W07-2: Software Quality

Dr. Alex Adkins

CPSC 3720: Software Engineering

checkin.cs.clemson.edu:

Understanding APIs Review

With your teams…

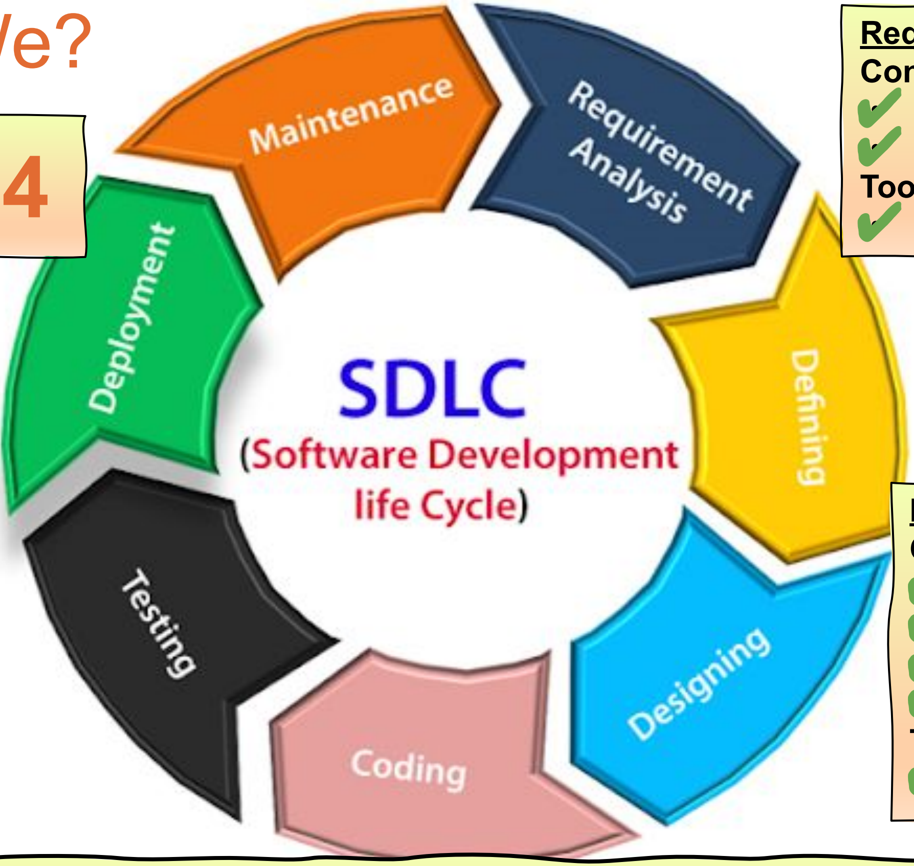
Each team member: Present your APIs!

- Demo your API to your Team
- Discuss:
  - Authentication method
  - Chosen Endpoints & Requests

As a team:

- What differed between APIs?
- Did some teammates have it easier than others with their chosen API? Why?
- Someone having trouble with their API? Use this change to help them!

Where are We?



Deployment & Maintenance 4

Concepts:
- DevOps

Requirements Analysis 1

Concepts:
- ✓ Epics/Stories
- ✓ Gathering Reqts

Tools:
- ✓ Trello

Define/Design 2

Concepts:
- ✓ Microservices
- ✓ Cohesion & Coupling
- ✓ API First
- ✓ API Specification

