A Web Based System to Support Testing Multiple Program Modules

Test Framework

Requirements

David Howick, Miriam Farrington, Mudit Vats, Elona Vabishchevich, Jeffrey Alexovich

Table of Contents

[Preface 2](#_Toc46238092)

[1 Introduction 3](#_Toc46238093)

[1.1 System Overview 3](#_Toc46238094)

[1.2 System Users 3](#_Toc46238095)

[1.3 System Usability 5](#_Toc46238096)

[Glossary 6](#_Toc46238097)

[2 System Requirements 7](#_Toc46238098)

[3. User Requirements 8](#_Toc46238099)

[3.1 User Requirement 01 8](#_Toc46238100)

[3.2 User Requirement 02 8](#_Toc46238101)

[3.3 User Requirement 03 9](#_Toc46238102)

[4 Constraints 10](#_Toc46238103)

[4.1 Technical Constraints 10](#_Toc46238104)

[4.2 Operational Constraints 10](#_Toc46238105)

[4.3 Business Constraints 10](#_Toc46238106)

[5 System Models 11](#_Toc46238107)

[5.1 Use Case Models and/or Scenarios 11](#_Toc46238108)

[5.2 Class Diagram 11](#_Toc46238109)

# Preface

Developing large scale software consisting of multiple packages requires frequent testing. If the software has complex features, interactions, or other complex system interfaces (APIs), then we want to build it incrementally and test each increment. The development team first designs and implements a very basic core with a small number of packages, then adds features one-at-a-time by adding new packages, or adding a few lines of code to an existing package. Each time new functionality is added, the application is built and tested. That way, if additions break existing code, the developers know where to look, e.g., in the newly added few lines of code. This Test Framework application will allow the development team(s) to use this incremental approach more efficiently.

The Test Framework application will allow the development team(s) and other users of the system to define many small tests, each of which run with exception handling and results logging. The goal of the Test Framework application is to do that without proliferating code with many try-catch blocks, debug statements, assertions, and abundance of verbose logging statements.

The Test Framework application will provide test results via logging as well as in saved test results files. So the logging mechanism will provide for several levels of logging. One level is just for quick basic test results and information about the test, such as how long the test took to complete. Another level will be verbose messages that dive into the details of a particular test and why that particular test failed.

The Test Framework application will be easy to use as it can be accessed by the user’s web browser. The system itself is cloud hosted in multiple regions so performance is always superb.

# 1 Introduction

There is a desire for a web based, cloud hosted solution that can test multiple program modules or blocks of program code simultaneously. This system will also have multiple methods for accessing the test module (dynamic link library, XML file, JSON file, etc.) and for delivering test results to the Test Framework application via logging and test results files.

In addition, there is the need to be able to test with multiple languages (C++, C#, Java, Python) and to add additional program language support to the application as needed for expandability.

Further, scalability is a concern as there will be multiple teams (multiple developers, engineers, testers, - users of the system) that will use the system at the same time so the system needs to be scalable, to support massive parallel processing/testing and support multiple concurrent users.

Because of the scalability issue, massive parallel testing that can be going on at any given time, the concurrency of tests and concurrency of users, system performance is also a prime concern. The system shall be hosted on cloud platforms and in multiple regions to address both performance requirements as well as system availability and business continuity concerns.

# 1.1 System Overview

# 1.2 System Users

There are 8 types of users for the system. They are:

1. Software Developers. These users are the developers of the software. They create designs for new features, build and run tests for each new feature.
2. Programmer Analysts. These users are also developers of the software, but usually working from a design document with strict guidance as to what is produced and tested.
3. Software Architects. These users are the designers of the overall application software, the structure of the application code (modules), the classes, dynamic behavior, and help elicit and define requirements.
4. Software Engineers. These users are the technical leads. In some organizations they are considered the software developers and in others they are the architects of the system.
5. System Architects. These users are responsible for the architecture and structure of the entire system, including the software, the hardware it runs on, the infrastructure it uses, the processes it follows, and the other systems the solution interfaces with.
6. System Engineers. These users are the technical leads. In some organizations they are considered the architects of the system.
7. Test Engineers (or QA personnel). These users develop detailed test cases from the requirements specifications, run the tests, and document the results in the test cases.
8. IT Managers (Mostly line level managers). These users are fun to watch; to see if they can make the software work, let alone figure out how to actually test something ☺.

