


# - 09 - Programming Foundations: Software Testing/QA

🔗 Certificate	<a href="#">CertificateOfCompletion_Programming_Foundations_Software_TestingQA.pdf</a>
# Completed Sections	6
🔗 Course Links	<a href="https://www.linkedin.com/learning/programming-foundations-software-testing-qa?resume=false">https://www.linkedin.com/learning/programming-foundations-software-testing-qa?resume=false</a>
# Goal Sections	6
☰ Quick note	<a href="https://www.linkedin.com/learning/certificates/0e2cdc1f721617e2c42246816bfa5d9419274a88fa886e69c5d6?trk=share_certificate">https://www.linkedin.com/learning/certificates/0e2cdc1f721617e2c42246816bfa5d9419274a88fa886e69c5d6?trk=share_certificate</a>
📅 Start and Finish Date	
🖼️ Thumbnail	
⌵ Time	53m 51s
⌵ Understanding %	70%-100%
Σ نسبة الإنتهاء	✅ Done

## What is quality assurance?

### ▼ Quality Assurance

- A systematic process used to determine whether a product meets specifications
- The name of the Specialists



- An individual in the QA role constantly questions parts of the software development process to ensure the team is building the right product and building it correctly

#### ◦ THE GOAL.

- should be to help their team move quickly with confidence.

### ▼ How to ensure quality

- Quality can be ensured across the code and influenced by process.
  - how to measure



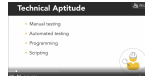
- Together work with:

- Clear specifications
- Code implemented
- Code tested
- Code released

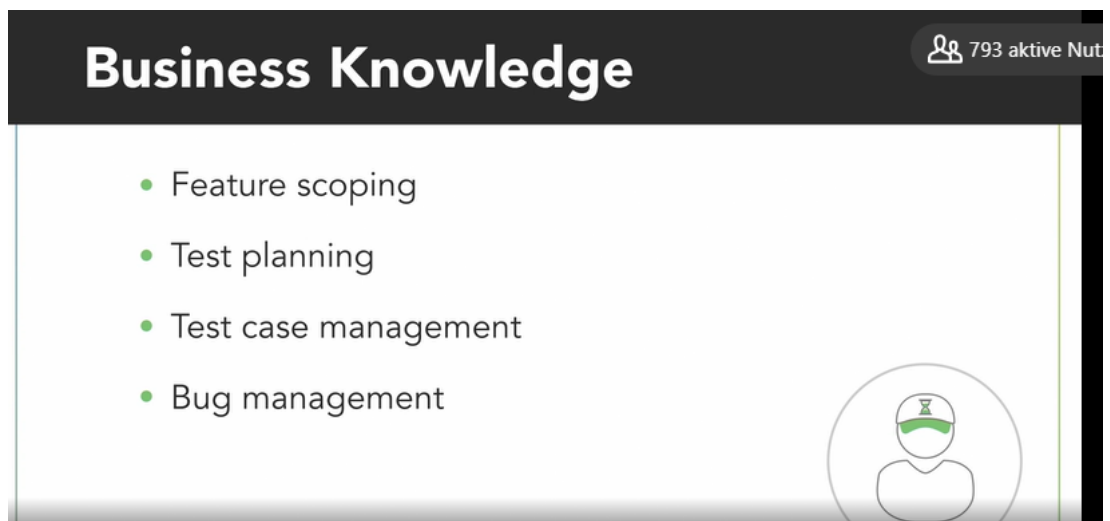
## The Role of QA

### ▼ Roles and responsibilities

- Technical Aptitude



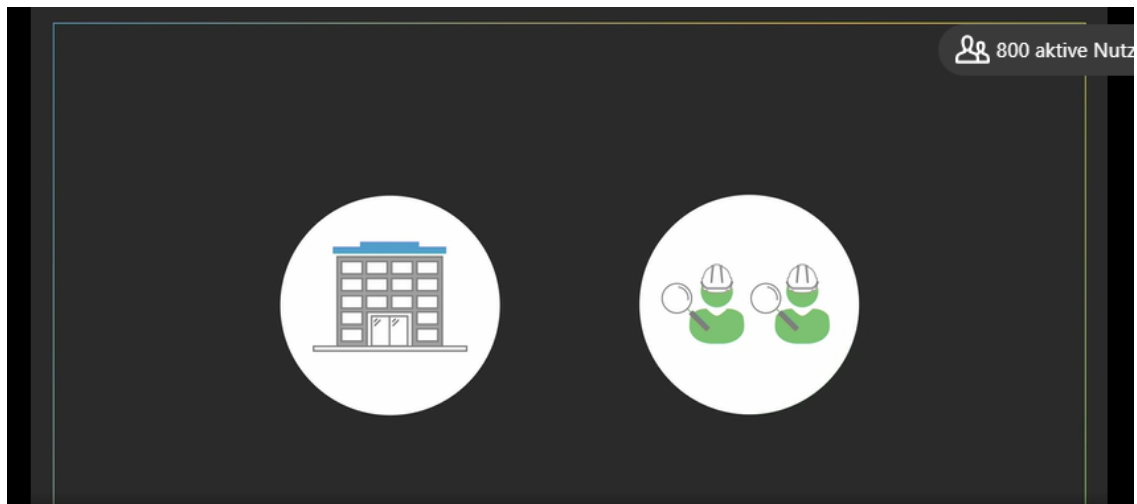
- Business Knowledge



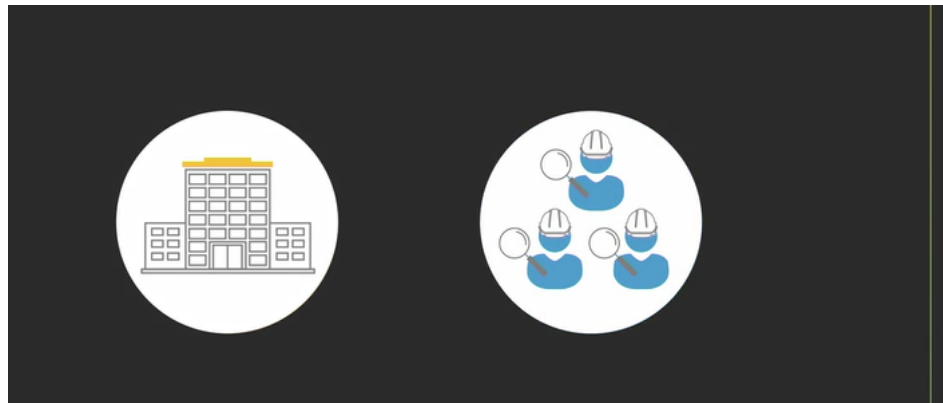
- Process and Release
  - Define and improve testing practices
  - Optimize release process
- Small startup



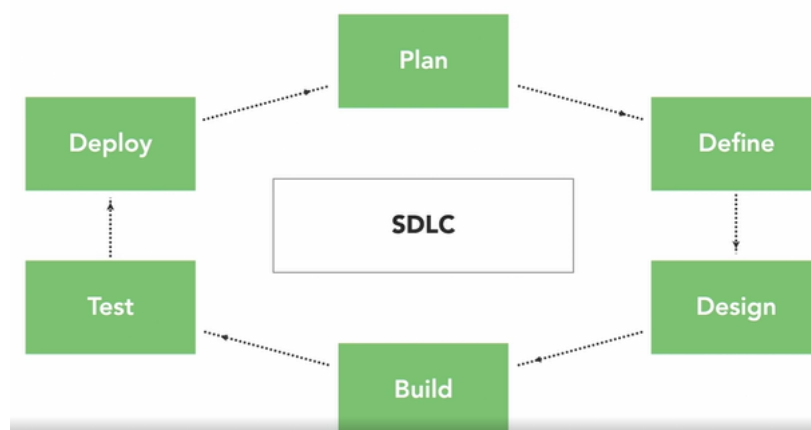
- Middle company



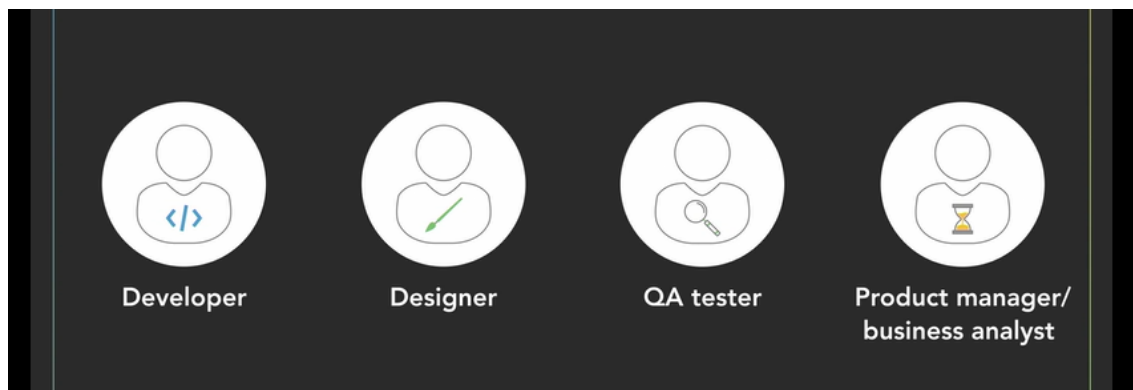
- QA could share responsibility when needed
- Work based on strengths and preference
- Big company