Tools:
- ✓ Postman

Coding and Testing 3

Concepts:
- Code Mgmt
- Test techniques

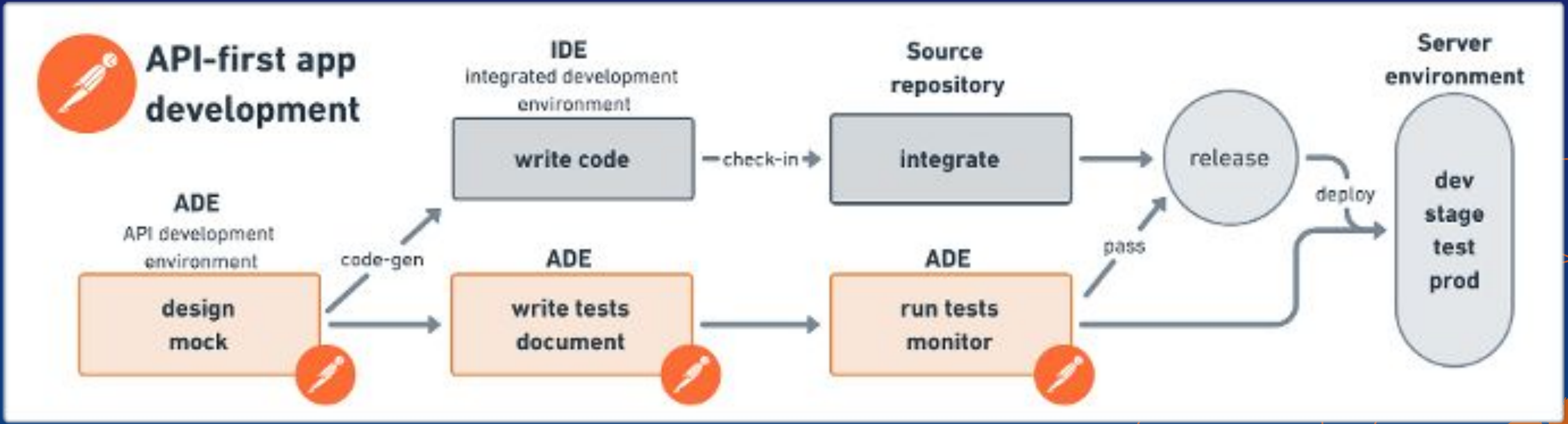
Tools:
- GIT
- AWS API Dev Tools
- Postman

FOUNDATIONAL CONCEPTS

- ✓ Use of a **software process** to address software complexity; **Agile/Scrum** for class
- ✓ **Effective Teaming** is essential throughout SDLC; Conway's Law + Team=Software
- ✓ Software **planning and estimation** is hard! Relative sizing and storypoints

Mocks & Examples: Why do we need them?

- Supports API First through enabling continuous development
  - Across all stages of the agile development process, teams use mocks to decouple the development process, empowering people to work independently and in parallel



Software Quality Learning Objectives

- Understand the importance of software quality in software development
- Discuss and review methods of ensuring quality
- Recap on mocks and examples for testing APIs continuously and supporting "shift left" quality

