Agenda.

→ Why Testing ?

→ TDD (Test Driven Development)

→ Flaky Test cases.

→ Types of Testing
  - Unit Testing
  - Integration Testing
  - Functional Testing

⇒

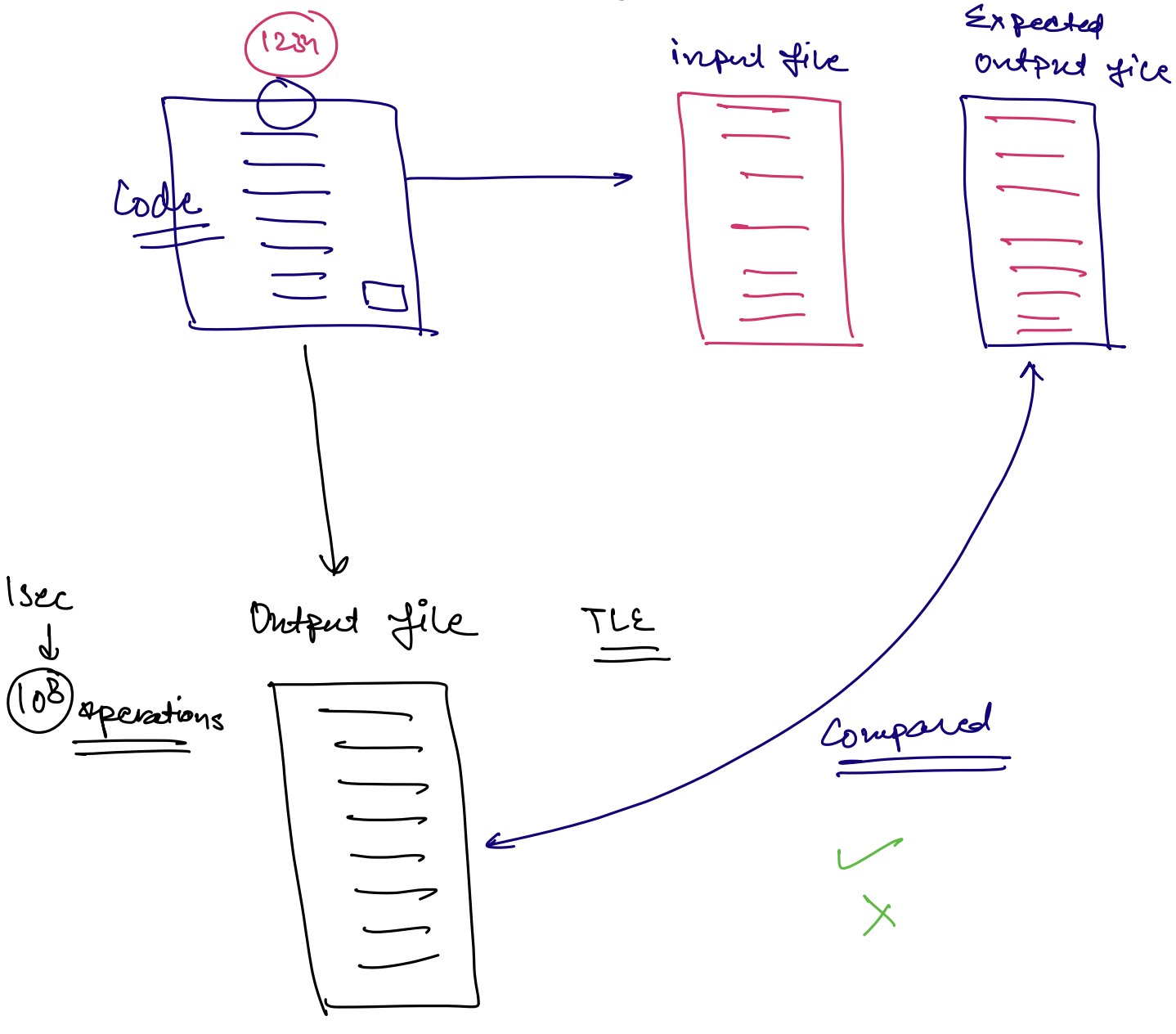
```
int numberOfLanes(int lanesOnOneSide){
    return multiplyUtil(lanesOnOneSide);
}

int multiplyUtil(int x){
    return x*2;
}                    3

int noOfMarriedPeople(int noOfUniquePairs){
    return multiplyUtil(noOfUniquePairs);
}

void testMultiplyUtil(){
    int x = multiplyUtil(2)
    if(x != 2) throw an Exception;
}
```

⇒ We should write test cases, that should get executed automatically before anyone is trying to submit the code and if any of the test case fail, code submission must be allowed.