# 1.3 System Usability

The system will be used by a range of professional IT development staff. This is a system that the developers, architects, engineers, and others should be able to learn to use quickly, enable quick testing of program code, get results back and view logs or other test output. The system should have:

1. Graphic User Interface (GUI).
2. Web enabled front end.
3. Capability to run multiple tests simultaneously.
4. Capability to ensure that no one test can tie up system resources.
5. Ability to allow multiple users to use the system at the same time.
6. System is highly available, disaster recoverable, and located in multiple regions of a cloud platform that allow for excellent performance, local scalability, and reduction in network latency.

# Glossary

**MVP** – Minimum Viable Product. This is the minimally acceptable product the user can test with and utilize for test purposes.

**GUI** – Graphic User Interface. Web based or Windows based client front-end to allow the user ease of use of the application.

**Concurrency** – Has multiple meanings depending upon context. Can mean multiple users, can also mean multi-threaded.

**Scalability** – There are two types. Horizontal scaling is the adding of additional infrastructure (namely servers) to handle increased loads or parallel processing. Vertical scaling is adding more resources (more memory or more CPU) to existing infrastructure.

**API** – Application Programming Interface. Program code that allows for communication between one application to another via a defined set of protocol (rules).

**HA** – High Availability. Usually measured by “nines”, like four nines (99.99%) is the measure of uptime and available for user use, of the system. Sometimes measured as AEC (Availability Environment Classification) scheme codes. A system with an AEC value of 2 is considered highly available.

**DR** – Disaster Recovery. This is the failover/recovery method of the system. Recovery levels usually range from zero to five, but there are two important measurements or requirements for determining system recoverability. RTO (Recovery Time Objective) is the measurement of how long the system can be down before it must be online again and available for use. RPO (Recovery Point Objective) is the measurement of how much time elapses between snapshots, copies or other replication of data. In other words, how much data can you afford to lose? The smaller the numbers in either case, the higher the Recovery level has to be.

# 2 System Requirements

2.1 The system shall be implemented as a client-server system.

2.2 Client access to the system shall be provided through a standard web browser. The Firefox web browser shall be the standard browser that is supported. However, the system should support the three top popular web browsers, Microsoft Edge, Google Chrome, and Firefox.

2.3 The system interface shall be Web enabled GUI.

2.4 The system shall be hosted on a cloud platform to support ease of resource acquisition and hosting, automatic scaling of system resources, built-in network infrastructure, and managed services where needed.

2.5 The system shall be hosted on servers located in multiple geographic regions to support disaster recovery and business continuity requirements (see availability requirements section), and to support continuous high performance and low latency. The system shall also be implemented in multiple availability zones for added high availability (see availability requirements).

2.6 Only one region shall be taken down for maintenance or enhancement at a time. The others shall be left up, running, and available. As the updated region comes back online, the next region can be taken offline for maintenance or enhancement until all regions have been updated.

2.7 The system shall maintain all program code in scripts that can be deployed to the cloud platform and have backup copies of these scripts located in a separate region. The scripts shall be stored on a distributed file system with built-in redundancy.

2.8 The system shall have a development environment for use by the software engineering and development team(s).

2.9 The system shall have a test environment to allow other users to test changes before committing them to production. This environment shall be implemented in multiple zones and multiple regions to enable testing of HA/DR requirements rather than taking production down.

2.10 The system shall have a production environment that is used by multiple users implemented in multiple regions and multiple availability zones

# 3. Availability and Business Continuity Requirements

## 3.1 Availability Requirement 01

The system shall support 24/7 availability. Routine downtime in a particular region necessary for maintenance or enhancement to the system shall take place after 21:00 EST on Saturday and shall end before 23:00 EST on Sunday.

3.1.1 Any scheduled downtime which takes place outside of the designated hours shall be reported to users no less than 48 hours in advance. In the case of emergency system outage, the notice period shall be waived but users shall be informed as soon as possible of any unplanned system outages.

## 3.2 Availability Requirement 02

The system shall be able to quickly recover from outages due to unforeseen circumstances while minimizing downtime. The system shall support the ability to create and issue automated alerts when downtime is encountered for any of the reasons stated below.

3.2.1 In the event of an unplanned outage due to the loss of a particular region or availability zone, the system shall immediately fail over to another region or availability zone as determined by the cloud provider.

3.2.2 In the event of an unplanned outage due to the failure of an instance on which the system is hosted, the system shall immediately fail over to a backup instance. In the event a backup instance does not exist, the system shall have the ability to immediately spin up a new instance and fail over to it using automated deployment.

3.2.3. In the event of an unplanned outage in any or all regions due to a software error, the system shall support the ability to quickly identify and create a restore point from the last known working backup. This process shall take no more than 1 hour to complete from the time the software malfunction is identified.

## 3.3 Availability Requirement 03

The system shall support high availability by being quickly accessible to users attempting to access it from any geographic region.