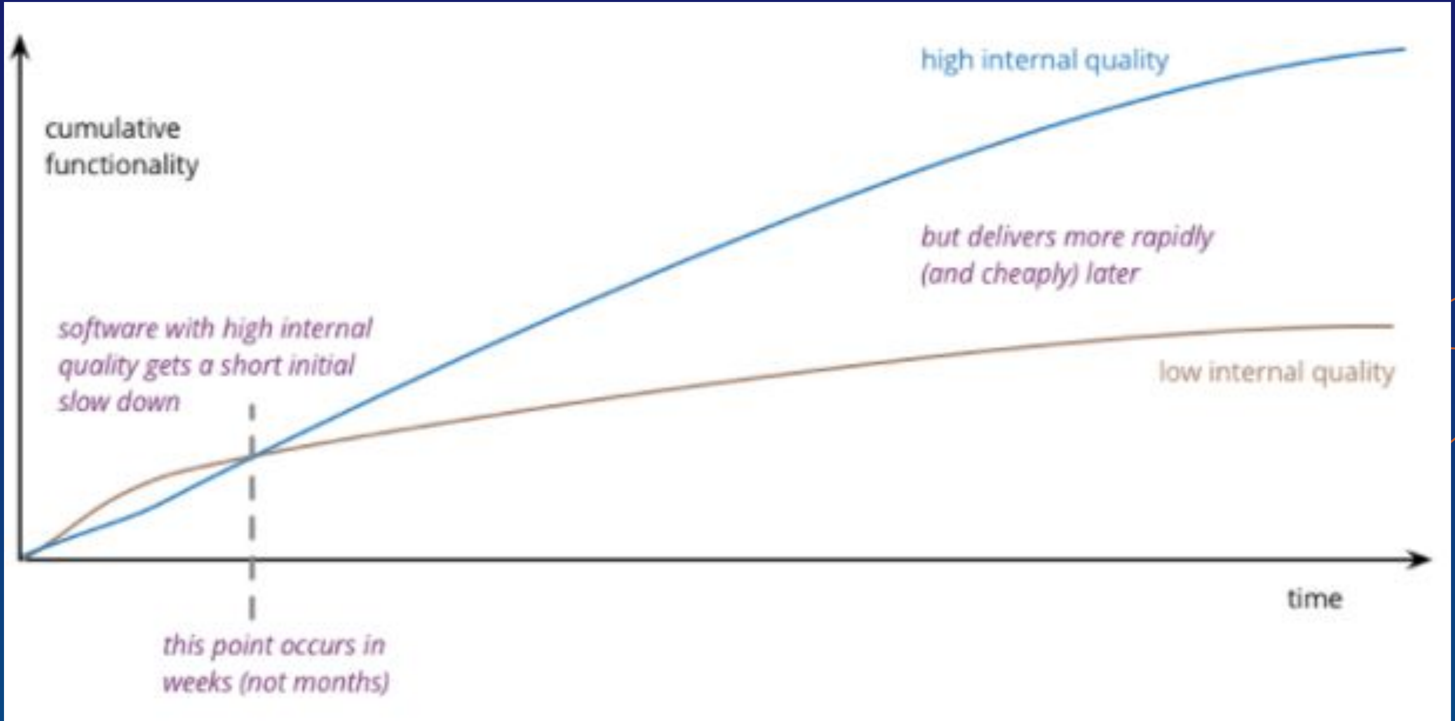
Is Quality Worth the Cost?

I SEE YOU TEST YOUR CODE IN PRODUCTION
I TOO LIKE TO LIVE DANGEROUSLY

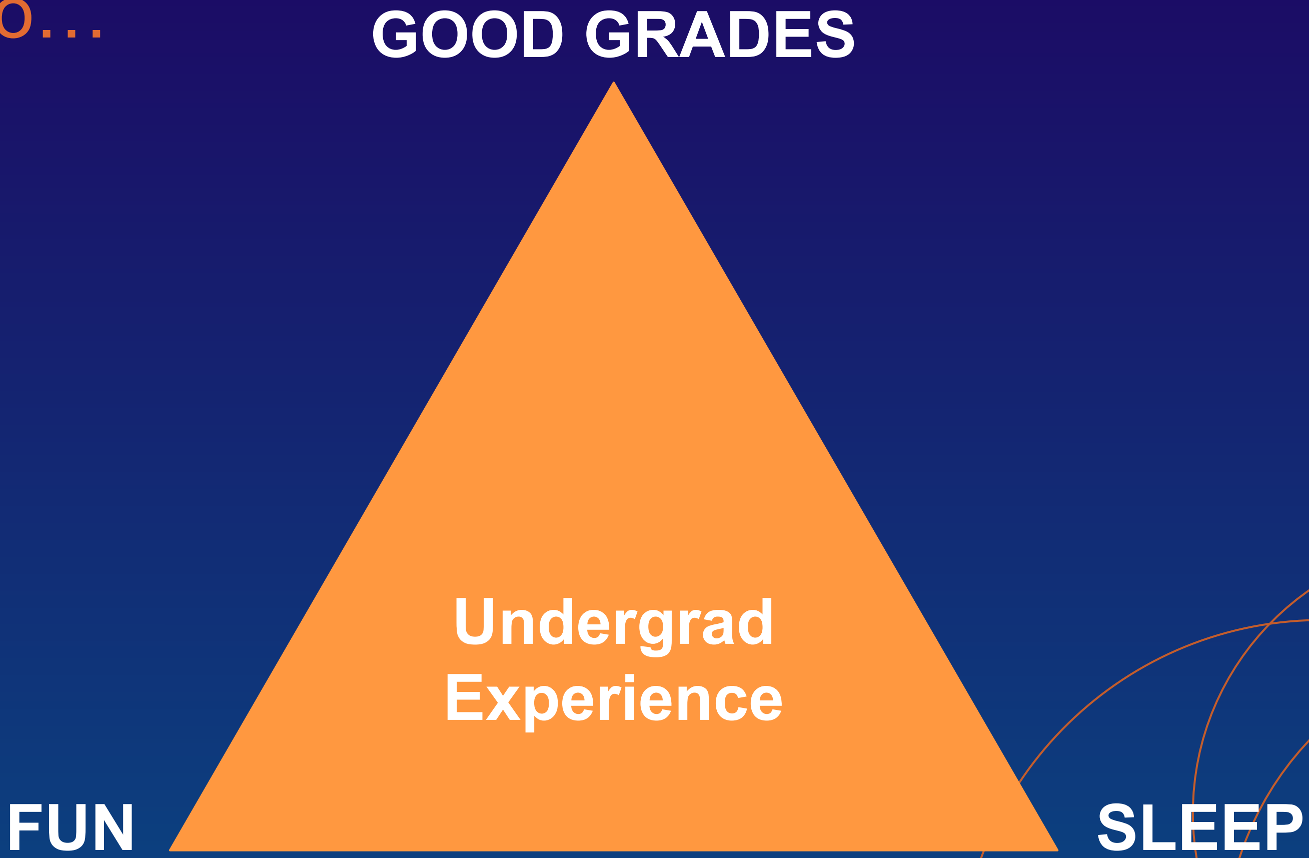
Is it worth it?

