COSC 4345 Project Guidelines

Overview

The class will be divided into teams of four people. All teams will work on the same project, as defined by the instructor. The project will be delivered over the course of three major milestones during the semester. At the end of the semester, each team will make a project presentation and demonstrate their project to the class.

What Is Important?

- Each team member must be responsible for, and be the owner of, at least one building block of the system. Building blocks may be the database, the PHP code, one or more of the tests, the Java SUT controller, etc. It is ok for team members to work on each other's building blocks, but each team members are ultimately responsible for their building block.
- Regular code check-ins expected into BitBucket.com. Each student is required to have an account on BitBucket.
- The instructor must have read-only access to each team's project files in Bitbucket.
- The instructor will provide example code for the client, server and tests. (See Project Description below).
- Students will be graded on each of the three project milestones as well as the final presentation.
- Each Friday during the semester, the instructor will meet with the teams. The instructor will play the role of a customer to provide input on the project, as it evolves during the semester. There may be other meeting times assigned outside of class time. The instructor will make every attempt to find a time that works for the students and the instructor.
- Each team should use the Agile process.
- At the beginning of the semester, the instructor will provide a list of initial, minimum project requirements. As the semester progresses, new requirements and features will be added. This evolutionary process is to simulate what happens during an actual Agile project.
- Teams will be responsible for not only code, but also documenting requirements, use cases, system diagram, a running features list, project schedule and any other project artifacts described during the course.
- This course will use a Test Driven Development approach to software development. Requirements are turned into test cases, then the software is developed or improved to pass the new test cases. Test cases must be checked into BitBucket for each project feature.
- Each API must have one or more test cases.
- In addition to the client and server code, each team will be responsible for designing and developing a minimum of ten tests. (See the Project Scenario Document.)

Each team will work on an automated test system, comprising four building blocks:

1) One or more clients (systems under test) that periodically connect to a server to determine which test to run. The client then executes the test and logs the result (e.g., pass or fail) back to the server. The client software must be written in Java.
2) Tests will execute on the client. Tests must be written in Python. There must be a minimum of ten tests. (See the Project Scenario Document)
3) A server with a MySQL database. The database has a list of tests, each with a corresponding schedule of when to run. The database also has a historical record of which tests have run and pass/fail results. The server software will be written in PHP. The database will be MySQL.
4) A web front end will manage tests and display test results. A web front-end must be developed to 1) Allow a user to add new tests and systems under test, to the database, and 2) Display test results, a list of tests and a list of test systems.

Students should assume the system under test has a JRE, network connectivity and a Python interpreter.

Recliam Hosting

- The university has provided a web hosting system for all six teams using Reclaim Hosting. All teams must use Reclaim Hosting.
- To create an account: https://create.stedwards.edu/
- For help regarding MySQL and Reclaim Hosting: http://jbryan2.create.stedwards.edu/mysqlHowTo.html

Development Expectations

Test Driven Development

Each team will use a test driven development approach to building the project. Using this approach, project requirements/feature requests are turned into very specific test cases and then the software is constructed to pass the test cases. The following sequence must be followed:

1) Feature requests are documented and prioritized

2) Test case(s) are developed

3) Write the code

4) Run the tests

5) Refactor the code

Teams will be penalized if the above steps not followed

BitBucket

All teams are expected to use BitBucket. Teams are required to give the instructor read-access to the team's code.

Running Features List

Each team will keep a running features list during the semester. As the project evolves (e.g., version 1.0, 2.0, 3.0) the feature list will grow. The running features list is a way to keep track of the features that are implemented and are to be implemented. Each Friday, the current features list should be submitted to Canvas by every student, and will be counted as an assignment grade. Every team has their own features list and each student on a team should turn in the exact same features list.

Each feature should be associated with a feature name, brief description, priority and an implementation date. (Features that have not yet been implemented do not have an implementation date.)

Priorities:

P0 – Must have feature

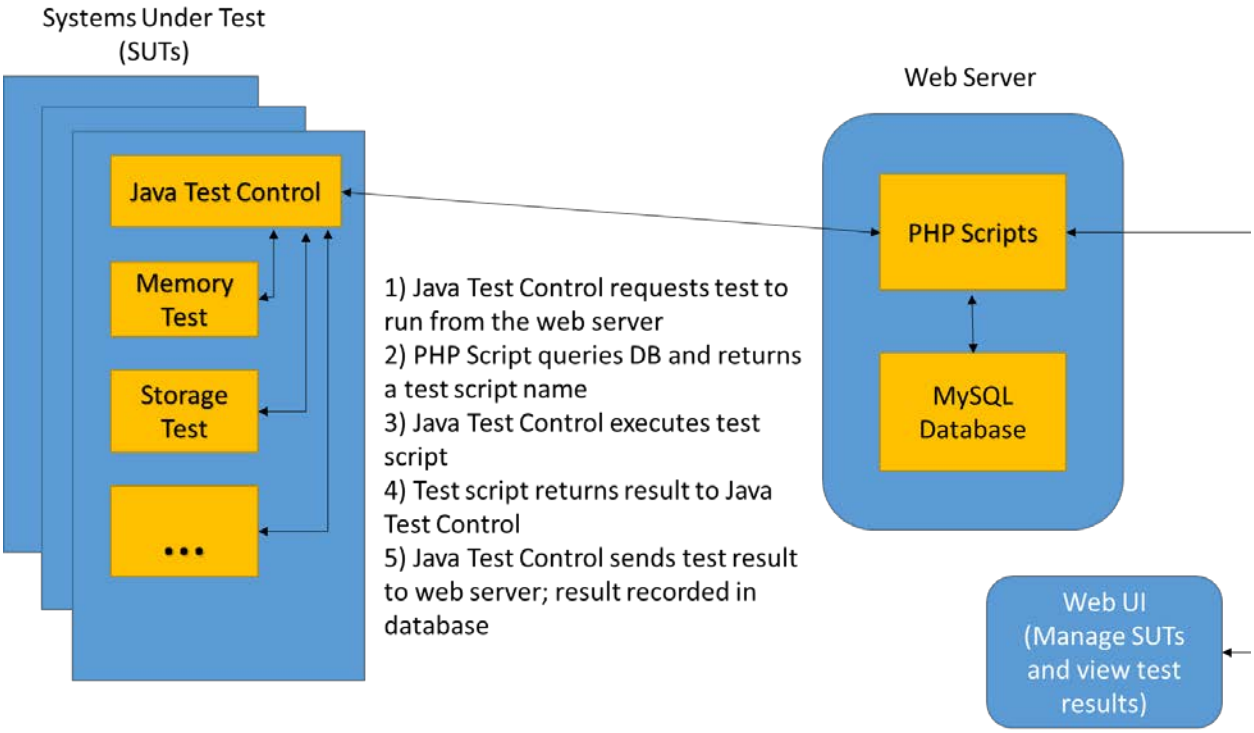
P1 – Should have feature

P3 – Nice To Have Feature

Team Member Responsibilities

1. Responsible for working together and with the instructor to develop the features list
2. Responsible for checking in code (BitBucket) on a regular basis
3. Each team member is the owner for one subsystem of the project. In other words, when the instructor has questions about a specific subsystem, he/she will go to the student responsible for that subsystem. The owner of the subsystem does not have to be the only person that codes that subsystem, but is the person responsible.
4. Responsible for attending all team meetings.
5. Be professional

Infrastructure Big Picture



API Overview

There will be three sets of APIs. Your team will work with the instructor to design the APIs. Below is a starting set of features. The feature list below is intentionally limited. Your team is responsible for interviewing the instructor, as if the instructor is your customer. The purpose of these interviews is to further define the features and requirements based on the information you learn from your customer.

Your team is responsible for designing the APIs, their parameters and return values.

The challenge for your team is to design a system that is extensible for the company. You should anticipate an ever-expanding features list and number of test clients.

Initial Server Features (i.e., PHP web service)

1. Set the frequency with which a test should be executed
2. Enable a test client to request a test to run
3. Enable test client to post a test result (e.g., Pass, Fail) to the server
4. Persistent storage for storing tests, test results and test client information
5. A web UI for viewing clients, tests and test results

Initial Client Test Controller Features (i.e., Java controller)

1. Set the interval between times that a client requests a new test from the server
2. Client requests test to run from a server
3. Client sends test results to the server

Initial Test Features (i.e., Python tests)

1. Run a test and return a SUCCESS/FAIL result. Other test result meta data must also be returned. Your team should work with the instructor to determine the meta data.
2. Every test must implement a run() method. Work with the instructor to determine the run() parameters and return values, which should be consistent across all tests.
3. Tests cannot report results directly to the server; only to the

Initial Web UI Features

1. Add a test client. Test client information should include at least a unique system ID, location and operating system
2. Remove a test client
3. Add a test – Test information should include at least a test ID, test description, and Python script name to execute
4. Remove a test
5. Display a list of test systems
6. Display a list of tests
7. Display a history of tests executed on a test system. The history should include at minimum the date/time a test was run, the test name, Pass/Fail results

Test Infrastructure Requirements:

- Tests must be written in Python to run on Windows or Mac OS
- Tests should only interact with the Java client test controller. In other words, tests should never connect to the web server.
- Test results must be logged to a MySQL database. Question for your team To Answer: What is a test result? Is it more than SUCCESS or FAIL? What will the engineers and management (i.e., your customers) want to see?
- There must be one or more web pages to display test results for the managers and the engineers. Question for your team: What will the engineers want to see?
- There must be one or more web pages to display test system details
- There must be a PHP web service that assigns tests to test systems and records test results to the database
- Each test system must have a Java client that communicates to the server to determine which test to run, when to run it and log results. The Java client is also responsible for executing the Python tests and reporting results back to the server.

Basic Test Requirements

There should be one or more tests for each of the following subsystems. There should be a total of ten tests:

- **Memory** – An example of a test to exercise memory is a Python program that creates a list with one million items. Use exception handling to determine if there is an error.
- **Storage** - An example of a test that exercises the storage system is a Python program that creates a one megabyte file with random characters. Use exception handling to determine if there is an error.
- **Networking** – An example of a test that exercises networking is a Python program that connects to a website, downloads an HTML file and verifies the contents.
- **CPU** – An example of a CPU test is a program that generates one million random numbers
- Video – An example of a Python program that tests the video subsystem is program that uses the pyscreenshot module to create a PNG file screenshot. The program should verify the file is created and has a > 0 file size.
- **Math Operations** - The math operations tests determine if the CPU can perform math operations correctly. Several operations should be tested, including:
    - **Integer Math Test** – Measures how fast the CPU can perform integer operations. An integer is a whole number with no fractional part.
    - **Prime Number Test** – Measures how fast the CPU can search for prime numbers. A prime number is a number that can only be divided by itself and 1. For example, 1, 2, 3, 5, 7, 11, etc. This algorithm should use a loop.
    - **Floating Point Test** – Performs the same operations as the integer math tests, however with floating point numbers. A floating point number is a number with fractional parts (e.g., 10.12345).
- **Video –** An example of this test is a test that uses the pyscreenshot module to capture a screenshot and store it as a PNG or JPG file.