Testing Concepts

* Testing – validating that our application works as it should
* Defect/bug – some behavior in our code that is not intended
  + Is not specifically bad (Often can be though)
    - Severity
      * How much a bug affects the functionality of a feature
    - Priority
      * How important it is this bug gets resolved
    - They do not always correlate to each other
  + Ex High Severity Low Priority
    - Editing an Employee’s last name is completely broken
      * High Severity because it greatly impacts employee editing feature
      * Low priority rarely used feature and employee id is used to track employees not their last name
  + Ex High Priority Low Severity
    - You log onto the website’s main page and competitors logo is used instead of your own
      * Low Severity because it doesn’t impact functionality
      * High priority because it confuses your customers

Test document vocab

* Test case – singular validation of a piece of code
* Test suite – group of related test cases
* Test Strategy
  + High level document
  + Usually company wide
    - Spans across all projects by a company
  + Explains how developers are supposed deal with bugs and testing
    - Roles and responsibilities
      * Ex all API endpoints must be accompanied by a API specification and Postman tests verifying functionality
        + Maybe includes a test report history showing test pass
    - Work flow for resolving bugs
      * All bugs should be reported to the project manager who will then delegate the task to who he deems appropriate
* Test Plan
  + Lower level document
  + Specific to a project
  + What exactly are we testing?
    - What methods are we going to test
    - How we are going to test them
      * Number of cases?
      * What technology
  + What the deadlines and schedule
    - Every Friday at 1:00 a weekly testing document showing progress is created

Types of Testing

* White box testing
  + Testing when you can see the code (JUnit5)
* Black box testing
  + Testing when you cannot see the code you are working on
    - API tests to a backend you do not have access to
* Gray box testing
  + Testing when you can see parts the code
    - Uses some library methods which you cannot see
* Positive testing
  + Testing to make sure something works as it should
* Negative testing
  + Testing to make sure something fails as it should
    - Throws the correct exception
* Acceptance testing (User Acceptance Testing)
  + Human beings test the application for friendliness.
    - Has to be done manually no machine can tell if the web page is intuitive to use
  + Alpha
    - Internal developers before UAT
  + Beta
    - Sample end users who are not connected to development use the application
* Regression testing
  + Testing that new code added the application does not break any existing code.
    - E2E tests and continuous are immensely helpful for this type of testing
* Retesting
  + Running your tests again. Usually done to see if you fixed a problem.
* Functional testing
  + Checking to make sure that your application procures the correct outputs for the given inputs
    - You click on a button you get the correct page
    - Given some parameters to method you get the correct output
    - NOT concerned with efficiency
* Performance testing
  + Test the efficiency of your application.
    - Can be trying to make several aspects or your application more efficient
      * Time
      * Memory
      * Processing power
  + Load testing- testing an application by sending it more and more requests until it breaks. Find the breaking point and see how it fails
    - Usually it mimics normal usage of the application just high traffic
  + Stress testing-
    - Trying to break an application but focusing on pressure points and users doing unusual things
      * Ex. What if 10,000 users all decided upload an image at the same exact time
  + Endurance Testing
    - Load testing + time
      * Helps spot problems that can take days/weeks to occur
  + Spike testing
    - Bit more specific to cloud computing
    - How quickly your application can scale to meet demand