Cruft

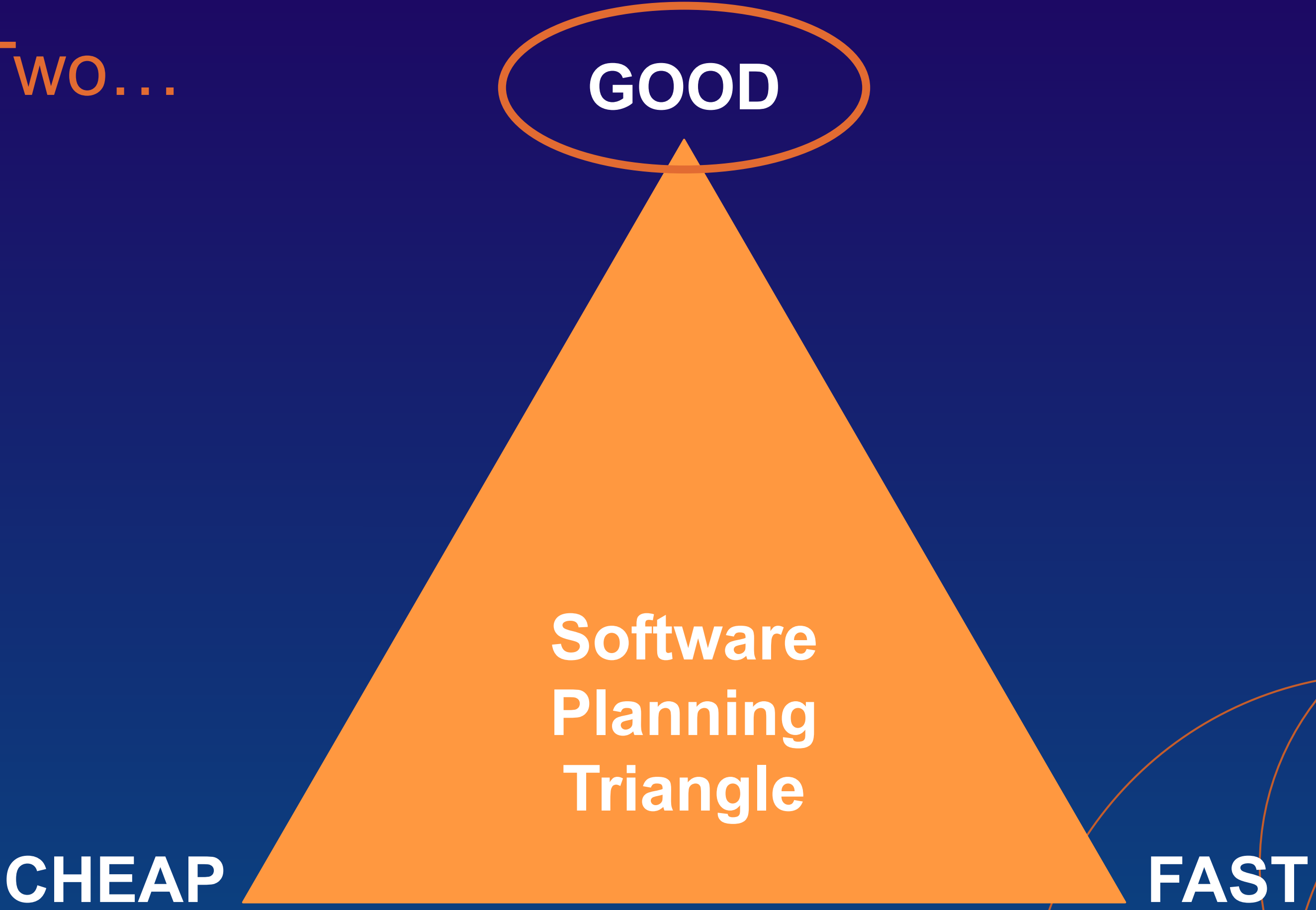
- **Cruft**: leftover, redundant, or poorly designed code
  - can lead to inefficiencies, bugs, or increased complexity
  - accumulates over time, usually from quick fixes, workarounds, or technical debt



