We have covered a lot of material since Test 1, and you all have gained a LOT of experience from your projects as well.  Here are the highlights. 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!**)

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

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

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?

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.

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

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/oracles.  What are these and how do you write these in your environment?

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

With that, how does mocking and stubbing change as you go from integration level testing to unit level testing?

9) Test coverage can also be measured in a number of ways- how are they used and what do these metrics mean? What are you analyzing with the tools you’re using? What do you use when you run a tool like SimpleCov and what level of coverage do you get from that? What’s that mean?

10) Legacy code- what is it, how can you deal with it, and what are characterization tests? (NOT ON THE TEST BUT IMPORTANT IN THE REST OF THE CLASS AND FOR JOB INTERVIEWS)

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

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

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

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?

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?

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?

**Very Shortened Version:**

* Why does software fail when we’re developers?
* BDD- what is it, how do we use it, and why? Good? Bad?
* TDD- what is it, how do we use it and why? Good? Bad?
* What is SMART? How can a user story be SMART? Why?
* When you design your project, how do you organize your code in terms of steps, features, scenarios, and user stories?
* How can you DRY out your code and tests?
* Scenarios can be written in different ways- how?
* Requirements come in different forms. Knowns and unknowns. Think on what that means given **explicit** requirements.
* Testing is changing in industry now- how?
* What is FIRST? What does each part of the acronym mean?
* What are the coverage metrics, and what are you using in your projects?
* What does the coverage tool in your project measure?
* What do these coverage metrics mean and how can they be used efficiently and effectively? Is it possible to test everything?
* What other testing metrics are out there? I introduced at least 4 more other than coverage
  + Remember that the point of these coverage metrics is to test the quality of your test suite- not your code
* What are the levels of testing in terms on complexity going from low to high? Hint: Unit 🡪Acceptance
* Red- Red- Green-Green-Reactor: What’s this pattern look like and what do you do? What breaks along the way? (Be able to draw the diagram.
* We have stubs/doubles. What are they and why do we use them in testing?
* If you call should\_receive/excepts, what are you doing? What is an expectation, and how do you use an expectation as a test? (Think about what seams are in general too- not great for all languages)
* What is the purpose of mocks and stubs/doubles?
* What about needing a real, more complex object? What are factories and fixtures?
* Why do we mostly want to stub out the internet or other tools?
* Look at your projects. Look at how they are set up. It is all coordinated into the MVC (remember what that is) structure.

I hope this short list helps you prepare for your test.