Github actions.

Edge Cases.

Developers should also write the test cases for the features that they have implemented.



⇒ Before we submit any piece of code all the test cases should be executed.

↓

Comprehensive

Project.

⇒ TDD

↓

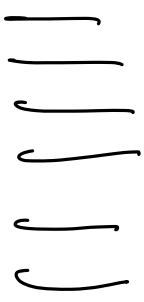
Test Driven Development.

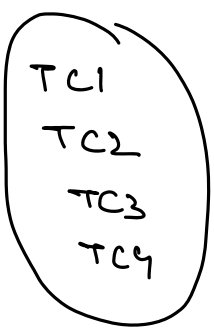
⇒ First write all the Test cases and the implement the feature.

DSA problems on any online judge

⇓

TDD.





⇒ We need to figure out the different conds where our code may break in future.

Flaky Tests. ⇒ Sometimes they pass & sometimes they fail.

↓

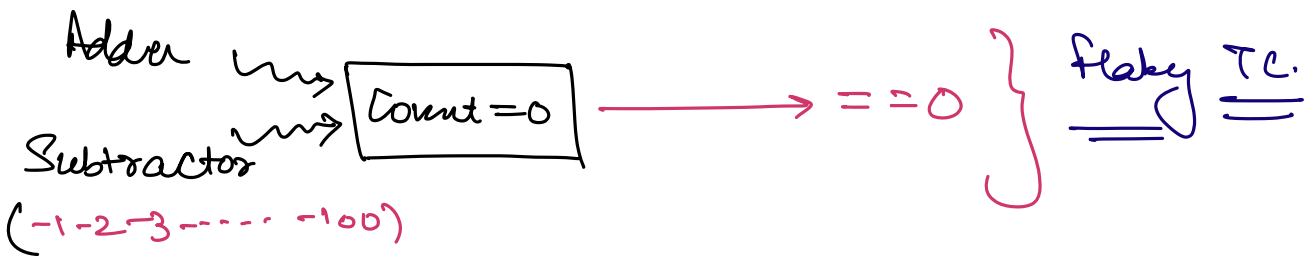
Inconsistent / Unreliable.

→ N/w Connection

→ Randomization

→ Concurrency

(+1+2+3+---- +100)



⇒ Try not to have any flaky TC in your code.

⇒ Critical Section

⇒ Race condition

Asynchronous ≡ Parallel.

⇒ Types of Testing

- Unit Testing
- Integration
- functional

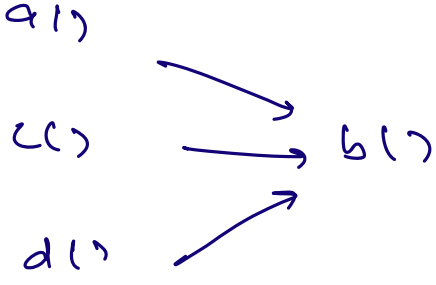
```
a() {
    ==========
    b();
    ==========
}
```

Test case of a() can fail :

1. a() has some bug.
2. b() has some bug

Ideally, the test case of fun (A) shouldn't fail because of a bug in fun (B).

⇒ Testing in Isolation.



=> Test case of fun^ a() should fail if there's an issue in a().

=> MOCKING : Hard coding the output from dependencies.

=> We should test our code in isolation.

UNIT TESTING.

Every individual piece of code should get tested by a testcase.

=> Every test case will be short & fast.

=> No dependency on any other fun^.
(Dependencies will be mocked).