Pick Two…



Pick Two…



checkin.cs.clemson.edu:

Discuss

At your tables…

What is “good” software?

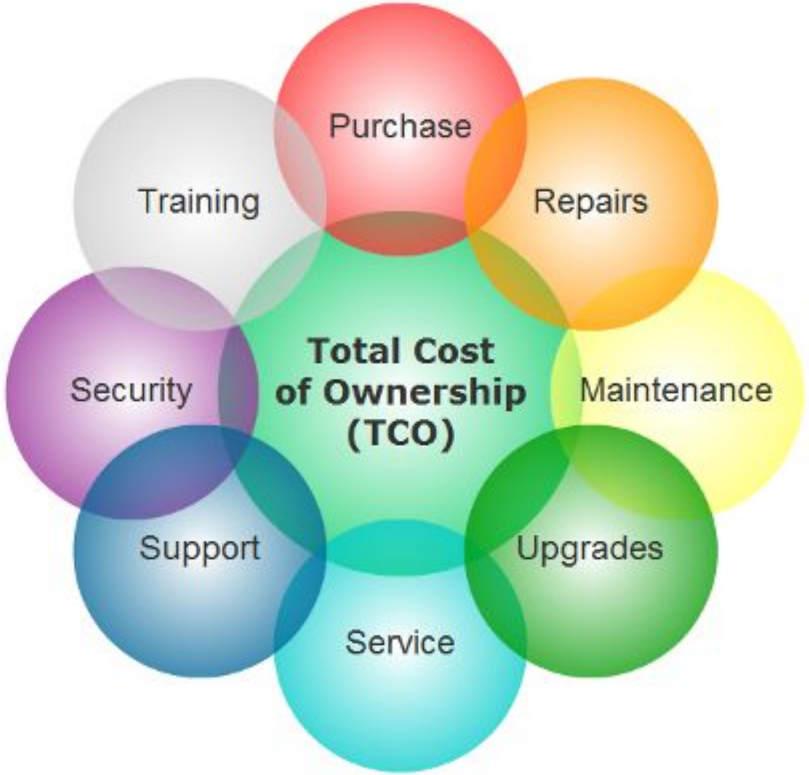
How do you deliver “good” software?