3.3.1 The system homepage shall take no more than an average of five (5) seconds to load from the time the URL is input from a web browser in any geographic region. This average shall be taken from 10 consecutive attempts to access the homepage.

3.3.2 Navigation actions (paging, links, etc.) should take no more than an average of three (3) seconds to load from the time the action is triggered. This average shall be taken from 10 consecutive attempts to perform the action.

# 4. User Requirements

## 4.1 User Requirement 01

The software developer(s) need to be able to run individual unit tests of program code in a test framework / harness.

4.1.1 The test framework / harness shall be an application that can run on the Windows platform. The user will access from the client web browser. The system should run on Linux and MAC platforms as well.

4.1.2 The application shall not require changing and recompiling the program each time a test is run.

4.1.3 The application will have a Graphic User Interface that will allow:

4.1.3.1 The developer to choose which test(s) to run (some type of file dialog / list box).

4.1.3.2 The developer to build a list of all test(s) to run (container object on GUI).

4.1.3.3 Shall show all tests selected to run (some type of dialog / list box).

4.1.3.4 Shall show test progress and status on the GUI.

4.1.4 The application shall allow multiple tests to run at one time. The application will allow the tests to run asynchronously so that no one test will hold up the other tests by tying up resources and starving the other processes (threads).

4.1.5 The application will log all test results to an external file AND to the screen in some type of GUI container.

## 4.2 User Requirement 02

The Software / System Architect(s) need to be able to run individual tests as well as multiple tests at once. They need to be able to stress performance, ensure scalability, and diagnose system interface issues.

4.2.1 The system shall be a web based client application available on demand.

4.2.2 Shall allow for installations on multiple machines (redundancy, performance, latency).

4.2.3 If during the test, the application throws an exception, the system shall handle it.

4.2.4 If during the test, the test itself throws an exception, the system shall handle it.

4.2.5 The thread pool shall be managed to a distinct size and can vary based on user demand

4.2.5.1 Starting default minimum thread count = 5.

4.2.5.2 Starting default maximum thread count = 15 (this keeps the application from spawning too many threads).

4.2.6 The tests shall show pass/fail.

4.2.7 The log component shall show different levels of logging (INFO, DEBUG, ERROR)

4.2.7.1 INFO describes specific information for test pass/fail reporting.

4.2.7.2 DEBUG describes programmer/developer provided information to aid in debugging the test.

4.2.7.3 ERROR describes the most detailed debugging output for examination of software test failures.

4.2.8 The log shall have time and date stamp and duration of test.

## 4.3 User Requirement 03

Test Engineer(s) or QA personnel, need to be able to provide a test case where several tests are sent in quick succession to demonstrate the application executes tests concurrently.

4.3.1 The test case shall be several tests of varying duration that can run simultaneously.

4.3.2 Upon completion, the system shall post a ready status message and await the next test.

# 5 Constraints

## 5.1 Technical Constraints

5.1.1 The system shall be developed using the C++ programming language and the C++ Standard Template Library (STL).

5.1.2 The system shall be developed using a publicly available source code editor which supports the C++ language.

5.1.3 The system shall have at least 75% unit test coverage of the source code. Unit tests shall be developed using the CppUnit testing library.

## 5.2 Operational Constraints

5.2.1 The cloud-hosted system shall be developed initially to be hosted on Amazon Web Services (AWS) but should be capable of being migrated to alternative commercially available cloud providers such as Microsoft Azure and Google Cloud Platform (GCP).

5.2.2 The Firefox web browser shall be the default browser supported to access the system. However, the system should be accessible using alternative browsers such as Microsoft Edge, Firefox, and Google Chrome. Alternative browsers will not be actively supported.

5.2.3 System source code and data shall be stored in a fault-tolerant, distributed file system such as Amazon S3 or HDFS where it can be accessible and deployed to support disaster recovery and availability requirements.

5.2.3 System source code shall be deployed to cloud instances hosted in several regions and availability zones.

5.2.3. Access to the system shall be controlled using defined, cloud managed IAM roles, which will allow for configurable levels of access to and control over the system and its resources.

## 5.3 Business Constraints

5.3.1 Disaster recovery shall be cost-effective and managed through the fault tolerance and high availability features of the cloud-based system architecture.

5.3.2 User training shall take no more than 1 business day to complete, regardless of the user’s role.

5.3.3 Granting access to a new user of the system shall take no more than 1 business day to complete.

5.3.4 Modifying or removing a user’s access to the system shall take no more than 1 business day to complete.

# 5 System Models

## 5.1 Use Case Models and/or Scenarios

## 5.2 Class Diagram