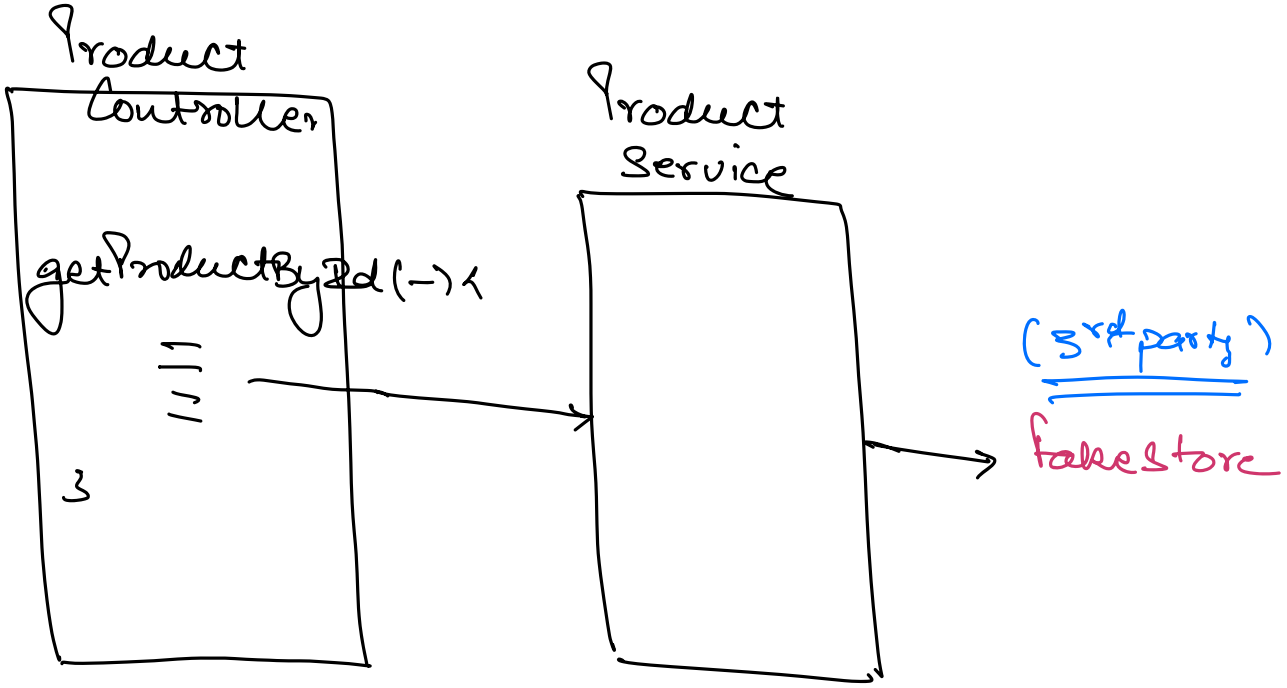
Test Coverage. => % of code covered by the test cases.

$\frac{980}{1000} \approx 98\%$ ✗ $\approx 80\%$

Integration Testing

⇒ Testing each functionality of our software system where all the dependencies will also be triggered as it is they are present in real codebase

⇒



⇒ Actual dependencies will be triggered, still some 3rd party depencies may be mocked.

