# CS 1632 Midterm 2 Study Guide

SPRING 2018

The second midterm is MONDAY 16 APR (for Mon/Wed classes) or TUESDAY 17 APR (for Tue/Thu classes).

Note that the second midterm is \_not\_ cumulative, except in the sense that the topics covered in the second half of the semester depend upon understanding of the more fundamental concepts taught in the beginning of the course.

## WEB TESTING

\* Be able to explain how you would test a web page with Katalon

- To test a web page, you can do the following:

assertTextPresent(asserts that the text exists by searching the entire page),

assertCookie, assertElementPresent(asserts the the element exists somewhere on the page, assertAlert

(asserts that an alert took place), assertEditable(asserts that an element is editable), assertEval(evalulate some js, then

assert the result)

\* You should know basic Selenese (Katalon/Selenium scripting language), e.g.

\* Opening a URL

\* Clicking on a link

\* Testing for text

\* You will not need to know about specifying target selection, but do need to know what target selection is and recognize it (i.e. if I show you a webpage, I will note specific target values for different elements - you will not need to read raw HTML)

## PROPERTY-BASED TESTING

\* Be able to write simple property-based tests with Rantly (BIG TOPIC)

\* Be able to name invariants given a function and sample input/output

\* Be able to show how invariants are broken

\* Understand what shrinking is and be able to shrink an input given an error

\* For what kinds of functions is property-based testing useful? For what kinds is it less useful?

## STATIC ANALYSIS

\* Understand static vs dynamic testing

- static analysis = code is not executed by the test

+ includes code coverage(simplecov), linters(rubocop), code metrics(cyclomatic complexity, number of lines), defect free

code.

- dynamtic analysis = code is executed by the test (everything we have done is dynamic testing)

\* Understand limitations of static testing

- Takes a long time to perform static testing, produces false positives and false negatives, does not support all programming

languages.

\* Know different kinds of static analysis, and tools and methods used (e.g. linters, bug finders, code coverage, code metrics, code reviews)

\* You do NOT need to know specific Rubocop/Reek errors, but should understand what they do and what they might catch or not

- Rubocop looks at each line (method has too many lines), reek looks at the overall picture (class has too many methods)

\* Understand code coverage and be able to calculate (simplecov)

\* Understand difference between statement and method coverage

- statement coverage looks at each individual line and checks if it has been tested. (percentage of statements that have been tested)

- method coverage looks at the method as a whole and checks if the test for that method returns correctly. (percentage of methods that have been tested)

## INTERACTING WITH STAKEHOLDERS

\* Be able to name some stakeholders and what is important to them (upper management, project management, testers, other developers)

- A stakeholder is a person or group who has direct interest in the completion and/or execution of the system

under development.

+ includes customers, users, project management,upper management, developers, testers, support staff, accessors

+ Software developers speak in terms of technology. Testers speak in terms of quality. Project Manager speak in

terms of deadlines. Upper management in terms of financials.

\* Be prepared for some "fake" interaction with various stakeholders

- ? what?

\* Be able to put together a red-yellow-green template report

- Red = The system or subsystem has some extremely serious issues which will almost certainly impact the timeline or

financials unless they are remediated.

- Yellow = There are some warning flags, but they can be dealt with. Outside help may be necessary.

- Green = Everything is A-OK or there are only minor issues.

## TESTING STRATEGY

\* Understand the testing pyramid

- 10% UI tests(tests which check the system end to end, 20% service tests, 70% unit test

\* Understand common anti-patterns (ice cream cone, cupcake)

- cupcake (lots of manual testing > unit testing)

- ice cream cone (70% UI tests, 20% service test, 10% unit tests)

\* Given a description of a program, be able to develop your own testing strategy

## SECURITY TESTING

\* The CIA/InfoSec Triad

- Confidentiality = No unauthorized users may be able to read data

- Integrity = No unauthorized users may be able to write data

- Availability = System is available for authorized users to read to and write to

\* Terminology: passive vs active, vulnerability, exploit

- passive = do not modify the system in any way (eavesdropping, monitoring, traffic analysis)

- active = modify the system in some (log in as a different user, fill up database with garbage, modify bank account info)

- vulnerability = identified weaknesses of the system

- exploit = technique used to compromise a system of its vulnerability

\* Terminology: interruption, interception, modification, fabrication

- interruption = attack on availability (pulling plug from network switch)

- interception = attack on confidentiality (eavesdropping, deals with reading data w/o permission)

- modification = attack on integrity (deals with modifying or deleting data, writing data w/o permission)

- fabrication = attack on integrity (deals with making up or inserting data, writing data w/o permission)

\* Be able to describe/test for common attacks: injection, broken authentication, XSS, insecure storage, buffer overruns, security misconfiguration, insecure object references, social engineering

- injection attacks = attacks on a database server that cause the database to fall apart (check on this)

- broken authentication = one user pretends to be another (breaks someone's password and hacks into account)

- XSS = (Cross Site Scripting) = gets a third party to execute code on their system (injection attack but with an intermediary)

- insecure storage = secure data is stored in an unsafe way (credit card numbers are being stored in a public directory)

- insecure object references = someone can access something by knowing where it is despite not having proper security credentials

- buffer overruns = trying to read or write more data than a buffer supposedly has access to (reading or writing past the end of the buffer)

- security misconfiguration = You have proper security, just not set up correctly (default passwords, insecure machine on secure network)

- social engineering = scammers psychologically fool others into giving away their sensitive information (you know them microsoft scammers on the web)

\* How does security testing differ from other kinds of testing?

- most other testing is dependent on testing the behavior of the system to see if the system returns the right values and behaves as it should, Security testing specifically checks on how cryptographically/algorithmically secure the system is instead of checking for the behavior of the actual system, so that hackers can not easily be able to hack through the software.