- QAs share responsibility
- Need to excel across some skills
- 
- ▼ Get involved throughout the SDLC
  - The software development lifecycle, is a process that processes high-quality software in the shortest amount of time



- Plan
    - Identify risks
    - Identify use cases
  - Define
    - Write specs and acceptance criteria
    - Decide what's in scope
    - Write test strategy
  - Design and Build
    - Solidify test scenarios
    - Get feedback on scenarios from team
    - Manual and automation tests
  - Test
    - Manual and automation test
    - Acceptance testing
  - Deploy
    - Validate functionality
    - Release
    - Test in production
- ▼ Collaborate with the team
- The team



- QA tester
    - Determine acceptance criteria and scope
    - Provide feedback on test scenarios
  - Designer
    - Provide feedback on mocks or prototype
    - Provide feedback on UI/UX
  - Developer
    - Pair on writing tests
    - validate functionality
- ▼ Set expectations and goals
- Build a relationship with your teammates
  - Know the responsibilities of each role

- collaborate
- share what it means to be a QA
- Sharing Goals
  - Accountability
  - Awareness
  - Support
- Regularly check in with teammate
- Communicate what you are working on.
- Ask for help
- Give and receive feedback
- jo leute was geht ab

## Test planning

### ▼ Create a test strategy

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a6f0c4d4-ad50-43c0-a87e-a85cc2949a77/Sample\\_Test\\_Strategy.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a6f0c4d4-ad50-43c0-a87e-a85cc2949a77/Sample_Test_Strategy.pdf)

### ▼ Make a test plan

The feature are

- add
- edit
- and delete

Scenario	Expected result	Latest result	Automated
Click a button to add a new user	The created user must appear and has two required fields for username and email. There is a create user button that becomes enabled once the fields are completed.	Pass	Yes

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0e050ce8-2316-492e-8f3e-1a532f86afdb/Sample\\_Test\\_Plan.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/0e050ce8-2316-492e-8f3e-1a532f86afdb/Sample_Test_Plan.pdf)

### ▼ Write acceptance criteria

- Conditions that a software product must satisfy to be accepted by a stakeholder
  - allowed developer to know what to implement code for
  - A business analyst what scope the story covers
  - And a QA which scenario to test
- HOW?
  - Given
    - This is a precondition or beginning state
  - When
    - **This describes the input or action of the scenario.**

- Then
  - This describes the expected outcome of the scenario.

▼ Example

- User story:
  - add a sold out item to a cart.

- Given I am viewing an item
- When I press the **Add to cart** button
- Then the item is added to the cart



- Given I am viewing an item that is sold out
- When I press the **Add to cart** button
- Then the item is not added to the cart
- **And** I see a message that the item is out of stock

▪

▼ Identify when testing is complete

- At some points of the SDLC, **it can be unclear to know when you have done enough** to move onto the next phase.
  - I am finish when
    - Test
      - Definition of done
    - Automate

# Manual and Automated Tests

789 aktive Nutzer

```
test "searching for one way flight" do
  fill_in "Origin", with: "SFO"
  fill_in "Destination", with: "LAX"
  fill_in "Depart Date", with: "2019-01-10"
  click_on "Search flights"
  assert_text "10 flights found"
end
```

- 
- Sign off AC
  - Sign off of acceptance criteria of a product

## Types of testing QA Focuses On

### ▼ Box testing

- Model testing as a box
- The whole idea is about looking at a box, or whatever application is under test from multiple angles to provide thorough test coverage. from multiple angles to provide thorough test coverage.
  - The black box
    - It is not possible to see inside of it
    - Manual
    - UI automation
  - The gray box testing
    - the box is transparent
      - Integration testing
      - trigger some action in the UI
      - Gray box testing examines the interaction between the outside and inside of a box.
  - The white box testing
    - Here the box is completely transparent and focuses on the internals of the application **and what is happening at the code or system level.**
      - Unit
      - System