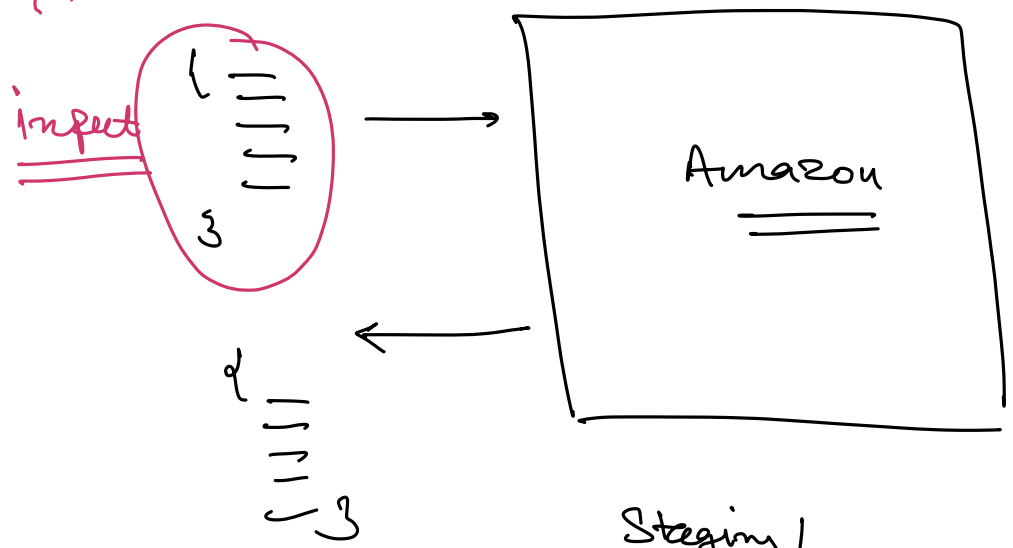
Functional = User Testing

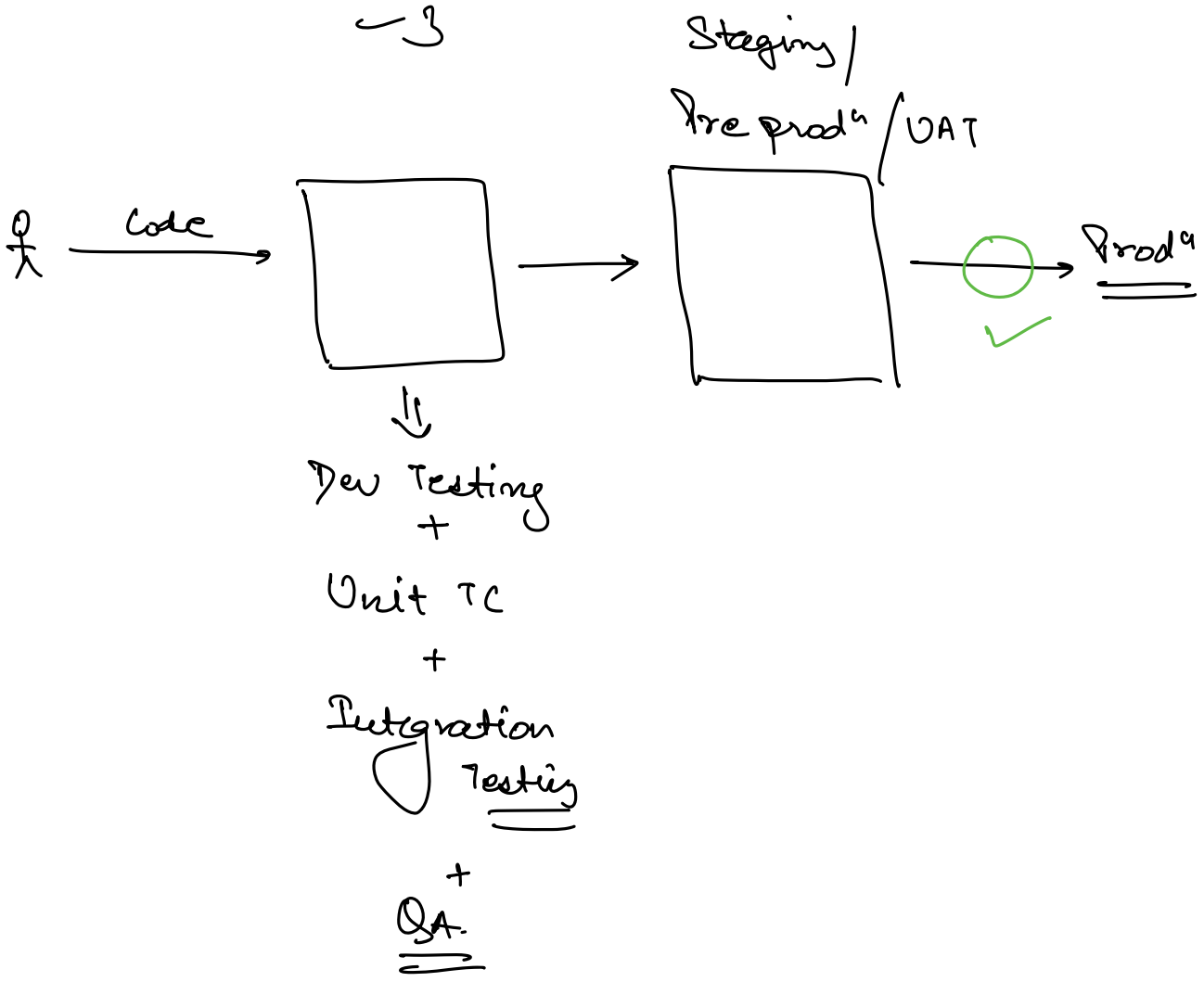
Pre-Prod

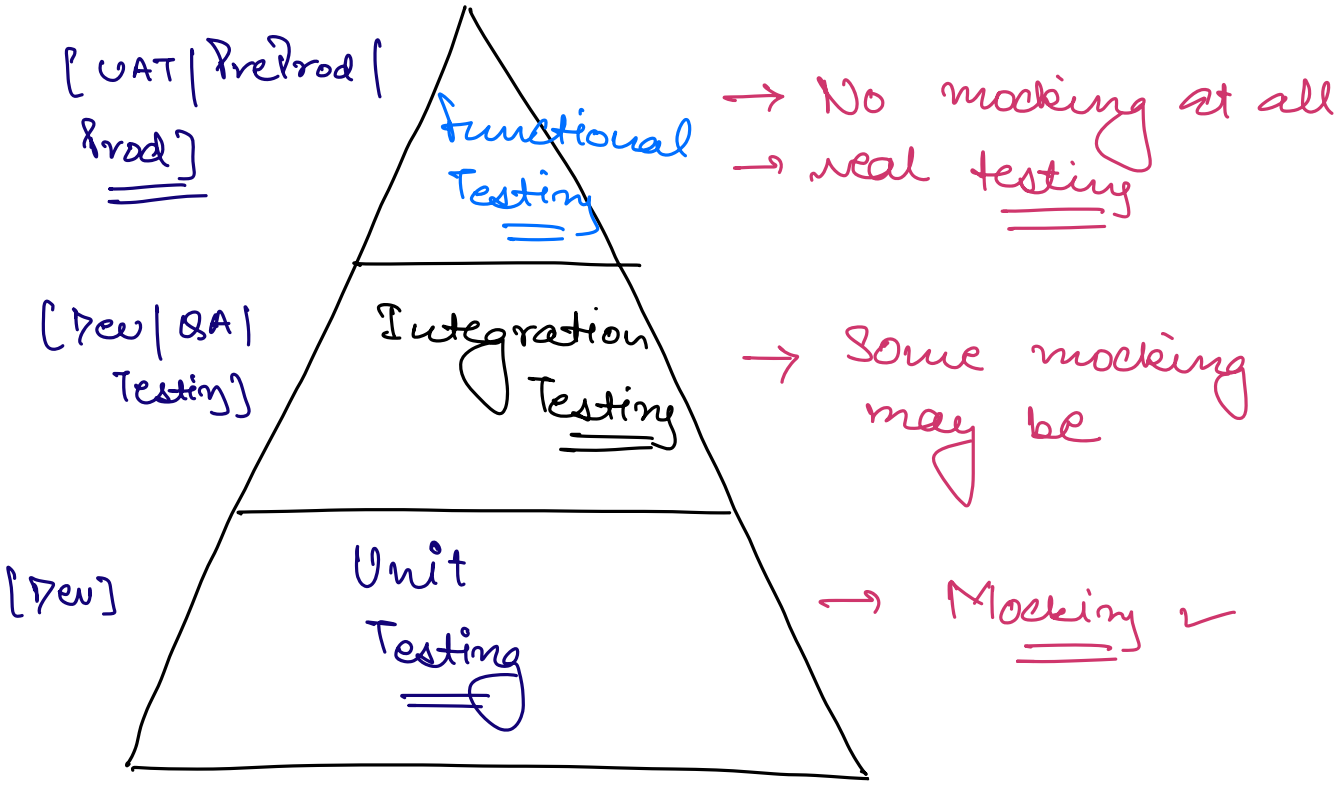
UAT

(User Acceptance Testing)

POST /products






[UAT | PreProd | Prod]
Functional Testing
→ No mocking at all
→ real testing
[Dev | QA | Testing]
Integration Testing
→ Some mocking may be
[Dev]
Unit Testing
→ Mocking

—— * ——