Streamlit—fulfilling the mini guide and deliverables with code examples:

**Test Coverage Dashboard Python Project**

**Project Overview**

* **Objective:** Analyze Python codebase test coverage visually
* **Tools:** coverage.py (measure coverage), pytest (run tests), matplotlib (visualize), Flask/Streamlit (UI)
* **Steps:** Clone repo, write tests, run coverage, parse data, plot coverage visuals, highlight uncovered files, show results in web UI
* **Deliverables:** JSON/HTML coverage reports, Matplotlib dashboard plot, interactive Flask/Streamlit app, sample repo structure

**Step 1: Clone Repo and Write Tests**

Create a simple calculator module calc.py:

python

*# calc.py*

**class** Calculator:

**def** add(self, a, b):

**return** a + b

**def** subtract(self, a, b):

**return** a - b

Create pytest tests tests/test\_calc.py:

python

*# tests/test\_calc.py*

**from** calc **import** Calculator

**def** test\_add():

calc = Calculator()

**assert** calc.add(2, 3) == 5

**def** test\_subtract():

calc = Calculator()

**assert** calc.subtract(5, 3) == 2

**Step 2: Run Coverage and Generate Report**

Install dependencies:

bash

pip install coverage pytest matplotlib flask streamlit

Run tests with coverage measurement and generate reports:

bash

coverage run -m pytest

coverage json -o coverage.json *# JSON output for parsing*

coverage html *# Visual HTML report in htmlcov/*

**Step 3: Parse JSON Coverage Data**

Create parse\_coverage.py:

python

**import** json

**def** parse\_coverage\_json(json\_path='coverage.json'):

**with** open(json\_path) **as** f:

data = json.load(f)

coverage\_info = {}

**for** filename, file\_data **in** data['files'].items():

total\_statements = file\_data['summary']['num\_statements']

missed\_statements = file\_data['summary']['missing']

coverage\_pct = 100 \* (total\_statements - missed\_statements) / total\_statements **if** total\_statements **else** 100

missing\_lines = file\_data.get('missing\_lines', [])

coverage\_info[filename] = {

'coverage': coverage\_pct,

'missing\_lines': missing\_lines

}

**return** coverage\_info

**if** \_\_name\_\_ == "\_\_main\_\_":

coverage = parse\_coverage\_json()

**print**(coverage)

**Step 4: Plot Coverage % by File (Matplotlib)**

Create plot\_coverage.py:

python

**import** matplotlib.pyplot **as** plt

**def** plot\_coverage(coverage\_dict, output\_file='static/coverage\_dashboard.png'):

files = list(coverage\_dict.keys())

coverage\_values = [coverage\_dict[f]['coverage'] **for** f **in** files]

colors = ['red' **if** c < 80 **else** 'green' **for** c **in** coverage\_values]

plt.figure(figsize=(10, 6))

plt.barh(files, coverage\_values, color=colors)

plt.xlabel('Coverage Percentage')

plt.title('Test Coverage by File')

plt.xlim(0, 100)

**for** i, val **in** enumerate(coverage\_values):

plt.text(val + 1, i, f"{val:.1f}%", va='center')

plt.tight\_layout()

plt.savefig(output\_file)

plt.close()

**Step 5: Display Results in Flask Web UI**

Create app.py:

python

**from** flask **import** Flask, render\_template

**import** os

**from** parse\_coverage **import** parse\_coverage\_json

**from** plot\_coverage **import** plot\_coverage

app = Flask(\_\_name\_\_)

@app.route("/")

**def** dashboard():

coverage = parse\_coverage\_json()

plot\_coverage(coverage)

coverage\_img = os.path.join('static', 'coverage\_dashboard.png')

**return** render\_template("index.html", coverage\_img=coverage\_img, coverage=coverage)

**if** \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

Create templates/index.html:

xml

<!DOCTYPE html>

<html>

<head>

<title>Test Coverage Dashboard</title>

</head>

<body>

<h1>Test Coverage Dashboard</h1>

<img src="{{ coverage\_img }}" alt="Coverage Chart"/>

<h2>Coverage Details</h2>

<ul>

{% for file, details in coverage.items() %}

<li><strong>{{ file }}</strong>: {{ details.coverage|round(2) }}% coverage |

Missing lines: {{ details.missing\_lines if details.missing\_lines else "None" }}</li>

{% endfor %}

</ul>

</body>

</html>

Create folder static/ for the plot image file.

**Deliverables**

* **Coverage reports:** coverage.json, visual HTML report htmlcov/index.html
* **Visual dashboard:** Bar chart image coverage\_dashboard.png with color-coded coverage by file
* **Sample repo:** Includes source code, tests, parsing, plotting, and Flask web app