A tester's appreciation of unit tests

Katie Fox

PNSQC 2021
Katie is a…

- Senior software test engineer with e-BUILDER®
- *formerly at Ultimate Software (now UKG)
- 5-year remote worker
- 4-time PNSQC attendee
- First time presenter
- Cat-mom, runner, houseplant enthusiast
Why do I care about unit tests??

- Presentation origin: interviews with senior tester applicants
- My first exposure to them: Ultimate Software tester interview
- Early career buzzwords: “test pyramid” “shift-left testing” “agile”
  - But what do they mean?
- Current focus + responsibilities: lots of tests that look like code
A tester’s appreciation of unit tests
A tester’s appreciation of unit tests
Real team examples over 7 years

• Lessons learned on 4 teams within UKG Payment Solutions

• Collaborative shift left strategy focusing on unit tests

• Most common improvements to unit tests and using them within big-picture quality effort
Baseline “how work got done”

- Cross-functional teams, Kanban agile, business-value stories
- Different technologies = different testing
- SDLC: analysis, coding, testing
Team 1: creating the test pyramid

• Both experienced and novice engineers

• Brand new project

• Tests as documentation and knowledge transfer

• Cohesive test pyramid, stayed in sync
Team 2: inheriting legacy code

• Building on top of legacy code, top down test pyramid

• No tests, no time

• Bad tests = bugs

• No established pattern, large team, different styles, struggle to sync
Team 3: avoiding technical debt

- Small team experienced engineers
- Starting from scratch both code and tests
- Integrating with legacy process, code duplication
- Use tests as clues to refactor proactively
Team 4: rediscovering quality

- Team out of practice with writing tests
- Began including a tester in planning
- Quality improved with a dedicated person
- Big-picture planning and impact
Team summary

- Team 1: starting strong without full appreciation for it
- Team 2: learning how painful it is to catch up
- Team 3: starting strong with full appreciation for its value
- Team 4: re-establishing testing without having to catch up
Collaborative shift-left strategies

• Story kick-offs
• Testers in code reviews
• Dev + Test automation discussions
• Story desk-checks
Improving unit tests

1. Name tests descriptively
2. Keep setup clean
3. Remove branching logic
4. Write assertions that fail better
Improving unit tests - 1

• Name tests descriptively

• [Insert example]
Improving unit tests - 2

- Keep setup clean
- [Insert example]
Improving unit tests - 3

- Remove branching logic

- [Insert example]
Improving unit tests - 4

• Write assertions that fail descriptively

• [Insert example]
Using unit tests as a tester - 1

• Simplifying webpage testing

• [Insert example]
Using unit tests as a tester - 2

- State transition rules

- [Insert example]
Using unit tests as a tester - 3

- Known risk – batching

- [Insert example]
Using unit tests as a tester - 4

- Known risk – idempotency

- [Insert example]
Conclusion: unit tests are valuable!!

• Establish an early foundation when possible
• Collaborate with developers to create valuable test scenarios
• Design them to provide useful errors
• Keep iterating and improving as a team
Thanks for listening!

• Questions, comments, concerns: let’s talk!
  • [www.linkedin.com/in/katielynnfox/](http://www.linkedin.com/in/katielynnfox/)
  • katiefoot9@gmail.com