### ▼ Example



- White
  - The quantity can be increased
  - The quantity can be decreased successfully

- The calculation of items is correct

▪

```
function increase(item)
{
    return item += 1;
}

function testIncreaseQuantityByOne(item)
{
    increase(item);
    assertEquals(item.quantity, 1);
}
```

- Gray

- Add an item to cart and confirm results from server

▪

```
function testItemAddedSuccessfully(item)
{
    addItemToCart(cart, item);
    assertEquals(cart.response, 200);
}
```

- black

- Add an item to a cart and confirm results from server

▪

```
function testAddItemToCart()
{
    $(".item").click();
    $(".cart").navigate();
    assertEquals(.quantity,1);
}
```

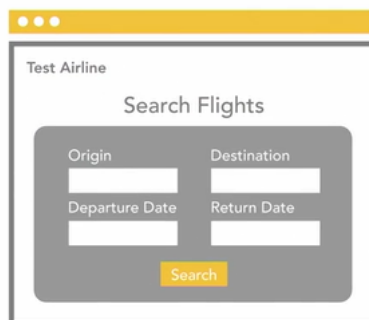
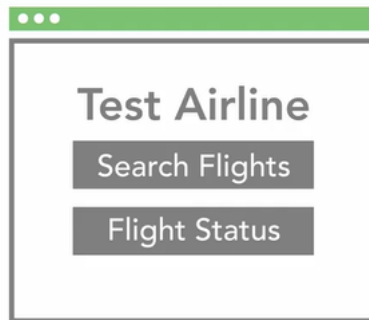
#### ▼ Manual testing

- is like the black box
- Before performing manual testing know what scenarios
  - Identify test scenarios before testing.



- Identify both typical and nontypical use cases.

▼ Example



- Happy path Scenarios
  - Search for one-way flight
  - Search for round-trip flight
- Sad Path Scenarios
  - Search
    - for same origin and destination
    - for invalid route
    - for route not in operation
  -

Scenario	Expected Result
Search for one-way flight	Flights displayed
Search for round-trip flight	Flights displayed
Search for same origin and destination	No flights displayed and error message "no flights available"
Search for invalid route	No flights displayed and error message "no flights available"
Search for route not in operation	No flights displayed and error message "no flights available"

Scenario	Origin	Destination	Departure Date	Return Date
Search for one-way flight	SFO	LAX	1/10	
Search for round-trip flight	SFO	LAX	1/10	1/15
Search for same origin and destination	SFO	SFO	1/10	
Search for invalid route	SFO	OAK	1/10	1/13
Search for route not in operation	SFO	SJC	1/10	1/18

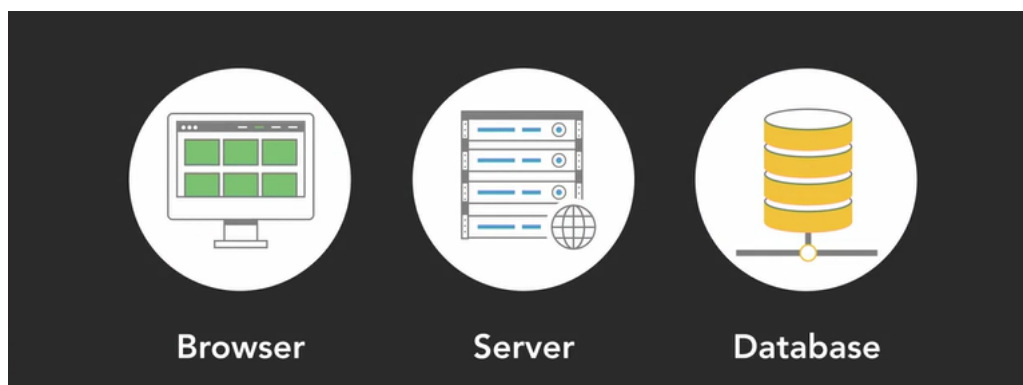
#### ▼ UI automation testing

- is like black box
- Benefit of UI Automation scenarios can be executed repeatedly and catch regressions introduced in the application
- It can run on all platforms.
- Like the example above

```
test "searching for one-way flight" do
  fill_in "Origin", with: "SFO"
  fill_in "Destination", with: "LAX"
  fill_in "Departure date", with: "2019-01-10"
  click_on "Search flights"
  assert_test "10 flights found"
```

#### ▼ Integration testing

- focuses on the interaction between application components.
- like gray box
- This level of testing covers similar scenarios as we've seen with manual and UI automation testing, but doesn't look at what's happening **at the UI level as a result.**
  - This type of integration tests the interaction between the application and the server to confirm that the right information is sent and received.