Good Software means Focusing on the "...ilities"

What do each of these mean for good software?

- Usability
- Maintainability
- Performance (or Perform-ability)
- Scalability
- Extensibility
- Security
- Portability (if delivered on-prem)
- Reliability and/or Availability
- Internationalize-ability
- Interoperability
- Audit-ability
- Administrability
- Configurability

Software Buyers Care about Quality as it relates to Total Cost Ownership



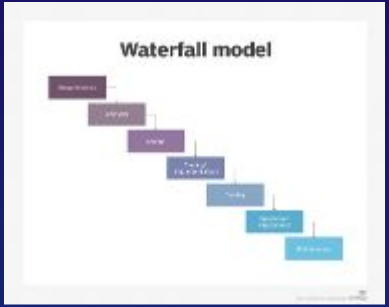
Good Software through Testing

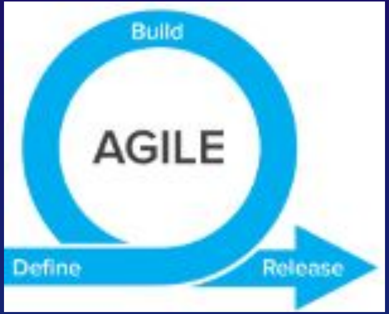
- **Functional Testing**:
  - Unit Testing: code working properly
  - Functional Testing: doing what you said it would do
  - System and Integration Testing: does it all work together
  - Exploratory Testing: testing on the fly
  - Regression Testing: Tests to ensure that new functionality did not break existing functionality
    - (can also be used for non-functional testing)
- **Usability Testing**:
  - A/B testing: usability tests for customers
  - Usability Studies: observe customers using the software software or mocks

Good Software through Testing

- **Non-functional Testing**:
  - Recovery testing: forces software to fail and verify that recovery is (ala Chaos Monkey) properly performed
  - Security testing: ensure protection mechanisms built into a system will, in fact, protect it from improper penetration
  - Stress or soak testing: executes system in a manner that demands resources in abnormal quantity, frequency, or volume
  - Performance Testing: test the run-time performance of software within the context of an integrated system

Testing - Waterfall vs. Agile

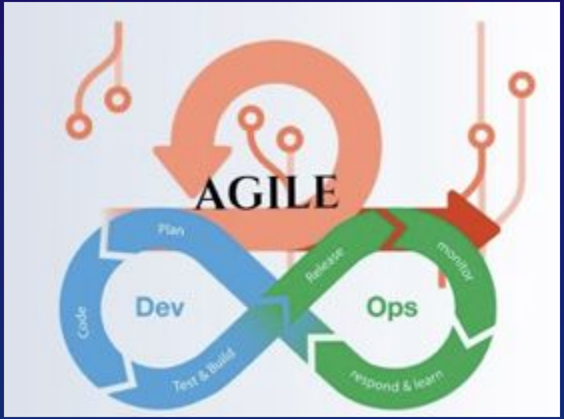




- Functional Testing at the end of development cycles
- Testing Teams usually separate organizations from development
- Regression testing for each release
- More manual testing

- Testing during each sprint and in parallel
- Scrum teams own responsibility for quality
- Regression testing ongoing
- Increased need for automation

