Material starts with Cucumber/behavioral tests.

Remember that the topics will mainly be covered at a high level.

**1) Why do we write user stories, how do we write them, and what are they good for?  How do we know if a story is good (remember SMART!) Specific , Measurable, achievable, relatable, time box**

* **User stories** are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template
* **It’s usually be written in index cards or sticky notes**
* **Benefits of the user stories that they can have a lot or a little bet on details to help the developer to exceed the customer need.**

2) **You can also use Lo-Fi UI diagrams and storyboards.  What are they and what are they good for?**

* They are a kind of prototype of making user stories, they show very basic layout
* The benefits:
* It doesn’t require any technical skills
* easy to get user feedback on the design

3**) BDD/TDD: What is the process of using these together? What do these terms mean? Why are they useful?**

BDD: Behavior-driven design: develop user stories to describe features

* BDD shows you how to test
* BDD benefit: eliminate issues that TDD might cause

TDD: Test-driven design: is a process for when you write and run your tests

* TDD is writing unit and functional tests before the code via cucumber and Rspec
* TDD benefit: reduces the number of bugs in your tests, makes it possible to have a very high test-coverage
* TDD could be difficult because of the fact that you have to write your tests before writing the code

a**) This is a big area- BDD is not Cucumber. TDD is not Rspec. What is BDD? What is TDD? How can they be used together? They don’t have to be used together. I’ll ask about both separately and combinations of processes. Remember Cucumber and Rspec are just tools toward a purpose- they happen to be helpful here.**

**4) You've also learned testing tools (Rspec, Cucumber, Selenium, etc…)- what do they do and what are they good for?**

* **RSpec** is a testing tool for Ruby, It is the most frequently used testing library for Ruby in production applications.
* Rspec resets all mocks & stubs after each example “ keep tests Independent”.
* Rspec tests a class or part of the app in isolation.
* Rspec provide certain feature that is helpful to people who write lots of test
* **Cucumber** is software for testing other software.
* It runs automated acceptance tests
* It betters for non-technical people.
* Cucumber tests helps coders think like consumers.
* **Selenium**-**cucumber** is a (BDD) approach to write automation test script to test Desktop Web .
* It enables you to write and execute automated acceptance, unit tests

**5) What are the properties of a good test (remember FIRST!)—How do we write tests and why? KNOW THE ACRONYM and be able to give examples.**

* First: Fast, Independent, repeatable, self-checking, and Timely.
* We could do testing by unit testing or Rspec
* It’s important to catch errors

**6) What levels of testing are there and when should we use them?**

The Four Levels of Software Testing

* **Unit Test:** to refer to a function, individual program or even a procedure.
* **Integration Testing:** to combine all of the units within a program and test them as a group, and that allow you to find interface defects between the modules/functions.
* **System Testing:** is testing the complete application as a whole to see if the system has complied with all the outlined requirements and check if it meets quality Standards.
* **Acceptance Testing:** to determine whether the system is ready for release

7) Seams!- There are several types of seams and ways to isolate your test cases.  There are expectations that can be set, you can make fake objects, you can make fake methods, and you can make assertions.  What are these and how do you write these in your environment?

* Seams : A place where you can change your app behavior without editing the code
* Useful for testing
* Seams uses Ruby to open classes to create a seam for isolating

8) Mocks are a pain in the butt- why use them or why not? Review <http://martinfowler.com/articles/mocksArentStubs.html#TheDifferenceBetweenMocksAndStubs>

Four kind of double

* Dummy objects are passed around but never actually used. Usually they are just used to fill parameter lists.
* Fake objects actually have working implementations, but they are not suitable because it uses shortcut
* Stubs provide canned answers to calls made during the test, usually not responding at all to anything outside what's programmed in for the test. Stubs may also record information about calls, such as an email gateway stub that remembers the messages it 'sent', or maybe only how many messages it 'sent'.
* Mocks are objects pre-programmed with expectations which form a specification of the calls they are expected to receive.

9) Test coverage can also be measured in a number of ways- how are they used and what do these metrics mean?

Simply code coverage is a way to make sure that your tests are actually testing your code, having 100% doesn’t mean that your code is fully tested, it means that every line of code is tested.

Way to do test coverage:

* By feature
* By instrumentation
* By structure
* By scenario

11) When working in a Software Development Lifecycle (SDLC), think about what changes. When do some techniques work better than others?

* The Software Development Life Cycle is a process that ensures good software is built.

**12) Metrics- We’ve talked about coverage, mutation score, and touched on “bad code.” How can we define these?**

Metrics code coverage is A number of different metrics are used to determine how well exercised the code is

a) In terms of coverage, what levels of coverage do we have and what does it mean?

* Statement coverage
* Documentation coverage
* Branch coverage
* Path coverage
* Time coverage
* Condition coverage

b) In terms of mutation score, what is it and why do we care?

* Mutation score is used to design new software tests and evaluate the quality of existing software tests.
* Mutation testing involves modifying a program in small ways

c) In terms of “code prettiness,” what metrics do we have and what can we learn from them?

d) What’s the point of metrics?

You know how much of the project you got done

13) Be ready to talk about the ideas of testing. What levels do we cover? Why do we do so? How does each level of testing help? What happens with multiple levels of testing in terms of metric results and believability? How do we know we’re doing well in testing?

* We covered integration

14) Think about products now. We used to have COTS materials. Now we’re moving to SaaS applications and similar. How has this changed us as developers and business people in how we design software and go about writing it?