- If the request returns the wrong status code, or other incorrect flight data, we know that there is a problem **with the flight search feature**.

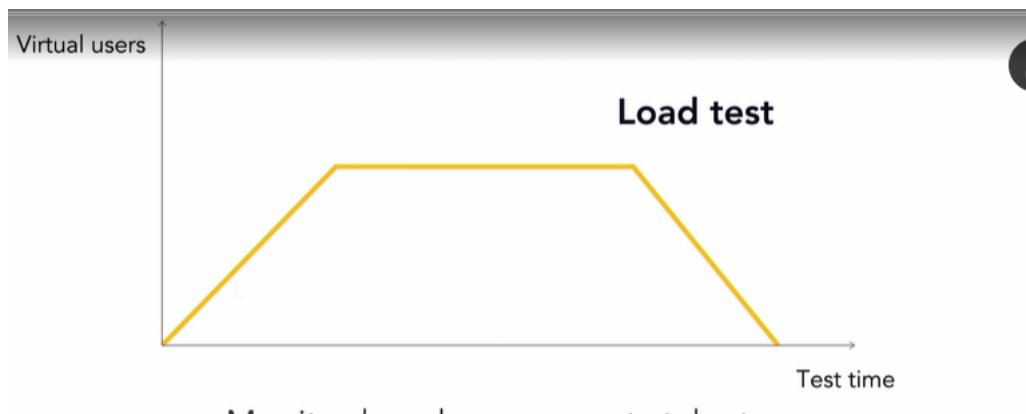
Expected	Actual
200	403

Could indicate an issue with feature

#### ▼ performance testing

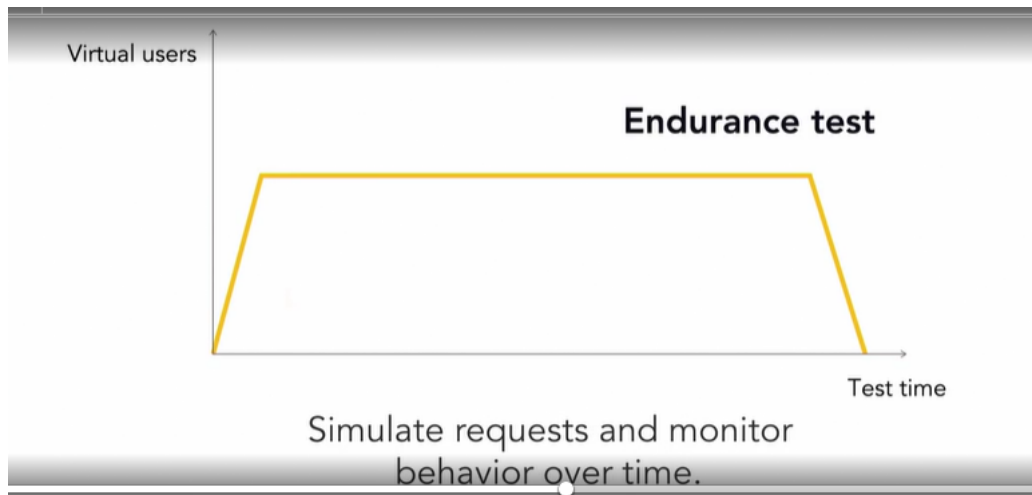
- is done to benchmarks how a system performs under load
- it is a type black box testing
- Common Performance Problems
  - Long load time
  - Poor scalability
  - Bottlenecking
- 3 types
  - load testing
    - checks the application's ability to perform under user loads