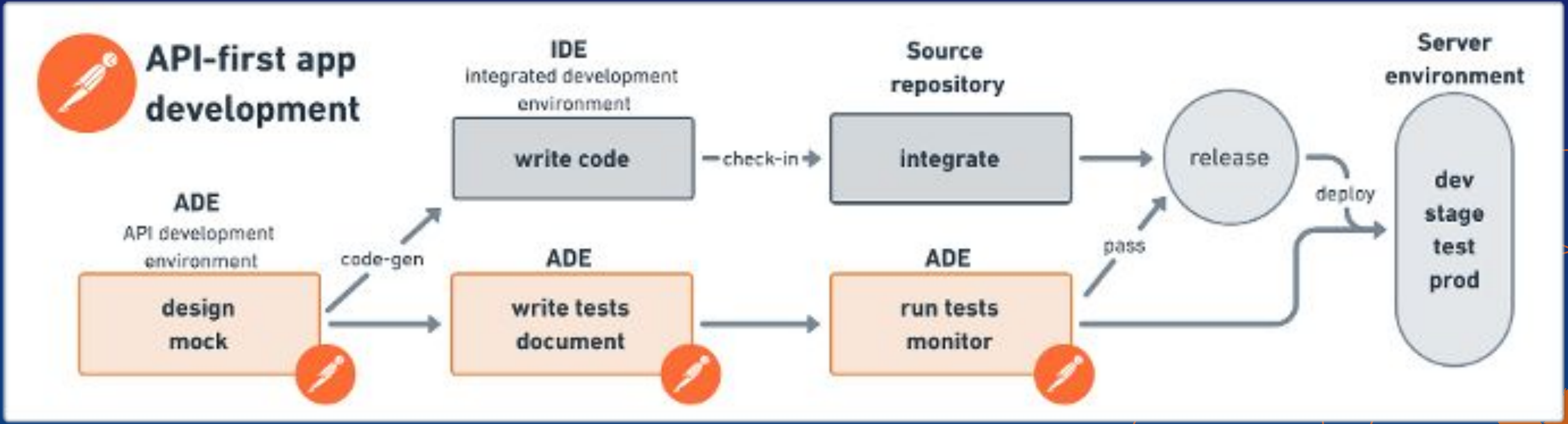
Testing - DevOps



- **“Shift Left”** and Test-Driven Development (TDD)
- More blur between Dev and Test roles
- Automation is ESSENTIAL

Postman Mocks and Examples: Support TDD/Shift Left

- A **TDD/Shift-left** practice which supports **API First** through enabling continuous development:
  - Across all stages of the agile development process, teams use mocks to decouple the development process, empowering people to work independently and in parallel



<<<<<<<<< SHIFT LEFT: TESTING THROUGHOUT THE SDLC <<<<<<<<<

CUSports Project: Sprint 1

- Sprint 1 (100 points)
- Due March 4
  - Some teams will demo on this day!

- You have been given Epics for Account & Notifications Services
  - Break down these Epics into User Stories & Tasks!
  - For each story you will likely need to create an API call in your Test Collection
- For each Service you will need to create:
  - API Specification in YAML/JSON
  - A Definition Collection in Postman
    - Should include Examples uploaded to a Mock Server
  - A Test Collection in Postman
    - This Test Collection should run with Postman's Runner tool!
    - Test Collection should "Demo" Definition Collection's Examples

CUSports Project - Sprint 1 Planning Goals

- Break up Epics into Stories & Tasks
  - Put them into the Stories List
  - Group overlapping or dependent stories
  - Make sure each story has acceptance tests & a size label
- Prioritize the Sprint 1 Stories and Order them Accordingly
- Identify Tasks/Stories that you can start working on after class today
  - Put these into the TODO list on the Kanban Board

CUSports Project - Sprint 1 Product Owner Help

- Ask questions if you don't understand the story
- Ask for help on creating the API Specification and Mocks/Examples
- Use office hours to your advantage!
  - Dr. Alex: Mondays 2:00-4:00 in McAdams 317
  - Shwetha: Mondays & Wednesdays 4:30-5:30 on Microsoft Teams
  - Kalyani: Tuesdays & Thursdays 3:00-4:00 on Microsoft Teams

Sprint 1 Planning Meetings

Define Sprint Goal & Deliverables:

1. Review Sprint 1 Epics
2. Breakdown Epics into Sub-Stories & Tasks
3. For each story identify required API endpoints, methods, & data required
   a. These will form your mocks & examples
   b. Identifying these can be their own task(s)
4. Assign Task Owners
5. Define Acceptance Criteria for each Story & Task
6. Time Estimations & Prioritize each Task
7. Update Kanban Board

Reminder: Midterm next Thursday!

- Study guide will be posted tomorrow morning