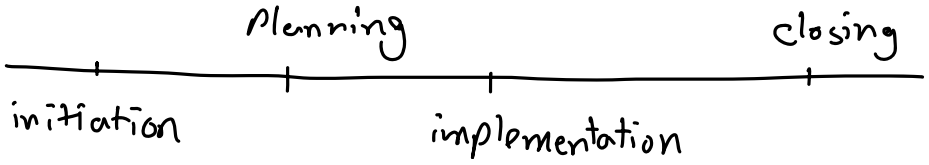
Agile Testing

⚠ Agile tesing ⟶ build software, then test X



⚠ testing at the beginning of a project

- review product requirements
  - ask clarifying questions
  - extract information to set a quality standard
  - create a plan outlining the product's testability
  - implement a structure to realize quality

⚠ Proper technical knowledge enables to dive into code to evaluate underlying risk

⚠ Build, then test problems

- delayed feedback
- disconnected teams
- costly, obrious bugs

⚠ shifting testing left

↳ verify and validate designs & code as they're created throughout a project's duration

⚠ the relation of testers with other departments like Tech, product, copy, legal is crucial because of creating faster, shorter feedback loops

⚠

existing software	new software
collect legacy knowledge	gather project assumptions
understand pain points	consider stakeholder requirements

⚠ A project plan addresses the way to test the product being built

⚠ Backlog grooming

↳ agile meeting where user stories are discusseed

⚠ Sprint planning

↳ evaluate the work to be done, by prioritizing work in the backlog

⚠ Grooming tickets

- Is the user story still valid?
- Is the use case relevant?
- what is the impact of the user story?
- how long will a solution take to build?

⚠ Tester role in backlog grooming

- outline dependencies
- determine testing timeline
- accurately determine testability
- expose scope shortcomings

⚠ Sprint planning

- outline details
- calculate measurements
- agree on acceptance criteria

⚠ During sprint planning, a tester determines if acceptance criteria can be properly validated and tested

⚠ The three amigos → stakeholder, developer, tester

⚠ Changes in technology are deliverables since ticket planning are considered at story kickoff

⚠ Feedback is the foundation of progress in agile projects

⚠ Retrospective goals

- understand team best practices
- note process issues
- discuss potential improvements

⚠ Agile processes are designed to provide constant and regular reminders to communicate with the team

⚠ Flagging issues and testing software health

- determines if a project is timely or at risk
- specific information will help refine the process
- feedback allows for future comparisons

⚠ Bug tracking common questions

- what's our naming conventions?
- where was the bug found?
- how can we reproduce the bug?

⚠ Determine severity of the bug

- how many it will affect?
- how much money will the company lose if released into production?
- how costly is the time to fix the issue going to be?

⚠ Team expectations

- definitions of ready and done
- decisions for tasks
- handoffs for stories
- creation of a workflow

⚠ what to test

- what are we building?
- why are we building this?
- who are we building this for?

⚠ A test plan is a general guide created to define the scope of work that will be applied during testing

⚠ Prioritized testing list

- browsers
- clients
- devices
- platforms

⚠ Manual testing include automated processes that need to be executed or stewarded by a person

⚠ Positive testing ~ ensure what is supposed to be present

⚠ Negative testing ~ ensure that any misinformation or incorrect entry is calculated appropriately

⚠ Exploratory testing

- the cognitive approach to addressing software by stretching, moving, trying, or even shrinking aspects of the software

⚠ load testing

- understanding the conditions of a software when under extreme exposure or visitation

⚠ Test automation

- a series of tests that need to be run in succession at rapid speed will be ideal for automation

⚠ Continous integration

- the practice of regularly pushing code to a shared repository
- can be completed at the end of each step or each epic