#### ▼ pic

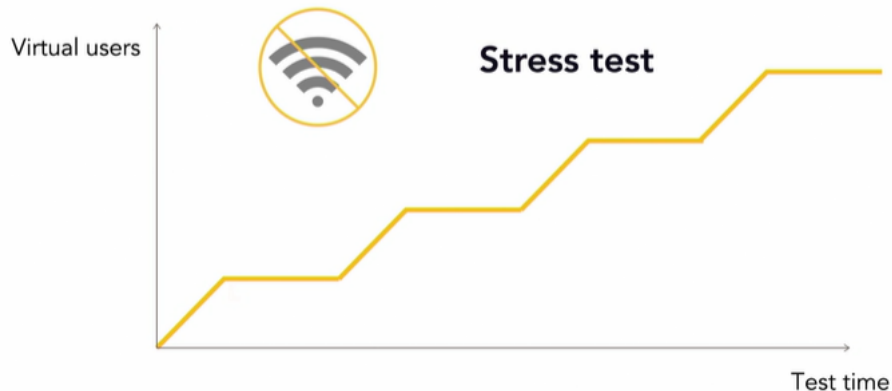


- Endurance testing
  - checks how an application handles load over a long period of time.
  - checks if there are system problem or not like memory issues

#### ▼ pic



- Stress testing
  - involves testing an application under extreme workloads. It is used for testing scalability
  - The goal of stress testing is to measure software stability.
    - When does it fail?
    - How does it recover?



- ▼ Security testing
  - exposes problems in the application that can cause unexpected behavior or crashes.
    - ▼ like
      - loss of customer data and trust
      - Decline in revenue
      - website downtime
  - SQL Injection
    - SQL injection is used to insert database statements into text fields and expose application information.
  - Denial of Service (DoS)
    - Denial of service (DoS) is an attack where hackers try to take down an application's servers or network
    - Vulnerabilities in dependencies can cause massive security problems.
    -

## Bug Reporting

### ▼ Identify bugs

- Bugs are inevitable. They occur when the system does not work as designed or specified, and result in incorrect or unexpected behavior in an application.

### ▼ Example

#### Examples of bugs can be when a vending machine

gives the wrong item. Perhaps I selected A5 but the item from B5 was dispensed instead. Or even worse, nothing dispenses at all. Another example of a bug can be when an ecommerce website does not calculate the right sales tax or total.




### ▼ Report bugs

- Bug Reporting System
  - Assign
  - Close
  - Report
  - triage
  - View
    - Bug reporting systems are all different.
      - Jira
        - Describe , Assign ,, priority , status ,owner.
      - GitHub
        - Description wit markdown
        - Label bugs

### ▼ Bug Report Example

## Bug Report

<b>Name:</b>	Example
<b>Description:</b>	Items are not added to the cart after pressing the <b>Add to cart</b> button.
<b>Steps to reproduce:</b>	Select the first item and choose <b>Add to cart</b> .
<b>Expected result:</b>	When I go to the cart, I can see the item I selected has been added.
<b>Actual result:</b>	When I go to the cart, there are no items.
<b>Screenshots:</b>	
<b>Browser/version:</b>	Chrome 71
<b>Logs:</b>	
<b>Tag:</b>	Todd
<b>Priority:</b>	High
<b>Status:</b>	Ready for development

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4ee4688a-98ef-43b8-8fad-9a2785debf7c/Bug\\_Report\\_Template.pdf](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4ee4688a-98ef-43b8-8fad-9a2785debf7c/Bug_Report_Template.pdf)

### ▼ Triage bugs

- Severity
  - How impactful the bug is to the business
- Priority
  - How fast the bug should be fixed



- - 4 - Low severity and low priority
    - Minimal customers
    - Does not affect major workflows
  - 3 - Low Severity and High Priority
    - Not causing a lot of disruption to the site
    - Visible , not preventing users from completing any workflow.
  - 2 - High Severity and Low Priority
    - Major problems only under some circumstances
  - 1 - High Severity and High Priority
    - Major Problems for many customers
    - Service disruption
    - Unusable application in some work flows

### ▼ Communicate bugs to the team

- Share top priority bugs
- Which bugs are up next
- Communicate Status of Bugs



- Have a project board
- Capture analytics about bugs
- Monitor where bugs come from

### ▼ Getting bugs fixed

- Your team can also set long term goals around bugs such as reducing the amount of open bugs by 15% in a quarter, or reducing the amount of open bugs to no more than 10. By setting these goals you will create a shared team investment and help squash bugs.

### ▼ Have bug bashes

- It's important for your team to dedicate time to both find and fix bugs. To find bugs, it can be great to have mob testing sessions around particular features in the application. Taking one hour, have a group of teammates get together to test.

- I recommend having bug bashes two to four times a year. They can be great for your application's health, and fun for your team.